ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996

Seria: INFORMATYKA z. 31 Nr kol. 1348

Dariusz AUGUSTYN

ARCHITEKTURA KLIENT-SERWER W HETEROGENICZNYCH SIECIACH KOMPUTEROWYCH NA PRZYKŁADZIE SYSTEMU ZARZĄDZANIA BAZĄ DANYCH FIRMY GUPTA TECHNOLOGIES$_{TM}$

Streszczenie. Artykuł dotyczy tematyki integracji baz danych i prezentuje zastosowanie produktów GUPTA SQL System w tworzeniu systemów obsługi baz danych, opartych na architekturze klient-serwer. W artykule opisywane są metody konfiguracji środowiska aplikacji klienta. Przedstawione zostały schematy komunikacji z różnymi serwerami baz danych w heterogenicznych środowiskach sieci lokalnych.

THE ENVIRONMENT OF DATABASE MANAGEMENT SYSTEM OF GUPTA TECHNOLOGIES$_{TM}$ AS AN EXAMPLE OF THE CLIENT-SERVER ARCHITECTURE IN HETEROGENEOUS LOCAL AREA NETWORKS

Summary. This article presents problems of a database integration in example of GUPTA SQL System products based on a client-server architecture. Methods of configurations of an application environments are described. Schemas of communications to different database servers in heterogeneous local area networks are presented too.

L'ARCHITECTURE CLIENT-SERVER DANS LES HETEROGENAUX RESAUX À BASE DU SYSTÈME DE GESTION DE BASE DE DONNÉES DE L'ENTREPRISE GUPTA TECHNOLOGIES.

Résumé. L'article concerne le problème d'intégration des bases do données et represente l'utulisation des produits de GUPTA SQL SYSTEM pour la création des syetèmes de service des bases de données utulisant l'architecture Client-Server. L'article presente aussi de differentes methodes de configuration de l'application du client. Les moyens de communication avec de différents servers de base de données dans les hetérogenaux résaux d'informatique sont presentés et etudiés.

1. Architektura klient-serwer

Achitektura klient-serwer polega na rozdzieleniu programu klienta i programu serwera, pozwalając na niezależny rozwój tych dwóch komponentów systemu. Jednym z najbardziej rozpowszechnionych zastosowań koncepcji klient-serwer są systemy zarządzania bazami danych.

Program uruchamiany na stacji roboczej, nazywany dalej aplikacją klienta (ang. front-end), przeznaczony jest głównie do obsługi ekranu, czyli formatek, menu itp. Nowoczesna aplikacja posiada graficzny interfejs użytkownika, tworzona jest na ogół z wykorzystaniem narzędzi programowych pracujących w trybie WYSIWYG i obiektowych języków, oparta jest na zdarzeniowej koncepcji sterowania.

Program serwera bazy danych (ang. back-end), dostarczany niezależnie od aplikacji, sterowany jest na ogół zdaniami języka SQL (ang. Structured Query Language) i optymalizowany jest pod kątem efektywności ich realizacji. Zaimplementowany w serwerze SQL jest językiem manipulowania danymi (ang. Data Manipulation Language), umożliwiającym dostęp do bazy danych z poziomu programu klienta.

Zróżnicowanie funkcjonalne aplikacji klienta oraz serwera pozwala na niezależny rozwój produktów dla obydwu tych klas oprogramowania i ich implemetacji dla różnych platform sprzętowo-programowych.

Zalety architektury klient-serwer ujawniają się szczególnie w środowisku sieci lokalnej. Żądania wyszukiwania czy modyfikacji danych, wyrażone w języku SQL, wysyłane są do programu serwera bazy danych przez programy klienta ze stacji roboczych sieci komputerowych. Program serwera bazy danych uruchamiany jest na wydajnym komputerze głównym sieci, tym samym, w którym zinstalowana jest pamięć dyskowa, przechowująca pliki bazy danych. Po realizacji przetwarzania ewentualne dane, stanowiące rezultat zadania SQL, są wysyłane z serwera bazy danych do stacji roboczej, dokładniej do zadania klienta.

Koncepcja klient-serwer umożliwia efektywne przetwarzanie ze względu na zmniejszenie obciążenia sieci poprzez minimalizację liczby przesyłów i rozmiaru przesyłanych danych (przesył zwrotny tylko danych spełniających kryteria zadania wyszukiwania) oraz z powodu możliwości wykorzystania wyróżnionego, wydajnego komputera z uruchomionym programem serwera bazy danych. Dla porównania w systemach o klasycznej architekturze, np. typu dBase, przetwarzanie odbywa się wyłącznie na komputerach stacji roboczych (efektywność przetwarzania uzależniona od wydajności stacji roboczych). W ramach takiego przetwarzania odbywa się transmisja całych plików danych. W uproszczeniu, komputer główny sieci komputerowej w kontekście klasycznego systemu zarządzania bazą danych funkcjonuje jedynie jako pamięć dyskowa dla plików bazy danych (dokładniej, udostępnia zdalny system plików i mechanizmy blokad regionów plików).

2. Charakterystyka systemu firmy GUPTA Technologies$_{TM}$

GUPTA SQL System$_{TM}$ posiada cechy charakterystyczne dla nowoczesnych produktów, przeznaczonych do tworzenia systemów obsługi baz danych. Przykładowo:

- efektywna w środowisku sieciowym architektura klient-serwer,
- produkty programowe, pozwalające na pracę w różnych konfiguracjach sprzętowo-programowych,
- serwer bazy danych z dostępem do bazy danych realizowanym poprzez język SQL,
- narzędzia programowe, umożliwiające interaktywną metodę tworzenia i uruchamiania aplikacji,
- możliwość tworzenia aplikacji dla popularnego systemu MS Windows, z graficznym interfejsem (oraz z wszystkimi innymi zaletami środowiska Windows),
- różnorodność i uniwersalność funkcjonalna produktów, wchodzących w skład systemu.

2.1. SQLBase - program serwera bazy danych

2.1.1. Język SQL, optymalizacja zapytań

SQLBase firmy Gupta Technologies$_{TM}$ jest produktem programowym realizującym zadania serwera bazy danych,wyrażone komendami języka SQL.

SQL jest nieproceduralnym językiem dostępu do bazy danych o strukturze relacyjnej. Umożliwia on formułowanie złożonych zadań w sposób zwięzły i przejrzysty oraz abstrahujący w dużym stopniu od sposobu realizacji zadania.

We współczesnych systemach zarządzania bazą danych (takich jak SQLBase firmy GUPTA Technologies$_{TM}$) optymalizacja zapytania (wyrażonego za pomocą języka SQL) jest zadaniem serwera bazy danych. Dla każdego zadania wyszukiwania serwer bazy danych na podstawie statystyk zawartych w bazie danych, dotyczących rozmiaru tablic, obecności indeksu, typu indeksów (indeksy oparte na b+-drzewach albo wykorzystujące funkcję mieszającą), głębokości drzewa indeksowego, dokonuje doboru indeksów, kolejności łączeń tablic czy tworzenia tablic tymczasowych. Serwer bazy danych wykorzystuje wbudowane algorytmy optymalizacji zapytań. Istnieje możliwość przełączenia (poprzez realizację instrukcji SET PLANONLY ON/OFF) w tryb śledzenia sposobu optymalizacji zapytania (etap "strojenia" systemu). W trybie tym zamiast wykonania komendy SQL uzyskuje się tzw. plan wykonania. Interpretacja planu wykonania może posłużyć do zmiany systemu indeksów w badanej bazie danych w celu poprawy efektywności wyszukiwania.

2.1.2. Stored Commands

SQLBase pozwala na przechowywanie wstępnie przetworzonych, pojedynczych, **zoptymalizowanych**, sparametryzowanych instrukcji SQL w bazie danych (ang. Stored Command). Możliwość ta wykorzystywana jest w celu przyspieszenia działania zadań wykonywanych wielokrotnie. W aktualnie wprowadzanej, najnowszej wersji SQLBase 6.0 zaimplementowano mechanizmy pozwalające na przechowywanie i wykonywanie całych procedur (ang. Stored Procedure). Mechanizm *stored procedures* zaimplementowany został wcześniej w innych systemach takich jak INFORMIX-OnLine czy Ingres.

2.1.3. System użytkowników i uprawnień

Język SQL w implementacji GUPTA SQL System$_{TM}$ (tzw. dialekt SQLBase) pozwala na definiowanie systemu użytkowników bazy danych. Uprawnienia dla poszczególnych użytkowników można przydzielać na poziomie całej bazy danych, poszczególnych tablic lub pojedynczych atrybutów określonych tablic.

Najwyższy poziom uprawnień posiada użytkownik o nazwie *SYSADM*. Może on dowolnie tworzyć i modyfikować uprawnienia innych użytkowników, zmieniać strukturę wszystkich tablic, perspektyw, indeksów. Nowo tworzeni użytkownicy (poprzez wykonanie przez użytkownika *SYSADM* komendy *GRANT*) mogą posiadać uprawnienia na poziomie bazy danych typu: *Connect*, *Resource*, *DBA*.

