1992

PTI

MATERIAŁY SZKOLENIOWE

Polskie
Towarzystwo
Informatyczne

Oddział
Górnośląski

PTI

Systemy relacyjnych baz danych: INFORMIX, INGRES, ORACLE, PROGRESS

Materiały trzeciej i czwartej
Górskiej Szkoły PTI
Szczyrk, 17 - 21 czerwca 1991
i 18 - 22 czerwca 1992

Katowice 1991 - 1992

Spis treści

Struktury danych 3

Co to jest IDMSX? 18

Charakterystyka systemu INFORMIX
z uwzgędnieniem architektury programowej
jego pakietów 27

Relacyjna baza danych INGRES 36

Architektura i narzędzia tworzenia systemów aplikacyjnych 36

Relacyjna baza danych ORACLE 50

Relacyjna baza danych 59

Wykłady zawarte w tym zeszycie zostały wygłoszone w trakcie Trzeciej Górskiej Szkoły PTI, która odbyła się w Szczyrku pod koniec czerwca 1991 roku. Blok poświęcony bazom danych cieszył się ogromnym powodzeniem, głównie dlatego, że bardzo wielu uczestników Szkoły stało przed koniecznością dokonanie wyboru systemu, który stanie się warsztatem ich pracy w najbliższych latach. Pojawiało się w trakcie wykładów pytanie o to, który z omawianych systemów jest najlepszy. Odpowiedź jest trudna, bowiem system bazy danych powinien odpowiadać zadaniom, dla których rozwiązania ma być użyty. Chcąc jednak udzielić więcej informacji pozwalających dokonać wyboru, postanowiliśmy wydać dostarczone na Szkołę materiały w postaci książki, zawierającej również tabele porównujące różne cechy omawianych systemów. Niestety, nasze możliwości organizacyjne okazały się nie wystarczające, ażeby pięciu autorów umieścić w jednym punkcie czasoprzestrzeni, na czas wystarczający dla dokonania porównań. Tak więc roczne opóźnienie w wydaniu niniejszego zbioru jest spowodowane naszą niewydolnością w organizowaniu spotkań, za którą wszystkich spragnionych wiedzy serdecznie przepraszamy.

Chcąc mimo wszystko udzielić częściowej odpowiedzi na dręczące pytania, postanowiliśmy w czasie Czwartej Górskiej Szkoły zorganizować mecz baz danych. Ideą meczu jest sforułowanie zadania, które cztery niezależne zespoły programistów będą musiały rozwiązać i zralizować przy użyciu czterech omawianych systemów. Mamy nadzieję, że takie praktyczne współzawodnictwo wykaże, w jakich obszarach zastosowań i które z użytych systemów okażą się najbardziej przydatne.

W tej sytuacji niniejszy materiał stał się "programem" dwóch kolejnych Szkół i przynajmniej połowa odbiorców otrzyma go w terminie. Tych, którzy musieli czekać, jeszcze raz serdecznie przepraszamy i pozostajemy w nadziei, że spotkamy się powtórnie na kolejnych edycjach szczyrkowskiej Szkoły.

Organizatorzy

Systemy relacyjnych baz danych:
INFORMIX, INGRES, ORACLE, PROGRESS

Górnośląski Oddział
Polskiego Towarzystwa Informatycznego
Katowice 1991-1992

Skład, DTP&Layout
WYDAWNICTWO
LUPUS
Warszawa

Korekta
Maria Omiecińska

Druk:
KOMPDRUK
Warszawa

Struktury danych

Zbigniew Benedykt

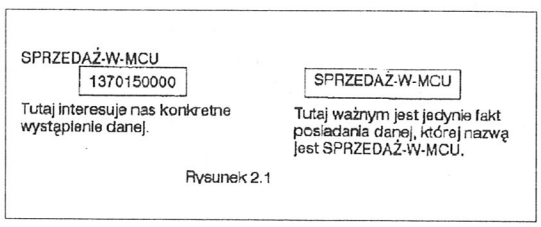
1. Struktury danych

Jednym z podstawowych zadań stojących przed informatykiem zajmującym się przetwarzaniem danych jest wyrażenie w systemie komputerowym związków zachodzących pomiędzy nimi. Związki te mogą być wyrażone w konstrukcjach struktur danych lub poprzez algorytmy realizowane w programach. Spróbujmy prześledzić, czym w tym zakresie dysponuje informatyk.

Dla naszych rozważań nie bardzo istotna jest definicja danej, ale dla porządku przypomnę, że daną nazywamy parę: nazwa, wartość. Następnym pojęciem, które godzi się tu przytoczyć, jest pojęcie typu danej. Przez typ danej najczęściej rozumie się zbiór wartości, jakie dana może przyjmować. Podstawową cegiełką danych jest dana elementarna. Jest ona dla informatyka tym, czym atom dla chemika, nie można jej rozłożyć na części, z nich tworzone są bardziej - czasami nawet bardzo - złożone struktury. Prosta konkatenacja danych jednego typu tworzy wektor lub jak wolą niektórzy - tablicę jednowymiarową. Konkatenacja tablic jednowymiarowych tego samego typu tworzy tablicę dwuwymiarową. Postępując w ten sposób możemy zbudować n-wymiarowe tablice.

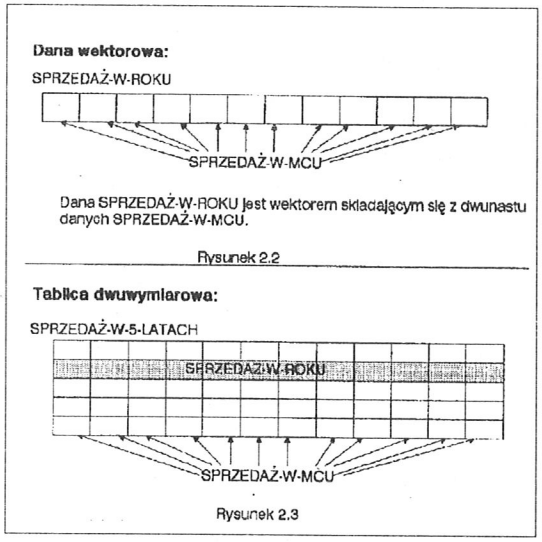
Wygodnym sposobem przedstawiania danych jest notacja graficzna.

Spróbujmy dane przedstawiać w postaci pudełek, w których można umieszczać wartości. Dane elementarne przedstawiać będziemy jako pojedyncze pudełka. Jeśli interesować nas będzie inkarnacja danej, nazwę umieścimy nad pudełkiem z lewej strony, zaś wartość wewnątrz pudełka; nazwę danej umieścimy wewnątrz pudełka, w przypadku gdy interesować się będziemy daną tylko z uwagi na jej istnienie. Dane złożone przedstawiać będziemy jako kolekcje połączonych danych podporządkowanych, nazwę danych umieszczać będziemy w lewym górnym rogu. Oto kilka przykładów:



Rysunek 2.1

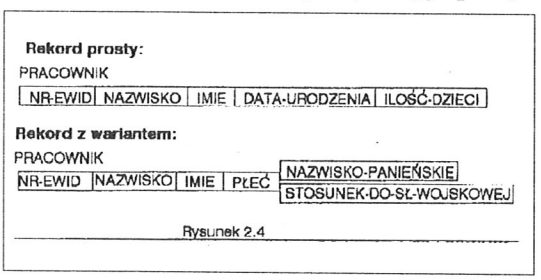
Wspólną cechą tablic jest możliwość szybkiego wyszukania elementów składowych poprzez wyliczenie położenia tych elementów w oparciu o znajomość indeksów. Tej własności nie posiada najczęściej używana w przetwarzaniu danych



Rysunek 2.2

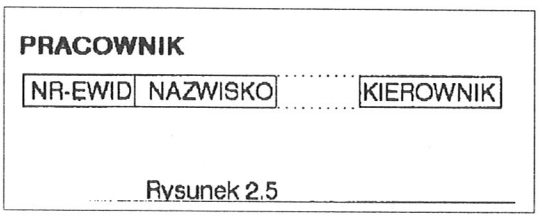
Rysunek 2.3

struktura, jaką jest rekord (zapis). Rekord tak jak i wektor jest konkatenacją danych składowych, z tą różnicą, że poszczególne dane składowe nie muszą być tego samego typu. Możemy nawet pójść dalej żądając , aby niektóre dane składowe mogły mieć dynamicznie zmieniany typ. Takie rekordy nazywane są rekordami z wariantami. Elementami rekordów mogą być dane elementarne, tablice, inne rekordy. W praktyce rekordy stanowią niezależne jednostki danych; jeśli spójnemu podciągowi elementów rekordu nadamy nazwę, to daną taką (mimo że spełnia podaną wyżej definicję rekordu) nazywać będziemy daną grupową np: datę urodzenia możemy traktować jak złożenie danych: dzień, miesiąc i rok urodzenia, wtedy data urodzenia staje się daną grupową.



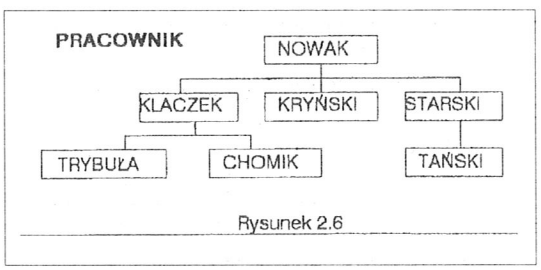
Rysunek 2.4

Przyjrzyjmy się następnej sytuacji (rys. 2.5):



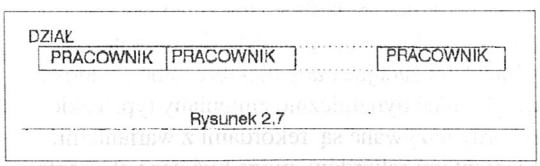
Rysunek 2.5

rekord PRACOWNIK zawiera daną KIEROWNIK opisującą przełożonego danego pracownika. Kierownik jest również pracownikiem, zatem dana KIEROWNIK powinna być tego samego typu co dana PRACOWNIK. W ten oto sposób dobrnęliśmy do struktur rekurencyjnych stwarzających wręcz nieograniczone możliwości twórcze dla informatyka. Struktura może być potencjalnie nieskończona, lecz każda jej inkarnacja musi być skończona, więc musi istnieć pracownik, który nie ma kierownika oraz żaden pracownik nie może być kierownikiem Jeśli zgodzimy się na wyżej postawione warunki, to łatwo możemy zauważyć, że zbudowaliśmy drzewo. Oto jak wygląda struktura PRACOWNIK przedstawiona w konwencji grafów (rys. 2.6):



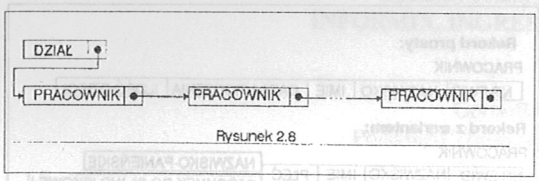
Rysunek 2.6

Stosując powyższą metodę możemy zapisać dowolny graf skierowany. Najprostszym z nich jest lista jednokierunkowa, którą można uzyskać stosując operator konkatenacji do rekordów jednego typu (zadziwiająca moc tego operatora).



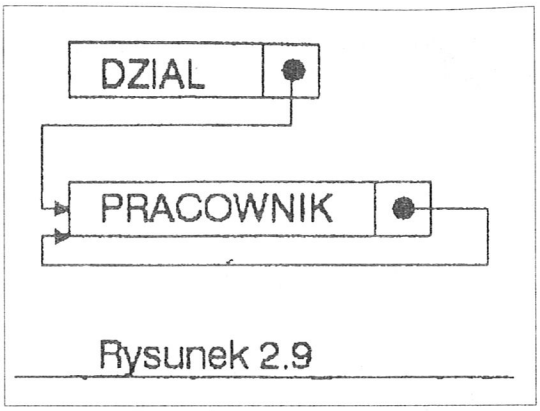
Rysunek 2.7

Struktury listowe przedstawia się zazwyczaj w postaci, która uwypukla istnienie oddzielnych rekordów jako jednostek semantycznych, ponadto odwzorowuje rzeczywistą implementację ich w pamięci komputerowej. Oto jak wygląda struktura DZIAŁ przedstawiona w konwencji listowej. Wydzielony prostokącik z kropką w środku reprezentuje odsyłacz do następnego



Rysunek 2.8

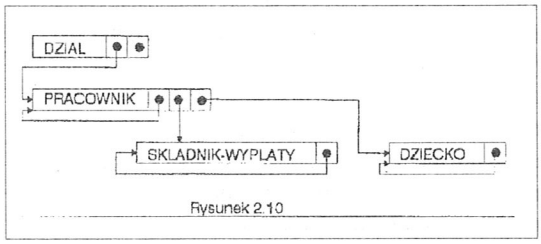
rekordu.Takie przedstawienie pozwala na pokazanie konkretnego wcielenia struktury. Jeśli wystarczy nam zobrazowanie jedynie zależności pomiędzy typami, możemy powyższą listę przedstawić w bardziej skondensowanej postaci. Demonstruje to rysunek 2.9. Tutaj nazwy w klatkach są nazwami typów rekordów, a nie jak wyżej nazwami rekordu. Ten dualizm nie powinien jednak prowadzić do nieporozumień. W ten sposób uzyskujemy możliwość prezentacji nawet bardzo wyrafinowanych struktur. Rysunek 2.10 pokazuje



Rysunek 2.9

przykład bardziej (chociaż jeszcze nie bardzo) złożonej struktury.

Podstawowy problem to: jak taką strukturę zaimplementować?. Rozwiązań jest wiele. Pierwsze z nich, to zastąpienie struktur trudnych do implementacji strukturami, które można stosunkowo łatwo zaimplementować. Tak się szczęśliwie składa , że każdą strukturę daje się przedstawić w postaci zbioru tablic (być może potencjalnie nies-



Rysunek 2.10

kończonych). Związki pomiędzy tablicami można wyrazić poprzez dane, których wartości stanowią o powiązaniu wierszy jednej tablicy z wierszami innej. Czasami istnieje jednak możliwość bezpośredniego zaimplementowania skomplikowanej struktury listowej. Jeśli inkarnację jej możemy zmieścić w pamięci operacyjnej komputera, problemu nie ma, użyjemy jednego z języków programowania, który umożliwia posługiwanie się strukturami listowymi (np. List, Prolog, Pascal, C). Jednak zazwyczaj w rzeczywistych systemach przetwarzania danych ich liczba jest tak duża, że nawet marzyć nie możemy o zmieszczeniu się w pamięci operacyjnej największego nawet komputera. Musimy posłużyć się urządzeniami pozwalającymi przechowywać dane masowe. Powoduje to powstawanie nowych, dodatkowych problemów zależnych od typów urządzeń. Kiedyś, kiedy podstawowym nośnikiem do zapisu danych była taśma magnetyczna, wszystkie dane musiały być zorganizowane w pliki sekwencyjne (wróciliśmy do transformacji w struktury podobne do tablic). Algorytmy kojarzenia danych uwzględniały związki pomiędzy ich poszczególnymi jednostkami semantycznymi. Zastosowanie nośników pamięci o bezpośrednim dostępie daje większą swobodę w organizowaniu zapisu danych. W pliku sekwencyjnym, aby uzyskać konkretny wiersz, musimy przejść kolejno poprzez wiersze poprzedzające. Tutaj możemy szybciej odnaleźć

dany wiersz na podstawie znajomości fizycznego adresu bloku, w którym się znajduje. Jeśli rekordy mają stałą długość, wiedzę taką posiadamy. Można - w końcu - podzielić całą dostępną nam pamięć na adresowalne jednostki i związać położenie każdego wystąpienia rekordu z taką jednostką, związki między rekordami wyrażają odsyłacze do ich konkretyzacji. Jest to metoda na bezpośrednią implementację skomplikowanych struktur listowych.

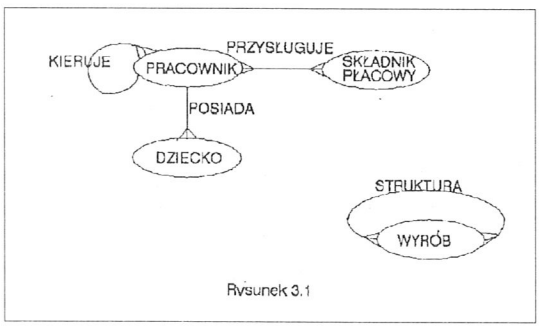
2. Specyfikacja danych systemu

W poprzednim rozdziale zajmowaliśmy się konstruowaniem różnych struktur danych, nie martwiąc się o to czy takie skomplikowane struktury są potrzebne w praktyce codziennej. Wprawdzie przytaczane przykłady pochodzą ze sfery administracyjnej, ale jak jest naprawdę? Przecież analityk systemu nie wykorzystuje struktur danych dlatego tylko, że istnieją takie teoretyczne możliwości, lecz modeluje rzeczywistość. Dopasowanie się do teoretycznie opracowanych narzędzi wyznacza metodę tworzenia systemów. Opracowanie teorii struktur danych pozwala wyodrębnić te funkcje systemu, które są wspólne dla całych ich klas. Nietrudno zauważyć, że zbudowanie oprogramowania realizującego ogólne funkcje administrowania danymi jest bardzo kosztowne. Rodzą się pytania:

- czy warto tworzyć takie oprogramowanie?
- jaką klasę struktur powinno obsługiwać?
- jaki będzie koszt jego użytkowania (wiadomo, że za ogólność płaci się efektywnością)?
- jakie są granice jego zastosowań?

Aby odpowiedzieć sobie na te pytania, spróbujmy po krótce przyjrzeć się pracy analityka systemu. Jednym z najważniejszych zadań jest wyspecyfikowanie danych systemu, wszak one tworzą dziedzinę wszystkich jego funkcji.

Jedną z najczęściej stosowanych technik do specyfikowania statycznych związków danych są diagramy związków encji. Istnieją różne wersje graficzne tych diagramów, również pojęcie encji w różnych metodach ma nieco różne znaczenie. Roztrząsanie tych problemów nie jest zadaniem tego opracowania. Należą one do zagadnień metodyk budowania systemów. Przypomnijmy, że encją nazywać będziemy rzeczy istniejące w świecie rzeczywistym lub obiekty powstałe w wyniku tworzenia klas takich przedmiotów, dające się zidentyfikować i wchodzące w zakres zainteresowań budowanego systemu np. człowiek, pracownik, partia materiału, asortyment (klasa wszystkich detali, którym przypisuje się jedną wartość indeksu materiałowego). Pomiędzy encjami zachodzą różne związki. Wiele z tych związków jest nieistotnych dla przyszłego systemu, ale niektóre są bardzo ważne i musimy je wyspecjalizować, np. pracownik może mieć dzieci, fakt posiadania dziecka jest ważny w systemie płacowym, ponieważ pracownikowi z tego tytułu przysługuje dodatek rodzinny, całkowicie nieistotny jest zaś stosunek uczuciowy pracownika do dzieci. W systemie płacowym takiej relacji nie będziemy specyfikować (co nie przeszkadza, aby związek uczuciowy był jednym z najważniejszych w systemie budowanym dla psychologa). Zadaniem diagramów związków encji jest prezentacja wszystkich encji systemu oraz pokazanie powiązań pomiędzy nimi. Jak już wspomniałem, istnieje wiele wariantów graficznych ich rysowania, prezentowany tutaj pochodzi z metody SSADM, która jest oficjalnym standardem w Wielkiej Brytanii. Encje reprezentowane są przez elipsy, związki między nimi przedstawia się w postaci linii, które mogą być zakończone "wronimi łapkami. Taka "łapka" oznacza, że po stronie gdzie występuje, może być połączonych wiele inkarnacji encji danego typu (diagramy związków encji przedstawiają związki pomiędzy typami encji!). Oto przykłady:



Rysunek 3.1

Co możemy odczytać z tych diagramów? Z pierwszego, że interesują nas takie byty jak pracownik, dziecko, składnik płacowy; ponadto, że pracownik może mieć wiele dzieci, wielu podwładnych, mogą mu przysługiwać należności z różnych tytułów płacowych oraz wielu pracowników może pobierać pieniądze z tego samego tytułu. Drugi diagram skupia nasze zainteresowanie na strukturze wyrobów. Konkretny wyrób może być kolekcją innych, może również być elementem składowym wielu wyrobów złożonych.

Myślę, że powyższe diagramy są podobne do struktur listowych. Odpowiednia transformacja jest sprawą czysto techniczną i bardzo łatwą. Tym samym dochodzimy do takich samych struktur z dwóch punktów startowych: pierwszy, teoretyczna konstrukcja - pozwala mieć nadzieję na zbudowanie formalnie poprawnych narzędzi w oparciu o poprawną teorię - drugi,łatwe i pełne odwzorowanie świata rzeczywistego w modele teoretyczne a zatem można skorzystać z raz zbudowanych narzędzi do realizacji funkcji zarządzania danymi. Narzędzia takie rzeczywiście powstały, noszą nazwę systemów baz danych.

3. Modele baz danych

Jak - mam nadzieję - wykazałem w poprzednich rozdziałach, istnieje możliwość jak i potrzeba budowania pakietów oprogramowania wspomagających administrowanie danymi. Należy w pierwszym rzędzie ustalić jakiego rodzaju struktury ma takie oprogramowanie obsługiwać i jaki ma być zakres tych usług.

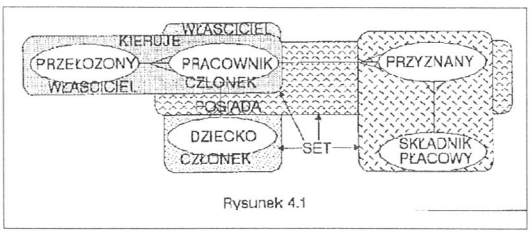
Pierwszym i chyba najważniejszym zadaniem jest utrzymywanie wewnętrznej niesprzeczności danych. Wiemy już, że dane wiążą się ze sobą wyrażając związki zachodzące w świecie opisywanym przez system informatyczny. Ponieważ nie są to niepodzielne jednostki, może dojść do utraty informacji wyrażonej przez powiązanie. System zarządzania danymi powinien je chronić przed taką ewentualnością. Jeśli jednak już do takiego zdarzenia dojdzie, winien wspomagać proces naprawy uszkodzenia. Następnym bardzo ważnym zadaniem systemu zarządzania danymi jest maksymalne ułatwienie korzystania z nich. Klasyczne języki programowania, takie jak np. COBOL, pozwalają korzystać z różnorodnych urządzeń przechowujących dane oraz z plików o różnych organizacjach (pliki sekwencyjne, randomowe, isam), lecz na czas realizacji programu "zawłaszczają dane". Żaden program nie może aktualizować danych, które zostały przyłączone już do innego programu z pozwoleniem na dokonywanie zmian. Problem ten jest szczególnie poważny w sytuacjach, kiedy dane obejmują szeroki zakres tematyczny i są ze sobą mocno powiązane. Wtedy program pracujący z nimi blokuje dostęp do dużego ich zakresu. Aby problem ten rozwiązać, należy zakazać bezpośredniego dostępu do danych i wszelki dostęp do nich zlecić wyspecjalizowanym modułom programowym. Tak opakowane dane (dane oraz procedury dostępu do nich) nazywamy bazami danych. Rozwiązanie to nasuwa wiele konsekwencji. Dane stają się dobrem chronionym przy pomocy zautomatyzowanych narzędzi, potrzeba więc formalnej ich rejestracji. No i, doszliśmy do kartoteki danych, narzędzia rejestrującego dane w taki sam sposób, jak baza danych rejestruje rzeczywistość. Musimy więc umieć opisać logiczną strukturę danych w sposób formalny. Następna konsekwencja to język manipulowania danymi. Jeśli nie możemy mieć bezpośredniego dostępu do danych, to musimy mieć taki sposób wyrażania naszych żądań, abyśmy mogli dotrzeć do każdej informacji zapisanej w bazie danych. Powstaje problem zdefiniowania i zaimplementowania odpowiedniego języka. Języki takie nazywane są językami zapytań lub językami manipulowania danymi. Mogą to być języki zanurzone w istniejących językach programowania (przeważnie w COBOL-u) lub języki niezależne posiadające wszystkie potrzebne programiście mechanizmy. Oddzielenie programu użytkownika od danych pozwala uniezależnić te programy od fizycznej alokacji danych. Umożliwia to zmiany w fizycznym rozmieszczeniu danych bez potrzeby dokonywania korekt w programach. Jest to bardzo ważny moment z uwagi na optymalizację dostępu do danych. Ale każdy medal ma dwie strony. Tą drugą stroną jest niemożność optymalizacji dostępu do danych przez programistę (trudno jednoznacznie ocenić tę cechę, wiem, że wielu programistów może znaleźć bliższe optymalnemu rozwiązanie niż system zbudowany dla zastosowań uniwersalnych, uważam jednak, że średnia wypada mocno na korzyść systemu).

Wróćmy teraz do struktur danych. W kartotece musimy umieć opisać w sposób formalny, przynajmniej, strukturę danych, aby móc realizować ich wyszukiwanie i uaktualnianie. Musimy również zapewnić sobie możliwość zbudowania sprawnego języka manipulowania danymi, który pozwoli nam wykorzystać każdą informację zapisaną w danych, wprowadzić do bazy każdą informację jaką możemy w wybranej strukturze zawrzeć i który nie wpadnie w pułapki strukturalne, jak na przykład nieskończona pętla (o co bardzo łatwo w strukturach rekurencyjnych). Ponadto struktura powinna być na tyle ogólna, aby można w nią odwzorować szeroki zakres danych pochodzących ze świata rzeczywistego. Pierwszymi strukturami, na których budowano systemy baz danych, były drzewa. Dobrze poznana teoria drzew dawała szansę zbudowania dobrze pracujących systemów z łatwymi językami zapytań. Bazy takie nazywane są bazami hierarchicznymi. Niestety praktyczne potrzeby często wykraczają poza zakres możliwości baz hierarchicznych i obecnie niemal całkowicie straciły one znaczenie. Aktualnie stosowane bazy danych, to bazy sieciowe - struktura danych jest grafem skierowanym o minimalnych ograniczeniach - oraz bazy relacyjne - struktura danych jest odwzorowana w tablice. Oba te modele danych pozwalają na odwzorowanie dowolnych związków zachodzących pomiędzy danymi. Różnią się zasadniczo sposobem zapisu tych związków, co powoduje, że w konkretnych sytuacjach powinniśmy starannie przemyśleć, jaki model zastosować, ponieważ nie są one wzajemnie zmienne. Również istnieją różnice pomiędzy systemami baz danych w ramach tego samego modelu. Rodzi się więc pytanie, czy użycie nowocześniejszego rozwiązania systemów baz relacyjnych nie wyeliminuje potrzeby stosowania baz sieciowych, które na ogół są trudniejsze w implementacji. Odpowiedź brzmi - nie. Przyczyna tkwi głęboko i chociaż nowocześniejsze rozwiązania oraz szybsze komputery przesuwają granice zastosowań na korzyść baz relacyjnych, jednak długo będziemy zmuszeni do korzystania z obu modeli. Oczywiście jest jeszcze trzecia możliwość, nie stosować bazy danych i użyć rozwiązań korzystających z plików niezależnych oraz udogodnień systemu operacyjnego (one też są

ciągle doskonalone). W przypadkach gdy dane są tylko luźno powiązane ze sobą, to rozwiązanie może okazać się tańsze i wydajniejsze. Aby dokonać właściwego wyboru, musimy poznać przynajmniej ogólne cechy każdego z rozwiązań.

Bazy sieciowe

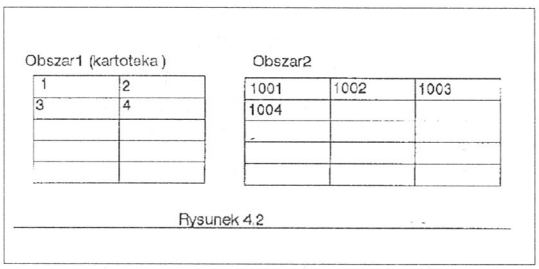
Istnieje więcej niż jedno rozwiązanie systemu zarządzania bazą danych dla modelu sieciowego. Dominują jednak rozwiązania zbudowane w oparciu o raporty Data Base Task Group organizacji CODASYL z lat 1971, 1972, 1976. Dlatego właśnie na przykładzie tego rozwiązania postaram się przedstawić podstawowe idee baz sieciowych. Strukturą danych takiej bazy może być dowolny graf skierowany, który lokalnie jest drzewem dwupoziomowym. Zatem niedopuszczalne są rekurencje bezpośrednie, lecz rekurencja pośrednia jest legalna w bazie codasylowskiej. To ograniczenie jest więc niezbyt krępujące, ponieważ zawsze możemy rekurencję bezpośrednią doprowadzić do pośredniej wykorzystując do tego celu dodatkowy rekord. Logiczną strukturę danych zapisujemy w postaci tzw. schematu (czasami nazywanego konceptualnym lub wewnętrznym). W schemacie opisujemy struktury wszystkich rekordów, atrybuty wszystkich pól rekordów oraz informacje o powiązaniach pomiędzy rekordami. Powiązania te wyraża się przy pomocy tzw. setów (inne używane nazwy to kolekcja oraz grupa logiczna). Set codasylowski, to drzewo dwupoziomowe, którego korzeń nazywany jest właścicielem setu, zaś liście członkami setu. Set może mieć człków więcej niż jednego typu. Rekord podlega takim samym ograniczeniom jak w COBOL-u. Również pola rekordu muszą spełniać wymogi COBOL-u. Aby dostosować wyspecyfikowaną strukturę danych do wymogów schematu codasylowskiego, należy usunąć wszystkie rekurencje bezpośrednie i podzielić całość na sety. Porównajmy rysunek 4.1 z rysunkiem 3.1. Rekord



Rysunek 4.1

PRZEŁOŻONY wprowadzony został w celu wyeliminowania bezpośredniej rekurencji, rekord PRZYZNANY - w celu rozłożenia zależności "wiele do wielu" na dwa związki typu "jeden do wiele". Teraz już bez trudu można podzielić całość na sety. W tym przypadku mamy ich cztery: KIERUJE, POSIADA i dwa nie nazwane z powodu mojej niezdolności do wymyślania ładnych nazw, jednak pisząc schemat musiałbym je nazwać. Oprócz nazwy setu w schemacie musimy podać nazwę właściciela, nazwy członków (w rozpatrywanym przykładzie nie ma setów z członkami wielu typów) oraz charakter powiązania. Charakter powiązania wyrażany jest przy pomocy dwóch wskaźników: kodu wprowadzania - wskazującego, czy w chwili wprowadzania do bazy rekordu będącego członkiem setu system będzie wymagał identyfikacji jego właściciela i w konsekwencji automatycznie przyłączy wprowadzany rekord do setu, czy nie - oraz kodu utrzymania - wskazującego czy system może pozwolić na odłączenie członka setu od setu.

Sposób rozmieszczania rekordów w fizycznych zasobach opisuje się w schemacie przydziału zasobów. Podstawowymi pojęciami w tej części opisu bazy danych są: obszar, plik logiczny, strona, rekord, set, umieszczenie. Całość zasobów bazy danych podzielona jest na obszary (Areas).

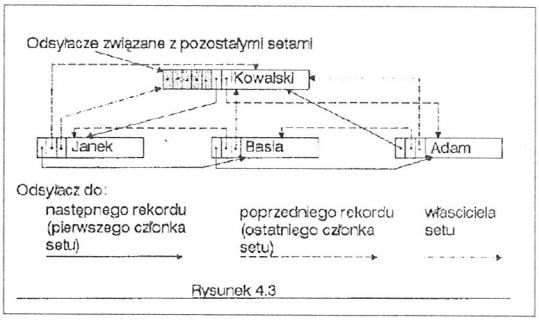


Rysunek 4.2

Każdy obszar podzielony jest na strony. Z każdą stroną związany jest jej numer, który jest unikatowy w całej bazie . Jeden obszar jest wyróżniony i służy do zapisywania informacji o samej bazie. Właśnie w tym obszarze zapisany będzie schemat bazy danych, schemat przydziału zasobów, podschematy (o tym później), tam również zapisane będą różne informacje niezbędne systemowi zarządzania bazą w czasie realizacji programów (wskaźniki stanu bazy danych, statystyki przebiegu programu itp.). Pozostałe obszary przeznaczone są na przechowywanie naszych danych. Obszarów tych może być więcej niż jeden, każdy musi mieć zadeklarowany przedział numerów stron. Wszystkie te przedziały muszą być parami rozłączne, luki w numeracji pomiędzy obszarami są pożądane ze względu na ewentualne powiększanie obszarów (jeśli system dopuszcza taką możliwość). Wielkość strony ustalana jest na podstawie wielkości bloku pliku logicznego i może być różna dla każdego pliku. Obszary sa odwzorowywane w pliki logiczne. Jeden obszar można zapisać w wielu plikach, również jeden plik może mieścić w sobie wiele obszarów. Połączenie plików logicznych z fizycznymi następuje dopiero w chwili wywołania programów użytkownika i leży w gestii systemu operacyjnego. Tak więc jeden zestaw plików logicznych może być łączony z wieloma kompletami plików fizycznych.

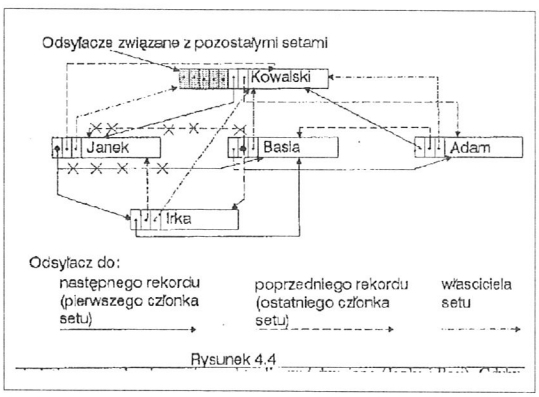
Strona dzieli się na linie. Linia w zasadzie odpowiada rekordowi - jeśli pominiemy takie szczegóły jak linie organizacyjne oraz rozszerzenia rekordu o dodatkowe pola (odsyłacze, wskaźniki rekordów zmiennej długości, wskaźniki

fragmentowania rekordu). Każda linia posiada kolejny numer w ramach strony. Para (numer strony, numer linii) jednoznacznie wyznacza położenie rekordu w bazie danych. Możemy więc umieścić rekord pod dowolnym adresem a połączenia setu realizować w postaci odsyłaczy do



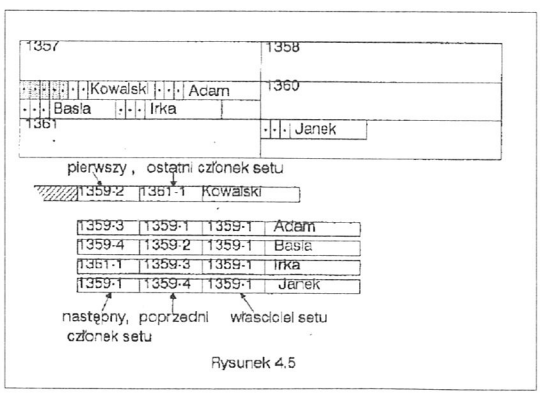
Rysunek 4.3

konkretnej strony i linii. Na rys. 4.3 przedstawiam przykładową inkarnację setu POSIADA. W celu sprawnej nawigacji zarówno w przód jak i do tyłu zazwyczaj set ma dwukierunkowe połączenia członków. Ponadto każdy członek posiada odsyłacz do właściciela setu. Przyjęcie takiego układu odsyłaczy nie jest obowiązkowe, lecz daje korzyści czasowe w procesie wyszukiwania informacji. Powoduje jednak konieczność aktualizacji dwóch rekordów dla każdego setu w przypadku wprowadzenia nowego członka setu. Rozpatrzymy to na przykładzie chęci dopisania panu Kowalskiemu kolejnego dziecka. Niech to będzie Irka, której zapis chcemy umieścić pomiędzy zapisami Janka i Basi. W tym celu oprócz wprowadzania



Rysunek 4.4

dodatkowego rekordu musimy zaktualizować dwa inne (Janka i Basi). Gdyby rekordy posiadały połączenia tylko w jednym kierunku, wystarczyłaby aktualizacja tylko jednego aktualnego rekordu, lecz kasowanie rekordu wymagałoby dodatkowego przejścia przez set. Następną informacją, którą musimy zapisać w schemacie przydziału zasobów, jest informacja o miejscu i sposobie lokalizacji poszczególnych rekordów. Miejsce określamy podając nazwy obszarów, w których dany typ rekordu może zostać zapisany, możemy również wyspecyfikować zakres stron dla każdego z obszarów dostępny dla typu rekordu. Istnieje wiele możliwości wyboru strony, w której konkretyzacja rekordu może być zapisana. Najszybsze z uwagi na dostęp do rekordu, jest odwzorowanie wartości pola (lub grupy pól) zwanego kluczem rekordu na numer strony z dostępnego zakresu i umieszczenie rekordu w tej wyliczonej stronie. Z oczywistych powodów może okazać się, że żądana strona jest już zajęta i rekord nie może być w niej zapisany. Problem ten rozwiązuje się łącząc wszystkie rekordy przynależące do jednej strony dodatkowym łańcuchem, który zawsze zaczyna się w linii zerowej strony tworząc dodatkowy organizacyjny set. Teraz już nie mieszczące się rekordy można zapisywać w najbliższej wolnej stronie (pierwszy poziom nadmiarowy) lub w przeznaczonym do tego celu specjalnym zakresie stron (drugi poziom nadmiarowy). Takie rozmieszczenie przy dobrym algorytmie randomizacji oraz upakowaniu około 65% powoduje, że dla uzyskania rekordu potrzeba średnio tylko 1.1 czytanych bloków. Rekordy będące członkami setu możemy umieszczać "via set" tzn. strona macierzysta rekordu jest macierzystą stroną właściciela setu. Odsyłacze setu rozwiązują tutaj również problem przepełnienia strony. Dostęp do rekordu umieszczonego "via set" może być uzyskany tylko poprzez set. Najkorzystniejszą sytuacją tutaj jest, aby możliwie najwięcej członków setu rzeczywiście mieściło się na tej samej stronie co właściciel. Jeśli sety są zbyt liczne, aby mogły pomieścić się na jednej stronie i nie możemy lub nie chcemy organizować dostępu poprzez klucze bezpośrednie, możemy zlecić umieszczanie rekordów systemowi w sposób całkowicie dowolny. Aktualnie oferowane na rynku systemy baz danych dopuszczają jeszcze inne sposoby lokacji rekordów. Jak zwykle nie ma idealnego sposobu na wszystkie okazje, każdy z nich ma swoje wady i zalety. Do nas należy wybór, a to w czym możemy wybierać, zależy od konkretnego systemu zarządzania bazą danych. Jeśli zdecydujemy, aby w naszym przykładzie rekord PRACOWNIK był umieszczany w stronie wyliczonej na podstawie nazwiska a rekordy typu DZIECKO via POSIADA, to możemy otrzymać takie oto rozmieszczenie (rys. 4.5):



Rysunek 4.5

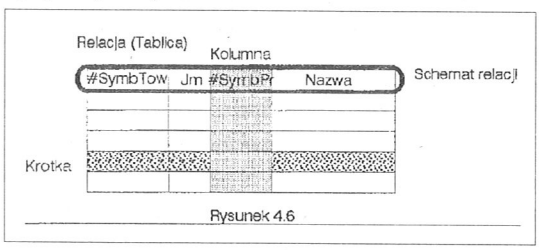
Schematy, konceptualny i przydziału zasobów, informują system zarządzania bazą danych o tym, co będzie przechowywane oraz w jaki sposób. Na ich podstawie system potrafi zarządzić danymi, lecz zazwyczaj schemat konceptualny jest

dość duży, podczas gdy poszczególne programy użytkownika wymagają dostępu tylko do fragmentu bazy. Aby nie przeciążać systemu zbyt dużymi tablicami opisu struktur w czasie realizacji programów użytkowych, wprowadzono jeszcze jeden poziom opisu danych. Realizują go podschematy bazy danych (czasami zwane schematami zewnętrznymi). Podschemat opisuje fragment bazy danych wykorzystywany przez program użytkownika, w kategoriach rekordów i setów. Język manipulowania danymi w bazie codasylowskiej (DML) jest kolekcją poleceń rozszerzających COBOL. Zatem COBOL jest podstawowym językiem programowania dla tych baz, a podschemat jest tym fragmentem definicji bazy, który jest kopiowany do Data Division programu. Dodatkowo w podschemacie możemy zabronić wykonywania przez program niektórych poleceń DML oraz wskazać systemowi, jakie powinien wybrać mechanizmy blokowania równoczesnego dostępu do danych. Możemy żądać blokowania całych obszarów lub tylko stron. Czasami zachodzi konieczność przeszukania seryjnego wszystkich rekordów danego typu (np. w celu wybrania rekordów spełniających zadany warunek). Ponieważ rekordy jednego typu nie są połączone ze sobą (chyba że sami zorganizujemy taki set), istnieje konieczność przeszukiwania obszarów, w których rekordy tego typu są zapisane. Obszar jest pojęciem należącym do schematu przydziału zasobów i programista nie ma do niego dostępu. W celu wskazania miejsc, gdzie dany typ rekordu może występować, w podschemacie definiuje się zakresy (Realms). Zakres może być zdefiniowany jako pojedynczy obszar lub pojedynczy typ rekordu.

Sieciowe bazy danych lat osiemdziesiątych znacznie wykraczają poza raport CODASYL-u rozszerzając możliwości baz w zakresie optymalizacji alokacji danych, możliwości precyzyjnego dostosowania systemu zarządzania do konkretnych potrzeb, wygodniejszego komunikowania się z użytkownikiem itp. Podstawowa idea pozostaje jednak niezmienna, podział na sety (lub drzewa wielopoziomowe w innych rozwiązaniach) oraz łączenie rekordów w łańcuchy. Pozwala to nam szybko wyszukać rekordy związane z rekordem aktualnie nas interesującym. Wadą tego podejścia jest konieczność bardzo starannego przygotowania schematu konceptualnego. Nie możemy zaimplementować wszystkich możliwych połączeń, jesteśmy zmuszeni dokonać wyboru. Jeśli wybór okaże się nietrafny, naprawienie tego błędu wymaga restrukturyzacji bazy danych, co jest procesem niezwykle kosztownym i trudnym. Preferowany w pierwszych implementacjach sposób alokacji rekordów w oparciu o randomizacje klucza powoduje trudności, gdy przybywa danych i należy powiększać obszary. Ponieważ wyliczany numer strony zależy od deklarowanej liczby stron w obszarze, zmiana tej deklaracji powoduje zmiany w adresach już zapisanych rekordów. Należy ponownie załadować bazę przeliczając wszystkie adresy i dokonując pewnych uaktualnień odsyłaczy, proces ten nosi nazwę reorganizacji bazy danych i jest również jak restrukturyzacja bardzo czasochłonny. Nowoczesne systemy zarządzania bazami danych oferują mechanizmy pozwalające łagodzić te niedogodności.

Bazy relacyjne

Bazy relacyjne są alternatywnym rozwiązaniem problemu budowy uniwersalnego systemu zarządzania danymi. Narodziły się tak jak i sieciowe w latach siedemdziesiątych. Fundamentem całej konstrukcji jest możliwość odwzorowania dowolnej struktury danych w tablice dwuwymiarowe. Łatwo to zauważyć, jeśli uświadomimy sobie, że tablica jest formą zapisu podzbioru iloczynu kartezjańskiego zbiorów, do których należą wartości poszczególnych kolumn. Podzbiór iloczynu kartezjańskiego w matematyce nazywa się relacją, stąd też nazwa "bazy relacyjne". Łatwo się domyślić, że skoro podstawowe pojęcie pochodzi z matematyki, to tak też będzie z pozostałymi elementami teorii (tak mówimy o teorii relacyjnych baz danych). Rzeczywiście istnieją teorie matematyczne pokazujące jak można manipulować relacjami. Podstawowe z nich to algebra relacji i rachunek predykatów. W trakcie badań okazało się, że obie te teorie są równoważne, jeśli rozważymy zupełność algebr. Zatem każda z nich może stanowić podstawę języka zapytań. W trakcie rozwoju systemów relacyjnych baz danych ukształtowały się standardowe języki zapytań (SQL oraz Query By Example), które nie są ani czysto algebraiczne, ani predykatowe, niemniej są im równoważne i każdy liczący się producent systemów baz danych stara się je zaimplementować. Również ustaliło się standardowe nazewnictwo specyficzne dla relacyjnych baz danych. Spójrzmy na rysunek 4.6:



Rysunek 4.6

Pojedynczą tablicę nazywa się relacją (lub po prostu tablicą), wiersz relacji - krotką (tak jak stokrotka), kolumnę stanowią wartości odpowiedniego atrybutu. Pojęcie schematu relacyjnej bazy danych nie ma istotnego znaczenia (schematem nazywa się zbiór schematów wszystkich relacji). Istotny natomiast jest schemat relacji, który przedstawia porządek atrybutów w relacji z zaznaczeniem ewentualnych atrybutów wchodzących w skład klucza. Zapisuje się go w postaci krotki, kluczowe pola wyróżnia się znakiem "#",

przykładowa relacja ma schemat: (#SymbTow,Jm,#SymbPr,Nazwa). Mówiąc o bazach sieciowych pominąłem całkowicie sprawę pól reaktorów stwierdzając jedynie, że rekord może mieć strukturę taką, jaką dopuszcza COBOL tzn. pola rekordu mogą mieć swoje podpola, możemy definiować rekordy wariantowe oraz zmiennej długości. Krotka podlega ostrzejszym rygorom, może składać się jedynie z pól elementarnych, o wariantowości ani zmiennej długości nie może być mowy. Mówimy, że krotka musi być w pierwszej postaci normalnej (spełniać warunki jak wyżej). Relacje nie są ze sobą sztywno powiązane tak jak rekordy w bazie sieciowej. Ewentualne wiązania są realizowane w fazie odpowiedzi na zapytanie użytkownika. Podstawową rolę odgrywają wtedy wartości atrybutów w poszczególnych kolumnach. Zapytania do systemu zarządzania bazą danych są kierowane przy pomocy poleceń języka zapytań. Odpowiedzią jest relacja lub wartość funkcji, której dziedziną jest relacja (np. suma wartości kolumny, wartość maksymalna w kolumnie). Istotne jest, aby język pozwolił skonstruować każdą wywodzącą się z bazy relację. Taką cechę mają oferowane obecnie języki zapytań. Przyjrzyjmy się, jak one to robią na przykładzie algebry relacji. Podstawowymi operacjami algebry relacji są: selekcja, rzutowanie (projekcja) oraz połączenie (niekiedy dołączone są operatory iloczynu kartezjańskiego, sumy mnogościowej i różnicy symetrycznej, nie stanowią one jednak o zupełności języka).

