ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z. 33 Nr kol. 1377

Dorota PIERZCHAŁA
Politechnika Śląska, Instytut Informatyki

# INTEGRACJA BAZ DANYCH W HETEROGENICZNYCH ŚRODOWISKACH SIECI KOMPUTEROWYCH NA PRZYKŁADZIE SERWERÓW MS SQL Server I GUPTA SQLBase[1]

**Streszczenie.** W artykule zaprezentowano możliwość integracji serwerów baz danych GUPTA SQLBase i MS SQL Server opartej na oprogramowaniu routera GUPTA SQLNetwork. Wskazano sposoby przeniesienia bazy danych z systemu SQLBase do SQL Servera i przystosowania gotowej aplikacji napisanej w SQLWindows do współpracy z nowym serwerem. Porównano czasy dostępu z jednej aplikacji do danych na integrowanych serwerach. Omówiono wyniki przeprowadzonych eksperymentów.

# THE INTEGRATION OF DATABASES IN HETEROGENOUS COMPUTER NETWORK'S ENVIRONMENTS BY EXAMPLE OF MS SQL Server AND GUPTA SQLBase SERVERS

**Summary.** Possibility of databases servers' GUPTA SQLBase and MS SQL Server integration using GUPTA SQLNetwork router programme is presented in this paper. Methods of database's transfer from SQLBase to SQL Server and adaptation of working SQLWindows' application to collaboration with the new server are described. results of experiments are discussed.

[1]Opracowanie wykonane częściowo w ramach grantu KBN 8 T11C 020 11.

## 1. Warunki sprzętowo-programowe integracji baz danych w heterogenicznych środowiskach sieci komputerowych

Poprzez integrację baz danych w niniejszym artykule rozumie się możliwość jednoczesnego korzystania z informacji zgromadzonych na integrowanych serwerach baz danych przez określoną aplikację.

Zakłada się, że serwery baz danych, pochodzące od różnych producentów, mogą być zainstalowane na komputerach o różnej architekturze, pracujących pod kontrolą odmiennych systemów operacyjnych. Komputery, na których zainstalowane są serwery baz danych, połączone są heterogeniczną siecią komputerową.

Podstawowym etapem integracji jest umożliwienie dostępu do różnych baz danych z poziomu tej samej aplikacji, co wymaga najczęściej zastosowania oprogramowania pośredniczącego w komunikacji między aplikacją a serwerem baz danych. Istotną rzeczą jest określenie wpływu zastosowania oprogramowania pośredniczącego na badaną aplikację - najczęściej wymagane są pewne zmiany w kodzie aplikacji, uwzględniające odmienność funkcjonalną integrowanych serwerów.

Prócz wpływu na funkcjonalność aplikacji, zastosowanie oprogramowania pośredniczącego w przesyłach informacji między klientem a serwerami najczęściej ma wpływ na czas dostępu do bazy danych z poziomu aplikacji klienta i jest zależne od rodzaju wykonywanych operacji i rozmiaru przesyłanych danych.

Ponieważ z integracją baz danych wiąże się często konieczność przeniesienia danych z jednego serwera na inny, należy wskazać związane z tym różnice między serwerami i sposoby migracji struktury i zawartości baz danych.

## 2. Cel i zakres badań

Celem badań była integracja systemów baz danych GUPTA SQLBase, zainstalowanego na serwerze Novell NetWare, i MS SQL Server, działającego pod kontrolą serwera MS Windows NT, w zakresie analizy współpracy narzędzi wchodzących w skład pakietu GUPTA SQLSystem z serwerem bazy danych MS SQL Server, ze szczególnym uwzględnieniem aplikacji stworzonych przez użytkownika za pomocą SQL Windows. Integracja badanych systemów oparta jest na pakiecie GUPTA SQLNetwork realizującym dostęp aplikacji GUPTA SQLSystem do różnych systemów baz danych, w szczególności do MS SQL Servera. Badaniom poddano możliwość przeniesienia wypełnionych baz danych z serwera SQLBase do środowiska MS SQL Server, określono konsekwencje techniczne, funkcjonalne wykorzystania serwera MS SQL Server z aplikacji SQLWindows zamiast serwera SQLBase firmy

GUPTA, przeprowadzono również porównanie czasów dostępu do serwerów z aplikacji klienta wykonującej typowe operacje. Opisywane badania są kontynuacją oraz uzupełnieniem wcześniej przeprowadzonych i opublikowanych eksperymentów dotyczących serwerów Ingres [6] i Informix [5].

### 2.1. Plan przeprowadzonych badań

Pierwszym etapem badań nad integracją baz danych było udostępnienie danych zgromadzonych w MS SQL Serverze aplikacji w SQLWindows za pomocą oprogramowania SQLNetwork. Rozdział 3 niniejszego artykułu zawiera szczegółowy opis konfiguracji oprogramowania przeprowadzonej w tym celu.

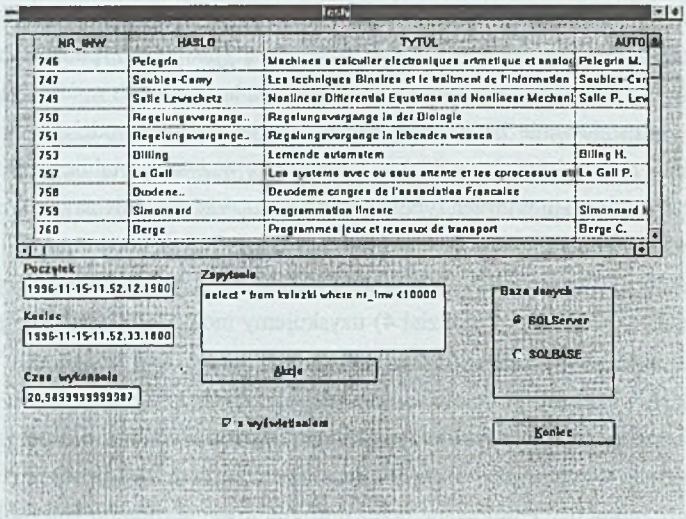
Integracja w postaci dostosowywania oprogramowania napisanego przy pomocy GUPTA SQLWindows do nowego serwera może wiązać się (jak w przypadku opisywanych tu eksperymentów) z potrzebą przeniesienia istniejącej już bazy danych w formacie SQLBase na SQL Server. Zainstalowanie bazy na serwerze należy przy tym podzielić na dwa odrębne etapy: przeniesienie struktury bazy danych (stworzenie wszystkich tablic bazy z uwzględnieniem potrzeby konwersji niektórych typów danych) oraz przeniesienie zawartości bazy danych (informacji w niej zgromadzonych).