Użytkownik z poziomem uprawnień *Connect* (standardowy, najniższy poziom uprawnień na poziomie bazy danych) może uzyskać jedynie połączenie z bazą danych, nie może tworzyć nowych, własnych tablic, indeksów, perspektyw, nie może również modyfikować struktur tablic innych użytkowników ani tworzyć indeksów dla tablic innych użytkowników. Możliwość wykonywania operacji na zawartości poszczególnych tablic/perspektyw wynika z przydzielonych uprawnień na poziomie tablic/perspektyw przez ich właściciela lub użytkownika SYSADM.

Użytkownik z poziomem uprawnień *Resource* oprócz praw wynikających z poziomu *Connect* posiada możliwość tworzenia własnych tablic, perspektyw i indeksów oraz możliwość przydzielania uprawnień innym użytkownikom do tablic i perspektyw przez siebie tworzonych.

Użytkownik z poziomem uprawnień *DBA* (ang. Database Administrator) może wykonywać wszystkie operacje na obiektach bazy danych (np. tworzenie i modyfikowanie własnych tablic/perspektyw, indeksów czy modyfikowanie tablic, perspektyw, indeksów utworzonych przez innych użytkowników) z wyjątkiem tworzenia innych użytkowników i modyfikowania uprawnień innych użytkowników.

Uprawnienia przydzielane są również na poziomie tablic (dotyczy to głównie użytkowników z uprawnieniami na poziomie bazy typu *Connect*).

Uprawnienia typu:

- *Select* – wykonanie odczytu danych poprzez instrukcję SELECT,
- *Update* – aktualizacja całych wierszy lub wybranych kolumn,
- *Insert* – wstawianie wierszy,
- *Delete* – kasowanie wierszy,
- *Alter* – zmiana struktury tablicy,
- *Index* – tworzenie/zmiana indeksów, budowanych na atrybutach danej tablicy,

są związane z rodzajami operacji wykonywanych na zawartości tablic lub perspektyw i przydzielane mogą być przez właściciela tablicy czy perspektywy albo użytkownika *SYSADM*.

W systemie SQLBase zaimplementowano mechanizm synonimów tablic/perspektyw, pozwalający na selektywne deklarowanie przez właściciela lub użytkownika *SYSADM* widoczności tablic/perspektyw dla wybranych użytkowników.

2.1.4. Więzy integralności

Język SQL serwera SQLBase pozwala na definiowanie prostych więzów integralności (ang. Referential Integrity) na poziomie struktury bazy danych. Ewentualne warunki integralności dotyczą hierarchicznych powiązań logicznych pomiędzy atrybutami tablic (mechanizm *Primary Key - Foreign Key*).

Przy wprowadzaniu nowego wiersza możliwe jest wymuszenie sprawdzania istnienia tej samej wartości atrybutu w tablicy nadrzędnej. Istnieje możliwość zadeklarowania wybranych atrybutów tablicy nadrzędnej jako tzw. kluczy podstawowych - inaczej kluczy pierwotnych (ang. Primary Key) oraz wybranych atrybutów tablicy podrzędnej jako tzw. kluczy obcych (ang. Foreign Key). Po takim określeniu więzów integralności nie da się wprowadzić nowego wiersza do tablicy podrzędnej z wartością atrybutu *Foreign Key*, dla której nie istnieje wiersz z tą samą wartością atrybutu *Primary Key* w tablicy nadrzędnej. Sprawdzanie istnienia odpowiednich wierszy w tablicy nadrzędnej jest automatycznie realizowane przez serwer bazy danych.

Przy usuwaniu wiersza z tablicy nadrzędnej możliwe jest wykonanie określonej akcji w stosunku do wierszy tablicy podrzędnej, związanych poprzez wartość atrybutów z usuwanymi rekordami tablicy nadrzędnej. Przykładowo usunięcie wiersza z tablicy narzędnej może powodować usunięcie wszyskich wierszy z tablicy podrzędnej, dla której występuje równość wartości kluczy *Primary Key* i *Foreign Key*. Więzy integralności można również określić tak, by w sytuacji usuwania wierszy z tablicy nadrzędnej do atrybutów *Foreign Key* zgodnych z wartościami kluczy *Primary Key* usuwanych wierszy wpisywane były automatycznie wartości *NULL* (puste).

2.1.5. Transakcyjne przetwarzanie danych

Serwer SQLBase posiada wbudowane mechanizmy przetwarzania transakcyjnego. W uproszczeniu transakcją nazywamy fragment programu, wyodrębniony przez programistę, który powinien być wykonany w całości ze względu na utrzymanie poprawności danych. Poprawne wykonanie takiego fragmentu programu nazywane jest wypełnieniem transakcji.

W przypadku niepełnej realizacji fragmentu programu wykonane częściowe modyfikacje są automatycznie wycofywane (przez system zarządzania bazą danych), dzięki czemu spójność danych zostaje zachowana. Przykładowo, mechanizm ten, określany nazwą *Rollback*, wykorzystywany jest w sytuacji bezpośrednio po awarii zasilania systemu. W takiej sytuacji następuje automatyczne odtworzenie stanu bazy danych z dokładnością do ostatniej wypełnionej transakcji po ponownym uruchomieniu serwera bazy danych. Możliwe jest również wykorzystywanie tego mechanizmu jawnie, tzn. w trybie programowym poprzez wykonanie komendy *Rollback*, np. w ramach obsługi błędu programowego typu kolizja dostępu.

W serwerze z pakietu GUPTA SQL System$_{TM}$ zaimplementowany został system "zagłębionych" transakcji. W programie możliwe jest wyróżnienie (poprzez umieszczenie odpowiedniej komendy SQL) specjalnych miejsc (ang. Save Point) określających podział bieżącej transakcji. Dzięki temu możliwe jest, np. w ramach obsługi błędów programowych, połączenie wycofywania (przeniesienia) sterowania z poszczególnych fragmentów programów (do miejsc wyznaczonego przez *Save Point*) z jednoczesnym wycofywaniem (przez serwer bazy danych) modyfikacji danych, dokonanych w tych fragmentach (tzn. modyfikacji dokonanych po wystąpieniu wskazanego *Save Point*-u) bez wycofania całej transakcji.

W oparciu o mechanizmy transakcyjne możliwa jest poprawna realizacja archiwizacji bazy danych, nawet w sytuacji równoczesnego korzystania z bazy danych przez inne aplikacje (mechanizm *Backup Snapshot*). Kopia bazy danych jest poprawna, nawet jeżeli była wykonywana w trakcie realizacji transakcji wynikających z normalnej eksploatacji systemu. Utworzona kopia bazy danych dotyczy stanu bazy z momentu wydania komendy archiwizacji.

Baza danych w standardowej instalacji SQLBase złożona jest z pojedynczego pliku z zawartością bazy danych (*.DBS) i plików z obrazem transakcji (*.LOG). Pliki z obrazem transakcji, zawierające opisy transakcji wypełnionych i aktualnie realizowanych,

wykorzystywane są bezpośrednio w mechanizmie *Rollback*. Dla mechanizmu odtwarzania transakcji informacja o wypełnionych transakcjach (dokładniej, o transakcjach z uwzględnionymi modyfikacjami w pliku bazy DBS) nie jest istotna i w związku z tym pliki obrazu transakcji zawierające tylko takie informacje są przez serwer bazy danych automatycznie usuwane.

Dla celów podwyższenia bezpieczeństwa systemu instalację bazy danych można zrealizować tak, by właściwy plik bazy danych (DBS) i pliki obrazów transakcji były umieszczone na różnych dyskach (wolumenach). Dodatkowo możliwe jest przełączenie (wykonanie komendy *SET LOGBACKUP ON*) trybu pracy serwera dla wybranej bazy danych, polegające na zachowywaniu wszystkich plików transakcyjnych (również dla transakcji wypełnionych). Przy takiej konfiguracji w przypadku uszkodzenia właściwego pliku bazy danych (np. awaria nośnika magnetycznego czy uszkodzenie logiczne struktury wolumenu) po skopiowaniu ostatnio zarchiwizowanej bazy danych możliwe jest uaktualnienie jej zawartości na podstawie zachowanych plików transakcyjnych (umieszczonych na innym nieuszkodzonym dysku/wolumenie). Mechanizm ten nazywany *Rollforward* pozwala na odtworzenie bazy danych z dokładnością do ostatniej wypełnionej transakcji (komenda *ROLLFORWARD TO END*) lub z dokładnością do ostatniej wypełnionej transakcji, zakończonej przed dowolną, podaną chwilą czasową, poprzedzającą awarię (komenda *ROLLFORWARD TO TIME* <data-czas>).