Towar

SymbTow	Jm	SymbPr	Nazwa
1231	20	7813	sprzeglo
1231	20	7815	sprzeglo
1237	33	1501	kit
1451	33	1501	klej
1311	20	1502	kolo zebate
1318	20	7815	slimak
1319	20	7815	slimacznica

SELECT Towar WHERE SymbPro = 7815

Rysunek 4.7

Operator selekcji wybiera z relacji krotki spełniające zadany warunek logiczny i tworzy z nich relację. Niekoniecznie musi być ona zapisana w fizycznych jednostkach pamięci, lecz język musi zapewniać możliwość wykonania na niej innych operacji. Na rysunku 4.7 zamieściłem przykładowe użycie operatora selekcji. W konkretnych realizacjach składnia może być inna, ale efekt działania operacji taki sam. W wyniku zastosowania operatora rzutowania otrzymuje się relację złożoną z wybranych kolumn relacji, na które działamy tym operatorem. W wyniku działania przedstawionego obok otrzymamy relację złożoną z dwóch kolumn: SymbTow i SymbPr. Jeśli w relacji wynikowej miałyby się pojawić

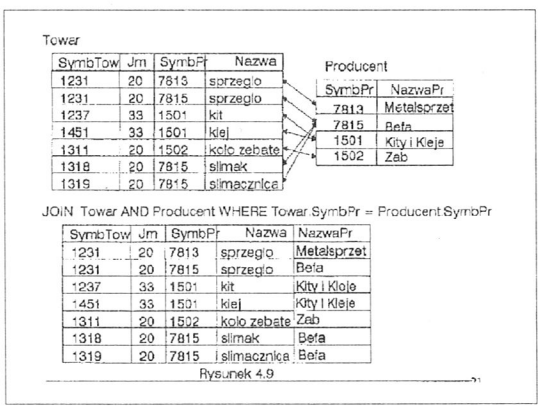
Towar

SymbTow	Jm	SymbPr	Nazwa
1231	20	7813	sprzeglo
1231	20	7815	sprzeglo
1237	33	1501	kit
1451	33	1501	klej
1311	20	1502	kolo zebate
1318	20	7815	slimak
1319	20	7815	slimacznica

PROJECT Towar OVER SymbTow SymbPr

Rysunek 4.8

krotki identyczne, to system powinien dokonać odpowiedniej redukcji. Operatory selekcji i rzutowania działają na pojedyncze relacje. Operator łączenia "pracuje" na parze relacji, w wyniku otrzymujemy relację złożoną z kolumn obu argu-



Rysunek 4.9

mentów. Poszczególne kroki obu relacji łączone są ze sobą, jeśli spełniają wyspecyfikowany warunek logiczny. Szczególnym przypadkiem operacji łączenia jest operacja iloczynu kartezjańskiego, wtedy w wyniku otrzymujemy połączenia krotek "każda z każdą". Operator łączenia pozwala wyeliminować z bazy danych sztywne odsyłacze do rekordów, co czyni bazy relacyjne znacznie elastyczniejszymi i łatwiejszymi do zaprojektowania. Niestety przekleństwem relacyjnych baz danych jest wykładnicza złożoność obliczeniowa tej operacji . Wyobraźmy sobie trzy relacje po 1000 krotek każda. Oszacujmy koszt połączenia ich. Dla każdej krotki pierwszej relacji musimy znaleźć odpowiednie krotki drugiej. Ponieważ każda może mieć wiele odpowiedników, nie możemy przerwać sprawdzania po znalezieniu pierwszego, zatem drugą relację musimy przejrzeć do końca. Musimy to uczynić tyle razy, ile jest krotek w pierwszej, co daje nam 1000*1000 porównań. Ponieważ mamy trzy relacje, otrzymujemy łącznie 10 porównań. Prawda, że trochę dużo, a przecież relacje są całkiem małe. Bardzo mocnym mechanizmem poprawiającym efektywność pracy baz relacyjnych są klucze. Jeśli krotkę możemy jednoznacznie zdefiniować na podstawie wartości zespołu atrybutów (może to być pojedynczy atrybut), to zespół ten (atrybut) nazywamy kluczem. W schemacie relacji mogą

się zdarzyć atrybuty będące kluczami innych relacji, takie atrybuty nazywa się kluczami obcymi. Połączenia według klucza mogą być wykonane znacznie szybciej, wykorzystując możliwości umieszczania krotek w obszarze pamięci wyliczonym na podstawie wartości klucza (np. tak jak opisałem to mówiąc o alokacji rekordów w bazach sieciowych). Takie połączenie nazywa się półpołączeniem, jego koszt wzrasta wielomianowo wraz z licznością relacji, szkoda tylko, że nie wszystkie problemy można rozwiązać przy pomocy połączeń. Chciałoby się powiedzieć: " no to budujmy relacje z dużą ilością atrybutów, w ten sposób wyeliminujemy wiele połączeń". Zobaczymy, co z tego wynika (patrz rys. 4.10). Wraz z

SymbTow	NazwaTow	Jm	Ilosc	DataDost	NazwaProd	AdresProd
1215	Poszwa	20	500	12.03.91	POLPOS	B-B Skosna 6
1218	Brokat	60	200	13.03.91	KOTEX	Andrychow Nowotki 3
1220	Poszewka	20	1000	13.03.91	POLPOS	B-B Skosna 6
1231	Firana	60	300	14.03.91	DEX	Warszawa Kwiatowa 8
1220	Poszewka	20	1000	15.03.91	POLPOS	B-B Skosna 6
1218	Brokat	60	200	15.03.91	POLPOS	Krakow Kopernika 8

Rysunek 4.10

każdą dostawą towaru musimy wprowadzić nazwę towaru oraz adres producenta - wartości, które jak łatwo zauważyć, wprowadzają niepotrzebną redundancję oraz utrudniają aktualizację. Nazwa towaru jest zależna funkcyjnie od symbolu towaru, więc wystarczy, aby była zapisana w bazie tylko w jednym miejscu (szczególnie, że zazwyczaj jest to dość długi tekst). Podobnie sprawa wygląda z adresem producenta, tutaj jednak nazwa nie identyfikuje producenta, należy więc wprowadzić dodatkowy identyfikator. Poprawiona baza może wyglądać tak:

Tower

#SymbTow	NazwaTow	Jm
1215	Poszwa	20
1218	Brokat	60
1220	Poszewka	20
1231	Firana	60

Producent

#SymbProd	NazwaProd	AdresProd
001	POLPOS	B-B Skosna 6
002	POLPOS	Krakow Kopernika 8
003	KOTEX	Andrychow Nowotki 3
004	DEX	Warszawa Kwiatowa 8

Dostawa

#SymbTow	#SymbProd	DataDost	Ilosc
1215	001	12.03.91	500
1218	003	13.03.91	200
1220	001	13.03.91	1000
1231	004	14.03.91	300
1220	001	15.03.91	1000
1218	002	15.03.91	200

Rysunek 4.11

Nie jest ona całkowicie równoważna pierwotnej relacji. Zauważmy, że poprzednio nie można było zapisać informacji o producencie ani o towarze, dopóki nie było dostawy towaru. Teraz możemy to uczynić - płacimy koniecznością wykonywania połączeń (w tym przypadku tylko półpołączeń). Jak widać, sztuka budowania baz relacyjnych jest sztuką wyszukiwania zależności funkcyjnych pomiędzy atrybutami i podziału bazy na relacje. Istnieją gotowe algorytmy postępowania doprowadzającego bazy relacyjne do "dobrej" postaci. Najważniejsze z nich to doprowadzenie do sytuacji, kiedy każdy atrybut jest zależny funkcyjnie od całego klucza (druga postać normalna) i następnie wyeliminowanie wszystkich tych atrybutów, które nie są bezpośrednio zależne od klucza (druga postać normalna). Relacje będące w trzeciej postaci normalnej również mogą posiadać niepożądane własności - potrzebna dalsza normalizacja. Sytuacje te są jednak znacznie mniej dokuczliwe. Normalizacja poprawia bazę danych eliminując niepożądane redundancje oraz usuwając niedogodności aktualizacji. W zamian prowadzi do zwiększenia ilości relacji, co z kolei zmusza nas do częstszego korzystania z operacji łączenia - coś za coś. Skoro tak, to musimy planować kolejność wykonania poszczególnych operacji prowadzących do uzyskania odpowiedzi w bazie w ten sposób, aby zminimalizować czas uzyskania tej odpowiedzi. Zobaczmy na przykładzie co możemy osiągnąć, weźmy następującą bazę.

Towar
(#SymbTowaru,Nazwa,Parametry)

Dostawca
(#SymbDostawcy, Region, Adres, Kontakt)

Zdolność·Produkcyjna
(#SymbDostawcy,#SymbTowaru,Wielkość Produkcji)

Odbiorca
(#SymbOdbiorcy, Region, Adres, Kontakt)

Zapotrzebowanie
(#SymbOdbiorcy,#SymbTowaru, Wielkość·Zapotrzebowania)

Spróbujmy skojarzyć poważnych dostawców i odbiorców tego samego towaru. Przyjmijmy, że poważny dostawca, to taki, którego wielkość produkcji przekracza 100 tys. jednostek towaru. Odpowiedni odbiorca zapotrzebowuje również ponad 100 tys. jednostek. Możemy nasze zadanie wykonać kilkoma sposobami, wybierzmy trzy.

Sposób 1:
Wynik:= JOIN ZdolnośćProdukcyjna AND Zapotrzebowanie WHERE MATCHING SymbTowaru AND (WielkośćProdukcji > 100000) AND (WielkośćZapotrzeb > 100000)

Sposób 2:
WZdolnośćProdukcyjna: =SELECT ZdolnośćProdukcyjna WHERE WielkośćProdukcji > 100000;
Wynik: =JOIN WZdolnośćProdukcyjna AND Zapotrzebowanie WHERE

MATCHINGSymbTowaruAND(WielkośćZapotrzeb > 100000.

Sposób 3:
WZdolnośćProdukcyjna: = SELECT ZdolnośćProdukcyjna WHERE WielkośćProdukcji > 100000;
WZapotrzebowanie: = SELECT Zapotrzebowanie WHERE WielkośćZapotrzeb > 100000;
Wynik: = JOIN WZdolnośćProdukcyjna AND WZapotrzebowanie WHERE MATCHING SymbTowaru.

Wszystkie rozwiązania należałoby jeszcze uzupełnić o dobieranie nazw i adresów potencjalnych kooperantów oraz nazw towarów, koszt tych operacji (półpołączeń) jest jednakowy dla każdego z nich, więc nie będę się nimi zajmował. Spróbujmy ocenić, które z tych rozwiązań wymaga najmniejszej ilości porównań (złożone wyrażenie logiczne traktuję jako jedno porównanie, ponieważ potrzebujemy tylko jednego dostępu do zapisu fizycznego). Przyjmijmy, że relacje ZdolnośćProdukcyjna i Zapotrzebowanie mają po tysiąc krotek. Sposób pierwszy wymaga 10 porównań. Ilość porównań w rozwiązaniu drugim zależy od ilości krotek po selekcji. Jeśli okaże się, że np. tylko 10 producentów może produkować po ponad 10 tysięcy jednostek towaru w jednostce czasu, to otrzymujemy 1000 porównań dla selekcji plus 10*1000 porównań w operacji łączenia. Razem rozwiązanie to kosztuje nas 11000 porównań. Na koszt rozwiązania trzeciego dodatkowo ma wpływ ilość odbiorców zgłaszających duże zapotrzebowanie, przyjmując tak jak poprzednio 1%, mamy 1000 porównań dla pierwszej selekcji, tysiąc dla drugiej i 10*10 dla łączenia. Razem łączny koszt wynosi 2100 porównań. Czy zawsze trzecie rozwiązanie będzie najtańsze? Z całą pewnością nie, zależy od efektów pierwszych selekcji. Użytkownik układając plan rozwiązania nie wie, które z rozwiązań jest najefektywniejsze, jeśli nie posiada dodatkowej wiedzy o rozkładzie wartości danych. Rozkład może zmieniać się w czasie i najlepsze rozwiązanie na dziś będzie niedobre za miesiąc! Istnieją systemy zarządzania relacyjnymi bazami danych prowadzące statystyki rozkładu wartości dla poszczególnych kolumn. Mogą to być rejestratory wartości ekstremalnych, system INGRES dostarcza również histogramy rozkładów wartości dla wybranych kolumn. Niemożność ustalenia najlepszego planu uzyskania odpowiedzi na postawione bazie pytanie obowiązujące niezależnie od jej zawartości stała się motorem poszukiwań rozwiązań automatycznych. Języki proceduralne zastąpiono językami deklaratywnymi. Użytkownik informuje system o swoich potrzebach, a plan otrzymania rozwiązania układa system, który następnie realizuje go. System może wtedy wykorzystać dodatkową wiedzę o bazie gromadzoną w całym okresie jej życia. Te właśnie mechanizmy zasadniczo różnią systemy zarządzania bazami danych w ramach modelu relacyjnego.

Bazy kontra pliki niezależne

Wielokrotnie spotkałem się z zarzutem, że systemy zarządzania bazami danych pożerają nadmierną ilość zasobów komputerowych. Rezygnując z nich możemy lepiej upakować dane i bardziej efektywnie wykorzystać czas procesora. To prawda, narzędzia uniwersalne (a takimi są w zakresie obróbki danych systemy zarządzania bazami) nie mogą być tak precyzyjnie dostosowane do specjalnych potrzeb jak narzędzia budowane na indywidualne potrzeby. Mimo to już od wielu lat nie chcemy programować w asemblerach (są wyjątki), lecz wolimy używać języków wyższych generacji. Straty poniesione z powodu niedoskonale wykorzystanych możliwości procesora z nawiązką są rekompensowane przez szybsze tworzenie programu, wdrażanie go do eksploatacji, jego przenośność oraz możliwość łatwiejszego dokonywania zmian. Analogicznie jest z bazami danych. Autorzy bazy IDMSX firmy ICL podają następujący statystyczny rozdział zasobów procesora: 40% system zarządzania bazą, 30% system zarządzania przetwarzaniem transakcji, 25% system operacyjny, 5% program użytkownika. Czy należy się smucić z tego powodu? Uważam, że wręcz przeciwnie, te pięć procent traktuję jako ilość roboty, która mi pozostała do zrobienia przy budowaniu systemu w stosunku do tego, co musiałbym zrobić nie posiadając bazy danych. Oczywiście zawsze jest kwestia czy operacje wykonywane przez system są mi potrzebne, czy warto za nie aż tyle płacić. Jeśli potrzebujemy raz na miesiąc zrobić listę na mydło dla pracowników, to nonsensem byłoby budowanie takiego "systemu" wykorzystując do tego celu bazę danych. Jeśli natomiast chcemy udostępnić do interaktywnej aktualizacji wiele powiązanych ze sobą danych, to bez wątpienia baza danych będzie dla nas kolosalnym ułatwieniem przy budowie takiego systemu. Systemy zarządzania bazami danych budowane są przez duże zespoły specjalistów i bardzo starannie weryfikowane. Użycie takiego systemu zwalnia nas od programowania "stałych elementów" zarządzania danymi. Unikamy w ten sposób wielu błędów, które mielibyśmy okazję popełnić.

Bazy relacyjne kontra sieciowe

Dobra teoria, wzrastająca szybkość komputerów, łatwość implementacji, przenośność i parę jeszcze innych zalet baz relacyjnych spowodowały gwałtowny ich rozwój. Wielka ilość oferowanych systemów zarządzania relacyjnymi bazami danych może powodować wrażenie, że tylko ten model

wart jest zachodu i jeśli już baza, to tylko relacyjna. Spróbujmy się zastanowić przez chwilę nad tą kwestią. Podstawową różnicę między modelem relacyjnym i sieciowym stanowi sposób rejestracji związków zachodzących między danymi. W bazach sieciowych łączy się na sztywno rekordy przy pomocy odsyłaczy (tzw. twarde sety). Ma to wiele złych stron. Po pierwsze - raz zaimplementowane połączenia obowiązują do czasu restrukturyzacji bazy danych, co zazwyczaj jest rzeczą kosztowną. Projektant bazy musi posiadać sporo wiedzy o systemie, potrzebach i zwyczajach użytkowników, aby w przyszłości nie okazało się, że chociaż informacja jest zapisana w bazie, to nie można lub trudno ją uzyskać. Bazy sieciowe są mało podatne na zmiany struktury. Następną niedogodnością jest potrzeba poświęcenia pamięci na zapisanie odsyłaczy. W przytaczanym wcześniej rozwiązaniu połączeń setu potrzeba w rekordzie zarezerwować po dwa słowa dla każdego setu, którego rekord jest właścicielem oraz po trzy dla każdego setu, którego jest członkiem. Praktycznie niemożliwe jest zbudowanie setów dla wszystkich możliwych powiązań danych. Brak tej pełności ogranicza możliwości budowania języków deklaratywnych "na wszystkie okazje". Przy wszystkich tych wadach to rozwiązanie ma jedną kapitalną zaletę, czas dostępu do danych nie zależy od danych zapisanych w bazie. Nawigując po setach, dane uzyskujemy mniej więcej w tym samym czasie, niezależnie czy w bazie będzie pięć tysięcy inkarnacji setów, czy pięć milionów. Czas dostępu zależy od czasu dostępu do pierwszego właściciela (zazwyczaj adres jego uzyskujemy z przeliczenia klucza) i od długości przeszukiwanych setów. Ta cecha powoduje, że bazy sieciowe nadają się do budowania systemów rejestrujących bardzo duże ilości danych. Powiązania w bazach relacyjnych są tworzone w czasie realizacji zapytania, to implikuje liniowy wzrost czasu oczekiwania na odpowiedź w stosunku do wzrostu ilości krotek w relacji i wykładniczy w stosunku do ilości relacji biorących udział w procesie obliczeniowym. Stąd całkiem przyzwoicie zachowujące się bazy relacyjne mogą nie nadawać się do użytku po przekroczeniu pewnego progu ilości danych. Zastosowanie mocniejszego komputera przesunie ten próg, lecz nie wyeliminuje jego istnienia. Bardzo pomocne są tu mechanizmy indeksowania. Jeśli wprowadzimy indeksy (pośrednie lub bezpośrednie), to wiele połączeń można zastąpić półpołączeniami a więc operacjami znacznie tańszymi. Indeksy hierarchiczne pozwalają z kolei wprowadzić pojęcie danych uporządkowanych. Daje to istotne korzyści eliminując dodatkowe sortowanie danych, dotyczy w równym stopniu baz sieciowych jak i relacyjnych. Często żądamy, aby system kontrolował w pewnym zakresie dane wprowadzane i nie dopuścił do zapisania niepoprawnych wartości. Kontrole wprowadzanych danych z uwagi na typ czy zakres mogą być takie same niezależnie od modelu bazy. Częstym problemem w relacyjnych bazach jest zachowanie zgodności referencyjnej, tzn. stawiamy obowiązek istnienia pewnego zapisu, w chwili gdy wprowadzamy do bazy inny np. musi istnieć dostawca, ażeby móc do niego sformułować zamówienie. Problem ten nie istnieje w przypadku baz sieciowych, tam proces ten jest naturalny, ponieważ musimy rekord przyłączyć do właściciela setu.

Struktura rekordu w bazach sieciowych może być znacznie bogatsza niż krotki. Korzystając z tej własności możemy często zmniejszyć zapotrzebowanie na pamięć konstruując rekordy zmiennej długości lub zapisując tablice w rekordzie (np. możemy rejestrować comiesięczny stan zapasów w magazynie w jednym rekordzie, oszczędzamy pamięci na wspólne identyfikatory i odsyłacze). Takich manipulacji nie można wykonywać w bazach relacyjnych, jest to własność pierwotna krotki. Z drugiej jednak strony normalizację możemy i powinniśmy stosować również dla baz sieciowych, daje to bardzo dobre wyniki i powoduje, że baza jest znacznie odporniejsza na zmiany decyzji użytkownika.

Wniosek płynący z tego co zostało wyżej wyrażone brzmi: żaden z prezentowanych tu modeli baz danych nie jest lepszy od drugiego. Każdy ma swoje zalety i wady. Skazani jesteśmy na dokonywanie wyboru w zależności od tego co i jak chcemy rozwiązać. Wydawać by się mogło, że w przypadku małych ilości danych i bardzo wyrafinowanych związków między nimi baza relacyjna jest rozwiązaniem doskonałym. Że tak nie jest, wiedzą ci, którzy sięgają po języki logiki i bazę wiedzy.

Bazy rozproszone

Rozwój sieci komputerowych nasuwa myśl zbudowania takiego systemu zarządzania bazą danych, który potrafiłby administrować danymi zapisanymi w różnych węzłach sieci i dostęp do niego również mógłby być uzyskany z dowolnego węzła. Byłaby to BAZA ROZPROSZONA. Zasadniczym problemem w realizowaniu tego celu jest znikoma szybkość przesyłania danych pomiędzy węzłami sieci w porównaniu z szybkościami uzyskiwanymi w pojedynczym systemie komputerowym. Próba zbudowania bazy rozproszonej opierając się na modelu sieciowym nie wróży powodzenia z wielu powodów. Istnieje podstawowa sprzeczność pomiędzy bazą sieciową a siecią komputerową. Sieci komputerowe tworzy się po to, aby dynamicznie zmieniać zasięg przetwarzania. Zaś baza sieciowa wymaga precyzyjnego zdefiniowania zarówno struktury jak i ilości danych. Zatem nie leży w naturze bazy sieciowej dołączanie nowych obszarów. Inną cechą bazy sieciowej jest tworzenie fizycznych połączeń pomiędzy rekordami w trakcie jej aktualizacji i ich

eksploatacja podczas czytania danych, stąd konieczność szybkiej i sprawnej nawigacji w bazie. Szybkość transmisji, jaką osiąga się w aktualnie istniejących sieciach, jest niewystarczająca dla takich celów. Poszczególne inkarnacje setów powinny być w całości zapisane w jednym węźle. Najwyższy już czas zastanowić się, czy nie lepiej w każdym węźle sieci posiadać bazy scentralizowane. Możliwość wymiany danych pomiędzy węzłami sieci jest "wbudowana" w sieć. Pewną nadzieję na bazę rozproszoną daje model relacyjny. Należy jednak ustalić zakres "rozpraszania" (pojedyncza relacja może być zapisana w wielu węzłach sieci, czy nie?). W zależności od odpowiedzi na to pytanie rodzą się następne. Jak prowadzić statystyki niezbędne do optymalizacji planu uzyskiwania odpowiedzi? Jak w końcu taki plan budować? Są to bardzo trudne problemy. Aktualnie nie znam ani jednego systemu spełniającego warunki stawiane bazom rozproszonym.

Jest jednak coś, co zostało rozwiązane. Otóż dosyć dokuczliwą własnością baz danych jest niemożność korzystania w jednym programie z danych zapisanych w dwóch lub więcej bazach. Istnieją systemy zarządzania bazą danych, które pozwalają na równoczesny dostęp do wielu baz przy pomocy specjalnych modułów programowych integrujących je (takimi są prezentowane w tej książeczce bazy ORACLE i INGRES). Nieważne przy tym jest, czy bazy zapisane są w jednym, czy wielu węzłach sieci. To rozwiązanie często nazywane jest systemem zarządzania bazą rozproszoną.

4. Systemy zarządzania bazami danych

W poprzednim rozdziale skupiłem uwagę na modelach baz danych i na konsekwencjach płynących z przyjęcia konkretnego rozwiązania. Teraz chciałbym zająć się pewnymi zjawiskami występującymi zawsze, gdy rozwiązujemy jakiś problem dotyczący przetwarzania danych niezależnie czy podeprzemy się systemem baz danych, czy zdecydujemy się budować całość przetwarzania na "własną rękę". Decydując się na użycie systemu zarządzania bazą danych musimy wiedzieć, jak on rozwiązuje te "stałe elementy" przetwarzania. Jest ich sporo, tutaj zajmę się tylko najważniejszymi.

Wielodostęp

Rozsądny system przetwarzania danych korzystający z udogodnień systemu zarządzania bazą danych zawiera wiele typów danych logicznie ze sobą powiązanych i pokrywa spory zakres tematyczny. Istotne staje się wtedy umożliwienie wielu użytkownikom jednocześnie korzystania z zasobów bazy danych zarówno w trybie odczytu jak i uaktualniania (dopisywanie, kasowanie, zmiana wartości danych). Aby każdy z tych użytkowników mógł być pewny poprawnego wykonania polecenia skierowanego do bazy danych, system musi posiadać mechanizmy synchronizujące ich pracę. Wyobraźmy sobie sytuację, kiedy dwóch użytkowników systemu płacowego sprawdza równocześnie zarobki ważnej osoby i stwierdza, że stawka jest błędnie zawyżona o 30% W celu doprowadzenia do poprawnego stanu każdy z nich nie wiedząc o poczynaniach drugiego zaniża wartość stawki o 30%. Łączne zmniejszenie wynosi więc 51%. Aby uniknąć tego typu nieprzyjemności, zasoby do których użytkownik zgłasza chęć aktualizacji, powinny być zablokowane dla innych użytkowników do czasu, kiedy użytkownik aktualizujący nie zwolni ich ponownie do ogólnego użytkowania. Rodzą się teraz wątpliwości: jakiego rodzaju blokada powinna być zastosowana oraz jaki fragment danych powinien być objęty tą blokadą. Żądanie danych dla aktualizacji nie musi powodować zakazu ich czytania ale może, powinno natomiast zakazywać równoczesnej aktualizacji innemu użytkownikowi. Jeśli idzie o zasięg blokady, sprawa jest bardziej skomplikowana. Najwygodniej z punktu widzenia użytkownika byłoby, gdyby blokowane były tylko rekordy podlegające aktualizacji, tyle tylko, że jest to najdroższe rozwiązanie. Ponadto często czytając rekord (w sieciowej bazie danych) nie wiemy czy będzie on aktualizowany, czy nie (proponuję przeanalizowanie procesu dopisywania rekordu do setu). Najtańszym rozwiązaniem jest blokowanie całych relacji lub w bazach sieciowych obszarów. W tym przypadku jednak istotnie jest zmniejszana dyspozycyjność bazy. Często przyjmowanym rozwiązaniem jest blokowanie fizycznych jednostek zapisu tj. bloków pliku, w których żądane rekordy są zapisane lub stron w bazach sieciowych. Konsekwencją blokowania fragmentów zasobów dostępnych wielu użytkownikom jest możliwość pojawienia się zakleszczeń (dead lock). Ponieważ system nie może zapobiec zakleszczeniom należy sprawdzić jak na nie reaguje, czy potrafi je wykryć. Użytkując bazę danych miewamy różne potrzeby i czasami zadeklarowane sposoby ochrony wielodostępu nie odpowiadają nam, powinniśmy mieć możliwość ich zmiany na jakiś czas lub nawet zaniechania ich użycia przez system. Każdy system posiada jakieś programy pracujące wsadowo w taki sposób, że żaden inny program w tym czasie nie może pracować, wtedy mechanizmy ochrony wielodostępu tylko przeszkadzają i nie należy z nich rezygnować na czas pracy tego programu.

Ochrona danych przed nielegalnym wykorzystaniem

Bazy danych nie są zazwyczaj jedynym dobrem w systemie komputerowym. Dobry system operacyjny musi zapewnić ochronę wszystkich

zasobów. Istnieją jednak pewne specyficzne sytuacje dla bazy danych, z którymi powinien radzić sobie system zarządzania bazą. Taka sytuacja pojawia się np. wtedy gdy chcemy zabronić czytania danych przez użytkownika, który musi mieć dostęp do rekordu ze względu na nawigację w bazie. CODASYL rozwiązuje ten problem w ten sposób, że oddziela dostęp do rekordu od czytania rekordu oraz pozwala w podschemacie bazy zakazać użycia dowolnego z poleceń języka manipulacji danymi. Inny problem wynika z faktu, że systemy operacyjne chronią pliki, podczas gdy w bazie chcemy chronić zapis z dokładnością do rekordu (krotki) lub nawet pola (kolumny). Ochronę taką może organizować system zarządzania bazą. Nietrudno również tak zbudować bazę danych, aby zawierała informacje o użytkownikach systemu i ich uprawnieniach z możliwością używania hasła.

Ochrona danych przed awariami sprzętu lub systemu

Sprzęt komputerowy - tak jak wszystko - może ulec awarii. Awarie mogą dotyczyć każdego elementu sprzętu i mogą się zdarzyć w dowolnym momencie. Jeśli awaria przydarzy się, kiedy baza nie pracuje (co jest mało prawdopodobne), ponadto nie dotyczy nośników, na których są zapisane nasze dane (co łącznie z poprzednim jest już całkiem nieprawdopodobne), to "nie ma sprawy". Na wypadek zniszczenia nośników z danymi, co pewien okres czasu dane kopiuje się na inne urządzenia tworząc tym samym możliwość ponownego wczytania na nośniki podstawowe stanu danych na dzień składowania. Istnieją systemy pozwalające zapisywać dane równocześnie w dwóch egzemplarzach bazy danych (takim jest IDMSX), wtedy w czasie awarii można przełączyć system na korzystanie z tych zapasowych danych a po usunięciu awarii przekopiować dane tworząc duplikat zapisów. Metoda ta jest jednak bardzo droga. Zazwyczaj wykonuje się kopie okresowo stosując do tego taśmy magnetyczne, które są znacznie tańsze niż dyski. Powstaje jednak problem z odzyskaniem aktualizacji wykonanych w czasie między składowaniami. W tym celu prowadzone są rejestracje wszystkich wykonanych transakcji. Każda wykonana transakcja jest zapisywana w specjalnie do tego celu przeznaczonym pliku. Oprogramowanie pomocnicze powinno umożliwić uzupełnienie danych z ostatniego składowania informacjami zawartymi w zbiorze transakcji. W ten sposób możemy zminimalizować straty. Duplikowanie zbiorów transakcyjnych pozwala zmniejszyć ryzyko straty tych danych, jest ono znacznie tańsze, niż duplikowanie całej bazy. Niestety nie jest to koniec kłopotów, jakie mogą nam się zdarzyć. Awaria może wystąpić w trakcie trwającego procesu aktualizacji. Aktualizacja nie jest pozaczasową zmianą stanu bazy, lecz zajmuje jakiś odcinek czasu, mówimy, że baza jest w stanie krytycznym. Jeśli awaria pojawi się, kiedy baza jest w stanie krytycznym, jedynym rozsądnym rozwiązaniem jest powrót do stanu przedkrytycznego. Można to wykonać, jeśli rejestrować będziemy stan przedaktualizacyjny każdego zapisu. Po wyjściu ze stanu krytycznego i zapisaniu kopii transakcji możemy skasować informacje o starych stanach rekordów (lub krotek).

Mimo najlepszych mechanizmów zabezpieczających dane może zdarzyć się, że nasza baza okaże się wewnętrznie niespójna (im więcej powiązań ich i kontroli, tym częściej może się to przydarzyć). Dobry system zarządzania bazą danych powinien zawierać oprogramowanie pozwalające wyłapać takie niezgodności (zerwanie powiązania, niekompletne indeksy itp.) i zlokalizować ich miejsce. Ponadto powinien dostarczać możliwości naprawy takich uszkodzeń bez odtwarzania danych ze zbiorów składowania (tego typu usterki wykrywane są najczęściej po bardzo długim czasie od pojawienia się ich i możemy już nie dysponować tak starymi zbiorami składowania lub odtworzenie wszystkich transakcji może być zbyt kosztowne). Wiąże się to również z dostarczeniem nam przez producenta obrazu fizycznego zapisu danych oraz programu monitorującego zapisy.

Wczytywanie dużych ilości danych

Korzystając z możliwości zapisu danych w sposób randomowy, sekwencyjny, wykorzystując indeksy hierarchiczne napotykamy na przykry efekt podczas pierwszego załadowania bazy danymi (i zawsze gdy potrzebujemy wprowadzić do bazy duże ilości danych). Ta nieprzyjemność, to bardzo długi czas realizacji tego zadania. Wynika to z faktu, że niemal każdy rekord wymaga transmisji danych do lub z urządzeń zewnętrznych. Można temu zaradzić, jeśli zna się fizyczne adresy każdego rekordu, wtedy odpowiednie przesortowanie danych uporządkuje je w kolejności zapisu na nośnikach, co powoduje wielokrotne skrócenie czasu wprowadzania danych. Systemy bogate w mechanizmy alokacji danych wymagają dużej wiedzy, aby taki proces samemu przeprowadzić. Producent powinien zatem dostarczyć odpowiedni pakiet oprogramowania realizujący tę funkcję. Nasze działanie w tym zakresie powinno być ograniczone do niezbędnego minimum.

Statystyki

Baza danych jest tworem złożonym, zmieniającym się w czasie. Zmiany te mogą wpływać na pogorszenie pracy systemu. Czasami mogą nawet zagrozić istnieniu bazy, może zdarzyć się to np. w przypadku przepełnienia obszarów. Aby móc czuwać nad całością bazy, niezbędne jest opro-

gramowanie pozwalające nam sprawdzić, jaki jest stopień zapełnienia fizycznych plików danymi i jaki jest ich rozkład. Producent powinien dostarczyć takie oprogramowanie i doradzić na jakie zjawiska powinniśmy zwracać szczególną uwagę.

Obok oprogramowania obrazującego stan bazy na dany moment, przydatne jest oprogramowanie zbierające statystyki pracy programów aplikacyjnych. Programy aplikacyjne zazwyczaj użytkowane są przez pracowników niewiele znających się na przetwarzaniu danych, ponadto jeśli program pracuje na rzecz wielu użytkowników równocześnie, to trudno ocenić jakość pracy tego programu. Może zdarzyć się, że program wydaje nadmierną ilość żądań do bazy danych, że nawigacja po bazie jest nieprzyzwoicie zła. Na takie okoliczności powinniśmy dysponować możliwością śledzenia zachowania się bazy danych w trakcie pracy programu lub pliku programów równolegle. Odpowiednie narzędzia powinien dostarczyć producent.

Innego rodzaju statystyki są związane głównie z bazami relacyjnymi. Istnieją różne możliwości wspomożenia procesu optymalizacji planu działania. Najprostszym jest rejestracja ilości krotek w poszczególnych tablicach. Pozwala to wybierać najmniej liczne krotki do pierwszych połączeń, co może istotnie zredukować czas przygotowania odpowiedzi. Bardziej wyrafinowane systemy rejestrują wartości ekstremalne w kolumnach a nawet tak jak INGRES rozkład poszczególnych wartości dla wartości rzadkich lub histogramy dla przedziałów wartości. Pozwala to optymalizować selekcje.

Restrukturyzacja

Każdej bazie może przydarzyć się nieszczęście kiedy to musi ulec zmianie. Może to być konieczność wprowadzenia dodatkowego atrybutu do krotki, podzielenia nie znormalizowanej krotki w celu doprowadzenia jej do postaci normalnej. Jeszcze trudniejsza sytuacja jest z bazami sieciowymi. Oprócz odpowiedników wyżej wymienionych zmian może zajść konieczność dobudowania setu. Proces ten nazywa się restrukturyzacją bazy danych. Jest on szczególnie trudny i czasochłonny w bazach sieciowych. Aby przeprowadzić go możliwie niskim kosztem, powinniśmy dysponować narzędziami specjalnie do tego celu zbudowanymi przez producenta bazy danych. Tylko on zna szczegóły implementacyjne i potrafi znaleźć najlepszy sposób wykonania tych standardowych przecież operacji. Istnieją rozwiązania, które pozwalają "wbudować" restrukturyzację do bazy. Polega to na deklaracji rekordów wariantowych w miejsce rekordów, które mogą ulec zmianie. Jeśli zaistnieje konieczność zmiany takiego rekordu, wystarczy dokonać odpowiednich zmian w schemacie i podschematach bazy danych natomiast zbędnym staje się fizyczne przebudowanie rekordów. Rozwiązanie to jest jednak bardzo drogie w eksploatacji. Wydaje mi się, że lepiej z niego nie korzystać.

Reorganizacja

Reorganizacja jest innego rodzaju zmianą bazy danych niż restrukturyzacja. Tutaj nie zachodzi potrzeba zmiany schematu bazy danych, lecz sprawa idzie o miejsce potrzebne do zapisania większej ilości danych, niż było to planowane na początku. Jeśli baza zbudowana jest jako kolekcja plików seryjnych, to sprawa jest prosta. Rozszerzenie pliku fizycznego pozwala na zapisanie dodatkowych danych. Często pliki seryjne są automatycznie zwiększane przez system operacyjny na żądanie systemu zarządzania bazą, wtedy problem znika całkowicie. Duże systemy zarządzania bazami oferują różne możliwości zapisu danych. Wiele z nich bazuje na metodzie obliczania fizycznego położenia zapisu w oparciu o wartość klucza i wielkość dostępnego obszaru pamięci. I tutaj sprawa się komplikuje. Po zmianie wielkości dostępnego obszaru zmieniają się fizyczne adresy wyliczone dla zadanego klucza, zatem wszystkie dane w pliku fizycznym zmieniają swe położenia. W bazie relacyjnej wystarczy wczytanie do nowego obszaru danych według nowych zasad. W bazie sieciowej może zaistnieć (najczęściej istnieje) sytuacja, gdy rekordy z tego obszaru są połączone z rekordami umieszczonymi w innych obszarach, wtedy należy zaktualizować odpowiednie odsyłacze. Łatwo sobie wyobrazić jak kosztowny jest ten proces. Znów pomoc producenta w postaci odpowiednich narzędzi jest niezbędna. Istnieje pewien bardzo sprytny sposób uniknięcia wielkich nakładów na taką reorganizację. Możemy po prostu zwiększyć stronę, wtedy zwiększamy ilość dostępnej pamięci nie zmieniając adresów już zapisanych rekordów. Strony jednak nie możemy zwiększyć ponad dopuszczalny przez system operacyjny limit. Jeśli system bazy danych oferuje dostatecznie szerokie możliwości alokacji danych, to możemy już w momencie budowania systemu zaplanować wzrost ilości danych i tak dokonać ich rozmieszczenia, aby reorganizacja nie była kosztowna. Warto mieć to na uwadze już przy zakupie systemu zarządzania bazą danych.

5. Krok dalej

Wdrożenie systemów baz danych do codziennego użytkowania wskazało na możliwość formalizacji i co za tym idzie automatyzacji pewnej części prac projektanckich. Rozszerzyło zakres zapisywanych w sposób formalny informacji o obiektach ze świata informatyki oraz doprowadziło do zastępowania w programach klasycznych instrukcji poleceniami realizującymi duże fragmenty przetwarzania. W tym rozdziale przytaczam kilka idei, które mają swoje praktyczne realizacje

na poparcie twierdzenia, że technologia baz danych jest jedną z ważniejszych dziedzin informatyki. Pozwala budować nie tylko systemy przeznaczone dla użytkowników ze sfery administracyjnej, co - jak odczuwam - niektórzy z tuzów informatyki uważają za zajęcie niegodne prawdziwego informatyka (jeśli ktoś nie wierzy, proszę przejrzeć programy studiów informatycznych, wynika z nich, że wszyscy informatycy powinni budować kompilatory, systemy operacyjne itp. - tylko po co?) lecz również w dziedzinie, która jest probierzem złożoności.

Słowniki danych

Każda baza danych musi posiadać kartotekę, w której zapisane są informacje o tym co zawiera, jak to jest zapisane na nośnikach itd. W bazach sieciowych kartoteka jest zapisana centralnie dla wszystkich zasobów. W bazach relacyjnych może to wyglądać różnie, ale i tu przeważają kartoteki cenralne. Skoro tak, to można pójść dalej i rozszerzyć kartotekę o informacje o całym systemie. Różnie można rozumieć pojęcie "cały system". Dla niektórych może to być specyfikacja programów oraz ich kody źródłowe, inni wymagają formalnej rejestracji wszystkiego co, się wokół systemu działo, dzieje i może dziać. W każdym przypadku prowadzi to do powstania bazy danych o systemie. Takie bazy wraz z oprogramowaniem nazywane są słownikami danych. Słowniki danych budowane są zazwyczaj w oparciu o bazę danych. Jeśli słownik ma wspomagać budowanie systemu w oparciu o konkretny system zarządzania bazą danych, to i słownik jest zbudowany w tym systemie. Im więcej informacji zapiszemy w słowniku, tym bardziej on może wspomagać nasze czynności przy budowie i pielęgnacji systemu. Zapisanie w słowniku modelu związków encji pozwala na automatyczne wygenerowanie schematów bazy danych, zapisanie pełnych definicji danych pozwala na przekopiowanie tych opisów do programów itd. Słownik pozwala szybko sprawdzić powiązania zapisanego w nim elementu z innymi elementami systemu. Ci, którzy wielokrotnie poszukiwali programów (i miejsc w programach), w których użyty został plik, rekord czy pole, wiedzą, ile to zajmuje czasu i jak łatwo można przeoczyć wiele miejsc. Słowniki posiadają własne języki, którymi użytkownicy (w tym przypadku są to informatycy) komunikują się. W słowniki wbudowywane są narzędzia wspomagające tworzenie obrazów ekranu wykorzystywanych w programach, czasami pracując w słowniku można uruchamiać kompilacje programów. Słownik jak każdy system baz danych powinien być wyposażony we wszystkie narzędzia, takie jak dla bazy danych. Specyficznym dla słowników jest wspomaganie utrzymania i ochrony kolejnych wersji systemu. Słownik może (często tak jest) zastępować kartotekę baz danych.

Generatory programów

Większość czynności, jakie wykonują klasyczne programy przetwarzania danych, związanych jest z czytaniem i dopasowywaniem danych do siebie. W systemach baz danych tymi problemami zajmuje się system zarządzania. Po zautomatyzowaniu wyszukiwania danych okazało się, że istnieje kilka grup programów, w ramach których są one do siebie bardzo podobne. Tak więc można wyróżnić programy sporządzania raportów, aktualizacji wsadowej i interaktywnej, interaktywnego przeglądania danych. Pozwala to na stworzenie szablonów tych programów, które uzupełnione przez użytkownika (programistę) mogą stanowić podstawę do wygenerowania tekstów źródłowych programów. W ten sposób otrzymaliśmy do dyspozycji nowe narzędzia, jakimi są generatory programów. Wprawdzie obecnie generatory szybko tracą swoją pozycję na korzyść języków czwartej generacji, ale wciąż są przydatnymi narzędziami.

Języki czwartej generacji

Trudno powiedzieć, że języki czwartej generacji stanowią następny krok w rozwoju generatorów, ponieważ języki zapytań dla relacyjnych baz danych nie miały swoich poprzedników w postaci generatorów a mają cechy tych języków. W innych przypadkach zdarzało się, że język taki powstał właśnie w wyniku ewolucji generatorów (np. 4GL firmy ICL). Główną cechą charakterystyczną języków czwartej generacji jest ich deklaratywność. Programista deklaruje co chce uzyskać, a zmartwieniem systemu jest jak to zrobić. Ponieważ zazwyczaj nie cały program jest jednostką deklaratywną (jak np. polecenie SELECT w SQL), istnieją w tych językach mechanizmy pozwalające określić kolejność wykonania poszczególnych poleceń. Polecenia te dotyczą zazwyczaj dużych jednostek semantycznych, takie czynności jak aktualizacja pól w rekordzie czy aktualizacja zmiennych programu, wyszukiwanie informacji w bazie, organizacja ekranu itp. są dla programisty niepodzielnymi jednostkami semantycznymi. Niemniej jednak możemy w językach czwartej generacji używać zmiennych, możemy program dzielić na podprogramy, używać procedur jak w językach takich jak COBOL, PASCAL i innych, którym od czasu pojawienia się czwartej generacji przypisuje się generację trzecią. Języki te stają się coraz doskonalsze i wbrew temu co twierdzą niektórzy, uważam, że w przetwarzaniu danych dekada lat dziewięćdziesiątych nie będzie należała do języków obiektowych (być może później), lecz do języków czwartej generacji.