Po przygotowaniu oprogramowania routera (rozdział 3) i przeniesieniu bazy danych wraz z zawartością do SQL Servera (rozdział 4) uzyskujemy możliwość łączenia się i korzystania z owej bazy danych za pomocą narzędzi GUPTA SQLSystem (takich jak SQLTalk, a przede wszystkim SQLWindows) w sposób niemalże identyczny z przypadkiem bazy danych SQLBase, dla której narzędzia te pierwotnie zostały stworzone. Oczywiście, ze względu na różnice w realizacji wielu funkcji przez oba serwery, niektóre z funkcji,aby zachować większą przezroczystość,mogą być emulowane przez oprogramowanie routera. Dla funkcji mogących w znacznym stopniu pogorszyć efektywność w dostępie do serwera emulacja ta jest opcjonalna, a informacja o aktualnych ustawieniach zawarta jest w odpowiedniej sekcji pliku konfiguracyjnego *sql.ini*. Natomiast realizacja niektórych bardziej zaawansowanych funkcji pozostaje odmienna od wersji dla serwera SQLBase. Szczegóły na ten temat zawiera rozdział 5.

Istotnym aspektem integracji są jej konsekwencje w postaci zmiany czasu dostępu do danych zgromadzonych w bazach danych, dlatego przprowadzono szereg eksperymentów pozwalających na porównanie czasu trwania typowych operacji dostępu do bazy danych z i bez oprogramowania pośredniczącego. Program, za pomocą którego dokonywano pomiaru szybkości dostępu do danych w bazie, realizował zapytanie w języku SQL wprowadzone przez użytkownika. Realizacja kończy się wyświetleniem całego zbioru wynikowego w specjalnej tabeli do przezentacji danych za pomocą tej samej standardowej funkcji (na wykresach istnieje

wówczas komentarz "z wyświetlaniem") albo pobraniem tylko pierwszego wiersza ze zbioru wynikowego do zmiennej programowej (komentarz: "bez wyświetlania"). Zmianę serwera bazy danych umożliwiało umieszczone na ekranie pole opcji, aby zapytania były realizowane na zmianę raz w jednej bazie danych, raz w drugiej w trakcie tego samego testu, w celu wykluczenia możliwości wpływu na wynik niejednakowego obciążenia sieci w czasie eksperymentu. Wygląd okna programu przedstawia rys. 1.

Na jednym i drugim serwerze została zainstalowana taka sama baza danych, zawierająca dane biblioteczne zgromadzone w 21 tablicach. Testy dotyczyły dwóch z tych tablic - tablicy "Książki", zawierającej 10 000 rekordów po 18 pól, dających razem 674 bajty na rekord oraz (w teście łączenia) związanej z nią przez identyfikator (pole numeryczne *id_wyd*) tablicy "Wydawcy", zawierającej 90 rekordów po 3 pola, dające 154 bajty na rekord.




Rys. 1. Wygląd okienka programu testującego
Fig. 1. Testing programme window's look

### *2.1.1. Rodzaje testów*

Test szybkości składał się z następujących zadań:

- **Selekcja** - zapytanie obejmujące zakres jednej tablicy, a wynik stanowi 10 procent jej zawartości. Test ten składał się z dwóch typów części:
  - **numeryczna** - selekcja następuje według pola poindeksowanego, posiadającego wartości unikalne, a kryterium poszukiwania obejmuje jeden ciągły zakres liczbowy (wyniki przedstawiono na rys. 3),

• **alfanumeryczna** - odnalezienie rekordów, których zawartość w danym polu dokładnie odpowiada zadanemu wzorcowi; istnieje indeks założony na tym polu, lecz wartości w tym polu nie są unikalne (wyniki przedstawiono na rys. 4),

• **Projekcja** - w wyniku uzyskujemy wartości występujące w jednej kolumnie podanej tablicy, obejmujące cały zakres tej tablicy; wyświetlana kolumna jest poindeksowana i zawiera wartości unikalne (wyniki przedstawiono na rys. 5),

• **Operacje agregujące** - sprawdzenie szybkości obliczania wartości minimalnej, maksymalnej, sumy, średniej oraz liczby wystąpień w tej samej kolumnie, która była wynikiem projekcji, a zakres obejmuje całą tablicę (wyniki przedstawiono na rys. 7 i rys. 8) lub jej 10% - warunek jak w selekcji alfanumerycznej (rys. 9 i rys. 10),

• **Łączenie** - przedmiotem testu jest połączenie tablic "Książki" i "Wydawcy" (typowe połączenie jeden do wielu), którego wynik stanowi 100 rekordów (wyniki na rys. 6),

• **Aktualizacja** - test ten został podzielony na dwie części:

  • **modyfikacja**, obejmująca zmianę wartości klucza głównego dziesięciu procent rekordów z tablicy, co zmusza serwer do jedoczesnej aktualizacji indeksów (wyniki przedstawiono na rys. 11, instrukcja "update"),

  • **usunięcie** 10% rekordów ze środka tablicy (wyniki przedstawiono również na rys. 12, instrukcja "delete").

Operacje aktualizacji zostały przetestowane z zatwierdzeniem i z wycofaniem transakcji. Wyniki przeprowadzonych testów i ich komentarz zawiera rozdział 6.