2.1.6. Poziomy izolacji

W serwerze SQLBase (tak jak w nowoczesnych serwerach baz danych innych producentów) zaimplementowano funkcje deklarowania poziomów izolacji, pozwalające na określanie sposobu i zakresu blokady odczytu (blokady w trybie dzielonym) w transakcjach.

Poziom izolacji *RR* (ang. Repeatability Read) oznacza blokadę w trybie dzielonym wszystkich odczytanych rekordów do zakończenia bieżącej transakcji.

Poziom *CS* (ang. Cursor Stability) oznacza blokadę odczytanych danych z dokładnością do pojedynczego bieżącego rekordu (dokładniej, pojedynczej, "fizycznej" strony (lub stron) bazy danych z bieżącym rekordem).

W sytuacji zadeklarowania poziomu izolacji *RL* (ang. Release Locks) zadanie, w ramach którego realizowany jest odczyt rekordów, nie zakłada blokad w trybie dzielonym, ale jeśli dane byłyby w trakcie modyfikacji przez inne zadanie (objęte inną transakcją), to operacja odczytu nie dojdzie do skutku (błąd kolizji dostępu).

W serwerze zaimplementowany jest mechanizm weryfikacji zmian, umożliwiający wykrycie modyfikacji rekordu przez inne zadanie (tzn. wykrycie faktu zmiany rekordu pomiędzy kolejnymi odczytami). Dokładniej mechanizm ten polega na wykorzystywaniu pola ROWID o wartościach unikalnych dla danej tablicy i zmieniających się automatycznie w sytuacji jakiejkolwiek zmiany któregoś z pozostałych pól rekordu.

Typowa aplikacja dla umożliwienia maksymalnie współbieżnej realizacji innych zadań, przy zapewnieniu kontroli nad poprawnością danych, na ogół wykorzystuje poziom izolacji *RL* w połączeniu z mechanizmem *ROWID*-u (sprawdzenie "wersji" danych).

Zaimplementowano dodatkowo nietypowy poziom izolacji *RO* (ang. Read Only), w którym w ramach danego zadania dostęp do bazy możliwy jest jedynie w trybie odczytu, tzn. operacje modyfikacji są niedozwolone. Możliwe jest jednak wykonywanie operacji modyfikacji tych samych danych przez inne zadania. Zapewniona jest współbieżność zadań (zadanie z poziomem izolacji *RO* nie blokuje odczytywanych danych). Aktualność danych odczytywanych przez zadanie z ustawionym poziomem izolacji *RO* odpowiada momentowi ustawienia poziomu izolacji. W przypadku modyfikacji danych przez inne zadania DBMS sam wykonuje kopie zmodyfikowanych fragmentów bazy (pliki *.his), udostępniając te kopie, w przypadku gdy zadanie z poziomem izolacji *RO* odwoła się do danych, dla których nastąpiła wspomniana modyfikacja przez inne zadania.

2.2. SQLWindows – programowe środowisko tworzenia aplikacji klienta

Jedną z charakterystycznych cech systemu firmy GUPTA Technologies$_{TM}$ jest możliwość łatwego tworzenia aplikacji dla popularnego środowiska MS Windows, oczywiście z zachowaniem pełnej funkcjonalności końcowych produktów. Tworzenie i uruchamianie aplikacji z profesjonalnym, graficznym interfejsem możliwe jest dzięki wykorzystaniu programu SQLWindows.

SQLWindows jest środowiskiem, w którym procesy interaktywnego konstruowania formatek (np. rozmieszczania elementów graficznych czy sterujących) i "pisania" kodu źródłowego są ze sobą ściśle zintegrowane (przykładowo: zmiana poprzez "dorysowanie" elementu sterującego jest automatycznie, na bieżąco odzwierciedlana w kodzie programu i odwrotnie).

System SQLWindows pozwala na użycie w tworzonych aplikacjach standardowych elementów graficznych, stosowanych w innych aplikacjach MS Windows (opcjonalnie wzbogacając takie elementy o efekt 3D), jak również własnych elementów, specjalizowanych pod kątem wykorzystania do baz danych (np. *Table Window* - potocznie "browser", element mogący służyć do wyświetlania lub modyfikacji tablic bazy danych).

Począwszy od wersji SQLWindows 4.0 wprowadzono elementy programowania obiektowego. Obiektowość dotyczy w szczególności możliwości definiowania klas sterujących elementów graficznych tworzonej aplikacji.

Sterowanie w aplikacjach utworzonych w SQLWindows opiera się na zdarzeniowej koncepcji programowania.

W SQLWindows zapimplementowano funkcje wykorzystujące mechanizmy *DDE* (ang. Dynamic Data Exchange - protokół komunikacji pomiędzy aplikacjami), pozwalające

na współpracę programów utworzonych w tym systemie z innymi aplikacjami dla MS Windows. Mechanizmy DDE mogą przykładowo służyć do sterowania przez aplikację ("napisaną" w SQLWindows) niezależnymi programami (np. MS Excel), realizującymi funkcje grafiki prezentacyjnej dla informacji pochodzących z bazy danych. Aplikacja utworzona w SQLWindows może być nie tylko klientem, *DDE*, ale i serwerem *DDE*, przykładowo służąc jako program pośredniczący w komunikacji z bazą danych dla innej, niezależnej aplikacji MS Windows.

Począwszy od wersji SQLWindows 4.0 wprowadzono elementy graficzne (elementy typu *Picture*) i funkcje obsługujące mechanizmy *OLE* (ang. Object Linking Embedding). Przykładowo, dzięki technice *OLE*, w formatkach tworzonej aplikacji możliwe jest osadzenie obiektów typu obraz, dźwięk, animacja itd. Wsparcie mechanizmu od strony serwera bazy danych polega na możliwości wykorzystania w strukturach tablic pól typu *LONGCHAR* dla przechowywania takich amorficznych obiektów i dostępu do nich poprzez standardowe komendy języka SQL.

Elastyczność środowiska przejawia się również w możliwości bardzo łatwego dołączania zewnętrznych funkcji, pochodzących z bibliotek *DLL* (ang. Dynamic Link Library), utworzonych za pomocą innych, klasycznych języków programowania (C, PASCAL itp.).

W SQLWindows możliwe jest tworzenie aplikacji, w której komunikacja z użytkownikiem opiera się na wykorzystaniu efektownej techniki *Drag and Drop*.

2.3. Różnorodność produktów wchodzących w skład systemu

Uniwersalność systemu wynika z różnorodności programów wchodzących w skład GUPTA SQL System$_{TM}$.

W ramach SQL GUPTA System$_{TM}$ dostarczane są między innymi takie produkty programowe jak np.:

SQLWindows – interaktywny system przeznaczony do tworzenia aplikacji klienta dla środowiska MS Windows.

SQLBase – grupa programów realizujących zadanie serwera bazy danych dla zróżnicowanych konfiguracji sprzętowo-programowych.

SQLTalk – grupa programów przeznaczonych do realizacji:

- pojedynczych zapytań, formułowanych w języku SQL,
- zadań administratora np.:
  tworzenie:
  – tablic, indeksów, perspektyw, synonimów,
  – użytkowników, haseł i z nadawaniem praw dostępu,
  – kopii archiwalnych bazy danych i odtwarzanie bazy,
- importu i eksportu danych w różnych formatach
  (w tym "format" sekwencji komend SQL, tworzących bazę),

- skryptów (sekwencji instrukcji SQL)
  (np. skryptów do utworzenia struktury bazy danych)
  itp...

Dostarczana jest wersja programu pracującego w środowisku znakowym (DOS, SunOS, Windows NT - character mode) lub graficznym systemie MS Windows.

C/API – (ang. C Language Application Programming Interface) biblioteka funkcji realizujących dostęp do bazy danych z własnych programów lub bibliotek *DLL*, pisanych w języku C.

ReportWindows – interaktywne, wygodne narzędzie do tworzenia szablonów raportów wykorzystywanych w aplikacjach utworzonych za pomocą SQLWindows.

QUEST – interaktywny program wykorzystywany do łatwego formułowania zapytań do bazy danych, generowania raportów, wprowadzania danych. Uniwersalny, efektowny program należący do klasy *QBE* (ang. Query By Example) z rozbudowaną graficznie techniką konstrukcji zapytań i prezentacji wyników. Przy użyciu programu możliwe jest również realizowanie wielu funkcji administratora bazy danych. W aplikacjach utworzonych za pomocą SQLWindows możliwe jest zastosowanie obiektu realizującego funkcje QUEST-a (QUEST instaluje się w systemie MS Windows jako serwer *OLE*).