CASE

CASE jest skrótem pochodzącym od angielskiego określenia Computer Aided Software (System) Engineering, co tłumaczymy jako komputerowe wspomaganie inżynierii oprogramowania lub komputerowe wspomaganie budowania systemów (informatycznych). Pojęcie to określa całość narzędzi komputerowych, których zadaniem jest pomóc w pracy informatykowi. Narzędzia nierozerwalnie związane są z metodami pracy, dlatego ja głosuję za tym, aby mówiąc o CASE myśleć również o metodach. Przejdźmy jednak "do rzeczy". Jakie by nie były metody, narzędzia wspomagające je budowane są w technologii baz danych. Sercem zintegrowanego pakietu CASE jest słownik danych. Daje on możliwość rejestracji poszczególnych obiektów systemu oraz związków zachodzących pomiędzy nimi. Następne co do ważności i użyteczności są generatory pozwalające na podstawie specyfikacji systemu zbudować bazę danych (o ile metoda wymaga, aby systemy były realizowane w tej technologii) oraz języki czwartej generacji pozwalające na szybkie i łatwe zbudowanie oprogramowania. W niektórych przypadkach oprogramowanie tak powstałe może być traktowane jako prototypowe, istnieją języki czwartej generacji przeznaczone do budowania oprogramowania eksploatacyjnego. Czasami taki pakiet zawiera oprogramowanie pozwalające wygenerować prototyp systemu całkowicie automatycznie. Inne elementy pakietów CASE bardziej uprzyjemniają życie informatykowi, niż są niezbędne, są jednak najbardziej widoczne i mogą być zwodnicze jak śpiew syren. Do tej grupy zaliczam wspomaganie prac przy pomocy menu i innych zaproszeń. Najbardziej efektywną grupą są interfejsy graficzne. Obraz niesie znacznie więcej informacji niż słowo, jest więc wspaniałym narzędziem, ale trzeba, aby w tle były równie dobre narzędzia.

Zbigniew BENEDYKT

Co to jest IDMSX?

System Zarządzania Zintegrowaną Bazą Danych IDMS (Integrated Database Management System) został napisany przez BF Goodrich Company of Acorn, Ohio w 1971 roku. Podstawą projektu był raport opracowany przez Data Base Task Group komitetu Codasyl wydany w kwietniu tegoż roku. W 1973 roku prawa rynkowe i rozwoju sprzedano Cullinance Corporation of Boston, Massachusetts. Od tej firmy w 1975 roku prawa wyłączności do IDMS nabyła brytyjska firma ICL. W 1978 r. ICL zaimplementował podzbiór cobolowych mechanizmów wspomagających zintegrowaną bazę danych zaproponowanych przez Codasyl Programing Languages Committee oraz Data Description Languages Committee. Komitety te opracowały standardy języków do definiowania zawartości i organizacji bazy danych oraz standardowe mechanizmy języka COBOL do manipulowania tymi danymi. Rozwiązania te implementowane były na wielu komputerach w środowiskach wielu systemów operacyjnych. ICL wciąż doskonalił swoją bazę wydając kolejne wersje: IDMS 320 w 1982 r., IDMS 350 w 1983, IDMS 400 w 1984, IDMSX 510 w 1989 roku. Wersja IDMSX jest daleko idącym rozszerzeniem wersji poprzednich. Tę wersję przedkładam uwadze Czytelników.

Pracuje ona na komputerach firmy ICL serii 2900 i 39 pod systemem VME. Może być eksploatowana w trybie wielodostępu.

Język manipulowania danymi

DML (Data Manipulation Language) nie jest samodzielnym językiem programowania lecz zestawem zdań uzupełniających COBOL. Składnia ENVIRONMENT DIVISION i DATA DIVISION COBOL-u została rozszerzona tak, aby umożliwić import do programu zapisów z kartoteki bazy danych oraz obszarów roboczych pełniących rolę interfejsu między systemem zarządzania a programem. PROCEDURE DIVISION został wzbogacony o polecenia otwierania i zamykania fragmentów bazy używanych przez program, definiowania krytycznych fragmentów przetwarzania (takich partii programu, które mogą powodować naruszenie stanu zgodności semantycznej bazy) oraz polecenia realizujące funkcje wyszukiwania i uaktualniania zapisów. Otwierając bazę informujemy równocześnie system o sposo-

bie ochrony danych przed kolizjami spowodowanymi dostępem do danych przez wiele programów równocześnie. Do wyboru mamy tryb używania: wyłącznego, zabezpieczonego (przed równoczesną wielokrotną aktualizacją) oraz nie zabezpieczonego. Tryby te łączą się z operacjami wyszukiwania lub/i aktualizacji danych. Krytyczne fragmenty programu oznacza się podając jego początek oraz koniec (taki fragment nazywany jest Success Unit - SU). System zarządzania prowadzi rejestr zakończonych transakcji wpisując doń pełne SU oraz otwarte SU. Dzięki temu po awarii potrafi odtworzyć stan bazy do ostatniego zakończonego SU oraz kontrolować - w czasie otwierania danych - czy poprzednie użycie bazy nie pozostawiło nie zakończonych SU. Pozostałe polecenia dotyczą takich funkcji jak: wyszukiwanie, czytanie, kasowanie, zakładanie, modyfikowanie rekordu oraz przyłączanie rekordu do setu i odłączanie rekordu od setu. Istnieją odpowiednie wersje polecenia FIND (szukaj rekordu) dla poszczególnych sposobów dostępu do zapisów. Przygotowany program tłumaczony jest na postać źródłową COBOL-u przy pomocy preprocesora bazy danych. Preprocesor wkopiowuje do programu żądane fragmenty kartoteki oraz zamienia polecenia DML na wywołania odpowiednich podprogramów. Następnie program jest kompilowany standardowym kompilatorem COBOLu.

Opis danych

Zgodnie z Raportami Codasylu opis zawartości i organizacji danych realizowany jest w standardowych jednostkach zwanych schematami. IDMSX używa trzech schematów oraz jednostki zwanej serwisem. Są to: schemat bazy danych (SCH - Schema), schemat przydziału zasobów (SSC, Storage Schema), podschemat (SUB, Subschema) oraz serwis bazy danych (DSE, Data Base Service).

Schemat bazy danych (SCH).

Globalny opis danych zawiera się w schemacie bazy danych (SCH), specyfikuje się w nim nazwy i struktury rekordów. Ponadto rekord może posiadać klucze. Klucz stanowi nazwane pole lub grupa pól (niekoniecznie występujących kolejno po sobie), które identyfikuje(ą) rekord. Jeden rekord może posiadać wiele kluczy, z których każdy może być uporządkowany lub nie. Pierwszy klucz nieuporządkowany służy do wyliczenia adresu strony, w której rekord powinien być umiejscowiony. Następne klucze, dla których nie zadeklaruje się porządku, stanowią tzw. klucze alternatywne. Powstaną z nich zapisy w bazie danych składające się z ich wartości oraz adresu, pod którym właściwy rekord został zapisany. Klucze z zadeklarowanym porządkiem rosnącym lub malejącym utworzą w bazie struktury hierarchiczne pozwalające odszukiwać rekordy wg zadanego porządku. Również kluczy porządkowych może być wiele dla jednego rekordu. Żaden z kluczy nie musi zawierać wyłącznie wartości unikatowych dla danego typu rekordu. Dobrze jest jednak, aby klucz główny spełniał warunek niepowtarzalności wartości kluczowej. Jeśli to zadeklarujemy, system nie dopuści do zapisania rekordu o wartości klucza obecnej już w bazie.

Rekordy mogą być stałej lub zmiennej długości, mogą zawierać pola grupowe i powtarzane. Typy pól są zgodne z COBOLem, nie nakłada się na nie żadnych dodatkowych ograniczeń poza cobolowymi.

W schemacie bazy deklaruje się również wszystkie sety. Dla każdego setu należy podać nazwę właściciela i nazwy członków setu. Jeden set może mieć członków wielu typów. Związek pomiędzy właścicielem i członkami precyzowany jest przy pomocy kodów: wprowadzania i utrzymywania. Kod wprowadzania może przyjmować wartość Automatic lub Manual. Kod utrzymania - Mandatory lub Optional. Wszystkie kombinacje tych wartości są dopuszczalne.

Schemat przydziału zasobów (SSC)

SSC zawiera opis sposobu, w jaki dane będą umieszczone na nośnikach fizycznych, ale nie zajmuje się przydziałem tych nośników. Całość opisu dotyczy struktury logicznej. Jedna baza danych może posiadać wiele SSC. Obiektem najbardziej zbliżonym do nośników fizycznych jest plik logiczny. Dane bazy mogą być umiejscowione w jednym lub wielu plikach logicznych. Plik logiczny może mieć zadeklarowaną wielkość bloku, wtedy plik fizyczny, który z nim skojarzymy, powinien mieć tę samą wielkość bloku. Wolno nam jednak nie deklarować wielkości bloku, wtedy wielkość zostanie dopasowana do bloku fizycznego w momencie przydzielania plików fizycznych (praktycznie wielkość bloku może wynosić od 512 b do 32 Kb). W tym przypadku mówimy, że plik ma blok zmiennej długości. Podobnie można postąpić z wielkością pliku, otrzymamy wtedy plik o zmiennej wielkości.

W pliki odwzorowywane są obszary. Obszar może być odwzorowany w jeden lub wiele plików. Również w jeden plik może być odwzorowanych wiele obszarów. Praktycznie jest jednak stosować odwzorowanie jeden do jeden. Obszar jest najmniejszą jednostką dla takich operacji jak składowanie, reorganizacja, odłączanie i przyłączanie do bazy.

Obszary dzielone są na strony, strona odpowiada jednemu blokowi pliku logicznego. Każda strona ma unikatowy w całej bazie numer z przedziału 1001 do 223-1 (pierwszy tysiąc numerów zarezerwowany jest dla kartoteki bazy danych) Strona podzielona jest na linie. Linia odpowiada jednemu zapisowi. Może ich być w stronie 256

(0 do 255), przy czym linia zerowa każdej strony jest zarezerwowana na cele organizacyjne. Strony mogą być zapisane w jednym z dwóch formatów (dla całego obszaru format strony musi być jednakowy). Pierwszy format jest standardowym formatem bazy danych. Narzędzia pozwalające na ingerencję w stronie przedstawiają ją w formacie standardowym niezależnie od tego jaki format naprawdę jest użyty. Drugi (używany częściej) pozwala na korzystanie z urządzenia zwanego CAFS (Contained Addressable File Store), odciążającego procesor komputera od seryjnego przeszukiwania danych spełniających zadany warunek logiczny (pamięć asocjacyjna).

Dla każdego typu rekordu w bazie musimy podać gdzie i jak ma być lokowany. Miejsce lokacji specyfikuje się podając nazwę obszaru lub obszarów, w których rekord może zostać zapisany. Ponadto w każdym obszarze możemy określić przedział stron dozwolony dla danego typu rekordu.

IDMSX daje do wyboru siedem sposobów alokacji rekordów: CALC, RANDOM, VIA, SYSTEM DERFAULT, PAGE DIRECT, DIRECT, SEQUENTIAL.

CALC - na podstawie wartości klucza rekordu oraz deklaracji dotyczących umiejscowienia, system wylicza numer strony gdzie rekord powinien być zapisany. Wszystkie rekordy o wspólnej stronie przeznaczenia łączone są łańcuchem (do następnego i do poprzedniego) zaczynającym się w linii zerowej strony. Zatem rekord niekoniecznie musi zostać zapisany w wyliczonej stronie, aby system mógł go odnaleźć znając wartość klucza. Deklarując ten sposób alokacji rekordów dla kluczy, których wartości nie muszą być unikatowe, możemy wybrać kolejność zapisywania rekordów. Do wyboru mamy: ostatni wprowadzany - ostatni w kolejności, ostatni wprowadzany - pierwszy w kolejności, wg decyzji systemu.

RANDOM - tutaj mamy dwie możliwości. Możliwość pierwsza, to randomizacja na podstawie zadanego klucza. Sposób ten podobny jest do rozmieszczenia CALC, lecz zapisy nie są łączone łańcuchami. Nie można zatem odnaleźć rekordu na podstawie wartości klucza. Możliwość druga - RANDOM bez klucza - umieszczenie zapisu w dowolnej stronie z dozwolonego dla typu rekordu zakresu.

VIA - jeśli rekord jest członkiem setu, to możemy żądać, aby stroną przeznaczenia była ta sama strona, w której zapisany jest właściciel setu.

SYSTEM DEFAULT - aktualnie taka deklaracja powoduje umieszczenie RANDOM bez klucza.

PAGE DIRECT - sposób analogiczny do CALC z tą różnicą, że algorytm randomizacji dostarcza program użytkownika.

DIRECT - stroną przeznaczenia jest strona wyliczona w oparciu o algorytm dostarczony przez program użytkownika, jednak rekordy o tej samej stronie przeznaczenia nie zostaną połączone łańcuchami. W zamian system podaje programowi użytkownika adres, pod którym rekord został zapisany (numer strony, numer linii).

SEQUENTIAL - system stara się zapisać rekordy tak, aby ich fizyczne umieszczenie odpowiadało porządkowi podanego klucza. Jednak w przypadku gdy jest to niemożliwe, system nie porządkuje zapisów, nie wykorzystuje również żadnych mechanizmów pośredniego porządkowania rekordów. Praktycznie, należy posortować rekordy przed wprowadzeniem ich do bazy. Późniejsze aktualizacje mogą zburzyć porządek zapisów. Z rekordami związane są klucze. Jeśli w SCH zadeklarowaliśmy klucze alternatywne (bezpośrednie bądź sortowania), to w SSC mamy możliwość zapisania swoich żądań dotyczących ich alokacji. Dla kluczy sortowania budowane są odpowiednie rekordy indeksowe tworzące drzewa. System może dokonać pełnej specyfikacji tych indeksów, możemy jednak dokonać kilku modyfikacji. W szczególności możemy zmienić miejsce alokacji tych zapisów oraz ilość kluczy zapisanych w jednym rekordzie indeksowym. Analogicznie bezpośrednie klucze alternatywne mogą być wyspecyfikowane przez system. Zostaną wtedy umiejscowione w obszarach, które zostały przydzielone dla odpowiedniego typu rekordu, jeśli chcemy to zmienić, to możemy.

W IDMSX możemy deklarować dwa sposoby implementacji setów.

Pierwszy - zwykły set łańcuchowy. Jego implementacja jest taka jak przedstawiłem to omawiając raport Codasylu (rozdział czwarty). Jeśli zrezygnujemy ze specyfikowania setu w SSC, to system uzna, że życzymy sobie, aby set był łańcuchowym, właściciel posiadał odsyłacze do pierwszego oraz do ostatniego członka setu, każdy członek posiadał odsyłacze do następnika, poprzednika i do właściciela. Ponadto każdy nowy członek setu będzie ostatnim w secie. Ustalenia te wolno nam zmienić. Możemy zrezygnować z odsyłacza do następnika lub do poprzednika (nie z obu). Możemy zdecydować się na inny sposób dopisywania rekordów (na początek łańcucha, po bieżącym rekordzie setu lub przed nim).

Drugi - jako set sortowany. W takiej organizacji pomiędzy rekordem właściciela i członkami setu umieszczana jest struktura indeksowa (jak dla indeksów hierarchicznych). Właściciel posiada odsyłacze do początku i do końca tej struktury, każdy członek posiada jeden tylko odsyłacz - do właściciela. W ten sposób otrzymujemy możliwość utrzymywania w bazie setów z upo-

rządkowanym wg zadanego klucza dostępem do jego członków. Dla setu wolno nam zdefiniować tylko jeden klucz. Możemy żądać, aby wartości klucza w ramach setu nie powtarzały się. Jeśli dopuszczamy możliwość wielokrotnego wystąpienia tej samej wartości klucza w secie, możemy ustalić następstwo zapisów w ramach jednej wartości.

Podschematy bazy danych (SUB)

W podschemacie musimy poinformować system, jakie rekordy i jakie sety dopuszczamy do użytkowania przez programy, jakie obszary i w jakim trybie może program otworzyć. Ponadto dla każdego rekordu specyfikujemy klucze dostępne programowi (niekoniecznie musimy pozwolić na wykorzystanie wszystkich zdefiniowanych kluczy). Możemy również zakazać używania dowolnego polecenia DML dla dowolnego z rekordów i setów. W podschemacie możemy zdefiniować zakresy (Realms). Realm może być równy obszarowi lub typowi rekordu. Żaden rekord nie może występować więcej niż w jednym zakresie. Ze względu na ograniczenie w języku 4GL lepiej nie korzystać z definiowania zakresów odpowiadających typowi rekordu i przyjąć równość zakresów i obszarów.

Wolno nam zapisać dowolną ilość podschematów.

Serwis bazy danych (DSE)

Istnieją informacje, które musimy przekazać systemowi zarządzania bazą a nie znalazły się one w schematach. W tym celu w IDMSX wprowadzono pojęcie serwisu. Serwis jest modułem podlegającym kompilacji, przechowywanym w bibliotece programów binarnych. Przed uruchomieniem jakiegokolwiek programu pracującego z bazą musimy uruchomić serwis. Serwis może być dzielony lub nie dzielony. Nie dzielony serwis pozwala na pracę tylko jednego programu użytkownika w danym czasie, zajmuje się tylko ochroną danych na wypadek awarii. Serwis dzielony zezwala na równoczesną pracę wielu programów. Oprócz ochrony danych przed awariami wykonuje operacje blokowania i zwalniania zasobów pozwalające na bezpieczną pracę wielu użytkowników. Zakres i sposób wykonywania tych czynności wybieramy spośród oferowanych przez producenta opcji. Ochronę bazy przed awariami możemy realizować kilkoma sposobami. Najprostszy i najtańszy z nich to rezygnacja z jakiejkolwiek ochrony. Nie jest to tak głupie, jak się na początku wydaje. System operacyjny może chronić bazę danych w dostępny dlań sposób. W VME można składować okresowo wszystkie lub wybrane pliki, można również dowolny plik zapisać w dwóch egzemplarzach i zażądać, aby system równolegle dokonywał aktualizacji w oryginale i kopii (pliki dupleksowane). Okresowe składowania w przypadku wyłącznie partiowego przetwarzania są całkiem dobrym rozwiązaniem. W przypadku awarii w trakcie pracy programu możemy odzyskać wyskładowane dane i przebieg powtórzyć. Dupleksowanie plików przez system operacyjny nie wydaje mi się rozwiązaniem rozsądnym. Dla przetwarzania wsadowego metoda ta jest zbyt droga. W przypadku przetwarzania interaktywnego nieskuteczna, ponieważ po awarii baza może pozostać wewnętrznie niezgodna. Wróćmy więc do możliwości IDMSX w tym zakresie. System może - w przypadku serwisu dzielonego jest to wymagane - prowadzić żurnal. Żurnal jest plikiem lub plikami przeznaczonymi do rejestracji każdej aktualizacji. Logicznie podzielony jest na dwie części. W jednej przechowywane są informacje o dokonanych zmianach bazy (after image), w drugiej informacje o stanie rekordów przed aktualizacją (before image). Obraz zmian dokonanych, służy do odtworzenia stanu danych do momentu ostatniego poprawnie zakończonego SU. Obraz danych przed aktualizacją pozwala wycofać wszystkie te zmiany, które mogłyby powodować wewnętrzne niezgodności danych w wyniku nie zakończonego SU. Obie te części możemy przechowywać w jednym pliku, żurnal taki nazywamy centralnym. Możemy części te rozdzielić. Ponadto możemy rejestrować wykonane aktualizacje osobno dla poszczególnych obszarów. Wszystkie te pliki mogą być dupleksowane (przez IDMSX). W ramach zarządzania wielodostępem, możemy wybrać pomiędzy blokowaniem obszarów a blokowaniem stron. Do dyspozycji mamy również mechanizm blokowania opóźnionego, który zmniejsza ilość rygorystycznie zablokowanych rekordów (szczegóły tego rozwiązania wymagają dokładniejszego omówienia całego mechanizmu blokowania). W DSE deklarujemy ponadto maksymalny czas oczekiwania przez program na dostęp do danych oraz maksymalną ilość programów pracujących równocześnie. Niektóre z deklaracji mogą być zmieniane w czasie uruchamiania serwisu. Tylko jeden serwis może być uruchomiony w danym czasie. Możemy jednak przygotować wiele serwisów na różne okazje.

Metody zapisu schematów

Oprócz języka manipulowania danymi DML podkomitety Codasylu do spraw języków opracowały języki: opisu schematu i podschematów DDL (Data Description Language) i opisu schematu przydziału zasobów DSDL (Data Storage Description Language). Ponadto dla opisu serwisów ICL przygotował język SDL (Service Description Language). Składnia tych języków przypomina cobolowe konstrukcje DATA DIVISION. Ich sztywny format jest odrażający. IDMSX posiada odpowiednie kompilatory tych języków i jeśli

kto lubi, to może swoje bazy opisywać przy ich pomocy. Jednak znacznie wygodniej możemy uczynić to w języku DDCL (Data Dictionary Control Language), którym posługujemy się dokonując zapisów w słowniku danych.

Kartoteka bazy danych

W kartotece bazy danych zapisujemy schemat, podschematy i schematy przydziału zasobów. Możemy uczynić do kompilując odpowiednie zapisy DDL i DSDL lub na podstawie zapisów w słowniku danych. System używa kartoteki do przechowywania zmiennych przebiegu programów (statystyki, informacje o otwartych SU itp.). Ponadto w kartotece zapisane są procedury pozwalające na odczyt danych z bazy bez potrzeby pisania programu tzw. CLUC utility. Preprocesor bazy danych na podstawie zapisów w kartotece koryguje program DMLowy doprowadzając go do postaci źródłowej COBOL-u. ICL oferuje swoim klientom język 4GL (o nim później). Przyjęte rozwiązanie eliminuje potrzebę stosowania kartoteki. Jak dotychczas nie wszystkie problemy można rozwiązać bez odwoływania się do COBOLu więc kartoteka jest jeszcze czasami potrzebna. Firma ogłosiła, że w niedługim czasie wyeliminuje całkowicie potrzebę tworzenia kartoteki również dla programów DML-owych.

Słownik danych

Z IDMSX współpracuje system słownika danych (DDS - Data Dictionary System). ICLowski DDS jest potężnym narzędziem wspomagającym cykl życia systemu zaczynając od jego specyfikacji. Jest systemem wielodostępnym zaimplementowanym w IDMSX. Dowolny zapis w nim może być zablokowany dla współużytkowników poprzez mechanizm ochrony własności. Możemy rejestrować wersje budowanego systemu, a użytkownicy posiadający dodatkowe opcje mogą wykorzystywać jeden słownik dla budowania wielu systemów.

Podstawową jednostką semantyczną słownika jest element. Element może należeć do jednego ze 107 typów predefiniowanych przez producenta. Pośród nich znajduje się kilka typów czysto administracyjnych (CLASSKEY, AUTHORITY itp.), pozostałe można podzielić na cztery kategorie. Dwie z nich dotyczą świata rzeczywistego i specyfikują dane oraz procesy, dwie to konstrukcja systemu podzielona na sfery danych i procesów. DDS przystosowany jest do budowania systemów metodą SSADM, stąd odpowiednie typy elementów (encja, akcja, atrybut, zdarzenie, efekt itd.). Do sfery danych systemu należą takie elementy jak: rekord, set, schemat, database itd. Jak widać, DDS jest przystosowany również do przyjęcia specyfikacji bazy danych. W wersji 800 można specyfikować IDMSX i pliki niezależne. Aktualnie dostępna jest wersja, w której można specyfikować również bazę relacyjną INGRES. Zapis w DDS wystarcza do wygenerowania bazy danych z pełnym zakresem możliwości. Jeśli nie potrzebujemy kartoteki, to w wyniku kompilacji podschematu i schematu przydziału zasobów otrzymujemy pliki binarne (tzn. tablice podschematu), które współpracują z kodami programów (napisanych w 4GL i odpowiednio skompilowanych) w trakcie ich realizacji. Jeśli musimy posiadać kartotekę, to możemy ją wygenerować z DDS. Praca z DDS jest znacznie łatwiejsza niż z kartoteką, ponadto mamy możliwość uzyskania informacji o związkach pomiędzy poszczególnymi elementami słownika. Możemy na przykład sprawdzić, w jakich programach (i miejscach programu) odwołujemy się do danego rekordu, pola itd.

Istnieje wiele opcji słownika ułatwiających i uprzyjemniających życie informatykowi. Dotyczą one nie tylko bazy danych, cała dokumentacja systemu może być zapisana w DDS. Szkoda tylko, że w DDS nie możemy skorzystać z graficznych metod komunikowania się z systemem. Wszystkie struktury przechowywane są tylko w postaci opisowej. ICL oferuje narzędzie graficzne wspomagające pracę analityka i projektanta sprzężone z DDSem. Jest to jednak temat na „całkiem inne opowiadanie".

Słownik danych można dopasować do indywidualnych potrzeb definiując nowe typy elementów. Jednym z predefiniowanych typów jest typ ELEMENT pozwalający na definiowanie przez użytkownika własnych typów (opcja za dodatkową opłatą).

Ochrona danych przed nielegalnym dostępem

Istnieją trzy podstawowe drogi dostępu do danych przez użytkownika. Pierwsza, to praca interaktywna z wykorzystaniem przygotowanych przez dział przetwarzania danych programów. Programy te pozwalają jedynie na dostęp do zasobów na przygotowanych formatkach ekranów, użytkownik nie ma możliwości oglądania ani aktualizacji danych nie udostępnionych przez program. Programy takie dostępne są tylko na z góry ustalonych terminalach na podstawie ich fizycznego adresu. Projektant może wbudować opcje sprawdzające identyfikatory i hasła użytkowników oraz pozwolić użytkownikom zmieniać swoje hasła, kiedykolwiek zechcą. Inną metodą używania bazy danych, to korzystanie z programów wsadowych. Aby uruchomić taki program, użytkownik musi zgłosić się do systemu operacyjnego. System posiada katalog użytkowników oraz rejestruje ich hasła i pozwala przyłączyć się tylko znanemu użytkownikowi po sprawdzeniu jego hasła. Trzecią drogą dostępu do danych (tylko do czytania) jest QUERYMASTER.

Jest to system, przy pomocy którego użytkownik może kierować zapytania do bazy w języku podobnym do SQL (system ten prezentuję na dalszych stronach). Dział przetwarzania danych przygotowując ten system dla użytkownika może ograniczyć dostęp do danych poprzez odpowiednie dopasowanie podschematu bazy danych oraz poprzez ukrycie dowolnego rekordu, pola lub setu wchodzącego w skład podschematu. QUERY-MASTER posiada możliwość definiowania własnych użytkowników oraz ich haseł.

Pliki IDMSX mają specyficzny status w systemie operacyjnym, który nie pozwala na jakikolwiek dostęp do ich zawartości narzędziami ogólnego przeznaczenia (przeglądanie, edycja, sortowanie plików). Ponadto definiując dowolne pole w DDS możemy przypisać mu procedury czytania i zapisywania. Pozwala to na szyfrowanie danych.

Ochrona danych przed zniszczeniem zapisów

Warianty ochrony danych przedstawiłem opisując specyfikację serwisu bazy danych. Warto podkreślić jedynie, że w IDMSX możemy całkowicie zrezygnować z mechanizmów ochrony danych przed awarią, co podnosi efektywność przetwarzania (programy wsadowe). Można jednak tak ustawić poziom ochrony, aby zapewnić nieprzerwaną pracę przez 24 godziny na dobę i siedem dni w tygodniu (oczywiście to kosztuje). To są skrajności, zazwyczaj wykorzystywane są stopnie pośrednie.

Wielodostęp

Serwis bazy danych możemy uruchomić jako nie dzielony lub dzielony. Serwis nie dzielony pozwala na pracę tylko jednego programu w danym czasie, dzielony organizuje pracę wieloprogramową. Istnieje jeszcze inny sposób udostępnienia danych wielu użytkownikom na raz. Program użytkownika może współpracować z wieloma terminalami równocześnie, sam organizując wielodostęp odciąża serwis bazy danych. To rozwiązanie stosowane jest do przetwarzania transakcyjnego. ICL dostarcza system TPMS (TRANSACTION PROCESSING MANAGEMENT SYSTEM), który wspomaga tworzenie i eksploatację takiego oprogramowania.

Narzędzia wspomagające budowanie systemów

QUICKBUILD PATHWAY

QUICKBUILD PATHWAY (QBP) jest oprogramowaniem typu CASE. Nie jest on składnikiem systemu IDMSX, ale ponieważ w dużym stopniu wspomaga prace związane z tworzeniem i oprogramowaniem bazy danych, postanowiłem włączyć krótką informację o nim do prezentacji IDMSX. QBP w swej większej części jest nakładką na DDS pozwalającą na wprowadzanie doń informacji w sposób sterowany przez strukturę menu. Nie wszystko, co można zapisać w słowniku, da się wprowadzić w oparciu o zapytania QBP (można wtedy skorzystać z wywołania DDS z poziomu QBP). Z kolei QBP wspomaga takie czynności jak: składowanie i odzyskiwanie danych, restrukturyzację bazy, kreowanie fizycznych plików i ich inicjację itp., a więc czynności, które ze słownikiem danych nie mają nic wspólnego. Ponadto QBP ma wbudowane dwa niezmiernie pomocne generatory.

Proces budowania systemu nie zaczyna się od definiowania bazy danych. Zanim dojdziemy do tego etapu, musimy wykonać szereg prac analitycznych. W ich wyniku posiadamy specyfikację danych interesującego nas fragmentu rzeczywistości. Jeśli jest to zapis w jakimś stopniu sformalizowany, to możemy ustalić reguły jego przekształcenia na schemat bazy danych. W DDSie jest miejsce na specyfikację modelu związków encji. Jeśli skorzystamy z tej możliwości oraz posiadamy pakiet oprogramowania nazwany przez ICL QUICKBUILD PATHWAY, to możemy schemat bazy danych wygenerować wprost ze związków encji. Oszczędność pracy jest ogromna. Tak otrzymany schemat niekoniecznie musi zadowalać projektanta, może on go skorygować, lecz jest to znacznie mniej pracy niż pisanie od początku.

Często potrzebna nam jest mała baza danych dla celów testowania. W tym przypadku niepotrzebne są nam wyszukane sposoby alokacji danych. Taką bazę w całości możemy wygenerować w QBP. Rozdzielenie schematu konceptualnego bazy i schematu przydziału zasobów powoduje, że możemy tych samych programów używać niezależnie od rozlokowania danych rzeczywistych. Stąd normalną jest praktyka posiadania bazy testowej przez cały okres eksploatacji systemu. Pozwala to w każdej chwili testować na tej bazie zmieniane oraz nowo tworzone programy. Program dający poprawne wyniki na jednej bazie daje również poprawne wyniki na drugiej.

Drugim niezmiernie pomocnym generatorem jest generator programów aplikacyjnych. Przez program aplikacyjny rozumie się w systemie ICL program aktualizacji i/lub przeglądania danych pracujący pod kontrolą TPMSu. Jeśli potrafimy nasz program aktualizacji podzielić na moduły: zakładania, aktualizacji (w tym kasowania) i przeglądania rekordów oraz analogiczne operacje dla setów, to wygenerowanie takiego programu możemy zlecić systemowi. W wyniku otrzymamy program w pełni użytkowy z „przyjaźnie" zaprojektowanymi ekranami. Jego pierwsze menu będzie stanowić lista operacji, dla jakich poleciliśmy wygenerować program.

Dla rekordów wprowadzanych do bazy program żąda wprowadzenia wartości pól rekordu

(nazwy pojawiające się na ekranie przeniesione są ze słownika) oraz identyfikatorów właścicieli wszystkich setów, których rekord jest członkiem. Użytkownik zobowiązany jest do wypełnienia wartości pól kluczowych oraz identyfikatorów właścicieli setów z automatycznym kodem wprowadzania.

Aktualizacja rekordów odbywa się w następujący sposób: użytkownik proszony jest o podanie identyfikatora rekordu. Jeśli system znajdzie żądany rekord, to wyświetla wartości jego wszystkich pól, wartości kluczy właścicieli setów, do których rekord należy i pytanie czy skasować rekord. Użytkownik nie może zmienić jedynie klucza rekordu oraz kluczy właścicieli setów, które posiadają obowiązkowy kod utrzymania rekordów. Każde inne pole można zmienić, program potrafi przełączyć rekord do innego setu. Możemy zażądać skasowania rekordu, wtedy rekord zostanie skasowany wraz z wszystkimi obowiązkowymi członkami oraz konsekwencjami kasowania tych członków.

Przeglądanie rekordów wygląda podobnie do aktualizacji, lecz nie można dokonać żadnych zmian w zapisach danych. Zakładanie setu polega na założeniu jego właściciela (nie może być wcześniej wprowadzony do bazy), a następnie zakładaniu ciągu jego członków.

W procesie aktualizacji setu możemy skasować właścicieli setu (z konsekwencjami), zmienić wartości pól właściciela, kasować dowolnego członka (z konsekwencjami), aktualizować członków oraz dopisać nowych członków do setu. Warto zauważyć, że w ten sposób możemy aktualizować rekordy nie posiadające klucza. Przeglądanie daje pełny obraz setu bez możliwości aktualizacji. Wynikiem generowania programu aplikacyjnego są kody źródłowe dla wszystkich operacji (w 4GL), formatki ekranów oraz odpowiednie produkty w kodzie wykonywalnym. Kody źródłowe mogą być podstawą do dalszej rozbudowy programów. Oprócz wyżej przedstawionych generatorów QBP ma wiele mechanizmów mniej lub bardziej związanych z IDMSX. Niektóre z nich mogą być kupione i używane jako niezależne produkty, inne mają mniejsze znaczenie dla samego IDMSX (a to jest przedmiotem tego opracowania).

Po rocznym dzień w dzień obcowaniu z IDMSX i jego środowiskiem, mogę śmiało stwierdzić, że QBP jest bardzo pożytecznym narzędziem pozwalającym zaoszczędzić wiele czasu i wysiłku. Mimo wielu ograniczeń, jakie posiada, wart jest pieniędzy na zakup. Uważam, że komuś kto ma niewielkie doświadczenie w pracy z tak potężną bazą sieciową jak IDMSX, jest wręcz niezbędny.

4GL

Język czwartej generacji proponowany przez ICL nie posiada nazwy własnej, dlatego symbol „4GL" przyjmowany jest jako nazwa tego języka. Język ten powstał w drodze ewolucji generatorów programów raportowania, aktualizacji wsadowej oraz aktualizacji interaktywnej. Ślady tego widoczne są chociażby w postaci nazw elementów DDS służących do ich zapisu. ICLowski 4GL jest językiem strukturalnym, posiada możliwość definiowania modułów i ma cechy pozwalające zaliczyć go do języków deklaratywnych. Program napisany w 4GL jest zbiorem elementów słownika danych. Elementy te to: BATCH-PROGRAM, REPORT-PROGRAM I APPLICATION (segmenty główne programów: aktualizacji wsadowej, wydruków i aktualizacji interaktywnej) oraz DIALOGUE, EXCHANGE, COMPONENT, MODULE (segmenty tworzące strukturę programów). Elementy DIALOGUE i EXCHANGE są elementami ściśle wyspecjalizowanymi i występują jedynie w programach aktualizacji interaktywnej. Jeśliby szukać odniesienia do języków trzeciej generacji, to można stwierdzić, że element jest odpowiednikiem bloku algolowskiego. Można w nim definiować zmienne lokalne z zasięgiem dostępu tak jak w Alogolu, wywoływać podporządkowane elementy (każdy jest zapisany osobno w DDS) oraz deklarować wykonanie specyficznych dla danego elementu funkcji. Na przykład element EXCHANGE realizuje komunikację z terminalem, wyszukiwanie danych w bazie oraz aktualizacje danych. Aby uściślić żądanie, programista deklaruje nazwę formatu ekranu przy pomocy atrybutu *DST, niezbędne rekordy przy pomocy atrybutów *SVI i ewentualnie *SEL oraz sposób i zakres aktualizacji przy pomocy atrybutu *CHA. Wyobraźmy sobie, że chcemy dopisać kolejną pozycję zamówienia. Identyfikatorem zamówienia jest NR-ZAM, którego wartość oraz wartość całego wiersza zamówienia użytkownik wprowadza na ekranie EK1. Odnośny fragment programu wygląda następująco:

```
* DST SCR EK1
* SVI ZAMÓWIENIE START
* SEL ZAMÓWIENIE WHERE MATCHING NR-ZAM
* CHA WIERSZ-ZAMÓWIENIA CREATED.
```

Zapis ten spowoduje wysłanie na terminal formatki ekranu EK1, po wypełnieniu jej przez użytkownika przesłanie wartości pól wypełnionych do odpowiednich zmiennych systemu, wyszukanie rekordu zamówienia z zadaną wartością pola NR-ZAM, dopisanie do setu nowego rekordu WIERSZ-ZAMÓWIENIA. Sądzę, że każdy kto musiał pisać programy dla tego typu aktualizacji, doceni 4GL. 4GL ma pewne ograniczenia, np. nie można w nim pracować z tablicami oraz polami zmiennej długości, wtedy można skorzystać z języków niższego poziomu (np. COBOLu, FORTRANu) i włączyć odpowiednie segmenty. Do tego celu przeznaczo-

ny jest element MODULE. ICL dostarcza również zestaw modułów standardowych realizujących najczęściej używane operacje. Szybkość i łatwość pisania programów w 4GL jest tak duża, że rekompensuje dodatkowe koszty poniesione z powodu ewentualnej nieoptymalności kodu wynikowego. Ostatecznie w szczególnych przypadkach program taki można traktować jako prototypowy i po akceptacji rozwiązań przez użytkownika, napisać go w języku trzeciej generacji lub nawet w asemblerze.

QUERYMASTER

QUERYMASTER (QM) jest systemem, który w oparciu o opis danych podobny do podschematu potrafi odpowiadać na pytania użytkownika. Mogą to być pytania formułowane ad hoc lub przygotowane w postaci makr, być może sparametryzowanych. Język zapytań jest podobny do SQL lecz, ponieważ system posiada informację o setach, jest od niego prostszy. ICL oferuje również opcję pozwalającą formułować zapytania w stylu QUERY BY FORMS. QUERYMASTER nie ogranicza się tylko do danych IDMSX, do opisu można dołączyć również informacje o plikach niezależnych, można również zdefiniować tzw. miękkie sety (logiczne połączenia danych spełniających zadany warunek). Opis danych dla QM (QUERY VIEW) można wygenerować na podstawie podschematu w QBP. Opis taki można następnie skorygować maskując pewne informacje oraz wprowadzając dodatkowe i wprowadzić użytkownikom QM.

Narzędzia wspomagające okres eksploatacji

Najważniejszymi narzędziami wspomagającymi administrowanie bazą danych są bez wątpienia programy składowania danych i ich odzyskiwania po ewentualnej awarii. Oczywiście IDMSX takie oprogramowanie posiada, jest ono bardzo sprawne i wygodne w użyciu. Możliwość składowania i odzyskiwania danych w czasie pracy bazy danych jest wielkim udogodnieniem.

Wprowadzanie dużych ilości danych

Nieprzyjemnym momentem w „życiu" bazy sieciowej jest wprowadzanie do niej dużej ilości danych (pierwsze wypełnianie bazy). Pakiet DB_LOAD wspomaga ten proces. Wymaga on wprawdzie napisania cobolowego programu przygotowującego dane wejściowe, co wydaje się zajęciem przykrym, kiedy pod ręką ma się 4GL, ale dalszy proces przebiega bardzo sprawnie dając kilkunastokrotne przyspieszenie w stosunku do zwykłego programu aktualizacji.

Restrukturyzacja i reorganizacja

Innym nieprzyjemnym zdarzeniem jest konieczność zmiany struktury danych lub zmiany ich alokacji. W niektórych przypadkach restrukturyzacja bazy danych może być przeprowadzona przy pomocy QBP. Wtedy wszystkie nasze "wyczyny" kontrolowane są przez system, który ogranicza możliwość popełnienia błędu. QBP pozwala zmienić strukturę bazy poprzez dopisanie rekordów i setów, dopisanie pola do końca rekordu oraz zmianę typu pola. Jest to zatem dość szeroki zakres możliwości. Poważniejszym problemem jest wielkość bazy. Po prostu QBP deklaruje zbyt małe pliki pośrednie, jakie potrzebne są w trakcie restrukturyzacji. Zatem najlepiej przeprowadzić próbną restrukturyzację na bazie testowej przy pomocy QBP a następnie, po sprawdzeniu wyników, powtórzyć przy pomocy oprogramowania specjalnie do tego celu przeznaczonego. Warto tu zwrócić uwagę, że pierwszą fazą restrukturyzacji jest przebudowa schematów bazy danych, a ta jest wspólna dla baz testowej i produkcyjnej.

Reorganizacja bazy danych nie wymaga zmian w schemacie konceptualnym, ale może wymagać zmian w schemacie przydziału zasobów i odpowiedniego przemieszczenia danych. Chodzi o to, aby to „może" jak najrzadziej przeradzało się w „musi". IDMSX i w tym względzie ma swoje zasługi. Mechanizmy zmiennej strony i zmiennego obszaru pozwalają minimalizować konieczność wykonywania reorganizacji. Jeśli jednak ona zaistnieje, to odpowiedni pakiet oprogramowania pozwala ją przeprowadzić w miarę bezpiecznie i szybko (chociaż dla dużych baz danych może to trwać wiele godzin).

Porządkowanie indeksów

W IDMSX możemy korzystać z dobrodziejstwa indeksów hierarchicznych. W bazie zapisywane są one jako drzewa. Niestety drzewa te mają brzydką skłonność do nierównomiernego wydłużania gałęzi. Prowadzi to do zmniejszenia efektywności przeszukiwania danych za ich pomocą. Od czasu do czasu należy zreorganizować indeksy tak, aby drzewo było zrównoważone. IDMSX posiada do tego celu odpowiedni program (IDMSX_INDEX_TIDY).

Statystyki

Aby nie być zaskakiwanym takimi zdarzeniami jak niemożność wprowadzenia zapisu danych do bazy z powodu braku miejsca, czy gwałtownym spadkiem efektywności programów, administrator potrzebuje odpowiednich informacji. W IDMSX, każdorazowo przy składowaniu bazy można otrzymać raport o stopniu zapełnienia poszczególnych obszarów oraz stopniu wypełnienia stron. Dokładny obraz wypełnienia bazy wraz

z odpowiednimi histogramami otrzymuje się przy pomocy programów specjalnie do tego celu przygotowanych. Można z nich dowiedzieć się właściwie wszystkiego o rozlokowaniu danych w bazie. Innym rodzajem statystyk są statystyki przebiegu programu oraz pracy serwisu. Można otrzymać je w trakcie pracy serwisu dzielonego. System zarządzania bazą podaje ilości żądań, czytań i zapisów stron dla każdego z obszarów, ilości ustawianych blokad, zakleszczeń itp. Podobne informacje podawane są dla każdego programu z osobna.

Kontrola integralności danych

Normalna, codzienna eksploatacja bazy danych nie powinna powodować sytuacji, które mogłyby doprowadzić do syntaktycznej niepoprawności bazy. Nawet w przypadku awarii lub brutalnego przerwania programu automatycznie działające mechanizmy przywrócą bazę do stanu przed awarią. Może się jednak zdarzyć, że coś będzie nie w porządku. Procedura IDMSX_CHECK_INTEGRITY kontroluje stan bazy danych. Sprawdza poprawność zapisów administracyjnych (prawdziwość zapisów na stronie, ilość wolnego miejsca, poprawność linii zerowej, zgodność typów rekordów), poprawność rekordów (ilość odsyłaczy, wskaźniki rekordów zmiennej długości i segmentacji) oraz w oddzielnej fazie kompletność i prawdziwość wszystkich połączeń. Procedurę tę można uruchomić przy pomocy QBP lub wywołując ją bezpośrednio. Jeśli baza ma jakieś usterki, to otrzymamy informację o ich rodzaju i numer strony (ewentualnie linii), na której wystąpiły.

Dostęp do zapisów fizycznych

Korekta zapisów polegająca na poprawianiu bajtów na stronie jest sprawą bardzo niebezpieczną. Jednak ICL dostarcza narzędzia pozwalające przeprowadzić taką operację. Są to dwa programy. Pierwszy IDMSX_PRINT, który drukuje stronę bajt po bajcie. Drugi IDMSX_PFIX, to edytor pozwalający poprawić dowolny bajt strony. Program drukujący prezentuje stronę w postaci hexadecymalnej, znakowej oraz rekord po rekordzie z rozkodowanymi odsyłaczami. Pozwala to na łatwe odnalezienie poszukiwanego zapisu, to z kolei pozwala łatwo ustalić odpowiedni bajt na ekranie monitora w programie edycji.

Podsumowanie

IDMSX ma już za sobą dwudziestoletni okres rozwoju. Jest więc bazą „dojrzałą" i to widać w trakcie pracy. Przygotowana jest na wszelkie okazje. Posiada nowoczesne środowisko wspomagające budowanie systemów przy jej użyciu, bardzo dobre zabezpieczenia danych i środowisko diagnostyczne. W IDMSX zrealizowany jest system ubezpieczeń społecznych dla Wielkiej Brytanii, dla mnie jest to bardzo dobra rekomendacja.