## 2.2. Charakterystyka integrowanych środowisk sprzętowo-programowych

W badaniach integrowano następujące konfiguracje serwerów baz danych:

• MS SQL Server 6.5 zainstalowany na komputerze Optimus Pentium 120 MHz wyposażonym w 32 MB pamięci operacyjnej pracującym pod kontrolą systemu MS Windows NT Server 3.51,

• GUPTA SQLBase 6.0 for NetWare 4.x, zainstalowany na komputerze Optimus 486 DX2/66 MHz, 32 MB RAM będącym serwerem sieci lokalnej Novell NetWare 4.02.

Badania dostępu do prezentowanych serwerów były przeprowadzane ze stacji klienta - komputera Optimus Pentium 75 MHz z 16 MB RAM. Wszystkie trzy komputery biorące udział w testach wyposażone były w magistralę PCI i karty sieciowe 3 COM Etherlink III. Na komputerze klienta w celu przeprowadzenia badań zainstalowane było następujące oprogramowanie:

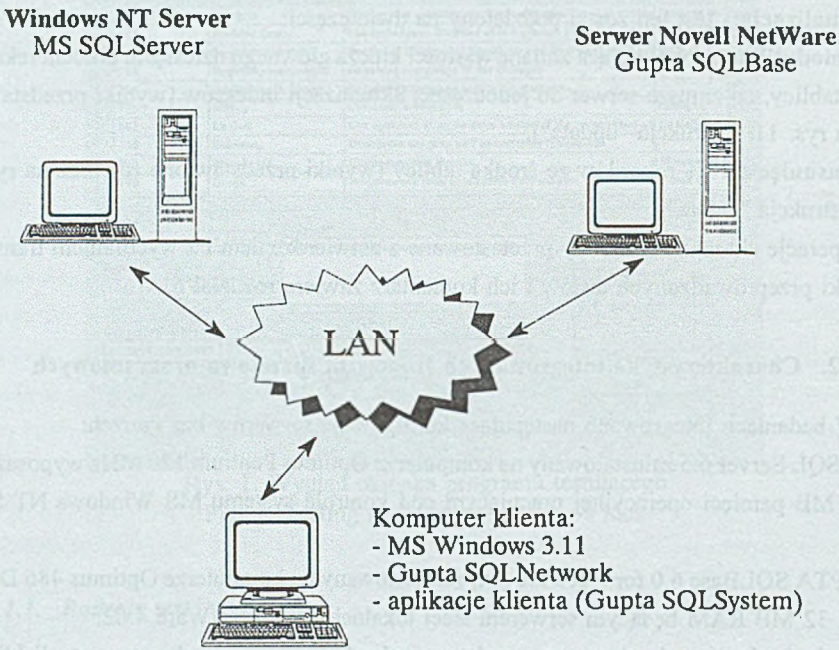
• MS Windows for Workgroups 3.11,

• Aplikacje wchodzące w skład pakietu GUPTA SQLSystem:

• SQLTalk - narzędzie do interaktywnej pracy z bazą danych, realizujące między innymi dostęp do danych za pośrednictwem języka SQL; użyto wersji 6.0 tego systemu dostarczanej wraz z serwerem SQLBase w wersji 6.0,

• WinTalk - narzędzie o podobnym zastosowaniu do wyżej wymienionego, wzbogacone jednak o możliwości wykonywania części czynności administracyjnych z systemu menu, m.in. importu i eksportu danych między określonymi formatami; wykorzystano wersję sytemu 3.2.0, wchodzącą w skład pakietu SQLWindows 5.0,

• SQLWindows - system przeznaczony do tworzenia aplikacji pracujących w środowisku MSWindows, korzystających z SQL-owej bazy danych, bazujący na języku czwartej generacji, programowaniu zorientowanym obiektowo, sterowanym zdarzeniami; w eksperymencie wykorzystano SQLWindows 5.0,




Rys. 2. Ogólna struktura platformy badań
Fig. 2. General structure of researches' platform

• SQLNetwork (SQLRouter for MS SQL Server) - oprogramowanie realizujące dostęp do baz danych MS SQL Servera z wymienionych wyżej aplikacji wchodzących w skład GUPTA SQLSystem; zbadano działanie wersji 5.0 tego programu,

• Oprogramowanie DB Library i Net-Library, konieczne do komunikacji z bazą danych MS SQL Server, prócz oprogramowania routera,

• Program bcp (bulk copy) umożliwiający ładowanie danych z odpowiednio przygotowanych plików tekstowych do SQL Servera. Ponieważ jest to program pracujący pod kontrolą systemu operacyjnego DOS, wymaga dodatkowych programów rezydentnych zapewniających dostęp do serwera bazy danych (w zależności od protokołu: ddmsspx, dbmsvine lub dbnmpipe).

Zakres przeprowadzonych czynności obejmował zainstalowanie i konfigurację wymienionych programów.

Ogólną strukturę platformy badań przedstawia rys 2.

## 3. Realizacja dostępu aplikacji klienta do serwera baz danych MS SQL Server

Podstawowym warunkiem udostępnienia baz danych, zainstalowanych na SQL Serverze, dla aplikacji wchodzących w skład GUPTA SQLSystem jest poprawna instalacja oraz konfiguracja oprogramowania zapewniającego komunikację między SQL Serverem a stacją klienta z poziomu specjalnie do tego przeznaczonych aplikacji-klientów SQL Servera. Należy zapewnić działanie sieciowego interfejsu komunikacyjnego i oprogramowania DB-Library, których opis wykracza poza temat niniejszego opracowania. Kolejnym etapem jest zainstalowanie komponentu pakietu SQLNetwork - routera do bazy danych SQL Server (biblioteki *sqlsqsw.dll*). Jednak połączenie z wybranymi, zdalnymi bazami danych SQL Servera możliwe jest pod warunkiem uzupełnienia pliku konfiguracyjnego, z którego korzysta oprogramowanie GUPTA SQLSystem, o nazwie *sql.ini*. Po pierwsze należy w nim wprowadzić informację na temat oprogramowania komunikacyjnego pomiędzy aplikacją klienta w środowisku Windows a serwerem baz danych. Informacje te zawiera sekcja *[winclent.dll]* pliku *sql.ini*, która powinna zostać uzupełniona o linię:

*comdll=sqlsqsw*

Wówczas pozostałe informacje dotyczące komunikacji z SQL Serverem poszukiwane będą w sekcji *[sqsgtwy]* pliku *sql.ini*. W tej sekcji należy przede wszystkim zawrzeć informacje o każdej bazie danych SQL Servera, którą chcemy udostępnić narzędziom GUPTY, w postaci linii o składni:

remotedbname=<db_name>, <server>, <db_name>

gdzie:

• pierwszy parametr, <db_name>, określa nazwę bazy danych, pod jaką będzie ona widoczna dla aplikacji klienta (nazwa ta nie może być dłuższa niż 8 znaków),
• <server> jest nazwą SQL Servera, na którym baza jest zainstalowana,
• trzeci parametr, <db_name>, powinien zawierać nazwę bazy danych, jaka jest znana dla SQL Servera (specyfikując trzeci parametr należy pamiętać o tym, że w przypadku rozróżniania przez SQL Server małych i wielkich liter, nazwa ta powinna być wprowadzona dokładnie tak, jak w SQL Serverze; wystąpienie tego parametru jest obowiązkowe tylko wówczas, gdy pierwszy z parametrów <db_name> jest od niego różny).

Poniżej zamieszczono przykładowy wiersz pliku *sql.ini*, umożliwiający dostęp klienta do bazy danych na serwerze o nazwie *ZTINT*. Baza ta znana jest dla SQL Servera pod nazwą *bibl*, natomiast dla oprogramowania klienta widoczna będzie pod nazwą *biblms*:
remotedbname=biblms, ZTINT, bibl .

W sekcji *[sqsgtwy]* pliku sql.ini można zadeklarować zbiór parametrów i opcji istotnych dla komunikacji klienta z serwerem. Szczegółowy opis ich składni zawiera dokumentacja [1], natomiast znaczenie i zastosowanie ważniejszych z nich zawierać będą następne rozdziały niniejszego opracowania.

## 4. Migracja bazy danych z serwera GUPTA SQLBase do MS SQL Servera

### 4.1. Przeniesienie struktury bazy danych

Pierwszym etapem powinno być zainstalowanie odpowiedniej bazy danych na SQL Serverze (stworzeniu katalogów systemowych) za pomocą narzędzi tego systemu (SQL Enterprise Manager), następnie można przystąpić do tworzenia wszystkich tablic użytkownika. Również przy tworzeniu obiektów użytkownika w bazie danych najlepiej jest posłużyć się przeznaczonymi do tego celu narzędziami firmy Microsoft. Struktura bazy danych nie wymaga wielu przekształceń, należy dokonać konwersji niektórych typów danych (patrz [1]) oraz uwzględnić odmienną składnię bardziej zaawansowanych poleceń. Pisanie skryptów tworzących przenoszoną strukturę bazy danych od podstaw, w dialekcie języka SQL charakterystycznym dla SQL Servera prócz dużej pracochłonności niesie w sobie również prawdopodobieństwo niepoprawnego (niedokładnego) przeniesienia istniejących struktur. Dlatego strukturę istniejącej bazy danych na serwerze SQLBase, wygodniej jest wyładować do pliku **.sql* za pomocą programu SQLTalk. Plik ten będzie zawierał instrukcje w języku

SQL, służące do założenia takiej samej bazy (na serwerze SQLBase). Dostosowanie tego skryptu do wymagań SQL Servera polega przede wszystkim na:

- odpowiedniej konwersji typów danych (m. in. typy *timestamp* należy zastąpić typem *datetime* lub *small datetime*),
- usunięciu w nazwach tablic i indeksów przedrostków oznaczających właściciela,
- usunięciu frazy *PCTFREE*, charakterysycznej dla SQLBase,
- usunięciu instrukcji *alter table*, dotyczących mechanizmów kontroli integralności referencyjnej, które lepiej założyć za pomocą odpowiednich funkcji do zarządzania bazą danych programu SQL Enterprise Manager,
- usunięciu instrukcji tworzenia synonimów, których celem jest tylko pomijanie w nazwach przedrostków oznaczających właściciela,
- usunięciu poleceń języka SQL definiujących użytkowników bazy danych, których należy stworzyć narzędziami SQL Servera, nie z poziomu języka SQL
- najlepiej wyodrębnić instrukcje służące do tworzenia indeksów (zwłaszcza w przypadku pól kluczowych) i wykonać je dopiero po przeniesieniu danych do bazy,
- średniki rozdzielające poszczególne polecenia języka SQL najlepiej zastąpić słowem kluczowym *go*.

W ten sposób zmodyfikowany skrypt, zawierający instrukcje w języku SQL służące do założenia bazy danych o odpowiedniej strukturze, najlepiej wykonać w programie *SQL Enterprise Manager* lub *ISQL/W*.

### 4.2. Przeniesienie zawartości bazy danych

O ile migracja struktury bazy danych nie nastręcza wielu problemów i można ją w dużym stopniu zautomatyzować, o tyle skopiowanie zawartości bazy danych nie jest już zadaniem trywialnym i wymaga większego nakładu pracy. W przypadku wcześniej badanych serwerów (Ingres [6] i Informix [5]) zadanie to również było łatwiejsze, gdyż:

- istniała możliwość importu danych bezpośrednio z narzędzi GUPTA SQLSystem, w formacie, w jakim można je było automatycznie wyeksportować z bazy danych SQLBase, niemal bez zmian,
- oba wcześniej testowane serwery posiadają dobrze opisane, stabilne funkcje ładowania danych z określonego formatu plików tekstowych, zintegrowane z pozostałymi narzędziami służącymi do zarządzania bazami danych na serwerze.