TEAM WINDOWS – program

- wspomagający tworzenie rozbudowanych aplikacji, należący do narzędzi klasy *Lower CASE*,
- stanowiący system zarządzania projektem aplikacji (elementami projektu),
  (Poszczególne elementy projektu typu: programy i biblioteki tworzone w SQLWindows, pliki pomocy, biblioteki *DLL*, dokumenty, bitmapy itp. przechowywane są w bazie danych (ang. Repository)),
- stanowiący system kontroli kolejnych wersji elementów projektu,
- uwzględniający poszczególne fazy tworzenia oprogramowania dla każdego elementu projektu:
  Development - faza tworzenia i rozwijania,
  Test/Shared - faza testu programów/udostępnienia bibliotek,
  Production - faza wdrożenia,
- przeznaczony dla zespołowego tworzenia oprogramowania (system użytkowników i uprawnień w repozytorium),
- realizujący zadania systemu komunikacji pomiędzy poszczególnymi osobami uczestniczącymi w realizacji projektu.

3. Praca w różnych konfiguracjach sprzętowo-programowych

3.1. Implementacje serwera bazy danych - SQLBase

3.1.1. Praca w trybie jednoużytkownikowym z lokalnym serwerem bazy danych

W najprostszej konfiguracji przetwarzanie danych może być realizowane na komputerze lokalnym użytkownika. Oprócz aplikacji klienta, uruchamiany jest program serwera lokalnego (w postaci programu dla systemu MS Windows), odbierający żądania wyrażone w postaci zadań języka SQL od właściwej aplikacji klienta. Komunikacja pomiędzy serwerem bazy danych (program *dbwindow.exe*) i aplikacją klienta odbywa się poprzez protokół *DDE*. (Fizyczne położenie pliku bazy danych może być różne, tzn. plik bazy danych może przykładowo znajdować się na dysku lokalnym albo na dysku sieciowym *file server*-a.)

Począwszy od wersji 4.0 w skład systemu SQLWindows wchodzi wersja jednoużytkownikowego, **wielozadaniowego** serwera bazy danych (program *dbwservr.exe*). Jednoużytkownikowy, wielozadaniowy serwer pozwala na jednoczesne połączenie z kilkoma aplikacjami (w ramach jednej sesji MS Windows), a tym samym może przykładowo pozwolić na testowanie sytuacji kolizji dostępu do bazy danych nawet w konfiguracji jednostanowiskowej.

3.1.2. Praca w trybie wielodostępnym w środowisku sieciowym

W instalacjach pracujących w środowisku sieciowym zadania wyrażone w języku SQL realizowane są przez program serwera bazy danych uruchamianego na wydajnym komputerze głównym sieci, na ogół różnym od komputera stacji roboczej.

Zaimplementowane zostały przez firmę GUPTA Technologies$_{TM}$ programy serwera bazy danych dla systemów takich jak:

- Novell NetWare 386,
- Novell NetWare 4,
- SunOS v.4.1.1 (system Unix dla komputerów firmy SUN Microsystems),
- DOS z komunikacją poprzez protokół NetBIOS,
- Windows NT,
- OS/2.

Najbardziej popularną w Polsce implementacją SQLBase jest program serwera dla systemu sieciowego Novell NetWare. Na program serwera składa się kilka modułów NLM (ang. Netware Loadable Modules): *dfd.nlm, dll.nlm, spxdll.nlm, dbnservr.nlm,* uruchamianych na komputerze *file server*-a systemu NetWare 386. Komunikacja aplikacji z serwerem odbywa się oczywiście za pośrednictwem protokołu SPX/IPX.

Zaimplementowano SQLBase dla komputerów firmy Sun Microsystems, Inc$_{TM}$, bazujących na architekturze SPARC RISC, z systemem operacyjnym Unix SunOs v4.1.1. Serwer realizowany jest w postaci jednego programu *dbuservr*. Komunikacja aplikacji z serwerem odbywa się za pośrednictwem protokołu TCP/IP. Do komunikacji wykorzystywany jest domyślnie port 2155/tcp. Implementacja ta do niedawna stwarzała perspektywę dotyczącą możliwości uruchamiania przyszłych, nowych wersji serwera bazy danych SQLBase na najnowszych maszynach wieloprocesorowych firmy Sun Microsystems,Inc$_{TM}$. Prawdopodobnie jednak ze względów komercyjnych (duża popularność i liczba komputerów typu Sun głównie w środowiskach akademickich) oraz z powodu wprowadzonych zmian w nowszych wersjach systemu operacyjnego Solaris 2.X zapowiadane jest zaniechanie produkcji nowych, kolejnych wersji SQLBase dla komputerów rodziny Sun.

Możliwa jest realizacja instalacji w wielodostępnym środowisku sieci komputerowej, w której serwer bazy danych jest programem wykonywalnym dla systemu DOS (program *dbxservr.exe*). W takiej konfiguracji jeden z komputerów sieci pracuje wyłącznie jako dedykowany serwer bazy danych z komunikacją poprzez protokół NetBIOS.

Począwszy od wersji 5.2 dostępna jest implementacja serwera bazy danych dla systemu Windows NT (program *dbntsrv.exe*). Komunikacja z serwerem może się odbywać między innymi poprzez protokół *NetBEUI* (rozszerzony protokół NetBIOS). Wersja serwera dla Windows NT z powodu implementacji samego systemu operacyjnego na różne platformy sprzętowe może w przyszłości pozwolić na rozszerzenie zakresu zastosowań na architektury wieloprocesorowe.

3.2. Konfiguracja środowiska aplikacji klienta

Typowe aplikacje klienta, tworzone w oparciu o produkty firmy GUPTA Technologies$_{TM}$, są na ogół programami dla systemu MS Windows, utworzonymi z wykorzystaniem:

- **narzędzia programowego SQLWindows**,
- zintegrowanych środowisk do tworzenia programów w języku C, np. Microsoft C++ czy Borland C++, współpracujących z bibliotekami umożliwiającymi dostęp do baz danych (poprzez *Runtime*, zawierający wspomniane biblioteki, dostarczany przez firmę GUPTA Technologies$_{TM}$).

Aplikacje klienta mogą być również tworzone w postaci programów dla środowiska znakowego w systemach: DOS, SunOs czy Windows NT (np. *sqltalk.exe*, *sqlutalk*, *sqlnttlk.exe*, własne programy napisane w C z wykorzystaniem bibliotek C/API).

Typowa aplikacja klienta komunikująca się z serwerami SQLBase może współpracować jednocześnie (chociaż nie z dokładnością do pojedynczego zdania wyrażonego w języku SQL) z kilkoma bazami danych obsługiwanymi przez różne serwery bazy danych (ang. Multidatabase Application).

Bez specjalnych trudności można stworzyć system obsługi baz danych w środowisku sieci heterogenicznej, tzn. takiej, w której poszczególne komputery (na których uruchomione są odpowiednie serwery baz danych) mogą różnić się architekturą i systemem operacyjnym. Ze strony programu klienta warunkiem uzyskania połączenia jest obecność odpowiednich bibliotek komunikacyjnych w środowisku aplikacji klienta. W samym programie klienta operuje się jedynie nazwą bazy danych, całkowicie abstrahując od typu programu serwera.

O możliwości współpracy aplikacji klienta z różnorodnymi serwerami bazy decyduje obecność bibliotek komunikacyjnych dla odpowiednich protokołów sieciowych. Plik konfiguracyjny (uwzględniany przez *Runtime*, dystrybuowany z aplikacją) określa pośrednio listę protokołów, z wykorzystaniem których następuje komunikacja z odpowiednimi serwerami bazy danych.

Konfiguracja produktów programowych (aplikacje klienta, programy serwerów bazy danych) określona jest poprzez wpisy w pliku o nazwie *SQL.INI*.

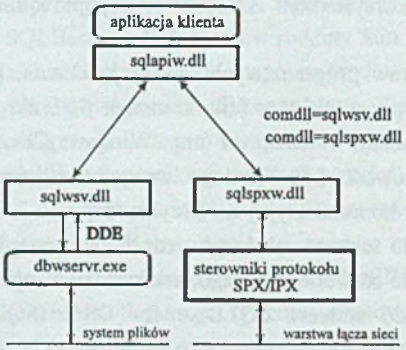
W pliku *SQL.INI* w sekcji *winclient.dll* (ang. Windows Client) dotyczącej konfiguracji aplikacji klienta dla środowiska MS Windows wpisy *comdll* (ang. Communication DLL) określają nazwy bibliotek komunikacyjnych. Przykładowo:

sqldbw.dll – dostęp do serwera lokalnego, jednozadaniowego,
sqlwsv.dll – dostęp do serwera lokalnego, wielozadaniowego,
sqlpipe.dll – dostęp do serwera SQLBase for OS/2 (SQLBase for Windows NT lokalnie),
sqlspxw.dll – dostęp do serwera SQLBase dla NetWare 386, poprzez protokół SPX/IPX,
sqlspxw4.dll – dostęp do serwera SQLBase dla NetWare 4,
sqltifsw.dll,
sqltinvw.dll – dostęp do serwera SQLBase for Unix, poprzez protokół TCP/IP,
sqlnbiow.dll – dostęp do serwera SQLBase for DOS, poprzez protokół NetBIOS), (w nowszej wersji, również dostęp do serwera SQLBase for Windows NT),
sqlirtw.dll – dostęp poprzez router do serwera DBMS INFORMIX,
sqlingw.dll – dostęp poprzez router do serwera DBMS Ingres,
sqlodbw.dll – komunikacja poprzez *ODBC* (wprowadzone od wersji 5.0 systemu SQLWindows).