CHARAKTERYSTYKA SYSTEMU INFORMIX Z UWZGLĘDNIENIEM ARCHITEKTURY PROGRAMOWEJ JEGO PAKIETÓW

mgr
Danuta
Kosmowska-Miszalska

mgr inż.
Stanisław
Wrona

Filia Nr 2
Wojskowego
Instytutu Informatyki
Warszawa

1. WSTĘP

System zarządzania relacyjną bazą danych INFORMIX pojawił się na rynku informatycznym w 1981 r. jako produkt firmy Relational Database Systems; wiosną 1982 r. firma oferowała wersję 2.00 tego systemu. Niemal jednocześnie pierwsze wersje systemu INFORMIX zostały opracowane i rozpowszechniane przez firmę Informix Software. Do chwili obecnej powstało wiele kolejnych wersji w coraz bogatszej, stale modyfikowanej konfiguracji, składającej się z różnych pakietów. Wiosną 1990 r. firma Informix Software udostępniła wersję 4.0 systemu INFORMIX, w postaci 18-pakietowej rodziny, dającej możliwość wyboru wielu różnych konfiguracji użytkowych.

Pakiety systemu INFORMIX dostępne są (w większości) pod systemami operacyjnymi UNIX/XENIX/MS-DOS/OS-2/VMS.

W rodzinie INFORMIX istnieją pakiety umożliwiające pracę w środowisku sieci lokalnych ze standardowymi protokołami (TCP/IP, Novell NetWare itp.).

Najważniejsze etapy w historii firmy INFORMIX:

- 1981 rok - pierwszy komercyjny system zarządzania relacyjną bazą danych;
- 1985 rok - pierwszy system zarządzania relacyjną bazą danych pod MS-DOS, pracujący w sieciach lokalnych;
- 1986 rok - pierwszy język czwartej generacji dla środowiska systemu operacyjnego UNIX;
- 1987 rok - pierwszy system zarządzania relacyjną bazą danych, przetwarzający transakcje na bieżąco (OLTP);
- 1988 rok - pierwszy system zarządzania relacyjną bazą danych z SQL, pracujący pod systemem operacyjnym OS-2;
- 1989 rok - pierwszy graficzny arkusz kalkulacyjny WINGZ.

Powyższe wydarzenia pozwoliły firmie INFORMIX zająć pierwszą pozycję wśród producentów systemów zarządzania relacyjnymi bazami danych, działającymi pod systemem UNIX (około 450 tys. instalacji). Obecnie INFORMIX zajmuje drugą pozycję wśród producentów oprogramowania dla systemu operacyjnego UNIX, za firmą AT&T. Firma INFORMIX zatrudnia ponad 1100 osób, a jej obroty w roku 1989 osiągnęły 145 milionów dolarów USA. Do największych jej klientów należą następujący producenci sprzętu komputerowego: AT&T, IBM, MIPS, Motorola, NCR, NEC Corp., Olivetti, Pyramid, Siemens, Sequent, UNISYS oraz następujące korporacje międzynarodowe: Boeing, BBC, BP, British Airports Authority, PHL, Ford Motor Corp., General Motors, British Telcom, McDonald's Corp., Middland Bank, British Royal Navy.

2. ARCHITEKTURA PAKIETÓW SYSTEMU INFORMIX

System INFORMIX składa się z szeregu w miarę niezależnych pakietów programowych. Wybór określonej ich konfiguracji powinien zależeć od potrzeb użytkowych.

Podstawę naszych doświadczeń (pod systemem operacyjnym SCO XENIX System V) stanowi edycja systemu INFORMIX z marca 1989 r., do której należą pakiety:

INFORMIX-4GL wersja 1.10.03K
INFORMIX-4GL Rapid Development System wersja 1.10.03K
INFORMIX-4GL Interactive Debugger wersja 1.10.03K
INFORMIX-SQL wersja 2.10.03K
INFORMIX-TURBO wersja 1.10.03K
INFORMIX-ESQL/C wersja 2.10.03K
INFORMIX-ESQL/COBOL wersja 2.10.03K
INFORMIX-ESQL/ADA wersja 2.10.03K

Oprócz powyższych pakietów systemu INFORMIX dystrybuowany jest w tej chwili pakiet C-ISAM wersja 3.10.03K.

Przewijające się w następnych punktach spostrzeżenia implementacyjne dotyczą wymienionych wersji i odnoszą się głównie do pakietów INFORMIX-SQL oraz INFORMIX-4GL.

Pakiety INFORMIX-4GL,
INFORMIX-4GL Rapid Development System,
INFORMIX-SQL

oraz wszystkie pakiety INFORMIX-ESQL/* i programy użytkowe utworzone przy ich użyciu realizują dostęp do bazy danych za pośrednictwem identycznego interpretera zdań języka SQL. Interpreter ten, zwany maszyną bazy danych, jest technologicznie zrealizowany jako samodzielny program w systemie operacyjnym.

Omówimy krótko każdy z wymienionych pakietów.

Pakiet INFORMIX-SQL zawiera moduł obsługi interakcyjnej pracy z bazą danych za pośrednictwem zdań języka SQL. Zdania te mogą być pisane wprost przez użytkownika i wykonywane na jego polecenie na bieżąco. Można jednak uruchomić z poziomu systemu operacyjnego program isql, podając jako parametr tego programu nazwę pliku tekstowego, zawierającego zdania języka SQL.

Pakiet INFORMIX-SQL zawiera ponadto uniwersalny interpreter formularzowego przetwarzania bazy danych, wraz z prostym generatorem i kompilatorem formularzy oraz kompilator i interpreter raportów, pozwalający tworzyć dowolnie zredagowane wydruki danych, pobieranych z bazy danych standardowymi zdaniami SELECT. W pakiecie tym istnieje możliwość budowy własnego menu użytkowego, co technologicznie realizowane jest poprzez odpowiednie zapisy w specjalnych tablicach w bazie danych, utworzonej przez użytkownika.

Reasumując, jest to uniwersalny pakiet, który może być używany nawet wprost do interakcyjnego tworzenia i przetwarzania bazy danych za pośrednictwem specjalnego menu systemowego. W konsekwencji pakiet ten jest najmniej elastyczny jako narzędzie do budowy systemów użytkowych, co może uzasadniać potrzebę użycia innych pakietów.

INFORMIX-4GL (tzw. wersja kompilacyjna), jest pakietem przeznaczonym do budowania systemów użytkowych w oparciu o specjalizowany język programowania, zaliczany do czwartej generacji (skrót 4GL jest przyjęty jako nazwa własna języka). Pakiet zawiera kompilator języka 4GL i daje możliwość tworzenia programów wielomodułowych oraz konsolidacji modułów pośrednich z innych języków programowania.

Technologicznie proces kompilacji przebiega przez dwie fazy prekompilacji - poprzez język ESQL/C (pośredni język Embedded SQL) oraz język C, z możliwością ingerencji programowej w każdą z tych faz.

INFORMIX-4GL Rapid Development System jest najnowszej generacji firmowym pakietem programowym, który umożliwia przetwarzanie bazy danych za pośrednictwem języka 4GL w odmiennej technologii niż wersja kompilacyjna. Programy są kompilowane do pseudokodu i wykonywane przez interpreter pseudokodu. Dodatkowym narzędziem, związanym z tym pakietem, jest INFORMIX-4GL Interactive Debugger, ułatwiający śledzenie, uruchamianie i testowanie programów.

Odmienna technologia (w stosunku do wersji kompilowanej) sprawia, że kompilacja programu do postaci gotowej do interpretacji przebiega wielokrotnie szybciej, powstałe programy są dużo mniejsze, a czas realizacji tych programów nieistotnie różni się od programów utworzonych wersją kompilowaną. Istnieje możliwość budowania specjalizowanych wersji interpretera pseudokodu poprzez rozszerzanie go o własne procedury, pisane w języku C. Praktycznie więc systemy użytkowe w języku 4GL, w każdej z dwóch omówionych technologii, mogą być tworzone z modułów, pisanych w języku 4GL i z procedur, pisanych w języku C.

Powyższe uwagi wskazują na pewną przewagę Rapida w stosunku do wersji kompilowanej.

INFORMIX-ESQL/C jest pakietem realizującym zanurzenie języka SQL w języku C. Umożliwia z poziomu programów w języku C współpracę z bazą danych przy użyciu języka SQL, a jednocześnie pozwala rozszerzać standardowy interpreter formularzy i raportów o własne funkcje w języku C.

INFORMIX-ESQL/COBOL (lub ADA) przeznaczone są dla programistów w tych językach i umożliwiają współpracę z bazą danych przy użyciu języka SQL, z poziomu programów w językach COBOL lub ADA.

INFORMIX-TURBO jest to szybka maszyna bazy danych. Współpracuje bezpośrednio z

urządzeniami wejścia-wyjścia, wykorzystując specjalizowany mechanizm wymiany stron we własnym buforze we-wy, kosztem zmniejszenia puli buforów we-wy systemu operacyjnego. Przeznaczona jest dla specjalizowanych zastosowań, o dużej liczbie użytkowników i dużej intensywności transakcji.

Pakiet C-ISAM (Indexed Sequential Access Method) jest to biblioteka funkcji w języku C, przeznaczonych do tworzenia i manipulacji na plikach indeksowych o tzw. strukturze C-ISAM. System INFORMIX bazuje na tej organizacji plików.

3. ORGANIZACJA PLIKÓW, METODY DOSTĘPU, OGRANICZENIA

Podstawową cechą metody organizacji plików C-ISAM jest tworzenie dla każdej tablicy pary plików - z danymi oraz indeksowego. Cecha ta uniemożliwia grupowanie w jednym pliku np. niewielkich słowników lub danych powiązanych logicznie. Ponadto pliki z danymi i indeksami umieszczane są zawsze parami we wskazanej kartotece, bez możliwości fizycznego ich rozdzielenia.

Implementacja tablic w metodzie C-ISAM opiera się na rekordach stałej długości. Istnieje możliwość tworzenia praktycznie dowolnej liczby indeksów, unikalnych lub nie. Właściwości te przejmuje INFORMIX. Z poziomu INFORMIXa indeks może być złożony maksymalnie z 8 pól, dowolnych typów, położonych niekoniecznie spójnie, o łącznej długości do 120 bajtów. Można wymusić fizyczne uporządkowanie pliku z danymi według wartości kluczy zadanego indeksu (tworząc indeks typu CLUSTER).

W pliku z indeksami, w każdym rekordzie indeksu, pamiętany jest wskaźnik do odpowiadającego mu rekordu z danymi. Struktura indeksów w metodzie C-ISAM bazuje na organizacji drzewa *B+*. Istnieje program usługowy bcheck, umożliwiający (poza sprawdzaniem poprawności struktury indeksów) ustalanie i zmiany wielkości porcji indeksującej, jako parametru wpływającego na efektywność organizacji zależnie od specyfiki danych. C-ISAM umożliwia kompresję wartości kluczy w indeksach. Istnieje możliwość ustawienia dowolnej kombinacji flag dla kompresji klucza w zakresie:

- powtarzających się początkowych znaków wartości klucza,
- końcowych spacji w wartości klucza,
- duplikatów wartości klucza w indeksie.

Kompresja może przynieść znaczne zyski w wielkości pliku z indeksami, oczywiście kosztem czasu dostępu. Globalny bilans zysków i strat zależy od specyfiki wartości kluczowych i długości indeksu. Zdaniem autorów metody C-ISAM progową długością indeksu, powyżej której można rozważać efektywność kompresji, jest 8 bajtów. Z poziomu INFORMIXa kompresja (wszystkich typów łącznie), dla indeksów o długości przekraczającej 8 bajtów, jest dokonywana automatycznie. Własnej ingerencji, modyfikującej rozwiązanie w konkretnym przypadku, można dokonywać jedynie z poziomu C-ISAM.

Z poziomu INFORMIXa indeksy można tworzyć interakcyjnie (tylko proste - bazujące na jednej kolumnie) lub programowo. Ze względu na składniową niezależność zdań SQLa od faktu istnienia indeksów można je tworzyć w dowolnym momencie. W procesach wyszukiwania w przypadkach łączenia tablic lub użycia frazy GROUP BY, INFORMIX może tworzyć na czas wykonywania zdań SQLa indeksy robocze.

Ogólnie INFORMIX daje możliwość dostępu do danych trzema metodami:

- sekwencyjnie,
- indeksowo-sekwencyjnie,
- bezpośrednio poprzez numer rekordu.

Dostęp bezpośredni realizuje się poprzez numer rekordu w pliku C-ISAM, używając formalnie kolumny o systemowo nadanej nazwie *rowid.* Stanowi ona systemowy identyfikator wierszy danej tablicy. Wartość rowid wierszy tablicy ulega zmianie wraz ze zmianą fizycznego uporządkowania rekordów w pliku.

C-ISAM wykorzystuje ponownie pamięć po skasowanych rekordach. Nowym rekordom INFORMIX nadaje wówczas odzyskane wartości rowid. Kasowanie rekordów nie powoduje zmniejszania pliku, tzn. C-ISAM nie zwalnia pamięci. Kasowane rekordy są zaznaczane flagami. Sekwencyjne przeglądanie pliku po flagach skasowanych rekordów w przypadku małego wypełnienia pliku trwa długo, szybszy jest wówczas dostęp indeksowy, bo tylko do istniejących rekordów. C-ISAM daje możliwość wyboru sposobu dostępu. Niestety INFORMIX nie korzysta z tej możliwości - zdanie SELECT bez fraz WHERE lub ORDER zawsze powoduje przeglądanie sekwencyjne.

4. MECHANIZMY OCHRONY PRZED NIEUPRAWNIONYM DOSTĘPEM

Bazową dla naszych doświadczeń konfiguracją jest system INFORMIX pod sterowaniem systemu operacyjnego XENIX. W tym przypadku w celu uzyskania określonego dostępu do obiektu należy pokonać dwie bariery ochrony.

Pierwszy próg to przyłączenie się do systemu XENIX, tzn. podanie nazwy użytkownika i hasła.

XENIX pozwala różnicować prawa dostępu do pliku z tytułu praw przypisanych właścicielowi, z tytułu praw przypisanych dla grupy (jeśli użytkownik należy do grupy przypisanej do pliku i nie jest jego właścicielem) lub z tytułu praw przypisanych dla pozostałych użytkowników (jeśli nie jest właścicielem pliku ani członkiem grupy). Superużytkownik (*root*) posiada wszelkie prawa dostępu do plików wszystkich użytkowników systemu.

Identyfikatorem użytkownika w systemie INFORMIX jest jego nazwa przyjęta w systemie XENIX.

Dla obiektów programowych INFORMIXa, a więc formularzy, programów SQL, raportów INFORMIXa-SQL, modułów programowych 4GL system INFORMIX nie daje żadnych mechanizmów ochrony. Obiekty te tworzone są w dowolnych kartotekach, z identyfikatorami grup właściwymi dla danych użytkowników i w stosunku do tych plików stosowane są mechanizmy ochrony systemu operacyjnego XENIX.

Wszystkie tablice bazy danych, tworzone przy użyciu INFORMIXa, są ewidencjonowane w systemie XENIX jako pliki, przypisane do grupy *informix* z prawem odczytu i zapisu dla użytkowników tej grupy. Każdy użytkownik bazy danych (bez względu na to, do jakiej należy grupy) widziany jest przez system operacyjny poprzez pakiet INFORMIX jako członek grupy *informix* (na czas realizacji danego procesu). Z tego względu, ze strony systemu XENIX, nie ma żadnych barier dla ochrony danych w bazie danych podczas pracy poprzez INFORMIX. W tym przypadku jedyną barierą są mechanizmy sterujące prawami dostępu w systemie INFORMIX.

Z poziomu systemu operacyjnego dostęp do plików z bazą danych ma tylko superużytkownik (*root*) oraz właściciel bazy danych, któremu ten dostęp jest potrzebny do działań organizacyjno-administracyjnych. Z punktu widzenia systemu XENIX, dostęp mieliby także członkowie grupy *informix*, ale tacy użytkownicy nie powinni być definiowani. Bowiem z poziomu systemu operacyjnego dostęp do bazy danych dla użytkowników innych niż administrator-właściciel nie jest pożądany, a z poziomu INFORMIXa, co już zostało powiedziane, są oni traktowani i tak jako członkowie grupy *informix*.

W systemie INFORMIX właściciel obiektu może wykonywać dowolne operacje na własnych obiektach oraz wybrane operacje w zakresie nadanych mu uprawnień na innych obiektach. Uprawnień dostępu do obiektu może udzielać właściciel obiektu lub użytkownik, któremu nadano takie uprawnienia (rekurencyjnie pochodzące od właściciela). Każdy użytkownik ma określone prawa dostępu z tytułu udzielenia tych praw jemu indywidualnie lub z tytułu przyznania określonych uprawnień wszystkim użytkownikom (tzn. użytkownikowi *PUBLIC*). Użytkownik korzysta z sumy logicznej uprawnień dostępu do obiektu - własnych i użytkownika PUBLIC. Przynależność użytkownika do grupy (w sensie systemu XENIX) nie ma na tym poziomie znaczenia. INFORMIX ignoruje również pojęcie superużytkownika, co należy mu poczytywać raczej jako zaletę. Traktowanie użytkownika *root* pod systemem INFORMIX jako zwykłego użytkownika jest zgodne z logiką traktowania superużytkownika jako administratora systemu operacyjnego, a właścicieli baz danych - jako administratorów tych baz.

Użytkownikom baz danych w systemie INFORMIX nadaje się uprawnienia dwojakiego typu: do przetwarzania bazy danych jako całości oraz do przetwarzania poszczególnych tablic lub ich fragmentów.

Dla przetwarzania bazy danych jako całości istnieją trzy poziomy uprawnień:

- zezwolenie na przetwarzanie tablic bazy danych w zakresie ograniczonym uprawnieniami do określonych tablic,
- przywilej poprzedni, ponadto zezwolenie na tworzenie nowych tablic bazowych,
- przywilej poprzedni, ponadto uprawnienia modyfikacji struktury bazy danych (bez względu na uprawnienia przetwarzania tablic).

Dla przetwarzania tablic bazy danych istnieje sześć rodzajów uprawnień: do odczytu określonych danych, aktualizacji, wprowadzania wierszy, kasowania, tworzenia nowych indeksów oraz modyfikacji struktury tablicy.

System INFORMIX umożliwia również selektywne ustalanie dostępu do wybranych kolumn tablicy. Ponadto przy użyciu tablic wirtualnych umożliwia ustalenie selektywnego dostępu jedynie do wyspecyfikowanych wierszy tablicy, a nawet dostęp jedynie do danych zbiorczych z tablicy.

5. MECHANIZMY OBSŁUGI WIELODOSTĘPU (ZAKRES I SPOSÓB BLOKOWANIA DANYCH)

System INFORMIX umożliwia blokowanie dostępu do danych na poziomie rekordów i tablic lub na poziomie całej bazy danych.

Inicjacja - z opcją *EXCLUSIVE* - pracy z bazą danych powoduje, aż do zamknięcia bazy, możliwość dostępu na wyłączność tylko danemu użytkownikowi.

Blokowanie danych na poziomie rekordów i tablic realizuje się w różny sposób, zależnie od stanu bazy danych (transakcyjna lub nie). Podczas przetwarzania nietransakcyjnej bazy danych następuje automatyczna blokada i odblokowywanie poszczególnych rekordów. Blokada i zwalnianie całych tablic odbywa się poprzez jawne użycie

komend *LOCK* i *UNLOCK*. Podczas przetwarzania transakcyjnej bazy danych w przypadku realizacji pojedynczych komend następuje automatyczna blokada (stopniowo) wszystkich modyfikowanych rekordów, aż do zakończenia realizacji komendy. Podczas realizacji komend wewnątrz bloku transakcyjnego wszystkie blokady są utrzymywane do momentu zakończenia transakcji.

System operacyjny nakłada ograniczenia na liczbę jednocześnie utrzymywanych blokad. Z tego względu w wielu przypadkach wskazane jest blokowanie całych tablic. Niestety w systemie INFORMIX decyzję tę zawsze musi podejmować użytkownik, chociaż wydawałoby się celowym istnienie mechanizmu warunkowego automatycznego przejścia systemu w tryb blokowania całej tablicy, w momencie wyczerpania limitu pojedynczych blokad.

W przypadku bloku transakcyjnego komenda blokady tablicy powinna występować po komendzie rozpoczynającej transakcję. Wewnątrz bloku transakcyjnego nie można tablicy odblokować (komendą *UNLOCK*), zwolnienie dostępu następuje w momencie zakończenia transakcji.

Ogólnie blokada dostępu do obiektów może być w trybie dzielonym (*SHARE*) lub na wyłączność (*EXCLUSIVE*). W przypadku blokowania tablic komendą *LOCK* specyfikuje się jawnie tryb blokady. Blokady automatyczne system nakłada w różnym trybie, zależnie od rodzaju operacji. Na przykład tworzenie indeksu odbywa się pod blokadą na wyłączność, a aktualizacja rekordów - w trybie dzielonym.

System INFORMIX realizuje procesy aktualizacji bazy danych bez tworzenia kopii aktualizowanego fragmentu bazy. Taka technologia zwiększa czas trwania blokad, ponadto stwarza możliwość istnienia sytuacji, w których jeden użytkownik wycofuje się z transakcji, a w tym momencie drugi wyciąga wnioski z udostępnionych mu do odczytu, przechwyconych - chwilowo zaktualizowanych danych. Jest to dodatkowy argument przemawiający za blokowaniem w pewnych sytuacjach całych tablic, co stwarza możliwość blokady na wyłączność.

Nie istnieje w systemie INFORMIX mechanizm blokowania dostępu do odczytu spójnych logicznie danych, tzn. nie ma językowej konstrukcji dla wyrażenia życzenia typu: chcę czytać dane, o ile nikt inny ich w tej chwili nie aktualizuje i nikt tego nie może uczynić. Dla uzyskania takiego zabezpieczenia trzeba wykonać "na wyrost" blokadę jak do aktualizacji (w trybie dzielonym), co z kolei ogranicza i wprowadza w błąd innych użytkowników.

Powyższe uwagi dotyczyły problemów blokad dostępu przy zastosowaniu mechanizmów językowych. Sprawa wygląda inaczej w przypadku aktualizacji danych standardowym programem poprzez formularz pakietu INFORMIX-SQL. Proces taki system realizuje jako ciąg kolejnych zdań aktualizacji, odnoszących się do pojedynczych rekordów.

Reakcje systemu INFORMIX w sytuacjach prób dostępu do zablokowanych danych pozostawiają wiele do życzenia. Można co prawda ustawić programowo tryb oczekiwania na zwolnienie dostępu do danych, ale jest to oczekiwanie bezwarunkowe i oczywiście grozi wówczas zakleszczenie (*deadlock*) w przypadku niepoprawnego zakończenia procesu blokującego dane. W trybie sygnalizacji błędu (bez oczekiwania na zwolnienie blokady) systemowa obsługa sprowadza się do zakończenia programu z sygnalizacją błędu. Obsługę błędów, dotyczących zdarzeń zablokowania danych, należy oprogramowywać indywidualnie dla każdego wybranego zdania, bez możliwości wydania, we wspólnej procedurze obsługi, komendy „powtórz próbę wykonania zdania". Ponadto nie ma wyodrębnionej grupy błędów, związanych ze współbieżnym dostępem.

6. JĘZYK ZAPYTAŃ

Wszystkie odwołania do bazy danych, zarówno z programów użytkowych, jak i systemowych, realizowane są za pośrednictwem języka zapytań klasy SQL. Język ten w systemie INFORMIX w zasadzie w pełni akceptuje standard ANSI. Posiada on jednak wiele zdań, które stanowią rozszerzenie tego standardu. Wśród dodatkowych zdań można wyróżnić takie, które służą do ręcznego sterowania przez programistę akcjami inicjowania transakcji, otwierania i zamykania bazy danych, blokowania tablic, inicjowania plików dla kronikowania zmian w bazie danych. Zdania te stanowią narzędzia zastępcze, udostępnione zamiast nie realizowanych automatycznie przez system odpowiadających im funkcji.

Drugą grupę roszczeń poza standard ANSI stanowią zdania, funkcje i pewne dodatkowe frazy zdań standardowych, które poprawiają walory użytkowe systemu. Do tych rozszerzeń można zaliczyć:

- dodatkowe typy danych,
- funkcje operowania na danych typu data i czas,
- tworzenie indeksów,
- zdania importu i eksportu danych z bazy danych do plików ASCII.

W pakietach INFORMIX-SQL oraz INFORMIX-4GL można wyróżnić dodatkowy język zapytań typu Query By Example, dostępny w procesie formularzowego przetwarzania bazy danych. Język ten służy do definiowania prostych kryteriów wyszukiwania danych z tablic bazy danych. W pakiecie INFORMIX-SQL język ten jest jedynym narzędziem definiowania kryteriów wyszukiwania w programie, realizującym formu-

larzowe przetwarzanie bazy danych.

W pakiecie INFORMIX-4GL dostępne jest zdanie, które tworzy odpowiedni tekst frazy WHERE zdania SELECT, w oparciu o interakcyjnie wprowadzone kryterium wyrażone w języku Query By Example. W efekcie odwołanie do bazy danych formułowane jest w języku SQL.

7. DODATKOWE JĘZYKI I GENERATORY

System INFORMIX oferuje wysokopoziomowy język programowania 4GL, język definiowania formularzy oraz język tworzenia raportów.

4GL jest specjalizowanym językiem programowania czwartej generacji, ukierunkowanym na przetwarzanie danych. Zawiera szereg nieproceduralnych konstrukcji, ułatwiających tworzenie menu użytkowych, współpracę z formularzami (również tablicowymi), generowanie raportów.

W języku 4GL zanurzony jest język SQL, z możliwością dynamicznej redakcji zdań dostępu do danych podczas wykonywania programu. Ważną konstrukcją są tzw. kursory - narzędzie służące do organizacji wielowierszowego przetwarzania danych wybranych z bazy lub do ich wprowadzenia. Istnieje możliwość deklarowania zmiennych programowych typu „...LIKE tablica_bd.pole", przez co uzyskuje się niezależność programu od zmian struktury używanych tablic. Język zawiera mechanizm operacji na okienkach ekranowych, oparty na zasadzie stosu okienek oraz dostarcza nieproceduralnych instrukcji pisania i czytania danych poprzez formularz.

Konstrukcja języka 4GL zawiera pewne niewygodne w skutkach usztywnienia. Obiekty takie jak formularze, okna i kursory są deklarowane statycznie jako obiekty globalne. Dotkliwym ograniczeniem jest fakt, że argumentem funkcji nie może być funkcja. Można mieć zastrzeżenia do tego, że w sytuacjach operowania na obiektach, których wartości muszą być znane (z opcją NOT NULL) nie można wykluczyć (jakąś dyrektywą kompilatora) arytmetyki NULL w tym sensie, że wszystkie operacje na wyrażeniach niepotrzebnie realizowane są przez ten sam rozbudowany mechanizm.

Zarówno w pakiecie INFORMIX-SQL, jak i INFORMIX-4GL, istnieją języki definiowania formularzy. Mają one zasadniczo podobną konstrukcję, ale istnieje wiele istotnych różnic. Szkielet struktury opisu jest ten sam, jako zaletę należy podkreślić możliwość ekranowego projektu postaci formularza. W INFORMIX-SQL formularz musi być związany z tablicami bazy danych, w INFORMIX-4GL może to być w istocie jedynie definicja typów, zgodnych z typami odpowiednich zmiennych programowych. Formularz w INFORMIX-4GL daje możliwość tworzenia tablic ekranowych, w INFORMIX-SQL formularz wyświetla tylko pojedyncze rekordy.

Wiele różnic w tych językach wynika z faktu, że w pakiecie INFORMIX-4GL praca z formularzem odbywa się programowo ze wszystkimi możliwościami języka 4GL, w pakiecie INFORMIX-SQL istnieje natomiast standardowy interpreter formularzy. Stąd z braku możliwości programowych pakietu w języku definicji formularzy w INFORMIX-SQL istnieje kilka specyficznych dla niego konstrukcji. Najistotniejsze z nich to możliwość realizowania (w różny sposób) operacji typu JOIN pomiędzy tablicami i programowania akcji dla zdarzeń realizowanych przez standardowy interpreter.

Język tworzenia raportów w pakiecie INFORMIX-4GL jest zanurzony w języku 4GL, natomiast w pakiecie INFORMIX-SQL istnieje samodzielnie. W specyfikacji raportu postać wydruku definiuje się programowo, mniej wygodnie niż w przypadku formularzy. Pomiędzy tymi językami istnieją nie uzasadnione drobne rozbieżności syntaktyczne w konstrukcjach semantycznie równoważnych (np. zdania PROMPT, DEFINE - DECLARE, funkcje SUM - TOTAL). Konwencja użycia znaczników dla rozróżnienia zmiennych i nazw pól tablic jest różna w obu językach.

W obu pakietach istnieją generatory formularzy, a w INFORMIX-SQL również generator raportów. Służą one wytworzeniu podstawowej wersji specyfikacji obiektu, szkieletu, podlegającego najczęściej istotnym modyfikacjom. Wszystkie te generatory są dość proste, nie dają możliwości interakcyjnej modyfikacji generowanych kodów. Wśród nich generator raportów jest fatalnie ogólnikowy, modyfikacja po jego generacji polega niestety głównie na dopisaniu najistotniejszej sekwencji.

8. MECHANIZMY OPTYMALIZUJĄCE PROCES DOSTĘPU DO DANYCH

Zagadnienie to w systemie INFORMIX nie doczekało się bogatych rozwiązań, a istniejące nie wydają się być przemyślane do końca. Dokumentacja niemal przemilcza ten fakt.

W języku SQL systemu INFORMIX istnieje jedna konstrukcja związana z tym zagadnieniem. Jest to zdanie UPDATE STATISTICS, powodujące (dla wybranej tablicy lub całej bieżącej bazy danych) przeniesienie z poziomu C-ISAM do opisu bazy danych informacji o liczności zapisów w plikach. Informację tę system wykorzystuje przy wyborze ścieżki dostępu dla łączenia dwu tablic.

Rozwiązanie to ma istotne wady. Oddaje sterowanie aktualizacją statystyki wyłącznie programiście. Wyniki wykonanej jednokrotnie aktu-

alizacji będą podstawą działania systemu nawet po długim czasie i statystycznie bardzo znaczących modyfikacjach zapisów w plikach, o ile nie wykona się ponownie zdania UPDATE STATISTICS w odpowiednim momencie. Rozwiązanie to wydaje się tym dziwniejsze, że na poziomie C-ISAM w plikach indeksowych informacja jest aktualizowana na bieżąco i INFORMIX mógłby wprost stamtąd z niej korzystać, a nie przenosić jej w ogóle do swojego opisu.

Jak już wspomniano przy omawianiu metod dostępu do plików, INFORMIX nie wykorzystuje informacji o stopniu wypełnienia pliku w procesie dostępu do danych.

Nie ma również prowadzonej statystycznej analizy wartości pól, skutkiem czego trudno mówić o pełnej optymalizacji sposobu wyszukiwania danych. Danymi wykorzystywanymi przez optymalizator realizacji zdań SELECT jest liczność zapisów w tablicach i fakt istnienia indeksów na kolumnach kryterialnych. Dla zdań o niezbyt złożonych konstrukcjach kryterialnych optymalizator ten działa poprawnie. Dla tablic wieloindeksowych i zagnieżdżonych konstrukcji kryterialnych sposób realizacji wyszukiwania zależy od kolejności warunków we frazie WHERE zdania SELECT. Zmiana tej kolejności może o rzędy wielkości zmienić czas wyszukiwania danych. Praktycznie trzeba jako pierwszy indeks z listy ustawiać pole możliwie najbardziej selektywne (w sensie największej liczby możliwych wartości oraz ich rozkładu), ale to już jest prowadzenie analizy przez programistę i zależność programów od danych.

Nie wspomina się w dokumentacji i nie mamy żadnych podstaw sądzić, że istnieją jakiekolwiek mechanizmy umożliwiające maszynie bazy danych „uczenie się" z doświadczeń przebiegu danej sesji. Ewentualne przyspieszenia realizacji np. powtarzających się zapytań wynikają raczej z poziomu systemu operacyjnego i stanu pamięci.

9. CHARAKTERYSTYKA ZINTEGROWANEGO SŁOWNIKA DANYCH

W zintegrowanym słowniku bazy danych można wyróżnić dwa poziomy: systemowe tablice opisu bazy danych, powstające automatycznie w momencie tworzenia bazy oraz pewne tablice specjalne, związane z pracą z formularzami oraz budowaniem menu użytkownika.

Systemowe tablice opisu są własnością systemu INFORMIX. Żaden użytkownik bazy danych nie może ich aktualizować (jawnie), można jedynie wyszukiwać z nich dane. Z dodatkowymi tablicami opisu użytkownik pracuje jawnie, istnieją do ich tworzenia i aktualizacji odpowiednie narzędzia.

Istnieje dziewięć tablic systemowych, zawierających następujące informacje:

SYSTABLES - o wszystkich tablicach bazy danych,
SYSCOLUMNS - o wszystkich kolumnach tablic,
SYSINDEXES - o indeksach,
SYSTABAUTH - o uprawnieniach dostępu do tablic,
SYSCOLAUTH - o uprawnieniach dostępu do kolumn tablic,
SYSDEPEND - o związkach pomiędzy tablicami wirtualnymi a tablicami, na bazie których powstały,
SYSSYNONYMS - o synonimach tablic,
SYSUSERS - o uprawnieniach dostępu użytkowników do bazy danych,
SYSVIEWS - o wszystkich tablicach wirtualnych, z definiującymi je zdaniami SELECT.

W przypadkach wykorzystywania w pracy z bazą danych formularzy pakietu INFORMIX-4GL istnieje możliwość utworzenia w bazie dwu specjalnych tablic SYSCOLVAL i SYSCOLATT. Można w nich umieszczać specyfikacje atrybutów pól formularzy, analogicznie jak w jawnych opisach poszczególnych formularzy. Informacje w nich zawarte dołączane są na etapie kompilacji formularzy (nie modyfikując samych plików specyfikacji), w przypadku konfliktu z priorytetem niższym. Ich rola polega na możliwości globalnego sterowania pewnymi cechami, ograniczeniami, wzajemnymi związkami, w szczególności również natury integralnościowej.

W pakiecie INFORMIX-SQL, jako uzupełnienie braku możliwości językowych, istnieje mechanizm tworzenia i wykorzystywania menu użytkowego (do 19 poziomów zagnieżdżeń i 28 opcji na każdym poziomie). Treść tego menu przechowywana jest w dwóch specjalnych tablicach bazy danych: SYSMENUS i SYSMENUITEMS.

Ogólnie trzeba niestety stwierdzić, że zintegrowany słownik danych w systemie INFORMIX jest bardzo ubogi. Podstawową wadą jest brak implementacji pojęcia klucza obcego relacji - oczywisty brak możliwości wyrażenia powiązań relacyjnych. Ponadto nie ma żadnej informacji np. o utworzonych formularzach, o związkach pomiędzy formularzami a tablicami, modułami a tablicami. Wydaje się to o tyle dziwne, że system i tak np. podczas kompilacji formularza sięga do opisu bazy danych po informacje o tablicach i kolumnach, mógłby więc niejako za darmo odnotowywać te powiązania. Patrząc od drugiej strony - system nawet z odnotowanych w opisie bazy informacji nie w pełni korzysta. Na przykład po skasowaniu tablicy bazowej kasuje automatycznie odpowiednie tablice wirtualne, ale w przypadku modyfikacji struktury tablic bazowych pozostawia tablice wirtualne nietknięte, nie generując nawet ostrzeżenia.

10. MECHANIZMY KONTROLI INTEGRALNOŚCI DANYCH PODCZAS WPROWADZANIA I AKTUALIZACJI

Istnieje kilka mechanizmów kontroli integralności danych podczas wprowadzania i aktualizacji, realizowanych na różnych poziomach: na poziomie opisu bazy danych, na poziomie formularzy oraz na poziomie programów użytkowych. Jedynym mechanizmem kontroli danych na poziomie opisu jest konstrukcja tablic wirtualnych (VIEW z frazą WITH CHECK OPTION). W definicji tablicy wirtualnej można zawrzeć dowolne kryteria, jakie muszą spełniać wartości kolumn wprowadzanych lub aktualizowanych wierszy; kryteria te mogą być dowolnie rozbudowanymi wyrażeniami logicznymi i odnosić się do stanu innych tablic. Nie ma natomiast mechanizmu automatycznego wymuszania w innych tablicach zmian, pochodnych względem zmian w danej tablicy.

Mechanizmu tablic wirtualnych można również użyć jako filtru zabezpieczającego przed kasowaniem wierszy danej tablicy, bez spełnienia warunków wymaganych przez związki z otoczeniem.

Formularzowa kontrola poprawności wprowadzanych danych nie wnosi nowych jakościowo możliwości. Pozwala jednak uzyskiwać wcześniejszą, bardziej selektywną identyfikację błędów, co jest szczególnie istotne w pracy interakcyjnej. Mankamentem tego poziomu mechanizmu jest fakt, że ta kontrola pozostaje w gestii programisty, co w przypadku zmian reguł semantycznych jest trudne do weryfikacji. Pewne ułatwienie to możliwość centralnego definiowania zbioru dopuszczalnych wartości dla pól formularzy w pakiecie INFORMIX-4GL (tablica SYSCOLVAL).

Pełną kontrolę reguł semantycznych umożliwia dopiero poziom programowy, realizowany ręcznie przez programistów, a jedynym ułatwieniem może być modularna organizacja programów i tworzenie wyodrębnionych modułów z funkcjami, realizującymi modyfikacje zawartości bazy danych.

Bardzo istotnym mechanizmem zabezpieczania spójności logicznej danych w procesie wprowadzania i aktualizacji jest transakcyjny tryb przetwarzania bazy danych, w tym możliwość tworzenia bloków transakcyjnych. W zasadzie nietransakcyjny tryb przetwarzania jest przeznaczony jedynie do wstępnego wsadowego ładowania bazy danych. Dyskusyjną cechą systemu INFORMIX jest istnienie tylko jednego poziomu transakcji, co uniemożliwia wycofywanie się fragmentami z realizowanych operacji logicznych.

11. NARZĘDZIA WSPOMAGAJĄCE BUDOWANIE SYSTEMÓW APLIKACYJNYCH

System INFORMIX jest tak zbudowany, że wszelkie czynności, związane z tworzeniem systemów użytkowych, można wykonywać za pośrednictwem menu poszczególnych pakietów, bez konieczności bezpośredniego użycia komend systemu operacyjnego, co uniezależnia programistów od znajomości tego systemu.

Mechanizmem wspomagającym proces tworzenia systemów aplikacyjnych są generatory formularzy oraz raportów w systemie INFORMIX-SQL.

Wśród programów usługowych istnieje program *dbschema*, który generuje opis bazy danych w postaci programowej, w oparciu o istniejącą bazę danych, co usprawnia proces tworzenia powielarnych struktur baz danych.

Dla potrzeb programowania, wspomagania procesu projektowania systemów użytkowych, pakiet INFORMIX-4GL buduje w sposób automatyczny (w odpowiednim momencie) bazę danych SYSPGM4GL, technologiczną bazę specyfikacji programów. Zakłada w niej specjalne tablice systemowe, zawierające informacje o modułach, programach i listach modułów, z których się składają, o opcjach kompilacji, o bibliotekach wymaganych do linkowania itp. Stosowany jest mechanizm selektywnej kompilacji, związany z datami dokonywanych zmian.

12. NARZĘDZIA KONTROLI INTEGRALNOŚCI BAZY DANYCH

Jedynym systemowym narzędziem, realizującym funkcję z tej grupy, jest pomocniczy program *bcheck*, kontrolujący zgodność pliku indeksowego z odpowiadającym mu plikiem danych, wymagany przez organizację C-ISAM. Program ten pozwala również odtworzyć poprawną zawartość pliku indeksowego w oparciu o plik z danymi, o ile nie została uszkodzona informacja o strukturze pliku indeksowego.

13. MECHANIZMY OCHRONY DANYCH PRZED FIZYCZNYM USZKODZENIEM

W systemie INFORMIX istnieją dwa mechanizmy służące odtwarzaniu stanu bazy danych po fizycznych uszkodzeniach. Podstawową konstrukcją jest plik transakcyjny, wykorzystywany przy odtwarzaniu stanu całej bazy, poza tym ist-

nieje możliwość kronikowania zmian w poszczególnych tablicach.

W przypadku dużej intensywności przetwarzania bazy danych można archiwować ciąg plików transakcyjnych. INFORMIX pozwala odtwarzać stan bazy danych jedynie procesem do przodu, ponadto tylko pełnymi plikami transakcyjnymi, bez możliwości wskazania punktu czasowego, do którego należy odtworzyć stan bazy danych. Nie ma możliwości cofnięcia procesu przetwarzania bazy danych do zadanego momentu czasu lub o określoną liczbę transakcji, co byłoby przydatne np. w sytuacjach zmiany koncepcji przetwarzania od pewnego momentu.

Sterowanie procesem tworzenia kopii bazy danych i kolejnych plików transakcyjnych jest realizowane jawnie komendami języka SQL, przy wyłączeniu współbieżnego przetwarzania bazy danych.

14. ZAKOŃCZENIE - UWAGI O NOWEJ WERSJI SYSTEMU

Obecnie istnieje już, dla systemu operacyjnego UNIX, wersja 4.0 systemu INFORMIX (z marca 1990 r.), składająca się z następujących pakietów:

INFORMIX-SQL,
INFORMIX-SQL Runtime Facility,
INFORMIX-ESQL/ADA,
INFORMIX-ESQL/C,
INFORMIX-ESQL/COBOL,
INFORMIX-ESQL/FORTRAN,
INFORMIX-ESQL Embedded Languages Runtime Facility,
INFORMIX-4GL,
INFORMIX-4GL Runtime Facility,
INFORMIX-4GL Rapid Development System,
INFORMIX-4GL Rapid Development System Runtime Facility,
INFORMIX-4GL Interactive Debugger,
INFORMIX-QuickStep
INFORMIX-SE
INFORMIX-OnLine,
C-ISAM,
INFORMIX-NET,
INFORMIX-STAR.

W architekturze pakietów, w stosunku do omawianej przez nas starszej wersji, dokonano szeregu zmian i rozszerzeń.

1. Maszyna bazy danych została wyodrębniona jako oddzielny pakiet i oferowana jest w dwu wariantach: INFORMIX-SE oraz INFORMIX-OnLine.
2. Zostały utworzone pakiety Runtime Facility, stanowiące odpowiedniki podstawowych pakietów, przeznaczone dla użytkowników oprogramowania „pod klucz", którym zbędne są kompilatory itp.
3. Pojawił się pakiet realizujący zanurzenie języka SQL w jeszcze jednym języku programowania - Fortranie.
4. Oferowany jest nowy pakiet INFORMIX-QuickStep, wspomagający budowanie systemów aplikacyjnych, poprzez generowanie kodów programowych w oparciu o tworzone interakcyjnie projekty raportów oraz zapytania do bazy danych.
5. Oferowane są dwa warianty pakietów pracy w sieci:
   INFORMIX-NET dla konfiguracji z maszyną bazy danych
   INFORMIX-SE oraz INFORMIX-STAR dla konfiguracji z INFORMIX-OnLine.

Niestety nie mamy żadnych doświadczeń z tą wersją systemu. Podamy jedynie kilka spostrzeżeń z poziomu składni języków.

Maszyna bazy danych INFORMIX-OnLine przeznaczona jest do pracy ciągłej (bez przerw technologicznych) i przy dużym obciążeniu. Steruje automatycznie archiwowaniem danych i organizuje odtwarzanie stanu bazy danych po awariach, bez jawnego udziału administratora (usunięto odpowiednie komendy języka). Nie jest oparta na pakiecie C-ISAM, zarządza poziomem wejścia/wyjścia bezpośrednio i optymalizuje podział pamięci. Wprowadza trzy nowe typy danych: znakowe zmiennej długości, teksty ASCII oraz typ BLOBs (Binary Large OBjects), przeznaczony do pamiętania modułów ładowalnych, arkuszy kalkulacyjnych, zapisu obrazów, głosu itp., o pojemności do 2 GB.

Ogólnie mówiąc, wersja 4.0 wprowadza pewne możliwości, których brak daje się odczuć w wersjach poprzednich. Na przykład dla opcji WAIT w zdaniu SET LOCK MODE pojawia się możliwość określenia maksymalnego czasu oczekiwania na zwolnienia dostępu do danych. Istnieje językowa możliwość specyfikacji unikalnych wartości pól w kolumnie, niezależnie od faktu istnienia indeksu. Można również zastrzec unikalność złożeń wartości kilku kolumn (opcja CONSTRAINT), co daje możliwość jawnego zdefiniowania (i umieszczenia w opisie bazy danych) kluczy głównych tablic bazowych. Opcja WITH HOLD w deklaracji kursorów umożliwia definiowanie kursorów, które nie są automatycznie zamykane w momencie zakończenia transakcji. Z rozwiązań technologicznych pojawiło się np. w języku 4GL zdanie FREE, umożliwiające programowe zwalnianie obszarów pamięci, zajętych dla dynamicznie konstruowanych zdań języka SQL.