Okazuje się, że narzędzia zintegrowane do zarządzania bazami danych na SQL Serverze umożliwiają import danych tylko z innego serwera tej samej firmy, a funkcja *load* działa tylko dla plików o wewnętrznym formacie uzyskiwanym przy składowaniu bazy z SQL Servera. Jedyną możliwością skopiowania danych z plików tekstowych jest wykorzystanie mało

przyjaznego dla użytkownika programu DOS-owego o nazwie *bcp* (ang. bulk copy) ze stacji klienta. Program ten wymaga zainstalowania na komputerze klienta dodatkowego oprogramowania komunikacyjnego do porozumiewania się z SQL Serverem z systemu DOS. Składnia wywołania programu *bcp* (którego opis w dokumentacji elektronicznej pozostawia wiele do życzenia) jest następująca:

*BCP.EXE <dbtable> {in ¦ out} <datafile>*
*[/m <maxerrors>] [/f <formatfile>] [/e <errfile>]*
*[/F <firstrow>] [/L <lastrow>] [/b <batchsize>]*
*[/n <native type>] [/c <character type>]*
*[/t <field terminator>] [/r <row terminator>]*
*[/i <inputfile>] [/o <outfile>] [/a <packetsize>]*
*[/E <explicit identity>] [/U <username>] [/P <password>]*
*[/S <server>] [/v <version>]*

Znaczenie podstawowych opcji wywołania tego programu jest następujące:

*<dbtable>* - nazwa tablicy bazy danych, do której lub z której dokonujemy importu/eksportu, podana w formacie: <nazwa_bazy>.<właściciel_tablicy>.<nazwa_właściciela>,

*in* - słowo kluczowe oznaczające proces importu danych do bazy,

*out* - słowo kluczowe oznaczające proces eksportu danych z bazy,

*<datafile>* - nazwa pliku, z którego dane zostaną wczytane do bazy danych lub do którego mają być zapisane, z pełną jego ścieżką dostępu,

*<formatfile>* - nazwa pliku z opisem formatu (typów) importowanych lub eksportowanych danych,

*/c* - opcja informująca, że plik z danymi do importu ma charakter znakowy,

*<field terminator>* - znak oddzielający wartości poszczególnych pól tablicy, który nie może być jednocześnie składnikiem żadnego z pól,

*<row terminator>* - znak oddzielający poszczególne wiersze, który nie może być jednocześnie składnikiem żadnego z pól,

*<username>* - nazwa użytkownika,

*<password>* - hasło podanego użytkownika (jeśli nie chcemy podać go jawnie w linii poleceń, omijamy tę opcję i pytanie o hasło pojawi się po uruchomieniu programu),

*<server>* - nazwa serwera, na którym zainstalowana jest baza danych.

Poniżej zamieszczono przykładowe wywołanie tego programu do załadowania tablicy *ksiazki* w bazie danych *biblms* na serwerze *ZTINT* jako użytkownik *sa* z pliku znakowego *ksiazki.txt*. Pola w tym pliku oddzielone są linią pionową (¦), a wiersze znakiem nowej linii: *bcp biblms.dbo.ksiazki in c:\dorota\ksiazki.txt /U sa /S ZTINT /c /t "¦" /r "\n"*

W trakcie eksperymentów nie udało się stworzyć pliku z poprawnymi, akceptowalnymi dla programu *bcp* danymi, nawet plik automatycznie generowany przez *bcp* w trakcie eksportu danych nie miał poprawnego formatu, który mógłby być wykorzystany do importu tych samych danych z powrotem do bazy.

Za pomocą programu *bcp* można zaimportować dane do pojedynczej, wybranej tablicy z pliku tekstowego o określonym formacie. Należy przy tym zwrócić uwagę na to, czy parametr wybranej bazy danych *Select Into/Bulk Copy* jest ustawiony, czy wyłączony. Domyślnie parametr ten jest wyłączony, co oznacza, że nie dopuszcza się operacji modyfikujących na bazie nie objętych transakcjami (takich operacji, które nie są zapisywane w pliku log). Wówczas należy zadbać o to, by nie doszło do przepełnienia miejsca na urządzeniu, na którym log jest zapisywany. Przepełnienie logu może prowadzić nawet do uszkodzenia całej bazy danych (jak występowało to w czasie prowadzonych eksperymentów). Opcja *Select Into/Bulk Copy* ustawiona sprawia, że operacje czynione przez program *bcp* nie są objęte transakcjami i nie są zapisywane w logu.

Jak już wspomniano w rozdziale 4, ładowanie danych do tablicy z programu *bcp* nie jest możliwe, jeśli dla tej tablicy istnieje indeks unikalny. Ze względów efektywności działania najlepiej usunąć wszystkie indeksy i założyć je dopiero po wprowadzeniu danych. Należy jednak uwzględnić niebezpieczeństwo doprowadzenia do niespójności danych, zwłaszcza jeśli wystąpiły sytuacje błędne w czasie importu. Jest to jeden z podstawowych mankamentów działania programu *bcp*.

Kolejnym mankamentem jest wypisywanie komunikatów zawierających mało precyzyjne informacje o błędach, które wystąpiły w czasie importu danych: brak jest danych, którego wiersza błąd dotyczy i precyzyjnego opisu tegoż błędu. W trakcie badań uzyskiwano następujące komunikaty o sytuacjach błędnych: "Attemp to convert data stopped by syntax error" oraz "Data conversion stopped by overflow" i lokalizacja wierszy będących ich przyczyną nie była zadaniem trywialnym, zwłaszcza że w niektórych przypadkach mimo wystąpienia błędu wszystkie wiersze zostały zaimportowane, a w innych dane zaimportowane były niekompletne i trzeba było odnaleźć przyczynę błędu. Zarówno ergonomia, jak i stabilność programu *bcp* pozostawiały wiele do życzenia.

## 5. Podstawowe różnice funkcjonalne w dostępie do integrowanych baz danych za pomocą narzędzi GUPTA SQLSystem

Jednym z celów badań było dostosowanie gotowej aplikacji współdziałającej z serwerem SQLBase do współpracy z SQL Serverem. Istotne jest więc podanie podstawowych różnic w wykorzystaniu nowego serwera - MS SQL Server w porównaniu do serwera SQLBase.

