Akademia Górniczo – Hutnicza im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

Katedra Informatyki

Praca magisterska

Analiza porównawcza wybranych własności systemów zarządzania bazami danych

Mirosław Lach

Numer albumu: 209062

Promotor:

Prof. dr hab. inż. Antoni Ligęza

Kraków 2008

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

Wstęp

Zdecydowana większość współczesnych systemów informatycznych jest budowana w oparciu o system bazodanowy jako podstawowy komponent do przechowywania danych. Bardziej skomplikowane systemy posiadają nawet więcej niż jedno źródło danych, bądź komunikują się z innymi systemami (opartymi o bazę danych). Systemy bazodanowe, lub bardziej precyzyjnie, Systemy Zarządzania Bazami Danych (ang. Database Management Systems) dzięki swojej wydajności, współbieżności, trwałości zapisu, bezpieczeństwu, mechanizmom utrzymania spójności oraz językowi manipulacji danymi stały się nierozłącznym i podstawowym elementem współczesnych systemów informatycznych.

Z kolei podstawowe wymagania stawiane systemom informatycznym, takie jak wydajność, niezawodność, skalowalność, utrzymywalność, łatwość obsługi czy bezpieczeństwo, powodują, iż deweloperzy aby sprostać oczekiwaniom rynku muszą umiejętnie poruszać się w świecie systemów bazodanowych. Pytania, jakie stawia aktualnie projektant systemowy, to nie czy wykorzystać SZBD w projektowanym systemie, ale jaki SZBD wykorzystać, aby spełnić wszystkie oczekiwania klienta jednocześnie nie zawyżając kosztów systemu.

Rozdział I Cel i zakres pracy

Aktualnie w literaturze brak jest pracy, która zakresem obejmowałaby przegląd i porównanie najpopularniejszych SZBD (zarówno komercyjnych jak i niekomercyjnych) oraz w praktyczny sposób przedstawiałaby metodologię procesu wyboru „najlepszego" SZBD (ang. DBMS) do projektowanego systemu informatycznego oraz prezentowałaby wyniki testów wydajnościowych dla przykładowego systemu. Niniejsza praca ma na celu wypełnienie tej luki w odniesieniu do wybranych, popularnych SZBD.

Praca rozpoczyna się od krótkiego wstępu teoretyczno – historycznego, którego celem jest wskazanie aktualnych trendów, w dziedzinie systemów bazodanowych. Ponadto praca zawiera rozdział poświęcony teorii systemów relacyjnych, które są najpowszechniej wykorzystywanym modelem systemów bazodanowych.

W dalszej kolejności zostanie przedstawiona krótka analiza aktualnego stanu rynku systemów zarządzania bazą danych, z wyróżnieniem systemów komercyjnych jak i nie komercyjnych (coraz bardziej zbliżonych funkcjonalnie do produktów komercyjnych).

Kolejny rozdział zostanie poświęcony omówieniu wybranych przez autora SZBD, do których należą: DB2 [VISSER 2003], Microsoft SQL Server [ITZIK 2006], MySQL [KOFLER 2005], Oracle [ALAPATI 2005] oraz PostgreSQL [MATTHEW 2005].

Rozdział następny będzie zawierał próbę zdefiniowania wymagań dotyczących SZBD, istotnych dla większości systemów informatycznych. Analiza będzie oparta o przytoczone najpopularniejsze benchmarki bazodanowe [KOZ 2007].

W rozdziale siódmym będą omawiane narzędzia, które można wykorzystać do monitorowania SZBD, a w szczególności te, które umożliwią pomiary „zachowania" różnych SZBD w zbliżonych warunkach (w celu ich porównania). Zostaną zaprezentowane zarówno narzędzia systemowe (oferowane na przykład przez system operacyjny Linux czy Windows) jak i narzędzie własne zaprojektowane na potrzeby niniejszej pracy.

Dalsza część pracy będzie się skupiała na testach wydajnościowych wybranych SZBD w możliwie zbliżonym środowisku, obsługujących przykładowa relacyjną bazę danych. Zostanie również przedstawiona, zaproponowana przez autora, metodologia testów, a w ostatnim rozdziale zostaną omówione otrzymane wyniki oraz zostanie przedstawione porównanie tychże wyników w kontekście testowanych SZBD.

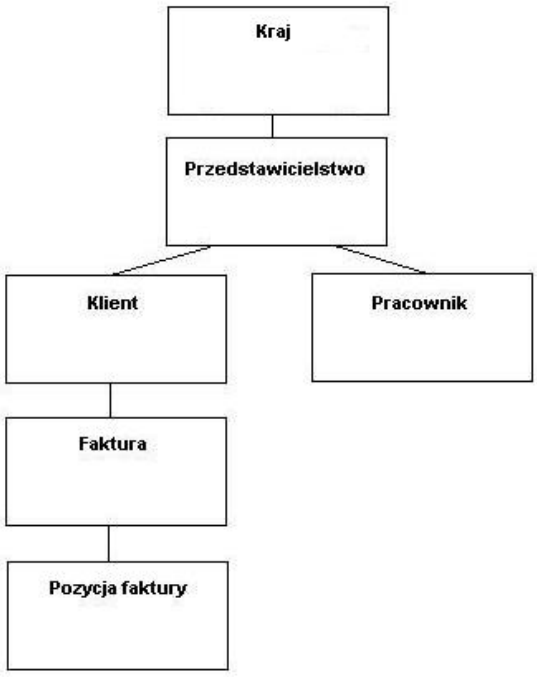
W ostatnim punkcie pracy, zostanie przedstawione krótkie podsumowanie otrzymanych rezultatów oraz zostaną przedstawione dalsze kierunki rozwoju niniejszego opracowania.

Rozdział II Historia systemów bazodanowych

Wynalezienie taśmy magnetycznej oraz dysku twardego umożliwiło składowanie dużej ilości danych w sposób, który umożliwiał ich przetwarzanie przez komputer [GRINKE 2002]. Najpierw dane przechowywano w postaci plików tekstowych lub plików rekordów [WIRTH 1989]. Proces przetwarzania danych odbywał się wówczas wsadowo. Nie istniała możliwość tworzenia zapytań „ad hoc" lub uaktualniania danych na bieżąco (ang. On-line). Rozwiązaniem było stworzenie abstrakcyjnego modelu danych widocznego dla użytkownika, który reprezentowałby dane oraz związki między nimi i równocześnie skrywałby strukturę fizyczna [GRINKE 2002].

2.1. Model hierarchiczny

Hierarchiczny model danych jest rozszerzeniem modelu opartego na rekordach składających się z pól zgrupowanych w plikach płaskich [WWWBD 2008]. W modelu hierarchicznym zostaje wprowadzony typ rekordów oraz związek nadrzędny – podrzędny pomiędzy rekordami (Rys. 1).



Rys.1. Model hierarchiczny

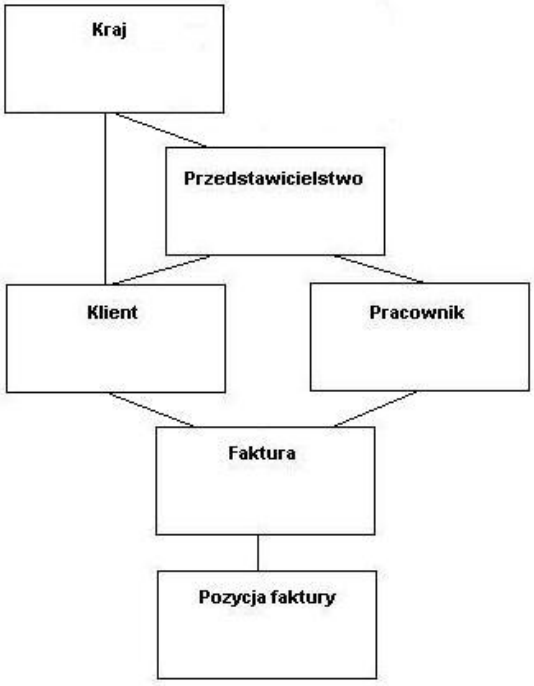
Typ rekordu to struktura danych, która posiada swoją nazwę, składająca się ze zbioru nazwanych pól. Każde zaś pole jest określone przez typ danych oraz umożliwia zapisanie pojedynczego atrybutu obiektu opisywanego przez rekord. Relacja nadrzędny – podrzędny tworzy strukturę drzewa, w której każdy rekord (z wyjątkiem korzenia, ang. *root*) związany jest z dokładnie jednym rekordem nadrzędnym, oraz każdy rekord (z wyjątkiem liścia, ang. leaf) związany jest z jednym lub wieloma rekordami podrzędnymi [WWWBD 2008].

W modelu hierarchicznym operowanie na danych sprowadza się do wyszukiwania rekordów określonego typu, będących w relacji podrzędnej względem danego rekordu i którego wartości atrybutów spełniają określone warunki. Inne operacje to dodawanie lub usuwanie rekordów oraz edycja pól rekordów.

2.2. Model sieciowy

Sieciowy model danych jest zbliżony do modelu hierarchicznego. Związek nadrzędny – podrzędny pomiędzy rekordami zostaje zastąpiony w tym modelu, tzw. typem kolekcji (ang. set) [WWWBD 2008]. Jest on złożonym typem danych zawierającym odniesienia do innych rekordów określonego typu. Określenie typu kolekcji polega na zdefiniowaniu typu rekordu „właściciela" oraz typów rekordów będących elementami kolekcji. Usuwa ono ograniczenia modelu hierarchicznego dotyczące związków – w modelu hierarchicznym mogły jedynie istnieć związki jeden do wielu (Rys. 2).

Operowanie danymi, tak jak w przypadku modelu hierarchicznego ma również charakter proceduralny i sprowadza się do wyszukiwania rekordu na podstawie zawartości pól i/lub przynależności do danego wystąpienia typu kolekcji [WWWBD 2008]. Możliwe jest również dokonywanie edycji atrybutów danego rekordu oraz jego usuwanie. Wyszukiwanie danych w modelu sieciowym wymaga wielokrotnych przejść pomiędzy rekordami a jego charakter określa się jako *nawigacyjny*.



Rys.2. Model sieciowy

2.3. Model relacyjny

Możliwości, które pojawiły się w modelu sieciowym, a także ograniczenia wynikające z jego istoty (m.in. nawigacyjny charakter wyszukiwania informacji) wpłynęły na badania nad systemem relacyjnym oraz na późniejszy ich rozwój. Model relacyjny, opracowany przez E. F. Codda w latach 70-80 XX wieku powstał w odpowiedzi na następujące potrzeby [GRINKE 2002]:

- zwiększenie niezależności danych w bazie;
- udoskonalenie zarządzania systemem;
- umożliwienie tworzenia zapytań ad hoc i selektywnego dostępu do danych;
- umożliwienia matematycznego opisania danych oraz ich wyszukiwania.

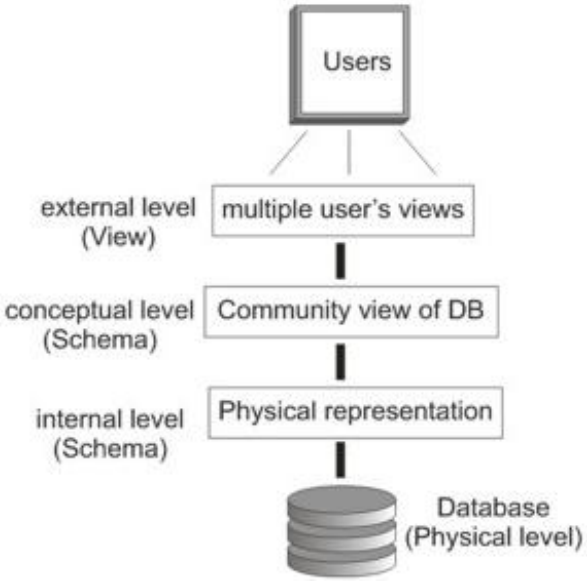
Model relacyjny oparty jest tylko na jednej strukturze – relacji (ang. relation) [WWWBD 2008]. Model ten zunifikował dane i metadane (dane opisujące dane) do jednej reprezentacji [GRINKE 2002]. Pojęcie relacji można uważać za abstrakcję bardziej intuicyjnego pojęcia tabeli (składającej się z wierszy i kolumn, w której na przecięciu

każdego wiersza z każdą kolumną występują atomowe wartości wybranych atrybutów) [WWWBD 2008].

2.4. Model semantyczny

Problem oddzielenia sposobu składowania danych od reprezentowania danych istniejących w rzeczywistości znalazł swoje odzwierciedlenie w architekturze ANSI/SPARC (American National Standards Institute, Standards Planning And Requirements Committee) (Rys. 3), a sam model relacyjny umożliwił opracowanie podejścia nazwanego *modelem semantycznym* [GRINKE 2002]. Założeniem tego modelu było dostarczenie mechanizmów, dzięki którym strukturalne aspekty modelowanej rzeczywistości mogły zostać odzwierciedlone w modelu relacyjnym [GRINKE 2002].

Najprostszym a zarazem popularnym spośród wielu modeli jest model związków encji (ang. entity-relationship model - ERM). W modelu tym poprzez diagramy związków encji (ang. *entity-relationship diagram* – ERD), można w graficzny sposób zaprezentować podstawowe elementy schematu bazy danych. Diagram ERD został rozwinięty pozwalając dodatkowo na przedstawienie semantyki danych. Modelu semantycznego nie zastosowano wprawdzie w najbardziej popularnych systemach bazodanowych, jednak przyczynił się on do wskazania ograniczeń modelu relacyjnego i jego udoskonalenia [GRINKE 2002].



Rys.3. Architektura ANSI-SPARC

2.5. Model obiektowy

Możliwość tworzenia dowolnych i często skomplikowanych typów danych, na co pozwalają języki programowania zorientowane obiektowo, powodowało, że model relacyjny w takich przypadkach był nieodpowiedni. Pomysł przechowywania struktur wytworzonych przez oprogramowanie zorientowane obiektowo prowadził do powstania modelu obiektowego (ang. object-oriented data model).
Organizacją powołaną do standaryzacji obiektowych baz danych została ODMG (Object Data Management Group) [GRINKE 2002]. Grupa skupiała głównych producentów obiektowych baz danych.

Podejście obiektowe umożliwia korzystanie z bogatszego zestawu struktur danych – klas tworzonych na potrzeby danego systemu oraz wiązanie definicji danych z metodami ich manipulacji – poprzez definiowanie metod. Definicja danych polega na określeniu klas obiektów (z zestawem atrybutów oraz metod) [SUBIETA 1999]. Związki pomiędzy obiektami zostają odzwierciedlone poprzez relację dziedziczenia oraz agregacji. Operowanie na danych jest realizowane poprzez odwołanie się do zdefiniowanych metod.

Pomimo możliwości definicji skomplikowanych struktur danych model obiektowy (a dokładnie obiektowe systemy baz danych) początkowo miał opinię modelu mniej wydajnego w porównaniu z modelem relacyjnym, jednak faktem jest, iż obiektowe bazy danych stają się coraz bardziej wydajnymi systemami [SUBIETA 1997] i są stosowane tam, gdzie wymagane są skomplikowanie struktury danych – na przykład: systemy GIS (ang. Geographic Information System) [GRINKE 2002], CAD (ang. Computer Aided Design) [GRINKE 2002], CAM (ang. Computer Aided Manufacturing) [GRINKE 2002], CASE (ang. Computer Aided Software Engineering) [SUBIETA 1997].

2.6. Systemy relacyjno – obiektowe (post-relacyjne)

Pojawienie się modelu obiektowego spowodowało reakcję firm rozwijających systemy relacyjne. Systemy te nie były systemami budowanymi od podstaw (tak jak w przypadku systemów obiektowych), lecz powstawały poprzez wzbogacenie systemów relacyjnych, które miało na celu powiększenie siły wyrazu ich modelu danych [GRINKE 2002]. Do rozszerzeń, które oferowały systemy relacyjno-obiektowe należały między innymi [DATE 2000]: złożone

struktury danych, abstrakcyjne typy atrybutów, zagnieżdżone relacje, atrybuty proceduralne (wirtualne), dziedziczenie (ang. inheritance), zapytania historyczne, rozszerzalne funkcje.

Przykładem systemu relacyjno-obiektowego jest PostgreSQL. Do obiektowych mechanizmów umożliwiających w łatwy sposób rozszerzenie systemu przez użytkownika należą [WWWPOSTGRE 2008]:

- dziedziczenie – umożliwia tabeli dziedziczenie kolumn od jej rodzica;
- złożone typy danych – umożliwiają tworzenie kolumn będących tablicami typów podstawowych;
- funkcje – PostgreSQL dostarcza cztery rodzaje funkcji: języka zapytań (SQL), języka proceduralnego (PL/pgSQL), wewnętrzne oraz języka C. Ponadto funkcje mogą być przeciążane (może istnieć klika funkcji o takiej samej nazwie, lecz różniących się argumentami).

Rozdział III Wstęp do systemów relacyjnych

Relacyjny model danych został zaproponowany przez E. F. Codda, w opublikowanej w 1970 roku pracy, która położyła fundament pod aktualnie najbardziej popularny model danych [CODD1 1970]. Model relacyjny spośród innych modeli danych wyróżnia się przed wszystkim jednolitymi i solidnymi podstawami teoretycznymi z dziedziny matematyki, a w szczególności z teorii relacji. Od 1968 do 1988 roku, Codd opublikował ponad 30 prac na temat relacyjnego modelu danych. W 1979 roku, na konferencji Australian Computer Society, Codd przedstawił pracę zawierającą rozszerzoną wersję relacyjnego modelu danych, nazwaną RM/T.

W 1985 roku Codd opublikował artykuł w Computerworld [GRINKE 2002], w którym podał 12 postulatów specyfikujących racjonalne wymagania w odniesieniu do relacyjnych baz danych oraz określania, kiedy baza danych jest relacyjna [CONN 2004]:

1. Postulat informacyjny – dane są reprezentowane jedynie poprzez wartości atrybutów w wierszach tabel.
2. Postulat (gwarantowanego) dostępu – każda wartość w bazie danych jest dostępna poprzez podanie nazwy tabeli, atrybutu oraz wartości klucza podstawowego.
3. Postulat dotyczący wartości NULL – dostępna jest specjalna wartość NULL dla reprezentacji wartości nieokreślonej jak i nieadekwatnej, inna od wszystkich i podlegająca przetwarzaniu.
4. Postulat dotyczący katalogu – informacje o obiektach baz danych tworzących schemat bazy danych są na poziomie logicznym zgrupowane w tabele i dostępne w taki sam sposób jak inne dane.
5. Postulat języka danych – system musi dostarczyć pełnego języka przetwarzania danych, który:
    - charakteryzuje się liniową składnią;
    - może być używany w trybie interaktywnym jak i w obrębie aplikacji;
    - obsługuję operacje definiowania danych, operacje manipulowania danymi, ograniczenia związane z bezpieczeństwem i integralnością oraz operacje zarządzania transakcjami.

6. Postulat modyfikowalności perspektyw – system musi umożliwiać modyfikowanie perspektyw, o ile jest ono semantycznie realizowalne.
7. Postulat modyfikowalności danych – system musi umożliwiać operacje modyfikacji danych (INSERT, UPDATE, DELETE).
8. Postulat fizycznej niezależności danych – zmiany fizycznej reprezentacji i organizacji dostępu nie wpływają na aplikacje.
9. Postulat logicznej niezależności danych – zmiany wartości w tabelach nie wpływają na aplikacje.
10. Postulat niezależności więzów spójności – więzy spójności są definiowane w bazie nie zależą od aplikacji.
11. Postulat niezależności dystrybucyjnej – działanie aplikacji nie zależy od modyfikacji i dystrybucji bazy.
12. Postulat bezpieczeństwa względem operacji niskiego poziomu – operacje niskiego poziomu nie mogą naruszać modelu relacyjnego i więzów spójności.

Do dwunastu postulatów istnieje uzupełnienie znane jako postulat zero [GRINKE 2002]: dla każdego systemu, który uważany jest za relacyjny musi istnieć możliwość zarządzania danymi wyłącznie poprzez jego relacyjne możliwości.

W 1990 roku E. F. Codd opublikował książkę, opisującą drugą wersję relacyjnego modelu danych [CODD2 1990]. Teorię modelu relacyjnego można podzielić na trzy zagadnienia:

- struktura danych;
- język manipulowania danymi;
- integralność danych.

3.1. Struktury danych

W relacyjnym modelu danych istnieje tylko jedna struktura danych nazywana relacją (ang. *relation*). Tabela jest mniej abstrakcyjnym określeniem pojęcia relacji, w której możemy wyróżnić kolumny oraz wiersze. Dla każdej tabeli (relacji) w modelu relacyjnym prawdziwe są poniższe zasady [GRINKE 2002]:

- każda tabela w bazie danych jest jednoznacznie określona przez swoją nazwę;
- każda kolumna w tabeli ma jednoznaczną nazwę w obrębie tej tabeli;

- wszystkie wartości w kolumnie są tego samego typu, a w każdej tabeli istnieje przynajmniej jedna kolumna. Zbiór możliwych wartości elementów danej kolumny nazywany bywa też jej dziedziną;
- nie są dozwolone powtórzenia wierszy w obrębie danej tabeli. Każdy wiersz jest różny. Tabela może istnieć bez wierszy. Wiersze relacji (tabeli) nazywa się też encjami (ang. entity);
- każde pole przecięcia wiersza z kolumną zawiera wartość atomową z dziedziny określonej przez kolumnę (nie jest możliwe przechowywanie zbioru wartości na przecięciu kolumny z wierszem). Brakowi wartości odpowiada wartość specjalna NULL, zgodna z każdym typem kolumny (chyba, że zostanie wykluczona przez definicję kolumny);
- kolumny tabeli tworzą zbiór nieuporządkowany. Kolumny nazywane są również atrybutami (ang. attribute);
- wiersze tabel tworzą zbiór nieuporządkowany (nie jest istotny ich porządek). Do rozróżnienia wierszy służą klucze główne (ang. primary key), a nie ich kolejność. Wiersze nazywane są również krotkami.
- każda tabela zawiera klucz główny (kolumnę lub zestaw kolumn), której wartości jednoznacznie identyfikują wiersz. Wartość klucza głównego nie może mieć wartości NULL.

E. F. Codd zastosował terminologię matematyczną do określenia elementów relacyjnego modelu danych. Kolumny nazwał atrybutami, a wiersze krotkami. Liczbę kolumn tabeli nazwał jej stopniem, natomiast liczbę wierszy w tabeli nazwał liczebnością. Do podstawowych pojęć dotyczących struktury danych należą:

- dziedzina (ang. domain);
- klucz główny (ang. primary key);
- klucz obcy (ang. foreign key);
- wartość NULL.

Dziedzina - zbiór elementów danych tego samego typu nazywany jest dziedziną. Inaczej mówiąc dziedzina jest zbiorem możliwych wartości, z którego pochodzą elementy pojawiające się w kolumnie danej tabeli.

Klucz główny - każda tabela (relacja) posiada klucz główny oparty o jedną lub większą liczbę kolumn, który jest unikalny oraz pozwala w jednoznaczny sposób zidentyfikować wiersz (encję). W każdej relacji może istnieć wiele kluczy, które jednoznacznie identyfikują wiersze. Nazywają się one kluczami kandydującymi, przy czym każdy z nich musi być jednoznacznie określony. Co więcej, klucz kandydujący nie może mieć wartości NULL. Spośród nich wybierany jest klucz główny. W szczególnym przypadku, jeśli zbiór kluczy kandydujących jest jednowartościowy, staje się on automatycznie kluczem głównym.

Klucz obcy - klucze obce umożliwiają powiązanie danych pomiędzy różnymi tabelami (relacjami). Klucz obcy to kolumna w danej tabeli, która przyjmuje wartości z tej samej dziedziny, co klucz główny tabeli powiązanej.

Wartość NULL - w celu umożliwienia wskazania, iż informacje jest niepełna, nieznana lub po prostu jej nie ma została wprowadzona przez Codda w 1979 roku wartość NULL. Wartość NULL jest zgodna z każdym typem danych [ULLMAN 1999] (chyba, że jest wymuszona niemożność przechowania w danej kolumnie wartości NULL). Wartość NULL jest różna od zera i nie jest to pusty łańcuch znaków. Szczególne zastosowanie posiada przy operacjach manipulacji danymi, a w szczególności przy złączeniu zewnętrznym (ang. *outer join*). Do wykrywania wartości NULL służy operator: IS NULL. Z kolei wyrażenie NULL = NULL zwraca wartość UNKNOWN (nieokreśloną). Poniżej przedstawiono tabele prawdy dla logiki trójwartościowej:

AND	**True**	**False**	**Unknown**
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

Tab. 1. Logika trójwartościowa - operator AND

OR	**True**	**False**	**Unknown**
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

Tab. 2. Logika trójwartościowa - operator OR

IS	True	False	Unknown
True	True	False	False
False	False	True	False
Unknown	False	False	True

Tab. 3. Logika trójwartościowa - operator IS

NOT	True	False	Unknown
	False	True	Unknown

Tab. 4. Logika trójwartościowa - operator NOT

3.2. Język manipulowania danymi

Relacyjny model danych proponuje dwa języki manipulowania danymi: algebrę relacyjną oraz rachunek relacyjny. Algebra relacyjna operuje na relacjach (tzn. argumenty operatorów są relacją lub relacjami, natomiast wynikiem działań operatorów jest zawsze relacja), jest językiem proceduralnym i algorytmicznym. Rachunek relacyjny jest natomiast językiem deklaracyjnym i nieproceduralnym (określa on, co ma zostać wyszukane, a nie w jaki sposób ma się odbyć wyszukanie).

E. F. Codd w zaproponowanym modelu relacyjnym przedstawił pięć aspektów operowania danymi:

- wyszukiwanie danych;
- porównywanie danych;
- wstawianie danych;
- usuwanie danych;
- edytowanie danych.

3.2.1. Algebra relacyjna

Za pomocą zbioru ośmiu operacji algebry relacyjnej, z których każdy jako argument bierze jedną lub dwie relacje (i zwraca jedną relację), realizowane jest wyszukiwanie danych. Operacje algebry relacyjnej to [GRINKE 2002]:

- operacja sumy;

- operacja przecięcia;
- operacja różnicy;
- operacja iloczynu kartezjańskiego;
- operacja selekcji;
- operacja projekcji (rzutowania);
- operacja złączenia;
- operacja dzielenia.

Pierwsze cztery operacje wzorowane są na operatorach pochodzących z teorii zbiorów. Kolejne cztery zostały zdefiniowane specjalnie na potrzeby modelu relacyjnego. Składnia zapisu podstawowych operatorów algebry relacyjnej nie została wyspecyfikowana (prezentowana w niniejszej pracy składnia uwzględnia składnie umowną, najczęściej spotykana w opracowaniach). Składnia operacji porównania zaproponowana została przez E. F. Codda.

3.2.1.1. Operacja sumy

Operacja sumy (ang. union operation) przyjmuje dwa argumenty w postaci relacji. Wynikowa relacja zawiera wszystkie wiersze (encje) z relacji wejściowych, lecz bez powtórzeń. Wymaganiem koniecznym przeprowadzenia operacji łączenia jest identyczny schemat obu relacji wejściowych. Odpowiadające sobie kolumny muszą należeć do tych samych dziedzin (mogą się natomiast różnić nazwami).
Notacja operacji łączenia:
<relacja_1> UNION <relacja_2> -> <relacja_wynikowa>

3.2.1.2. Operacja przecięcia

Operacja przecięcia (ang. intersection operation) przyjmuje dwa argumenty w postaci relacji. Wynikowa relacja zawiera wiersze wspólne dla obu wejściowych relacji. Również w tym przypadku schemat relacji wejściowych musi być identyczny. Operacja ta działa odwrotnie niż operacją łączenia, która odrzuca wspólne wiersze.
Notacja operacji przecięcia:
<relacja_1> INTERSECTION <relacja_2> -> <relacja_wynikowa>

3.2.1.3. Operacja różnicy

Operacja różnicy (ang. difference operation) przyjmuje dwa argumenty w postaci relacji. Wynikowa relacja zawiera wszystkie wiersze z pierwszej relacji wejściowej niewystępujące w drugiej relacji wejściowej. Wymaganiem koniecznym przeprowadzenia operacji przecięcia jest identyczny schemat obu relacji wejściowych. Istotna w tym przypadku jest kolejność podawania relacji wejściowych. Operacja różnicy nie jest przemienna.
Notacja operacji różnicy:
<relacja_1> DIFFERENCE <relacja_2> -> <relacja_wynikowa>

3.2.1.4. Operacja iloczynu kartezjańskiego

Operacja iloczynu kartezjańskiego (ang. *Cartesian product operation*) przyjmuje dwa argumenty w postaci relacji. Wynikowa relacja składa się z wszystkich możliwych kombinacji wierszy z jednej relacji z wierszami z drugiej relacji. Operacja nie wymaga, aby argumenty posiadały identyczne schematy relacji. Ilość wierszy w tabeli wynikowej jest równa liczebności relacji będących argumentami operacji.
Notacja operacji iloczynu kartezjańskiego:
PRODUCT <relacja_1> WITH <relacja_2> -> <relacja_wynikowa>

3.2.1.5. Operacja selekcji

Operacja selekcji (ang. restrict operation) przyjmuje jako argument jedną relację. Wynikowa relacja zawiera ograniczoną liczbę wierszy relacji wejściowej, stosownie do określonych warunków, jakie mają spełnić wybrane atrybuty. Inaczej mówiąc operacja selekcji wybiera te wiersze, które pasują do zadanego warunku, z relacji wejściowej.
Notacja operacji ograniczenia:
RESTRICT <relacja> WHERE <warunek> -> <relacja_wynikowa>

3.2.1.6. Operacja projekcji (rzutowania)

Operacja projekcji (ang. *project operation*) przyjmuje jako argument jedną relację. Wynikowa relacja zawiera ograniczoną liczbę kolumn z relacji wejściowej. Działa zatem podobnie względem atrybutów relacji, jak operacja selekcji względem krotek.
Notacja operacji projektu:
PROJECT <relacja> <lista_kolumn> -> <relacja_wynikowa>

3.2.1.7. Operacja złączenia

Operacja złączenia (ang. *join operation*) przyjmuje dwa argumenty w postaci relacji. Dodatkowy argument stanowią nazwy kolumn nazywane kolumnami złączenia. Najczęściej są to odpowiadające sobie klucze główny i obcy relacji wejściowych. Warunek, na podstawie którego łączone są wiersze relacji wejściowych nie musi być warunkiem równości. Schemat obu wejściowych relacji nie musi być identyczny, natomiast wartości kolumn złączenia musza należeć do tej samej dziedziny. Operacja złączenia jest złożeniem operacji iloczynu kartezjańskiego, selekcji oraz rzutowania. Nie zawsze używane są wszystkie z wymienionych operacji. Złączenie tego typu nosi nazwę złączenia theta, natomiast wyróżnia się kilka szczególnych przypadków operacji złączenia.