Została wprowadzona opcja nowego trybu przetwarzania bazy danych - MODE ANSI. Baza w tym trybie jest przetwarzana dokładnie wg standardu ANSI języka SQL - użycie dowolnych rozszerzeń jest sygnalizowane ostrzeżeniem. W szczególności w tym trybie wszystkie operacje

przetwarzania bazy danych są realizowane. W blokach transakcyjnych - zdanie COMMIT WORK zatwierdza transakcję i jednocześnie automatycznie rozpoczyna niejawnie następną. W trybie ANSI, zgodnie ze standardem, nazwy tablic nie muszą być unikalne w całej bazie danych i mogą być kwalifikowane nazwą użytkownika, będącego właścicielem tablicy.

Istotnym rozszerzeniem, dostępnym w wersji 4.0, jest możliwość tworzenia pliku, śledzącego kroki realizacji zdań języka SQL, w tym tekst realizowanego zdania, liczbę dostępów do dysku, liczbę pobranych rekordów, używane indeksy lub tworzone indeksy robocze, kolejność przetwarzania tablic. Informacje te może wykorzystać administrator bazy danych dla ręcznego dostrojenia systemu (poprzez przebudowę indeksów lub ewentualną dekompozycję zdań, nieefektywnie realizowanych przez system).

15. Z OSTATNIEJ CHWILI

We wrześniu 1991 roku wprowadzono na rynek system INFORMIX w wersji 4.10. Maszyna bazy danych INFORMIX-OnLine w wersji 4.10 jest pierwszym pracującym pod systemem UNIX pakietem, który spełnił wszystkie warunki i uzyskał certyfikat National Institute of Standards and Technology (NIST). Certyfikat ten potwierdza spełnienie przez INFORMIX normy FIPS-127.1, określającej wymagania stawiane przez instytucje rządowe USA systemom zarządzania relacyjną bazą danych. Należy podkreślić, że norma FIPS-127.1 znacznie przewyższa wymagania określone przez normę ANSI w tym zakresie.

INFORMIX-OnLine w wersji 4.10 charakteryzuje się następującymi zaletami:

- nowe komendy ułatwiające monitorowanie bazy danych,
- ułatwienie administrowania bazą danych,
- możliwość wykonywania równoległego sortowania z wykorzystaniem multiprocesorowości,
- zredukowana liczba słów zastrzeżonych - ułatwienie tworzenia oprogramowania aplikacyjnego,
- ułatwienie zarządzania zmiennymi typu BLOB.

Firma INFORMIX również we wrześniu 1991 r. rozpoczęła sprzedaż oprogramowania typu CASE pod nazwą OpenCase Program.

---

dr inż. Jacek Stochlak

Pre-sales Support Manager International Computers Limited

Relacyjna baza danych INGRES

Architektura i narzędzia tworzenia systemów aplikacyjnych

1. Wprowadzenie

INGRES jest środowiskiem programowania czwartej generacji zbudowanym w oparciu o system zarządzania relacyjną bazą danych. Zostało ono opracowane w ramach projektu badawczego na Uniwersytecie Kalifornijskim w Stanach Zjednoczonych. Pierwsze prototypy systemu powstały w roku 1975 i były rozwijane w środowisku akademickim do roku 1981.

INGRES jako produkt handlowy został wprowadzony na rynek przez firmę Relational Technology Inc. (RTI) w roku 1981. Sukces. jaki odniósł i opinie użytkowników zdecydowały, że firma zmieniła nazwę na INGRES Corporation w USA i INGRES Limited w Wielkiej Brytanii. Obecnie pracuje na świecie około 16000 instalacji systemu INGRES wykorzystujących różne platformy systemowe (duże systemy komputerowe, minikomputery i mikrokomputery).

INGRES Corporation od momentu powstania rozwijała środowisko INGRES głównie dla platform systemowych pracujących pod kontrolą systemu operacyjnego UNIX. Dodatkowo INGRES RDBMS został przeniesiony w środowisko systemu operacyjnego VMS minikomputerów firmy DEC, systemu operacyjnego VME komputerów firmy ICL oraz na pewne systemy komputerowe firmy IBM. Obecnie są dostępne również wersje środowiska INGRES RDBMS dla systemów mikrokomputerowych pracujących pod kontrolą systemów operacyjnych OS/2 i MS-DOS.

INGRES jest jednym z pierwszych systemów zarządzania relacyjną bazą danych zbudowanych

w oparciu o wielowątkową oraz wieloserwerową architekturę serwera bazy danych (Multithreaded-Multiserver Architecture). Architektura ta wykorzystuje wszystkie możliwości przetwarzania współbieżnego dostępnego na systemach wieloprocesorowych i systemach sieciowych bazujących na modelu klient-serwer. Przyjęte zasady implementacji środowiska INGRES sprawiają, że główną jego zaletą są własności przetwarzania rozproszonego (Distributed Processing), obejmujące zarówno rozproszenie przetwarzania jak i rozproszenie danych. Umożliwiają one również programom aplikacyjnym i użytkownikom systemu INGRES korzystanie z informacji przechowywanych w innych bazach danych oraz pozwalają na pełną integrację z systemami mikrokomputerowymi IBM PC i PS/2. Środowisko programowania INGRES powstało na bazie doświadczeń eksperymentalnej implementacji koncepcji relacyjnego podejścia do danych powstałej w latach 70. Ze względu na to, że relacyjne podejście do zarządzania danymi sprawdziło się i stało się pożądane przez użytkowników systemów komputerowych, INGRES Corporation rozbudowało produkt w dwóch kierunkach. Z jednej strony wzbogacony on został o bardziej złożone funkcje istotne dla rozbudowanych instalacji. Z drugiej natomiast, bardziej zorientowany na użytkownika poprzez wykorzystanie graficznych metod współpracy (Graphical User Interface) opartych o system okien i inne obiekty środowiska graficznego. Tym samym INGRES RDBMS stał się systemem programowania i zarządzania relacyjną bazą danych, który może być wykorzystany do implementacji systemów aplikacyjnych w różnych obszarach zastosowań i na różnych platformach systemowych. Ze względu na swoją otwartą architekturę może spełniać obecne wymagania użytkowników, jak również te, które mogą powstać w przyszłości.

INGRES jest strategiczną relacyjną bazą danych firmy ICL dostępną na wszystkich platformach systemowych i pracuje we wszystkich środowiskach operacyjnych (VME, UNIX, OS/2 i MS-DOS).

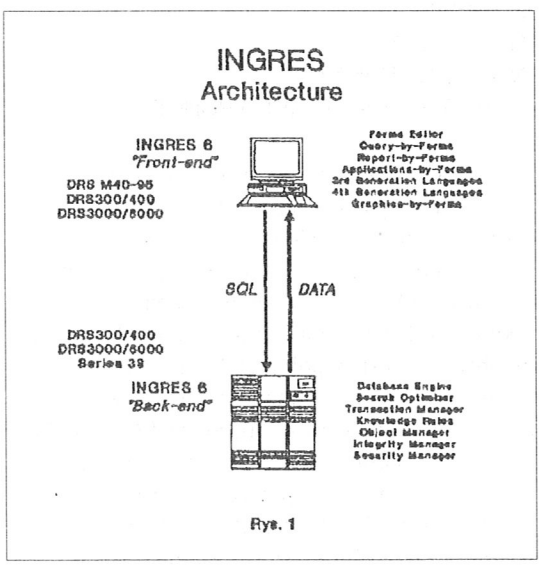
2. Architektura i elementy systemu

INGRES składa się z systemu zarządzania relacyjną bazą danych (RDBMS) i ściśle z nim zintegrowanego zestawu narzędzi przeznaczonych dla twórców oprogramowania i użytkowników systemu (End-Users). Architektura systemu INGRES wyraźnie rozdziela system zarządzania bazą danych oraz aplikacje i narzędzia, które z nim współpracują (rys. 1). System zarządzania jest nazywany serwerem bazy danych (Database Server, Back-End), programy aplikacyjne i narzędzia współpracy - serwerami klientów (Client Server, Front-End).

Serwer bazy danych jest podstawowym elementem środowiska INGRES. Jest on jednym z pierwszych "inteligentnych" serwerów relacyjnej bazy danych specjalnie zaprojektowanych do zarządzania wszystkimi trzema istotnymi dzisiaj zasobami informacyjnymi. Są nimi: dane, wiedza i obiekty. Ze względu na swoje unikalne cechy środowisko zarządzania bazą danych INGRES pozwala na włączenie zasad funkcjonowania systemów aplikacyjnych do serwera bazy danych. Do chwili obecnej musiały być one uwzględniane w programach aplikacyjnych.

INGRES DBMS Server jest w rzeczywistości serwerem języka SQL (Strukture Query Language). Został on zaprojektowany w taki sposób, aby efektywnie obsługiwał zarówno krytyczne czasowo aplikacje przetwarzania transakcyjnego jak również złożone środowiska informatycznych systemów zarządzania oraz systemów wspomagania podejmowania decyzji. Wszystkie aplikacje i narzędzia pracujące w serwerach klientów kontaktują się z nimi za pośrednictwem komunikatów będących zleceniami języka SQL. Komunikaty przekazywane w przeciwnym kierunku, z serwera bazy danych do serwerów klientów, są danymi żądanymi zleceniami języka SQL.

Narzędzia tworzenia oprogramowania środowiska obsługi INGRES zmieniają dotychczasowe podejście do procesu opracowywania systemów aplikacyjnych. Uwalniają one projektantów i programistów od trwającego czasami wiele miesięcy pracochłonnego i nużącego procesu przygotowania i uruchamiania programów, dając im do dyspozycji w pełni zintegrowany system programowania 4. generacji. INGRES jest jedną z niewielu relacyjnych baz danych, którą dają do dyspozycji swoich użytkowników zestaw narzędzi zorientowanych zarówno na użytkownika nie posiadającego głębokiej wiedzy informatycznej, jak również przeznaczonych dla analityków, projektantów i programistów złożonych systemów informatycznych. Narzędzia te obejmują języki programowania tabelarycznego (Query-by-Forms,



Rys. 1

Report-by-Forms, Application-by-Forms), języki programowania trzeciej i czwartej generacji oraz narzędzia CASE (Computer Aided Software Engineering). Tak zbudowane środowisko współpracy z relacyjną bazą danych pozwala na bardzo efektywne opracowanie, uruchamianie i konserwację oprogramowania aplikacyjnego przy znacznie zmniejszonych kosztach i w krótszym czasie.

Narzędziami tworzenia oprogramowania aplikacyjnego i współpracy z bazą danych są:

INGRES/ISQL
INGRES/MENU
INGRES/FORMS
INGRES/QUERY
INGRES/REPORTS
INGRES/Simplify
INGRES/GRAPHICS
INGRES/ESQL
INGRES/APPLICATIONS 4GL
INGRES/WINDOWS 4GL

Znane firmy komputerowe, do których należą między innymi DEC, IBM, ICL, Data General, Santa Cruz Operation/open Desktop, Interactive Systems Corporation i Tandem Computers, potwierdziły zalety narzędzi środowiska INGRES, poprzez wybranie go jako produktu oferowanego na własnych platformach systemowych.

Jednym z najbardziej istotnych wymagań stawianym obecnie systemom informatycznym jest zapewnienie szybkiego i niezawodnego dostępu do informacji. Dostęp ten powinien być zagwarantowany bez względu na to, gdzie się one znajdują, na jakim sprzęcie i w jakim środowisku pracują bazy danych zawierające te informacje oraz niezależnie od typów sieci łączących wykorzystywane systemy komputerowe. Ingres Corporation biorąc pod uwagę te wymagania, rozbudowało system obsługi relacyjnej bazy danych o mechanizmy przetwarzania rozproszonego. Zapewniają je następujące moduły systemu:

INGRES/NET
INGRES/STAR
INGRES/GATEWAY
INGRES/PCLINK i INGRES/PCHOST

INGRES/NET, pierwszy z tych modułów, pozwala na separację serwerów bazy danych i serwerów klientów. Mogą one pracować na różnych systemach komputerowych i w różnych środowiskach operacyjnych połączonych sieciami lokalnymi (LAN) i zdalnymi (WAN). Obsługuje on obszerny zestaw połączeń komunikacyjnych i sieciowych będących standardami międzynarodowymi i przemysłowymi (OSI, TCP/IP, DECnet, X25, SNA). Drugim modułem istotnym dla realizacji przetwarzania rozproszonego środowiska INGRES jest INGRES/STAR. Moduł ten odpowiada za zarządzanie rozproszeniem danych zapewniając, ze twórcy oprogramowania i użytkownicy systemu widzą je jako jednorodną logiczną bazę danych niezależną od rozmieszczenia danych, sprzętu i środowiska operacyjnego serwerów oraz protokołów komunikacyjnych wykorzystywanych do realizacji dostępu do tych systemów. Kolejnym modułem przetwarzania rozproszonego środowiska INGRES jest INGRES/GATEWAY. Pozwala on programom aplikacyjnym i użytkownikom systemu INGRES wykorzystywać informacje przechowywane w innych niż INGRES bazach danych. Obecnie dostępne są bramy dla baz danych DB2 iIMS firmy IBM, IDMSX firmy ICL, Rdb/RMS firmy DEC oraz baz rodziny dBASE systemów mikrokomputerowych IBM PC i PC/2.

Uzupełnieniem wymienionych wyżej modułów przetwarzania rozproszonego jest INGRES/PCLINK i INGRES/PCHOST. Umożliwiają one pełną integrację systemów mikrokomputerowych IBM PC i PS/@ ze środowiskiem obsługi i programowania relacyjnej bazy danych INGRES.

Biorąc pod uwagę fakt, że jednym z istotnych warunków sukcesu w dzisiejszym szybko zmieniającym się i konkurencyjnym świecie biznesu jest usprawnienie i skrócenie czasu koniecznego do opracowania i wdrożenia systemów aplikacyjnych, INGRES Corporation uzupełnia środowisko bazy danych INGRES o narzędzia CASE firmy Cade Technologies. W wyniku współpracy obu firm zostało opracowane środowisko INGRES/teamwork I-CASE (Interactive CASE) uzupełniające narzędzia tworzenia oprogramowania o produkty służące do analizy i projektowania systemów aplikacyjnych. Połączenie wszystkich tych elementów daje użytkownikom relacyjnej bazy danych INGRES dużo większą i pełniejszą kontrolę nad wszystkimi etapami procesu tworzenia systemów aplikacyjnych.

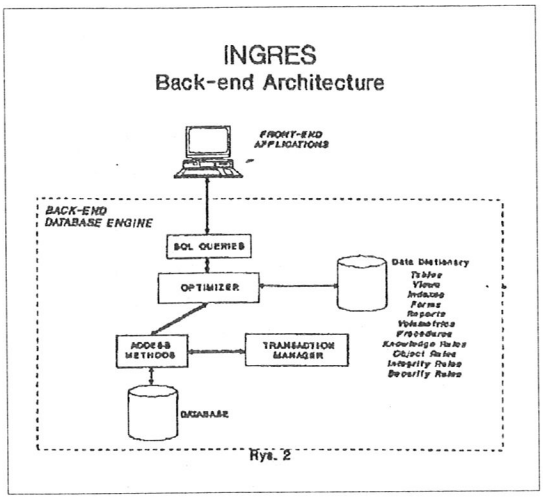
Modułami INGRES/teamwork I-CASE są:

- teamwork/IM (modelowanie informacji)
- teamwork/SA (analiza strukturalna)
- teamwork/RT (modelowanie strukturalne)
- teamwork/SD (projektowanie strukturalne)
- teamwork/DBD (projektowanie i generacja bazy danych)

3. Serwer bazy danych

Architektura środowiska INGRES wyraźnie oddziela serwer bazy danych oraz programy aplikacyjne i narzędzia współpracy użytkownika z bazą danych pracujące w serwerach klientów. Serwer bazy danych jest serwerem języka SQL wykorzystywanym przez serwery klientów do współpracy z bazą danych (rys. 2). Zlecenia języka SQL umożliwiają tworzenie bazy danych, wyszukiwanie i przetwarzanie informacji, definiowanie uprawnień dostępu (access rights), zasad

poufałości (privacy), rzetelności (integrity) oraz bezpieczeństwa danych (security). Uzupełniają je zlecenia usługowe istotne dla administratora bazy danych do jej efektywnego i bezpiecznego zarządzania.



Rys. 2

Jednym z najważniejszych elementów systemu środowiska INGRES jest zintegrowany słownik danych (data dictionary). Jest on zbiorem katalogów systemowych opisujących różnego rodzaju obiekty występujące w bazie danych, obowiązujące w niej uprawnienia dostępu, zasady poufności, rzetelności i bezpieczeństwa danych. Obiektami opisywanymi w słowniku danych są między innymi: użytkownicy i grupy użytkowników bazy danych, tablice, kolumny, projekcje (views), indeksy, wzorce, formaty raportów i wykresów itd. Sam słownik danych jest bazą danych opisującą system, przechowującą informacje w postaci dokładnie zdefiniowanych tablic. Użytkownicy systemu oraz programy aplikacyjne mogą wykorzystywać zlecenia języka SQL do kierowania żądań do słownika danych. Mogą również zmieniać informacje w nim zawarte, pod warunkiem jednak, że posiadają odpowiednie uprawnienia nadane im przez administratora bazy danych.

Centralnym elementem serwera bazy danych INGRES jest moduł optymalizacji żądań (Optimizer). Współpracuje on z modułem obsługi zleceń (SQL Query) odpowiedzialnym za współdziałanie z serwerami klientów. Zadaniem modułu optymalizacji żądań jest analiza każdego otrzymanego zlecenia SQL i wybranie najbardziej korzystnej i optymalnej metody dostępu do żądanych danych. W procesie tym wykorzystywane są dane statystyczne opisujące bazę danych, które są przechowywane w słowniku danych. Informacje te opisują budowę i wielkości tablic, wykorzystywane fizyczne struktury pamięci (metody dostępu), dostępne tablice indeksowe oraz ich rozmiary, rozkłady kluczy w rekordach poszczególnych tablic itd.

Moduł optymalizacji żądań dostępu jest kluczowym elementem decydującym o wydajności systemu obsługi relacyjnej bazy danych. Spowodowane jest to zasadami obowiązującymi przy korzystaniu z relacyjnej bazy danych. Użytkownik systemu, kierując żądanie do bazy danych, podaje jedynie, jakie dane chce otrzymać, pozostawiając systemowi decyzję jaką metodę dostępu wykorzystać. O wydajności systemu nie decydują więc umiejętności użytkownika, lecz zdolności, spryt i inteligencja modułu optymalizacji żądań.

Serwer bazy danych INGRES sprawuje pełną kontrolę nad współbieżnością dostępu (cuncurrency control), rzetelnością i poufnością informacji oraz odpornością (resilience) systemu na różnego rodzaju awarie. Elementem odpowiedzialnym za większość z tych funkcji jest moduł obsługi transakcji (Transaction Manager). Wykorzystuje on zestaw zamków (locks) do kontroli współbieżnego dostępu do bazy danych wielu użytkowników systemu, odpowiada za obsługę wielozleceniowych transakcji (multi-statement transactions) oraz wykrywanie zakleszczeń systemu (dead locks). Do jego zadań należy również sterowanie procesami odtwarzania w przód (roll-forward) i odtwarzania wstecz (roll-backward). Procesy te korzystają ze specjalnych dzienników systemu (before-image journal, after-image journal) i decydują o odporności systemu obsługi bazy danych na różnego rodzaju awarie sprzętu i oprogramowania.

Logiczna i fizyczna struktura bazy danych INGRES jest w pełni zgodna z relacyjnym modelem danych. Nie istnieją ograniczenia na liczbę obsługiwanych relacji oraz wielkości tablic. Fizyczna budowa bazy danych jest pod pełną kontrolą użytkownika i administratora systemu. Tablice bazy danych są przechowywane w systemie plików środowiska operacyjnego z możliwością wykorzystania wielu metod dostępu. Należą do nich: pliki sekwencyjne indeksowane (ISAM), pliki o dostępie bezpośrednim oparte o funkcję mieszającą (hashed random), pliki o strukturze B-drzew (B-trees) oraz pliki typu HEAP i HEAPSEQ. Wybrana metoda dostępu definiuje podstawowy klucz dostępu do tablicy (primary key). Dodatkowo z każdą tablicą może być związana dowolna liczba indeksów zbudowanych w oparciu o klucze pomocnicze (secondary key). Klucze pomocnicze są pojedynczymi polami lub złożeniem max. 6 pól tablicy.

Elementem serwera bazy danych INGRES odpowiedzialnym za obsługę metod dostępu jest moduł metod dostępu (Access Method). Jedną z najbardziej interesujących cech serwera bazy danych środowiska INGRES jest to, że został on specjalnie zbudowany do zarządzania trzema zasobami informacyjnymi występującymi w obecnie wymaganych bazach danych. Są nimi: dane, wiedza i obiekty. W dalszej części rozdziału omówione zostaną podstawowe zagadnienia związane z zarządzaniem tymi zasobami.

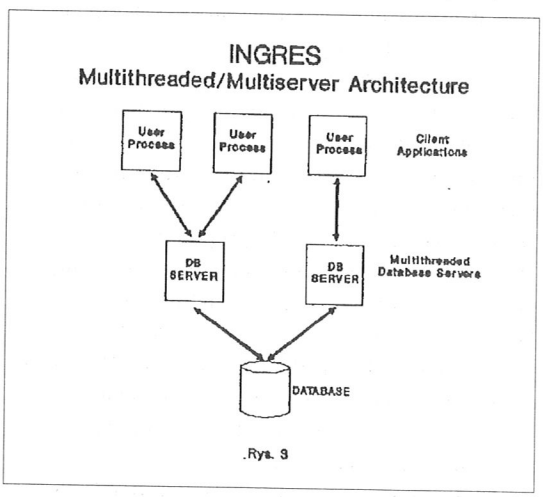
Zarządzanie danymi

Do najbardziej istotnych cech zarządzania danymi wprowadzonych w najnowszej wersji środowiska IGRES należą:

- architektura wielowątkowa i wieloserwerowa (Multithreaded Multiserver Architecture),
- wbudowane procedury bazy danych (Compiled Database Procedures),
- "inteligentny" optymalizator dostępu (Inteligent Optimizer),
- współbieżne składowanie bazy danych (On-line Archiving),
- redukcja operacji wejścia /wyjścia (I/O Reduction Techniqes),
- wieloplikowe i wielowolumenowe struktury pamięci (Multifile and Multiwolume Storage),
- dwufazowy protokół potwierdzeń (Two-Phase Commit).

Wielowątkowa i wieloserwerowa architektura

Środowisko programowania INGRES było jednym z pierwszych, w którym implementację serwera bazy danych oparto o architekturę wielowątkową i wieloserwerową. Architektura ta pozwala na współistnienie wielu serwerów obsługujących jedną bazę danych. Każdy z serwerów posiada wiele wątków związanych z aktywnymi serwerami klientów obsługujących programy aplikacyjne i narzędzia wykorzystywane przez użytkowników do współpracy z bazą danych (rys. 3).



.Rys. 3

Administrator systemu, w zależności od potrzeb oraz obciążenia systemu, może uruchomić dowolną liczbę serwerów bazy danych na dostępnych procesorach systemów komputerowych. Poszczególne serwery mogą pracować z różnymi priorytetami dla wybranych prac lub typów prac występujących w systemie. Dla przykładu, serwery bazy danych odpowiadające za obsługę krytycznych czasowo operacji przetwarzania transakcyjnego powinny posiadać wyższy priorytet w stosunku do serwerów obsługujących informacyjne systemy zarządzania lub systemy wspomagania podejmowania decyzji.

Przyjęta architektura systemu obsługi i zarządzania bazą danych pozwala na:

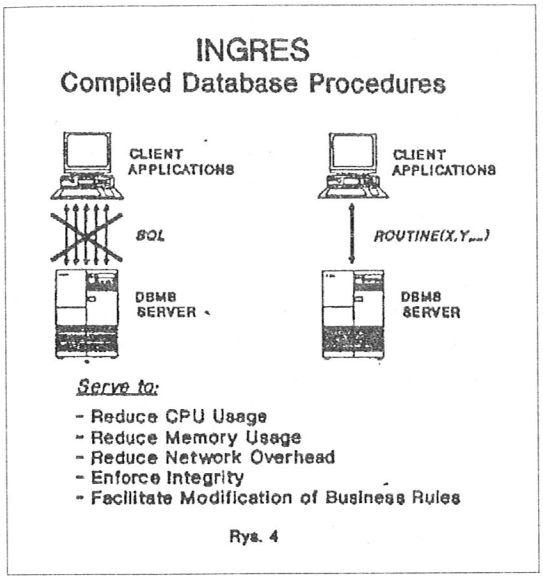
- Zmniejszenie obciążenia systemu na jednego użytkownika bazy danych ("Per-user" Overhead). Badania wykazały, że obciążenie wielowątkowego serwera bazy danych liczone na jednego użytkownika zostało zredukowane do około 30-50Kb dla typowej usługi przetwarzania transakcyjnego. Dla porównania obciążenie jednowątkowego serwera dla tych samych warunków pracy wynosi 200-400Kb.
- Pełne wykorzystanie mocy obliczeniowej systemów wieloprocesorowych i wielonodalnych.
- Elastyczność w dostosowaniu konfiguracji systemu obsługi bazy danych do aktualnych potrzeb i obciążenia systemu.

Wbudowane procedury bazy danych

Kolejnym nowym elementem zarządzania danymi środowiska INGRES są wbudowane procedury bazy danych. Procedury bazy danych są funkcjami przygotowywanymi w języku SQL/4GL, które po kompilacji są przechowywane i zarządzane przez serwer bazy danych (rys. 4). Mogą być one wykorzystane między innymi do implementacji predefiniowanych operacji przetwarzania transakcyjnego. Pozwalają na eliminację częstych interpretacji zleceń języka SQL opisujących te transakcje i zastąpienie ich wykonaniem wersji skompilowanej przechowywanej we współdzielonej przez wszystkich użytkowników pamięci serwera (Shared Memory).

Wprowadzenie wbudowanych procedur serwera bazy danych pozwala na:

- zmniejszenie obciążenia procesora systemu poprzez zastąpienie kosztownego procesu interpretacji przez wykonywanie wersji skompilowanej,
- zmniejszenie wymaganej wielkości pamięci operacyjnej poprzez wyeliminowanie konieczności przechowywania oddzielnej kopii skompilowanej procedury w serwerach klientów i zastąpienie jej przez jedną kopię, współdzieloną przez wszystkich użytkowników, przechowywaną w serwerze bazy danych,
- redukcję obciążenia sieci komunikacyjnej przez ograniczenie liczby komunikatów przepływających pomiędzy serwerami bazy danych i serwerami klientów,
- zwiększenie poziomu rzetelności danych przez zezwolenie na dostęp do krytycznych danych jedynie za pośrednictwem wbudowanych procedur bazy danych.



Rys. 4

"Inteligentny" optymalizator dostępu

W poprzednim rozdziale zostało już wspomniane, że optymalizacja żądań jest kluczowym czynnikiem decydującym o wydajności systemu obsługi i zarządzania relacyjną bazą danych. "Inteligentny" optymalizator żądań środowiska INGRES jest jednym z niewielu dostępnych obecnie, który zapewnia optymalną strategię obsługi żądań, bazując na danych statystycznych opisujących bazę danych oraz technikach zaczerpniętych z dziedziny sztucznej inteligencji (Artifical Intelligence). Zbieranie danych statystycznych odbywa się pod pełną kontrolą użytkownika i administratora bazy danych, którzy posiadają możliwości dynamicznego sterowania zakresem analizy statystycznej zapobiegające nadmiernemu obciążeniu serwera bazy danych.

Optymalizator żądań dostępu środowiska INGRES, w odróżnieniu od większości spotykanych obecnie, bazujących o analizę składni zleceń języka SQL (Syntax-based Optimizer), pracuje w oparciu o oszacowanie kosztów obsługi żądania (Cost-based Optimizer). Bierze on pod uwagę liczbę koniecznych operacji wejścia/wyjścia, obciążenie procesora systemu oraz obciążenie sieci komunikacyjnej komunikatami i danymi przesyłanymi pomiędzy serwerami baz danych i serwerami klientów. Został on zaprojektowany w taki sposób, aby wydajność systemu była niezależna od aktualnego położenia danych (Data-Independent Performance) oraz składni języka SQL opisującego zlecenie (Syntax-Independent Performance).

Współbieżne składowanie baz danych

Zapewnienie bezpiecznej pracy systemu obsługi i zarządzania bazy danych wymaga cyklicznego składowania zawartych w niej danych na taśmach magnetycznych lub kasetowych. Zabezpiecza to przed utratą danych na skutek niespodziewanych awarii sprzętu lub błędów obsługi oprogramowania.

System zarządzania bazy danych INGRES zapewnia własne programy składowania działające bez konieczności przerywania jej normalnej pracy (On-line Archiving). Rezultatem procesu składowania jest pełny obraz bazy danych (snapshot), który obowiązywał w momencie startu procesu. Poza programami składowania pełnej zawartości bazy danych, dostępne są również programy składowania inkrementacyjnego. Pracują one również bez konieczności przerywania normalnej pracy serwera bazy danych i składują tylko te fragmenty bazy, które zostały zmienione od chwili poprzedniego składowania. Wykorzystanie składowania inkrementacyjnego w znacznym stopniu redukuje czas składowania i liczbę koniecznych nośników magnetycznych.

Redukcja operacji wejścia/wyjścia

Serwer bazy danych środowiska INGRES wykorzystuje różnorodne techniki żmniejszające liczbę operacji wejścia/wyjścia koniecznych do obsługi transakcji z bazą danych. Ma to istotny wpływ na czas reakcji systemu, poprawienie współbieżności oraz wydajności i przepustowości systemu. Wskaźniki te są szczególnie ważne dla krytycznych czasowo systemów przetwarzania transakcyjnego. Wykorzystywane techniki obejmują:

- Fast Commit (Diferet Writes) - redukujący liczbę operacji wejścia/wyjścia dla obsługi aplikacji przetwarzania transakcyjnego,
- Group Commit (Piggybacked Commit) - eliminujący wąskie gardła dostępu do dzienników systemu poprzez grupowanie rekordów potwierdzeń zakończonych transakcji i zapisywanie ich jedną operacją wejścia/wyjścia,
- Multiblock Reads (Read Ahead) - redukujący liczbę operacji odczytu podczas przeszukiwania bazy danych poprzez odczyt kilku stron (pages) bazy danych do buforów systemowych jedną operacją wejścia / wyjścia,
- Multiblock Writes - redukujący liczbę operacji zapisu podczas aktualizacji bazy danych poprzez transmisję kilku stron z buforów systemowych jedną operacją wejścia/wyjścia.

Wieloplikowe i wielowolumenowe struktury pamięci

System obsługi i zarządzania bazą danych INGRES wykorzystuje wieloplikowe (Multi-file) i wielowolumenowe (Multi-volume) struktury pamięci do odwzorowania logicznych struktur relacyjnej bazy danych. Tablice bazy danych są przechowywane w niezależnych plikach środowiska operacyjnego, a nie w specjalnie wydzielonych

partycjach pamięci dyskowej systemu. Wieloplikowe i wielowolumenowe struktury danych pozwalają na niezauważalne przez użytkownika rozbicie relacji i przechowywanie jej w wielu plikach na różnych jednostkach dyskowych. Przyjęte zasady odwzorowania logicznych struktur danych w fizyczne struktury danych dają następujące korzyści:

- wprowadzają "Fire Walls" pomiędzy tablice, czyniąc system obsługi bazy danych bardziej odporny na awarie pamięci dyskowych oraz znacznie redukują czas konieczny do odtworzenia baz danych,
- poprawiają przepustowość i czas reakcji systemu zezwalając na przechowywanie tablic bazy danych na wielu niezależnych jednostkach dyskowych,
- eliminują konieczność dokonywania wstępnej alokacji pamięci dyskowej dedykowanej do przechowywania relacji i innych obiektów bazy danych.

Dwufazowy protokół potwierdzeń

Jedną z dwunastu zasad, jakie powinien spełniać system zarządzania rozproszoną bazą danych (Date's Twelve Rules for Distributed Database Systems in InfoDB by Chris J. Date), jest sposób obsługi transakcji rozproszonych. Transakcje rozproszone korzystają i modyfikują kilka baz danych obsługiwanych przez systemy komputerowe pracujące w sieci. Zachowanie rzetelności danych systemu wymaga, aby transakcja ta była traktowana z punktu widzenia użytkownika jako niepodzielna operacja (atomic transaction). System zarządzania bazą danych INGRES jest przygotowany do obsługi transakcji przetwarzania rozproszonego w oparciu o przyjęty jako standard protokół Two-Phase Commit.

Zarządzanie wiedzą

Wiedza jest jednym z zasobów informacyjnych bazy danych opisującym wzajemne zależności danych, podstawowe reguły funkcjonowania systemu, uprawnienia dostępu oraz limity zasobów systemowych, które mogą być wykorzystywane przez użytkowników systemu. Zagadnienia te albo przekraczały zakres systemów zarządzania relacyjnych baz danych bądź ich implementacja była bardzo niezgrabna lub całkowicie pominięta.

Brak zarządzania wiedzą jest jednym z czynników wydłużających i komplikujących proces tworzenia systemów aplikacyjnych. Spowodowane jest to tym, że twórcy oprogramowania muszą uwzględniać zasady rzetelności danych oraz reguły funkcjonowania systemu w każdym opracowanym programie aplikacyjnym. Zmuszają one również organizacje, albo do pełnego wyeliminowania żądań ad hoc do bazy danych albo do akceptowania potencjalnego bałaganu i niskiej rzetelności danych. Problemy te szczególnie ujawniają się w systemach zdecentralizowanych, zbudowanych zgodnie z architekturą klient/serwer. W przeszłości, kiedy praca systemu opierała się na centralnym systemie komputerowym, każda zmiana reguł biznesu wymagała modyfikacji pojedynczej centralnej kopii systemu aplikacyjnego. Obecnie, gdy systemy bazują o architekturę klient/serwer, każda zmiana tych reguł pociąga za sobą zmianę dziesiątek lub setek programów aplikacyjnych. Żądanie to jest praktycznie niewykonalne.

Serwer bazy danych środowiska INGRES rozwiązuje problemy związane z zarządzaniem wiedzą wykorzystując:

- system reguł (Rule System),
- system kontroli zasobów (Resource Control System),
- system kontroli uprawnień dostępu (Access Control System).

System reguł

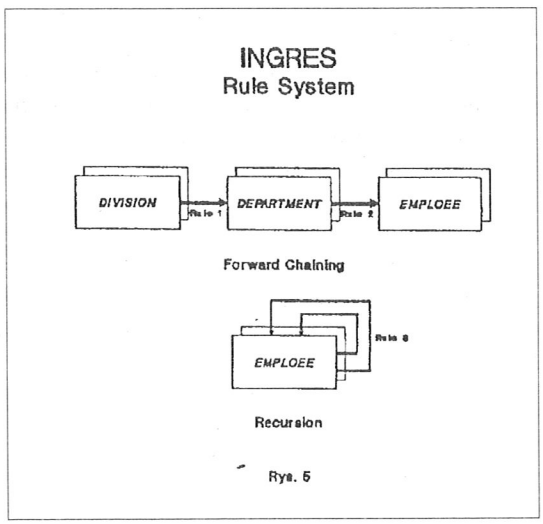
Reguły obowiązujące w środowisku INGRES są procedurami, które automatycznie wymuszają spełnienie obowiązujących zasad funkcjonowania systemu oraz obiektywną i wzajemną rzetelność danych (entity integrity, referential integrity). Serwer bazy danych uruchamia te procedury w każdej chwili, gdy zostaną spełnione kryteria aktywacji zdefiniowane przez użytkownika systemu. Reguły obowiązujące w systemie zarządzania bazą danych są tworzone nowym zleceniem języka SQL - CREATE RULE. Są one standardowymi procedurami bazy danych, przygotowanymi w języku INGRES/SQL lub INGRES/4GL. Ze względu na swoją modularną architekturę te same procedury mogą być uruchamiane przez różne reguły biznesu określające zasady funkcjonowania systemu aplikacyjnego.

System reguł środowiska INGRES pozwala na wymuszenie następujących zasad obowiązujących przy współpracy z relacyjną bazą danych:

- obiektową rzetelność danych, odpowiadającą za to, aby żaden z atrybutów definiujących podstawowy klucz rekordu nie został pominięty,
- wzajemną rzetelność danych, odpowiadającą za poprawność powiązań pomiędzy różnymi typami rekordów w bazie danych (np. w systemie kadrowym rekordy opisujące pracowników mogą być jedynie powiązane z rekordami opisującymi aktualnie istniejące departamenty),
- reguły biznesu, odpowiadające za uruchamianie obsługi zdarzeń zgodnie z obowiązującymi zasadami (np. w systemie

gospodarki magazynowej reguły te mogą automatycznie uruchamiać zadanie przygotowanie zamówienia na części w momencie, gdy stan zapasów zmniejszy się poniżej dopuszczalnego poziomu).

Implementacja systemu reguł środowiska INGRES dopuszcza łańcuchową i rekurencyjną aktywację obowiązujących zasad funkcjonowania systemu aplikacyjnego (rys. 5). Może to prowadzić do niekiedy złożonego sposobu pracy serwera bazy danych.



Rys. 5

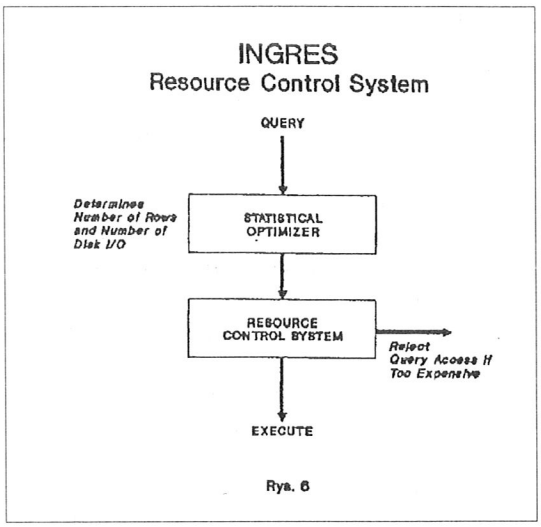
Wprowadzenie sterowanego centralnie systemu reguł daje możliwość bardzo szybkiego dopasowania systemów aplikacyjnych do zmieniających się wymagań użytkowników i obowiązujących warunków pracy. System ten uwalnia również twórców programów aplikacyjnych od konieczności uwzględniania w każdym z nich obowiązujących reguł działania systemu i zasad rzetelności danych. Pozwala na udostępnienie użytkownikom narzędzi służących do kierowania do bazy danych żądań ad hoc, bez obawy o naruszenie rzetelności danych i obowiązujących reguł biznesu. Ma to szczególnie istotne znaczenie w systemach informacyjnych zarządzania i systemach wspomagania podejmowania decyzji.

System reguł środowiska INGRES jest jedynym dostępnym komercyjnie pełnym systemem obsługującym:

- nieograniczoną liczbę niezależnych reguł związanych z dowolną tablicą bazy danych,
- aktywację reguł (firing) uzależnioną od spełnienia warunków zbudowanych w oparciu o wartości pól rekordów tablicy,
- łańcuchową i rekurencyjną aktywację reguł,
- implementację procedur obsługi w języku INGRES/SQL i INGRES/4GL.

System kontroli zasobów

System kontroli zasobów serwera bazy danych INGRES ma za zadanie kontrolę zasobów systemowych koniecznych do obsługi żądań użytkowników. Zapewnia on zachowanie przewidywanej wydajności systemu poprzez eliminację "niekończących" się żądań i zabezpieczenie użytkownikom systemu stałego dostępu do kluczowych danych istotnych dla funkcjonowania systemu.



Rys. 6

Działanie systemu kontroli zasobów jest bardzo silnie związane z rezultatami pracy systemu optymalizacji żądań w wyniku analizy statystycznej, określa przewidywaną liczbę rekordów bazy danych spełniających warunki żądania oraz liczbę operacji wejścia/wyjścia koniecznych do jego wykonania. Wielkości te są porównywane z limitami zasobów przyznanych poszczególnym użytkownikom przez administratora bazy danych. Jeśli zostaną one przekroczone, żądanie użytkownika nie jest obsługiwane.

Systemy kontroli zasobów występują również w innych dostępnych obecnie systemach zarządzania bazami danych. Działanie ich jednak ogranicza się do sprawdzania liczby rekordów spełniających warunki żądania. Pomijana jest próba oszacowania wielkości zasobów systemowych wymaganych do jego wykonania. Systemy te nie posiadają skutecznego sposobu wykrywania "niekończących" się żądań, gdyż "przechwycenie" ich jest możliwe zbyt późno już po wykorzystaniu znacznych zasobów systemowych serwera bazy danych.

System kontroli uprawnień dostępu

Kontrola uprawnień dostępu serwera bazy danych INGRES została w znaczny sposób rozbudowana w stosunku do znanych obecnie rozwiązań. Obejmuje ona poza uprawnieniami dostępu określonymi dla poszczególnych użytkowników baz danych, uprawnienia dostępu zdefiniowane dla grup użytkowników oraz samych programów aplikacyjnych. Uprawnienia dostępu nie dotyczą jedynie informacji przechowywanych w bazie danych. Obejmują one również zlecenia języka SQL, jakie mogą być używane do współpracy z bazą danych.

Wielopoziomowy system kontroli uprawnień dostępu daje administratorowi bazy danych dużą elastyczność w zapewnieniu użytkownikom systemu dostępu do wymaganych danych, z zachowaniem wysokiego stopnia ich bezpieczeństwa i poufałości.

Zarządzanie obiektami

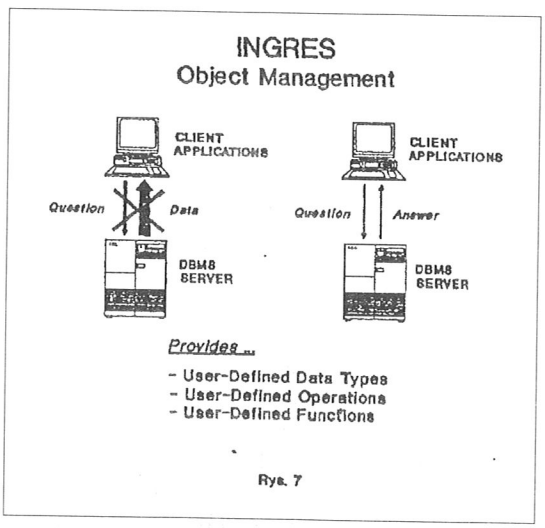
Systemy zarządzania relacyjnymi bazami danych są coraz częściej wykorzystywane przy opracowywaniu systemów obsługi działalności różnych organizacji i przedsiębiorstw. W znacznym stopniu ułatwiają one zarządzanie informacjami istotnymi dla funkcjonowania systemu, lecz w dalszym ciągu typy danych przechowywanych i przetwarzanych w ich środowisku ograniczają się do danych elementarnych, którymi są znaki i liczby. Systemy wykorzystujące bardziej złożone typy danych zmuszone są do ich obsługi poza środowiskiem relacyjnych baz danych. Ograniczenia te w znacznym stopniu zmniejszają zalety systemów zarządzania bazami danych, gdyż konieczność przetwarzania złożonych obiektów, takich jak: długość i szerokość geograficzna, współrzędne, tablice, wektory, obrazy itd. na zewnątrz systemu wprowadza wiele problemów.

Pierwszym z nich jest znaczne wydłużenie procesu opracowywania aplikacji, spowodowana koniecznością poznania zasad przetwarzania złożonych danych i uwzględniania ich we wszystkich programach aplikacyjnych. Wiąże się to z przygotowaniem procedur obsługi tych obiektów poza relacyjną bazą danych. Zmniejsza się również sama rzetelność danych. Przechowywanie i przetwarzanie złożonych obiektów poza serwerem bazy danych nie pozwala bowiem na zastosowanie centralnie sterowanego mechanizmu kontroli ich poprawności.

Kolejnym problemem wynikającym z braku możliwości obsługi złożonych obiektów w serwerze bazy danych jest znaczne pogorszenie się wydajności systemu zbudowanego w oparciu o architekturę klient/serwer. Spowodowane jest to koniecznością przekazywania poprzez sieć wszystkich złożonych obiektów informacyjnych do serwerów klientów, gdzie odbywa się ich analiza i przetwarzanie. Dotyczy to również systemów zarządzania bazami danych, które pozwalają na przechowywanie w swych tablicach obiektów binarnych BLOB (Binary Large Objects), lecz nie rozpoznają ich semantyki. Przetwarzanie złożonych obiektów informacyjnych w serwerach klientów wprowadza znaczne obciążenie sieci komunikacyjnej i może doprowadzić w skrajnym przypadku do załamania się systemu.

Serwer bazy danych środowiska INGRES poprzez wbudowane mechanizmy zarządzania obiektami rozwiązuje problemy i ograniczenia występujące w dostępnych obecnie systemach zarządzania relacyjnymi bazami danych. Mechanizmy te pozwalają na definiowanie składni i semantyki nowych typów danych oraz metod ich przechowywania i przetwarzania. Przyjęte rozwiązanie przesuwa odpowiedzialność obsługi złożonych obiektów informacyjnych z serwerów klientów na serwer bazy danych i w znacznym stopniu ułatwia i skraca proces tworzenia oprogramowania aplikacyjnego. Zapewnia również wyższy poziom rzetelności danych oraz poprawia wydajność i czas reakcji systemu.