Dla całego zestawu narzędzi GUPTA SQLSystem należy uwzględnić, że:

1. Chcąc korzystać z automatycznego zatwierdzania transakcji (ang. autocommit), które jest realizowane przez oba serwery, należy ustawić opcję *sybautocommit=on* w pliku *sql.ini*. Jednak i wówczas jest możliwość objęcia transakcją większej liczby operacji (i ewentualnego jej wycofania) z aplikacji w SQLWindows, poprzez podanie początku i końca transakcji explicite (wykonanie polecenia *begin transaction* i odpowiednio *commit transaction* lub *rollback transaction*).

2. Mimo tego, że MS SQL Server nie wspiera mechanizmu zmiennych wiążących (ang. bind variables), można z tego mechanizmu korzystać, gdyż jest on realizowany przez oprogramowanie SQLNetwork, które w momencie kompilacji zapytania zachowuje zapytanie i wszystkie nazwy zmiennych, a w czasie wykonywania zapytania - związuje wartości zmiennych z nazwami i tak przygotowane zapytanie zawierające wartości zmiennych przesyła do SQL Servera.

3. Klauzuli *check exists* na końcu zapytania modyfikującego lub usuwającego dane z tablicy można używać z SQL Serverem tylko wtedy, gdy ustawione jest słowo kluczowe *checkexists* w *sql.ini*. Klauzula ta powoduje, że serwer baz danych przekazuje do aplikacji informację o sytuacji błędnej w przypadku, gdy w tablicy nie było wierszy spełniających podane warunki modyfikacji lub usuwania. Ponieważ MS SQL Server nie wspiera tej możliwości, klauzula *check exists* nie będzie respektowana w obrębie procedur bazy danych (ang. stored procedures), gdyż wówczas zawartość zapytania w języku SQL nie jest dostępna dla SQLNetwork.

4. W SQL Serverze używa się pojęcia DBProcess dla struktury, która śledzi połączenia do bazy danych i zapewnia komunikację aplikacji z serwerem. DBProcesy są wzajemnie od siebie niezależne, podczas gdy w serwerze SQLBase połączenia do bazy danych z jednej aplikacji są od siebie zależne. Jeśli jeden z DBProcesów wykonuje zatwierdzenie transakcji, nie wpływa to na inne procesy tej aplikacji, każdy z nich jest bowiem oddzielną transakcją. Kursory w serwerze SQLBase nie stanowią w tym sensie odrębnych transakcji, lecz są częściami tej samej transakcji - jeśli zatwierdzamy transakcję związaną z jednym kursorem, to jednocześnie zatwierdzamy transakcje związane z pozostałymi kursorami tej aplikacji.

SQLNetwork posiada dwie zmienne środowiska dostępne przez słowa kluczowe w pliku *sql.ini*, umożliwiające dokładniejszą kontrolę operacji wykonywanych na bazach danych SQL Servera:

• *enhancedcursors* - umożliwia przekazanie zarządzania procesami związanymi z bazą danych (DBProcesami) programowi SQLNetwork, co powoduje lepsze wykorzystanie zasobów i poprawia wydajność;

• *mapgticurors* - powoduje skierowanie wszystkich odwołań modyfikujących bazę danych do jednego dodatkowego i ukrytego procesu bazy danych (DBProcesu).

Ponieważ opcja *enhancedcursors* jest rozszerzeniem opcji *mapgticursors*, to jeśli druga z tych opcji jest wyłączona, ustawienie pierwszej jest ignorowane. Jeśli aplikacja wymaga przełączania się między różnymi bazami danych na jednym SQL Serverze, ustawienia powyższe powinny być wyłączone.

5. Zachowywanie kontekstu kursora (czyli zbioru będącego wynikiem zapytania) po zatwierdzeniu transakcji nie jest realizowane przez sam SQL Server, mechanizm ten może być symulowany przez SQLNetwork, jeśli ustawimy opcję *mapgticursors*.

6. SQL Server nie wspiera też różnych poziomów izolacji. W wyniku zapytania o aktualny poziom izolacji otrzymamy odpowiedź *CS* (poziom *Cursor Stability*). Blokady do odczytu (ang. shared locks) zakładane są na stronę, na której znajduje się pobierany wiersz, zapobiegając modyfikacjom tej strony, a zwalniane są przy pobieraniu wiersza z innej strony. Można dodać klauzulę *HOLDLOCK* do jednej lub większej ilości tablic wymienionych we frazie *FROM* instrukcji *SELECT*, a wtedy blokady są zwalniane dopiero po zatwierdzeniu transakcji, co odpowiada poziomowi izolacji *RR* (*Read Repeatability*). Poziomy *RO* (*Read Only*) i *RL* (*Release Locks*) nie są dostępne dla klienta korzystającego z baz danych na SQL Serverze.

7. Dopuszcza się korzystanie z procedur baz danych (ang. stored procedures) charakterystycznych dla SQL Servera. Mogą one zawierać operacje definiowania struktury bazy danych i kontrolę dostępu do danych, jak też zmienne wiążące i dane wejściowe z nimi wiązane. Procedury mogą składać się z wielu wyrażeń, które po kompilacji stanowią pojedynczą operację do wykonania.

8. Można używać reguł bazy danych (ang. rules) i wyzwalaczy (ang. triggers) w postaci typowej dla SQL Servera.

9. SQLNetwork posiada bezpośredni dostęp do katalogów systemowych SQL Servera. Warunkiem jest posiadanie przez użytkownika prawa do odczytu (READ) do następujących tablic: *dbo.sysobjects, dbo.syscolumns, dbo.sysindexes, dbo.sysusers, dbo.sysprotects, dbo.systypes, master.dbo.spt_values*.

10. Nie są dostępne niektóre funkcje charakterystyczne dla SQLBase (należy wyeliminować je z odwołań do bazy danych, a o odpowiedni kod uzupełnić aplikację) m.in. funkcje SQLBase, wykorzystywane wewnątrz instrukcji SQL rozpoczynających się od znaku "@" (np. funkcje operujące na łańcuchach znakowych, na danych typu data/czas, funkcje matematyczne).

11. Język SAL został z kolei rozszerzony o kilka funkcji dedykowanych dla SQLServera (rozpoczynających się przedrostkiem *Sqs-*), m. in. do wykonywania procedur bazy danych, obsługi błędów pochodzących od SQL Servera, zapisu danych o długości większej niż 64 kB, szczegółowy opis tych funkcji znajduje się w dokumentacji [1].