3.2.1.7.1. Równozłączenie

Operacja równozłączenia zwraca w relacji wynikowej połączone w jeden wiersz tylko te pary wierszy z relacji wejściowych, w których wartości w kolumnach złączenia są sobie równe.
Notacja operacji równozłączenia:
EQUIJOIN <relacja_1> WITH <relacja_2> ON <kolumna_1> = <kolumna_2> -> <relacja_wynikowa>

3.2.1.7.2. Złączenie naturalne

Operacja złączenia naturalnego zwraca w relacji wynikowej połączone w jeden wiersz tylko te pary wierszy z relacji wejściowych, w których wartości w kolumnach złączenia są sobie równe oraz eliminuje z relacji wynikowej powtórzenia kolumn złączenia przy użyciu operacji projekcji.
Notacja operacji złączenia naturalnego:
JOIN <relacja_1> WITH <relacja_2> ON <kolumna_1> = < kolumna_2> -> <relacja_wynikowa>

3.2.1.7.3. Złączenie zewnętrzne

Operacja złączenia zewnętrznego w odróżnieniu od operacji złączenia naturalnego pozwala zachować w relacji wynikowej wszystkie wiersze z relacji będących argumentami operacji, nawet te z nich, dla których wartość występująca w kolumnie złączenia nie ma swojego odpowiednika w drugiej relacji. Wyróżnia się cztery rodzaje złączeń zewnętrznych:

- lewostronne złączenie zewnętrzne;
- prawostronne złączenie zewnętrzne;
- obustronne złączenie zewnętrzne;
- UNION – zawiera tylko niepasujące wiersze z pierwszej i z drugiej relacji.

3.2.1.8. Operacja dzielenia

Operacja dzielenia (ang. division operation) przyjmuje dwa argumenty w postaci relacji binarnej oraz relacji unarnej. Wynikowa relacja zawiera wszystkie wartości jednego atrybutu relacji binarnej, które zgadzają się względem wartości drugiego atrybutu z wartościami relacji unarnej.
Notacja operacji podziału:
<relacja_1> DIVISION <relacja_2> -> <relacja_wynikowa>

3.2.1.9. Operacje porównania

Operacje porównania (ang. comparision operations) to zbiór sześciu operacji, których argumentami są zawsze dwa wyrażania. Wynikiem natomiast są zawsze wartości logiczne: PRAWDA bądź FAŁSZ.
Lista operacji porównania wraz ze składnią:

- równe: <wyrażenie_1> = <wyrażenia_2>;
- nierówne: <wyrażenie_1> != <wyrażenia_2>;
- mniejsze lub równe: <wyrażenie_1> <= <wyrażenia_2>;
- mniejsze: <wyrażenie_1> < <wyrażenia_2>;
- większe lub równe: <wyrażenie_1> >= <wyrażenia_2>;
- większe: <wyrażenie_1> ><wyrażenia_2>.

3.2.2. Rachunek relacyjny

Istnieją dwa równoważne rodzaje rachunku relacyjnego: rachunek na krotkach i rachunek na dziedzinach. Rachunek relacyjny na krotkach jest podstawą języka SQL (ang. Structured Query Language), z kolei rachunek relacyjny na dziedzinach jest podstawą języka QBE (ang. Query By Example).

3.2.3. Operacje wspólne algebry i rachunku relacyjnego

Operacje te modyfikują zawartość bazy danych, niezbędne zatem jest zapewnienie zachowania integralności (Rozdział 3.3).

3.2.3.1. Wstawianie danych

Operacja wstawiania danych (ang. insert operation) wstawia krotkę do relacji. Pomimo, iż zdefiniowana w relacji kolejność kolumn nie jest istotna, to operacja wstawiania danych (dla której nie wskaże się listy kolumn) przyjmuje porządek kolumn podany przy jej definicji.

Składnia operacji:
INSERT (<wartość>, <wartość>, …) INTO <relacja>

3.2.3.2. Usuwanie danych

Operacja usuwania danych (ang. delete operation) usuwa krotki z relacji spełniające określone warunki. Operacja pozwala na usunięcie wszystkich wierszy w przypadku, gdy pomija się warunki. Możliwe jest w tej operacji określenie warunku poprzez *search_conditions*.
Składnia operacji:
DELETE <relacja> WITH <warunek>

3.2.3.3. Modyfikacja danych

Operacja modyfikacji danych (ang. update operation) zmienia wartość jednego lub więcej atrybutów, dla określonych warunkiem krotek.
Składnia operacji:
UPDATE <relacja> WHERE <warunek> SET <atrybut> = <wartość>

3.3. Integralność

Pod pojęciem integralności bazy danych, rozumiana jest jej zgodność ze stanem faktycznym fragmentu modelowanej rzeczywistości. W relacyjnym modelu danych integralność jest zapewniania przez tak zwane wewnętrzne reguły integralności, do których należą [ULLMAN 1999]: integralność encji (wiersza danych) oraz integralność referencyjna (pomiędzy związanymi relacjami). Ponadto zostało wprowadzone pojęcie dodatkowych więzów integralności, które pozwalają na zachowanie integralności dziedziny (zwane również integralnością semantyczną).

3.3.1. Integralność encji

Jest definiowana przez regułę, która mówi, iż każda relacja w bazie danych musi posiadać klucz główny. Integralność encji zapewniana jest na etapie definiowania schematu bazy danych poprzez wymuszenie określenia atrybutu (bądź zestawu atrybutów) będącego kluczem głównym.

3.3.2. Integralność referencyjna

Jest definiowana przez regułę, która określa stany, w jakich może znajdować się wartość klucza obcego. W skład dopuszczalnych stanów wartości klucza obcego wchodzi albo wartość równa wartości klucza głównego albo wartość NULL. Wartość NULL jest raczej rzadkim przypadkiem i powinno się dążyć do wyeliminowania (a raczej do niedopuszczenia) do takich przypadków. Możliwe jest również określenie na etapie definiowania schematu bazy danych, że wartość klucza obcego nie może przyjmować wartości NULL.

Zachowanie integralności referencyjnej w dużej mierze wiąże się z usuwaniem oraz z modyfikacją krotek. Model relacyjny określa sposób zachowania dla takich przypadków, który zostaje przypisany do każdego związku relacji. Istnieją trzy możliwości zachowania:

- zmiany ograniczone (ang. *restricted*) – uniemożliwione jest usunięcie krotki zawierającej wartość klucza głównego oraz zmiana tej wartości, jeśli odwołuje się do niej jakikolwiek klucz obcy;
- zmiany kaskadowe (ang. *cascades*) – usunięcie krotki powoduje usunięcie wszystkich krotek w bazie danych odwołujących się do niej poprzez wartość klucza obcego. Również zmiana wartości klucza głównego powoduje kaskadową zmianę wartości wszystkich odwołujących się do niej kluczy obcych;
- zmiany ustawiające wartość klucza obcego na NULL (ang. nullifies) – usunięcie lub zmiana wartości klucza głównego krotki powoduje ustawienie wartości wszystkich odwołujących się do niej kluczy obcych na wartość NULL.

3.3.3. Integralność semantyczna

W rzeczywistym świecie istnieją związki pomiędzy wartościami atrybutów, które wymagają dodatkowych reguł integralnościowych. Aby zapobiec nieprawidłowym stanom bazy danych (lub, chociaż ograniczyć do minimum możliwość wystąpienia błędnych wartości danych) zostały wprowadzone tak zwane ograniczenia (ang. constraints), które definiują dodatkowe więzy pomiędzy obiektami. Ograniczenia te są brane pod uwagę podczas każdej operacji zmieniającej stan bazy danych.

Rozdział IV Rynek SZBD

System Zarządzania Bazą danych (ang. Database Management System, DBMS) nazywany również serwerem bazy danych lub systemem bazy danych to oprogramowanie implementujące konkretny model danych [HECTOR 2006] (w omawianym przypadku dotyczy modelu relacyjnego - RSZBD), charakteryzujące się następującymi własnościami funkcjonalnymi:

- gwarantowanie trwałości i spójności danych w bazie danych;
- zapewnienie współbieżnego dostępu do przechowywanych danych;
- autoryzacja dostępu do serwera bazy danych;
- efektywny i przeźroczysty dostęp do danych w środowisku scentralizowanym i rozproszonym.

Funkcje SZBD [NOWAK 2005]:

- zapis, odczyt i aktualizacja danych;
- katalog dostępny dla użytkowników;
- obsługa transakcji;
- sterowanie współbieżnością;
- obsługa odtwarzania bazy;
- obsługa autoryzacji;
- obsługa transmisji danych;
- obsługa integralności danych;
- usługi wspierające niezależność danych;
- programy narzędziowe.

Zalety i wady SZBD [NOWAK 2005]:

a) zalety:

- rozszerzona obsługa składowania i odtwarzania bazy danych po awarii;
- zwiększenie możliwości wielodostępu;
- łatwość utrzymania wynikająca z niezależności danych;
- wzrost wydajności;

- poprawa zakresu i czasu dostępu dodanych;
- godzenie sprzecznych wymagań;
- oszczędność;
- przestrzeganie standardów;
- poprawa bezpieczeństwa;
- wspólny dostęp do danych;
- więcej informacji w oparciu o same dane;
- spójność danych;
- kontrola redundancji.

b) wady:

- złożoność;
- rozmiar;
- koszt SZBD;
- dodatkowe koszty sprzętu;
- koszt przeniesienia;
- sprawność;
- większy zasięg awarii.

4.1. Liderzy na rynku SZBD

W 2003 roku Gartner [WWWGARTNER 2008] oszacował rynek systemów do zarządzania relacyjnymi bazami danych (RDBMS) na 7,1 mld USD. Z tego udział w rynku liderów był następujący:

- IBM - 33,8%;
- Oracle - 32,6%;
- Microsoft - 18,7%.

W tym samym roku szacunki IDC [WWWMSI 2008] (jedna z największych firm zajmujących się badaniem rynku teleinformatycznego) różniły się od Gartnera – rynek RDBMS był wart 13,6 mld USD, a udział w rynku liderów przedstawiał się następująco:

- Oracle - 39,8%;
- IBM – 31,3%;
- Microsoft - 12,1%.

Przy tym należy zaznaczyć, iż szacunki IDC oprócz przychodów ze sprzedaży nowych licencji (szacunki Gartner obejmują jedynie dochody ze sprzedaży nowych licencji) obejmują również przychody z usług wsparcia klientów.

W kolejnym 2004 roku, według IDC wartość światowego rynku RDBMS osiągnęła 14,9 mld USD, a udział liderów w rynku przedstawiał się następująco:

- Oracle - 41,3%;
- IBM - 30,6%;
- Microsoft - 13,4%;
- Sybase - 3,1%;
- Teradata - 3,1%.

Z kolei Gartner [WWWGARTNER 2008] oszacował rynek na 12,8 mld USD z następującym udziałem:

- Oracle - 48,9%;
- IBM - 22,4%;
- Microsoft - 13,9%;
- Teradata - 3,2%;
- Sybase - 3,0%.

Według szacowań Gartner'a [WWWGARTNER 2008] rynek RDBMS w 2005 roku został wyceniony na 13,3 mld USD z następującym udziałem poszczególnych liderów:

- Oracle - 45,8%;
- IBM - 22,1%;
- Microsoft - 15,6%;
- Teradata - 3,5%;
- Sybase - 3,4%.

Z kolei 2006 rok, oszacowany na 15,2 mld USD niewiele się zmienił w porównaniu z poprzednim:

- Oracle - 47,1%;
- IBM - 21,1%;
- Microsoft - 17,4%;
- Teradata - 3,2%;
- Sybase - 3,2%.

Natomiast IDC oszacował 2006 rok na 16,5 mld USD, z następującym udziałem korporacji:

- Oracle - 47,9%;
- IBM - 21,1%;
- Microsoft - 18,4%;
- Sybase - 3,2%.

IDC szacuje, że w 2008 roku wartość rynku relacyjnych baz danych powinna przekroczyć 20 mld USD.

Dość ciekawie zarysowuje się udział RDBMS wykorzystywanych w systemach ERP. Procentowy udział w rynku systemów ERP w 2006 roku przedstawia poniższy rysunek (Rys. 4) [WWWMSI 2008]:

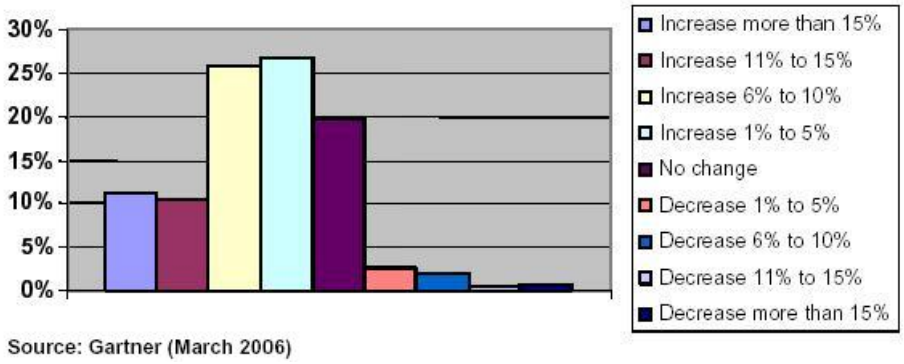
Rys.4. Udział RDBMS w systemach ERP

Istotnym elementem na rynku IT, tym samym na rynku systemów zarządzania bazami danych są produkty Open Source (open-source software, OSS). I chociaż dużo mniejszy jest udział OSS w rynku RDBMS niż na przykład w rynku systemów operacyjnych, udział ten jest jednak widoczny i godny uwagi. Pomimo iż, jak podaje Gartner, OSS RDBMS wykorzystywane są w mniej krytycznych obszarach oraz w przedziale małych i średnich przedsiębiorstw, to jednak udział ten będzie się zwiększał.

W 2008 roku przewiduje się, że z prawdopodobieństwem 70%, będą wykorzystywane w ponad 70% organizacji IT przynajmniej w jednej aplikacji, pomimo iż niewiele z nich będzie krytycznych (w sensie strategicznym przedsiębiorstwa) [GARTNER2 2006]. Analitycy zwracają również uwagę na fakt, iż właśnie produkty Open Source działają stymulująco i równoważąco na liderów rynkowych (firmy IBM, Oracle oraz Microsoft) – dostawcy rozwiązań komercyjnych w większym stopniu biorą pod uwagę sugestie oraz oczekiwania klientów, jak również otwarte standardy przemysłowe [WWWMSI 2008]. Do najpopularniejszych produktów Open Source należą MySQL (posiadająca ponad 10 milionów aktywnych instalacji), Firebird oraz PostgreSQL [WWWMSI 2008].

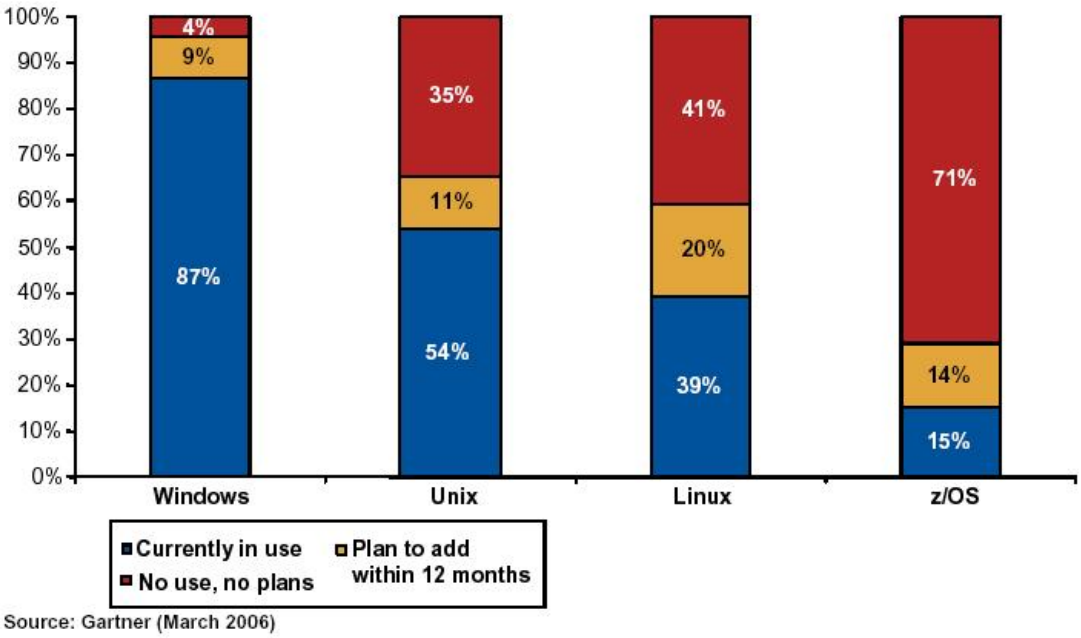
4.2. Trendy na rynku systemów zarządzania bazami danych

Gartner prowadząc analizy rynku w 2006 roku [GARTNER1 2006] zapytał respondentów z całego świata (ponad 1800 organizacji) o plany związane z finansowaniem DBMS w 2006 roku. Poniższy wykres przedstawiam zestawienie odpowiedzi [GARTNER1 2006]:



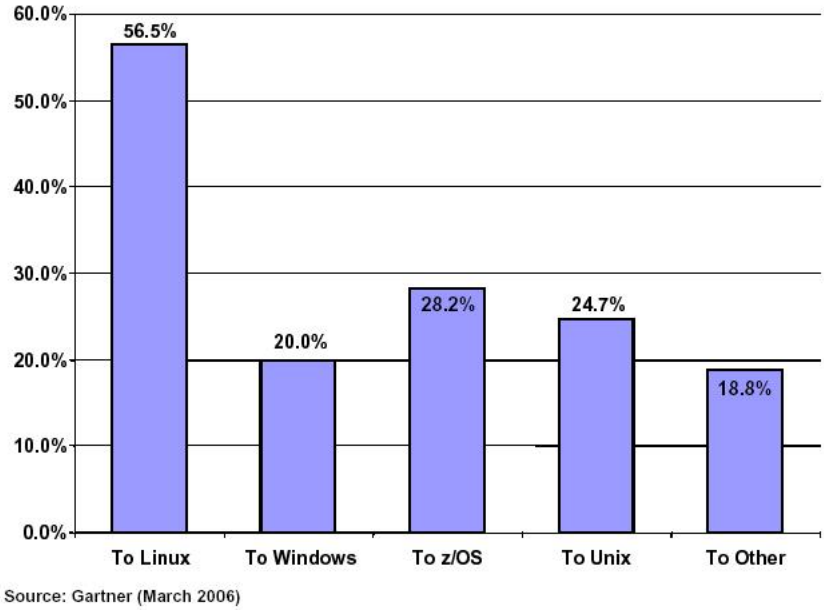
Rys.5. Plany wydatków przeznaczonych na DBMS

Kolejną istotną analizą przeprowadzona prze Gartner w 2006 roku [GARTNER1 2006], było zapytanie respondentów o plany związane z systemem operacyjnym instalowanym pod DBMS. W tym przypadku liczba respondentów wynosiła 652. Poniższy rysunek (Rys. 6) [GARTNER1 2006] zawiera zestawienie odpowiedzi respondentów, wraz z informacją o bieżącej konfiguracji:



Rys.6. Platformy systemów operacyjnych pod DBMS

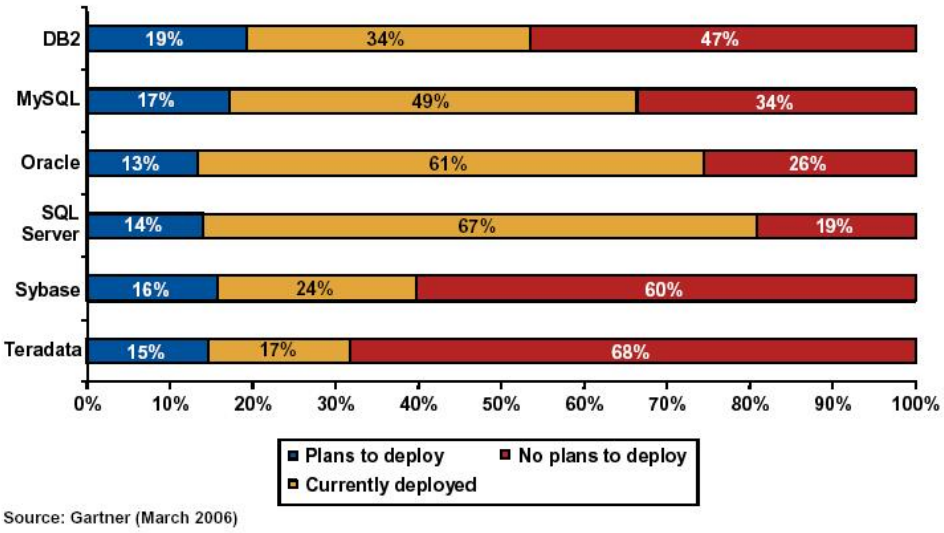
Pomimo miażdżącej dominacji systemów operacyjnych z rodziny Microsoft Windows, ankietowane organizacja deklarują [GARTNER1 2006] zdecydowana migrację na system Linux (Rys. 7) [GARTNER1 2006]:



Rys.7. Deklarowana migracja systemów operacyjnych

Jednymi z ciekawszych i chyba najbardziej związanych z tematem pracy tematów badań przeprowadzonych przez Gartner, były badania dotyczące wdrożonych w badanych

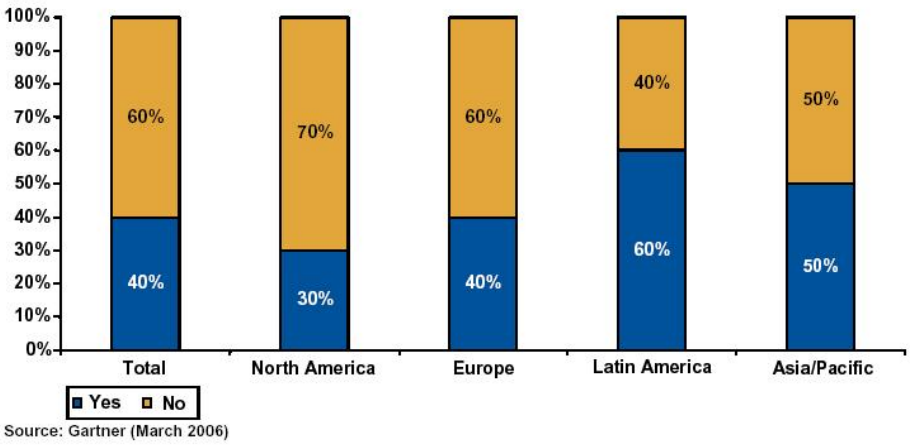
organizacjach DBMS oraz dalszych planów z nimi związanych. Jak deklaruje przeprowadzający badania, wyniki zostały opracowane na podstawie serii pytań zadanych respondentom [GARTNER1 2006]:



Rys. 8. Instalacje DBMS i plany związane z dalszymi wdrożeniami

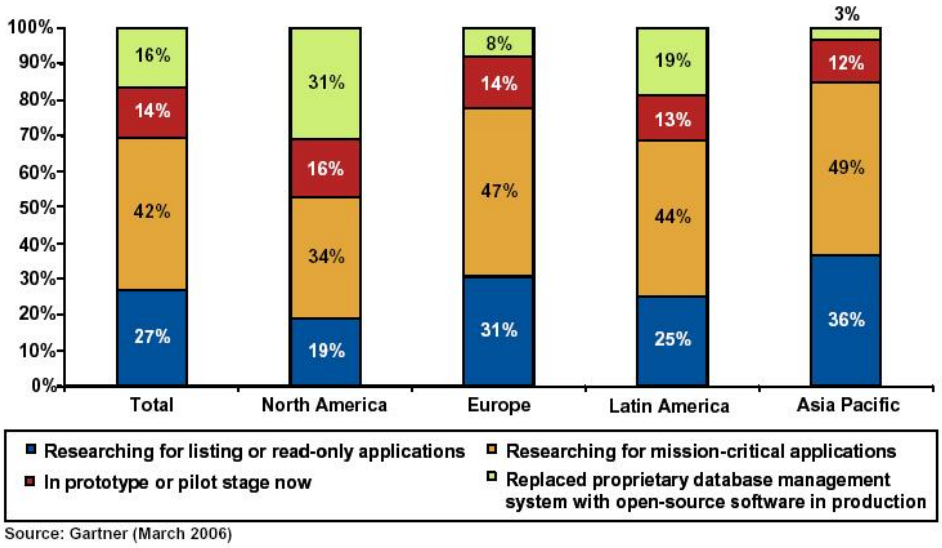
Z powyższych danych jednoznacznie wynika, że w najbliższym czasie największą popularnością będzie się cieszył system Microsoft SQL Server. Gartner jednak zaznacza, że MS SQL Server jest serwerem wykorzystywanym na szczeblu oddziałowym w przedsiębiorstwach (stąd tak wysoka liczba instalacji), a ponad 15% planowanych instalacji tegoż systemu jest związanych z nową wersją serwera bazodanowego – SQL Server 2005.

Z kolei Oracle jest systemem relacyjnym często wykorzystywanym w aplikacjach na poziomie dużego przedsiębiorstwa. Godne uwagi są również plany związane z instalacją produktu DB2 firmy IBM oraz udział OSS DBMS w rynku. W przypadku systemów OSS Gartner zauważa, że jest on wykorzystywany eksperymentalnie lub w prostych środowiskach. Ponieważ udział OSS w rynku DBMS jest istotny, zostało przeprowadzone jeszcze jedno interesujące badanie. Respondenci zostali zapytani plany zastąpienia komercyjnego DBMS rozwiązaniem OSS (Rys. 9) [GARTNER1 2006]:



Rys. 9. Plany zastąpienia komercyjnych DBMS rozwiązaniem OSS

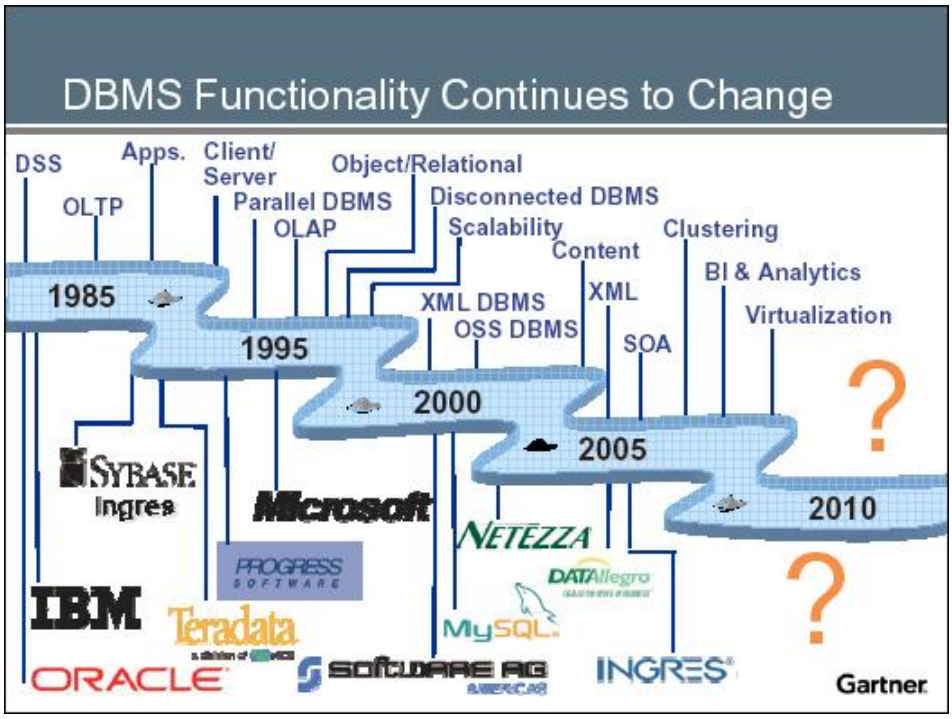
Ostatnimi wynikami badań, wartymi przytoczenia, są plany związane z dziedzinami zastosowań OSS DBMS (Rys. 10) [GARTNER1 2006]:



Rys. 10. Dziedziny zastosowań OSS DBMS

4.3. Trendy technologiczne w SZBD

Historia SZBD to ponad 20 lat doświadczeń, modernizacji oraz wprowadzania innowacyjnych rozwiązań, które odzwierciedlają się w dzisiejszych produktach. Najbardziej istotne to: OLTP (Online Transaction Processing), architektura klient/Server, OLAP (Online Analitical Processing), technologia obiektowa, XML, BI (Business Intelligence). Historię rozwoju funkcjonalnego DBMS przedstawia Rys.11 [GARTNER2 2006]:

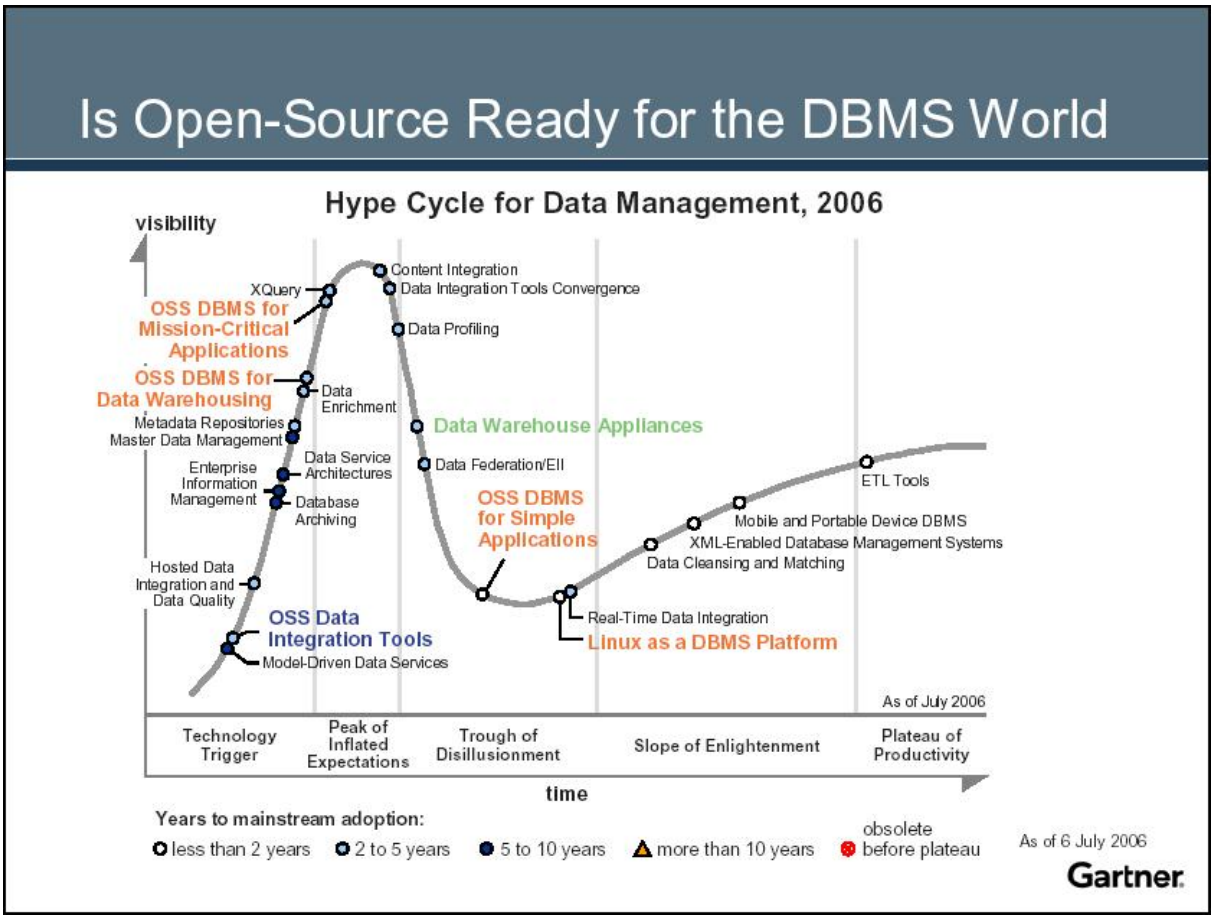


Rys. 11. Zmiany funkcjonalne DBMS

O ile określenie punktu, w którym aktualnie znajdują się DBMS pod względem funkcjonalnym nie jest problemem, o tyle problematyczne jest określenie trendu związanego z funkcjonalnością DBMS. Gartner w swoich badaniach [GARTNER2 2006] wylicza kilka kluczowych punktów, które będą istotne z punktu widzenia rozwoju funkcjonalnego DBMS:

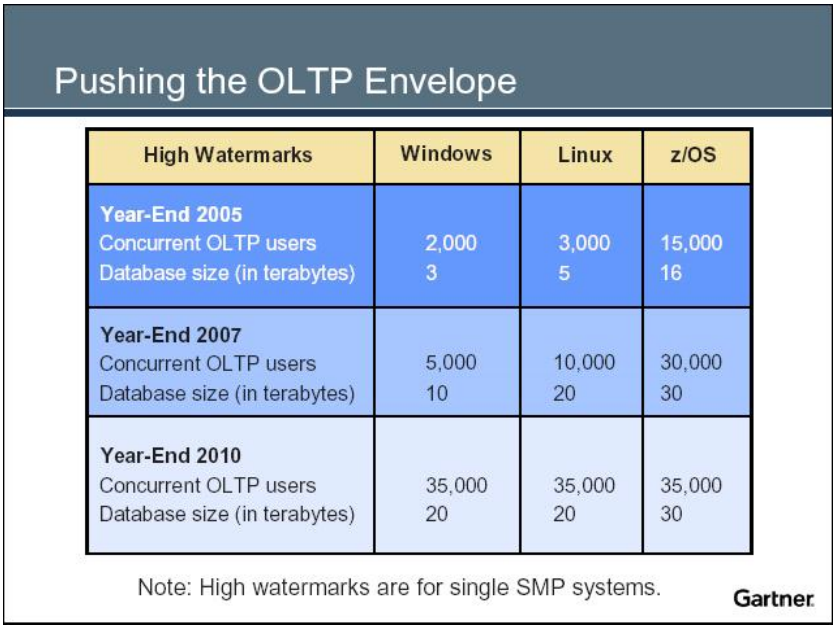
- walka o wpływy pomiędzy Linuxem i Uniksem – Gartner szacuje, że do 2010 roku wszystkie organizacje, które nie preferują systemu Microsoft Windows, będą wykorzystywały system Linux dla wszystkich aplikacji;
- rozwój funkcjonalny RDBMS w zakresie bezpołączeniowych DBMS;
- rozwój funkcjonalny RDBMS w zakresie danych niestrukturalnych – Gartner szacuje, że do 2011 roku tradycyjne bazy danych (przechowujących dane strukturalne) będą przechowywały poniżej 20% danych potrzebnych dla podejmowania strategicznych decyzji. Pozostałe dane, to dane niestrukturalne;
- rozwój funkcjonalny RDBMS w zakresie skalowalności;
- rozwój narzędzi wspierających hurtownie danych (ang. data warehouse) oraz BI (ang. Business Intelligence) – Gartner szacuje, że do 2012 roku użytkownicy będą oddziaływali z BI w ponad 85% biznesowych aplikacji;

- zwiększenie samozarządzalności DBMS (autonomiczny tuning, uczący się optymalizator, automatyczne zarządzanie obiektami, samo-tuningujący się backup oraz przywracanie baz danych) oraz redukcja zasobów wspierających – Gartner szacuje, że do 2008 roku kluczowi dostawcy DBMS osiągną bliski zeru procent planowanej niedostępności DBMS oraz zredukują o ponad 50% zasoby administratorów baz danych (ang. Database Administrator, DBA) niezbędne do zarządzania DBMS;
- rozwój open-source software (OSS) DBMS – przewidywany rozwój funkcjonalny produktów OSS (według Gartner'a) przedstawia Rys.12 [GARTNER2 2006];
- zmiany w technologii składowania danych – Gartner szacuje, że do 2010 roku ponad 75% spośród profesjonalnych organizacji wdroży hurtownie danych o najwyższej dostępności (ang. high-availability) w pełni odporne na awarie (ang. full fail-over).



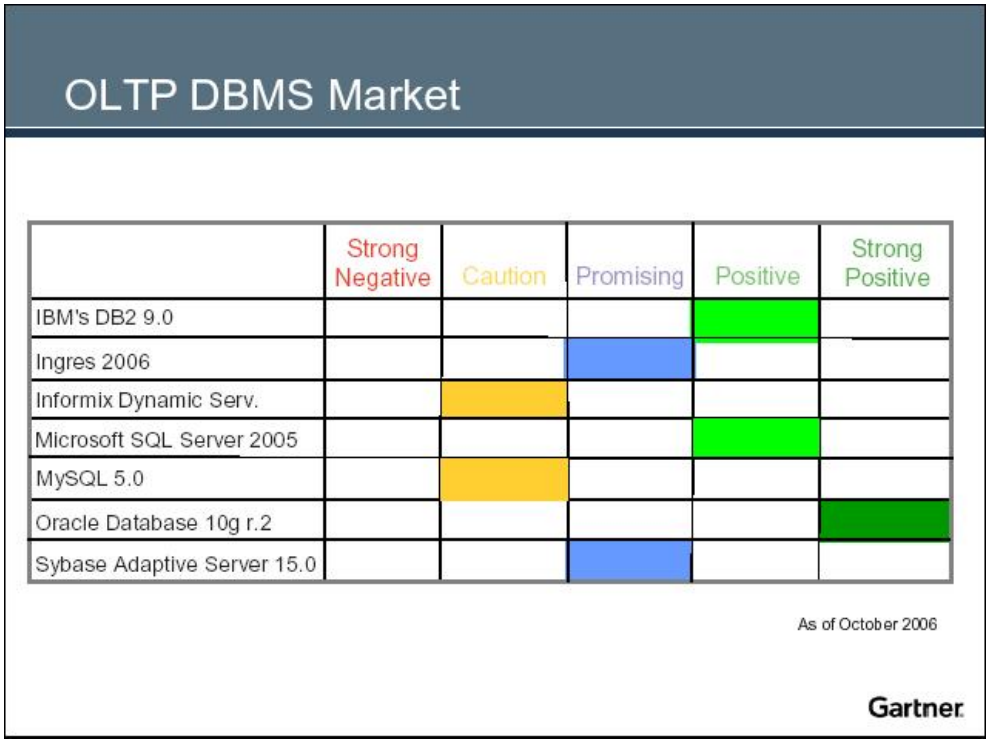
Rys.12. Rozwój funkcjonalny OSS DBMS

Interesujące są prognozy Gartnera, związane z równoległym dostępem użytkowników oraz z wielkością bazy danych w zależności od systemu operacyjnego. Prognozy przedstawiono na Rys.13 [GARTNER2 2006].



Rys. 13. Dostęp i wielkość bazy danych

Analizując badania związane z rozwojem technologicznym DBMS, Gartner wystawił oceny podsumowujące dla liderów tej branży [GARTNER2 2006]. Poniżej prezentacja wyników (Rys. 14) [GARTNER2 2006]:



Rys. 14. Ocena DBMS wg. Gartner (stan na 2006 rok)

Rozdział V Krótkie omówienie wybranych SZBD

Analiza aktualnego stanu rynku Relacyjnych Systemów Zarządzania Bazą Danych oraz perspektyw zarówno pod względem finansowym jak i technologicznym, dokonana we wcześniejszym rozdziale jednoznacznie wskazała bezwzględnych liderów omawianego segmentu IT. Kierując się powyższą analizą w dalszej części pracy zostaną omówione RDBMS następujących producentów: IBM, Microsoft oraz Oracle. Zważając jednak na coraz liczniejszy udział w rynku produktów OSS zasadne wydało się uwzględnienie również rozwiązań niekomercyjnych. Z tego względu do omawianych systemów dołączono lidera rozwiązań OSS - produkt MySQL. Jednak, aby analiza była bardziej wartościowa pod względem produktów open-source dodano jeszcze jeden produkt niekomercyjny – PostgreSQL.

5.1. DB2

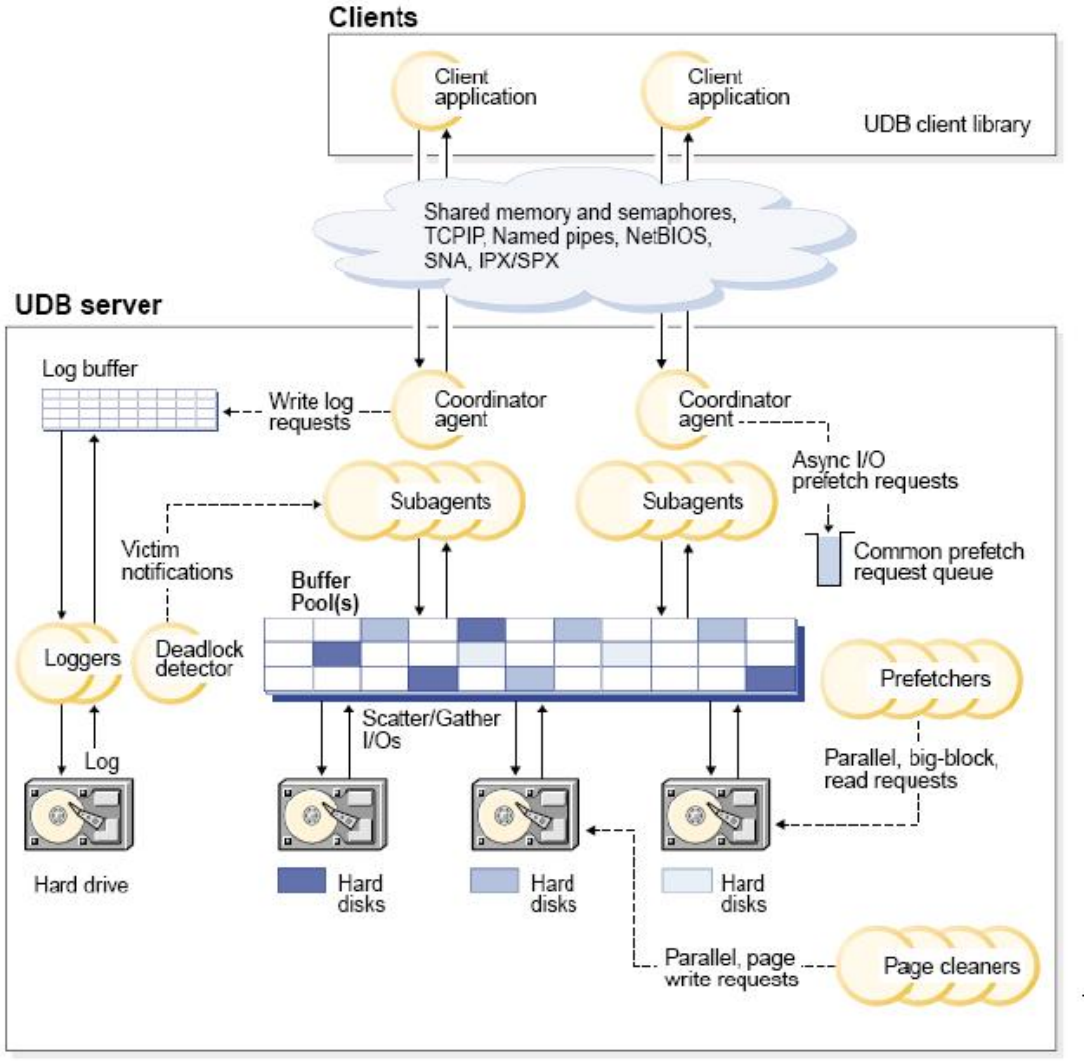
Historia bazy danych DB2 sięga połowy lat 70-tych XX wieku, kiedy Edgar Codd pracujący dla IBM, opublikował prace [CODD1 1970] dotyczącą relacyjnego modelu organizacji danych [WWWWIKI 2008]. Model relacyjny został zaimplementowany przez IBM po raz pierwszy w produkcie pod nazwą System R (System Relational) w latach 1974 – 1978 [WWWWIKI 2008].

Nazwa DB2 (Database 2 – oznacza bazę danych drugiej generacji; pierwszą był model hierarchiczny) została użyta po raz pierwszy dla RDBMS w 1983 roku, kiedy to IBM udostępnił pierwszą wersję produktu na platformę MVS (ang. Multiple Virtual Storage) [HVB 2006]. W 1987 roku została oddana pierwsza wersja przeznaczona na platformę otwartą [HVB 2006].

Kolejnym przełomem było wprowadzanie wersji 5 w 1997 roku. Wersja ta była w stanie przechowywać każdy rodzaj danych elektronicznych (audio, wideo i inne) oraz wspierała większość platform: OS/2, Windows, AIX, HP-UX czy Solaris. Ponadto od wersji 5 DB2 działa na różnych konfiguracjach sprzętowych od platform jednoprocesorowych, poprzez SMP (ang. Simmetric Multiprocessing), MPP (ang. Massively

Parallel Processors) kończąc na klastrach. W grudniu 2007 roku została udostępniona najnowsza edycja IBM DB2 v.9.5, w której wykorzystano doświadczenie z okresu modelu hierarchicznego, wprowadzając silnik hybrydowy hierarchiczno – relacyjny w celu wspomożenia obsługi dokumentów XML (ang. Extensible Markup Language).

Schemat architektury systemu DB2 przedstawia poniższy rysunek (Rys. 15) [HVB 2006]:



Rys. 15. Architektura serwera DB2

5.1.1. Edycje DB2

Baza IBM DB2 v.9.5 [WWWIBM 2008] dostępna jest w trzech edycjach komercyjnych oraz jednej edycji bezpłatnej. Wszystkie edycje są oparte o ten sam kod, natomiast różnią się ograniczeniami licencyjnymi (ograniczenia dotyczące platformy sprzętowej oraz dostępności opcjonalnych komponentów):

a) Edycje komercyjne:

- Express:
    - Maksymalna liczba procesorów w serwerze: 200 PVU (ang. Procesor Value Unit). PVU jest jednostką miary dla korów procesora (ang. core) procesora (z zakresu 30-120 PVU/kor);
    - Maksymalny rozmiar wykorzystanej pamięci: 4 GB;
    - Obsługiwane platformy: Linux, Windows, Sun Solaris (x86).
- Workgroup:
    - Maksymalna liczba procesorów w serwerze: 400 PVU;
    - Maksymalny rozmiar wykorzystanej pamięci: 16 GB;
    - Obsługiwane platformy: Linux, Windows, UNIX.
- Enterprise:
    - Maksymalna liczba procesorów w serwerze: bez ograniczeń;
    - Maksymalny rozmiar wykorzystanej pamięci: bez ograniczeń;
    - Obsługiwane platformy: Linux, z/ Linux, Windows, UNIX.
- Express-C FTL (Fixe Time License):
    - Maksymalna liczba rdzeni: 4 rdzenie na maksymalnie dwóch procesorach;
    - Maksymalny rozmiar wykorzystanej pamięci: 4 GB;
    - Obsługiwane platformy: Linux, Windows, Sun Solaris (x86).

b) Edycje bezpłatne:

- Express-C:
    - Maksymalna liczba rdzeni: 2 rdzenie;
    - Maksymalny rozmiar wykorzystanej pamięci: 2 GB;
    - Obsługiwane platformy: Linux, Windows, Sun Solaris (x86).

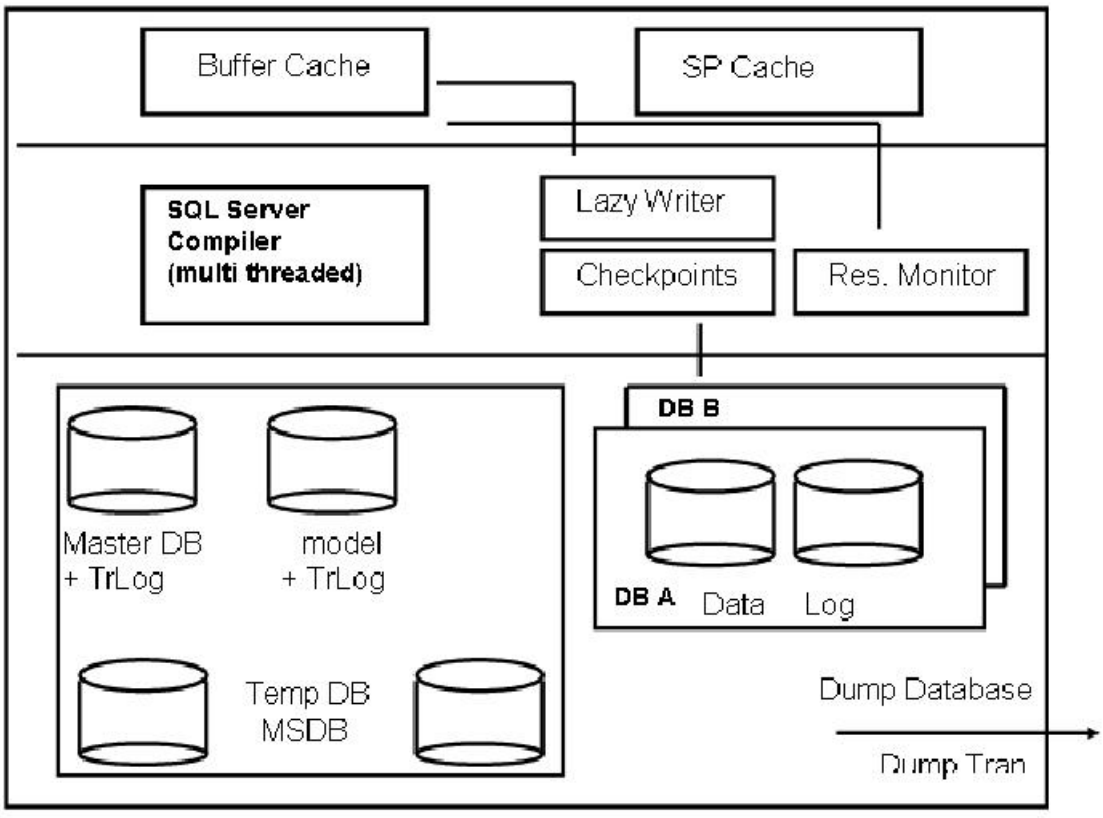
5.2. MS SQL Server

W 1989 roku Microsoft, Sybase oraz Ashton-Tate połączyły siły, aby utworzyć konkurencyjny produkt dla Oracle'a oraz DB2 [WWWWIKI 2008]. Pierwsza wersja 1.0 dedykowana dla systemu OS/2 była identyczna z Sybase SQL Server 3.0 (na platformę UNIX oraz VMS) [HVB 2006]. Pierwszą wersją, która nie zawierała żadnych naleciałości od Sybase była wersja 6.0 (1995 rok). Była to zarazem pierwsza wersja zaprojektowana dla systemu NT

(ang. New Technology). Po rozwiązaniu umów, Sybase zmienił nazwę produktu na Adaptive Server Enterprise. Natomiast wersja 7.0 (1998 rok), w której cały kod został przepisany z oryginalnego kodu Sybase zawierał pierwsze profesjonalne GUI (ang. Graphical User Interfeace).

Kolejna wersja to wersja 8.0, czyli MS SQL Server 2000, która była pierwszą edycją uruchamianą w architekturze IA-64. Kolejną wersją był MS SQL 2005 (rok 2005), w którym poprawiono wydajność, dodano narzędzia IDE (ang. Integrated Development Environment), Reporting Server, narzędzia ETL (ang. Extract, Transform, Load, przeznaczone dla hurtowni danych), OLAP, obsługę XML wraz ze schematami XSD oraz obsługą XQuery, dostęp poprzez web services [WWWWIKI 2008].

Schemat architektury systemu SQL Server przedstawia poniższy rysunek (Rys. 16) [HVB 2006]:



Rys. 16. Architektura serwera MS SQL Server

5.2.1. Edycje MS SQL Server

Baza MS SQL Server 2005 dostępna jest w trzech edycjach komercyjnych oraz jednej edycji bezpłatnej [WWWMS 2008]. Wszystkie edycje są oparte o ten sam kod, natomiast różnią się ograniczeniami licencyjnymi (ograniczenia dotyczące platformy sprzętowej oraz dostępności opcjonalnych komponentów):

a) Edycje komercyjne:
   - Workgroup:
     - Maksymalna liczba procesorów w serwerze: 2 CPU;
     - Maksymalny rozmiar wykorzystanej pamięci: 3 GB;
     - Obsługiwane platformy: Windows [1, 2, 3, 4, 5, 6, 7, 8, 9].
   - Standard:
     - Maksymalna liczba procesorów w serwerze: 4 CPU;
     - Maksymalny rozmiar wykorzystanej pamięci: bez ograniczeń;
     - Obsługiwane platformy: Windows, wsparcie dla 64bit [1, 2, 3, 4, 5, 6, 7, 8, 9].
   - Enterprise:
     - Maksymalna liczba procesorów w serwerze: bez ograniczeń;
     - Maksymalny rozmiar wykorzystanej pamięci: bez ograniczeń;
     - Obsługiwane platformy: Windows, wsparcie dla 64bit [2, 3, 4, 6, 7, 8].

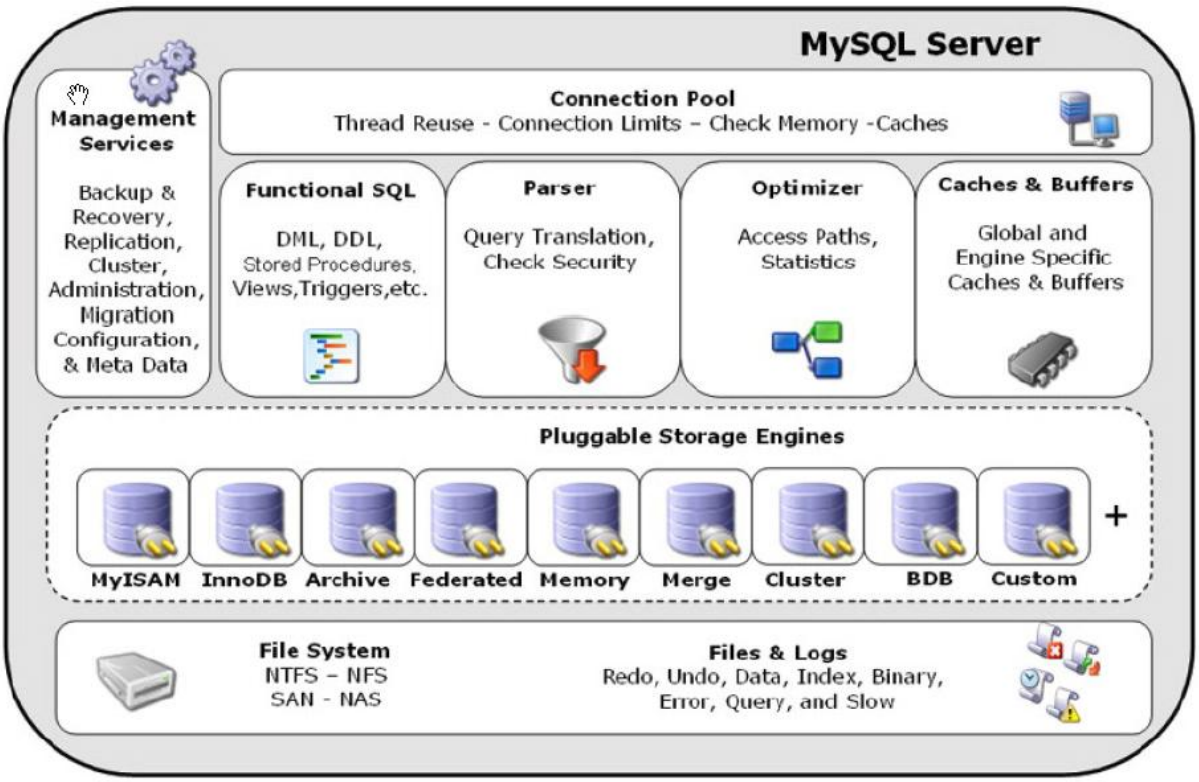
b) Edycje bezpłatne:
   - Express:
     - Maksymalna liczba procesorów w serwerze: 1 CPU;
     - Maksymalny rozmiar wykorzystanej pamięci: 1 GB;
     - Rozmiar bazy danych: 4 GB;
     - Obsługiwane platformy: Windows 64bit [1, 2, 3, 4, 5, 6, 7, 8, 9].

[1] Windows 2000 Professional Edition SP4
[2] Windows 2000 Server SP4
[3] Windows 2000 Advanced Server SP4
[4] Windows 2000 Datacenter Edition SP4
[5] Windows XP Professional Edition SP2
[6] Windows Server 2003 Server SP1
[7] Windows Server 2003 Enterprise Edition SP1
[8] Windows Server 2003 Datacenter Edition SP1
[9] Windows Vista Business Edition / Enterprise Edition

5.3. MySQL

Pierwsza opublikowana wersja systemu MySQL pojawiła się w maju 1995 roku [WWWMYSQL 2008]. Trzy lata później w 1998 roku udostępniono wersję na Windows 95 oraz NT. Następną wersję 3.21 beta opublikowana w 2000 roku. Dwa lata później w 2002 roku pojawiła się kolejna wersja beta (4.0), która rozszerzyła funkcjonalność o unie (ang. union). Kolejną istotną wersją była wersja 4.1 (2004 rok), w której wprowadzono: R-drzewa oraz B-drzewa, a także podzapytania. W kolejnej wersji 5.0 (2005 rok) zaimplementowano kursory, procedury składowane, triggery, widoki. Aktualnie, od 2005 roku powstaje wersja 5.1, która ma poszerzyć system MySQL o schedule, partycjonowanie, replikację bazującą na wierszach oraz tabele logowania. Produkt MySQL Server jest dostępny na licencji GNU General Public License.

Schemat architektury systemu MySQL przedstawia poniższy rysunek (Rys. 17) [HVB 2006]:



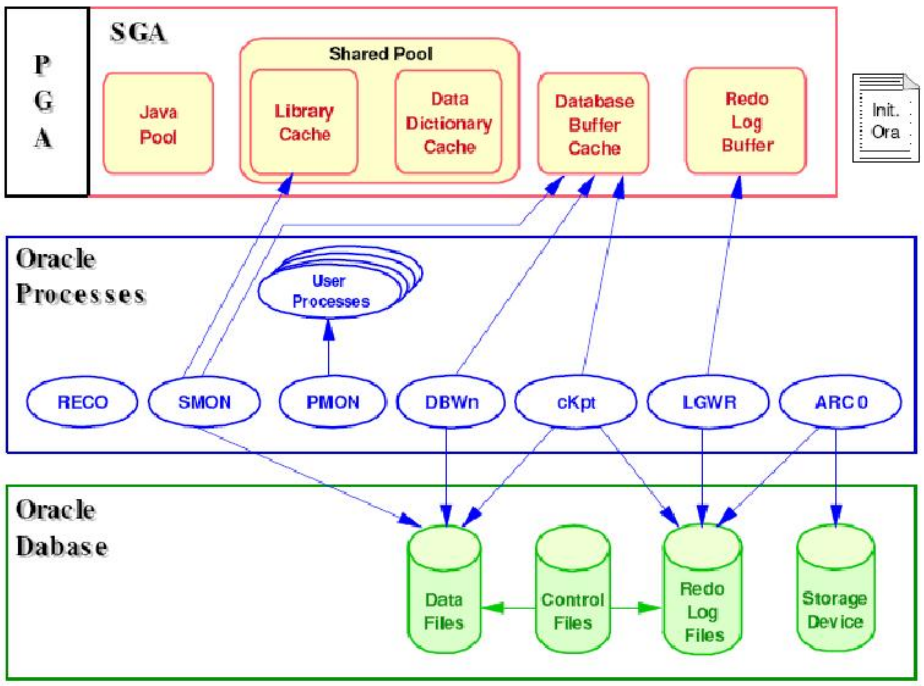
Rys. 17. Architektura serwera MySQL

5.4. Oracle

Historia systemu Oracle sięga 1975 roku, kiedy został zaimplementowany (w przeciągu niecałego roku) RDBMS pod nazwą Oracle 2 [HVB 2006]. W 1983 roku została przepisana wersja poprzednia w języku C, i tym samy wygenerowano wersję 3, niezbyt stabilną. Międzyczasie E. Codd opublikował 12 postulatów RDBMS. Na przełomie 1985 i 1986 roku powstała wersja 5.1, w której prekompilator został zaimplementowany w języku Cobol oraz Fortran. W wersji 6 (1988-1991 rok) zaimplementowano blokady na poziomie wierszy oraz backup on-line. W wersji 7 (1992 rok) pojawiły się triggery, role, procedury storowalne.