Złożone obiekty informacyjne są przechowywane i przetwarzane przez serwer bazy danych. Użytkownicy systemu i programy aplikacyjne korzystają z tych danych standardowymi zleceniami języka SQL (rys. 7). Spotykana bardzo często obecnie konieczność odwzorowywania złożonych obiektów informacyjnych na dane elementarne i przystosowywania ich do ograniczeń wykorzystywanej bazy danych, może być zastąpiona procesem "nauku" serwera bazy danych rozpoznawania i przetwarzania tych obiektów w takiej postaci, w jakiej są one używane i przestrzegane. Czyni to dane bardziej użyteczne i zrozumiałe przez użytkowników systemu.



Rys. 7

System zarządzania obiektami środowiska INGRES opiera się o definiowane przez użytkowników:

- typy danych
- funkcje,
- operatory.

Definicje typów danych obejmują zestaw procedur określających, w jaki sposób serwer bazy danych przechowuje i przetwarza nie znane mu obiekty informacyjne. Są one najczęściej przechowywane w językach programowania trzeciej generacji, które zapewniają dużą efektywność obsługi tworzonych obiektów. Procedury te podają początkowe i dopuszczalne wartości definiowanych typów danych, sposób obsługi operacji wejścia/wyjścia, zasady sortowania itd.

Serwer bazy danych wykorzystuje te procedury, gdy podczas obsługi żądania dostępu zostaną

napotkane związane z nimi obiekty informacyjne.

Drugim elementem systemu zarządzania obiektami są definiowane przez użytkowników nowe funkcje języka SQL. Mogą być one związane z przetwarzaniem wbudowanych lub definiowanych typów danych. Funkcje te są włączane do serwera bazy danych i automatycznie uruchamiane w chwili ich napotkania w zleceniu języka SQL. Podobnie jak procedury opisujące nowe typy danych, funkcje definiowane przez użytkowników są przygotowywane przez programistów systemowych lub administratora systemu w językach programowania 3. generacji.

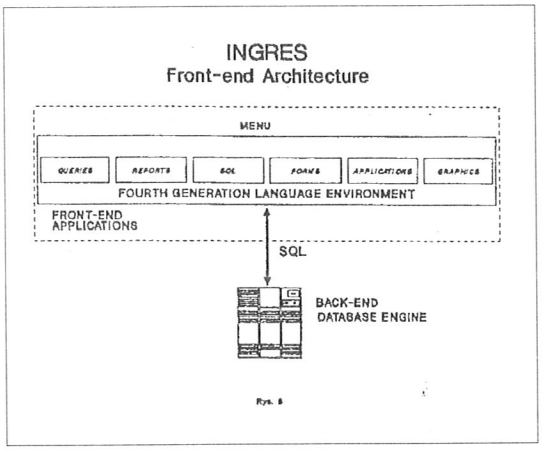
Ostatnim elementem istotnym w systemie zarządzania obiektami są definicje nowych operacji rozpoznawanych przez serwer bazy danych. Operacje te określają sposób interpretacji klasycznych operatorów arytmetycznych, operatorów relacji, operatorów logicznych itp. dla nowych typów danych. Definicje te pozwalają również na wprowadzenie do systemu specyficznych operacji dostępnych jedynie dla tworzonych obiektów informacyjnych.

4. Narzędzia tworzenia systemów aplikacyjnych

Środowisko tworzenia oprogramowania aplikacyjnego relacyjnej bazy danych INGRES obejmuje rodzinę narzędzi przeznaczonych dla użytkowników (End-Users) i programistów systemów informatycznych. Narzędzia te oraz programy aplikacyjne przygotowane przy ich pomocy, pracują w serwerach klientów, które współdziałają z serwerem bazy danych INGRES DBMS za pośrednictwem języka SQL (rys. 8). Serwer bazy danych oraz serwery klientów mogą wykorzystywać ten sam system komputerowy jak również mogą pracować na innych systemach i w różnych środowiskach operacyjnych, połączonych ze sobą lokalną lub rozległą siecią komunikacyjną.

Centralnym modułem środowiska programowania INGRES jest INGRES/MENU, który zapewnia wspólny punkt wejścia dla wszystkich pozostałych modułów systemu. Są nimi:

- narzędzia przeznaczone dla użytkownika bez szczegółowej wiedzy informatycznej pozwalające na kierowanie zapytań do bazy danych, tworzenie raportów i prezentację danych w postaci graficznej,
- narzędzia przeznaczone dla twórców systemów aplikacyjnych, które obejmują języki programowania 3. i 4. generacji,
- edytor wzorów (forms) oraz język tworzenia raportów,
- moduł interakcyjnej współpracy z relacyjną bazą danych w języku SQL,
- gotowe programy aplikacyjne, raporty i zapytania dostępne dla wykorzystywanej bazy danych.



Wszystkie moduły środowiska INGRES wykorzystują te same zasady współpracy z użytkownikiem systemu (look and feel). Obowiązują one również programy aplikacyjne przygotowane przy ich pomocy. Użytkownik systemu poznawszy zasady współpracy z jednym z modułów środowiska INGRES może bez dodatkowych nakładów pracy wykorzystywać pozostałe.

A oto ich podstawowe cechy:

INGRES/MENU

INGRES/MENU jest modułem tworzącym jednorodne zasady współpracy użytkownika ze wszystkimi modułami środowiska INGRES. Współpraca ta jest oparta o serię wzorców (forms) i list wyboru.

INGRES/ISQL

INGRES/ISQL jest procesorem języka SQL (Structured Query Language) zgodnym ze standardem ANSI Level II SQL, zapewniającym interakcyjny dostęp do serwera bazy danych. Umożliwia on tworzenie, konserwację i administrację bazy danych oraz dostęp i modyfikację informacji w niej zawartych.

INGRES/FORMS

jest ekranowym edytorem (Visual-Forms-Editor) przeznaczonym do projektowania wzorców (forms), które mogą być wykorzystywane przez pozostałe modułu środowiska INGRES.

INGRES/QUERY

INGRES/QUERY jest procesorem języka Query-by-Forms, którego operacje są definiowane w postaci tabelarycznej z wykorzystaniem wzorców. Przeznaczony jest głównie do kierowania do bazy danych żądań ad hoc dotyczących tablic (tables), spojrzeń na bazę (views) oraz relacji wirtualnych powstających w wyniku operacji łączenia (join). Może być również wykorzystywany do wprowadzania, modyfikacji i usuwania rekordów, jeśli użytkownik systemu posiada odpowiednie uprawnienia nadane mu przez administratora bazy danych. Wyniki żądań są

najczęściej prezentowane w postaci predefiniowanych wzorców tworzonych przez system na bazie budowy tablic. Istnieje jednak możliwość wykorzystywania specjalnie przygotowanych wzorców modułem INGRES/FORMS.

INGRES/REPORT

INGRES/REPORT jest procesorem języka Report-by-Forms służącym do przygotowywania raportów w oparciu o informacje przechowywane w bazie danych. Zapewnia on trzy poziomy tworzenia raportów. Pierwszy z nich bazuje na predefiniowanej strukturze raportów określanej przez budowę tablic lub spojrzeń na bazę. Drugi poziom umożliwia zmianę budowy raportu oraz uporządkowania prezentowanych danych zgodnie z potrzebami użytkownika. Poziom trzeci jest specjalizowanym językiem opisu raportów przeznaczonym głównie dla programistów, dający im pełne możliwości dopasowania raportu do wymagań systemu informatycznego.

INGRES/Simplify

INGRES/Simplify jest procesorem interakcyjnego i wizualnego języka obsługi dostępu użytkowników do relacyjnej bazy danych. Język ten wykorzystuje graficzne zasady współpracy z systemem (Graphical User Interface) zgodne ze standardem OPEN LOOK. Produkt ten w znacznym zakresie ułatwia współpracę z bazą danych, przyspiesza przygotowywanie żądań i raportów oraz usprawnia zarządzanie bazą danych. Użytkownicy systemu mają natychmiastowy obraz tego, jakie dane są dostępne, jak są ze sobą powiązane i w jaki sposób najlepiej je wykorzystać.

INGRES/Simplify składa się z trzech elementów: DataBrowse - wizualnego edytora żądań i prezentera danych; ReportWrite, interakcyjnego edytora raportów; SchemaDesing, edytora schematów bazy danych.

INGRES/GRAPHICS

INGRES/GRAPHICS jest procesorem języka Graphics-by-Forms przeznaczonym do prezentacji w formie graficznej informacji przechowywanej w bazie danych. Operacje języka definiuje się w postaci tabelarycznej za pomocą wzorców i list wyboru. Pozwalają one na przedstawianie danych w postaci wykresów słupkowych (bar charts), wykresów kołowych (pia charts), wykresów liniowych (line charts) i wykresów punktowych (scatter charts). Innymi elementami i parametrami definiowanymi przez użytkownika są: tytuły i legendy, typy punktów i linii, fonty liter, skale wykresów itd. Graficzne formy prezentacji danych mają istotne znaczenie w informatycznych systemach zarządzania oraz systemach wspomagania podejmowania decyzji ze względu na swoją przewagę nad formami opartymi o surowe dane w postaci tabel lub raportów.

INGRES/ESQL

INGRES/ESQL jest rodziną preprocesorów języków programowania trzeciej generacji, umożliwiających wykorzystywanie języka SQL do współpracy z bazą danych. Pozwalają one również na pełną integrację programów aplikacyjnych, przygotowanych w tych językach, ze wzorcami opracowanymi modułem INGRES/FORMS. Obecnie dostępne są procesory następujących języków programowania: C, COBOL, FORTRAN, PASCAL, BASIC, PL/1 i ADA.

INGRES/APPLICATIONS 4GL

INGRES/APPLICATIONS 4GL jest procesorem języka programowania Applications-by-Forms umożliwiającym tworzenie, modyfikację i uruchamianie programów aplikacyjnych wykorzystujących do współpracy z użytkownikiem wzorce i listy wyboru. Język ten jest uzupełnieniem innych modułów środowiska INGRES wykorzystujących do współpracy z użytkownikiem wzorce i listy wyboru. Język ten jest uzupełnieniem innych modułów środowiska INGRES wykorzystujących języki programowania tabelarycznego (INGRES/QUERY, INGRES/REPORTS, INGRES/GRAPHICS) i pozwala na ich pełną integrację w ramach jednego programu aplikacyjnego.

INGRES/APPLICATIONS 4GL wykorzystuje język programowania czwartej generacji OSL (Operation Specification Language). Jest rozwinięciem języka SQL o elementy związane ze współpracą z systemem wzorców, kontrolą przebiegu zadania i cechy języków programowania proceduralnego i nieproceduralnego. Umożliwia również włączenie procedur przygotowanych w językach programowania trzeciej generacji.

INGRES/WINDOWS 4GL

INGRES/WINDOWS 4GL jest najnowszym produktem firmy INGRES Corporation przeznaczonym do tworzenia oprogramowania aplikacyjnego wykorzystującego graficzne zasady współpracy użytkownika z relacyjną bazą danych. Produkt ten jest rodziną wzajemnie uzupełniających się i zintegrowanych narzędzi koniecznych do przygotowania programów aplikacyjnych. Obejmują one wizualne edytory przeznaczone do tworzenia obiektów wykorzystywanych do współpracy użytkownika z systemem (okna, menu, wzorce itd.) oraz zorientowany obiektowo język programowania 4. generacji INGRES/4GL. Środowisko programowania INGRES/Windows 4GL jest przygotowane do współpracy z najbardziej rozpowszechnionymi systemami okien. Należą do nich: DECWindows, OSF/Motif, OpenWindows (OPEN LOOK), Windows/386, OS/2 Presentation Manager i Macintosh.

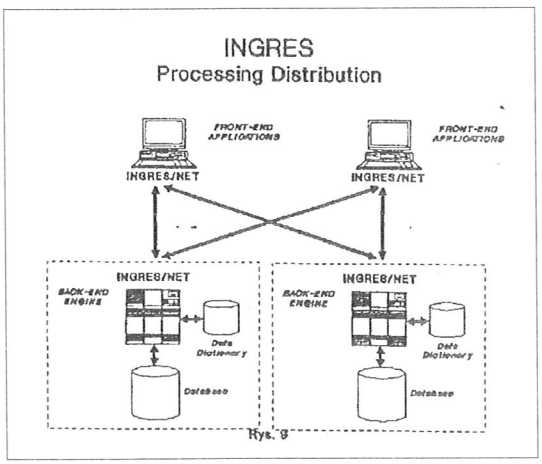
5. Przetwarzanie rozproszone środowiska INGRES

INGRES/NET

Szybki i niezawodny dostęp do informacji, bez względu na to gdzie się ona znajduje, jest bardzo istotny w dzisiejszym szybko rozwijającym się i konkurencyjnym świecie biznesu. Zbudowanie takiego systemu informatycznego, który zaspokajałby te wymagania, nie jest jednak proste. Dane wykorzystywane przez system obsługi działalności różnych organizacji i przedsiębiorstw są bardzo często przechowywane w odmiennych bazach danych. Bazy te pracują na różniących się systemach komputerowych i są najczęściej połączone różnymi typami sieci komputerowych.

Wykorzystywana do chwili obecnej architektura systemów informatycznych, oparta o model klient/serwer i zabezpieczająca proste połączenia pomiędzy aplikacjami klientów i serwerem bazy danych, nie jest wystarczająca. Systemy informatyczne wymagane obecnie muszą zapewniać możliwości współpracy wielu użytkowników z niejednorodnymi bazami danych pracujących na różnych systemach komputerowych poprzez wiele odmiennych połączeń komunikacyjnych.

INGRES-NET jest elementem środowiska zarządzania relacyjną bazą danych INGRES odpowiedzialnym za implementację przetwarzania rozproszonego. Do jego zadań należy przyjmowanie zleceń języka SQL, przekazywanie ich do wymaganego serwera bazy danych, a następnie dostarczanie wymaganych informacji do klientów (rys. 9).

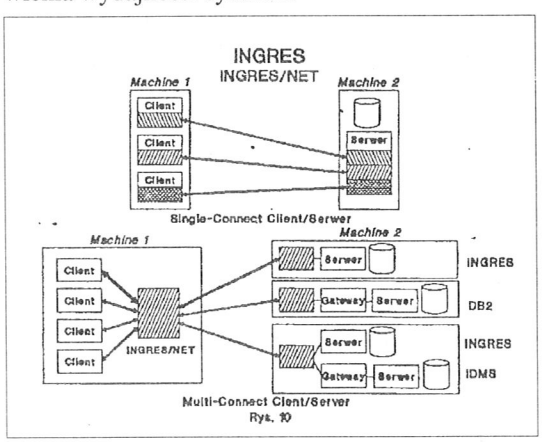


Rys. 9

Podstawowymi cechami przetwarzania rozproszonego modułu INGRES/NET są:

- wielowątkowy serwer komunikacyjny (Multi-client/Multi-server Communication),
- współbieżność obsługi połączeń komunikacyjnych i sieciowych (Multi-network Connections),
- bogata lista obsługiwanych środowisk operacyjnych i protokołów komunikacyjnych,
- pełna zgodność ze standardami ISO określonymi dla przetwarzania rozproszonego (RDA Remote Data Access),
- możliwość współdziałania z elementami INGRES/GATEWAY i INGRES/STAR,
- pełna "przezroczystość" połączeń sieciowych obejmująca formaty danych oraz wybór wymaganego serwera bazy danych.

Moduł INGRES/NET jest wielowątkowym serwerem komunikacyjnym pozwalającym na współpracę wielu użytkowników i programów aplikacyjnych z wieloma serwerami bazy danych. Dla każdego aktywnego serwera klienta tworzony jest niezależny wątek obsługujący połączenia klient/serwer (rys. 10). W sytuacji gdy praca systemu wymaga bardzo intensywnego wykorzystywania połączeń komunikacyjnych, administrator bazy danych może uruchomić wiele serwerów komunikacyjnych INGRES/NET w celu poprawienia wydajności systemu.



Rys. 10

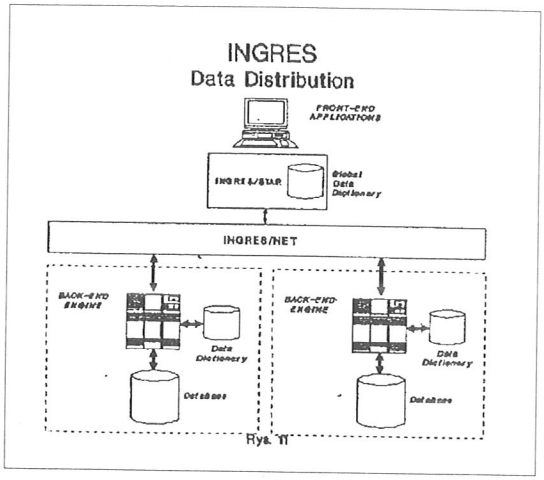
Bardzo ważną cechą modułu INGRES/NET jest możliwość jednoczesnej współpracy z wieloma serwerami bazy danych. Mogą być one bazami danych INGRES, rozproszonymi bazami danych INGRES oraz bazami danych innych producentów. Współpraca ta jest całkowicie niezauważalna dla użytkownika systemu, który może traktować wszystkie dane dostępne w systemie jako jednorodny zasób informacyjny.

Kolejną istotną cechą modułu INGRES/NET jest bardzo bogata lista obsługiwanych połączeń komunikacyjnych i sieciowych. Należą do nich: TCP/IP,OSI,DECnet, PC LAN, NetBIOS, X25, SNA oraz połączenia asynchroniczne.

Środowisko INGRES pracujące z modułem INGRES/NET zapewnia administratorowi bazy danych pełną kontrolę nad uprawnieniami dostępu do baz danych, systemów komputerowych oraz połączeń sieciowych. Ma on do dyspozycji specjalne programy usługowe służące do definiowania konfiguracji sieci komputerowej, położenia i typów baz danych oraz uprawnień dostępu dla użytkowników systemu.

INGRES/STAR

INGRES/STAR jest serwerem rozproszonej bazy danych środowiska INGRES. Pozwala on

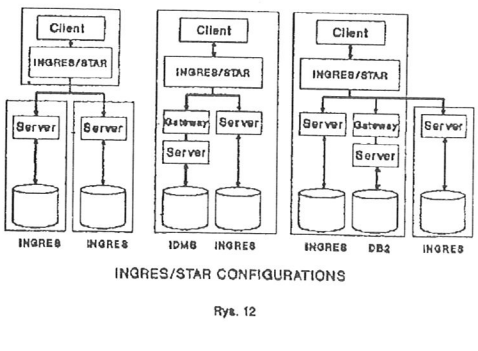


Rys. 11

użytkownikom systemu i twórcom programów aplikacyjnych na traktowanie informacji dostępnych w systemie jako jednorodnego zasobu systemowego. Zasób ten mogą tworzyć dane znajdujące się w lokalnych i zdalnych bazach danych INGRES oraz dane przechowywane w bazach danych innych producentów, dla których są dostępne bramy środowiska INGRES - INGRES/GATEWAY. Poszczególne bazy danych zachowują swoją autonomię, a zakres informacji udostępnianych innym użytkownikom jest pod pełną kontrolą administratora systemu (rys. 11).

Podstawowymi cechami serwera rozproszonej bazy danych INGRES/STAR są:

- pełna integracja z modułami INGRES/NET i INGRES/GATEWAY,
- obsługa rozproszonych żądań dostępu obejmujących wiele baz danych (Multisite Distributed Queries),
- obsługa rozproszonych transakcji obejmujących wiele baz danych z wykorzystaniem protokołu Two-phase Commit (Multisite Distributed Transactions),
- pełna kontrola funkcjonowania rozproszonej bazy danych z wykorzystaniem INGRES/STARView monitora serwera bazy danych.



Rys. 12

Serwer rozproszonej bazy danych INGRES/STAR może pracować na różnych platformach systemowych i w różnych środowiskach operacyjnych. Jest przygotowany do jednoczesnej współpracy z wieloma lokalnymi i zdalnymi bazami danych (rys. 12). Rozproszona baza danych utworzona w oparciu o INGRES/STAR posiada identyczne cechy jak zwykła baza danych INGRES. Mogą z nią współpracować wszystkie elementy środowiska INGRES dostępne dla użytkowników i twórców oprogramowania aplikacyjnego.

INGRES/GATEWAY

INGRES/GATEWAY są elementami systemu zarządzania bazą danych, które umożliwiają wykorzystywanie informacji przechowywanych w odmiennych niż INGRES bazach danych przez użytkowników i programy aplikacyjne środowiska INGRES. Elementy INGRES/GATEWAY zapewniają:

- "przezroczysty" dostęp do informacji przechowywanych w bazach danych odmiennych niż INGRES,
- automatyczną konwersację typów danych obowiązujących we współdziałających środowiskach operacyjnych,
- dostępność interfejsu dla wszystkich narzędzi i programów aplikacyjnych "front-end'u",
- pełną integrację z elementami INGRES/NET i INGRES/STAR,
- ścisłą współpracę z systemami obsługi odmiennych baz danych pod względem kontroli współbieżności dostępu, obsługi transakcji współpracy, zapewnienia bezpieczeństwa i poufności danych.

INGRES/GATEWAY pracuje pomiędzy narzędziami "front-end'u" i serwerem współpracującej bazy danych. Żądania dostępu są przekazywane do INGRES/GATEWAY w języku INGRES/OPEN SQL. Sa one następnie przekształcane do postaci wymaganej przez serwer obsługiwanej bazy danych i przekazywane do wykonania. Rezultaty żądań są zwracane do modułów "front-end'u" tą samą drogą, a INGRES/GATEWAY odpowiada za przeprowadzenie wymaganych konwersji typów danych oraz obsługę sytuacji wyjątkowych.

Obecnie dostępne są moduły INGRES/GATEWAY dla następujących baz danych:

- RMS i Rdb firmy DEC,
- DB2,SQL/DS i IMS firmy IBM,
- IDMS firmy ICL,

oraz baz danych takich firm jak Oracle, Teradata, Tandem oraz Data General.

INGRES/PCHOST i INGRES/PCLINK

INGRES/PCHOST i INGRES/PCLINK są współdziałającymi elementami środowiska INGRES umożliwiającymi pełną integrację użyt-

kowników systemów mikrokomputerowych IBM PC iPS/2 z systemem zarządzania relacyjnej bazy danych. Współpraca obu środowisk wykorzystuje asynchroniczne łącza komunikacyjne i obejmuje:

- emulację terminal obsługi systemu relacyjnej bazy danych
- specjalizowany język żądań dostępu do bazy danych (Visual Query Language),
- programy usługowe przesyłania plików pomiędzy systemem obsługi baz danych i systemami mikrokomputerowymi.

Użytkownik systemu mikrokomputerowego, wykorzystujący emulator terminala systemu, ma pełny dostęp do wszystkich elementów współpracy z bazą danych środowiska INGRES. System, na którym pracuje, jest traktowany przez środowisko operacyjne serwera bazy danych jako kolejny terminal systemu komputerowego.

Specjalizowany język żądań dostępu do bazy danych jest "wizualnym" językiem programowania tabelarycznego prezentującym dane w postaci zbliżonej do arkusza elektronicznego. Dane te mogą być przedstawione na monitorze terminala, wprowadzane na urządzenia drukujące lub przekazywane w różnych formatach do plików systemu mikrokomputerowego. Dostępne formaty zapisu tych plików pozwalają na wykorzystywanie otrzymanych danych w popularnych pakietach kalkulacyjnych (LOTUS 1-2-3, Multiplan), bazach danych (dBASE III+, dBASE IV), procesorach tekstów oraz pakietach graficznych.

Miejsce na notatki

dr inż.
Wacław
Iszkowski

Relacyjna baza danych ORACLE

Informacja o produkcie ORACLE

Firma Oracle Corporation jako pierwsza w świecie zaimplementowała w 1979 roku System Zarządzania Relacyjną Bazą Danych (SZRBD) ORACLE (RDBMS - Relational Data Base Management System) z wykorzystaniem języka SQL. W 1983 roku baza ORACLE działała już na wielu dużych komputerach, stacjach roboczych oraz minikomputerach i mikrokomputerach personalnych.

Podstawowym elementem produktu oferowanego przez Oracle Corporation jest SZRBD ORACLE będący uniwersalnym systemem tworzącym i bezpośrednio obsługującym środowisko relacyjnej bazy danych z poziomu językaSQL. We wszystkich wersjach pakiet ten poprzez akceptację instrukcji standardu języka SQL umożliwia przenośność aplikacji pomiędzy różnymi typami komputerów i systemów operacyjnych. SZRBD ORACLE jest bardzo efektywnym wielodostępowym systemem o następujących cechach - minimalizowanie zakresu blokowania pól aktualizowanych zapisów, optymalizacja operacji wejścia-wyjścia oraz efektywny mechanizm odporności na awarie i z odpowiednimi zabezpieczeniami przed nieuprawnionym dostępem.

Obecnie SZRBD jest dostępny na około 80 różnych platformach czyli komputerach z systemem operacyjnym - poczynając od prostego IBM PC/AT (z 2.5 MB RAM) z systemem MS-DOS lub OS/2, poprzez stacje robocze SUN, microVAX i DECstation, minikomputery DEC-system, Data General, Philips, Bull DPX, ICL DRS, UNISUS z systemem UNIX, aż do dużych maszyn VAX 4000, 6000 i 9000, IBM 43xx oraz CDC i różnego typu komputerach równoległych takich jak Pyramid i Sequent. SZRBD ORACLE może funkcjonować w środowisku wielu systemów operacyjnych - na przykład w około 60 systemach - pochodnych systemu UNIX.

Jednocześnie produkt ORACLE jest zgodny ze standardami określonymi dla baz danych. Wykorzystuje on język zapytań SQL (Structured Query Language) będący standardem przemysłowym, przyjętym przez ANSI oraz ISO. Oracle Corp. jest też członkiem SQL-Acces Group (grupującej takie firmy jak Ashton-Tate, Digital Equipment, Fujitsu America, Hewlett-Packard, Informix, Ingres, Metaphore, NCR, Sun Microsystems, Tandem Computers oraz Teradata) mającej na celu stałą pracę nad rozszerzaniem standardu SQL oraz ujednolicaniem formatu i struktury relacyjnej bazy danych. Działania takie mają na celu uproszczenie w przyszłości dostępu do różnych baz danych w ramach rozległych sieci komputerowych. Podobne działania Oracle Corp. prowadzi w dziedzinie opracowywania nowych systemów operacyjnych - jak na przykład projektu OSF (Open Software Foundation).

Pakiety narzędziowe ORACLE'a

Dostęp do bazy danych uzyskuje się poprzez instrukcje języka SQL wprowadzane interakcyjnie lub z poziomu programów tworzących aplikację. Dla ułatwienia dostępu i implementacji aplikacji Oracle dostarcza zestaw pakietów narzędziowych:

SQL*Plus - Pakiet obsługi interakcyjnie wprowadzanych poleceń języka zapytań SQL (zgodnego ze standardem ANSI), pozwalający na bezpośredni dostęp do bazy danych.

SQL*Forms - Pakiet interakcyjnego wspomagania użytkownika w generowaniu formatek aplikacji w języku 4 poziomu bez potrzeby samodzielnego pisania programów.

SQL*Menu - Pakiet do projektowania i generacji dynamicznych menu upraszczających obsługę tworzonej aplikacji, również bez konieczności pisania programów.

SQL*Report -Pakiet obsługi interakcyjnie żądanych raportów tworzonych z informacji przechowywanych w bazie danych.

SQL*ReportWriter - Pakiet generatora programów obsługi raportów, sterowany przez menu, umożliwiający zaprojektowanie złożonych raportów bez konieczności pisania programów.

SQL*TextRetrieval - Pakiet rozszerzający cechy funkcjonalne aplikacji o możliwość zarządzania dowolnymi tekstami (indeksowanie, wyszukiwanie, przetwarzanie) związanymi z informacjami zapisanymi w bazie danych.

ORACLE for dBASE Interpreter Compiler (Quicksilver) - Pakiet kompilatora języka dBase III Plus.

SQL*Loader - Pakiet konwersji plików z danymi w innych formatach (np. Lotus 1-2-3 oraz dBASE III Plus) do formatu bazy danych ORACLE.

SQL*CALC - Pakiet arkusza kalkulacyjnego, wykorzystujący informacje zawarte w bazie danych.

SQL*Graph - Pakiet pozwalający na graficzną

prezentację informacji generowanych w postaci raportów na podstawie danych zawartych w bazie ORACLE.

SQL*Net - Pakiety umożliwiające komunikację pomiędzy rozproszonymi bazami danych.

SQL* Connect - Pakiet umożliwiający współpracę ORACLE'a z innymi bazami danych jak na przykład DB2, SQL/DS i RMS.

Implementacja aplikacji dla ORACLE'a.

Wymienione pakiety narzędziowe umożliwiają interakcyjne zaimplementowanie aplikacji wykorzystującej SZRBD ORACLE. Uzyskana aplikacja zostaje zapisana w formie interpretowanej. Do jej wykorzystywania konieczne jest stałe istnienie pakietów SQL*Forms, SQL*ReportWriter i SQL*Menu. Aplikacja ta może być przenoszona w inne środowisko SZRBD ORACLE, działające z innym systemem operacyjnym i na innym sprzęcie. Inną metodą implementacji aplikacji, jest jej napisanie w jednym ze standardowych języków programowania (np. C, Pascal, FORTRAN, COBOL, ADA) przy pełnym wykorzystaniu instrukcji języka SQL, wkładanych bezpośrednio w tekst programów. Oracle Corp. dostarcza dla tych języków pakiety prekompilatorów PRO*C, PRO*Pascal, itp. dokonujące wstępnego przekształcenia instrukcji SQL w wywołania procedur w konwencji danego języka programowania. Tak zaimplementowana aplikacja może funkcjonować tylko z samym SZRBD ORACLE. Dopuszcza się też uzupełnienie aplikacji wytworzonej interakcyjnie o fragmenty napisane w języku programowania.

Połączenia baz ORACLE

Dołączenie do pakietu SZRBD ORACLE pakietu SQL*Net umożliwia wykorzystanie różnych protokołów komunikacyjnych (np. DECnet,TCP/IP, SNA, Novell SPX, Named Pipes, NetBIOS, itp.) do użytkowania bazy danych rozmieszczonych na różnych platformach. Baza taka jest dla każdego z użytkowników zintegrowana logicznie. Należący do tej samej rodziny pakiet SQL*Connect zezwala za pośrednictwem środowiska ORACLE'a dostęp do innych baz (np. DB2,SQL/DS, i RMS). Najnowszym dodatkowym elementem SZRBD ORACLE jest pakiet SQL*Star tworzący zintegrowany otwarty system rozproszonej bazy danych rozmieszczonej na komputerach o różnej architekturze i systemach operacyjnych.

Narzędzia metody CASE

Firma Oracle Corp. opracowała również pakiety wspomagające projektowanie i implementację złożonych aplikacji, korzystając ze standardowej metody inżynierii programowania. W najprostszym rozumieniu metoda ta jest przepisem, jak w uporządkowany sposób należy wykonywać poszczególne kroki projektowe, aby z najmniejszymi nakładami uzyskać poprawną aplikację, bez konieczności wracania z etapu wstępnej eksploatacji do etapu analizy czy wręcz weryfikacji strategii. Zastosowanie tej metody pozwala wielokrotnie przyśpieszyć prace nad zrealizowaniem nowej aplikacji. Dodatkowo Oracle oferuje narzędzia wspomagające stosowanie tej metody:

CASE*Designer - Pakiet do projektowania pełnego graficznego modelu aplikacji bazodanowej w formie graficznej, łącznie z możliwością weryfikacji projektu poprzez analizę diagramów zależności pomiędzy obiektami bazy, hierarchiami funkcji, przepływu danych opisów macierzowych.

CASE*Dictionary - Pakiet tworzenia i zarządzania słownikiem przechowującym informacje o tworzonej aplikacji w każdym z kroków metody CASE z opisami na modelu aplikacji na poziomie fizycznym i logicznym.

CASE*Generator - Pakiet generacji gotowej aplikacji w konwencji SQL*Forms na podstawie jej opisu zawartego w słowniku.

Ten skrótowy opis nie wyczerpuje wszystkich cech funkcjonalnych oprogramowania oferowanego przez Oracle Corp. Oprogramowanie to wprowadza bowiem do praktycznego wykorzystania nowoczesne techniki projektowania i implementacji aplikacji, przyśpieszając sam proces tworzenia oraz upraszczając późniejszą eksploatację.

Narodowa wersja językowa

Produkty ORACLE mogą być skonfigurowane w danej wersji językowej. Przy zachowaniu standardu języka SQL komunikaty informacyjne oraz sygnalizujące błędy, a także format daty oraz sekwencja uporządkowania znaków może być zrealizowana w określonym języku - francuskim, niemieckim. Wszystkie komunikaty ORACLE'a są umieszczone w osobnych plikach. Również kody dodatkowych liter alfabetu mogą wykorzystywać wartości 8-bitowe uwzględniane w porządku sortowania. Nacjonalizacja systemu ORACLE może być jednak dokonana jedynie przez firmę ze względu na konieczność wewnętrznej zmiany tabeli określającej kolejność uporządkowania znaków.

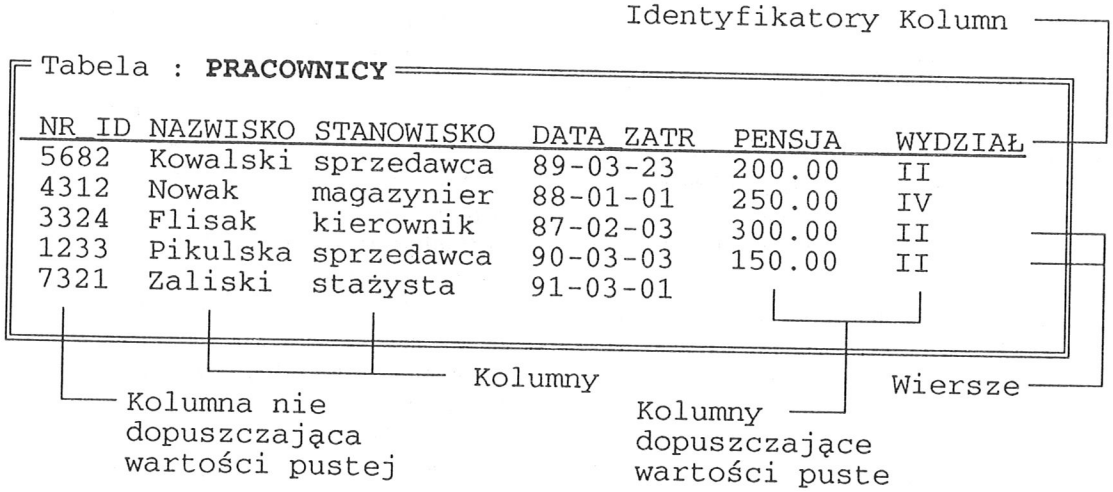
2. Architektura bazy danych ORACLE

Baza danych

Baza danych ORACLE jest zbiorem danych traktowanych jako całość przechowywanym w trzech typach plików, będących jednostkami bazy danych:

* Obiekty bazy danych (database files) przechowujące gromadzone informacje oraz definicje obiektów: tabel, projekcji, indeksów i klastrów,
* Dzienniki (redo log files) historii operacji na bazie danych, zawierające informacje niezbędne do odzyskania bazy po awarii,
* Informacje kontrolne (control files) o strukturze bazy danych.

Przykład III.1 : Tabela.



Tabele

Tabela (table) identyfikowana w bazie danych unikalnym identyfikatorem jest zbiorem kolumn przechowującym informacje tego samego rodzaju. Wiersz tabeli tworzy rekord zawierający zbiór informacji dotyczącej tej samej rzeczy. Niektóre pozycje wiersza w danych kolumnach mogą być puste (są wtedy oznaczone jako NULL), które nie są wtedy przechowywane w pliku baz danych. Po zdefiniowaniu tabeli można do niej wstawić, usuwać i aktualizować poszczególne jej wiersze oraz zadawać zapytania (querries).

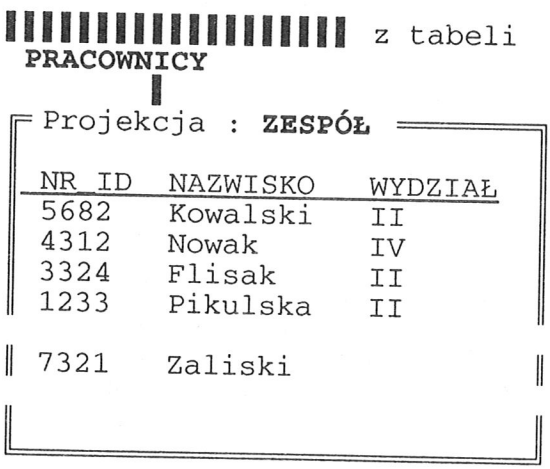
Projekcje

Projekcja (view) jest prezentacją danych zawartych w jednej lub wielu tabelach, dostosowaną do danej aplikacji. Projekcja sama nie zawiera danych, wykorzystując dane umieszczone w tabelach bazowych. Do projekcji, identyfikowanych w bazie danych unikalną nazwą, można zadawać zapytania oraz można ją aktualizować aktualizując tym samym tabele bazowe.

Projekcje są najczęściej wykorzystywane do:

* wprowadzania dodatkowego poziomu zabezpieczania dostępu do niektórych kolumn tabeli z danymi,

Przykład III.2 : Projekcja.



* uproszczenia dostępu do danych zawartych w kilku tabelach, poprzez złączenie ich w jednej projekcji,
* upraszczania instrukcji zapytania lub aktualizacji danych,
* prezentacji danych w innej postaci, kolejności kolumn czy ich identyfikacji,
* składowania wyliczonych wartości na podstawie zawartości danych w tabelach.

Indeksy

Indeksy (indexes) są opcjonalnymi strukturami stowarzyszonymi z tabelami i klastrami. Przyśpieszają one wyszukiwanie informacji w

Tabela : **PRACOWNICY**

NR_ID	NAZWISKO	STANOWISKO	DATA_ZATR	PENSJA	WYDZIAŁ
5682	Kowalski	sprzedawca	89-03-23	200.00	II
4312	Nowak	magazynier	88-01-01	250.00	IV
3324	Flisak	kierownik	87-02-03	300.00	II

Przykład III.3 : Indeksy.

Tabela : **PRACOWNICY**

NR_ID	NAZWISKO	STANOWISKO	DATA_ZATR	PENSJA	WYDZIAŁ
5682	Kowalski	sprzedawca	89-03-23	200.00	II
4312	Nowak	magazynier	88-01-01	250.00	IV
3324	Flisak	kierownik	87-02-03	300.00	II
1233	Pikulska	sprzedawca	90-03-03	150.00	II
7321	Zaliski	stażysta	91-03-01		

Unikalne indeksy

Tabela : **WYDZIAŁY**

WYDZIAŁ	NAZWA	LOKALIZACJA
I	Sprzedaż	Warszawa
II	Laboratorium	Radom
III	Produkcja	Grójec
IV	Magazyn	Janki

Pierwotne klucze

Zewnętrzne klucze

tabelach po zadaniu zapytania. Dodatkowo mogą one weryfikować unikalność każdej wartości w danej kolumnie.

Przy omawianiu indeksów warto zwrócić uwagę na różnicę pomiędzy:

* indeksami będącymi logicznym uporządkowaniem danych umieszczonych w jednej lub wielu kolumnach istniejącej tabeli, a tworzonymi dynamicznie instrukcjami SQL,
* kluczami pierwotnymi (primary keys) będącymi kolumną lub grupą kolumn o wartościach identyfikujących jednoznacznie każdy wiersz tabeli,
* kluczami zewnętrznymi (foreign keys) będącymi kolumną lub grupą kolumn odpowiadających kluczom pierwotnym w innej tabeli, a wykorzystywanymi w załączeniu tabel.

Klastry

Klaster (cluster) jest opcjonalnym obiektem bazy danych grupującym fizycznie dwie lub więcej tabel z pojedynczym zapisem wspólnej kolumny z kluczami zewnętrznymi. W klastry są organizowane grupy tabel, które powinny być przechowywane razem, ponieważ zawierają wspólne kolumny. Dane z tych kolumn są przechowywane jeden raz. Istnieją również indeksy wskazujące wiersze z wszystkich tabel należących do tego samego klastra.

Słownik danych

Informacje o obiektach bazy danych są przechowywane w Słowniku Danych (Data Dictionary). Słownik ten jest również zbiorem tabel i projekcji dostępnych dla administratora systemu oraz użytkownika systemu standardowo poprzez zapytania instrukcjami SQL. Przykładowo w słowniku są przechowywane :

* nazwy i hasła wszystkich zarejestrowanych użytkowników,
* przywileje użytkowników (profile),
* nazwy wszystkich obiektów bazy,
* domniemane wartości dla kolumn,
* rozmiar pamięci przydzielonej użytkownikowi,
* informacje o śledzeniu operacji.

Dziennik

Dziennik (Redo Log) ORACLE'a rejestruje wszystkie zmiany dokonywane w bazie przez

użytkowników. Zmiany te są rejestrowane natychmiast po ich wprowadzeniu, co umożliwia odzyskanie danych w przypadku awarii systemu. ORACLE przyjmuje, że aktualną zawartością bazy danych jest zawartość wszystkich tabel łącznie z zapisaną w dzienniku ich dotychczasową aktualizacją. Aktualizacja plików danych w tabelach następuje w tle łącznie z uwzględnieniem synchronizacji i aktualizacją indeksów.

System ma co najmniej dwa pliki dzienników, z których jeden jest aktualnie wypełniany. Rozpoczęcie wypełniania dziennika oznacza zaznaczenie punktu kontrolnego stanu bazy. Pozwala to na łatwe odtworzenie stanu bazy danych po awarii.

3. Organizacja bazy danych ORACLE

Struktury logiczne pamięci

Informacje bazy danych są zapisywane w jednolitym środowisku umieszczonym w plikach pamięci masowej i częściowo skopiowanych do buforów pamięci operacyjnej. Logicznie pamięć bazy danych jest podzielona na:

* partycje
* segmenty
* rozpiętości

Partycje

Partycja (tablespace, partition) jest logiczną jednostką pamięci wykorzystującą jeden lub więcej plików danych. Partycja może być wykorzystana do:

* kontroli zajętości dysku dla danych baz danych,
* przypisywania użytkownikom określonych wielkości pamięci na dane,
* kontroli dyspozycyjności danych przez kontrolę bezpośredniego lub pośredniego dostępu,
* zmniejszenia obciążenia operacjami składowania i odtwarzania,
* rozmieszczania danych na kilku urządzeniach dla zwiększenia wydajności systemu.

W każdym SZRBD ORACLE istnieje partycja SYSTEM, w której oprócz tabel administracji bazy danych mogą istnieć tabele z danymi użytkownika. Oprócz partycji SYSTEM można zadeklarować inne partycje deklarując ich rozmiar i przypisując im odpowiednie pliki.

Segmenty i rozpiętości

Segmenty (segments) są zbiorem bloków alokowanych dla danych, którymi mogą być dane z tabel, indeksów lub zmiennych roboczych. Segment może należeć tylko do jednej partycji, ale może być przechowywany w kilku plikach.

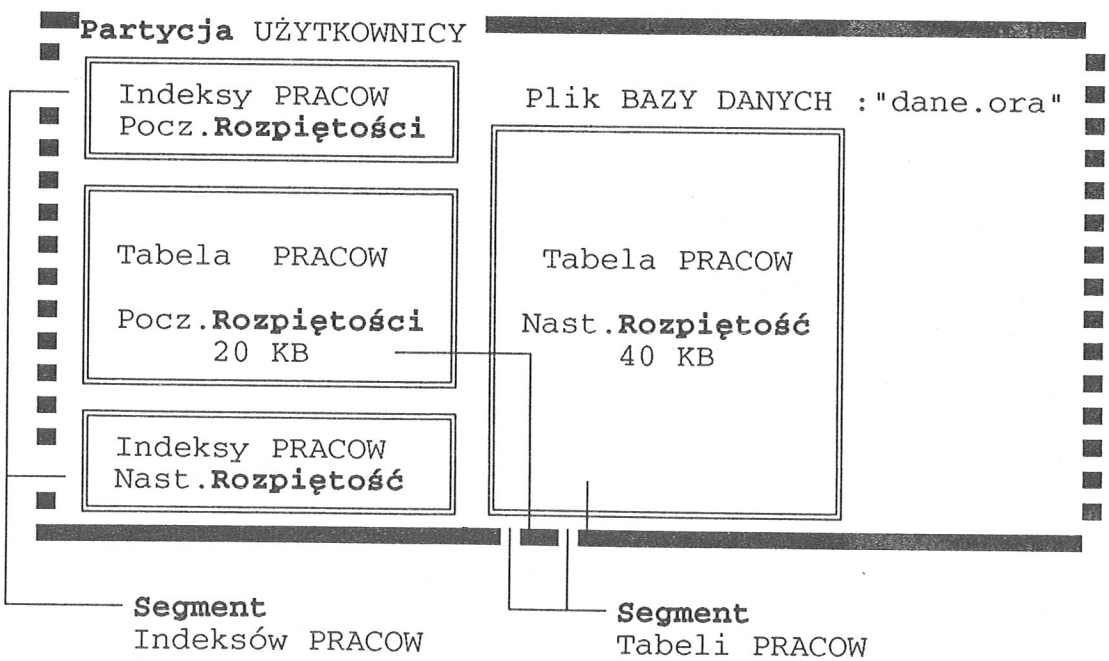
Rozpiętość (extent) określa wielkość ciągłego obszaru alokacji przestrzeni bazodanowej, wyrażonej w liczbie bloków danych.