W konfiguracjach, w których serwer bazy danych jest programem uruchamianym na komputerach z systemem operacyjnym Unix (program SQLBase dla SunOS), na stacjach roboczych uruchamiane są odpowiednie programy obsługi protokołu sieciowego TCP/IP z pakietu LAN WorkPlace for DOS firmy Novell® albo PC/TCP firmy FTP Software®. W zależności od rodzaju pakietu należy wybrać odpowiednią bibliotekę komunikacyjną *sqltinvw.dll* albo *sqltifsw.dll*.

Kolejność wystąpień nazw bibliotek komunikacyjnych wyznacza pośrednio kolejność "poszukiwania" wśród rodzajów serwerów odpowiedniej bazy danych. W typowej aplikacji nie determinuje się serwera bazy danych ani protokołu operując jedynie nazwą bazy danych (zapewniając tym samym wspomnianą wcześniej niezależność od rodzaju serwera).

Każda aplikacja klienta, tworzona z wykorzystaniem narzędzi firmy GUPTA Technologies$_{TM}$, realizując dostęp do baz danych wywołuje odpowiednie funkcje z biblioteki *sqlapiw.dll* (wchodzącej w skład runtime-u SQLWindows), z której to z kolei następują wywołania odpowiednich bibliotek komunikacyjnych, na podstawie wpisów *comdll*. Przykładowo ogólny schemat komunikacji aplikacji klienta skonfigurowanej do pracy z serwerem lokalnym i serwerem dla systemu Novell Netware przedstawiony jest na rysunku 1.



Rys. 1. Schemat komunikacji aplikacji z serwerem lokalnym i serwerem dla Novell Netware

Fig. 1. The communication schema of application connected to a local database server and a server for Novell Netware

4. Współpraca z serwerami baz danych innych producentów

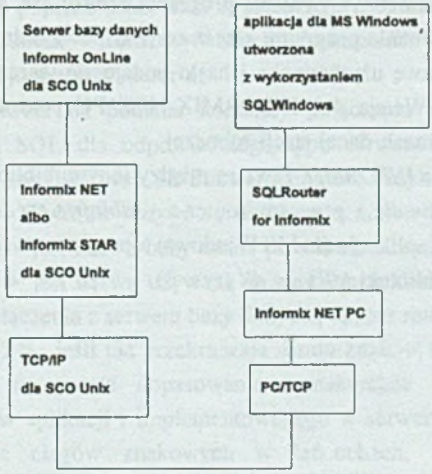
Dla systemów o bardzo wysokich wymaganiach efektywnościowych (duża liczba jednocześnie otwartych połączeń i jednocześnie przeprowadzanych transakcji, pożądany szybki czas reakcji) istnieje możliwość wykorzystania programu serwera bazy danych pochodzącego od niezależnego producenta, uruchamianego na szybkich mikro- lub minikomputerach.

4.1. Opisy przykładowych, testowych instalacji, wykorzystujących oprogramowanie router-ów

Firma GUPTA Technologies$_{TM}$ dostarcza pakiet *SQLNetwork*, w skład którego wchodzą router-y do serwerów innych baz danych, takich jak INFORMIX, Ingres, Oracle, SQLServer, Sybase, SQL/400 (dla komputera AS/400) czy DB2. W takich rozwiązaniach typowa aplikacja z graficznym interfejsem, utworzona w SQLWindows, współpracuje z bardzo wydajnymi serwerami innych, uznanych firm. Połączenie z takim serwerem możliwe jest oczywiście dopiero po zainstalowaniu podstawowego oprogramowania obsługi sieciowego protokołu komunikacyjnego, pakietu oprogramowania klienta, odpowiedniego dla rodzaju serwera (dostarczanego przez producenta serwera) oraz pakietu router-a dla odpowiedniego typu serwera (dostarczanego przez firmę GUPTA Technologies$_{TM}$).

4.1.1. Komunikacja z serwerem DBMS INFORMIX

Dla przykładu przedstawiono opis zrealizowanej instalacji umożliwiającej współpracę aplikacji utworzonej w SQLWindows z serwerem INFORMIX OnLine 5.0 for UNIX®, uruchomionym na komputerze PC z systemem operacyjnym SCO UNIX System V/386® (rys. 2.).



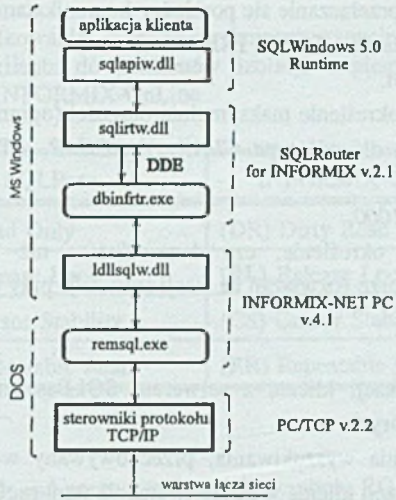
Rys. 2. Schemat ogólny połączenia aplikacji klienta z DBMS INFORMIX
Fig. 2. The general schema of connnection of a client application to DBMS INFORMIX

Na stacji roboczej (komputer klasy PC) zainstalowano odpowiednio:

- pakiet obsługi protokołu TCP/IP, dostarczany przez firmę FTP Software®: PC/TCP for DOS, ,
- pakiet oprogramowania klienta, dostarczany przez firmę Informix Software, Inc.: INFORMIX-NET PC® ver 4.1,
- pakiet router-a "GUPTA" <-> "INFORMIX", dostarczany przez firmę GUPTA Technologies$_{TM}$: SQLRouter for INFORMIX® ver. 2.1.

INFORMIX-NET PC w wersji 4.1 zrealizowany jest w postaci DOS-owego programu rezydentnego *remsql.exe* (ang. remote SQL). Właśnie ze względu na zastosowaną, wczesną wersję pakietu INFORMIX-NET PC **dla DOS** w zasadzie w celu zapewnienia komunikacji z serwerem bazy z pakietu PC/TCP istotne są moduły dla środowiska DOS (tzn. opcjonalne są moduły dla środowiska MS Windows). Komunikacja z serwerem INFORMIX-a odbywa poprzez usługę o nazwie *sqlexec* z portem *1525/tcp*. INFORMIX-NET PC stanowi końcową warstwę dla "klienckich" aplikacji INFORMIX-owych (np. znakowy, interaktywny program narzędziowy *isql.exe*, umożliwiający kierowanie do serwera zadań wyrażonych w języku SQL czy własne, znakowe aplikacje, tworzone w oparciu o narzędzie I4GL dla DOS, dostarczane przez Informix Software,Inc$_{TM}$). Nazwę komputera z serwerem INFORMIX-a, użytkownika, hasła, usługi należy podać poprzez ustawienia odpowiednich zmiennych środowiskowych DOS: *SQLHOST*, *SQLUSER*, *SQLPASS*, *SQLSERVICE* przed wywołaniem programu *remsql.exe* albo jako parametry wywołania programu *remsql.exe*, bądź też po załadowaniu *remsql.exe* poprzez wykonanie programu *setnet.exe* z odpowiednimi parametrami. Niestety parametry te, w tym nazwę użytkownika i hasło podaje się jeszcze przed uruchomieniem systemu MS Windows. Wersja 4.1 INFORMIX-NET PC pozwala na zestawienie tylko jednego połączenia w ramach danej stacji roboczej.

Pakiet *SQLRouter for INFORMIX* zawiera między innymi bibliotekę komunikacyjną dla aplikacji klienta - *sqlirtw.dll* i program router-a - *dbinfrtr.exe*. Komunikacja pomiędzy programem *dbinfrtr.exe* (aplikacja dla MS Windows) a programem *remsql.exe* (TSR) odbywa się za pośrednictwem biblioteki *ldllsqlw.dll* (rys. 3).