W wersji 8 (1997 rok) z ważniejszych funkcjonalności Oracle dostarczył: optymalizator bazujący na kosztach, transakcje rozproszone, pytania równoległe, partycje, widoki materializowane, zaawansowane indeksy, wsparcie dla technologii zorientowanej obiektowo, wsparcie dla platformy Windows. Kolejna wersja 8i (1998 rok) wprowadziła przede wszystkim natywne środowisko Java Runtime, zarządzanie zawartością multimedialną, wsparcie dla XML, wsparcie dla platformy Linux. W 2001 roku wersja 9i wprowadziła technologie klastrową (Oracle Cluster Technology). Wersja 10g (2004 rok) głównie wprowadziła narzędzia wspierające dla systemu Oracle [HVB 2006].

Schemat architektury systemu Oracle przedstawia poniższy rysunek (Rys. 18) [HVB 2006]:



Rys. 18. Architektura serwera Oracle

5.4.1. Edycje Oracle

Baza danych Oracle dostępna jest w trzech wersjach komercyjnych oraz w jednej wersji bezpłatnej [WWWORACLE 2008]. Wersje komercyjne to wersje 11g, natomiast wersja bezpłatna jest wersją 10g:

a) Edycje komercyjne:
   - Standard Edition One:
     * Maksymalna liczba procesorów w serwerze: 2 rdzenie;
     * Maksymalny rozmiar wykorzystanej pamięci: bez ograniczeń;
     * Obsługiwane platformy: Linux, Windows, Unix, wsparcie dla 64bit.
   - Standard Edition:
     * Maksymalna liczba procesorów w serwerze: 4 rdzenie;
     * Maksymalny rozmiar wykorzystanej pamięci: bez ograniczeń;
     * Obsługiwane platformy: Linux, Windows, UNIX, wsparcie dla 64bit.
   - Enterprise Edition:
     * Maksymalna liczba procesorów w serwerze: bez ograniczeń;
     * Maksymalny rozmiar wykorzystanej pamięci: bez ograniczeń;
     * Obsługiwane platformy: Linux, Windows, UNIX, wsparcie dla 64bit.

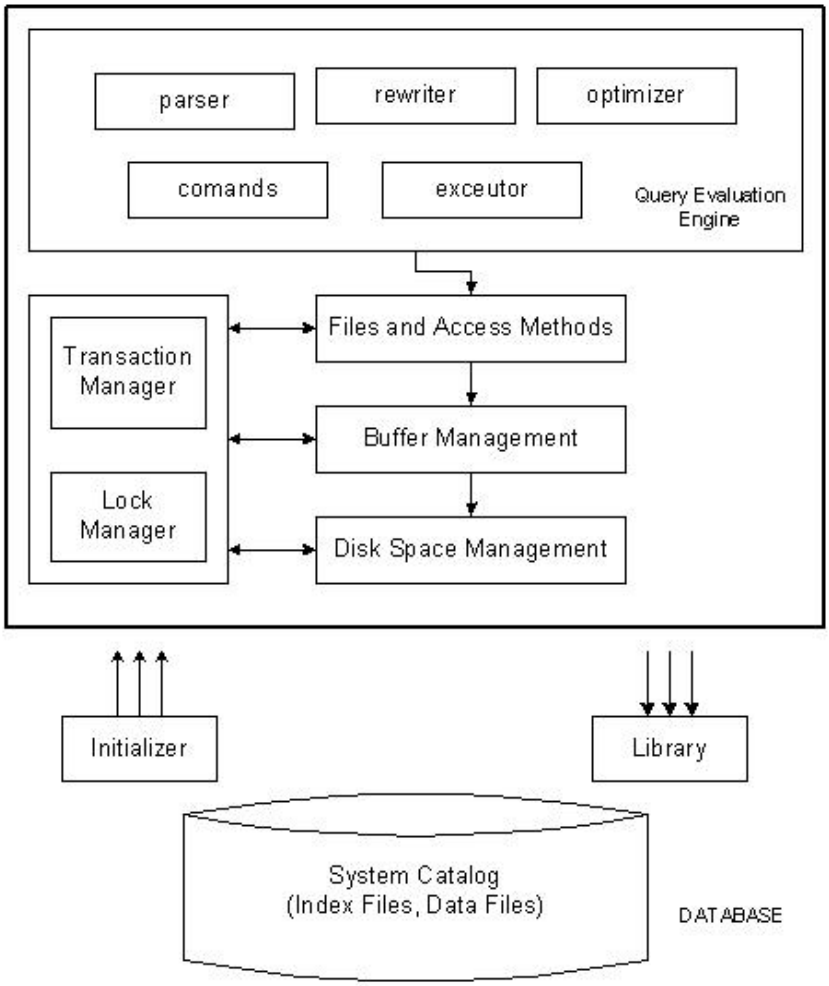
b) Edycje bezpłatne:
   - Express Edition 10g:
     * Maksymalna liczba rdzeni: 1 CPU;
     * Maksymalny rozmiar wykorzystanej pamięci: 1 GB;
     * Wielkość bazy danych: 4 GB;
     * Obsługiwane platformy: Linux, Windows.

5.5. PostgreSQL

Historia PostgreSQL bierze swój początek w 1973 roku, kiedy dr Michale Stonebraker z Eugene Wong rozpoczął badania nad relacyjnymi systemami baz danych, co dało początek projektowi Ingres na uniwersytecie Kalifornijskim w Berkley [WWWWIKI 2008]. W 1985 roku projekt zmienił nazwę na Postgres. W 1989 roku opublikowano pierwszą wersję systemu, z funkcjonalnością reguł, procedur, typów, elementów obiektowych oraz z zaawansowanym językiem zapytań POSTQUEL (ang. Postgres Query Language). Kolejnie w 1990 roku opublikowano wersję drugą, w której przepisano system reguł. W 1991 roku udostępniono wersję 3 wraz z poprawionym silnikiem zapytań. Dopiero w 1994 roku dodano interpreter zapytań SQL i tym samym zastąpiono język POSTQUEL, a produkt udostępniono pod nazwą Postgres95.

W 1996 roku dalsze prace nad projektem podjęła społeczność Open Source, zmieniając nazwę projektu na PostgreSQL i powołując organizację do koordynacji rozwoju projektu pod nazwą PostgreSQL Global Development Group. W kolejnych latach firmy takie jak Pervasive Software, EnterpriseDn czy Sun Microsystem zaczęły oficjalnie wspierać produkt.

PostgreSQL wspiera wiele platform min: Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) oraz Windows. PostgreSQL wraz z kodem źródłowym jest dostępny na licencji open source. W lutym 2008 roku została opublikowana wersja 8.3 [WWWPOSTGRE 2008] systemu PostgreSQL.
Schemat architektury systemu PostgreSQL przedstawia poniższy rysunek (Rys. 19) [WWWSWAG 2008]:



Rys. 19. Architektura serwera PostgreSQL

5.6. Porównanie wybranych właściwości omawianych SZBD

Ponieważ celem pracy jest porównanie wybranych Systemów Zarządzania Bazą Danych, a nie ich prezentacja, po ogólnym omówieniu produktów, zostanie przedstawione tabelaryczne zestawienie wybranych cech RDBMS. Poniższe zestawienia zostały wykonane dla następujących wersji DBMS (chyba, że zaznaczono inaczej) [WWWWIKI 2008]:

- DB2 9.5;
- MS SQL Server 9.00.3042 (2005 z SP2);
- MySQL 5.0.67;
- Oracle 11g (Release 1);
- PostgreSQL 8.3.3.

Pierwsze zestawienie dotyczy wsparcia dla systemów operacyjnych (Tab.5) [WWWWIKI 2008]:

	DB2	MS SQL Server	MySQL	Oracle	PostgreSQL
BSD	NIE	NIE	TAK	NIE	TAK
Linux	TAK	NIE	TAK	TAK	TAK
Mac OS X	NIE	NIE	TAK	TAK	TAK
UNIX	TAK	NIE	TAK	TAK	TAK
Windows	TAK	TAK	TAK	TAK	TAK
z/OS	TAK	NIE	NIE	TAK	NIE

Tab. 5. Wsparcie dla systemów operacyjnych

Kolejne zestawienie dotyczy podstawowych cech Relacyjnych Systemów Zarządzania Bazą Danych (Tab. 6) [WWWWIKI 2008]:

	DB2	MS SQL Server	MySQL	Oracle	PostgreSQL
ACID	TAK	TAK	TAK	TAK	TAK
Integralność referencyjna	TAK	TAK	TAK	TAK	TAK
Obsługa transakcji	TAK	TAK	TAK	TAK	TAK
Obsługa UNICODE	TAK	TAK	Częściowo	TAK	TAK
Intersect	TAK	TAK	NIE	TAK	TAK
Except	TAK	TAK	NIE	TAK	TAK
Inner join	TAK	TAK	TAK	TAK	TAK
Outer join	TAK	TAK	TAK	TAK	TAK
Merge join	TAK	TAK	TAK	TAK	TAK
Full join	TAK	TAK	NIE (v.5.0.18)	TAK	TAK
Natura join	NIE (v.9.1)	NIE	TAK	TAK	TAK
Cross join	NIE (v.9.1)	TAK	TAK	TAK	TAK
Domeny danych	Przez CHECK CONSTRAINS	TAK	NIE	TAK	TAK
Kursory	TAK	TAK	TAK	TAK	TAK
Wyzwalacze	TAK	TAK	TAK	TAK	TAK
Funkcje	TAK	TAK	TAK	TAK	TAK
Procedury	TAK	TAK	TAK	TAK	TAK

Tab. 6. Podstawowe cechy RDBMS

Następnym istotnym zestawieniem są ograniczenia omawianych systemów (Tab. 7) [WWWWIKI 2008]:

	DB2	**MS SQL Server**	**MySQL**	**Oracle**	**PostgreSQL**
Maksymalna wielkość bazy danych	512 TB	524 TB	Bez limitów	Bez limitów (4 GB * wielkość bloku / przestrzeń tabel)	Bez limitów
Maksymalny rozmiar tabeli	512 TB	524 TB	2 GB (win32 FAT) do 16 TB (Solaris)	4 GB * wielkość bloku	32 TB
Maksymalny rozmiar wiersza	32, 677 B	8060 B	64 KB	Bez limitów	1,6 TB
Maks. ilość kolumn/wiersz	1012	1024	3398	1000	250-1600 w zależności od typu
Maks. rozmiar BLOB/CLOB	2 GB	2 GB	4 GB	4 GB (lub maksymalna wielkość pliku danych dla platformy)	1 GB
Maks. rozm. CHAR	32 KB	8000 B	64 KB	4000 B	1 GB
Maks. rozm. NUMBER	64 bity	64 bity	64 bity	126 bits	Bez limitów

Tab. 7. Ograniczenia omawianych RDBMS

Ostatnie zestawienie zawiera porównanie wybranych zaawansowanych technologii w systemach relacyjnych (Tab. 8) [WWWWIKI 2008]:

	DB2	**MS SQL Server**	**MySQL**	**Oracle**	**PostgreSQL**
Tabele tymczasowe	TAK	TAK	TAK	TAK	TAK
Zmaterializowane widoki	TAK	TAK	NIE	TAK	NIE
Indeksy R-/R+ tree	NIE	?	Tylko tabela MyISAM	TAK	TAK
Indeksy haszowane	?	Nieklastrowane	Tylko wybrane tabele syst.	Klastrowane tabele	TAK

Indeksy wyrażeniowe	TAK	TAK	NIE	TAK	TAK
Indeksy częściowe	NIE	TAK	NIE	TAK	TAK
Indeksy rewersyjne	TAK	TAK	NIE	TAK	TAK
Indeksy bitmapowe	TAK	NIE	NIE	TAK	TAK
Indeksy GiST	NIE	NIE	NIE	NIE	TAK
Indeksy GIN	NIE	NIE	NIE	NIE	TAK
Range partitioning	TAK	TAK	TAK	TAK	TAK
Hash partitioning	TAK	NIE	TAK	TAK	TAK
Composite partitioning (Range+Hash)	TAK	NIE	TAK	TAK	TAK
List partitioning	TAK	NIE	TAK	TAK	TAK
Obsługa typu BOOLEAN	NIE	NIE	TAK	NIE	TAK

Tab. 8. Zaawansowane technologie w omawianych RDBMS

Rozdział VI Wymagania wobec SZBD

W poprzednich rozdziałach zostały omówione podstawowe funkcje Systemów Zarządzania Bazą Danych, trendy związane z rozwojem technologii oraz zostało przedstawione zestawienie poszczególnych własności. Zebrane informacje (głównie z dokumentacji produktów) pozwalają na wybór RSZBD na podstawie takich kryteriów jak:

- popularność systemu (a zatem i jego przetestowanie w warunkach produkcyjnych);
- dostępne narzędzia wspomagające (administracyjne, deweloperskie);
- wspierana platforma;
- zastosowane zaawansowane technologie baz danych;
- ograniczenia;
- cena produktów.

Te wszystkie kryteria można jasno sformułować i podejmować decyzję na podstawie specyfikacji systemowej. Nie zawierają one jednak tak istotnych aspektów (niezbędnych w SZBD eksplorowanych przez duże systemy informatyczne) jak wydajność serwera bazy danych (rozumiana na przykład w czasie odpowiedzi na zadane pytanie). Na podstawie dostarczonych przez producentów dokumentacji nie jesteśmy w stanie stwierdzić, który z oferowanych przez rynek produktów będzie bardziej wydajny dla projektowanego (bądź migrowanego) systemu.

Oczywiście zrozumiałe jest, iż na przykład firmy deweloperskie w ramach swojej działalności nie korzystają z pełnego asortymentu dostępnego na rynku. Raczej starają się standaryzować swoje rozwiązania i dążyć do ujednolicania technologii. Czy jednak w sytuacji konieczności zmiany systemu zarządzania bazą danych (problemy wydajnościowe, zmiana platformy sprzętowej bądź programowej, wdrażanie polityki ujednolicania technologii) można wesprzeć się literaturą bądź innymi opracowaniami?

6.1. Benchmarki dostępne na rynku

Na rynku są dostępne benchmarki mierzące wydajność systemów bazodanowych. Najczęściej omawianym w literaturze produktem jest zestaw benchmarków opracowanych przez TPC (Transaction Processing Performance Council) [KOZ 2007]. Oprócz wymienionego wyżej benchmarku (a dokładniej rodziny benchmarków), na rynku jest kilka istotniejszych, specyficznych benchmarków bazodanowych, które zostaną omówione w poniższych podrozdziałach. Warto podkreślić [KOZ 2007], że za wyjątkiem projektów z rodziny wolnego i otwartego oprogramowania, nie opublikowano żadnych kodów źródłowych. Dla większości nie opublikowano również szczegółów zastosowanej w procesie testowania metodologii. Z kolei dla projektów, których metodologia badań została przedstawiona, nie ujawniono wyników badań.

6.1.1. Transaction Processing Performance Council

TPC jest organizacją non-profit powołaną przez korporacje w celu definiowania benchmarków bazodanowych jak i przetwarzania transakcyjnego oraz w celu rozpowszechniania obiektywnych i prawdziwych danych wydajnościowych dla przemysłu. Opracowane przez TPC benchmarki [WWWTPC 2008]:

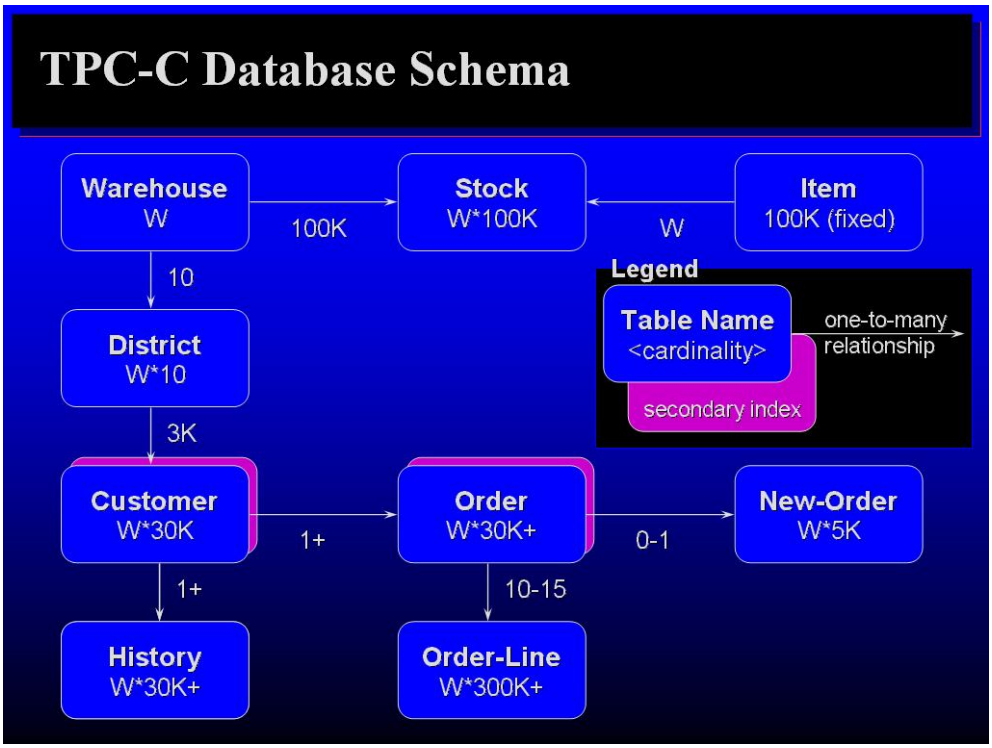
- TPC-C – standardowy benchmark dla systemów realizujących OLTP (ang. On-Line Transaction Processing). Składa się z dużej ilości prostych zapytań. Jest mieszanką operacji zapisu, odczytu, usuwania oraz aktualizacji. Jednocześnie jest wykonywanych dużo transakcji. Jest reprezentatywny dla większego obszaru zastosowań. Symuluje typową pracę systemu bazodanowego w architekturze klient-serwer;
- TPC-E – dotyczy wydajności przetwarzania złożonych transakcji OLTP, a test wykorzystuje scenariusz typowej platformy informatycznej obsługującej firmę brokerską. Dotyczy węższej klasy zastosowań biznesowych. Głównie składa się z wyspecjalizowanych transakcji i operacji;
- TPC-App – dotyczą aplikacji e-commerce, symulując działanie transakcyjnego serwera aplikacyjnego oraz jego usług sieciowych;

- TPC-H – dotyczą systemów wspomagania zarządzania, składającym się z kompletu tzw. zapytań ad-hoc (zadawanych na bieżąco podczas realizacji innych algorytmów) i równoległych modyfikacji danych;
- benchmarki już wycofane z użycia: TPC-A, TPC-B, TPC-D, TPC-R, TPC-W.

Metryki benchmarku TPC-C:

- tpm-C – ilość nowo przyjętych zamówień na minutę, przy jednoczesnej realizacji przyjętych wcześniej;
- $/tpm-C – stosunek ceny do wydajności.

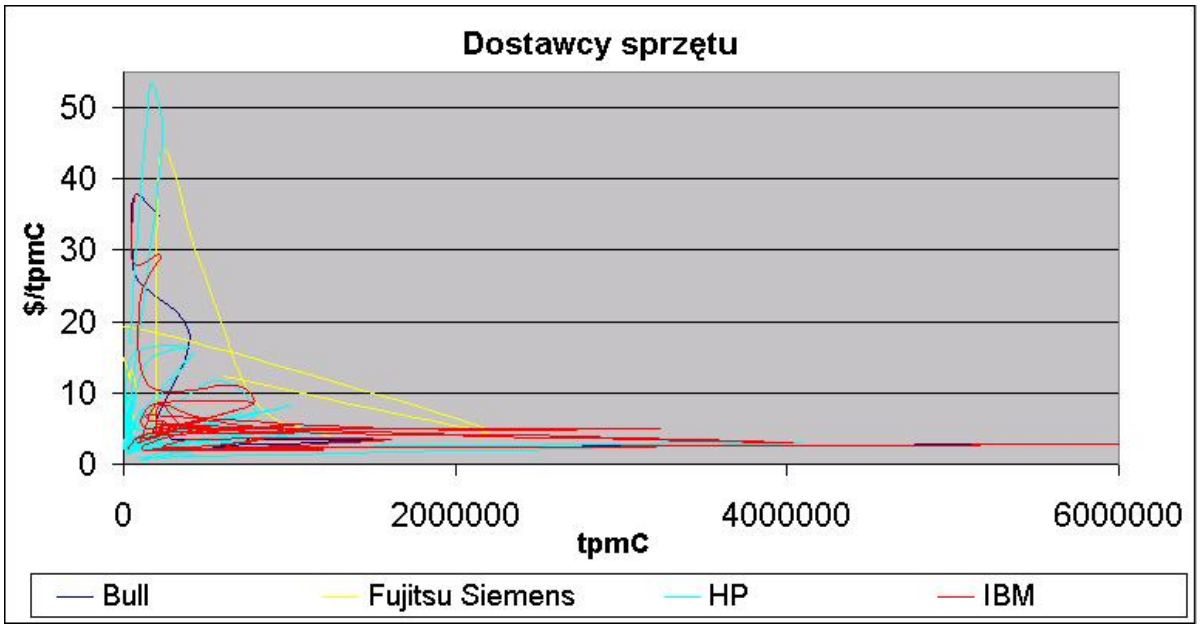
Schemat bazy danych wykorzystywanej w benchmarku TPC-C [WWWTPC 2008]:



Rys. 20. Schemat bazy danych dla TPC-C

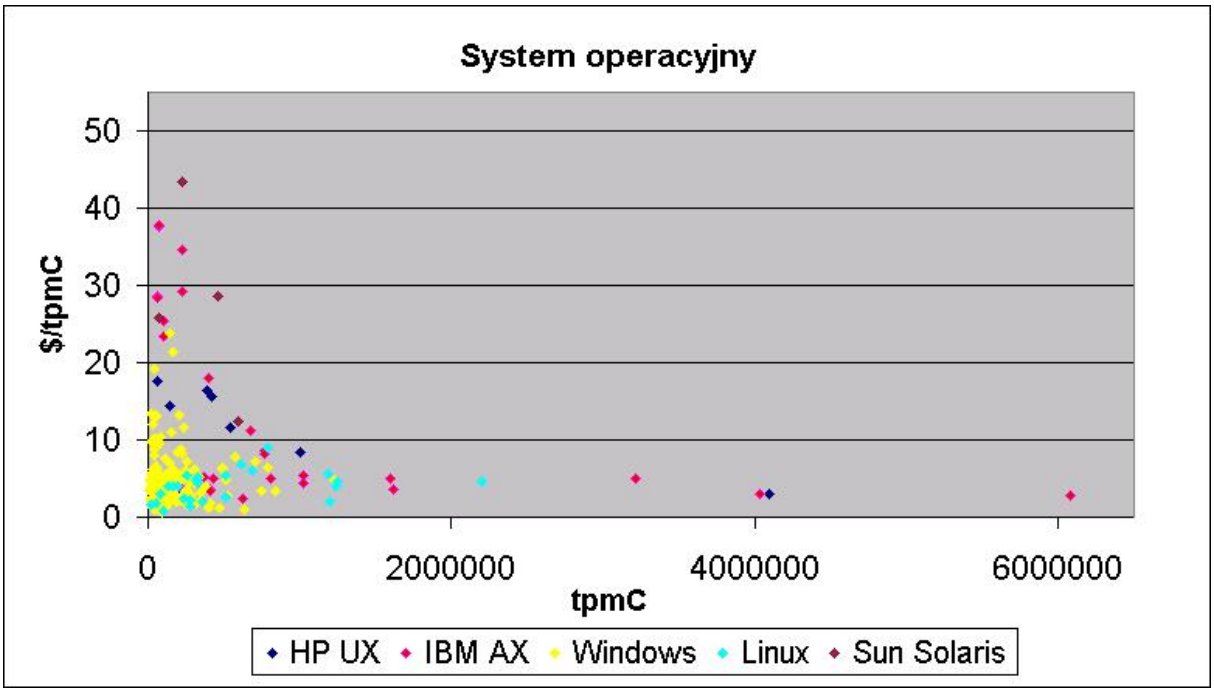
TPC publikuje na swojej witrynie [WWWTPC 2008] aktualne rezultaty przeprowadzonych testów. Publikowane wyniki można zinterpretować na kilka sposobów. Poniżej zostały zaprezentowane dane z dnia 19-08-2008 roku.

Po pierwsze można dokonać porównania dostawców sprzętu (Rys.21) [WWWTPC 2008] pod względem wartości metryk benchmarku TPC-C:



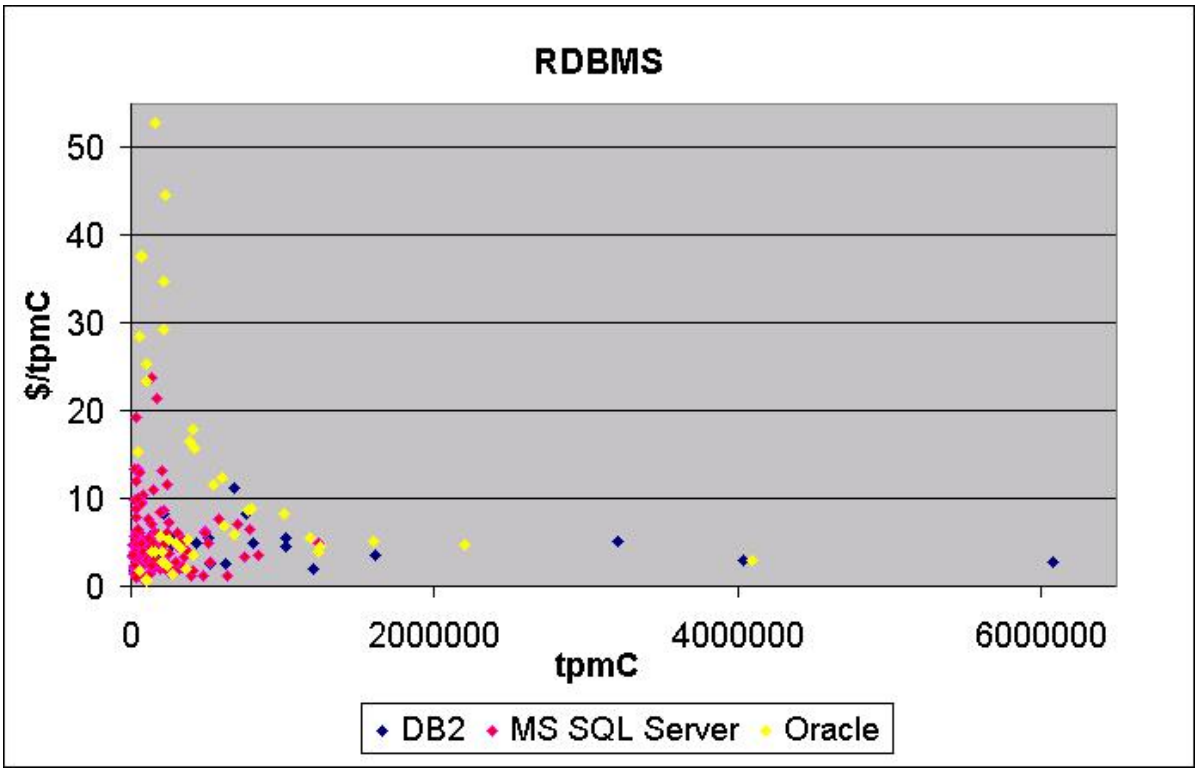
Rys. 21. Ranking dostawców sprzętu na podstawie TPC-C

Kolejnym dość wartościowym zestawieniem jest porównanie wydajności transakcji pod kątem systemu operacyjnego (Rys. 22) [WWWTPC 2008]:



Rys. 22. Ranking systemów operacyjnych na podstawie TPC-C

Ostatnią i chyba najbardziej interesującą analizą wyników benchmarku TPC-C jest zestawienie dostawców relacyjnych systemów zarządzania bazą danych (Rys. 23) [WWWTPC 2008]:



Rys. 23. Ranking RDBMS na podstawie TPC-C

Z przedstawionych wyżej analiz można wywnioskować, iż najbardziej wydajną konfiguracją według metryki tpmC benchmarku TPC-C była:

- platforma sprzętowa: IBM;
- system operacyjny IBM AX;
- baza danych DB2.

W publikowanych przez TPC wynikach testów można dowiedzieć się jeszcze następujących informacji o najwydajniejszej instalacji:

- nazwa systemu: IBM Power 595 Server Model 9119-FHA;
- tpmC: 6085166;
- $/tpmC: 2,81;
- całkowity koszt systemu: 17.111.788 USD;
- baza danych: IBM DB2 9.5;
- system operacyjny: IBM AIX 5L V5.3;

- ilość CPU: 32;
- typ CPU: Intel Pentium III Xeon – 900 MHz.

6.1.2. Open Source Development Labs Database Test Suite

OSDL jest organizacją wspierającą rozwój społeczności związanej z systemem Linux [KOZ 2007]. Jednym z opracowanych produktów przez OSDL [KOZ 2007] jest zbiór czterech benchmarków, które zostały opracowane na wzór testów TPC:

- DBT-1;
- DBT-2;
- DBT-3;
- DBT-4.

6.1.3. Network Database Benchmark

Network Database Benchmark jest narzędziem służącym do pomiaru wydajności baz HLR (ang. Home Location Register). Bazy HLR to centralne bazy danych, które zawierają szczegóły każdego abonenta sieci telefonii komórkowej, autoryzowanego do skorzystania z usług sieci GSM (ang. Globar System for Mobile Communications). Baza HLR przechowuje szczegóły dotyczące każdej karty SIM (ang. Subscriber Identity Module) wydanej do użytku przez operatora sieci komórkowej.

Głównym celem projektu [WWWNDB 2008] jest wspomożenie producentów systemów bazodanowych w ulepszaniu ich produktów wymagającym środowisku telekomunikacyjnym. Projekt został zapoczątkowany przez Nokia Corporation. Aktualna wersja benchmarku jest zgodna z bazą danych MySQL, ale jest planowane wsparcie dla innych systemów bazodanowych.