## 6. Porównanie czasów realizacji podstawowych operacji przez aplikację klienta na danych w bazach zainstalowanych na obu serwerach

Wszystkie testy wykonane były co najmniej po 10 razy dla każdego serwera i każdego ustawienia (z wyświetlaniem i bez dla zapytań, których wynikiem są zbiory rekordów lub z potwierdzeniem i wycofaniem transakcji dla zapytań modyfikujących dane). Prezentowane wyniki stanowią wartości średnie uzyskane w czasie testów. Wyniki badań prezentowane są na rysunkach (rys. 3 - rys. 12).

Ponieważ dane z SQL Servera do aplikacji transmitowane są za pośrednictwem programu routera (SQLNetwork) można się było spodziewać, że związane z tym będą czasowe narzuty na komunikację, co wydłuży czasy dostępu do danych na tym serwerze w stosunku do serwera SQLBase, zwłaszcza jeśli wynikiem zapytania będą większe zbiory danych. Taki efekt można istotnie zaobserwować w wynikach testu zapytań realizujących selekcję numeryczną i alfanumeryczną: w przypadku pobierania przez aplikację ze zbioru wynikowego tylko jednej wartości czas dostępu do niej z obu serwerów jest dla użytkownika niezauważalny (choć dla serwera SQLBase jest krótszy), natomiast gdy w aplikacji wyświetlamy cały zbiór wynikowy, można zaobserwować wyraźną (przeszło 100%) różnicę w czasach dostępu do danych, na niekorzyść SQL Servera.

Test projekcji danych z jednej tablicy dawał natomiast dość nieoczekiwane rezultaty - w przypadku niewyświetlania danych otrzymywane wyniki były niemal identyczne z opisanymi wcześniej wynikami testów selekcji, natomiast przy wyświetlaniu całego zbioru wynikowego niespodziewanie zanotowano czas dostępu o 25% dłuższy dla danych przechowywanych na serwerze SQLBase.

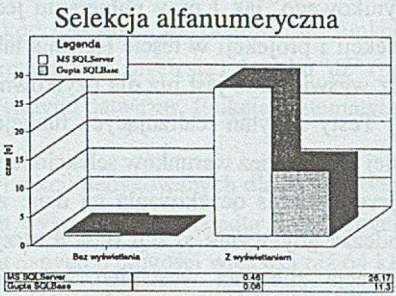
Wyniki zapytania zawierającego łączenie dwóch tablic otrzymujemy w aplikacji znacznie szybciej z serwera SQLBase niż MS SQL Servera zarówno przy wyświetlaniu całego zbioru

wynikowego, jak i przy pobieraniu jednego wiersza z tego zbioru. W przeciwieństwie do selekcji i projekcji w teście łączenia tablic czas oczekiwania na dane z SQL Servera nawet bez wyświetlania jest już dla użytkownika zauważalny (prawie 5 sekund).

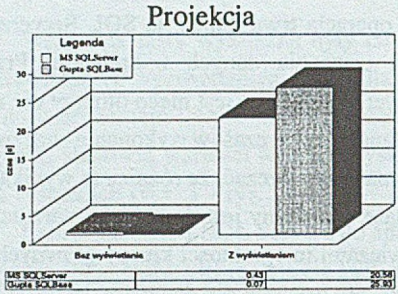
Testy zapytań realizujących funkcje agregujące obliczane na danych z jednej kolumny całej tablicy (bez warunków selekcjonujących dane) dawały rezultaty znacznie lepsze dla SQL Servera - czas oczekiwania na dane z tego serwera był dla użytkownika niezauważalny, podczas gdy wynik z SQLBase'a otrzymywano po około 2 sekundach. Wyjątkiem jest tu operacja zliczania wystąpień, która dawała niemalże natychmiastowe wyniki dla obu serwerów. Natomiast dla agregatów obliczanych na 10% tablicy czas ich uzyskania z obu serwerów jest niezauważalny dla użytkownika, choć nieco dłuższy dla serwera SQLBase, znów z wyjątkiem zliczania wystąpień, którego wynik uzyskiwany jest nieco szybciej z serwera SQLBase.

Test czasu modyfikacji danych, wykonany transakcyjnie, dawał niemalże identyczne wyniki na obu serwerach, gdy zawierał zatwierdzenie transakcji, w przypadku wycofania transakcji z punktu widzenia użytkownika aplikacji operacja trwa dłużej na SQL Serverze. Zupełnie inna jest sytuacja przy pomiarze czasu usuwania danych z aplikacji. Przy zatwierdzaniu transakcji czas trwania operacji na serwerze SQLBase jest nieco dłuższy niż na SQL Serverze, natomiast gdy transakcja zostaje wycofana, różnica czasów wykonania znacznie się zwiększa, również na niekorzyść SQLBase'a. Można przypuszczać, że różnice te wynikają z faktu, iż proces usuwania danych z serwera SQLBase obarczony jest dodatkowo narzutem czasu na kontrolę spójności danych zadeklarowaną więzami integralności kluczy głównych i obcych. Mimo tego, że tego samego rodzaju więzy zostały ustalone dla danych w bazie SQL Servera, serwer ten nie przeprowadza żadnej kontroli w momencie usuwania danych, co prowadzi często do utraty ich spójności - pozostania w bazie wierszy potomnych z wartościami klucza obcego, które nie posiadają odpowiedników w tablicy nadrzędnej, z której dane zostały usunięte. W czasie prowadzonych eksperymentów SQL Server sprawdzał zgodność wartości kluczy obcych z kluczem głównym tylko przy operacji wstawiania nowych danych lub modyfikacji danch istniejących, natomiast operacja usuwania danych ignorowała zadeklarowane więzy i prowadziła do niespójności danych.