Rys. 3. Schemat komunikacji aplikacji z DBMS INFORMIX, poprzez moduły SQLRouter i INFORMIX-NET PC 4.0

Fig. 3. The communication schema of application connected to DBMS INFORMIX using modules: SQLRouter and INFORMIX-NET PC 4.0

Konfiguracja router-a odbywa się (tak jak dla większości innych produktów firmy GUPTA Technologies$_{TM}$) poprzez ustawienia w pliku *SQL.INI*. Dla zadeklarowania użycia router-a należy ustawić *comdll* = *sqlirtrw* w sekcji *winclient.dll*. W sekcji *infogtwy* można określić parametry funkcjonowania router-a:

- *remotedatabase* pozwala na podanie komendy połączenia z bazą danych zgodnie z syntaktyką języka SQL dla odpowiedniego typu serwera INFORMIX-a: OnLine (wykorzystywany w instalacji) lub SE-Standard Engine. Przykład dla serwera OnLine:
  remotedatabase = <*nazwa_bazy_1*>, *database* <*nazwa_bazy_2*>, gdzie:
  <nazwa_bazy_2> jest nazwą bazy znaną po stronie zdalnego systemu,
  <nazwa_bazy_1> jest nazwą używaną w aplikacjach klienckich dla MS Windows przy realizacji połączenia z serwem bazy danych poprzez router. (<nazwa_bazy_1> i <nazwa_bazy_2>, jeśli nie przekraczają ośmiu znaków, mogą być tożsame.)
- *substitute* pozwala na proste dopasowanie syntaktyczne dialektów języka SQL wykorzystywanego w aplikacji i implementowanego w serwerze bazy danych poprzez proste zastępowanie ciągów znakowych w łańcuchach, stanowiących wysyłane i odbierane komendy języka SQL. Przykładowo po wpisie *substitute* = *syssql.sys, gti* każdy ciąg znaków *syssql.sys* po stronie klienta zastępowany jest przez *gti* po stronie serwera.

- *maperror* pozwala na przełączanie się pomiędzy komunikatami o błędach w standardzie SQLBase albo w standardzie DBMS INFORMIX.

  Np. *maperror = on.*
- *longbuffer* umożliwia określenie maksymalnej długości (ograniczenie do podanej liczby początkowych bajtów) dla pól typu *CHAR*, *VARCHAR*, *BYTE*, *TEXT* z pobieranych wierszy.

  Np. *longbuffer = 1000.*
- *ferslong* pozwala na określenie, czy dane dłuższe niż 254 bajty powinny być przechowywane w zbiorze rozwiązań na stacji roboczej, przy pracy w trybie *Result Set.*

  Np. *ferslong = on.*

Przy współpracy aplikacji klienta z serwerem SQLBase na ogół wykorzystuje się tzw. tryb *Result Set*, w którym:

- zbiór rozwiązań zadania wyszukiwania, przechowywany w pamięci *cache* serwera, udostępniany jest aplikacji klienta w miarę tworzenia (tzn. możliwe jest pobieranie przez aplikację wierszy, nawet gdy serwer jest w trakcie wyszukiwania kolejnych, innych wierszy rozwiązania),
- umożliwione jest przesuwanie się w zbiorze wierszy w górę, w dół i na wiersz o dowolnym numerze.

Serwery INFORMIX-a nie posiadają takiego udogodnienia. W takiej sytuacji, przy współpracy aplikacji klienta z serwerem poprzez router, symulacja trybu *Result Set* polega na tworzeniu zbioru rozwiązań na stacji roboczej (ang. Frontend Result Set) w plikach tymczasowych o nazwach *FRS<n>*.

Z programu klienta (komunikującego się poprzez moduły pakietu SQLRouter for INFORMIX) możliwa jest zmiana poziomów izolacji, implementowanych w serwerze INFORMIX-OnLine. Przyporządkowanie nazw poziomów izolacji (pochodzących od nazewnictwa stosowanego w implementacji SQLBase), ustawianych w aplikacji utworzonej z wykorzystaniem produktów GUTPA SQL System (komunikujących się poprzez bibliotekę *sqlapiw.dll*) i poziomów izolacji DBMS INFORMIX-OnLine przedstawione jest w tabeli 1.

Tabela 1

Przyporządkowania symboli wykorzystywanych po stronie aplikacji klienta do poziomów izolacji implementowanych w DBMS INFORMIX-OnLine

SQLBase	INFORMIX-OnLine
(RO) Read Only	(DR) Dirty Read
(RL) Release Lock	(RL) Release Lock
(CS) Cursor Stability	(CS) Cursor Stability
(RR) Repeatable Read	(RR) Repeatable Read

Mylące może się wydawać przyporządkowanie symbolu RO (*Read Only* w standardzie SQLBase oznacza poziom izolacji określający możliwość dostępu do bazy danych tylko do odczytu, ale z zapewnieniem logicznej spójności danych, odczytywanych w różnych momentach czasowych) do poziomu izolacji *Dirty Read*, który w standardzie DBMS INFORMIX-OnLine oznacza umożliwienie odczytu z pominięciem systemu blokad, tzn. poza transakcyjnym przetwarzaniem (możliwy odczyt niespójnych logicznie danych, aktualnie modyfikowanych przez inne zadanie).

Na komputerze z systemem operacyjnym SCO Unix z uruchomionym programem serwera bazy danych INFORMIX-OnLine zainstalowane zostały wcześniej dodatkowo:

- pakiet TCP/IP dla SCO.Unix,
- pakiet *Informix Communication Software*: INFORMIX-STAR. for Unix (ewentualnie INFORMIX-NET. for Unix).

W prostej konfiguracji dla realizacji architektury klient-serwer po stronie serwera bazy danych wystarczy zainstalować pakiet INFORMIX-NET for Unix w celu udostępnienia usług serwera bazy danych dla innych host-ów. Pakiet INFORMIX-STAR oprócz realizacji wspomnianych funkcji komunikacyjnych (realizowanych przez INFORMIX-NET) pozwala na zestawienie instalacji rozproszonego systemu zarządzania bazą danych (serwer rozproszonej bazy danych) (ang. Distributed DBMS Server). W ramach przeprowadzanych testów skonfigurowano system instalując na dwóch komputerach klasy PC odpowiednio: SCO.TCP/IP, INFORMIX-OnLine. for Unix, INFORMIX-STAR. for Unix. Przy konfiguracjach takiego typu możliwa jest realizacja pojedynczego zapytania (tzn. wyrażonego pojedynczym zdaniem języka SQL), w którym zawarte tablice należą do różnych baz danych, obsługiwanych przez różne serwery typu INFORMIX-OnLine, na różnych komputerach.

Przykładowo:

```
SELECT t1.a1, t2.a1
   FROM baza1@host1:tablica1 t1, baza2@host2:tablica2 t2
      WHERE t1.a2 = t2.a2
```

gdzie:

host1, *host2* – nazwy komputerów z uruchomionymi programami serwera bazy danych,

baza1, *baza2* – nazwy lokalnych baz danych, obsługiwanych na odpowiednich komputerach,

tablica1, *tablica2* – nazwy tablic, należących do odpowiednich baz danych,

a1, *a2* – nazwy kolumn w tablicach.

Przezroczystość (w tym przypadku, rozumiana jako pozorne ukrycie faktycznej przynależności tablic do różnych baz danych, obsługiwanych przez różne serwery) można uzyskać definiując odpowiednie synonimy we wszystkich lokalnych bazach danych.

Instalacja z aplikacją klienta dla MS Windows (np. utworzoną za pomocą SQLWindows) i serwerem rozproszonej bazy danych (INFORMIX OnLine, INFORMIX-STAR) funkcjonuje poprawnie tzn. SQLRouter for INFORMIX uwzględnia możliwość przesyłania zdań języka SQL, rozszerzonego o składnię wynikającą z obsługi systemu zarządzania rozproszoną bazą danych.

W ramach przeprowadzonych testów skofingurowano instalację środowiska aplikacji klienta z nowszą wersją pakietu INFORMIX-NET PC o numerze 5.01. Pakiet ten eksploatowany może być tylko w środowisku MS Windows. (W tej nowej wersji zrezygnowano z DOS-owego programu rezydentnego *remsql.exe*, wykorzystywanego w wersji 4.10.).

Komunikacja z wykorzystaniem INFORMIX-NET PC 5.01 odbywa się według schematu:

– w wersji dla protokołu TCP/IP:
  aplikacja klienta -> isqlinet.dll-> winetman.dll -> winsock.dll(v1.1)...
  -> (i dalsze moduły do obsługi TCP/IP, pochodzące od dowolnego producenta)
– w wersji dla protokołu IPX:
  aplikacja klienta -> isqlinet.dll -> inetipx.dll -> inetipx.exe...

W zrealizowanej instalacji, w której komunikacja z serwerem bazy danych INFORMIX-OnLine odbywa się poprzez protokół TCP/IP, wykorzystano pakiet PC/TCP firmy FTP Software.

Dla zachowania kompatybilności z wersją 4.10 pakiet INFORMIX-NET PC 5.01 zawiera nową wersję biblioteki *ldllsqlw.dll*. Aplikacje przystosowane do współpracy z wersją 4.10, tzn. poprzednio odwołujące się poprzez funkcje z *ldllsqlw.dll* do usług programu rezydentnego *remsql.exe*, w obecnej wersji tej biblioteki odwołują się pośrednio do funkcji z *isqlinet.dll* (i dalej według nowego scenariusza schematu odwołań, przedstawionego powyżej).

Ustawienia typu *host* - nazwa lub adres IP komputera z serwerem bazy danych, *user, passwd* - nazwa użytkownika, hasło, *service* - nazwa usługi, *protocol* - rodzaj protokołu przechowywane są w pliku kofiguracyjnym *informix.ini*.