Charakterystyka bazy danych HLR:

- czas odpowiedzi dla transakcji odczytu nie powinien przekraczać 5ms;
- czas transakcji aktualizacji danych nie powinien przekraczać 100ms;
- baza danych powinna się skalować dynamicznie;
- powinna istnieć możliwość dodania nowego węzła (serwera) bez przerwania pracy systemu;
- system powinien zapewniać 99,999% dostępność;

- operacje dodanie/usunięcia abonenta zostały pominięte, ponieważ nie wymagają tak dużych wydajności;
- długie transakcje oraz rozbudowane złączenia (ang. *join*) zostały również pominięte, ponieważ są wykorzystywane na administracyjnych bazach danych.

Kod źródłowy oraz dokumentacje można pobrać ze strony projektu (kod jest udostępniany na licencji GPL).

Benchmarkiem opartym o Network Database Benchmark jest Telecom One Benchmark (TM1) [WWWTM1 2008] również udostępniany z kodem źródłowym na licencji GNU.

6.1.4. Open Source Database Benchmark

OSDB powstał z mniejszego projektu w Compaq Computer Corporation [KOZ 2007], w celu oszacowania przepustowości i mocy przetwarzania systemu GNU Linux/Alpha. Zbudowany jest na bazie AS3AP (ANSI SQL Standard Scalable and Portale), czyli testu systemu przeznaczonego dla relacyjnych systemów baz danych, stworzonego przez D. Bitton'a i C. Turbyfill'a. Tym, co odróżnia go od AS3AP jest raportowanie wielu miar badanego systemu, nie ograniczając się jedynie do rozmiaru maksymalnej bazy danych, jaka może być użyta w celu ukończenia kompletu testów AS3AP w mniej niż 12 godzin. Zaletą OSDB jest możliwość pracy z niekompletnymi implementacjami języka SQL lub implementacjami jego pozbawionych.

6.1.5. OLTP-2

OLTP-2 jest przykładem wewnętrznego benchmarku bazodanowego opracowanego przez producenta sprzętu Fujitsu-Siemens Computers. Test OLTP-2 jest zbliżony do benchmarku TPC-E, i również ma za zadanie pomiar wydajności systemów OLTP. To, co odróżnia OLTP-2 od TPC-E, to brak standaryzacji. Ponadto, OLTP-2 nie zawiera informacji o kosztach systemu (bądź cenie przypadającej na transakcję), ponieważ jego celem jest jedynie pomiar wydajności systemu oraz umożliwienie pomiarów skalowalnych serwerów rodziny PRIMERGY. Benchmark OLTP-2 nie nadaje się do porównań konfiguracji

sprzętowych innych producentów (jest dedykowany do serwerów produkcji Fujitsu – Siemens).

6.2. Zdefiniowanie kryteriów porównawczych

Przytoczony powyżej zestaw testów zaproponowany przez TPC z pewnością jest profesjonalnym narzędziem dostarczającym informacji statystycznych dotyczących przeciętnej konfiguracji klient-serwer serwera bazodanowego. Z pewnością publikowane wyniki testów przez TPC są pomocne w następujących przypadkach:

- wybór platformy sprzętowej;
- wybór platformy programowej;
- wybór DBMS (dla standardowych zastosowań).

Co jednak w przypadku, gdy nasza baza danych nie jest „standardowa”? Co w przypadku, kiedy chcemy wybrać najlepszą konfigurację dla naszego środowiska (które nie jest „standardowe”). Pod pojęciem niestandardowości można rozumieć:

- małą ilość danych, ale bardzo dużo współbieżnych transakcji;
- dużą ilość danych i specyficzne zapytania;
- bardzo mało bardzo skomplikowanych zapytań.

Niestety nie są dostępne takie publikacje. Co więcej nie ma metodologii, jaką należałoby się posłużyć, aby w prosty sposób wybrać możliwie dobre rozwiązanie dla specyficznego środowiska. Również nie są znane autorowi niniejszej pracy, żadne statystyki, które ilustrowałyby wpływ ilości danych czy wpływ ilości użytkowników korzystających z bazy danych na wydajność RDBMS. Wydaje się zatem zasadne zdefiniowanie testów, które pozwoliłyby na zbadanie:

- bazy danych o zadanym schemacie i zdefiniowanej ilości danych w różnych środowiskach bazodanowych;
- wydajności zdefiniowanych przez użytkownika zapytań;
- zachowania RDBMS ze względu na:
    * zmieniającą się ilość danych;
    * zmieniającą się liczbę użytkowników.

Rozdział VII Narzędzia do monitorowania środowiska SZBD

Systemy zarządzania bazą danych w większości przypadków oferują narzędzia służące do monitoringu stanu systemu. Jednak w przypadku, gdy chcemy porównywać różnorodne SZBD, należy posłużyć się uniwersalnym narzędziem, który umożliwi pomiary interesujących wartości zadanych parametrów w sposób jednolity (tzn. niezależny od DBMS).

7.1. Narzędzia systemowe środowiska Linux

Systemy operacyjne z rodziny Linux posiadają zestaw standardowych narzędzi umożliwiających pomiar środowiska. Do istotnych można zaliczyć:

- top – program konsoli wyświetlający odświeżaną samoczynnie listę procesów aktualnie pracujących w systemie. Prezentowane dane zawierają takie informacje jak: identyfikator procesu, nazwę procesu, użycie procesora, użycie pamięci, czas aktywności [MCCARTY 2002];
- sar – System Activity Report (SAR) to paczka zawierająca zestaw skryptów oraz programów służących do monitorowania stanu systemu, oraz tworzenia raportów na podstawie otrzymanych danych. Przeważnie można ją odnaleźć w pakiecie „sysstat package" [WWWLINUX 2008];

  Przykład użycia polecenia: > *sar 1 10.*

  W rezultacie na bieżąco użytkownik dostanie odpowiedź na standardowe wyjście z podstawowymi parametrami systemu: czas pomiaru, ilość użytkowników, czas procesora (odrębne statystyki dla pracy w trybie jądra oraz w trybie użytkownika), operacje I/O (w większości wartości prezentowane są w procentach).

  Bardziej zaawansowane parametry narzędzia sar:

  * -d – aktywność dysku;
  * -g – stronicowanie;
  * -k – pamięć jądra;
  * -r – stan pamięci;
  * -u – użycie procesora;
  * -w – przełączanie procesów.

- iostat – to narzędzie do raportowania stanu jednostki centralnej (CPU) oraz statystyk dotyczących operacji wejścia / wyjścia dla urządzeń oraz partycji [WWWLINUX 2008]. Oferuje dużą ilość statystyk operacji I/O, między innymi:
    * Blk_read/s (Blk_wrtn/s) - średnia ilość danych odczytanych (zapisanych) z urządzenia wyrażona w ilości przeczytanych (zapisanych) bloków na sekundę;
    * Blk_read (Blk_wrtn) – całkowita liczba przeczytanych (zapisanych) bloków;
    * await – średni czas (w milisekundach) dla operacji I/O oczekujących na obsługę przez urządzenie. Do czasu wlicza się zarówno przebywanie w kolejce jak i czas poświecony na obsługę kolejki.
- prostat – narzędzie udostępnia statystyki aktywnych procesów;
- mpstat – narzędzie udostępnia statystyki procesorów;
- vmstat – narzędzie udostępnia statystyki pamięci wirtualnej.

7.2. Narzędzia systemowe środowiska Windows

System Microsoft Windows wśród narzędzi administracyjnych posiada narzędzie *Wydajność* (wersja polska systemu) bądź *Performance* (wersja angielska) [MS2790A 2006]. Narzędzie znajduje się w Panelach sterowania w katalogu Narzędzia administracyjne (narzędzie można również wywołać poleceniem *perfmon*). Umożliwia ono definiowanie tzw. liczników (ang. counter) należących do różnego typu obiektów (na przykład: pamięć, dysk logiczny), które mierzą zdefiniowane elementy środowiska. Zestawienie ważniejszych liczników [MS2790A 2006], mających wpływ na wydajność serwera bazodanowego zawiera Tab.9.

Obiekt	Licznik	Opis	Zalecana wartość
Logical Disk(*)	Avg. Disk Queue Length	Średnia wartość żądań odczytów oraz zapisów, które czekają w kolejce dla wybranego dysku w zadanym interwale	Średnia wartość >1.5 * ilość dysków fizycznych
Logical Disk(*)	Disk Bytes / sec	Prędkość, z jaką są transferowane dane z lub do	10MB/sec

		dysku podczas operacji czytania i zapisu	
Memory	Page faults/sec	Ilość odwołań do strony w pamięci, która nie zostaje odnaleziona w oczekiwanej lokalizacji. Obejmuje tzw. Hard Page Faults (strony nie ma w pamięci - dane są odczytywane z dysku) oraz Soft Page Faults (strona znajduje się w innym miejscu pamięci – angażowany jest procesor)	0-100 akceptowane przy normalnej pacy systemu
Memory	Avaliable MBytes	Ilość dostępnej wolnej pamięci	10 % dostępnej pamięci fizycznej
Memory	Pages /sec	Ilość stron zapisanych lub odczytanych z dysku w celu rozwiązania problemu Hard Page Faults	Średnio 5. Niepokojące: >20
Network Interface(*)	Bytes Total/sec	Ilość odebranych oraz przesłanych ramek dla wybranej karty sieciowej	50%
Processor(_Total)	% Privileged Time	Odsetek w jakim wątki są uruchomione w trybie uprzywilejowanym	10%
Processor(_Total)	% Processor Time	Czas wyrażony w procentach, w jakim procesor wykonuje nie bezczynne wątki	80%
Paging File(_Total)	% Usage	Suma instancji pliku stronicowania będącego w użyciu, wyrażona w procentach	70%
PhysicalDisk(*)	% Disk Time	Wyrażony w procentach czas wykonywania przez dysk operacji zapisu oraz odczytu	90%
PhysicalDisk(*)	Avg. Disk Queue Length	Średnia ilość żądań odczytu oraz zapisu oczekująca w kolejce dla wybranego dysku w wybranym przedziale czasowym	Nie powinna przekraczać 0,5 * ilość dysków fizycznych
Server Work Queues(0)	Queue Length	Bieżąca wartość kolejki serwera dla CPU	4
System	Context Switches/sec	Sumaryczna ilość przełączeń pomiędzy wątkami dla wszystkich procesorów	Średnio 1000 dla procesora
System	Processor Queue Length	Liczba wątków w kolejce procesora	10 * liczba procesorów

Tab.9. Zestawienie liczników systemowych

7.3. Narzędzia dla środowiska Linux oraz Windows

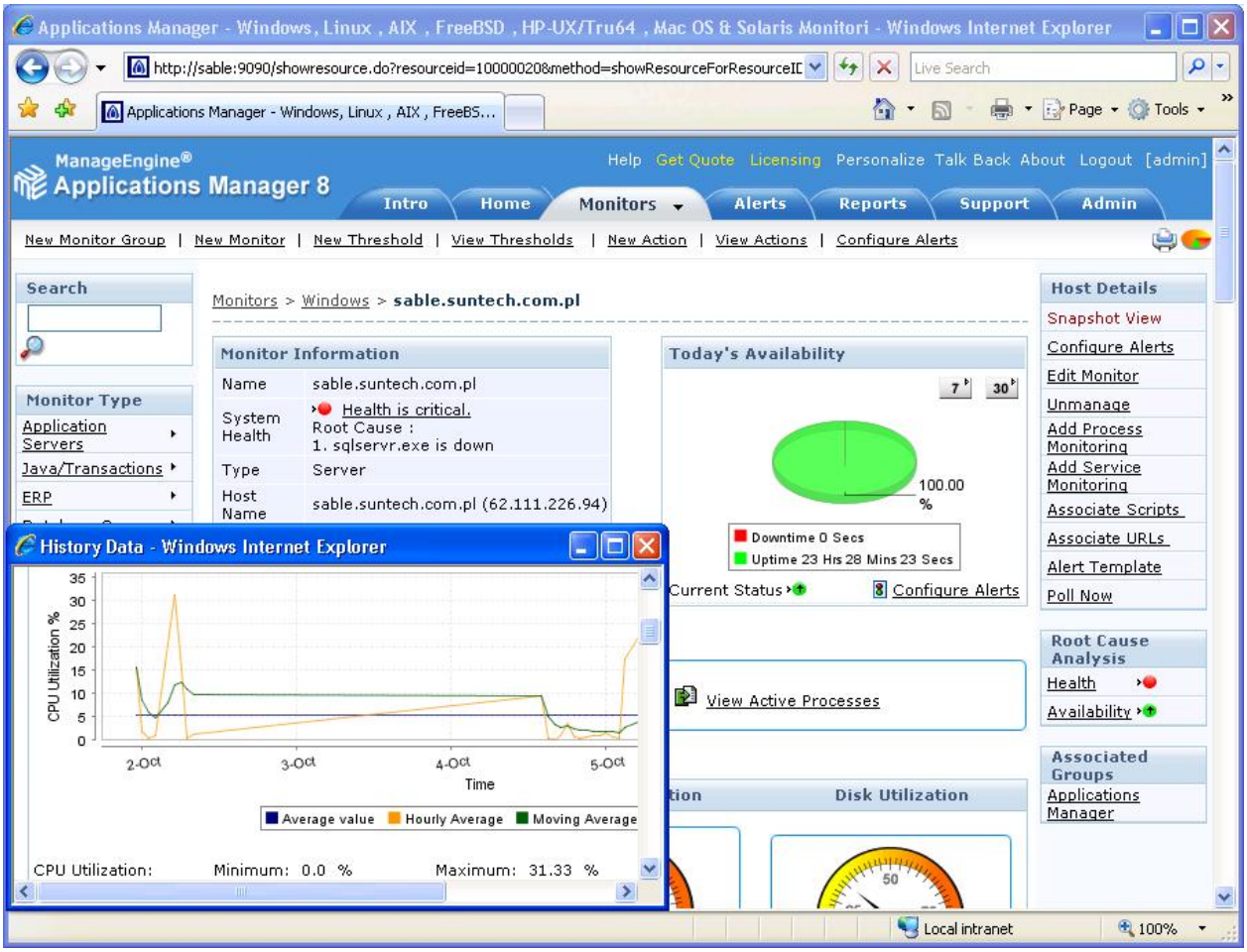
W przypadku porównywania zachowania się systemów bazodanowych w różnych środowiskach systemów operacyjnych, z pewnością niezbędne będzie jednolite narzędzie dokonujące pomiaru w różnych systemach w sposób podobny oraz oferujące identyczny zestaw pomiarów systemów. Jednym z wielu narzędzi jest aplikacja *Applications Manager*

zaimplementowana i oferowana przez AdventNet [WWWAM 2008]. System dostępny jest zarówno na platformę Linux jak i Windows. Rozwiązanie jest produktem komercyjnym, ale można stosować je bezpłatnie z ograniczeniem do pięciu monitorowanych systemów (tzw. monitorów).

Monitorowanym systemem może być zarówno system operacyjny (kategoria: serwer) jak i system bazodanowy (kategoria: baza danych). Na potrzeby testów wydajności środowiska bazodanowego w zupełności wystarczające są bardziej ogólne informacje (użycie procesora, wykorzystanie pamięci, użycie dysków fizycznych, i tym podobne) dotyczące środowiska (a zatem monitor serwera, czy to systemu Linux czy Windows). Narzędzie umożliwia monitoring następujących systemów:

- Windows 2000, 2003, XP;
- Linux;
- Sun OS;
- IBM AIX
- HP Unix;
- Tru64 Unix;
- FreeBSD;
- Mac OS.

Oczywiście możliwe jest bardziej precyzyjne monitorowanie samego systemu bazodanowego (pozyskując takie informacje jak liczba użytkowników, ilość zapytań, ilość blokad i inne), ale w zależności od serwera bazodanowego, do dyspozycji są różne statystyki (zatem trudno w takim przypadku o porównanie otrzymanych rezultatów). Ponadto *Applications Manager* w wersji 8, wspiera tylko pięć systemów bazodanowych: MySQL, Oracle, MS SQL Server, DB2, Sybase. Narzędzie oferuje intuicyjny graficzny interfejs webowy (Rys.24).



Rys.24. Applications Manager

7.4. Narzędzia własne

W celu umożliwienia przeprowadzenia jednolitych pomiarów dla wszystkich DBMS, a w szczególności zadanie tych samych zapytań SQL przez taką samą liczbę połączeń oraz w celu ujednolicenia pomiaru czasu odpowiedzi zostało opracowane przez autora niniejszego opracowania narzędzie o roboczej nazwie DBTester.

7.4.1. Zarys specyfikacji narzędzia DBTester

Narzędzie DBTester zostało zaimplementowane w języku Java, w środowisku programistycznym Eclipse (Eclipse-SDK-3.3.1.1-win32). Narzędzie składa się z następujących klas:

- DBTester – klasa główna;
- Watek_1 – klasa implementująca interfejs *Runable* uruchamiająca zestaw wątków (tablicę o zadanej ilości elementów), które pobierają listę zapytań SQL ze źródła_1;
- Watek_2 - klasa implementująca interfejs *Runable* uruchamiająca zestaw wątków (tablicę o zadanej ilości elementów), które pobierają listę zapytań SQL ze źródła_2.

Klasa Watek_X przyjmuje następujący zestaw parametrów:

- int _n – ilość powtórzeń sekwencji pytań wczytanych ze zmiennej _queryFile wykonywanych w danym wątku;
- int _id - identyfikator wątku;
- String _queryFile – ścieżka do pliku z sekwencją pytań SQL. Zapytania SQL powinny być poprawne dla wszystkich testowanych DBMS. Każde zapytanie należy rozpoczynać od nowego wiersza;
- int _connType – typ połączenia, tzn. DBMS, dla którego ma zostać nawiązane połączenie;
- int _databaseType – rodzaj bazy danych, (narzędzie to jest przewidziane do uruchamiania na skalowanych pod względem ilości danych bazach danych, w ramach tych samych DBMS).

Do głównych zmiennych, programu umożliwiających zarządzanie narzędziem DBTester należą:

- int N – ilość powtórzeń sekwencji pytań przechowywanych w zmiennej queryFile_1;
- int N2 – ilość powtórzeń sekwencji pytań przechowywanych w zmiennej queryFile_2;
- int NN - ilość wątków realizujących sekwencję pytań queryFile_1;
- int NN2 - ilość wątków realizujących sekwencję pytań queryFile_2;
- int connType – DBMS, do którego łączy się aplikacja;
- int databaseType – rodzaj bazy danych;
- String queryFile_1 – ścieżka do pliku tekstowego zawierającego sekwencję zapytań SQL dla wątków typu pierwszego;

- String queryFile_2 – ścieżka do pliku tekstowego zawierającego sekwencję zapytań SQL dla wątków typu drugiego;
- boolean WriteLog – decyduje o zapisie otrzymywanych rezultatów do pliku logu;
- String LogPath – ścieżka do pliku logu.

Narzędzie DBTester oprócz bieżącej prezentacji na standardowym wyjściu aktualnie wykonywanego polecenia, dodatkowo zapisuje istotne informacje do pliku logu (zmienna String LogPath), o ile zmienna boolean WriteLog jest ustawiona na true (wartość domyślna). Opis tworzonego logu:

- nagłówek: ----- STARTED at <czas> ON DATABASE: <_databaseType> -----
- połączenie: <connection string>
- część właściwa: <czas> w_typ_<wątek ½> {<nr wątku>}[<nr sekwencji powtórzeń>][<wiersz, z którego zostało przeczytane zapytanie SQL>](<treść zapytania SQL>)
- podsumowanie czasu: TIME: <czas sekwencji pytań w sek.>
- stopka: ---------- STOPED at <czas> ----------

7.4.2. Obsługa narzędzia DBTester

Aby uruchomić narzędzie DBTester niezbędne jest:

- środowisko Eclipse;
- zainstalowane sterowniki (drivery) JDBC dla testowanych DBMS. Aktualnie aplikacja obsługuję następujące DBMS:
    * com.microsoft.sqlserver.jdbc.SQLServerDriver;
    * com.ibm.db2.jcc.DB2Driver;
    * com.mysql.jdbc.Driver;
    * oracle.jdbc.driver.OracleDriver;
    * org.postgresql.Driver.

W celu poprawnego uruchomienia narzędzia DBTester należy po wcześniejszym uruchomieniu testowanego DBMS skonfigurować *connection string* oraz ustawić główne zmienne opisane w rozdziale 7.4.1.

Przykładowe wartości connection string dla omawianych w pracy DBMS przedstawiono poniżej:

- jdbc:sqlserver: //sable: 1433; databasename= DziennikSzkolny","sa","haslo";
- jdbc:oracle:thin:@sable:1521:ORACLEDB","system","haslo";
- jdbc:db2://sable:50000/DS","db2admin","haslo";
- jdbc:mysql://sable/DziennikSzkolny_dbo","root","haslo";
- jdbc:postgresql://sable:5432/DziennikSzkolny","root","haslo".

Zaleca się zapisywanie rezultatów wykonywanych testów do plików logów. Pliki logów przy kolejnych uruchomieniach aplikacji zostają dopisywane do końca istniejącego pliku wskazywanego przez zmienną LogPath. Aplikacja nie kontroluje wielkości pliku.
Zebrane w logu dane można analizować dowolnymi narzędziami. Przykładowy fragment pliku logu przedstawiono poniżej:

----------STARTED at 2008-07-18 01:37:43:439 ON DATABASE: 1----------

oracle.jdbc.driver.T4CConnection@cbdb20
2008-07-18 01:37:37:371: w_typ_1{0}[0][0](select ocena from oceny)
2008-07-18 01:37:37:372: w_typ_1{0}[0][1](select imie from uczniowie)
2008-07-18 01:37:37:375: w_typ_1{0}[0][2](select pr_id from prowadza)
2008-07-18 01:37:37:372: w_typ_1{0}[0][3](select max(n_id)from nauczyciele)
2008-07-18 01:37:37:374: w_typ_1{0}[0][4](select z_id from przedmioty ,prowadza, zaliczenie where przedmioty.p_id=prowadza.p_id and prowadza.pr_id = zaliczenie.pr_id)
2008-07-18 01:37:38:381: w_typ_1{0}[0][5](select p.pr_id, imie, nazwisko, nazwa from nauczyciele n, prowadza p , przedmioty prz where p.p_id = prz.p_id and p.n_id = n.N_id)
2008-07-18 01:37:38:384: w_typ_1{0}[0][6](select k_id from klasy where nr_klasy = 1 and indeks_klasy = 'A')
.
.
.
.
2008-07-18 01:37:42:421: w_typ_1{1}[8][7](select z_id, imie, nazwisko, nazwa from zaliczenie z, uczniowie u, prowadza pro, przedmioty prz where z.u_id=u.u_id and pro.pr_id=z.pr_id and pro.p_id=prz.p_id)
2008-07-18 01:37:42:429: w_typ_1{1}[8][8](select imie, nazwisko, nazwa, ocena, data from zaliczenie z, uczniowie u, prowadza pro, przedmioty prz, oceny o where z.u_id=u.u_id and pro.pr_id=z.pr_id and pro.p_id=prz.p_id and o.z_id=z.z_id)
2008-07-18 01:37:42:426: w_typ_1{1}[9][0](select ocena from oceny)
2008-07-18 01:37:42:423: w_typ_1{1}[9][1](select imie from uczniowie)
2008-07-18 01:37:42:421: w_typ_1{1}[9][2](select pr_id from prowadza)

2008-07-18 01:37:42:428: w_typ_1{1}[9][3](select max(n_id)from nauczyciele)

2008-07-18 01:37:42:425: w_typ_1{1}[9][4](select z_id from przedmioty ,prowadza, zaliczenie where przedmioty.p_id=prowadza.p_id and prowadza.pr_id = zaliczenie.pr_id)

2008-07-18 01:37:42:427: w_typ_1{1}[9][5](select p.pr_id, imie, nazwisko, nazwa from nauczyciele n, prowadza p , przedmioty prz where p.p_id = prz.p_id and p.n_id = n.N_id)

2008-07-18 01:37:43:434: w_typ_1{1}[9][6](select k_id from klasy where nr_klasy = 1 and indeks_klasy = 'A')

2008-07-18 01:37:43:431: w_typ_1{1}[9][7](select z_id, imie, nazwisko, nazwa from zaliczenie z, uczniowie u, prowadza pro, przedmioty prz where z.u_id=u.u_id and pro.pr_id=z.pr_id and pro.p_id=prz.p_id)

2008-07-18 01:37:43:439: w_typ_1{1}[9][8](select imie, nazwisko, nazwa, ocena, data from zaliczenie z, uczniowie u, prowadza pro, przedmioty prz, oceny o where z.u_id=u.u_id and pro.pr_id=z.pr_id and pro.p_id=prz.p_id and o.z_id=z.z_id)

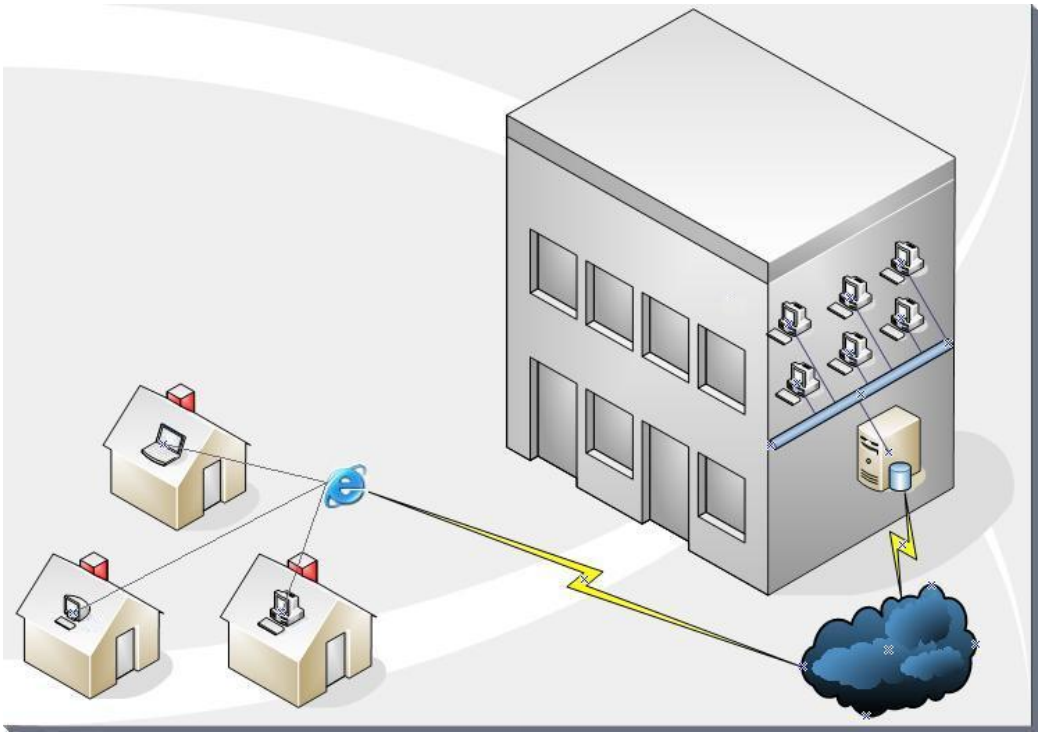
TIME: 6.097 sek

----------STOPED at 2008-07-18 01:37:43:439----------

Rozdział VIII Projekt bazy danych wykorzystanej w pracy

Celem projektowanego systemu [LACH 2006] było opracowanie elektronicznej wersji tradycyjnego dziennika lekcyjnego, który umożliwi bieżącą kontrolę wyników w nauce oraz frekwencji uczniów (w łatwy i mniej pracochłonny sposób) dostarczając równocześnie narzędzi raportujących.

System powinien zapewniać możliwość wprowadzania, edycji oraz usuwania danych dotyczących uczniów, nauczycieli, przedmiotów, ocen oraz obecności. Ponadto należy umożliwić dodawanie użytkowników mogących korzystać z zasobów bazy. Dodatkowo system ma za zadanie uprościć czynności wykonywane przez nauczycieli, zminimalizować możliwość popełnienia błędów, zwiększyć bezpieczeństwo związane z dostępem do przechowywanych danych oraz zapewnić powszechny i szybki dostęp do informacji o uczniu (umożliwiając zestawienie i raportowanie danych) [LACH 2006].



Rys.25. Wizja systemu

8.1. Opis ogólny dziennika szkolnego

Dziennik szkolny, dokument wewnętrzny wykorzystywany obowiązkowo w szkolnictwie, regulowany przez normy prawne, zawiera następujące elementy [LACH 2006]:

- alfabetyczny spis uczniów – zawiera indeks uczniów należących do jednej klasy w kolejności alfabetycznej (zaczynając od nazwiska);
- spis przedmiotów – zawiera indeks przedmiotów nauczanych w danej klasie. Każdy uczeń posiada rubryki, w których pedagog ma możliwość wpisywania osiągnięć bądź porażek ucznia w formie oceny. W spis ten wchodzi również zachowanie, które wzorem innych przedmiotów podlega ocenie. Aktualnie skala ocen należy do przedziału od 1 (niedostateczny) do 6 (celujący) i przyjmuje wartości całkowite. Ponadto każda wartość należąca do przedziału <2;5> może otrzymać dodatkową informację w postaci znaku „+” (plus) nadającego większą wartość ocenie wystawianej oraz będąca wartością mniejszą od oceny wyższej ze znakiem „-” i na odwrót. Ocena niedostateczna może posiadać jedynie znak „+”; ocena celująca - „– ”. Oczywiście zdarzają się wyjątki, ale nie są one częste (choć nie należy ich ignorować). Warto jeszcze zaznaczyć, iż oceny na koniec semestru oraz roku muszą posiadać wartości całkowite;
- lista obecności – lista, w której każdy uczeń w danym dniu nauczania ma przyporządkowaną liczbę pól równą ilości godzin przeznaczonych na zajęcia edukacyjne w danej placówce. Istnieją cztery rodzaje zapełnień: obecny, nieobecny, usprawiedliwiony lub spóźniony (przy czym często pole nieobecny zostaje zmienione na usprawiedliwiony lub spóźniony);
- informacje o uczniu – zbiór danych o uczniu, które są niezbędne w celach urzędowych oraz kontaktowych dla jednostki oświatowej;
- spis uwag – miejsca przeznaczone do krótkich opisowych notatek wykonywanych przez pedagogów odnoszących się do zachowania i postawy ucznia. Uwagi ogólnie można podzielić na pozytywne (otrzymywane za wyróżniające się zachowanie lub postawę) oraz negatywne (otrzymywane za zachowania godne nagany).