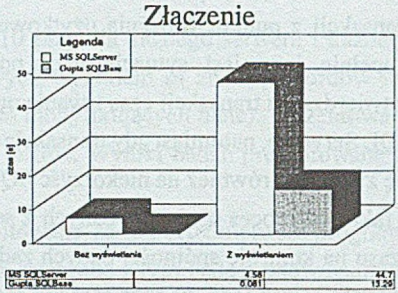

Rys. 3. Selekcja numeryczna
Fig. 3. Numerical selection



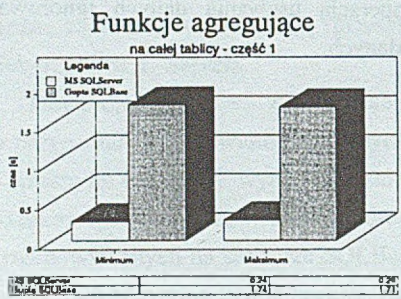

Rys. 4. Selekcja alfanumeryczna
Fig. 4. Alphanumerical selection



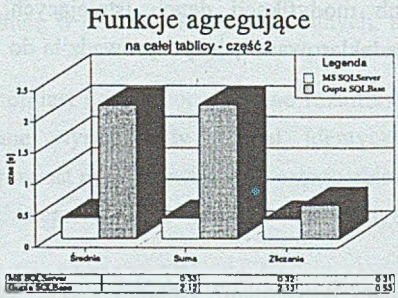

Rys. 5. Projekcja
Fig. 5. Projection



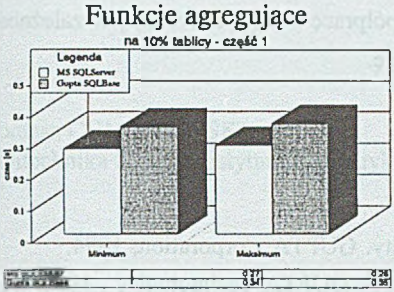

Rys. 6. Złączenie
Fig. 6. Join



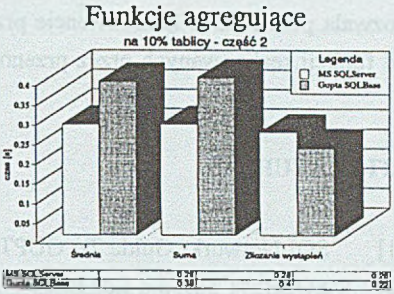

Rys. 7. Funkcje agregujące część 1
Fig. 7. Aggregate functions part 1



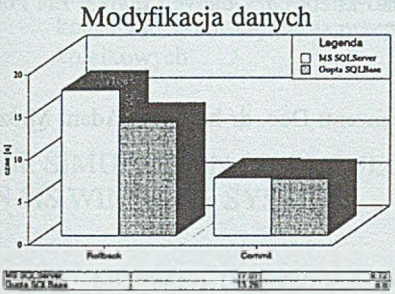

Rys. 8. Funkcje agregujące część 2
Fig. 8. Aggregate functions part 2



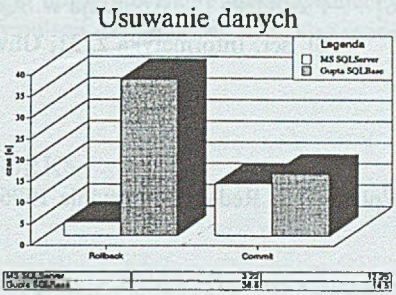

Rys. 9. Funkcje agregujące z selekcją część 1
Fig. 9. Aggregate functions with selection part 1




Rys. 10. Funkcje agregujące z selekcją część 2
Fig. 10. Aggregate functions with selection part 2




Rys. 11. Modyfikacja danych
Fig. 11. Updating of data




Rys. 12. Usuwanie danych
Fig. 12. Deleting of data

## 7. Podsumowanie

Praca stanowi porównanie działania oprogramowania powstałego za pomocą GUPTA SQL System we współpracy z dwoma serwerami baz danych - GUPTA SQLBase i Microsoft SQL Server. Przedstawiono poszczególne etapy realizacji współpracy pakietu GUPTA SQLSystem z bazą SQL Server za pośrednictwem routera. Omówiono metody przeniesienia bazy danych z serwera SQLBase na SQL Server. Wskazano podstawowe różnice funkcjonalne w dostępie do integrowanych baz danych, jakie należy uwzględnić, by aplikacja mogła współpracować z nowym serwerem baz danych. Przedstawiono również wyniki przeprowadzonych testów

czasów dostępu do danych na obu serwerach z aplikacji napisanej w SQLWindows, która pozwala przewidzieć konsekwencje przejścia na współpracę z SQL Serverem, w zależności od funkcji realizowanych przez przenoszoną aplikację.

## LITERATURA

[1] SQLNetwork: Guide To GUPTA Connectivity. GUPTA Corporation, 1994.

[2] SQLBase SqlTalk Language Reference Manual. GUPTA Technologies , 1991.

[3] Relacyjne bazy danych. Świeża krew, świeża siła. PCMagazine po polsku, wrzesień 1993.

[4] Dokumentacja elektroniczna oprogramowania MS SQL Server.

[5] Pierzchała D.: Dostęp aplikacji w SQLWindows do baz danych na różnych serwerach. ZN Polit. Śl. ser. Informatyka z. 30, Gliwice 1996.

[6] Pierzchała D.: Aplikacja w SQLWindows jako klient bazy danych Ingres. ZN Polit. Śl. ser. Informatyka z. 31, Gliwice 1996.

Recenzent: Doc. dr hab. inż. Adam Mrózek

Wpłynęło do Redakcji 9 grudnia 1996 r.

**Abstract**

The integration of database systems GUPTA SQLBase (installed on Novell NetWare server) and MS SQL Server (working on MS Windows NT Server) using GUPTA SQLNetwork router programme was the aim of described researches.

The methods of database's structure and data transfer from SQLBase to SQL Server are discussed in the paper. Comparison of SQLWindows' application's collaboration with two database servers-SQLBase and SQL Server is accomplished. The main changes needed in application to retrieve data from the new server are indicated. The last section encloses results of measuring the speed of access to data on SQLBase and SQL Server from the same application written in SQLWindows for selection, projection, aggregate functions, joins, modifying and deleting data (Fig. 3 - Fig. 12).