Nowa wersja INFORMIX-NET PC (5.01) między innymi umożliwia:

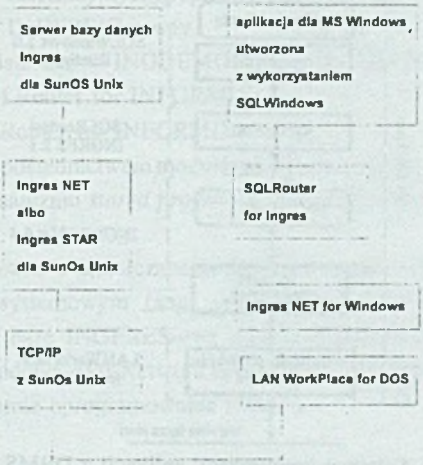
- realizację równoczesnych połączeń z bazą danych dla kilku uruchomionych aplikacji MS Windows (wielowejściowa biblioteka komunikacyjna),
- opcjonalne, dynamiczne ustawianie nazwy użytkownika i hasła na etapie nawiązywania połączenia,
- dodatkowego oddawania sterowania do systemu Windows (ang. yielding) w celu obsługi komunikatów kierowanych do innych, jednocześnie uruchomionych aplikacji.

W skład pakietu wchodzą również proste programy narzędziowe:

- *setnet.exe* do ustalenia parametrów połączenia (typu *host*, *user*...),
- *ilogin.exe* prosta aplikacja klienta do testowania poprawności połączenia na poziomie pakietu INFORMIX-NET PC,
- *finderr.exe* do odczytu komunikatów o błędach na podstawie ich kodów.

4.1.2. Komunikacja z serwerem DBMS Ingres

W celach testowych wykonano instalację, w ramach której aplikacja klienta utworzona w SQLWindows łączyła się za pośrednictwem router-a z serwerem bazy danych INGRES, uruchomionym na komputerze rodziny Sun (rys. 4).



Rys. 4. Schemat ogólny połączenia aplikacji klienta z DBMS Ingres
Fig. 4. The general schema of connnection of a client application to DBMS Ingres

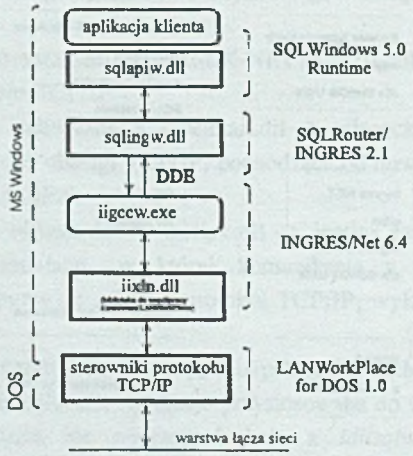
Na stacji roboczej zainstalowana została część pakietu LANWorkPlace for DOS 1.0, zapewniająca komunikację aplikacji MS Windows poprzez protokół TCP/IP.

Dostęp do usług serwera bazy danych z poziomu aplikacji MS Windows umożliwiony został dzięki instalacji pakietu Ingres/Net for Windows 6.4. Zasadniczą część pakietu stanowi program *iigccw.exe*, korzystający z odpowiednich bibliotek komunikacyjnych opisanych w pliku konfiguracyjnym *protocol.net* (gdzie odpowiednie wpisy definiują nazwy bibliotek DLL takich jak np: *iixln.dll* dla plakietu LANWorkplace for DOS firmy Novel, *iiftp.dll* dla pakietu PCTCP firmy FTP Software czy *iinfs.dll* dla pakietu SunPC-NFS firmy SunSoft itd.).

Konfiguracja stacji roboczej jako klienta systemu z serwerem bazy danych INGRES polega między innymi na zdefiniowaniu ustawień dotyczących tzw. węzła wirtualnego (ang. virtual node). Program *netu* pozwala na określenie parametrów:

- *virtual node* - nazwa węzła (np. my_node),
- *remote node* - adres IP komputera z uruchomionym serwerem INGRES-a,
- *protocol* - nazwa "protokołu", wynikająca z wykorzystywanego pakietu komunikacyjnego (np. LAN-WORKPLACE),
- *listen address* - identyfikator portu komunikacyjnego (ustalanego po stronie serwera bazy danych na etapie jego instalacji) (np. ps),
- nazwy użytkownika, znanej po stronie zdalnego systemu,
- hasła,

zawartych w plikach *node.net* i *login.net* przechowywanych na stacji roboczej.



Rys. 5. Schemat komunikacji aplikacji z DBMS Ingres, poprzez moduły SQLRouter i Ingres/Net 6.4

Fig. 5. The communication schema of application connected to DBMS Ingres using modules: SQLRouter and Ingres/Net 6.4

W celu umożliwienia komunikacji aplikacji utworzonej w SQLWindows z serwerem bazy danych INGRES na stacji roboczej zainstalowano pakiet SQLRouter/INGRES. Zasadniczą częścią pakietu jest biblioteka *sqlingw.dll*, za pośrednictwem której aplikacja z poziomu modułu *sqlapiw.dll* komunikuje się z programem *iigccw.exe* (z pakietu INGRES/Net) (rys. 5.). Dla zadeklarowania użycia router-a należy ustawić *comdll* = *sqlingw* w sekcji *winclient.dll*. W dodatkowej sekcji *inggwty* można określić parametry funkcjonowania router-a, gdzie:

- *remotedatabase* pozwala na skojarzenie nazwy bazy danych, wykorzystywanej po stronie aplikacji z parametrami określającymi połączenie z serwerem INGERS-a:
  remotedatabase = *<nazwa_bazy_1>*, *<vnode>*::*<nazwa_bazy_2>*/*<typ_serwera>*,
  gdzie:
  - <nazwa_bazy_1> jest nazwą używaną w aplikacjach dla MS Windows, przy realizacji połączenia z serwerem bazy danych poprzez router,
  - <vnode> to identyfikator połączenia ustawiany dla danej stacji roboczej poprzez program *netu*,
  - <nazwa_bazy_2> jest nazwą bazy znaną po stronie zdalnego systemu,
  - <typ_serwera> określa rodzaj zdalnego serwera:
    - *ingres* – standardowy typ serwera,
    - *star* – instalacja INGRES/Star (dostęp do rozproszonego systemu zarządzania bazą danych),
    - *gateway* – dostęp do innych, nieINGERS-owych baz danych (nawet nierelacyjnych) poprzez SQL, za pośrednictwem oprogramowania INGERS/Gateway.

  Przykładowo: remotedatabase = INGDEMO, my_node::DEMO/ingres,
- *substititute* (jak w SQLRouter for INFORMIX),
- *maperror* (jak w SQLRouter for INFORMIX).

Z poziomu aplikacji za pośrednictwem modułów SqlRouter/Ingres, INGRES/Net możliwe jest wykorzystywanie mechanizmu *stored procedure* zaimplementowanych w serwerze bazy danych.

Wraz z pakietem SqlRouter-a dostarczane są skrypty tworzące odpowiednie perspektywy odpowiadające tablicom systemowym (ang. system katalog) w standardzie SQLBase, upodobniając tym samym bazę INGERES-ową do baz SQLBase-owych. W najnowszych wersjach pakietów SQLWindows, SQLNetwork dopasowanie tablic systemowych realizowane jest dynamicznie, na poziomie oprogramowania router-a.

4.2. Opisy przykładowych, testowych instalacji, wykorzystujących mechanizmy ODBC

W wersji SQLWindows 5.0 zaimplementowano moduły pozwalające na komunikację utworzonych aplikacji klienta dla MS Windows z różnymi DBMS poprzez mechanizmy ODBC (ang. Open DataBase Conectivity).

W dużym uproszczeniu, standard ODBC (dotychczas popierany przez firmę Micorsoft) zawiera specyfikacje protokołu i funkcji, mających na celu SQL-ową unifikację metod dostępu do danych (niekoniecznie SQL-owych baz danych). Standard jest implementowany w zasadzie tylko dla środowiska MS Windows. Dwupoziomowa struktura "16-bitowego" ODBC obejmuje uniwersalny sterownik (ang. driver) ODBC (*odbc.dll*) i specjalizowany sterownik, charakterystyczny dla określonego typu bazy danych, pochodzący na ogół od niezależnego producenta. Konfiguracja modułu ODBC odbywa się poprzez program panelu sterowania (ang. Control Panel) (dokładniej, programu *odbcadm.exe* zainstalowanego w panelu sterowania) i polega na odwoływaniu się do funkcji z biblioteki *odbcinst.dll*, na przykład w celu:

- edycji listy sterowników specjalizowanych poprzez zamianę ustawień w plikach *odbcinst.ini* dotyczących między innymi:
  - nazwy biblioteki DLL, będącej sterownikiem specjalizowanym (wpis *driver*),
  - nazwy biblioteki DLL, umożliwiającej zamianę ustawienień parametrów funkcjonowania tegoż sterownika specjalizowanego (wpis *setup*),
- definicji źródeł danych (ang. data sources) i ich parametrów (umieszczanych w pliku konfiguracyjnym *odbc.ini*). Źródła danych odpowiadają nazwom baz danych na poziomie aplikacji, komunikujących się za pośrednictwem ODBC. (Opis każdego źródła danych zawarty w oddzielnej sekcji pliku *odbc.ini*).