Najważniejsze czynności związane z „eksploatacją” dokumentu [LACH 2006]:

- wypełnienie dziennika przed rozpoczęciem roku szkolnego - wpisanie uczniów do danej klasy, przypisanie nauczycieli do prowadzonych przedmiotów w danych klasach, przypisanie wychowawcy do klasy, przypisanie sali klasowej (którą opiekują się uczniowie danej klasy i w której mają wszystkie lub większość zajęć edukacyjnych);
- wypełnianie listy obecności wiążącej się z prowadzonym przez nauczyciela przedmiotem;
- usprawiedliwianie nieobecności przez wychowawcę klasy;
- wpisywanie ocen otrzymanych przez ucznia przez nauczyciela prowadzącego dany przedmiot. Oczywiści zdarzają się sytuacje, gdy nauczyciel w zastępstwie prowadzi zajęcia i ma możliwość sprawdzenia obecności oraz wystawienia ocen;
- poprawianie ocen wystawionych przez nauczycieli – na zasadzie wykonania przez ucznia zadania poprawkowego (poprawa sprawdzianu, odpowiedzi, zadania i tym podobne);
- wypełnianie dziennika na koniec semestru nauki: wystawianie ocen końcowych, liczenie średnich ocen ucznia, przedmiotu, klasy.

8.2. Analiza wymagań użytkowników

Na samym początku należy określić użytkowników, zanim zostaną poddane analizie ich oczekiwania wobec systemu. W myśl założenia projektu (którego głównym celem jest realizacja powszechnego dostępu nauczyciela, ucznia oraz jego opiekuna prawnego do informacji pochodzących z placówki oświatowej) użytkowników możemy podzielić na dwie grupy: Nauczyciele oraz Uczniowie. Oczywiście grupy te można dalej dzielić (Nauczycieli na zwykłych – prowadzących zajęcia i uprzywilejowanych – dyrektorów szkół mających dostęp do większej ilości informacji; Uczniów na zwykłych i członków samorządów, którzy również będą mieli większe uprawnienia), lecz nie jest to istotne na etapie określania wymagań użytkowników [LACH 2006].

Podstawowe wymagania użytkowników należących do grupy Nauczyciele [LACH 2006]:

- dostęp do listy obecności uczniów na zajęciach z danego przedmiotu nauczania

w celu umożliwienia sprawdzenia frekwencji;

- dostęp wychowawcy do list obecności swoich uczniów w celu ich weryfikacji oraz możliwości usprawiedliwiania nieobecności. Ponadto należy przewidzieć wypadek modyfikacji obecności ucznia, który spóźnił się na zajęcia i dotarł po sprawdzeniu listy obecności;
- dostęp do arkusza ocen w celu przypisania uczniowi oceny z danego przedmiotu oraz dołączenia krótkiej notatki (opcjonalnej) odnoszącej się do wystawianej oceny (na przykład: zakres materiału, forma ocenianej odpowiedzi i inne);
- dostęp do wystawionych już ocen w celu ich modyfikacji lub wykorzystania do określenia dalszej pracy;
- dostęp do formularzy wstawiania uwag o uczniu;
- dostęp do wszystkich ocen danego ucznia (na przykład w celu wystawienia ocen końcowych);
- dostęp do dodatkowych informacji o uczniu w celu nawiązania kontaktu z opiekunem lub wykonania innych czynności administracyjnych, które wymagają posiadania dodatkowych informacji (na przykład: data urodzenia ucznia, miejsce zamieszkania i inne);
- dostęp do grupy raportów w celu umożliwienia wykonania wydruków różnego typu zestawień.

Podstawowe wymagania użytkowników należących do grupy Uczniowie [LACH 2006]:

- dostęp do arkuszu ocen zalogowanego użytkownika (lub wychowanka zalogowanego opiekuna prawnego);
- dostęp do informacji dotyczących obecności;
- dostęp do uwag wystawianych na temat danego ucznia.

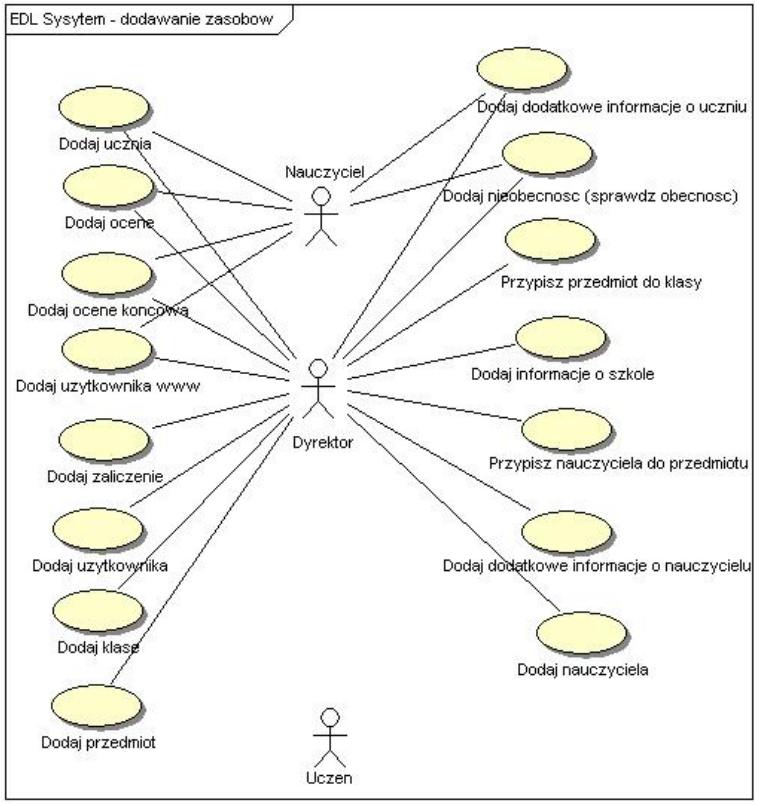
Sposób realizacji powyższej funkcjonalności [LACH 2006]:

- alfabetyczny spis uczniów jest dostępny w kontekście danej klasy i realizacji takich funkcjonalności jak: wpisywanie ocen (cząstkowych i końcowych) czy sprawdzanie obecności;
- spis przedmiotów jest dostępny pośrednio w kontekście funkcjonalności związanych z wystawianiem ocen czy sprawdzaniem obecności. Dodatkowo spis ten jest dostępny podczas definiowania / przeglądania przydziałów

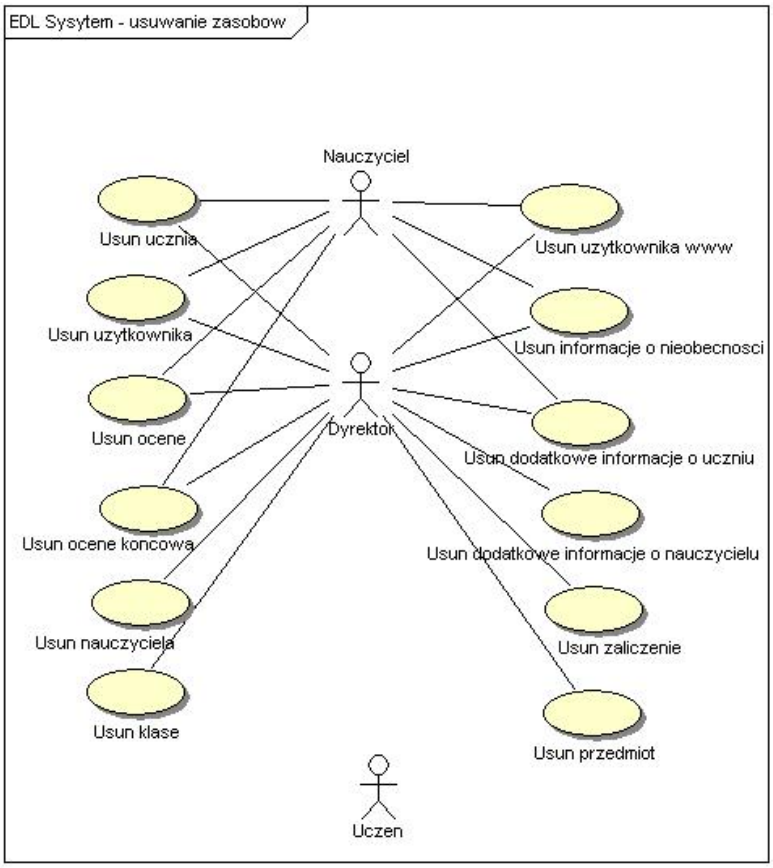
(jest to przypisanie przedmiotu oraz nauczyciela prowadzącego dany przedmiot do klasy);

- lista obecności ograniczona jest do nieobecności. Brak informacji o nieobecności jest równoznaczny z potwierdzeniem obecności ucznia na zajęciach. Przy czym obecność może być usprawiedliwiona bądź nieusprawiedliwiona;
- system posiada dodatkowy zbiór informacji zarówno o uczniu jak i o nauczycielach (wykorzystywany w tym przypadku do celów kontaktowych jak i kadrowo-płacowych);
- spis uwag – realizowany jest przez funkcjonalność związaną z ocenami. Prowadzący chcąc wpisać uczniowi uwagę wystawia mu ocenę cząstkową z przedmiotu zwanego ‘zachowanie’ oraz w miejscu na komentarz wpisuje treść uwagi.

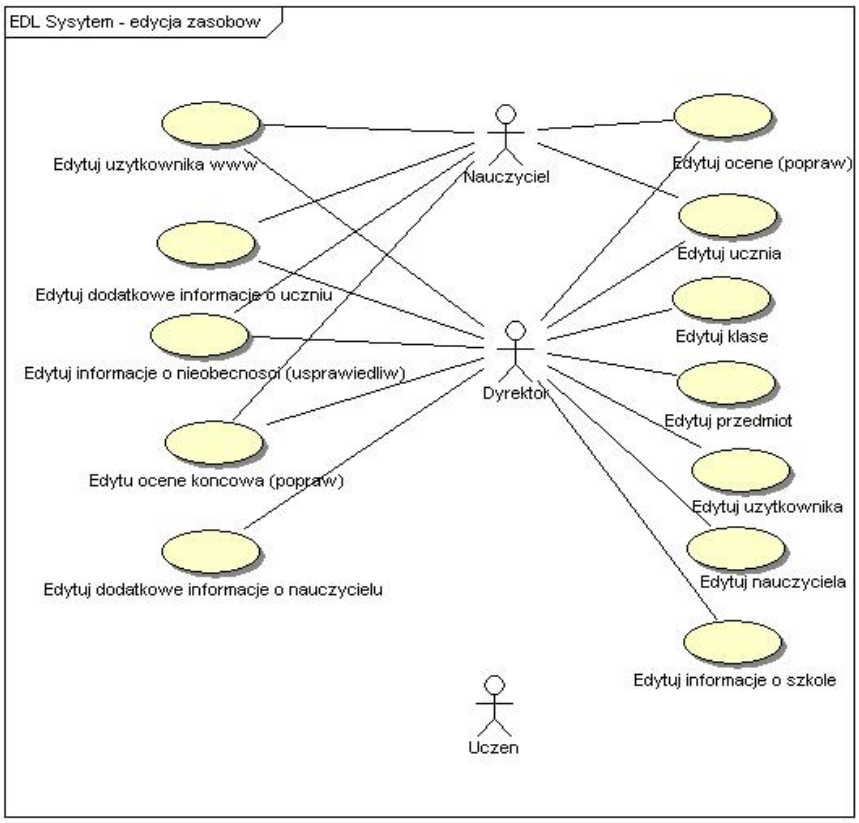
Poniżej zamieszczono diagramy przypadków użycia:



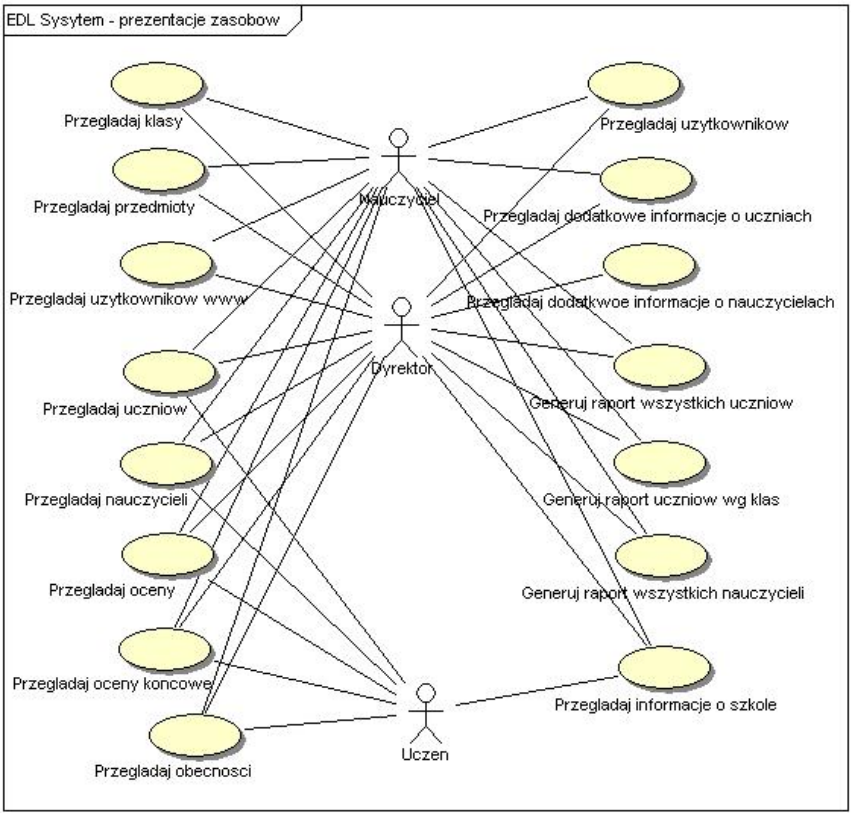
Rys. 26. Diagram przypadków użycia – dodawanie zasobów



Rys. 27. Diagram przypadków użycia – usuwanie zasobów



Rys. 28. Diagram przypadków użycia – edycja zasobów



Rys. 29. Diagram przypadków użycia –prezentacja zasobów

8.3. Diagram Związków Encji

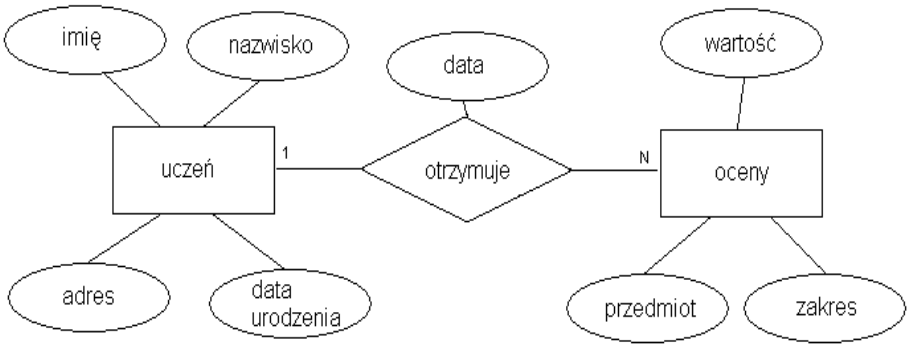
Diagram Związków Encji (DZE) (ang. Entity Relationship Diagram (ERD)) - jest modelem abstrakcyjnie przedstawiającym strukturę bazy danych [HECTOR 2006], służącym do opisu świata rzeczywistego, czyli informacji, które tworzą część rzeczywistości niezbędną dla użytkownika systemu. Opis ten jest graficzny. Składa się z encji i łączących je związków oraz atrybutów. Diagram Związków Encji tworzą następujące symbole graficzne:

- prostokąty (obrazują zbiory encji);
- romby (obrazują związki zachodzące pomiędzy encjami);
- owale (obrazują atrybuty encji lub związków).

Encja (*entity*) – jest terminem, niepodlegającym definiowaniu. Encja jest reprezentacją „obiektu” ze świata rzeczywistego, który istnieje i można go odróżnić od innej encji. Encje o podobnej charakterystyce tworzą zbiory encji, które są analogiczne do klas [ULLMAN 1999].

Związki (*relationships*) – są to połączenia (zależności) pomiędzy zbiorami encji [LACH 2006].

Atrybuty (*attributes*) – ich wartości opisują właściwości encji [ULLMAN 1999], na podstawie których można doszukiwać się podobieństwa encji w celu tworzenia ich zbiorów. Z drugiej strony wartości atrybutów pozwalają na odróżnienie poszczególnych encji w danym zbiorze. Na Rys. 30 przedstawiono przykładowy Diagram Związków Encji [LACH 2006].



Rys. 30. Przykładowy Diagram Związków Encji

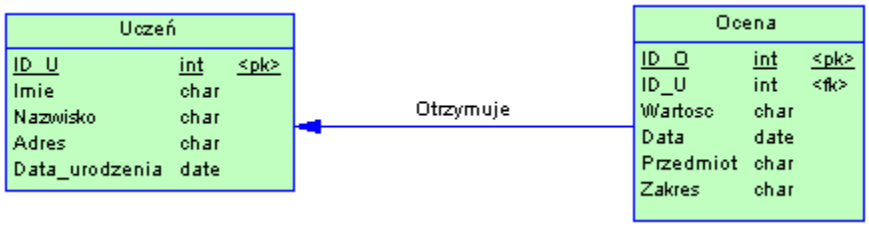
Na powyższym diagramie można zauważyć, iż związek (w tym przypadku jest to związek binarny) jest dodatkowo opisany typem asocjacji. W przypadku Rys. 30 jest to 1 : N (oznaczający, iż z jedną z encji o nazwie uczeń może zachodzić jeden lub więcej związków z encją oceny) [LACH 2006].

8.4. Transformacja modelu konceptualnego do modelu logicznego

Przekształcenie Diagramu Związków Encji (model konceptualny) w model relacyjny polega na zastosowaniu następujących ogólnych reguł:

- zbiór encji przekształcamy w relację (tabelę);
- atrybuty encji przekształcamy w atrybuty relacji (kolumna tabeli);
- związki pomiędzy zbiorami encji przekształcamy w związki pomiędzy relacjami.

Przykładowy Diagram Związków Encji przedstawiony na Rys. 30 będzie zatem wyglądał w następujący sposób (Rys. 31) [LACH 2006]:



Rys. 31. Przykładowy model logiczny

Oznaczenia występujące w powyższym modelu:

- <pk> - Klucz główny (ang. primary key);
- <fk> - Klucz obcy (ang. foreign key).

8.5. Normalizacja

Po wygenerowaniu modelu logicznego musi zostać przeprowadzona weryfikacja poprawności utworzonego modelu. W dalszej kolejności należy przeprowadzić normalizację otrzymanego modelu.

Normalizacja jest procesem usuwania z relacji niepożądanych właściwości (anomalii). Wyróżnia się następujące anomalie (ang. anomalies) [ULLMAN 1999]:

- redundancja danych (powtórzenie danych);
- istnienia bądź wstawiania (należy wprowadzić związane dane przy wstawianiu danych);
- modyfikacji (zmiana danych następuje tylko w jednym miejscu mimo innych wystąpień);
- usuwania (utrata powiązanych danych przy usunięciu danych).

Wyróżnia się pięć Postaci Normalnych (ang. Normal Form) oraz Postać Normalną Boyce Codd'a (ang. Boyce Codd Normal Form). W praktyce stosuje się trzy pierwsze postaci normalne (1NF, 2NF, 3NF), dlatego tylko one zostaną omówione.

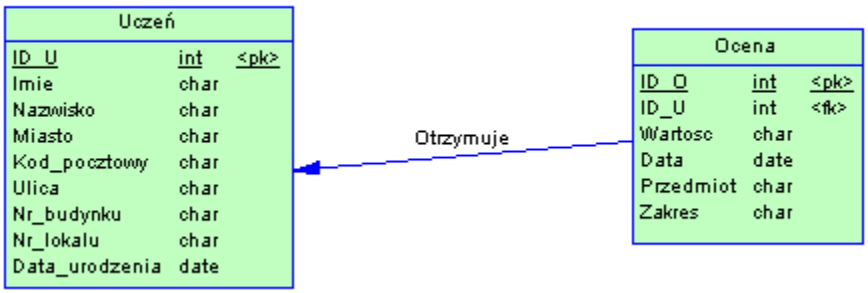
1NF – Pierwsza Postać Normalna – relacja jest w 1NF, jeżeli każda składowa w każdej krotce ma wartość atomową [ULLMAN 1999]. Oznacza to, iż dziedziny wszystkich

atrybutów danej relacji są zbiorami wartości prostych (czyli są niepodzielne). Oczywiście należy w tym miejscu uwzględnić poziom szczegółowości danych wymagany w danym systemie.

Przykładowa relacja z Diagramu Związków Encji (Rys. 30) w postaci 1NF wygląda następująco (Rys. 32) [LACH 2006].

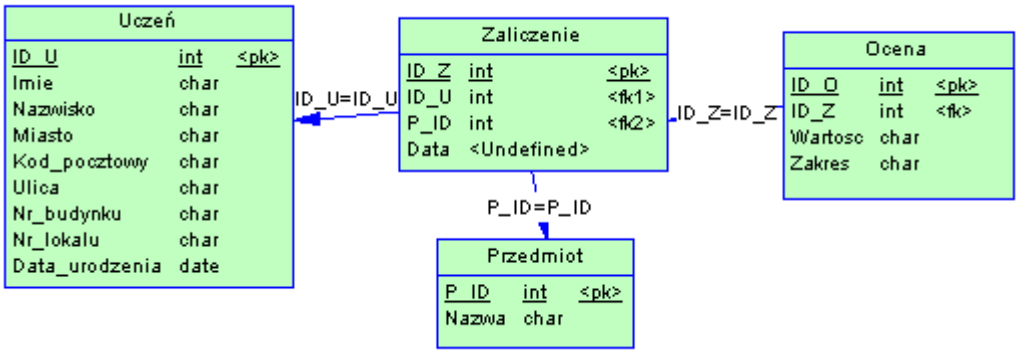
Rys. 32. Przykładowa relacja w 1NF

2NF – Druga Postać Normalna – relacja jest w 2NF, jeżeli jest w 1NF oraz każdy atrybut wtórny (nienależący do żadnego klucza) jest w pełni funkcyjnie zależny od klucza schematu. Jeżeli schemat posiada klucz prosty (złożony z jednego atrybutu) i jest w 1NF, to jest automatycznie w 2NF. Przykładową relację z Rys. 32. w postaci 2NF przedstawiono na Rys. 33 [LACH 2006].



Rys. 33. Przykładowa relacja w 2NF

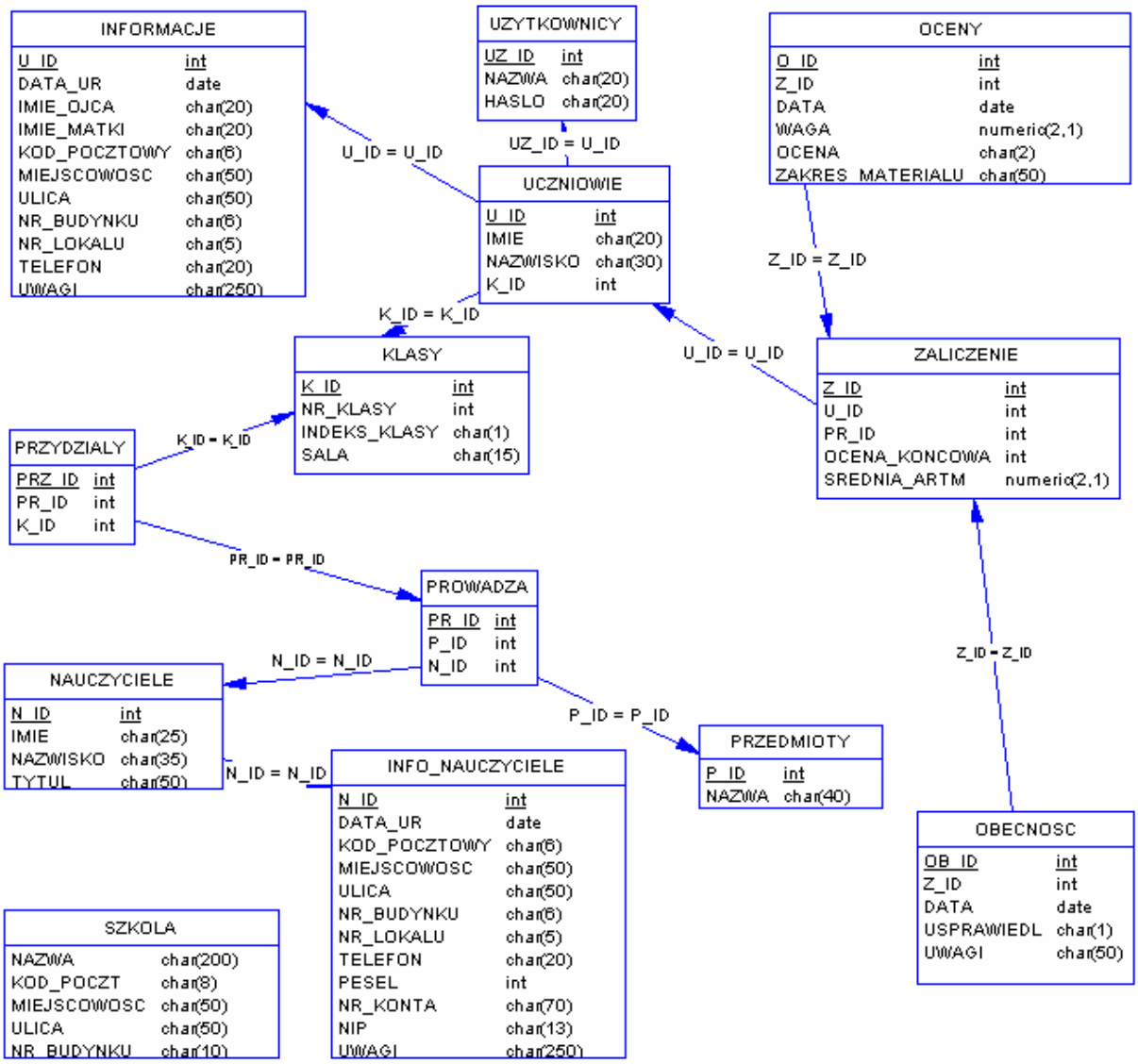
3NF – Trzecia Postać Normalna – relacja jest w 3NF, jeżeli jest w 2NF oraz każdy atrybut wtórny jest bezpośrednio (a nie przechodnio) zależny od klucza schematu. Przykładową relację z Rys. 32. w postaci 3NF przedstawiono na Rys. 34 [LACH 2006].



Rys. 34. Przykładowa relacja w 3NF

Krótki opis najbardziej istotnych tabel (schemat bazy danych, dla projektowanego systemu przedstawiono na Rys. 35):

- Uczniowie - tabela główna zawierająca podstawowe informacje o uczniach;
- Zaliczenie - zawiera informację, z jakiego przedmiotu dany uczeń ma prowadzone zajęcia oraz kto jest prowadzącym zajęcia. Dodatkowo stanowi ona wyjście (jest połączona kluczami) dla tabel: obecność i oceny. Co więcej przechowuje informacje o ocenie końcowej z danego przedmiotu.
- Oceny – tabela zawiera informacje o ocenie, dacie oraz krótki opis. Co więcej zawiera atrybut waga (jest one jednak na razie nieobsługiwany);
- Przydziały – tabela zawiera informacje o przypisaniu przedmiotów (oraz prowadzących przedmioty) do poszczególnych klas.



Rys. 35. Schemat bazy danych

Rozdział IX Metodologia testów

Jak już wspomniano na wstępie, myślą przewodnią niniejszego opracowania jest wytworzenie prostego, powtarzalnego schematu postępowania umożliwiającego, w krótkim czasie wybór najodpowiedniejszego DBMS dla analizowanego schematu bazy danych w określonym środowisku. Istotnym aspektem omawianego problemu jest uzyskanie porównywalnych warunków testów.

9.1. Instalacja SZBD z domyślną konfiguracją

Każdy testowany SZBD powinien zostać zainstalowany z domyślną konfiguracją. Oczywisty jest fakt, iż tunning systemu bazodanowego może znacznie zwiększyć jego wydajność. Jednak każdy producent DBMS dostarczając system domyślnie konfiguruje go w taki sposób, aby był jak najbardziej wydajny w standardowych zastosowaniach. Ponieważ celem jest uproszczenie testów, zostaje przyjęte, iż instalowane systemy zarządzania bazą danych nie podlegają dodatkowym konfiguracją.

Oczywiście w przypadku, jeśli z pełnej palety systemów po pierwszych testach zostaną nam dwa lub trzy systemy, które będą miały podobne osiągnięcia, można a nawet trzeba pokusić się o tunning systemów w celu znalezienia najbardziej wydajnego produktu w zadanym środowisku.

W analizach nie uwzględniono zależności ekonomicznych w podejmowaniu decyzji.

9.2. Pomiary środowiska

Pomiary środowiska, których należy dokonać dzielimy na dwa przypadki:

- pomiary dokonane w stanie „zero” – oczywiście prawidłowym pomiarem byłby pomiar środowiska przed zainstalowaniem DBMS. Ponieważ metodologia ma być jednak jak najbardziej przystępna, a zarazem miarodajna, pomiaru dokonujemy przy zatrzymanych usługach systemów zarządzania bazami danych;

- pomiary dla wybranego DBMS – tak, jak w powyższym przypadku najbardziej prawidłowym zachowaniem byłaby każdorazowa instalacja systemu testowanego na danej maszynie. Oczywiście, jeśli mamy dostęp do kilku maszyn o identycznej konfiguracji sprzętowej (kwestia zakupu identycznego sprzętu) i programowej (kwestia identycznej instalacji systemu operacyjnego, bądź jego odtworzenie z obrazu systemu) najlepiej będzie dokonywać pomiaru na „dedykowanych" maszynach dla danego DBMS. Dążąc jednak do uproszczenia przyjmiemy, iż dokonanie pomiarów dla danego sytemu zarządzania bazą danych, rozumiemy poprzez uruchomienie testów w przypadku, gdy na maszynie oprócz uruchomionych usług / procesów w stanie „zero" uruchomione są jedynie składowe DBMS danego dostawcy.

Istotne jest, aby zarówno narzędzie testujące (takie jak DBTester), jak i narzędzia dokonujące pomiarów stanu środowiska (na przykład monitor wydajności systemu Windows) uruchamiać na innej maszynie.