Większość segmentów jest tworzona z początkową rozpiętością przestrzeni i rozszerzaną rozpiętością dynamiczną przydzieloną na żądanie.

Na rysunku jest pokazana zależność pomiędzy partycją, segmentem i rozpiętością.

Pliki systemu ORACLE

SZRBD ORACLE wykorzystuje i administruje trzema rodzajami plików:



- plikami bazy danych (od 1 do 255),
- plikami dziennika (minimum 2)
- plikami informacyjnymi (minimum 1)

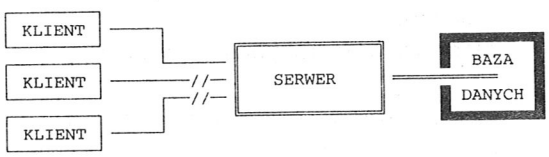
Plik bazy danych przechowuje dane będące elementami tabel, projekcji, indeksów i klastrów. Plik ten może być związany tylko z jedną partycją bazy danych. Raz utworzony, plik nie zmienia swojej długości.

4. Architektura systemu ORACLE

Architektury SZRBD

Istnieją dwa rozwiązania umożliwiające dostęp wielu użytkowników do tej samej bazy danych.

* W tradycyjnej architekturze Wielu_Klientów-Jeden_Serwer istnieje jeden tylko jeden proces serwera obsługujący wielu klientów według określonego algorytmu obsługi. Rozwiązanie to ogranicza równoległość funkcjonowania systemu, w przypadku większej liczby zgłoszeń klientów powoduje



Wielu Klientów - Jeden Serwer.

szybkie zmniejszenie wydajności a zwiększanie czasu odpowiedzi.

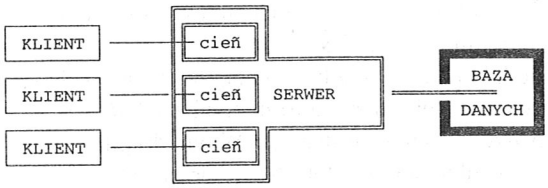
* W architekturze ORACLE'a Wiele_Par Klient-Serwer 2 w serwerze istnieje tyle procesorów obsługi (cieni) ilu jest klientów.

W zależności od architektury systemu cyfrowego pary zadań klient-obsługa są realizowane przez jeden lub dwa procesory. W systemach sieciowych para ta jest rozmieszczona w odrębnych systemach - klient w stacji Klienta, a obsługa (cień) w stacji Serwera. Komunikacja pomiędzy nimi odbywa się poprzez pakiety SQL*Net.

Architektura systemu ORACLE

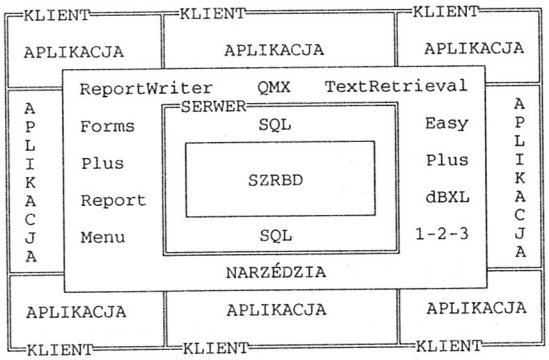
Podstawowa struktura systemu ORACLE jest złożona z:

* Serwera (Serwer) zawierającego System Zarządzania Relacyjną Bazą Danych (SZRBD) oraz procesor języka SQL,
* Klienta (klientów) (Client) obejmującego części narzędzi ORACLE przeznaczone do interpretacji tekstu aplikacji użytkownika.



Wiele Par Klient - Serwer.

Oba elementy struktury ORACLE'a mogą tworzyć jeden wspólny obiekt lub też być rozproszone.



Wspólny obiekt Klient - Serwer.

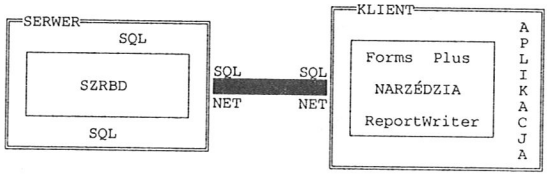
SZRBD-System Zarządzania Relacyjną Bazą Danych Oracle (RDBMS-Relational Data Base Management System) jest jądrem systemu. Zajmuje się on między innymi:

- administrowaniem bazy danych,
- realizacją wszystkich operacji na bazie danych (tworzeniem tabel, uzyskiwaniem odpowiedzi na zapytania, aktualizacją itp.)
- okresowym składowaniem i odtwarzaniem,
- zabezpieczeniem dostępu przed nieuprawnionym użytkownikiem.

Do komunikacji z otoczeniem SZRBD wykorzystywany jest procesor języka zapytań SQL(SQL-System Querry Language). Język ORACLE SQL jest nadzbiorem języka SQL zdefiniowanego w standardzie ISO 9075 ("Information Processing Systems-Database Language SQL). Trzeba jednak tutaj stwierdzić, że liczba rozszerzeń jest dość znaczna (np. słów zarezerwowanych jest 115 w stosunku do 56 w standardzie). Takie rozbudowanie języka ORACLE SQL ułatwia programowanie aplikacji, ale utrudnia jej przenoszalność w środowisko innych SZRBD.

Narzędzia ORACLE'a (ORACLE TOOLs) spełniają dwie role - służą do realizacji aplikacji metodami konwersacyjnymi, a następnie do jej interpretacyjnego wykonywania na rzecz określonego klienta. Każde z narzędzi ORACLE'a dokonuje przekształcenia polecenia użytkownika, wyrażonego w konwencji danego narzędzia, na instrukcję SQL, przekazywaną następnie do realizacji przez procesor SQL w serwerze.

Aplikacja łącznie z modułami wykonawczymi (Run-time modules) narzędzi ORACLE'a tworzy moduł klienta. Moduły te mogą być zwarte z Serwerem na tej samej platformie, lub też być rozproszone na innych stacjach. W tym drugim przypadku moduły SQL*Net poprzez jeden z wybranych protokołów komunikacyjnych łączą Klientów z ich Serwerem.



Rozproszone obiekty Klient - Serwer.

5. Organizacja systemu ORACLE

Organizacja Serwera

Organizację Serwera - struktury pamięci oraz powiązania procesów ORACLE przedstawia rysunek.

Organizacja pamięci

ORACLE w pamięci operacyjnej alokuje trzy główne struktury danych:

* Globalny Obszar Systemowy,
* Globalny Obszar Programu,
* Obszary Kontekstów.

Globalny Obszar Systemowy (SGA-System Global Area), rezerwowany w pamięci operacyjnej, zawiera informacje z jednej bazy danych rozmieszczone w dwóch typach buforów:

* pamięć kieszeniowa przechowująca ostatnio wykorzystywane informacje z bazy danych,
* bufory dziennika przechowujące modyfikowane dane

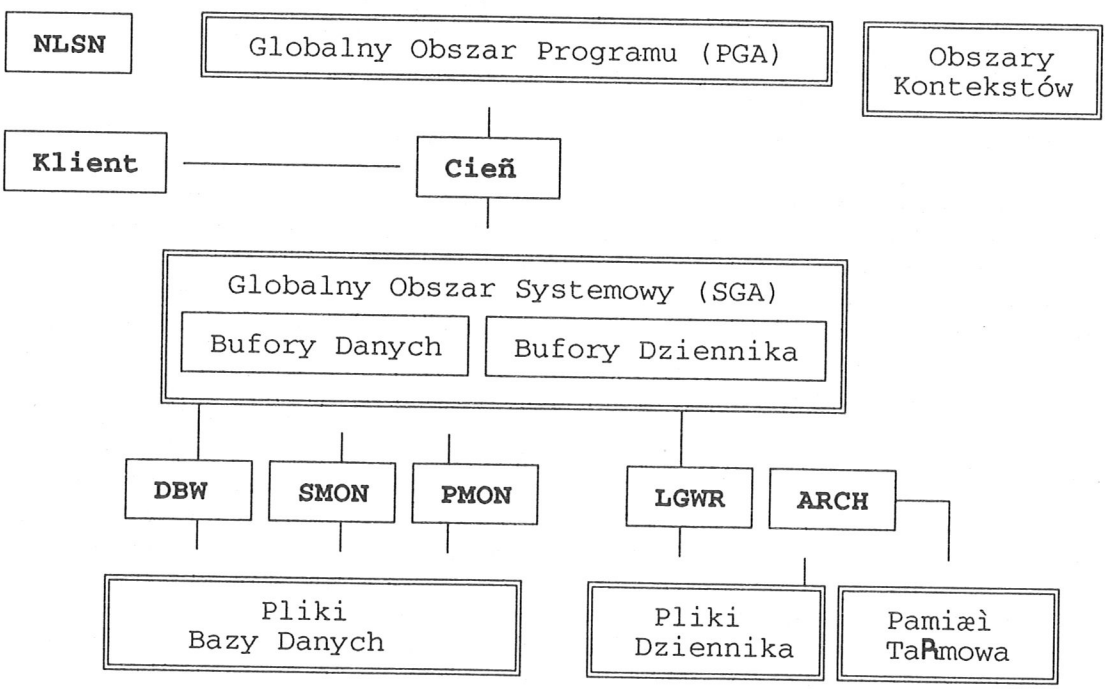
Dane do bufora dziennika są zapisywane po wykonaniu zdań SQL i są w nim przechowywane do momentu wydania operacji zakończenia transakcji (COMMIT lub ROLLBACK). Operacja COMMIT powoduje zmodyfikowanie danych w pamięci kieszeniowej. Operacja ROLLBACK usuwa z bufora dziennika zmodyfikowane dane.

Globalny Obszar Programu (PGA-Program Global Area) zawiera dane i informacje dla pojedynczego procesu użytkownika o komunikacji z programem i wykonanymi funkcjami.

Obszar Kontekstu (Cursor Area) jest buforem zawierającym aktualny stan pojedynczego zdania SQL. Obszar ten zawiera między innymi:

* tekst przetwarzanego zdania SQL
* postać wewnętrzną tego zdania SQL,
* ostatni wiersz wynikowy oraz niektóre wartości pośrednie,
* informacje wykorzystywane przy sortowaniu.

Liczba obszarów kontekstu jest określana parametrami systemu.



Organizacja Systemu Zarządzania Bazą Danych.

Procesy

SZRBD jest systemem współbieżnie działających procesów , należących do dwóch grup:

* Procesy Klienta (Client Process) wykonujące akcje określone aplikacją użytkownika. Każdy Klient współpracujący z Serwerem wymaga do swojej obsługi jednego procesu,
* Procesy Serwera (Serwer Process) realizujące funkcje SZRBD. W ORACLE'u istnieją dwa rodzaje takich procesów:
  - procesy cienia
  - procesy tła.

Procesy cienia (shadow process) obsługują procesy Klienta w bezpośrednich operacjach na bazie danych. Każdy proces Klienta ma swój własny proces cienia. Jest to realizacja koncepcji architektury Par Klient-Serwer.

Procesy tła (background process) realizują transmisje pomiędzy SGA a pamięciami masowymi. Pracują one asynchronicznie względem procesów realizujących zadania klienta i serwera. Istnieje pięć (plus jeden w konfiguracji sieciowej) standardowych procesów tła:

* Proces zapisu bazy danych (DBWR-Database Writer), czyli zmodyfikowanych bloków z pamięci kieszeniowej obszaru SGA do plików bazy danych,
* Proces zapisu dziennika (LGWR-Log Writer), czyli informacji o zmodyfikowanych elementach bazy danych z bufora do pliku dziennika,
* Proces monitora systemu (SMON-System Monitor) odzyskuje system po awarii i podczas gorącego startu,
* Proces monitora procesów (PMON-Process Monitor) odzyskuje procesy użytkowników po awarii,
* Proces archwizatora (ARCH-Archivier) kopiuje pliki dziennika na taśmę,
* Proces nasłuchu sieci (NLSN-Network Listener) wykrywa zgłoszenia z sieci od klientów do serwera.

6. Współbieżność i zwartość

Kontrola współbieżności

ORACLE jest systemem dopuszczającym współbieżną pracę wielu użytkowników, którzy mogą korzystać i aktualizować te same dane. Dla zapewnienia synchronizacji dostępu stosowane są mechanizmy blokad:

- wewnętrznych,
- słownika danych,
- tabel i wierszy.

Oprócz blokad ustawianych przez użytkownika, ORACLE V6 z OLTP w dużym stopniu sam realizuje kontrolowanie wielu dostępów do tych samych danych. Tym samym upraszcza się zarządzanie wykorzystaniem wspólnej bazy danych przez wielu użytkowników.

Blokowanie przez użytkownika

Blokady wewnętrzne są wykorzystywane do kontroli dostępu do wewnętrznych struktur ORACLE'a. Obejmują one:

- blokady plików zawierających bloki bazy danych przed operacją ich modyfikacji,
- struktur opisujących aktualnie dołączonych użytkowników.

Blokady słownika blokują dostęp do słownika danych, uniemożliwiając jednoczesne zmiany przez dwóch lub więcej użytkowników. Istnieją trzy typy takich blokad.

* Blokada operacji słownikowej (Dictionary Operation Lock) gwarantuje, że w tym samym czasie może być wykonana tylko jedna operacja na słowniku.
* Blokada definicji słownika (Dictionary Definition Lock) uniemożliwia dostęp do słownika w czasie wykonywania operacji definicji lub kontroli danych (blokada wyłączna) oraz uniemożliwia zmianę struktury tabel w trakcie rozbioru zdań języka SQL (blokada współdzielona).
* Blokada definicji tabel (Table Definition Level) uniemożliwia zmianę struktury tabeli w trakcie wykonywania operacji na jej danych (blokada współdzielna).

Blokady tabel-wierszy gwarantują zawartość danych, na których są wykonywane współbieżne transakcje. Zakres tych blokad jest określany wprost przez użytkownika lub automatycznie przez system. Istnieją trzy typy tych blokad.

* Blokada tabeli w trybie współdzielenia (SHARE Mode Table Locking),deklarowana wprost przez użytkownika. Wielu użytkowników może blokować tabelę oraz zablokowana tabela może być odczytywana przez wielu użytkowników, ale tylko ten, który ją zablokował, może zmienić jej zawartość.
* Blokada tabeli w trybie wyłącznym (EXCLUSIVE Mode Table Locking) jest zakładana automatycznie w czasie wykonywania zadań SQL ją modyfikujących. Blokada ta może być też wprowadzana na zlecenie użytkownika.
* Blokada tabeli w trybie współdzielonej aktualizacji (SHARE UPDATE Mode or Row

Level Locking) gwarantuje zachowanie wysokiego stopnia współbieżności i zwartości danych. Ten typ blokady gwarantuje , że dany wiersz tabeli nie będzie zmieniany przez wielu użytkowników jednocześnie.

Kontrola współbieżności na poziomie wiersza

ORACLE V6 z OLTP automatycznie zapewnia integralność danych przy wielodostępie poprzez:

- wielowersyjny spójny odczyt na poziomie wiersza,
- blokowanie na poziomie wiersza,
- generator sekwencji numerów.

Wielowersyjny model migawkowy spójnego odczytu dopuszcza:

- zapytania z odczytem bez blokady,
- zapytania bez blokady innych zapytań,
- zapytania bez blokady aktualizacji,
- aktualizację bez blokady zapytań.

Przed aktualizacją wiersza zapamiętywana jest jego dotychczasowa zawartość. Miejscem zapisu jest pamięć operacyjna lub segmenty nawrotów. Segmenty te są wykorzystywane do odtwarzania zawartości baz danych oraz do zapewniania spójności odczytu dla zapytania. Oznacza to, że aktualizacje dokonywane w trakcie pobierania wierszy w odpowiedzi na zapytanie nie mają nawet cząstkowego wpływu na jednorodność odczytu (wszystkie informacje pochodzą z chwili rozpoczęcia operacji).

Przykład VII.1:
Równoczesny dostęp.
W pewnym banku rozpoczęto operacje wyprowadzania raportu o sumarycznym stanie oszczędności na kontach klientów. W trakcie gromadzenia tej informacji klienci chcą dokonać wpłat i wypłat z ich kont. Umożliwiając jednocześnie operacje na kontach otrzymamy nieprawdziwe dane w raporcie. Pozostaje więc albo zablokować wszystkie konta na czas raportowania, albo skorzystać z właściwości ORACLE'a dostarczającego migawkowego obrazu całej bazy danych.

Blokowanie na poziomie wiersza

Najważniejszą cechą ORACLE'a V6 jest istnienie blokowania rekordów na poziomie pojedynczego wiersza (zamiast na poziomie całej strony - jak to jest w innych bazach danych). Zaletą takiego rozwiązania jest:

* możliwość jednoczesnej aktualizacji wielu wierszy z tej samej strony,
* możliwość dokonywania wielu blokad przez jednego użytkownika,
* brak eskalacji blokad prowadzącej do powstawania zakleszczenia,
* uzyskanie tej samej wydajności nawet przy intensywnym wykorzystywaniu danych i indeksów,
* jednoczesne wielowątkowe wstawianie informacji do wierszy z tej samej strony.

Generator sekwencji numerów

W wielu momentach funkcjonowania SZRBD konieczne jest uzyskanie kolejnego unikalnego numeru. W tym przypadku ORACLE dysponuje wysokowydajnym generatorem liczb udostępniającym kolejne liczby w systemie wielowątkowym bez strat czasu.

Administrowanie bezpośrednie

Podstawowym wymaganiem na funkcjonowanie systemów o działaniu bezpośrednim jest ich nieprzerywalna dyspozycyjność - 24 godziny i 7 dni w tygodniu. ORACLE dla spełnienia tego wymagania umożliwia wykonanie szeregu czynności z zakresu administrowania bazą danych bezpośrednio (online) w trakcie jej eksploatacji. Możliwe jest więc

* rekonfigurowanie,
* diagnozowanie,
* składowanie,
* odtwarzanie.

Bezpośrednia rekonfiguracja pozwala na dodawanie i modyfikowanie istniejących tabel poprzez zmiany liczby kolumn. Możliwe jest też tworzenie nowych klastrów, projekcji i indeksów dla zwiększania wydajności SZRBD. Również dla poprawienia wydajności można przenosić pliki bazy na inne dyski lub składać je na taśmie.

Bezpośrednia diagnostyka pozwala na bieżąco śledzić wydajność SZRBD w danym środowisku. Uzyskane wyniki mogą następnie posłużyć do optymalizacji wartości parametrów systemu przetwarzania transakcji. Monitor wydajności funkcjonowania ORACLE'a dostarcza między innymi następujących informacji:

* liczba operacji we-wy dla danego użytkownika i dla pliku,
* blokady dokonane przez użytkownika lub system bazy danych,
* informacje z segmentu nawrotu,
* statystyka systemu,
* informacje o dostępach do tabel,
* informacje o sesji użytkownika.

Zamiast podsumowania

Zaprezentowany opis systemu ORACLE nie udziela odpowiedzi na wszystkie pytania nasuwające się przy podejmowaniu bardzo istotnej decyzji o wyborze nowego narzędzia do projektowania i implementacji nowoczesnych złożonych aplikacji baz danych. Materiał ten został opracowany na podstawie informacji firmowych. Tam też należy poszukiwać odpowiedzi na wszystkie wątpliwości. Jednocześnie warto, przed podjęciem ostatecznych decyzji, zażądać od tej i innej firmy dokładniejszych informacji, umożliwiających porównanie efektywności działania poszczególnych SZRBD. Trzeba tutaj podkreślić, że uniwersalność tych SZRBD pociąga za sobą ograniczenia efektywności. Dlatego istotne jest zweryfikowanie ich efektywności w znanym (lub projektowanym) środowisku sprzętowym i dla danego typu aplikacji - wyniki ze standardowych "benchmarków" mogą być często bardzo mylące. Warto się też zastanowić nad ewentualną zmianą rodzaju sprzętu dla sprawniejszego działania naszej przyszłej aplikacji.

RELACYJNA BAZA DANYCH

PROGRESS - ogólna charakterystyka systemu

mgr inż.
Roman
Łowkis

Computer
Systems for Business
International Ltd
02-119 Warszawa,
ul. Pruszkowska 17
40-955 Katowice,
ul. Bytkowska 1b

1. Ogólna charakterystyka firmy

PROGRESS Software Corporation jest amerykańską firmą zajmującą się produkcją oprogramowania z zakresu relacyjnych baz danych. Głównymi produktami oferowanymi przez PSC jest system zarządzania bazami danych (RDBMS) oraz język czwartej generacji PROGRESS 4GL. W skład oferty uzupełniającej wchodzą pakiety narzędziowe usprawniające proces tworzenia aplikacji, a nawet umożliwiające bezpośrednią obsługę bazy danych.

Oprócz oprogramowania PSC oferuje swoim klientom szczegółową dokumentację, a także rozbudowany system szkoleń i doradztwa technicznego. Jedną z podstawowych dewiz firmy jest duża dbałość o klienta. Ta cecha w powiązaniu z nowoczesnym stylem zarządzania, a przede wszystkim znakomitym pod względem technicznym produktem jest kluczem do sukcesu firmy. W katalogu 500 najszybciej rozwijających się amerykańskich firm prywatnych PROGRESS Software Corporation znalazła się w 1989 r. na 28 miejscu.

2. Historia powstania i rozwoju PSC

PROGRESS Software Corporation powstała w grudniu 1981 roku. Jej głównym celem było stworzenie nowoczesnego narzędzia umożliwiającego obsługę relacyjnych baz danych.

Cel ten, którego końcowym efektem jest oferowany dziś PROGRESS Application Development System, realizowany był wieloetapowo. Równocześnie z rozwojem koncepcji merytorycznej systemu prowadzone były intensywne prace mające na celu maksymalne rozszerzenie bazy sprzętowej, na którą PROGRESS był oferowany. Historię powstania dzisiejszej wersji PROGRESS-a wyznaczają następujące daty:

lipiec '83 - początkowa wersja PROGRESS-a - etap testowania,
sierpień '84 - pierwsza komercyjna wersja PROGRESS-a na UNIX,
styczeń '85 - komercyjna wersja na MS-DOS,
czerwiec '85 - wersje komercyjne na systemy XENIX, ULTRIX,

marzec '86 - PROGRESS wersja 3,

kwiecień '87 - sieciowa wersja PROGRESS-a na UNIX
październik '87 - PROGRESS na VAX/VMS,
- PROGRESS wersja 4,

maj '88 - PROGRESS pod CTOS/BTOS,
wrzesień '88 - opublikowanie FAST TRACK-a,

maj '89 - PROGRESS wersja 5,

lipiec '90 - możliwość współpracy z bazami ORACLE oraz plikami RMS,
- PROGRESS wersja 6,

listopad '91 - PROGRESS for Windows, serwer NLM na Novella oraz mosty do narzędzi CASE (Excelerator i Knowledgeware)

luty '92 - bramy do AS/400, C-ISAM oraz CT-ISAM.

3. PSC i jej klienci

Być klientem PSC to duża przyjemność. Od momentu nabycia pierwszego produktu i podpisania polisy serwisowej klient znajduje się pod ciągłą opieką firmy. W ramach polisy otrzymuje bezpłatnie pojawiające się regularnie nowe wersje nabytego produktu. Również bezpłatnie otrzymuje biuletyn techniczny PSC - w którym szczegółowo omawiane są wykryte błędy systemu, wyznaczony (i dotrzymywany) termin korekty, a także metody uniknięcia skutków. W biuletynie technicznym można też znaleźć informacje na temat wszystkich nowości w ofercie, planów rozwoju systemu, a także zmian organizacyjnych zachodzących w firmie.

Klient PSC ma prawo do bezpłatnego korzystania z telefonicznego serwisu technicznego czynnego w USA 24 godziny na dobę (w Polsce, z powodów organizacyjnych, tylko w godzinach pracy CSBI). Istnieje również możliwość bezpośredniego (on-line) połączenia się z PROGRESS Customer Bulletin Board - bazą danych zawierającą katalog technicznych problemów (wraz z rozwiązaniami), jakie kiedykolwiek zostały zgłoszone w dziale technicznym firmy.

Często wraz z rozwojem aplikacji, rozrostem bazy danych lub liczby użytkowników zachodzi konieczność wymiany sprzętu. Z tym również nie ma problemu. Klient nie napotyka praktycznie żadnych ograniczeń w wyborze sprzętu. Koszty takiej operacji równają się jedynie różnicy cen pomiędzy starą a nową wersją produktu.

Poza realizacją zobowiązań wynikających z polisy serwisowej, PSC prowadzi szeroką działalność w zakresie kontaktów z klientami oraz pomiędzy klientami. Takim celom służy sieć klubów użytkowników PROGRESS-a (PROGRESS Users Groups) obejmująca USA, Australię oraz dziewięć krajów europejskich, w tym (od maja 1992) Polskę. Co roku użytkownicy mają okazję spotkać się na specjalnie organizowanych mityngach, gdzie oprócz bezpośredniego kontaktu z twórcami PROGRESS-a mają okazję zaprezentowania własnych aplikacji i projektów związanych z PROGRESS-em. Corocznie wydawany jest katalog aplikacji progressowych, zawierający ponad 1000 pozycji. Każdy użytkownik PROGRESS-a może zamieścić w nim darmowy opis swojego produktu, co - biorąc pod uwagę światowy zasięg wydawnictwa i jego wielotysięczny nakład - jest wspaniałą metodą promocji swoich aplikacji.

Dużo informacji na temat PSC, a także powstających firm satelickich, uzyskać można prenumerując „Profiles" - ilustrowany magazyn PSC wydawany w cyklu miesięcznym.

4. PSC w oczach innych

Od początku swej kariery rynkowej PROGRESS Software Corporation stanowi obiekt wnikliwego zainteresowania zarówno konkurencji, jak i niezależnych organizacji, które zajmują się oceną światowego rynku oprogramowania. Do takich organizacji należy DATAPRO, która w swych raportach z lat 1988, 1989, 1990, 1991 umieściła PROGRESS na pierwszym miejscu w swojej klasie, pośród takich systemów jak SYBASE, ORACLE, INGRESS, INFORMIX, czy FOCUS. Należy zaznaczyć, że DATAPRO ocenia nie tylko produkt, ale również całokształt usług oferowanych przez firmę. Na ocenę ogólną mają więc wpływ takie czynniki jak jakość dokumentacji firmowej, zakres prowadzonych szkoleń, zakres udzielanej licencji serwisowej, a także poziom i dyspozycyjność serwisu technicznego. Zwycięstwo w tej klasyfikacji jest prawdziwym powodem do dumy, choć nie stanowi zaskoczenia dla osób, które zetknęły się z PSC jako pracownicy lub klienci.

5. PSC w Polsce

Jeszcze do niedawna nie można było mówić o żadnej obecności PROGRESS-a w Polsce, ponieważ na produkty tak zaawansowanej technologii, jaką reprezentuje ten system, obowiązywało embargo COCOM-u. Począwszy od ubiegłego roku, kiedy embargo zostało zniesione, nastąpiła gwałtowna ekspansja PROGRESS-a na rynki wschodnioeuropejskie. PROGRESS wkroczył już do Polski, Czechosłowacji i Węgier, a w najbliższej przyszłości należy oczekiwać jego wejścia na pozostałe rynki wschodniej Europy, nie wyłączając rynku radzieckiego.

W Polsce interesy PROGRESS Software Corporation reprezentuje CSBI (Computer Systems

for Business International) - polsko-brytyjskie przedsiębiorstwo joint venture będące równocześnie dystrybutorem sprzętu, a także gotowych aplikacji napisanych w PROGRESS-ie. Oprócz funkcji handlowych firma CSBI zapewnia użytkownikom PROGRESS-a doradztwo techniczne, organizuje seminaria i pokazy z udziałem przedstawicieli PSC. Z drugiej strony dzięki naciskom (a także dużemu nakładowi pracy) ze strony CSBI powstała już częściowo spolonizowana wersja PROGRESS-a, a w najbliższym czasie ukaże się jego pełna, polska wersja narodowa.

Polska lista referencyjna zawiera już kilkadziesiąt pozycji, obejmujących banki, wyższe uczelnie, zakłady przemysłowe, firmy informatyczne a także urzędy państwowe.

• Produkty oferowane przez PSC

1. PROGRESS 4GL & RDBMS

System zarządzania relacyjną bazą danych i język czwartej generacji PROGRESS są flagowymi produktami PROGRESS Software Corporation. Stanowią one komplet narzędzi potrzebnych do utworzenia i uruchomienia złożonej, wieloużytkownikowej aplikacji pracującej praktycznie na dowolnej platformie sprzętowej.

Produkty te dostarczane są w zintegrowanym pakiecie obejmującym również interakcyjny słownik danych, edytor z analizatorem składni, system bieżących podpowiedzi oraz bibliotekę gotowych procedur. Na szczególną uwagę zasługuje słownik danych, który oprócz swych podstawowych funkcji - definicji struktury bazy i indeksów - umożliwia interakcyjne operacje na danych, takie jak: DUMP/LOAD, import i eksport plików w zadeklarowanym formacie, import definicji struktur oraz danych bezpośrednio z plików DBASE, przyłączenie lub odłączenie dowolnej bazy, a także zamrożenie części danych. Słownik umożliwia też utworzenie struktury użytkowników i administratorów bazy, a także generację różnorodnych raportów.

2. PROGRESS FAST TRACK

Jest rozbudowanym generatorem aplikacji stanowiącym pierwszy krok w technologii CASE w odniesieniu do PROGRESS-a. Jego cztery główne składowe, to edytor menu, edytor formatek ekranowych, projektant raportów i generator procedur QBF (Query By Form).

Menu Edytor (edytor menu) umożliwia błyskawiczne projektowanie dowolnych struktur menu, a także przypisywanie poszczególnym punktom różnych funkcji, od standardowych (return, quit, shell), aż do uprzednio stworzonych procedur. W ten sposób możliwe jest używanie funkcji FAST TRACK-a jednocześnie z „klasycznie" utworzonymi fragmentami aplikacji.

Screen Painter (edytor formatek) umożliwia komponowanie ekranu poprzez rozmieszczanie na nim pól bazy, zmiennych wraz z etykietkami i komentarzami. Dzięki filozofii WYSIWYG (What-You-See-Is-What-You-Get)[1] narzędzie to jest wygodne nawet dla początkujących użytkowników. Zaprojektowana formatka może zostać wykorzystana do wygenerowania procedury QBF lub zapisana na dysku w postaci pliku ASCII do wykorzystania w dowolnej procedurze jako plik włączony.

QBF - „zapytanie przez formatkę" jest pojęciem wprowadzonym przez twórców PROGRESS-a. Procedura QBF (zawsze związana z konkretną formatką) umożliwia przegląd bazy (formatka po formatce), wyszukiwanie żądanej informacji poprzez konstruowanie dowolnie złożonych warunków logicznych (system dba o ich poprawność), wprowadzanie, kasowanie i modyfikację informacji, generację raportu, a także przejście do innej procedury QBF (powiązanej z poprzednią, np. poprzez wartość pola).

Report Writer (projektant raportów) umożliwia, w podobny sposób jak edytor formatek, zaprojektowanie szaty graficznej i zawartości raportów, które będą generowane w systemie. Całość systemu przechowuje zadeklarowane struktury (formatki, menu i raporty) w specjalnie do tego celu rozszerzonej strukturze bazy danych. Dzięki temu w każdej chwili możliwe jest powrócenie do pracy nad projektem, uruchomienie dowolnego jego fragmentu, a także wygenerowanie tekstu odpowiedniej procedury w 4GL. W rezultacie uzyskujemy bardzo silne narzędzie do tworzenia kompletnych aplikacji, szczególnie zalecane do tworzenia prototypów funkcjonalnych.

3. PROGRESS RESULTS

PROGRESS RESULTS jest jednym z najnowszych produktów PSC. Jego główną ideą jest umożliwienie obsługi każdej bazy danych bez konieczności pisania kosztownej, dedykowanej aplikacji. Absolutna uniwersalizacja funkcji takiego pakietu jest oczywiście niemożliwa. PROGRESS RESULTS stanowi tu rozsądny kompromis.

Składa się z pięciu podstawowych modułów:

Query - umożliwia przegląd i modyfikację informacji na poziomie rekordu, wykorzystuje technikę zapytań QBE (Query By Example);

Reports - pozwala ujmować zawarte w bazie informacje w dowolne zestawienia;

Labels - automatyczny wydruk etykiet (adresy, identyfikatory, etykiety produktu);
Data Export - segment transferu informacji do innych pakietów software'owych (w formatach ASCII, DIF, SYLK, WordStar, MS-Word, WordPerfect, LOTUS oraz OfisWriter);
Administration - pozwala dostosować RESULTS do środowiska i potrzeb konkretnej aplikacji.

Moduły te obsługują większość typowych potrzeb użytkowników bazodanowych. Wszystkie nietypowe żądania mogą być zaspokojone poprzez połączenie RESULTS z dedykowaną aplikacją użytkownika. Struktura pakietu umożliwia zarówno włączenie niezależnych procedur do menu RESULTS, jak i wywołanie RESULTS z menu aplikacji.

4. Pozostałe pakiety

PROGRESS RUN-TIME jest minimalnym środowiskiem niezbędnym do korzystania z gotowej aplikacji progressowej. Umożliwia on uruchomienie aplikacji w postaci wynikowej, bez dostarczania użytkownikowi kodu źródłowego. Nie daje natomiast możliwości pisania własnych aplikacji ani też modyfikacji struktury bazy. Jego cena jest kilkakrotnie niższa od ceny PROGRESS 4GL & RDBMS.

PROGRESS DEVELOPER'S TOOLKIT jest pakietem narzędziowym, umożliwiającym programiście przeprowadzenie adjustacji gotowej aplikacji u użytkownika. Możliwości TOOLKIT-a obejmują:
- rekompilację aplikacji w nowym środowisku (bez modyfikacji kodu!),
- zmodyfikowanie metaschematu bazy,
- nałożenie ograniczeń na dostęp do definicji struktur bazy. Pakiet ten jest potrzebny jedynie programistom dostarczającym oprogramowanie w PROGRESS-ie. Użytkownik aplikacji lub programista piszący dla własnych potrzeb nie muszą zaopatrywać się w to narzędzie.

PROGRESS TEST-DRIVE jest w pełni funkcjonalną wersją PADS (patrz p. 5) zawierającą pewne wbudowane ograniczenia ilościowe (liczba dostępów do tej samej bazy). Jest ona rozprowadzana po symbolicznych cenach w celach reklamowo-szkoleniowych. Zakupienie TEST-DRIVE-a jest najlepszą metodą na poznanie PROGRESS-a i znalezienie odpowiedzi na pytanie, czy produkt ten odpowiada naszym potrzebom. Wbudowane ograniczenia nie pozwalają jednak wykorzystać go jako docelowego narzędzia obsługi bazy danych.

5. Zestawy pakietów

Pakiety oferowane przez PSC uzupełniają się nawzajem, dlatego też oferowane są w zestawach, których cena łączna jest niższa od sumy cen poszczególnych składników. Podstawowym takim zestawem jest PROGRESS APPLICATION DEVELOPMENT SYSTEM (PADS). Składa się on z PROGRESS 4GL & RDBMS oraz FAST TRACK-a.

Drugi zestaw to PROGRESS QUERY/REPORT. Nie jest on jednak prostą sumą paru innych produktów. Oparty o PROGRESS RUN-TIME nie daje możliwości modyfikacji struktury bazy ani też pisania w 4GL. Udostępnia natomiast pełne możliwości PROGRESS RESULTS oraz modułu Report Writer pakietu FAST TRACK. Zestaw ten stanowi minimum niezbędne do utrzymania własnej bazy danych przy maksymalnej redukcji kosztów.

• System PROGRESS - charakterystyka techniczna

1. Wstęp

PROGRESS 4GL & RDBMS jest nowoczesnym, szybkim i bezpiecznym narzędziem służącym do zarządzania dużą ilością danych w rozproszonym i niejednorodnym środowisku systemowym. Podstawowa jego cecha to niemal całkowita niezależność od sprzętu, systemu operacyjnego czy konfiguracji sieciowej, w której pracuje. Jest to możliwe dzięki opracowaniu wersji systemu na kilkaset platform sprzętowych, obejmujących wszystkich czołowych producentów oraz różne opcje systemowe, a także dzięki implementacji najczęściej spotykanych protokołów sieciowych.

Język PROGRESS 4GL należący do grupy języków czwartej generacji jest bardzo wydajnym i elastycznym narzędziem pozwalającym skrócić czas pisania aplikacji około dziesięciokrotnie w stosunku do języków trzeciego poziomu. Zapewniając pełną kontrolę klawiatury oraz ekranu umożliwia on dostosowanie interfejsu użytkownika do indywidualnych potrzeb i przyzwyczajeń. Zastosowane mechanizmy wspomagania programisty zwiększają komfort pracy, a elastyczna struktura systemu ułatwia wprowadzanie zmian już po zamknięciu projektu.

System jako całość wyposażony jest w rozbudowany mechanizm ochrony danych. Obejmuje on ochronę przed niepowołanym dostępem, kontrolę spójności logicznej informacji oraz wielopoziomowe zabezpieczenia danych przed fizycznym uszkodzeniem. Wydajność systemu w środowisku wielodostępnym zapewniona jest przez zastosowanie najnowocześniejszych technik przetwarzania oraz maksymalne wykorzystanie możliwości sprzętowych (np. w przypadku wieloprocesorowych maszyn Sequenta).

2. Środowisko pracy PROGRESS-a

a) Platformy sprzętowe

Niezależność od sprzętu jest jedną z najważniejszych zalet PROGRESS-a. Oznacza ona pełną przenośność aplikacji pomiędzy różnymi platformami sprzętowymi, a także możliwość łączenia różnych systemów w spójne środowisko, w którym przetwarzanie rozproszone, a także dostęp do nielokalnych (również wielowolumenowych) baz danych kontrolowany jest przez PROGRESS.

Lista dostępnych dla PROGRESS-a platform sprzętowych jest prawdopodobnie najszersza spośród list systemów konkurencyjnych. W chwili obecnej obejmuje ona ponad 200 (a z uwzględnieniem różnych wersji maszyn - ponad 500) pozycji. Obecnie są na niej produkty takich firm jak: ALR, Altos, Apple, Apricot, Arix, AT&T, Bull, Compaq, Data General, DDE, DEC, Dell, DIAB, Dolphin, Edisa, Everex, Harris, Hewlett-Packard, IBM, ICL, Intel, ITL, MIPS, Motorola, NCR, Nixdorf, Nokia Data, Norsk Data, Olivetti, Opus Systems, Philips, Prime, Pyramid, Sanyo/Icon, Sequent, SID INFORMATICA, Siemens, Silicon Graphic, Solbourne, Sun, Tandem Integrity, Texas Instruments, TIS Limited, UNISYS, ZENITH.

Różnorodne architektury komputerów znajdujących się na tej liście oparte są o procesory M680X0, 80X86, RISC, VAX, 88000 oraz SPARC.

b) Systemy operacyjne

PROGRESS pracuje pod kontrolą takich systemów operacyjnych jak DOS, OS/2, UNIX i jego pochodne (XENIX, ULTRIX, AIX, A/UX, HP-UX itd.), VMS, BTOS, CTOS.

Do uruchomienia PROGRESS-a w środowisku UNIX wymagane jest min. 2 MB pamięci operacyjnej. Minimalne wymagania co do środowiska DOS-owego są następujące:

- system w wersji 3.1 lub wyższej,
- dysk twardy (min. 17 MB miejsca dla PADFS + miejsce na bazę i aplikacje),
- stacja dysków miękkich dużej gęstości (DS/HD),
- 640 KB pamięci operacyjnej (wersja pamięciooszczędna), 2 MB pamięci operacyjnej (wersja pełna).

Istnienie wersji pamięciooszczędnej na DOS (całkowicie równoważnej funkcjonalnie w stosunku do wersji pełnej) jest cechą wyróżniającą PROGRESS spośród innych systemów tej klasy. Jest to niesłychanie istotne z punktu widzenia użytkowników, którzy chcieliby zainstalować system na już posiadanym sprzęcie, bez konieczności jego modyfikacji. Oczywiście użytkowanie tej wersji jest niewygodne, pozwala jednak rozłożyć w czasie koszta związane z zakupem nowego systemu. Wersje pełna i pamięciooszczędna sprzedawane są jednocześnie, a więc w przypadku dokupienia pamięci cała operacja zmiany wersji sprowadza się do przestawienia jednej zmiennej systemowej.

c) Praca w sieciach

Mocną stroną PROGRESS-a są jego możliwości pracy w środowisku sieciowym. Możliwości te obejmują pracę w sieciach jednorodnych opartych na protokołach TCP/IP, NetBios, SPX/IPX oraz DECnet, a także pracę w sieciach heterogenicznych.

Praca w DOS-owych sieciach rozległych sprowadza się do komunikacji procesu klienta z odległym serwerem bazy danych za pośrednictwem modułu CCM[2]. Możliwe jest przy tym wykorzystanie wszystkich wymienionych protokołów równocześnie. W sieciach lokalnych PROGRESS umożliwia dostęp do bazy ze wszystkich węzłów sieci, a także obsługę żądań odległych poprzez CCM z wykorzystaniem protokołów NetBios oraz SPX/IPX. Proces serwera, a także procesy klientów, mogą zostać uruchomione na dowolnym komputerze w sieci, jednakże - ze względu na jednozadaniowość DOS-a - nie jest możliwe uruchomienie wielu procesów na jednej maszynie.

Najnowszym produktem PROGRESS-a (wersja 6.2H) jest moduł NLM[3]. Jego zastosowanie w sieciach Novell (wersje od 3.11) daje znaczne korzyści czasowe dzięki lepszemu wykorzystaniu możliwości sprzętowych.

PROGRESS pracuje z dwoma wersjami protokołu TCP/IP - firm Excelan oraz FTP Software. Interfejs Excelan obsługiwany jest przez wbudowane procedury PROGRESS-a. Aby wykorzystać wersję FTP, konieczne jest zainstalowanie właściwego programu obsługi.

W celu przyłączenia PC do bazy danych pracującej pod VMS należy zakupić DECnet DOS Digital Equipment Corporation. Oprogramowanie to tłumaczy wywołania NetBiosa na wywołania protokołu DECnet, co umożliwia serwerowi VMS obsługę żądań klienta DOS-owego.

Większość instalacji systemu UNIX posiada wbudowany protokół TCP/IP. Jest on wykorzystywany przez PROGRESS. Dla pozostałych wersji UNIX-a i XENIX-a PROGRESS zakłada wykorzystanie TCP/IP z interfejsem Excelan. Karty sieciowe i oprogramowanie należy zakupić wówczas oddzielnie w firmie Excelan. Najnowszym produktem PROGRESS-a jest moduł umożliwiający łączenie instalacji UNIX-owych z siecią Novell.

Implementacja protokołu sieciowego DECnet pozwala na tworzenie sieci pod VAX/VMS. Po-

przez DECnet DOS możliwe jest przyłączenie DOS-owych klientów do serwera bazy pracującego pod VAX/VMS. PROGRESS daje również możliwość rozdziału aplikacji pomiędzy klastery VAX-a.

Instalacje PROGRESS-a w środowisku CTOS/BTOS, ze względu na cechy wbudowane tych systemów, nie wymagają żadnych sieciowych interfejsów (wersja sieciowa jest wersją standardową). Komunikacja zewnętrzna odbywa się za pośrednictwem protokołu TCP/IP. Wewnątrz sieci BTOS/CTOS możliwe jest również wykorzystanie BNET, CTNET oraz APNET.

Serwer PROGRESS-a pracujący pod kontrolą OS/2 obsługuje zgłoszenia klientów sieci za pośrednictwem protokołów NetBios lub SPX. Ponieważ klienci DOS-owi mogą pracować z wykorzystaniem tych samych protokołów, mają oni wygodny dostęp do informacji zgromadzonej w bazach pozostających pod kontrolą OS/2.

d) Współpraca z systemami okien

Ekspansja systemów okienkowych powoduje coraz powszechniejsze dostosowywanie oprogramowania do ich wymogów. PROGRESS posiada wbudowane możliwości współpracy z GUI[4] bazującym na standardzie X-Windows. Należą do nich: MOTIF, Open Look, Open Desktop, DEC Windows oraz Presentation Manager. Oprócz tego PROGRESS potrafi współpracować z NewWawe Hewlett-Packarda, a ostatnio ujrzała światło dzienne wersja przeznaczona dla Microsoft Windows 3.0.

Istotną cechą systemu PROGRESS jest to, że istniejące aplikacje nie pisane z myślą o systemach okienkowych mogą bez żadnych zmian zostać na nich uruchomione. Wbudowane cechy PROGRESS-a zapewnią bezbłędną obsługę wszystkich funkcji takich systemów. Należy jedynie zauważyć, że zastosowanie systemu MS-Windows 3.0 podnosi wymagania sprzętowe instalacji. Komputer z taką konfiguracją oprogramowania wymaga już wersji DOS-a 3.3 lub wyższej, minimum procesora 80286, chociaż zalecany jest 80386 i wyższe oraz min. 2 MB pamięci operacyjnej (zalecane 4 MB).

e) Import i eksport danych

System PROGRESS wyposażony jest w wygodny mechanizm wymiany danych z otoczeniem. Z poziomu aplikacji możliwy jest import oraz eksport danych z/do plików ASCII o określonym formacie. Jeżeli plik nie spełnia wymogów formatu PROGRESS-a, można go przetworzyć za pomocą programu QUOTER. W trakcie pracy interaktywnej (jedna z opcji Data Dictionary) możliwy jest eksport danych w formacie ASCII, DIF (Data Interchange Format), SYLK (SYmbolic LinK)[5], WordPerfect, WordStar, Word oraz BTOS OfisWriter. Możliwy jest też import danych w formacie DIF, SYLK, ASCII[6]. W PROGRESS Data Dictionary istnieje również opcja pozwalająca na import definicji struktur oraz danych bezpośrednio z bazy DBASE (II, III, III+, IV).