Od strony aplikacji klienta, tworzonych z wykorzystaniem produktów firmy GURTA Technologies$_{TM}$, komunikacja poprzez ODBC obywa się poprzez bibliotekę *sqlodbw.dll*. Nazwę tej biblioteki komunikacyjnej należy wpisać do omawianej wcześniej sekcji *winclient.dll* pliku kofiguracyjnego *SQL.INI*. Komunikacja przebiega według schematu:
aplikacja klienta -> sqlapiw.dll -> sqlodbw.dll -> odbc.dll -> sterownik specjalizowany...

W ramach testów wykonano instalacje, w których aplikacja mogła komunikować się z serwerem DBMS INFORMIX-OnLine, DBMS Ingres, zbiorem plików DBF.

W przedstawianych instalacjach uniwersalne moduły ODBC (*odbc.dll, odbcadm.exe, odbcinst.dll*...) pochodzą z firmy Microsoft.

4.2.1. Komunikacja z serwerem DBMS Ingres

W zrealizowanych, testowych instalacjach sterownik ODBC przeznaczony do współpracy z DBMS Ingres pochodzi od producenta serwera bazy danych. Rolę specjalizowanego sterownika ODBC pełni biblioteka *sqling.dll*. Funkcje z biblioteki *sqlingstp.dll* (wywoływane z poziomu *odbcinst.dll*) pozwalają na zmianę ustawień parametrów, wykorzystywanych przez *sqling.dll* (zmiana wpisów w sekcji *ingodbc* w pliku *odbc.ini*). Przykładowo zmiana może dotyczyć opisu źrodła danych, typu serwera bazy danych INGRES, nazwy bazy danych, nazwy węzła wirtualnego itp. (w większości omawianych wcześniej przy okazji przedstawiania parametrów SQLRouter-a for INGRES). Podobnie do konfiguracji z routerem, sterownik ODBC współpracuje z modułami pakietu INGRES/Net PC. Uproszczony schemat komunikacji zastosowany w testowej instalacji:

aplikacja klienta -> sqlapiw.dll -> sqlodbw-> odbc.dll ->
-> sqling.dll DDE> iigccw.exe -> iixln.dll ->
-> (moduły pakietu LAN WorkPlace do obsługi protokołu TCP/IP)...

4.2.2. Komunikacja z serwerem DBMS INFORMIX

W ramach testów wykonano instalację, w której aplikacja klienta (utworzona z wykorzystaniem produktów frirmy GUPTA Technologies$_{TM}$) komunikowała się z DBMS Informix poprzez mechanizmy ODBC. Zastosowany pakiet ODBC Drivers pochodził od firmy INTERSOLV,Inc. Rolę specjalizowanego sterownika ODBC i jednocześnie modułu z funkcjami realizującymi zmianę parametrów funkcjonowania tegoż sterownika lub parametrów odpowiednich źródeł danych pełni biblioteka *qeinf506.dll*. Sterownik ODBC współpracuje bezpośrednio z modułami pakietu INFORMIX-NET PC 5.01. Uproszczony schemat komunikacji zastosowany w przykładowej instalacji:

aplikacja klienta -> sqlapiw.dll -> sqlodbw -> odbc.dll ->
-> qeinf506.dll -> isqlnet.dll -> ... (jak dla SqlRouter for Informix)...

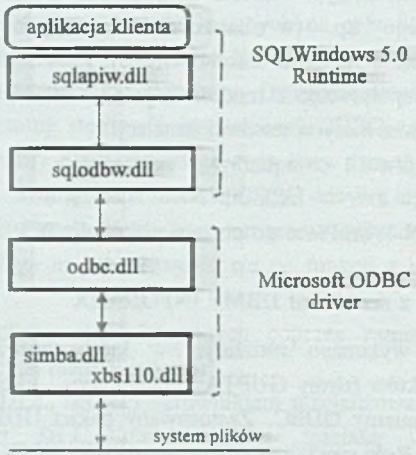
4.2.3. Wykorzystanie baz danych w formacie DBF

W ramach testów zestawiono instalację, w której aplikacja komunikowała się z bazą danych w standardzie XBASE, odpowiadającą zbiorowi plików danych formatu DBF i plików indeksowych (pochodzących z systemów dBaseIII/IV lub FoxPro 2.0/2.5). W rozwiązaniu zastosowano moduły z pakietu pochodzącego od firmy Microsoft, w skład którego wchodzą między innymi:

simba.dll – specjalizowany sterownik ODBC,
simadmin.dll – biblioteka odpowiedzialna za realizację zmian parametrów działania sterownika i źródeł danych,

xbs110.dll – biblioteka umożliwiająca dostęp do danych poprzez indeksową metodę zgodną ze standardem CISAM - (parametry w pliku *odbcisam.ini*).

Schemat komunikacji aplikacji, tworzonej z wykorzystaniem narzędzi firmy GUPTA Technologies$_{TM}$ z bazami standardu XBASE przedstawiony jest na rysunku 6.



Rys. 6. Schemat komunikacji aplikacji wykorzystującej mechnizmy ODBC do współpracy z bazą danych klasy XBASE

Fig. 6. The communication schema of application connected to XBASE databases using ODBC mechanisms

Sterowniki ODBC, umożliwiające dostęp do plików XBASE należą do klasy sterowników jednowarstwowych (ang. single tier driver), tzn. nie stanowią tylko wstępnej, pierwszej pośredniczącej (unifikacyjnej) warstwy komunikacyjnej (tak jak w przypadku sterownika ODBC INFORMIX-a czy INGRES-a), ale spełniają również funkcje prostego systemu zarządzania plikami bazy danych.

5. Podsumowanie

Celem artykułu jest prezentacja nowych, już komercyjnie stosowanych technologii i narzędzi programowych służących do tworzenia wydajnych i skalowalnych systemów obsługi baz danych, tworzonych w oparciu o architekturę klient-serwer.

W pierwszej części artykułu dokonana została charakterystyka produktów programowych służących do tworzenia aplikacji, administrowania bazą danych czy w końcu samej obsługi bazy danych (charakterystyka programu serwera bazy danych) na przykładzie nowoczesnego systemu firmy GUPTA Technologies$_{TM}$.

Druga część zawiera opisy wykonanych, testowych instalacji. Przedstawione zostały wybrane konfiguracje systemów o architekturze klient-serwer w heterogenicznych sieciach komputerowych z uwzględnieniem możliwości łączenia ze sobą produktów programowych różnych producentów. Przykładowe opisy instalacji miały na celu między innymi ukazanie praktycznych sposobów realizacji skalowalności systemu (dzięki wykorzystaniu niezależności komponentów, tzn. aplikacji kilenta i programu serwera bazy danych) poprzez wykorzystanie pewnej swobody w doborze programu serwera bazy danych.

Artykuł przeznaczony jest dla osób zainteresowanych zagadnieniami implementacji metod integracji systemów zarządzania bazami danych.

LITERATURA

[1] Gietz W.: SQLWindows Basic Programming Techniques and Strategies.
[2] SQLWindows Technical Reference Manual.
[3] SQLTalk.
[4] SQLBase SQLTalk.
[5] SQLTalk/Windows.
[6] Quest Formula Editor and Function Reference Manual.
[7] Quest Advanced User's Guide.
[8] Team Windows Users's Guide.
[9] ReportWindows Reference Manual.
[10] SQLBase C Application Programming Interface.
[11] SQLRouter/INFORMIX Installation and Users Manual.
[12] SQLRouter/Ingres Installation and Users Manual.
[13] INFORMIX-NET PC Installation and Configuration Guide.
[14] INFORMIX-NET INFORMIX-STAR User Guide.
[15] Ingres Tools for Windows.

[16] LAN WorkPlace for DOS Administrator's Guide.
[17] LAN WorkPlace for DOS User's Guide.
[18] INTERSOLV DataDirect ODBC Drivers.

Recenzent: Dr hab. inż. Stanisław Wołek

Wpłynęło do Redakcji 4 grudnia 1995 r.

Abstract

The article presents the GUPTA SQL System as an example of Database Management System with a client-server architecture. In first part of the article new, sophisticated products of GUPTA SQL System are shown. Development tools for creating front-end-s and SQLBase - a database server and his heterogeneous implementations are described in details. Second part of this article focuses on methods of integration a client application created using GUPTA SQL System products with efficient database servers of other software producers. Configurations of a client workstation for making connections to DBMS Informix or DBMS INGRES are described in details.