9.3. Skalowanie bazy i użytkowników

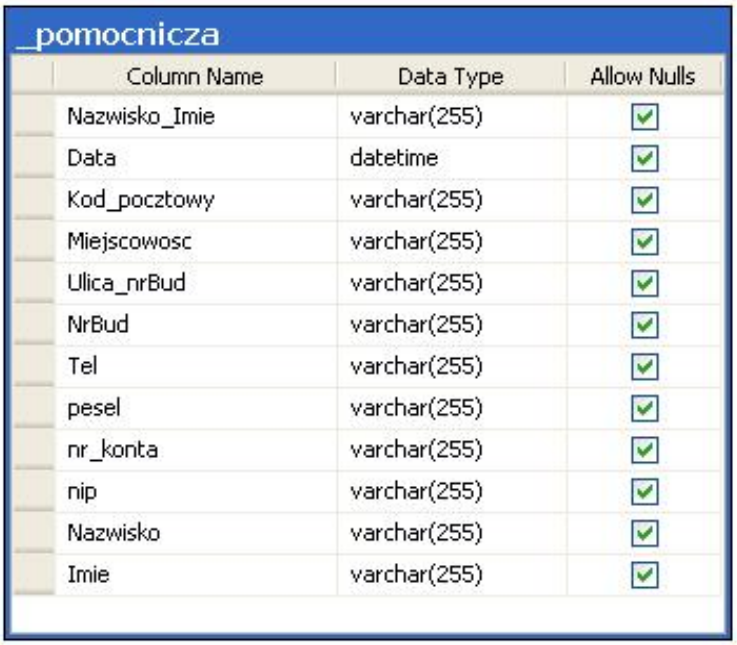
Jak wspomniano we wcześniejszych rozdziałach wydajność testowanych systemów, będzie weryfikowana ze względu na skalowanie:

- ilości danych;
- ilości użytkowników równocześnie korzystających z bazy danych.

O ile zmianę aktywnych równoległych użytkowników korzystających z bazy danych umożliwia narzędzie DBTester (rozdział 7.4), to w przypadku dodania skalowalności względem ilości danych zadanie jest bardziej skomplikowane. Prostą metodologią postępowania w celu prawidłowego skalowania danych jest:

- utworzenie tabeli / tabel pomocniczych zawierających zróżnicowane dane, które posłużą do wypełnienia tabel właściwych;
- zasilenie tabeli pomocniczej dużą ilością przykładowych danych;
- utworzenie skryptów sparametryzowanych [LADANYI 2000] [GRUBER 2000], wypełniających wartości w tabelach (w kolejności pozwalającej zachować integralność danych).

Przykładowy schemat tabeli pomocniczej dla analizowanej bazy danych:



_pomocnicza

Column Name	Data Type	Allow Nulls
Nazwisko_Imie	varchar(255)	☑
Data	datetime	☑
Kod_pocztowy	varchar(255)	☑
Miejscowosc	varchar(255)	☑
Ulica_nrBud	varchar(255)	☑
NrBud	varchar(255)	☑
Tel	varchar(255)	☑
pesel	varchar(255)	☑
nr_konta	varchar(255)	☑
nip	varchar(255)	☑
Nazwisko	varchar(255)	☑
Imie	varchar(255)	☑

Rys. 36. Tabela pomocnicza

Kolejnym krokiem jest wykonanie skryptów SQL umożliwiających pompowanie danych do tabel właściwych [KLINE 2002]. Przykład skryptu dla MS SQL Server, ładującego losowo wyszukane dane do tabeli Uczniowie (przed wykonaniem tego skryptu należy załadować dane do tabeli Klasy):

```
-----------
-- UCZNIOWIE --
-----------
select * from Klasy
select * from Uczniowie
create table #T (id int identity(1,1) not null, i varchar(255), n varchar(255))
insert into #T (i,n)
select top 1000000  a.imie, b.nazwisko from _pomocnicza a, _pomocnicza b where a.imie is not
  null and b.nazwisko is not null
declare @i int
set @i=123
while @i<185
begin
        declare @j int
        set @j=0
        while @j<40
        begin
                declare @ran int
                declare @im varchar(255)
                declare @n varchar(255)
                set @ran=(SELECT 1000000*RAND( (DATEPART(mm, GETDATE()) * 100000 )
                        + (DATEPART(ss, GETDATE()) * 1000 )
                        + DATEPART(ms, GETDATE()) ))
                set @im=(select i from #T where id=@ran)
                set @n=(select n from #T where id=@ran)
                insert into uczniowie (imie, nazwisko, k_id) values (@im, @n, @i)
                set @j=@j+1
        end

        set @i=@i+1
end

drop table #T
```

Skrypt dla każdej klasy o identyfikatorze 1-185 tworzy i przypisuje grupę 40 uczniów z losowo wybranymi danymi (imieniem oraz nazwiskiem).

9.4. Migracja baz danych do wybranych SZBD

Jednym z trudniejszych etapów przygotowywania danych do testów jest ich migracja pomiędzy różnymi dostawcami DBMS. Ponieważ procesy migracji są dość rozbudowanymi zagadnieniami zostaną w tym miejscu zaznaczone praktyczne wskazówki, które były wykorzystywane do migracji systemów z MS SQL Server 2005 do różnych DBMS.

W części omawianych przypadków do wyeksportowania danych z bazy danych MS SQL Server do pliku płaskiego, wykorzystane będzie narzędzie bcp [ITZIK 2006], dostępne z wiersza poleceń (ang. Command Prompt), po zainstalowaniu MS SQL Server. Składnia polecenia dostępna jest po wpisaniu bcp w narzędziu Command Prompt. Przykład użycia narzędzia:

- eksport całej tabeli: *bcp %SQLDB%.dbo.Uczniowie out %RESULTDIR%\uczniowie.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%*;
- eksport rezultatów zapytania: *bcp "select N_ID,KOD_POCZTOWY ,MIEJSCOWOSC, ULICA,NR_BUDYNKU,NR_LOKALU,TELEFON,PESEL,NR_KONTA,NIP,UWAGI,substring(cast(year(data_ur) as varchar(255)), 1,4)+'/'+cast(month(data_ur) as varchar(255))+'/' +cast(day(data_ur) as varchar(255)) as data_ur from DS_2.dbo.INFO_NAUCZYCIELE" queryout %RESULTDIR%\INFO_NAUCZYCIELE.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%*.

Dla powyższych przykładów zdefiniowane zostały zmienne, które wykorzystano w powyższych przykładach. Oto nagłówek pliku batch'owego (rozszerzenie .bat):

```
@echo off
SET SQLSERVER=sable
SET SQLUSER=sa
SET SQLPWD=haslo
SET SQLDB=DS_2
SET BCPCONN=-S "%SQLSERVER%" -U %SQLUSER% -P %SQLPWD%
SET OSQLCONN=-S "%SQLSERVER%" -U %SQLUSER% -P %SQLPWD%
SET RESULTDIR=.
```

9.4.1. DB2

W przypadku DBMS firmy IBM migracja danych do DB2 odbywała się w następujący sposób:

- wykonanie eksportu danych (ang. unload) z MS SQL Server do plików danych za pomocą narzędzia bcp [ITZIK 2006]:

```
@echo off
SET SQLSERVER=sable
SET SQLUSER=sa
SET SQLPWD=haslo
SET SQLDB=DS_3
SET BCPCONN=-S "%SQLSERVER%" -U %SQLUSER% -P %SQLPWD%
SET OSQLCONN=-S "%SQLSERVER%" -U %SQLUSER% -P %SQLPWD%
SET RESULTDIR=.
echo START: %date%_%time%
bcp %SQLDB%.dbo.Uczniowie out %RESULTDIR%\uczniowie.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.Przedmioty out %RESULTDIR%\przedmioty.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.Nauczyciele out %RESULTDIR%\Nauczyciele.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp "select N_ID,KOD_POCZTOWY,MIEJSCOWOSC,ULICA,NR_BUDYNKU,NR_LOKALU,TELEFON, PESEL,NR_KONTA,NIP,UWAGI,substring(cast(year(data_ur) as varchar(255)), 1,4)+'/'+cast(month(data_ur) as varchar(255))+'/'+cast(day(data_ur) as varchar(255)) as data_ur from DS_2.dbo.INFO_NAUCZYCIELE" queryout %RESULTDIR%\INFO_NAUCZYCIELE.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp "select U_ID,IMIE_OJCA,IMIE_MATKI,KOD_POCZTOWY,MIEJSCOWOSC,ULICA,NR_BUDYNKU,NR_LOKALU,TELEFON,UWAGI,substring(cast(year(data_ur) as varchar(255)), 1,4)+'/'+cast(month(data_ur) as varchar(255))+'/'+cast(day(data_ur) as varchar(255)) as data_ur from DS_2.dbo.informacje" queryout %RESULTDIR%\INFORMACJE.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.KLASY out %RESULTDIR%\KLASY.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp "select OB_ID,Z_ID,substring(cast(year(data) as varchar(255)), 1,4)+'/'+cast(month(data) as varchar(255))+'/'+cast(day(data) as varchar(255)) as data,USPRAWIEDL,UWAGI from DS_2.dbo.obecnosc" queryout %RESULTDIR%\OBECNOSC.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
```

```
bcp "select o_id, z_id, substring(cast(year(data) as varchar(255)), 1,4)+'/'+cast(month(data) as varchar(255))+'/'+cast(day(data) as varchar(255)) as data, ocena, zakres_materialu, waga from DS_2.dbo.oceny" queryout %RESULTDIR%\OCENY.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.PROWADZA out %RESULTDIR%\PROWADZA.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.PRZYDZIALY out %RESULTDIR%\PRZYDZIALY.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.UZYTKOWNICY out %RESULTDIR%\UZYTKOWNICY.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
bcp %SQLDB%.dbo.ZALICZENIE out %RESULTDIR%\ZALICZENIE.dat /c /t";" /E /S%2 /C1250 %OSQLCONN%
echo STOP: %date%_%time%
pause;
```

- stworzenie schematu bazy danych za pomocą skryptu SQL – przygotowanego w narzędziu Generate Scripts (dostępne z menu kontekstowego bazy danych w sekcji Tasks);
- import danych za pomocą dostępnego w Centrum sterowania systemu DB2 narzędzia Import Data [VISSER 2003]. W celu poprawnego importu danych należało (w przypadku eksportu danych powyższym skryptem) ustawić następujące właściwości importu (przy wybraniu formatu z ogranicznikami):
    * Strona kodowa: „1250”;
    * Ogranicznik kolumny: „;”;
    * Niestandardowy format daty: „YYYY/MM/DD”;
    * Niestandardowy format datownika: „YYYY/MM/DD”.
- ostatnim elementem importu było stworzenie indeksów.

9.4.2. MySQL

Migracja z systemu MS SQL Server 2005 do MySQL była najprostszą (co nie idzie w parze z wydajnością) migracją danych. W tym przypadku zostało wykorzystane bezpłatne narzędzie MySQL Migration Toolkit (wersja 1.1.12) [WWWMYSQL 2008], choć równie dobrze można było skorzystać z polecenia *LOAD DATA* [KOFLER 2005]. Narzędzie to (MySQL Migration Toolkit) umożliwia zdefiniowanie w kilku krokach kreatora migracji pomiędzy różnymi DBMS a MySQL. Jedyną godną uwagi opcją, którą należy zaznaczyć

przy mapowaniu danych jest opcja ‘Multilanguage’, umożliwiająca załadowanie znaków kodowych używanych w kodowaniu 1250.

9.4.3. Oracle

Migracja danych do bazy Oracle wykonywana była przy wykorzystaniu narzędzia bcp przy pobieraniu danych z MS SQL Servera oraz narzędzia sqlldr przy ładowaniu danych do bazy danych Oracle. Cała procedura przebiegała bardzo szybko. Procedura postępowania przedstawia się następująco:

- za pomocą narzędzia SQL Server Import and Export Wizard (instalowanego wraz z MS SQL Server 2005) [ITZIK 2006] należy wykonać eksport struktury bazy danych do systemu Oracle;
- wykonać eksport danych z MS SQL Server 2005 do plików płaskich za pomocą narzędzia bcp (w identyczny sposób jak w przypadku systemu DB2);
- załadować dane z plików płaskich do systemu Oracle za pomocą polecenia sqlldr [URMAN 2003] [ALAPATI 2005]. Ponieważ pliki definicji importu były zapisywane z rozszerzeniem *.ctl możliwe było wykonanie polecenia sqlldr w pętli:

```
@echo off
SET ORAUSER=system
SET ORAPWD=haslo
SET RESULTDIR=.
For %%i in ("%~d0%~p0*.ctl") do sqlldr control="%%i"
                    userid=%ORAUSER%/%ORAPWD%
pause
```

Przykładowy plik definicji importu:

```
load data  infile 'OBECNOSC.dat'
      replace  into table OBECNOSC
      fields terminated by ";" optionally enclosed by '"'
      TRAILING NULLCOLS
      (OB_ID,Z_ID,DATA,USPRAWIEDL,UWAGI)
```

- ostatnim krokiem było wykonanie skryptów, których zadaniem było:
  * dodanie kluczy głównych;
  * dodanie kluczy obcych;

* dodanie indeksów.

9.4.4. PostgreSQL

W przypadku serwera PostgreSQL procedura, pomimo iż wykorzystywała narzędzie importu bezpośrednio wykorzystywane (z okna komend SQL), była dużo bardziej skomplikowana niż podobna (z założenia procedura w przypadku Oracle'a). W postępowaniu zdefiniowano następujące kroki:

- eksport danych z MS SQL Server za pomocą narzędzia bcp;
- edycja wyeksportowanych danych w celu poprawnego wczytania ich poleceniem COPY [MATTHEW 2005]:
  * pliki otworzyć do edycji i zapisać z kodowaniem UTF-8;
  * plikom należy nadać uprawnienia do modyfikacji dla użytkownika root (tworzonego podczas instalacji serwera PostgreSQL);
  * ponieważ puste wartości muszą zostać oznaczone, należy w edytorze tekstu zamienić występowanie znaków ';;' na ';\N;';
  * z tego samego powodu należy w edytorze tekstu umożliwiającym wyszukiwanie i zamianę wyrażeń regularnych (na przykład TextPad) wykonać zamianę ciągu: ';\n' na ';\N\n'.
- stworzenie bazy danych i tabel za pomocą skryptu SQL, z uwzględnieniem:
  * kodowania: UTF-8;
  * przestrzeni tabel: pg_default.
- wykonać skrypt ładujący dane do tabel. Przykład ładowania danych do tabeli nauczyciele: *copy nauczyciele from 'F:/Mirek/agh/pracaMGR/_migracja/PSQL/nauczyciele_utf8.dat' with delimiter ';';*
- wykonać skrypt, który doda:
  * klucze główne;
  * klucze obce;
  * indeksy.

Rozdział X Omówienie otrzymanych rezultatów

Większość testów przeprowadzona była w następującym środowisku (chyba, że zaznaczono inaczej):

- Maszyna: HP Compaq dc7700;
- Procesory: 1x Core 2 Duo 2,13GHz;
- Pamięć fizyczna: 2GB DDR2;
- OS: Microsoft Windows XP Professional (Service Pack 2) x86;
- Dyski: SATA 3,0 Gb/s 160GB.

Przed przeprowadzeniem testów wytypowano zestaw reprezentatywnych zapytań SQL (pytania składają się głównie z operacji selekcji, ze względu na fakt, iż najmniejszą wydajność analizowany system osiągał w czasie raportowym, podczas którego duża liczba użytkowników zadawał zapytania o dużą liczbę danych):

```
select count(ocena) from oceny
select imie from uczniowie
select pr_id from prowadza
select max(n_id)from nauczyciele
select z_id from przedmioty ,prowadza, zaliczenie where przedmioty.p_id=prowadza.p_id and
  prowadza.pr_id = zaliczenie.pr_id and przedmioty.nazwa='matematyka'
select p.pr_id, imie, nazwisko, nazwa from nauczyciele n, prowadza p , przedmioty prz where
  p.p_id = prz.p_id and p.n_id = n.N_id and prz.nazwa like 'jêz%'
select k_id from klasy where nr_klasy = 1 and indeks_klasy = 'A'
select z_id, imie, nazwisko, nazwa from zaliczenie z, uczniowie u,  prowadza pro, przedmioty
  prz where z.u_id=u.u_id and pro.pr_id=z.pr_id and pro.p_id=prz.p_id and u.u_id>50 and
  u.u_id<1543
select imie, nazwisko, nazwa, ocena, data from zaliczenie z, uczniowie u,  prowadza pro,
  przedmioty prz, oceny o where z.u_id=u.u_id and pro.pr_id=z.pr_id and pro.p_id=prz.p_id and
  o.z_id=z.z_id and z.pr_id=2
```

We wszystkich testach konfiguracja narzędzia Performance [MS2790A 2006] systemu Windows była następująca:

- interwał: 1sek;
- format pliku: tekstowy;
- liczniki:
    * LogicalDisk(_Total)\Avg. Disk Queue Length;
    * LogicalDisk(_Total)\Disk Bytes/sec;
    * Memory\Available Mbytes;
    * Memory\Page Faults/sec;
    * Memory\Pages/sec;

    * Network Interface\Bytes Total/sec;
    * Paging File(_Total)\% Usage;
    * PhysicalDisk(_Total)\% Disk Time;
    * Processor(_Total)\% Privileged Time;
    * Processor(_Total)\% Processor Time;
    * System\Context Switches/sec;
    * System\Processor Queue Length.

Do testów skalowania danych wykorzystano następujące bazy danych:

- Baza danych umownie oznaczona DS_1, z następującą ilością danych w wybranych tabelach:
    * uczniowie: 300 wierszy;
    * nauczyciele: 15 wierszy;
    * klasy: 10 wierszy;
    * oceny: 100 wierszy;
    * obecnosc: 150 wierszy.
- Baza danych umownie oznaczona DS_2, z następującą ilością danych w wybranych tabelach:
    * uczniowie: 2545 wierszy;
    * nauczyciele: 222 wierszy;
    * klasy: 67 wierszy;
    * oceny: 162.840 wierszy;
    * obecnosc: 260.527 wierszy.
- Baza danych umownie oznaczona DS_3, z następującą ilością danych w wybranych tabelach:
    * uczniowie: 26.103 wierszy;
    * nauczyciele: 2022 wierszy;
    * klasy: 760 wierszy;
    * oceny: 1.694.510 wierszy;
    * obecnosc: 2.711.199 wierszy.

Testy skalowania danych wykonywane były w następującej konfiguracji narzędzia DBTester:

- ilość powtórzeń serii zapytań: 5;

- ilość wątków: 10;

Do testów skalowania ilości użytkowników wykorzystano bazę DS_2, z następującą konfiguracją narzędzia DBTester:

- ilość powtórzeń serii zapytań: 2;
- ilość wątków: 10, 50 oraz 100.

Do testów skalowania sprzętu (jedynie dla MS SQL Server), wykorzystującej bazę danych DS_3 wykorzystano dodatkowo następujące urządzenia:

- umowne oznaczenie: FS-S2:
  * Maszyna: Fujitsu-Siemens Primergy RX300 S2;
  * Procesory: 2x Dual-Core 3.2 GHz Intel Xeon (Hyper-Threaded);
  * Pamięć fizyczna: 8GB;
  * OS: Microsoft Windows Server 2003 Standard x64Edition (SP 2);
  * Dyski:
    # 6x 146GB SCSI RAID 10k;
    # Macierz Fujitsu-Siemens FibreCAT SX80 3x500GB SATA 7,2k.
- umowne oznaczenie: FS-S3:
  * Maszyna: Fujitsu-Siemens Primergy RX300 S3;
  * Procesory: 2x Quad-Core 2,66 GHz Intel Xeon (R) X5355;
  * Pamięć fizyczna: 16GB;
  * OS: Microsoft Windows Server 2003 Standard x64Edition (SP 2);
  * Dyski:
    # 4x 150GB SAS RAID 15k;
    # Macierz Fujitsu-Siemens FibreCAT SX80 3x500GB SATA 7,2k.

Konfiguracja narzędzia DBTester, przy testach skalowania sprzętu była następująca:

- ilość powtórzeń serii zapytań: 5;
- ilość wątków: 10.

10.1. Stan zero

Przed przystąpieniem do testów wydajnościowych dokonano pomiarów środowiska testowego. Średnie wartości poszczególnych liczników systemowych zawiera poniższa tabela (Tab. 10):

Licznik	Wartość średnia
LogicalDisk(_Total)\Avg. Disk Queue Length	0,00005
LogicalDisk(_Total)\Disk Bytes/sec	2711
Memory\Available Mbytes	1378
Memory\Page Faults/sec	67
Memory\Pages/sec	0
Network Interface\Bytes Total/sec	16.302
Paging File(_Total)\% Usage	1,8
PhysicalDisk(_Total)\% Disk Time	0,00229
Processor(_Total)\% Privileged Time	0,02201
Processor(_Total)\% Processor Time	0,047
System\Context Switches/sec	1743
System\Processor Queue Length	0

Tab.10. Liczniki w stanie zero

10.2. DB2

Testy produktu DB2 przeprowadzane były na wersji 9.5. Bardzo pozytywnym aspektem tego systemu w porównaniu z innymi dużymi produktami (przede wszystkim MS SQL Server oraz Oracle), był fakt bardzo szybkiego startu serwisów systemu. Średnie wartości poszczególnych liczników w zależności od typu skalowania (dane, połączenia) zawierają tabele w kolejnych podrozdziałach.

10.2.1. DB2 – skalowanie danych

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla trzech testowanych wielkości baz danych zawiera poniższa tabela (Tab. 11).

Warto zauważyć, iż wyniki dla bazy DS_2 znacznie odbiegają od wyników DS_1 oraz DS_3 – i są o rząd wielkości większe. Zauważona podczas testów, że niezależnie od wielkości bazy danych często na jednym z pierwszych zapytań SQL wysłanych przez aplikację testującą serwer DB2 blokuje pozostałe połączenia i dość długo generuje odpowiedź na zapytanie. W momencie zwrócenia odpowiedzi, kolejne zapytania wykonują się w standardowym czasie. Ponieważ w innych serwerach bazodanowych ani razu nie stwierdzono takiego problem, a w DB2 problem ten często się powtarzał uznano za stosowne umieszczenie przynajmniej w jednej ze statystyk pomiarów z takiego przypadku.

Prawdopodobną przyczyną występowania tego zjawiska był długi czas ustanawiania pierwszego połączenia do bazy danych, spowodowany alokowaniem zasobów (po uprzednim, automatycznym ich zwolnieniu po okresie braku aktywności) [SCOTT 2005]. Występowaniu tego zjawiska można zapobiec, pozostawiając wybrane bazy danych w stanie czuwania (polecenie: *db2 ACTIVATE DATABASE <nazwa_bazy_danych>*) [SCOTT 2005].

Licznik	**Wartość średnia dla bazy DS_1**	**Wartość średnia dla bazy DS_2**	**Wartość średnia dla bazy DS_3**
LogicalDisk(_Total)\Avg. Disk Queue Length	0,048	4,723	0,024
LogicalDisk(_Total)\Disk Bytes/sec	872.821	3.514.486	721.181
Memory\Available Mbytes	997	1122	942
Memory\Page Faults/sec	90	543	96
Memory\Pages/sec	2,489	0,679	1,261
Network Interface\Bytes Total/sec	44324	38.000	46.664
Paging File(_Total)\% Usage	2,548	2,542	2,54
PhysicalDisk(_Total)\% Disk Time	4,795	472	2,364
Processor(_Total)\% Privileged Time	3,155	2,194	4,047
Processor(_Total)\% Processor Time	7,288	6,333	53,506
System\Context Switches/sec	4265	3692	3107
System\Processor Queue Length	0	0,025	2,349
Łączny czas odpowiedzi RDBMS [sec]	45,031	111,234	54,188

Tab.11. DB2 – skalowanie danych

10.2.2. DB2 – skalowanie użytkowników

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi

na ostatnie zapytanie) dla różnej ilości połączeń do bazy danych przedstawia poniższa tabela (Tab. 12).

Licznik	Wartość średnia dla 10 połączeń	Wartość średnia dla 50 połączeń	Wartość średnia dla 100 połączeń
LogicalDisk(_Total)\Avg. Disk Queue Length	4,424	6,141	6,161
LogicalDisk(_Total)\Disk Bytes/sec	3.232.468	4.091.090	4.307.411
Memory\Available Mbytes	1.117	1.109	1.102
Memory\Page Faults/sec	446	500	537
Memory\Pages/sec	1,477	0,362	0,206
Network Interface\Bytes Total/sec	35.182	39.703	40.861
Paging File(_Total)\% Usage	2,519	2,519	2,516
PhysicalDisk(_Total)\% Disk Time	455	616	620
Processor(_Total)\% Privileged Time	2,263	2,508	2,865
Processor(_Total)\% Processor Time	5,820	5,585	11,046
System\Context Switches/sec	3.317	3.615	3.645
System\Processor Queue Length	0	0	0,049
Łączny czas odpowiedzi RDBMS (sek)	45,360	207,584	418,484

Tab.12. DB2 – skalowanie połączeń

10.3. MS SQL Server

Testy serwera Microsoft MS SQL Server przeprowadzane były na wersji 9.0.3042 (wersja 2005 z dodatkiem Service Pack 2). Średnie wartości poszczególnych liczników w zależności od typu skalowania (dane, połączenia, sprzęt) zawierają tabele w kolejnych podrozdziałach.

10.3.1. MS SQL Server – skalowanie danych

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla trzech testowanych wielkości baz danych zawiera poniższa tabela (Tab. 13).

Licznik	Wartość średnia dla bazy DS_1	Wartość średnia dla bazy DS_2	Wartość średnia dla bazy DS_3
LogicalDisk(_Total)\Avg. Disk Queue Length	0,0002	0,141	0,028
LogicalDisk(_Total)\Disk Bytes/sec	57.061	762.412	172.603
Memory\Available Mbytes	1.211	1.194	1.066
Memory\Page Faults/sec	278	543	526
Memory\Pages/sec	3,790	3,617	1,707
Network Interface\Bytes Total/sec	114.667	108.860	58.333
Paging File(_Total)\% Usage	2,590	2,628	2,597
PhysicalDisk(_Total)\% Disk Time	0,022	6,816	2,760
Processor(_Total)\% Privileged Time	2,788	2,610	1,221
Processor(_Total)\% Processor Time	12,881	36,332	75
System\Context Switches/sec	3.884	3.248	3.405
System\Processor Queue Length	0	0,182	1,306
Łączny czas odpowiedzi RDBMS (sek)	11,188	13,859	41,234

Tab.13. MS SQL Server – skalowanie danych

10.3.2. MS SQL Server – skalowanie użytkowników

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla różnej ilości połączeń do bazy danych przedstawia poniższa tabela (Tab. 14).

Licznik	Wartość średnia dla 10 połączeń	Wartość średnia dla 50 połączeń	Wartość średnia dla 100 połączeń
LogicalDisk(_Total)\Avg. Disk Queue Length	0,001	0,003	0,001
LogicalDisk(_Total)\Disk Bytes/sec	88.036	83.239	48.799
Memory\Available Mbytes	1.073	1.022	975
Memory\Page Faults/sec	535	1.008	558
Memory\Pages/sec	4,696	2,217	1,515
Network Interface\Bytes Total/sec	63.774	130.349	157.674
Paging File(_Total)\% Usage	2,558	2,557	2,556
PhysicalDisk(_Total)\% Disk Time	0,068	0,291	0,130
Processor(_Total)\% Privileged Time	1,399	3,530	4,000
Processor(_Total)\% Processor Time	22,543	43,269	49,678
System\Context Switches/sec	2.878	6.544	4.802

System\Processor Queue Length	0	0,194	0,379
Łączny czas odpowiedzi RDBMS (sek)	6,828	26,672	49,000

Tab.14. MS SQL Server – skalowanie połączeń

10.3.3. MS SQL Server – skalowanie sprzętu

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla trzech różnych środowisk zawiera poniższa tabela (Tab. 15).
Należy zwrócić uwagę na fakt, iż na maszynach FS-S2 oraz FS-S3 były uruchomione inne aplikacje – co oznacza, że wartości liczników nie oddają jedynie obciążenia samym systemem operacyjnym oraz systemem zarządzania bazą danych, lecz również na systemie pracowały inne aplikacje (w przeciwieństwie do danych dla maszyny HPdc7700).

Licznik	**HPdc7700 (DS_3)**	**FS-S2 (DS_3)**	**FS-S3 (DS_3)**
LogicalDisk(_Total)\Avg. Disk Queue Length	0,028	0,119	0,010
LogicalDisk(_Total)\Disk Bytes/sec	172.603	267.496	182.292
Memory\Available Mbytes	1.066	1.231	3.514
Memory\Page Faults/sec	526	748	500
Memory\Pages/sec	1,707	0,106	2,276
Network Interface\Bytes Total/sec	58.333	b.d.	b.d.
Paging File(_Total)\% Usage	2,597	8,441	31,654
PhysicalDisk(_Total)\% Disk Time	2,760	2,983	0,169
Processor(_Total)\% Privileged Time	1,221	3,857	1,935
Processor(_Total)\% Processor Time	75	79,567	35,169
System\Context Switches/sec	3.405	10.044	22.720
System\Processor Queue Length	1,306	3,536	0,84
Łączny czas odpowiedzi RDBMS (sek)	41,234	59,094	13,828

Tab.15. MS SQL Server – skalowanie sprzętu

10.4. MySQL

Testy systemu zarządzania bazą danych MySQL przeprowadzane były na wersji 5.0.46. Średnie wartości poszczególnych liczników w zależności od typu skalowania (dane, połączenia) zawierają tabele w kolejnych podrozdziałach.

10.4.1. MySQL – skalowanie danych

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla trzech testowanych wielkości baz danych zawiera poniższa tabela (Tab. 16).

Dość istotnym problemem, który się pojawił w trakcie trwania testów (jedynie dla bazy DS_3), to kilkakrotnie występujący błąd „java.lang.OutOfMemeoryError: Java heap space", którego przyczyną było przekroczenie 128MB pamięci przeznaczonej na stos Java. Oczywiście błędu można uniknąć zwiększając domyślne ograniczenie pamięci (inicjalne=32m oraz maksymalne=128m) poleceniem:

Java –Xms< initial heap size> -Xmx<maximum heap size>

Licznik	Wartość średnia dla bazy DS_1	Wartość średnia dla bazy DS_2	Wartość średnia dla bazy DS_3
LogicalDisk(_Total)\Avg. Disk Queue Length	0,002	0,174	2,167
LogicalDisk(_Total)\Disk Bytes/sec	72.478	557.453	2.789.078
Memory\Available Mbytes	1.283	1.270	1.237
Memory\Page Faults/sec	3.028	1.571	86,215
Memory\Pages/sec	5,220	2,943	0,789
Network Interface\Bytes Total/sec	262.386	3.250.391	304.804
Paging File(_Total)\% Usage	2,538	2,538	2,573
PhysicalDisk(_Total)\% Disk Time	0,176	17,371	218
Processor(_Total)\% Privileged Time	1,836	1,853	0,956
Processor(_Total)\% Processor Time	22,445	30,365	7,164
System\Context Switches/sec	2.185	2.978	4.294
System\Processor Queue Length	0	0,259	0,048
Łączny czas odpowiedzi RDBMS (sek)	8,156	16,891	488,359

Tab.16. MySQL – skalowanie danych

10.4.2. MySQL – skalowanie użytkowników

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla różnej ilości połączeń do bazy danych przedstawia poniższa tabela (Tab. 17).