Inną metodą komunikacji PROGRESS-a z otoczeniem są potoki[7]. Są one jednym z mechanizmów komunikacji międzyprocesorowej dostępnym w systemie operacyjnym UNIX. Stwarzają duże możliwości przekazywania informacji pomiędzy aplikacją PROGRESS-owską a programami napisanymi w innych językach. Poprzez potok UNIX-owy możliwe jest na przykład wysłanie zapytania SQL do aplikacji PROGRESS-owej i uzyskanie w odpowiedzi żądanego zestawu danych z bazy PROGRESS-a.

Wykorzystanie potoków stanowi alternatywę w stosunku do niektórych zastosowań interfejsu HLC umożliwiającego wywołanie PROGRESS-a z poziomu języka trzeciej generacji.

3. Baza danych

a) Organizacja bazy PROGRESS

Omawiając strukturę bazy danych w systemie PROGRESS, należy zacząć od definicji podstawowych pojęć. Nie zawsze bowiem nazwa danego elementu będzie miała znaczenie zgodne z naszymi przyzwyczajeniami. Mankament ten dotyczy niemal wszystkich istniejących systemów, stąd najlepsze wydaje się pozostanie w obrębie terminologii ustalonej przez twórców systemu, którego używamy.

Najmniejszą jednostką informacji jest w PROGRESS-ie pole rekordu. Jest ono ekwiwalentem zmiennej języka i może przyjmować te same co zmienna typy (Character, Integer, Logical, Date, Raw oraz RecID). Z założenia tego wynika nietypowa dla innych systemów możliwość przyjmowania przez pole wartości tablicowej. Pole jest wówczas wektorem zmiennych (w takim sensie jak np. tablica w PASCAL-u).

Zestaw pól tworzy rekord. Inne nazwy rekordu to wiersz relacji lub krotka. Bardzo ważną cechą PROGRESS-a jest zmienna długość rekordu (a więc i pola). Deklarowany format pola jest tylko formatem stosowanym do operacji wejścia/wyjścia. Informacja pamiętana jest w bazie w całości, nawet jeżeli długość zapisu przekracza aktualnie zadeklarowaną.

Zbiór rekordów nazywany jest w PROGRESS plikiem (ang. file), co może prowadzić do nieporozumień, ponieważ nie ma on nic wspólnego z plikiem systemowym. Plik PROGRESS-a należy rozumieć jako tablicę (termin SQL) lub relację.

Baza danych jest z kolei zbiorem plików. Całość bazy mieści się w jednym pliku systemowym. W pliku tym - oprócz samych danych - pamiętane są definicje struktury bazy, indeksów,

etykiety i opisy pól rekordów, warunki dostępu, a nawet klauzule walidacji sprawdzane w trakcie modyfikacji pól bazy. Takie rozwiązanie ogranicza liczbę odwołań do systemu operacyjnego, a więc jest korzystne ze względu na efektywność i niezawodność systemu.

b) Słownik danych

Bardzo istotnym elementem systemu PROGRESS jest słownik danych (Data Dictionary). Słownik ten umożliwia definicję i modyfikację struktury bazy, przegląd zdefiniowanych struktur SQL, przyłączenie i odłączenie bazy, wybór bazy aktywnej, generację plików różnicowych, generację plików błędów, import i eksport danych i struktur, wprowadzenie struktury użytkowników i ograniczeń dostępu, a także generację różnorodnych raportów. Wszystkie funkcje słownika dostępne są w trybie interaktywnym dzięki aktywnemu Pull-Down Menu. Są one pogrupowane w następującym podmenu:

MODIFY-SCHEMA

Znajdują się tu funkcje umożliwiające definiowanie i modyfikowanie schematu bazy danych. Definiuje się strukturę plików w bazie, strukturę rekordu w danym pliku oraz indeksy. Możliwa jest zmiana struktury istniejącej bazy bez naruszenia zawartych w niej danych, jednakże nie można zmienić typu zadeklarowanego pola. Tego rodzaju operację należy przeprowadzić etapowo, definiując nowe pole, przepisując dane ze starego pola za pomocą odpowiedniej procedury konwersji, po czym usuwając stare pole. Ze względu na zmienną długość rekordu w bazie PROGRESS-a wszelkie zmiany formatu pola są dozwolone i nie mają wpływu na informację przechowywaną w bazie.

Indeks może zostać utworzony poprzez konkatenację pól. Może on być zadeklarowany jako unikalny. System nie dopuści wówczas do powstania dwóch rekordów o tej samej wartości klucza, może też być zadeklarowany jako skrócony, co umożliwi wyszukiwanie rekordów o kluczu zaczynającym się od podanego ciągu znaków. Każde pole wchodzące w skład klucza może mieć przypisaną prostą lub odwróconą regułę kolejności.

SQL

Podmenu SQL zawiera funkcje umożliwiające przegląd zdefiniowanych struktur SQL. Można wyświetlić listę istniejących projekcji[8], obejrzeć ich struktury, a także wygenerować fragment procedury napisanej w SQL DDL[9] odpowiadającej danej projekcji.

Database

Funkcje zawarte w tym podmenu pozwalają na przyłączenie i odłączenie bazy danych, wybór bazy aktywnej, utworzenie nowej, pustej bazy danych oraz na wykonanie niektórych szczególnych operacji specyficznych dla poszczególnych typów baz (PROGRESS, AS/400, ORACLE, Rdb, RMS). Operacje na bazach nie progressowych stają się jednak dostępne dopiero po zakupieniu dodatkowych modułów (bram).

Dla baz progressowych możliwe jest utworzenie pliku różnicowego pomiędzy bazami (plik ten może posłużyć później do uzupełnienia brakujących danych w bazie, lub też do innych celów), konwersja pliku różnicowego z wersji 5 na 6, utworzenie pliku z danymi w formacie wersji 5 oraz utworzenie listy parametrów domyślnych baz automatycznie przyłączanych przez system w trakcie wykonywania aplikacji.

Admin

Podmenu Admin {istration} obsługuje wszystkie funkcje związane z importem i eksportem danych oraz systemem bezpieczeństwa. Tutaj również umieszczone zostały funkcje zrzutu i odtworzenia stanu bazy (DUMP/LOAD Data Def. & File Contents). W trakcie odtwarzania stanu bazy możliwa jest dezaktywacja indeksów (względy czasowe) oraz utworzenie specjalnego pliku rekordów błędnych, który, po poprawieniu za pomocą dowolnego edytora, można później wprowadzić jako plik różnicowy.

Utilities

Podmenu narzędziowe zawiera edytor zbiorów parametrów (*.pf), interfejs do programu Quoter - formatera plików, generator plików włączonych obsługujący najczęściej spotykane zastosowania instrukcji ASSIGN, FORM oraz DEF WORKFILE języka 4GL. Tu umieszczona jest funkcja Info dostarczająca skondensowanej informacji o stanie systemu, jak również funkcja Freeze/Unfreeze pozwalająca na tymczasowe zamrożenie dowolnego pliku bazy danych.

Reports

Tu zgromadzone są funkcje umożliwiające przegląd i wydruk (możliwe jest zapisanie w pliku) informacji ogólnych na temat użytkowanej bazy. Można więc uzyskać szczegółowy raport na temat zdefiniowanych plików i struktur rekordów tych plików, listę samych plików, listę pól rekordu w danym pliku, strukturę indeksów, listę użytkowników, listę zdefiniowanych projekcji oraz listę implikowanych (poprzez nazwę) relacji.

c) Bazy innych systemów

PROGRESS 4GL jest językiem czwartej generacji niezależnym od bazy danych. Może on zostać wykorzystany do obsługi baz danych innych systemów. W chwili obecnej istnieją bramy umożliwiające bezpośrednią obsługę z poziomu 4GL struktur ORACLE, RDB, RMS, AS/400 oraz C-ISAM. Struktury te (w postaci baz lub plików nierelacyjnych) rozmieszczone na rozproszonym i niejednorodnym sprzęcie, znajdujące się pod kontrolą różnych systemów operacyjnych i połączone za pomocą różnych protokołów sieciowych mogą być wykorzystane jednocześnie, przez jedną aplikację, a dane z nich pochodzące mogą być modyfikowane w jednej formatce ekranowej. Całość działań przebiega pod kontrolą 4GL, a więc nie zachodzi potrzeba odwoływania się do języków niższego poziomu. Prowadzi to do uzyskania stopnia spójności aplikacji niedostępnego w innych systemach.

d) Ograniczenia liczbowe

Dewizą PROGRESS Software Corporation jest „Usual limits don't apply"[11]. Najlepszą weryfikacją tej tezy jest przegląd ograniczeń liczbowych systemu PROGRESS w wersji 6:

Rozmiar bazy:	200 GB,
Liczba aktywnych baz:	240 baz jednocześnie,
Długość rekordu:	32000 B,
Długość pola:	32000 B,
Liczba pól w rekordzie:	Jedyne ograniczenie wynika z długości rekordu,
Liczba plików w bazie:	1023,
Liczba indeksów / plik:	1023,
Liczba indeksów w bazie:	1023,
Całkowita liczba użytkowników:	Zależnie od sprzętu, maksymalnie 2048,

Typy danych:

charakter	maks. 3200 B, pełny zbiór ASCII,
decimal	maks. 500 cyfr, w tym do 10 miejsc po przecinku,
integer	od -2 147 483 648 do +2 147 483 647,
logical	true/false, yes/no lub w/g deklaracji,
date	1.1.32768 p.n.e. do 12.31.32767 n.e.

null istnieje możliwość posługiwania się wartością pustą.

Zamieszczone zestawienie jest dużym skrótem szczegółowego wykazu ograniczeń dostępnego w Progress Help.

4. Język

a) PROGRESS 4GL

PROGRESS 4GL jest językiem czwartej generacji pomyślanym jako uniwersalna nadbudowa nad mechanizmem zarządzania bazą danych (RDBMS). Elastyczność ta została osiągnięta poprzez przemyślany dobór poleceń języka oraz zastosowanie mechanizmu bram pośredniczących w transferze danych z i do bazy. Użytkownik korzystający z PROGRESS 4GL może więc obsługiwać wiele różnych baz jednocześnie w jednej spójnej aplikacji.

Określenie język czwartej generacji w odniesieniu do PROGRESS 4GL oznacza, że jest to język raczej deklaratywny niż sekwencyjny. Innymi słowy użytkownik deklaruje w nim chęć osiągnięcia określonego celu zamiast kodowania sekwencji operacji do niego prowadzących. Metody osiągania tego celu są już domeną systemu. Na obecnym etapie rozwoju języków komputerowych cechy deklaratywne nie eliminują sekwencyjnego układu programu. Jest on zresztą zgodny z naturalnym sposobem analizy problemu przez człowieka.

Składnia 4GL jest wyjątkowo przejrzysta i konsekwentna. Mimo iż w wielu przypadkach liczba parametrów pojedynczego polecenia może być bardzo duża, w praktyce większość z nich jest przyjmowana bądź wyliczana automatycznie na podstawie bieżących parametrów systemu. Posługiwanie się wartościami domyślnymi jest kluczem do siły tego języka. Umożliwia ono programiście skoncentrowanie się na schemacie funkcjonalnym projektowanego systemu bez tracenia czasu na szczegóły. Pielęgnacja strony użytkowej aplikacji może nastąpić później już w trakcie eksploatacji, chociaż często nie jest w ogóle potrzebna.

Język zapytań będący podzbiorem 4GL jest sprzężony z mechanizmem obsługi ekranu, co istotnie ułatwia projektowanie przeglądów. Forma zapytań zbliżona jest do naturalnego języka angielskiego. Użycie indeksów podczas wyszukiwania danych następuje automatycznie, chociaż istnieje możliwość wyboru konkretnego indeksu, którym w danym zapytaniu chcemy się posłużyć. Jeżeli indeks dla danego klucza nie został zdefiniowany, wyszukanie danej jest w dalszym ciągu możliwe, choć czasochłonne.

Ponieważ aplikacje muszą być dopasowane do różnych potrzeb i przyzwyczajeń użytkowników, PROGRESS 4GL posiada wygodny mechanizm kontroli klawiatury. Umożliwia on narzucenie własnego schematu obsługi w każdym z interakcyjnych elementów języka.

Najważniejszą cechą PROGRESS-a jest jego kompletność. Oznacza to możliwość stworzenia całości aplikacji w 4GL bez konieczności odwoływania się do języków trzeciego poziomu.

b) PROGRESS SQL

PROGRESS SQL jest językiem obsługi relacyjnych baz danych opartym na standardzie SQL86 ANSI. Spełnia on wszystkie wymagania

pierwszego poziomu oraz większość wymagań poziomu drugiego standardu. Dodatkowo język ten udostępnia wiele funkcji charakterystycznych dla PROGRESS-a.

W przeciwieństwie do innych istniejących implementacji SQL język ten zanurzony jest w 4GL, a więc nie wymaga korzystania z języków trzeciej generacji. Instrukcje SQL mogą być przeplatane z kodem 4GL.

Rozszerzenia w stosunku do standardu SQL86 stanowią polecenia ALTER TABLE, DROP TABLE, DROP VIEW, CREATE INDEX, DROP INDEX, REVOKE, charakterystyczne dla PROGRESS-a typy danych DATA i LOGICAL, a także frazy definicji formatu danych i formatki ekranowej. Definicje te są wspólne dla PROGRESS SQL oraz 4GL

c) Wykorzystanie języków trzeciej generacji (C, COBOL, PASCAL)

Mimo iż w większości zastosowań nie zachodzi potrzeba użycia języków trzeciej generacji, to jednak możliwość taka istnieje i jest wykorzystywana wszędzie tam, gdzie PROGRESS pracuje w nietypowym otoczeniu.

Istnieją dwa interfejsy służące do komunikacji PROGRESS-a z 3GL[11]. Pierwszy z nich - HLC[12] - pozwala na wywołanie procedur napisanych w C z poziomu 4GGL. Daje to możliwość zdefiniowania nietypowych funkcji numerycznych, bezpośredniej obsługi urządzeń zewnętrznych jak np. sprzęt video, czy przetworniki przemysłowe, a także umożliwia bezpośrednie operacje na danych innych programów.

Drugi interfejs - HLI[13] - umożliwia umieszczenie zapytań PROGRESS SQL w kodzie języka niskiego poziomu (C, COBOL-u, PASCALA)[14]. Daje to możliwość wykorzystania mechanizmu zarządzania bazą danych PROGRESS-a z poziomu 3GL. Oznacza to nie tylko dostęp do danych, ale również pełne ich bezpieczeństwo zapewnione przez RDBMS. Ważnym elementem jest możliwość dynamicznego formułowania zapytań SQL w trakcie pracy programu 3GL.

5. Mechanizmy pracy współbieżnej

W środowisku wielodostępnym, podczas równoczesnej pracy wielu użytkowników, występują dwa problemy. Pierwszy z nich to powstawanie „wąskich gardeł", co prowadzi do bardzo szybkiego spadku wydajności systemu wraz ze wzrostem liczby aktywnych klientów. W przypadku PROGRESS-a antidotum stanowi architektura CLIENT/SERVER oraz zastosowanie przetwarzania wielotorowego.

Znacznie bardziej złożony jest problem współdzielenia zasobów. PROGRESS udostępnia całkowicie automatyczny mechanizm rezerwacji zasobów na poziomie pojedynczych rekordów lub całych plików. W zależności od rodzaju wykonywanej operacji rezerwacja dokonywana jest w trybie wyłącznym (EXCLUSIVE LOCKING) lub w trybie dzielonego dostępu (SHARED LOCKING). Sterowanie rezerwacją zasobów możliwe jest przez odpowiednie konstruowanie bloków w procedurach.

6. Mechanizmy ochrony danych

a) Ochrona przed niepowołanym dostępem

W niektórych zastosowaniach, jak np. bankowość, czy wojskowość, szczególnie ważna staje się ochrona danych przed niepowołanym dostępem. PROGRESS wyposażony jest we wbudowany system ochrony danych umożliwiający utworzenie struktury użytkowników weryfikowanych za pomocą indywidualnych haseł, nadanie uprawnień administratora systemu wyodrębnionej podgrupie użytkowników, a także zdefiniowanie zakresu dostępu do danych dla każdego użytkownika. Jeżeli struktura taka została zdefiniowana, system automatycznie weryfikuje tożsamość użytkownika w momencie startu, a następnie sprawdza uprawnienia przy każdej próbie dostępu do danych.

Za pomocą omówionego mechanizmu administrator systemu może zezwolić, bądź zabronić jednego z czterech (Can-Read, Can-Write, Can-Create, Can-Delete) typów dostępu na poziomie plików, a także jednego z dwóch typów dostępu (Can-Read, Can-Write) na poziomie pól rekordu.

Jeżeli twórca aplikacji zamierza wprowadzić bardziej złożone mechanizmy kontroli dostępu (np. blokada pewnych operacji, a nie tylko dostępu do danych), może dokonać tego, wykorzystując istniejące w 4GL funkcje CAN-DO oraz USERID.

b) Zapewnienie spójności logicznej podczas wprowadzania

Bardzo rzadko zdarza się, że wprowadzana dana jest całkowicie niezależna od pozostałej informacji zgromadzonej w systemie. Najczęściej dane uzupełniają się nawzajem i są sensowne tylko w pewnych konfiguracjach (nie ma sensu np. rachunek bez właściciela czy nr telefonu bez przypisanego abonenta). Również wartość danych zazwyczaj nie może być dowolna. Ponieważ zawsze najbardziej zawodnym elementem systemu jest człowiek, sensowność danych należy sprawdzać przede wszystkim podczas ich wprowadzania lub modyfikacji.

Do tego celu służą w PROGRESS-ie klauzule walidacji. Są to często złożone warunki logiczne sprawdzane automatycznie podczas edycji pola lub zmiennej. W odniesieniu do pól bazy są one

definiowane łącznie ze strukturą bazy w Data Dictionary. Warunek walidacji zmiennej możemy zadeklarować w momencie deklaracji samej zmiennej. Jeżeli warunek walidacji pola lub zmiennej nie będzie spełniony podczas edycji, użytkownik nie będzie mógł opuścić edytowanego pola, chyba że poprawi błąd lub zrezygnuje z edycji, co jest jednoznaczne z zachowaniem starej wartości.

Do niespójności danych może doprowadzić również usunięcie części z nich, jeżeli do usuniętych rekordów odnoszą się dane zgromadzone w innej części bazy. Przed takim działaniem można zabezpieczyć się w PROGRESS-ie, definiując w opisie pliku warunki konieczne do spełnienia w momencie usuwania rekordów z pliku.

Zarówno w pierwszym, jak i drugim przypadku istnieje możliwość sformułowania komunikatu, który będzie automatycznie wyświetlany w przypadku stwierdzenia nieprawidłowości.

c) Ochrona przed uszkodzeniem lub utratą danych

W dużych systemach istnieje realne niebezpieczeństwo utraty części, a nawet całości danych, na skutek awarii sprzętu lub systemu operacyjnego. Pomysłowość materii nieożywionej wydaje się być tutaj nieskończona, tak że trudno spodziewać się, aby problem ten został kiedykolwiek definitywnie rozwiązany. Minimalizacja prawdopodobieństwa utraty danych, bądź zaistnienia w nich przekłamań (często gorszych w skutkach niż całkowita utrata danych), zależy od ilości i różnorodności zastosowanych zabezpieczeń. Mechanizmy zaimplementowane w systemie PROGRESS obejmują:

Two-phase commit[15]

Przy odwołaniach do wielu baz w środowisku rozproszonym spójność danych gwarantowana jest przez dwufazowy protokół potwierdzeń. Obejmuje on fazę potwierdzenia zapisu przez poszczególne serwery. Jeżeli z jakichkolwiek powodów część danych nie została zapisana, całość operacji jest anulowana.

Before-imaging[16], transakcje

Jeżeli przechowywana w bazie informacja ma być spójna, przy każdej operacji modyfikującej dane powinna obowiązywać zasada „wszystko albo nic". Zabezpieczenie przed częściową, niekompletną modyfikacją stanu bazy osiągane jest w PROGRESS-ie poprzez zastosowanie mechanizmu transakcji. W momencie rozpoczęcia transakcji bieżący stan modyfikowanej części bazy zapamiętywany jest w specjalnym pliku *.bi (tzw. before-image file). Jeżeli z jakichkolwiek powodów (np. zaniku zasilania, błędu systemu lub zaniechania operacji przez użytkownika) transakcja nie zostanie pomyślnie zakończona, system gwarantuje odtworzenie stanu bazy sprzed rozpoczęcia transakcji.

W systemie PROGRESS możliwe jest zagnieżdżanie transakcji. Dzięki temu możliwe jest częściowe wycofanie dokonanych już zmian w bazie, jeżeli w jednym z kolejnych kroków podczas wprowadzania (lub usuwania) większej porcji danych będziemy chcieli wycofać ostatnią operację, nie wycofując równocześnie poprzednich. W przypadku awarii systemu odtwarzanie stanu bazy obejmie oczywiście najbardziej zewnętrzną transakcję.

After-imaging[17], Roll Forward Recovery[18]

O ile system transakcji oraz przechowywania obrazu pierwotnego zabezpiecza użytkownika przed utratą danych na skutek awarii systemu, o tyle jest on bezradny w przypadku fizycznego uszkodzenia nośnika. Jeżeli dane zawarte w bazie są szczególnie ważne, wówczas możemy zlecić systemowi oddzielne zapamiętywanie po zakończeniu transakcji wszystkich zmian, które zostały w niej wprowadzone. Mechanizm ten zwany after-imaging nie jest automatyczny. Wymaga on zadeklarowania lokalizacji pliku *.ai z obrazem końcowym w jednej z opcji startowych systemu. Idealnym rozwiązaniem jest lokalizacja plików *.ai oraz *.bi na różnych dyskach, ponieważ prawdopodobieństwo utraty obydwu plików jednocześnie jest wtedy minimalne.

Wykorzystanie mechanizmu after-image wiąże się z wprowadzeniem procedury regularnego składowania stanu bazy. W przypadku awarii nośnika należy odtworzyć stan bazy z ostatniego pełnego składowania, a następnie, wykorzystując odpowiednie narzędzia oraz plik *.ai, doprowadzić bazę do stanu bezpośrednio poprzedzającego awarię. Istotne jest ustalenie właściwego protokołu składowania, ponieważ nadmierny rozrost pliku *.ai może być przyczyną nieprzewidzianych kłopotów (np. zapełnienie dysku zawierającego ten plik). Proces odtwarzania stanu bazy na podstawie zapisów kolejnych modyfikacji nazywany jest odtwarzaniem sukcesywnym (Roll Forward Recovery).

Składowanie stanu bazy

System PROGRESS dostarcza bardzo wielu możliwości składowania stanu bazy. Służy do tego narzędzie PROGRESS Backup. Umożliwia ono dokonanie składowania w tle (online backup) bez konieczności przerywania pracy systemu. Możliwości PROGRESS Backup zależą od systemu operacyjnego i trybu użytkowania bazy. W systemach DOS, OS/2 oraz BTOS/CTOS możliwe jest

dokonanie pełnego bądź przyrostowego[19] składowania na dwa rodzaje nośników:
- dyskietki,
- pojedynczy zbiór na dysku twardym.

W systemach DOS i OS/2 można przenieść taki zbiór na taśmę za pomocą narzędzi systemowych. Bazy pracujące pod kontrolą systemów UNIX lub VMS mogą być bezpośrednio składowane na jeden z następujących nośników:
- dyskietki,
- taśmę 9-cio ścieżkową,
- taśmę kasetową,
- dysk kasetowy,
- niezależne urządzenie dyskowe,
- oddzielny plik w obrębie systemu,
- oddzielny plik pozasystemowy.

Nie jest możliwe dokonanie składowania w tle w systemach pracujących bez podziału pamięci (jak DOS, czy BTOS) oraz w przypadku, gdy baza użytkowana jest w trybie jednodostępnym (singleuser mode). Jeżeli baza jest dużych rozmiarów, mamy możliwość dokonania składowania przyrostowego. Zamiast całości bazy zapamiętywana jest wówczas jedynie informacja z tych jej sektorów, które uległy modyfikacji. Dzięki temu mechanizmowi pełne składowanie może odbywać się raz na tydzień lub miesiąc, w zależności od intensywności zmian w bazie.

7. Mechanizmy wewnętrzne a efektywność systemu

Architektura CLIENT/SERVER

Z myślą o przetwarzaniu danych w środowisku sieciowym projektanci systemu już od pierwszych wersji rozdzielili funkcje serwera bazy danych od procesów klienta. Proces klienta komunikuje się z użytkownikiem, obsługując stronę zewnętrzną aplikacji oraz interfejs użytkownika. Proces serwera zarządza bazą danych, kontrolując dostęp i modyfikacje danych. Technika ta znana pod nazwą cooperative processing[20] daje znaczące korzyści czasowe zauważalne w dużych, wieloużytkownikowych systemach.

Przetwarzanie wielotorowe (MULTI-THREADED/MULTI-SERVER processing)

Rozwiązaniem zwiększającym wydajność systemu w środowisku rozległym jest zastosowanie przetwarzania wielotorowego na poziomie serwerów[21]. Każdy z lokalnych procesów klientów komunikuje się z bazą poprzez własny proces serwera. Dzięki temu ograniczana jest komunikacja międzyprocesowa. Serwery obsługujące żądania odległe komunikują się zazwyczaj z wieloma procesami klientów. Aby nie dopuścić do powstania wąskiego gardła, którym byłby z pewnością przeciążony serwer, wprowadzony został dodatkowy proces administrujący (ang. broker), który rozdziela nadchodzące zgłoszenia pomiędzy najmniej obciążone procesy serwerów. Wraz ze wzrostem obciążenia systemu (włączaniem się nowych użytkowników) broker uruchamia nowe procesy serwerów, starając się utrzymać jednakowy czas średni odpowiedzi systemu na żądanie.

Zmienna długość rekordu

Zapamiętywanie danych ze zmienną długością jest nie tylko wygodne z punktu widzenia programisty. Podstawowa korzyść płynąca z zastosowania tej techniki to oszczędność miejsca na dysku dochodząca w niektórych systemach do 60%. Rozmiar danych dyskowych ma duży wpływ na szybkość działania systemu, a więc uzyskiwane są również korzyści czasowe.

Mechanizm indeksów

Indeksacja zawartości pliku według pewnego klucza jest świadomym działaniem programisty. Zadaniem indeksu jest nie tylko organizacja zgromadzonej informacji w zadanym porządku, ale również zapewnienie jak najkrótszego czasu wyszukiwania danych. Efektywność tej techniki zależy w dużej mierze od wiedzy programisty o sposobach wykorzystania projektowanego systemu przez jego przyszłych użytkowników, a także od umiejętności prawidłowego wykorzystania tej wiedzy. Nawet jednak najlepiej zaprojektowana struktura indeksów jest tylko na tyle wydajna, na ile wydajny jest algorytm indeksacji. W systemie PROGRESS wykorzystano metodę znaną jako skondensowane wielopolowe B-drzewo[22]. Jest to bardzo elastyczny i wydajny algorytm, mający zdecydowany wpływ na ogólną efektywność systemu.

8. Polska wersja językowa

PROGRESS Software Corporation stara się dostosowywać dostarczane oprogramowanie do wymogów tych krajów, w których jest ono dostępne. System PROGRESS operuje na rozszerzonym zestawie znaków ASCII, a więc pozwala na posługiwanie się alfabetem narodowym. Istnieje również możliwość zdefiniowania własnych reguł kolejności znaków alfabetu, co jest niezbędne dla pełnej użyteczności systemu.

Zostały opracowane również specjalne wersje PROGRESS-a dedykowane dla poszczególnych krajów. W chwili obecnej istnieją wersje: holenderska, francuska, włoska, szwajcarska, angielska, niemiecka, hiszpańska, duńska, norweska, szwedzka oraz fińska. Oprócz z góry przygotowanych tablic kolejności znaków są one wyposażone w pełen zestaw komunikatów systemowych w danym języku. Uwzględniają również specyficzne dla danego kraju reguły pisania daty, godziny czy liczb dziesiętnych.

Opracowanie każdej wersji językowej to olbrzymia praca. W przypadku Polski naj-

ważniejsza jej część polegająca na przetłumaczeniu ok. 2000 komunikatów systemowych (wraz z kilkuwierszowymi objaśnieniami) została już wykonana. Ponieważ od listopada 1991 PROGRESS daje możliwość deklaracji własnych tablic sortowania i konwersji, w chwili obecnej zaspokaja on większość potrzeb polskiego użytkownika. Pełna polska wersja językowa, która jest przygotowywana przez PSC wraz z CSBI, obejmie dodatkowo przetłumaczenie wszystkich programowych elementów środowiska PROGRESS-a (Data Dictionary, Fast Track, Results), a także włączenie Polski, jako kraju wybieranego opcją startową systemu (o ile uda się do tego czasu stworzyć polski standard tablicy kodowej).

• Korzystanie z systemu

1. Projektowanie struktur danych

Pierwszą fazą w tworzeniu nowej aplikacji jest zaprojektowanie struktur danych. Informacja zawarta w bazie powinna być kompletna. Od umiejętnego jej pogrupowania oraz poindeksowania zależy wydajność systemu. W krańcowym przypadku brak logiki w strukturach danych może doprowadzić do całkowitej bezużyteczności systemu, mimo uwzględnienia wszystkich niezbędnych informacji.

W systemie PROGRESS definiowanie struktur danych odbywa się w trakcie interakcyjnej pracy ze słownikiem danych. Możliwe jest również wcześniejsze przygotowanie takich definicji i wprowadzenie ich za pomocą opcji LOAD Data Definitions z menu Admin w Data Dictionary. Jeszcze innym sposobem jest wykorzystanie komend DDL[23] zawartych w oferowanym przez PROGRESS standardzie SQL. Podstawową metodą pozostaje jednak bezpośrednia praca ze słownikiem danych.

W przypadku konieczności zmiany istniejącej definicji można operację taką przeprowadzić w dowolnym momencie, również na wypełnionej danymi bazie. Należy jednak przy tym pamiętać, że nie jest dozwolona zmiana typu danych.

2. Budowa aplikacji prototypowej

Mając przygotowane struktury danych, możemy przystąpić do budowy funkcjonalnego szkieletu aplikacji. Bardzo pomocnym narzędziem jest tutaj FAST TRACK. Za pomocą edytora menu możemy stworzyć niemal dowolną strukturę menu. Poszczególnym punktom menu przypisujemy już istniejące, bądź dopiero planowane funkcje. Menu tworzone za pomocą FAST TRACK-a można w dowolnej chwili przeprojektować, a równocześnie jest ono cały czas gotowe do wykorzystania.

Operacje przeprowadzane na bazie danych wiążą się zazwyczaj z konkretnymi formatkami ekranowymi. Wbudowany w FAST TRACK-a edytor formatek ekranowych daje nam możliwość projektowania zawartości ekranu w sposób interakcyjny. Jest to bardzo wygodne, ponieważ zwalnia programistę od konieczności jawnego deklarowania dziesiątków szczegółów niezbędnych do opisania formatki. Ma to oczywisty wpływ na tempo pracy nad aplikacją.

Jeżeli operacje związane ze zdefiniowaną uprzednio formatką mieszczą się w standardowym zestawie obejmującym przegląd, modyfikację, dopisanie i usunięcie danych, możemy wykorzystać generator procedur QBF.

Inną standardową czynnością wykonywaną często w aplikacjach jest generacja raportów. Tu również FAST TRACK daje nam narzędzie umożliwiające szybkie, interakcyjne definiowanie struktury raportu. Możemy niezależnie definiować stronę tytułową, nagłówki i stopki poszczególnych stron, podsumowania częściowe i całościowe raportu, a nawet tworzyć raporty zagnieżdżone, oparte o dane z plików powiązanych relacją 1-n.

W ten sposób, bez pisania pojedynczej linii kodu, możemy stworzyć prototypowe środowisko pracy użytkownika, dające mu możliwość wykonywania większości operacji dostępnych w kompletnym systemie. Niektóre fragmenty aplikacji, a w szczególności procedury przetwarzające dane w sposób niewidoczny dla użytkownika, muszą być napisane w 4GL. Edytor menu umożliwia włączenie takich procedur do prototypu, który w tym momencie może stać się wersją docelową. Z biegiem czasu możliwe jest zastępowanie poszczególnych fragmentów prototypu odpowiednimi procedurami języka 4GL, dla wielu użytkowników jest to jednak nieopłacalne.

3. Tworzenie aplikacji w 4GL

Chociaż wykorzystanie FAST TRACK-a daje duże korzyści czasowe i organizacyjne, to jednak niektóre zastosowania wymagają napisania dużej części lub nawet całości oprogramowania w języku PROGRESS 4GL. Zmuszać do tego mogą szczególne wymagania dotyczące wyglądu lub obsługi menu, zastosowanie nietypowych trybów edycji lub wiele innych powodów.

PROGRESS Editor

W języku PROGRESS 4GL pisze się, wykorzystując wbudowany edytor PROGRESS-a lub dowolny inny edytor plików ASCII dostępny w danym systemie operacyjnym. Edytor wbudowany umożliwia nam stałą kontrolę poprawności programu. Procedura może zostać w każdej chwili

uruchomiona, zaś jakikolwiek błąd kompilacji bądź wykonania jest natychmiast raportowany. Edytor wyposażony jest we wszystkie standardowe funkcje z wyszukiwaniem oraz kopiowaniem fragmentów tekstu włącznie. Akcje przypisane poszczególnym klawiszom mogą początkowo wydać się zbyt udziwnione, jednak ich niezaprzeczalną zaletą jest niezależność od systemu operacyjnego. Edytor PROGRESS-a pracuje wszędzie tak samo, przyzwyczajać się trzeba tylko raz.

PROGRESS Help

Klawisz F2 służy w PROGRESS-ie do wywoływania pomocnika. Dotyczy to zarówno środowiska pracy programisty (edytora), jak i gotowych aplikacji, gdzie przypisanie funkcji pomocy klawiszowi F2 jest zalecane, choć nie obowiązkowe. W środowisku edytora naciśnięcie klawisza pomocy powoduje pojawienie się menu z zestawem kilkunastu funkcji. Można wówczas skorzystać ze skorowidza błędów, zawierającego poszerzone opisy wszystkich możliwych błędnych sytuacji, zasięgnąć informacji na temat składni języka, dostępnych funkcji, operatorów i zastrzeżonych słów kluczowych, przejrzeć listę wszystkich ograniczeń PROGRESS-a, przypomnieć sobie funkcje poszczególnych klawiszy, przejść do Data Dictionary lub modułu bibliotekarza, obejrzeć zawartość bieżącej kartoteki systemowej, wyjść do systemu operacyjnego, a także zakończyć sesję.

Korzystanie z bibliotek gotowych rozwiązań

W języku PROGRESS procedury mogą być przechowywane w postaci źródłowej, bądź skompilowanej do tzw. r-kodu. Procedury skompilowane można z kolei umieszczać w zbiorach bibliotecznych, co nie tylko porządkuje pracę, ale również przyspiesza wykonanie programu.
Wraz z systemem dostarczane są biblioteki procedur spakowanych w postaci źródłowej. Procedury te napisane przez doświadczonych programistów PSC można włączyć do własnej aplikacji lub posłużyć się nimi jako wzorem przy pisaniu własnych wersji. Procedury przykładowe mają charakter uniwersalny i obejmują większość „stałych fragmentów gry", z którymi styka się programista podczas pracy nad aplikacją. Należą do nich różne formaty menu, procedur wyboru, przeglądów, obliczeń matematycznych itp. W niektórych przypadkach mogą one stanowić alternatywę dla funkcji FAST TRACK-a.

Korzystanie z generatorów programu

Działanie FAST TRACK-a polega na wykorzystaniu procedur uniwersalnych oraz danych zgromadzonych w specjalnie zmodyfikowanej strukturze bazy. Dzięki temu aplikacja FASTTRACK-owa jest spójna i łatwiejsza do kontroli. Jednakże na specjalne życzenie programisty FAST TRACK jest w stanie wygenerować fragment programu w 4GL ekwiwalentny w stosunku do dowolnej ze swoich funkcji. Fragmenty te, bezpośrednio lub po modyfikacji, można wykorzystać w tekście własnej aplikacji.
Również w Data Dictionary istnieje możliwość skorzystania z generatora fragmentów kodu. Generuje on gotowe pliki włączane zawierające złożone instrukcje ASSIGN, FORM oraz DEFINE WORKFILE.

Wykorzystanie raportów słownika danych i kompilatora

Podczas pracy nad aplikacją ważnym elementem jest zapewnienie możliwości przeglądu wszystkich zdefiniowanych struktur. Służy do tego podmenu Reports w Data Dictionary.
Inne możliwości raportowania, również istotne dla programisty, ma kompilator PROGRESS-a. Dają je opcje LISTING oraz XREF komendy COMPILE. Pierwsza z nich generuje kompletny listing programu zawierający:

- nazwę kompilowanego pliku,
- nazwę bazy danych,
- datę i godzinę kompilacji,
- numer każdej linii procedury,
- numer bloku, do którego należy każda instrukcja,
- kompletny tekst wszystkich plików włączonych,
- pełne nazwy wszystkich podprocedur.

Opcja XREF generuje listę powiązań[24] pomiędzy obiektami procedur i baz danych. Każda linia opisuje jedno powiązanie za pomocą następujących pięciu elementów:

- nazwa procedury,
- nazwa pliku (procedura może być w kilku),
- numer linii,
- typ powiązania,
- identyfikator obiektu.

Przy dużym rozmiarze aplikacji korzystanie z wymienionych możliwości raportowania jest bardzo pomocne, a czasami nawet niezbędne.

4. Plany rozwoju systemu

Zazwyczaj, jeżeli jakaś firma ogłasza swoje zamierzenia, traktujemy to niepoważnie. Rzeczywiście, w większości przypadków realizacja planów nie następuje zgodnie z podanym harmonogramem. Niektóre z planów pozostają na zawsze w sferze zamierzeń. PROGRESS Software Corporation jest w tej dziedzinie chlubnym wyjątkiem. Do swoich zobowiązań podchodzi bardzo poważnie, a kolejne wersje produktów wydawane są z regularnością szwajcarskiego zegarka.

Wersje produktów PSC oznaczone są numerem oraz dużą literą łacińską. Każdy kolejny numer oznacza nową generację produktów o rozszerzonych cechach użytkowych. Równocześnie poszerzana jest lista dostępnych platform sprzętowych.

Ponieważ realizacja wszystkich założeń nowej generacji produktu jest procesem długotrwałym, jest ona dzielona na szereg etapów, w których opracowywane są kolejne wersje wewnątrz danej generacji. Wersje te, oznaczane literami, wzbogacane są o kolejne cechy, aż do całkowitej realizacji pierwotnego projektu. W ramach tych wersji mogą również zaistnieć pewne rozszerzenia sprzętowe, nie wychodzące jednak zazwyczaj poza już oprogramowane klasy sprzętu.

Ze względu na stały kontakt z użytkownikami plany rozwoju systemu są cały czas dopasowywane do bieżących potrzeb rynku. Przykładem elastyczności może być tutaj błyskawiczna reakcja na wzrost popularności systemów okienkowych.

Poniżej wymieniono główne kierunki rozwoju systemu PROGRESS.

Opracowanie zaawansowanego środowiska CASE

Narzędzia CASE nie są zwykłymi generatorami aplikacji, chociaż generacja kodu może być jedną z funkcji środowiska CASE. Zastosowanie narzędzi tego rodzaju wiąże się przede wszystkim z obraną metodologią tworzenia oprogramowania i wspomaga wszystkie jej etapy. W chwili obecnej PROGRESS nie posiada własnego środowiska CASE w pełnym tego słowa znaczeniu. Jednakże pozwala na wykorzystanie istniejących i uznanych w świecie środowisk Excelerator i Knowledgeware. W najbliższym czasie spodziewana jest też premiera narzędzia CASE powstałego w całości w PSC. Gwarancją jakości tego narzędzia ma być nazwisko kierownika tego projektu - Elliota Chikofskiego - jednego z najlepszych na świecie specjalistów w tej dziedzinie.

Ewolucja w kierunku systemu wielojęzycznego

Już dzisiaj dostępnych jest kilkanaście wersji narodowych PROGRESS-a. W wersjach tych wszystkie komunikaty systemowe wyświetlane są w języku użytkownika. Charakterystyczne dla danego języka są również reguły kolejności znaków, sposób pisania daty, godziny oraz liczb dziesiętnych. W wersji 7 PROGRESS-a przewidziano mechanizmy dynamicznego przełączania wersji językowej w trakcie pracy systemu. Umożliwi to zastosowanie PROGRESS-a wszędzie tam, gdzie potencjalni użytkownicy są różnych narodowości, a więc w obsłudze ruchu turystycznego, kompaniach wielonarodowych itp. Wprowadzenie tej możliwości wiąże się z modyfikacją struktury bazy, w której każdy identyfikator, etykieta czy komunikat będzie deklarowany w kilkunastu wersjach.

Zapewnienie bezpośredniej przenośności oprogramowania

W wersji 6 PROGRESS-a przeniesienie aplikacji w nowe środowisko wiąże się z koniecznością jej rekompilacji. W następnych wersjach problem ten przestanie istnieć, ponieważ r-kod generowany w trakcie kompilacji będzie całkowicie niezależny od sprzętu.

Nowe bramy do baz innych systemów

Do czasu ukazania się wersji siódmej przewiduje się udostępnienie możliwości bezpośredniej komunikacji z bazami SYBASE oraz DB2. Trwają również prace nad bramami do INFORMIX-a i INGRESS-a.

5. Od autora

Niniejszy tekst opracowałem na bazie materiałów dostarczonych przez firmę PROGRESS Software Corporation. Nie jest on z pewnością dokumentacją techniczną, omawia jednak większość elementów systemu dając czytelnikowi możliwość wyrobienia sobie poglądu zarówno na temat firmy, jak i jej produktu.

Oryginalna dokumentacja firmowa liczy ponad 4000 stron (13 podręczników). Do tego dochodzą publikacje uzupełniające, biuletyny techniczne, marketingowe, czasopisma i inne materiały związane z systemem. Zgromadzonych w nich informacji nie da się przedstawić w tak krótkim opracowaniu nie pomijając pewnych szczegółów, czy nawet całych tematów. Tych, którzy zainteresowani są uzyskaniem odpowiedzi na bardziej szczegółowe pytania, wypada odesłać do firmy CSBI, która dysponuje pełnym zestawem materiałów firmowych. Jej pracownicy mogą również służyć osobistym doświadczeniem nabytym podczas trzyletniej już pracy z systemem.

W opracowaniu świadomie nie podjąłem próby porównania systemu PROGRESS z jego konkurentami. Każdy system posiada swoje słabe strony, a ich wypominanie jest moim zdaniem przejawem braku szacunku dla dziesiątek programistów tracących swój czas i zdrowie przy pracy nad kolejnymi wersjami produktu. Nie podejmując krytyki systemów konkurencyjnych chciałbym jednak wyrazić przekonanie, że to właśnie PROGRESS, a nie ORACLE czy INFORMIX, jest systemem najlepiej dostosowanym do polskich warunków. Myślę, że po zapoznaniu się z niniejszą charakterystyką, a szczególnie jej częścią omawiającą przenośność systemu i różnorodność platform sprzętowych, przekonania takiego nabierze również czytelnik.

1 ang. dosł. co widzisz, to masz.
2 skr. ang. Client Communication Module - moduł komunikacyjny klienta.
3 skr. ang. Network Loadable Module - moduł rezydujący jako zadanie dodatkowe na serwerze sieci Novell.
4 skr. ang. Graphical User Interface - graficzny interfejs użytkownika.
5 co umożliwia wymianę danych z pakietami takimi jak: Lotus 1-2-3, Excel, Uniplex, MultiPlan, 20/20.
6 możliwe są dwie opcje: - z wyróżnionymi ogranicznikami pól i rekordów, - ze sztywnym formatem.
7 ang. Named Pipes, inaczej: kolejki FIFO - First In First Out.
8 ang. views.
9 skr. ang. Data Definition Language - język definicji danych.
10 ang. zwykłe ograniczenia dla nas nie istnieją.
11 skr. ang. 3rd Generation Language - język trzeciej generacji.
12 skr. ang. Host Language Call.
13 skr. ang. Host Language Interface.
14 dostępność poszczególnych języków zależy od użytkowanego sprzętu.
15 ang. dwufazowy protokół potwierdzeń.
16 ang. przechowanie obrazu pierwotnego.
17 ang. zabezpieczenie obrazu końcowego.
18 ang. odtwarzanie sukcesywne.
19 ang. incremental backup.
20 Brak jest jednoznacznego odpowiednika polskiego tego terminu.
21 ang. Multi-Threaded Multi-Server processing.
22 ang. compressed multi-field B-tree indexing.
23 skr. ang. Data Definition Language - język definicji danych.
24 ang. cross reference list.