Licznik	Wartość średnia dla 10 połączeń	Wartość średnia dla 50 połączeń	Wartość średnia dla 100 połączeń
LogicalDisk(_Total)\Avg. Disk Queue Length	0,015	0,0002	0,0005
LogicalDisk(_Total)\Disk Bytes/sec	6.9631	38.145	57.284
Memory\Available Mbytes	1.253	1.254	1.254
Memory\Page Faults/sec	790	1.624	1.997
Memory\Pages/sec	4,426	1,943	9,787
Network Interface\Bytes Total/sec	1.957.078	4.265.325	5.212.506
Paging File(_Total)\% Usage	2,512	2,508	2,513
PhysicalDisk(_Total)\% Disk Time	1,400	0,024	0,051
Processor(_Total)\% Privileged Time	0,944	2,381	2,448
Processor(_Total)\% Processor Time	22,969	37,609	43,605
System\Context Switches/sec	2.317	3.375	4.056
System\Processor Queue Length	0,278	0,463	0,642
Łączny czas odpowiedzi RDBMS (sek)	7,390	32,109	58,516

Tab.17. MySQL – skalowanie połączeń

10.5. Oracle

Testy serwera Oracle przeprowadzane były na wersji 10.2.0. Średnie wartości poszczególnych liczników w zależności od typu skalowania (dane, połączenia) zawierają tabele w kolejnych podrozdziałach.

10.5.1. Oracle – skalowanie danych

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi

na ostatnie zapytanie) dla trzech testowanych wielkości baz danych zawiera poniższa tabela (Tab. 18).

Licznik	Wartość średnia dla bazy DS_1	Wartość średnia dla bazy DS_2	Wartość średnia dla bazy DS_3
LogicalDisk(_Total)\Avg. Disk Queue Length	0,009	0,0296	0,0098
LogicalDisk(_Total)\Disk Bytes/sec	74.353	387.040	119.982
Memory\Available Mbytes	1.043	999	685
Memory\Page Faults/sec	5.286	10.994	9.361
Memory\Pages/sec	3,459	3,459	2,946
Network Interface\Bytes Total/sec	223.314	225.597	193.232
Paging File(_Total)\% Usage	2,563	2,562	2,529
PhysicalDisk(_Total)\% Disk Time	0,901	2,958	0,986
Processor(_Total)\% Privileged Time	10,599	12,484	10,533
Processor(_Total)\% Processor Time	21,032	26,317	35,185
System\Context Switches/sec	4.452	4.777	4.923
System\Processor Queue Length	0	0,130	0,222
Łączny czas odpowiedzi RDBMS (sek)	14,437	14,547	16,375

Tab.18. Oracle – skalowanie danych

10.5.2. Oracle – skalowanie użytkowników

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla różnej ilości połączeń do bazy danych przedstawia poniższa tabela (Tab. 19).

Licznik	Wartość średnia dla 10 połączeń	Wartość średnia dla 50 połączeń	Wartość średnia dla 100 połączeń
LogicalDisk(_Total)\Avg. Disk Queue Length	0,047	0,0077	0,004
LogicalDisk(_Total)\Disk Bytes/sec	183.138	129.376	92.681
Memory\Available Mbytes	949	923	922
Memory\Page Faults/sec	7.650	14.038	922
Memory\Pages/sec	6,003	2,280	1,369
Network Interface\Bytes Total/sec	136.562	292.044	343.745
Paging File(_Total)\% Usage	2,521	2,521	2,521
PhysicalDisk(_Total)\% Disk Time	4,708	0,794	0,427

Processor(_Total)\% Privileged Time	9,052	16,211	19,427
Processor(_Total)\% Processor Time	23,785	32,540	37,242
System\Context Switches/sec	4.963	6.464	6.938
System\Processor Queue Length	0,125	0,171	0,170
Łączny czas odpowiedzi RDBMS (sek)	7,485	26,485	51,25

Tab.19. Oracle – skalowanie połączeń

10.6. PostgreSQL

Testy serwera PostgreSQL Server przeprowadzane były na wersji 8.2.5-1. Średnie wartości poszczególnych liczników w zależności od typu skalowania (dane, połączenia) zawierają tabele w kolejnych podrozdziałach.

10.6.1. PostgreSQL – skalowanie danych

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla trzech testowanych wielkości baz danych zawiera poniższa tabela (Tab. 20).

Licznik	Wartość średnia dla bazy DS_1	Wartość średnia dla bazy DS_2	Wartość średnia dla bazy DS_3
LogicalDisk(_Total)\Avg. Disk Queue Length	0,019	0,006	0,436
LogicalDisk(_Total)\Disk Bytes/sec	149.074	377.620	4.363.429
Memory\Available Mbytes	1.289	1.280	1.224
Memory\Page Faults/sec	33.387	33.706	38.258
Memory\Pages/sec	10,170	68,261	352
Network Interface\Bytes Total/sec	66.522	7.146.709	9.390.294
Paging File(_Total)\% Usage	2,554	2,712	2,638
PhysicalDisk(_Total)\% Disk Time	2,036	0,617	44,084
Processor(_Total)\% Privileged Time	26,283	20,350	15,467
Processor(_Total)\% Processor Time	37,535	47,814	58,453
System\Context Switches/sec	4.561	3.499	3.122
System\Processor Queue Length	0	0,323	1,670
Łączny czas odpowiedzi RDBMS (sek)	13,125	21,297	82,64

Tab.20. PostgreSQL – skalowanie danych

10.6.2. PostgreSQL – skalowanie użytkowników

Zestawienie średnich wartości liczników systemowych oraz czasy, jakie zajmował pełny test (a więc czas od przesłania pierwszego zapytania SQL do otrzymania odpowiedzi na ostatnie zapytanie) dla różnej ilości połączeń do bazy danych przedstawia poniższa tabela (Tab. 21).

Licznik	Wartość średnia dla 10 połączeń	Wartość średnia dla 50 połączeń	Wartość średnia dla 100 połączeń
LogicalDisk(_Total)\Avg. Disk Queue Length	0,004	0,002	0,009
LogicalDisk(_Total)\Disk Bytes/sec	404.924	95.553	106.638
Memory\Available Mbytes	1.276	1.273	1.277
Memory\Page Faults/sec	21.820	38.064	44.212
Memory\Pages/sec	80,228	3,252	4,005
Network Interface\Bytes Total/sec	4.664.673	8.180.598	9.354.666
Paging File(_Total)\% Usage	3,016	3,105	3,220
PhysicalDisk(_Total)\% Disk Time	0,406	0,209	0,922
Processor(_Total)\% Privileged Time	13,623	24,023	27,768
Processor(_Total)\% Processor Time	35,010	52,612	60,038
System\Context Switches/sec	2.803	3.644	4.009
System\Processor Queue Length	0,211	0,519	0,645
Łączny czas odpowiedzi RDBMS (sek)	9,109	42,781	83,765

Tab.21. PostgreSQL – skalowanie połączeń

10.7. Porównanie otrzymanych wyników

Porównanie otrzymanych wyników przeprowadzonych testów wydajnościowych wybranych DBMS, zostanie oparte głównie o czas odpowiedzi na zadane serię zapytań w danym środowisku. W szczególnych przypadkach zostaną również przytoczone wartości wybranych liczników stanu systemu.

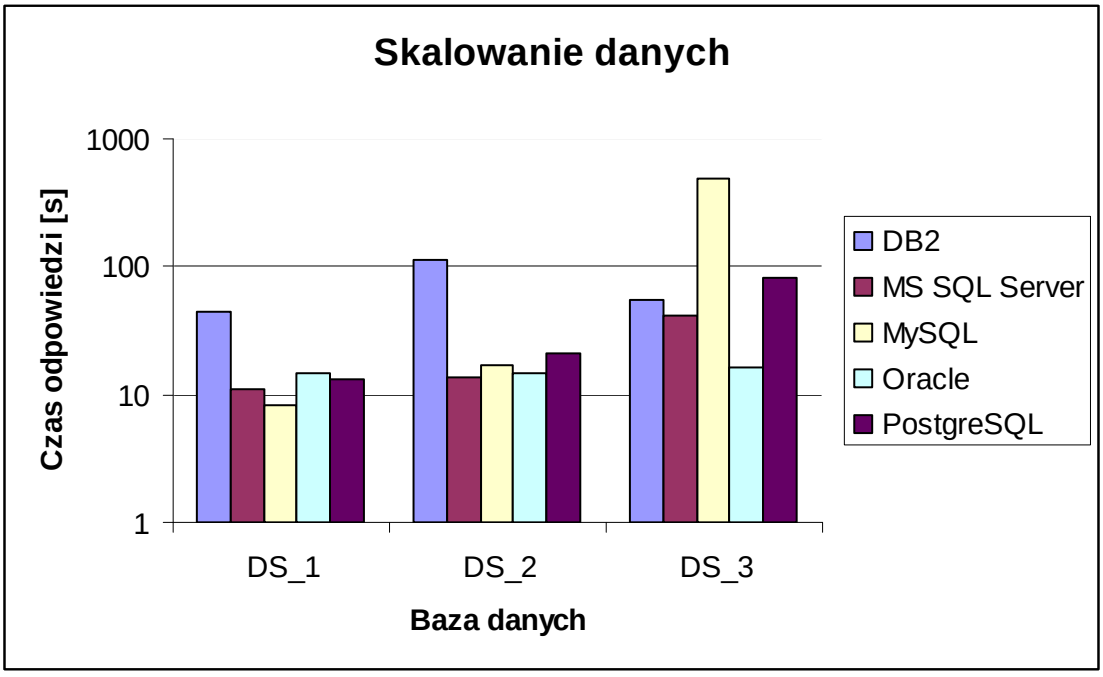
Analiza porównawcza wyników (w danej kategorii) dzieli się na trzy części:

- spostrzeżenia – najważniejsze uwagi podsumowujące wyniki z danego testu;
- zalecenia – wskazanie najlepszego / najlepszych rozwiązań;
- uwagi – ostrzeżenie przed złymi rozwiązaniami.

Omawiając wyniki, zostało założone, iż celem analizy jest wybranie najlepszego z wymienionych RDBMS (bez względu na ograniczenia ekonomiczne, biznesowe czy standaryzujące). Również założono, iż wskazany DBMS będzie wykorzystywany podczas całego cyklu życia systemu informatycznego, w skład którego wchodzi analizowana baza danych. Zaprezentowane dane można jednak analizować pod względem migracyjnym systemów w zależności od przyrostu danych czy użytkowników. Można zatem założyć początkowy stan systemu i wybrać dla niego najlepszy (pod względem wydajnościowym, biznesowym i ekonomicznym) serwer bazodanowy, oraz analizować możliwość migracji do najlepszego systemu bazodanowego w przypadku wystąpienia problemów wydajnościowych (przyrost danych czy użytkowników). Takiej analizy niniejsza praca nie zawiera.

10.7.1. Skalowanie danych

Jak wspomniano wcześniej testy skalowania danych, czyli badania zachowania systemu zarządzania bazą danych konkretnego dostawcy przeprowadzano w oparciu o trzy bazy danych (DS_1, DS_2 i DS_3), w których kolejno wzrastała ilość danych we wszystkich tabelach. Wszystkie testy prowadzono przy dla dziesięciu równoległych połączeń bazą danych. Prezentację graficzną otrzymanych wyników zawiera Rys.37.



Rys.37. Skalowanie danych – porównanie wyników

Aby możliwe było omówienie wyników przedstawionych na Rys.37, dane zestawiono w poniższej tabeli (Tab. 22).

RDBMS\DB	DS_1 czas odpowiedzi [sek.]	DS_2 czas odpowiedzi [sek.]	DS_3 czas odpowiedzi [sek.]	SUMA
DB2	45,031	111,234*	54,188	210,453
MS SQL Server	11,188	13,859	41,234	66,281
MySQL	8,156	16,891	488,359	513,406
Oracle	14,437	14,547	16,375	45,359
PostgreSQL	13,125	21,297	82,64	117,062

Tab.22. Skalowanie danych – porównanie wyników

* - przyczynę tak długiego czasu odpowiedzi opisano w rozdziale 10.2.1

Wnioski na podstawie zestawienia czasów odpowiedzi na zapytania:

a) spostrzeżenia:

- najbardziej wydajnym systemem bazodanowym dla wszystkich trzech wielkości baz danych jest Oracle;
- drugie miejsce pod względem wydajności zajął MS SQL Server;
- pomimo relatywnie niewielkiej różnicy sumarycznego czasu pomiędzy MS SQL Server a Oracle należy zwrócić uwagę na fakt, iż przy małych i średnich bazach danych (DS_1, DS_2) czasy odpowiedzi obu serwerów są zbliżone, natomiast dla dużej bazy danych (DS_3) odpowiedź MS SQL Server'a jest prawie trzykrotnie dłuższa niż odpowiedź Oracle (dla którego notabene czas odpowiedzi dla wszystkich baz jest zbliżony);
- zły wynik produktu DB2 spowodowany jest wysokim czasem odpowiedzi dla bazy DS_2. Można przypuszczać, iż jeśli udałoby się rozwiązać problem zdarzających się długich czasów odpowiedzi na pierwsze zapytanie, sumaryczny czas odpowiedzi wynosiłby w przybliżeniu 150s i byłby najgorszym wynikiem spośród komercyjnych systemów a nawet gorszym od PostgreSQL.

Ponadto widać, że dla małych baz danych DB2 nie jest dobrym produktem.

b) zalecenia:

- jeśli projektowana baza danych w wyniku swojego życia może osiągnąć wielkość zbliżoną do DS_3 (lub ją przekroczyć), to zdecydowanie należy wybrać serwer Oracle;
- jeśli projektowana baza danych nie przekroczy wielkością bazy DS_2 to można skorzystać z produktów OSS, a w szczególności z produktu MySQL, który dla małych baz (DS_1) jest najwydajniejszym produktem. Jeśli jednak jest możliwy wzrost danych ponad wielkość średniej bazy należy wybrać PostgreSQL.

c) uwagi:

- jeśli baza danych może osiągnąć wielkość DS_3 nie należy wybierać produktów OSS, a w szczególności MySQL. Jasno widać, że przy dużych ilościach danych system ten znacznie traci na wydajności;
- jeśli baza danych może osiągnąć duże rozmiary nie należy wybierać spośród produktów komercyjnych produktu DB2.

Ponieważ interesujące są wyniki testów dla SQL Server'a oraz Oracle'a, poniżej zestawiono wartości wybranych liczników dla tychże produktów (Tab. 23), ograniczając je do bazy danych DS_3:

Licznik	**MS SQL Server (DS_3)**	**Oracle (DS_3)**	**Zalecane wartości**
LogicalDisk(_Total)\Avg. Disk Queue Length	0,028	0,0098	<1,5
LogicalDisk(_Total)\Disk Bytes/sec	172.603	119.982	<10.000.000
Memory\Available Mbytes	1.066	685	>200
Memory\Page Faults/sec	526	9.361	<100
Memory\Pages/sec	1,707	2,946	<20
Paging File(_Total)\% Usage	2,597	2,529	<70
PhysicalDisk(_Total)\% Disk Time	2,760	0,986	<90
Processor(_Total)\% Privileged Time	1,221	10,533	<10
Processor(_Total)\% Processor Time	75	35,185	<80
System\Context Switches/sec	3.405	4.923	<2.000
System\Processor Queue Length	1,306	0,222	<20

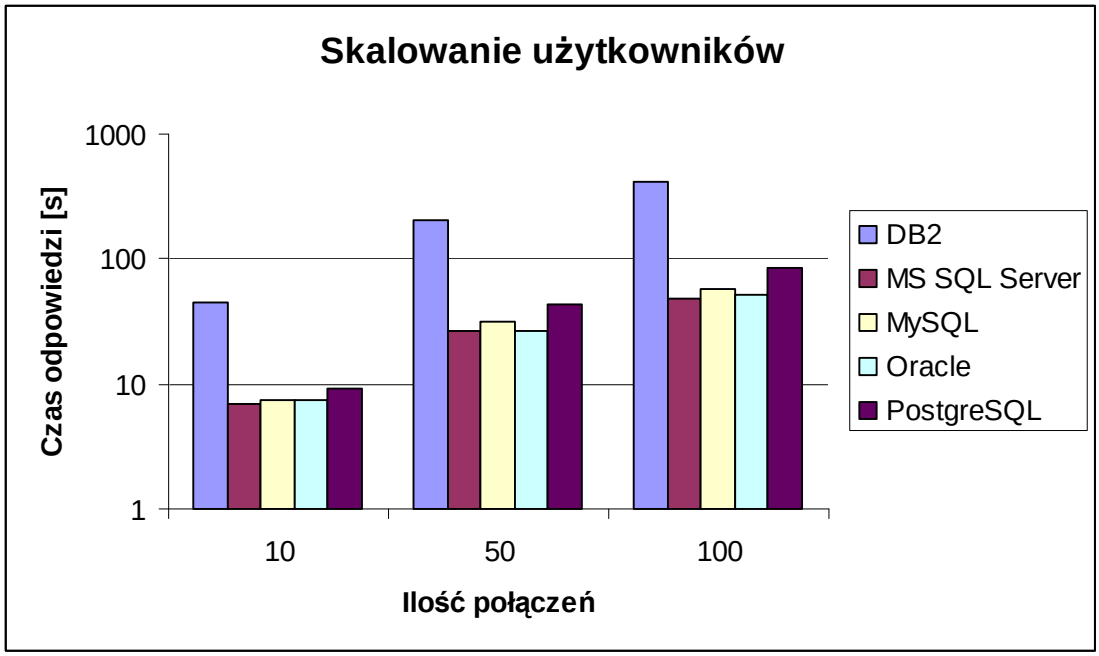
Tab.23. Porównanie wybranych liczników dla SQL Server i Oracle

Z powyższego zestawienia można wysnuć następujące wnioski:

- Oracle jest bardziej wymagającym systemem pod względem pamięci. Znaczne przekroczenie zalecanej wartości licznika Memory\Page Faults/sec oznacza konieczność dołożenia pamięci fizycznej dla maszyny (w przypadku Oracle ilość pamięci musi być większa niż w przypadku SQL Server'a);
- SQL Server zbliżył się do krytycznej granicy obciążenia procesora, co może sugerować konieczność dodanie kolejnych procesorów. Z kolei Oracle nieznacznie przekroczył odsetek pracy wątków w trybie uprzywilejowanym;
- można wnioskować, iż niewielkim kosztem – dodanie pamięci fizycznej można zwiększyć wydajność serwera Oracle, natomiast w przypadku SQL Servera wydajność może zwiększyć dodanie procesorów.

10.7.2. Skalowanie użytkowników

Jak wspomniano wcześniej testy skalowania użytkowników (ilości połączeń do serwera), czyli badania zachowania systemu zarządzania bazą danych konkretnego dostawcy przeprowadzano w oparciu o trzy wartości ilości połączeń (10, 50 i 100). Badania przeprowadzano na bazie danych ze średnią ilością danych (DS_2). Prezentację graficzną otrzymanych wyników zawiera Rys.38.



Rys. 38. Skalowanie użytkowników – porównanie wyników

Aby możliwe było omówienie wyników przedstawionych na Rys.38, dane zestawiono w poniższej tabeli (Tab. 24).

RDBMS\Połączenia	10 połączeń czas odpowiedzi [s] / czas/1 połączenie	50 połączeń czas odpowiedzi [s] / czas/1 połączenie	100 połączeń czas odpowiedzi [s] / czas/1 połączenie	SUMA
DB2	45,360 4,536	207,584 4,152	418,484 4,185	671,428
MS SQL Server	6,828 0,683	26,672 0,533	49,000 0,490	82,5
MySQL	7,390 0,739	32,109 0,642	58,516 0,585	98,015
Oracle	7,485 0,749	26,485 0,530	51,25 0,513	85,22
PostgreSQL	9,109 0,911	42,781 0,856	83,765 0,838	135,655

Tab.24. Skalowanie użytkowników – porównanie wyników

Wnioski na podstawie zestawienia czasów odpowiedzi na zapytania:

a) spostrzeżenia:

- tak jak w przypadku testów związanych ze skalowaniem danych najlepsze wyniki osiągnęły produkty: SQL Server oraz Oracle, tym razem niewielką przewagą produktu firmy Microsoft. Zważając jednak na fakt, iż dla bazy DS_2, czas odpowiedzi produktu Microsoft był nieznacznie lepszy od Oracle, można przypuszczać, iż jeśli testy przeprowadzono by na większej bazie (DS_3), Oracle znacznie wyprzedziłby produkt Microsoft'u;
- produkt OSS – MySQL nieznacznie odbiega od komercyjnych produktów (warto tu jednak przypomnieć, iż testy były przeprowadzane na bazie danych średniej wielkości i z pewnością dla większych baz wyniki byłyby gorsze);
- we wszystkich systemach wraz ze wzrastająca liczbą połączeń czas obsługi przypadający na jedno połączenie maleje;
- DB2 wypadł w testach nieproporcjonalnie słabo w porównaniu z pozostałymi systemami. Może to być spowodowane

przystosowaniem systemu do pracy na dużych bazach danych. W przypadku małych narzut na zarządzanie systemem jest zbyt duży. Podobnie zachowuje się Oracle dla najmniejszej bazy DS_1, dla której ma w porównaniu do innych systemów długi czas. Natomiast wraz ze wzrostem ilości danych obsługa tych samych zapytań tylko w nieznaczny sposób jest spowolniona przyrastającą ilością danych.

b) zalecenia:

- dla średniej wielkości baz zalecane jest użycie produktów MS SQL Server bądź Oracle. Jednak w przypadku możliwego wzrostu ilości danych zalecane jest użycie serwera Oracle;
- rozwiązanie OSS – produkt MySQL odbiega od najlepszych rozwiązań komercyjnych o 20%. Jeżeli najwyższa wydajność nie jest wymagana warto zastanowić się nad tym rozwiązaniem.

c) uwagi:

- zdecydowanie nie jest zalecane użycie systemu DB2;
- nie poleca się również przy liczbie połączeń większej niż 10 produktu PostgreSQL.

Ponieważ interesujące są wyniki testów dla SQL Server'a oraz Oracle'a, poniżej zestawiono wartości wybranych liczników dla tych produktów (Tab. 25), ograniczając je do wyników dla 100 połączeń:

Licznik	**MS SQL Server (100 połączeń)**	**Oracle (100 połączeń)**	**Zalecane wartości**
LogicalDisk(_Total)\Avg. Disk Queue Length	0,001	0,004	<1,5
LogicalDisk(_Total)\Disk Bytes/sec	48.799	92.681	<10.000.000
Memory\Available Mbytes	975	922	>200
Memory\Page Faults/sec	558	922	<100
Memory\Pages/sec	1,515	1,369	<20
Paging File(_Total)\% Usage	2,556	2,521	<70
PhysicalDisk(_Total)\% Disk Time	0,130	0,427	<90
Processor(_Total)\% Privileged Time	4,000	19,427	<10
Processor(_Total)\% Processor Time	49,678	37,242	<80
System\Context Switches/sec	4.802	6.938	<2.000
System\Processor Queue Length	0,379	0,170	<20

Tab.25. Porównanie wybranych liczników dla SQL Server i Oracle

Tak jak w przypadku analizy liczników dla skalowania danych, tak i w tym przypadku widać, że problemem jest pamięć (również bardziej odczuwalna w przypadku produktu Oracle) oraz czas procesora pracującego w trybie uprzywilejowanym. Różnicą w stosunku do wcześniejszego zestawienia liczników jest obciążenie procesora dla SQL Servera – widać, że lepiej sobie radzi w przypadku zwiększającej się liczby połączeń niż w przypadku zwiększającej się liczby danych.

10.7.3. Skalowanie sprzętu

Testy wydajnościowe w zależności od sprzętu przeprowadzono (ze względów na ograniczone możliwości techniczne) jedynie dla serwera bazodanowego MS SQL Server, w trzech omawianych wcześniej konfiguracjach sprzętowych (HPdc7700, FS-S2, FS-S3). Wszystkie testy przeprowadzano dla największej przygotowanej bazy danych (DS_3). Warto w tym miejscu przypomnieć, iż na urządzeniach FS-S2 i FS-S3 oprócz niezbędnych procesów systemowych i systemu zarządzania bazą danych uruchomione były również inne aplikacje.

Z tego względu porównując wyniki (Tab. 26) zostaną uwzględnione wartości liczników w stanie zerowym (tzn., sprzed uruchomienia narzędzia testującego DBTester). Różnica w stanie zerowym dla środowiska HP w odróżnieniu od środowisk FS-S2/3 polega na tym, że pomiary stanu zerowego dla HP dokonywano przed uruchomieniem systemu zarządzania bazą danych, natomiast dla sprzętów z rodziny Fujitsu-Siemens pomiarów zerowych dokonywano przy uruchomionym serwerze bazodanowym.

Licznik	**HPdc7700 (ZERO)**	**HPdc7700 (DS_3)**	**FS-S2 (ZERO)**	**FS-S2 (DS_3)**	**FS-S3 (ZERO)**	**FS-S3 (DS_3)**	**Wartości zalecane**
LogicalDisk(_Total)\Avg. Disk Queue Length	0,00005	0,028	0,045	0,119	0,008	0,010	<1,5 (HP) <6 (S2iS3)
LogicalDisk(_Total)\Disk Bytes/sec	2711	172.603	44.048	267.496	81.396	182.292	<10.000.000
Memory\Available Mbytes	1378	1.066	1.287	1.231	3.527	3.514	>200 (HP) >800 (S2) >1.600 (S3)
Memory\Page Faults/sec	67	526	533	748	376	500	<100
Memory\Pages/sec	0	1,707	0,421	0,106	0,670	2,276	<20
Paging File(_Total)\% Usage	1,8	2,597	8,449	8,441	31,709	31,654	<70

PhysicalDisk(_Total)\% Disk Time	0,00229	2,760	1,132	2,983	0,139	0,169	<90
Processor(_Total)\% Privileged Time	0,02201	1,221	1,359	3,857	0,751	1,935	<10
Processor(_Total)\% Processor Time	0,047	75	2,913	79,567	2,275	35,169	<80
System\Context Switches/sec	1743	3.405	6.231	10.044	9.672	22.720	<2.000 (HP) <4.000 (S2) <8.000 (S3)
System\Processor Queue Length	0	1,306	0,067	3,536	0	0,84	<20 (HP) <40 (S2) <80 (S3)
Łączny czas odpowiedzi RDBMS (s)	Nie dotyczy	41,234	Nie dotyczy	59,094	Nie dotyczy	13,828	Nie dotyczy

Tab.26. Porównanie wybranych liczników w zależności od maszyn

Wnioski na podstawie zestawienia czasów odpowiedzi na zapytania:

a) spostrzeżenia:

- im potężniejszy sprzęt tym wydajność coraz lepsza (ograniczeniem są możliwości finansowe, ograniczenia sprzętu oraz software);
- wyniki dla FS-S2 są gorsze od wyników dla HP, jednak widać po wartości liczników w stanie zero, iż serwer wykonywał inne zadania, które obciążały zarówno pamięć jak i dysk;
- porównując czasy dla HP oraz FS-S2 można zauważyć, iż przy stosunku pamięci 1:4 oraz procesora 1:4 i prędkości dysków około 1:2, wydajność wzrasta bardziej niż liniowo. Co więcej spada znacznie wartość licznika Memory\Page Faults/sec, co oznacza, iż znaczny wpływ na wydajność ma wielkość pamięci fizycznej.

b) zalecenia:

- wskazane jest zastosowanie serwera FS-S3, o ile wymagana jest tak wysoka wydajność (serwer mógłby współdzielić zasoby z innymi aplikacjami), bądź przetestować serwer FS-S2 dedykowany dla testowanego systemu.

c) uwagi:

- nie jest wskazane zastosowanie serwera FS-S2, który nie byłby dedykowany tylko do omawianego systemu. W przeciwnym razie

koszty sprzętu i oprogramowania będą bardzo wysokie, a wydajność mniejsza niż w przypadku zastosowania dedykowanej stacji HPdc7700.

10.8. Wnioski

Przeprowadzone testy dla wybranych systemów zarządzania bazą danych odbywały się przy zmieniającym się środowisku, w skład którego wchodziły:

- baza danych (wielkość);
- połączenia do bazy danych (ilość);
- sprzęt (zasoby).

Do niezmiennych elementów środowiska testowego należały:

- rodzina systemów operacyjnych (Windows);
- narzędzie testujące (DBTester);
- zestaw pytań testujących.

Przeprowadzone testy wykazały, iż dla zadanego zestawu reprezentatywnych zapytań SQL, w środowisku Windows, przy wyżej wymienionych warunkach:

- produkty MS SQL Server oraz Oracle okazały się najbardziej wydajnymi produktami (z przewagą serwera Oracle przy dużej ilości danych). Warto w tym miejscu zaznaczyć, iż produkt SQL Server jest dedykowany (a właściwie ograniczony) do systemu operacyjnego z rodziny Windows. Z kolei produkt Oracle jest dedykowany do systemu Linux, co przy zastosowaniu podobnej konfiguracji sprzętowej może spowodować wzrost wydajności serwera (stwierdzenie jest również poparte przez statystyki benchmark TPC (rozdział 6.1);
- produkt OSS odbiegają od produktów komercyjnych, z czego MySQL znacznie odbiega przy pracy na dużej ilości danych, oraz nieznacznie dla dużej liczby połączeń;
- skalowanie sprzętu pozwala w sposób zbliżony do liniowego zwiększać wydajność serwera bazodanowego.

Podsumowanie

Oprócz dokonania analizy porównawczej wybranych systemów zarządzania bazą danych, co było głównym celem niniejszej pracy, została zaproponowana przejrzysta metodologia postępowania przy ocenianiu DBMS (w konkretnym zastosowaniu). W dużym przybliżeniu postępowanie powinno koncentrować się na następujących zagadnieniach:

- określenie stałych elementów środowiska (to, co należy przyjąć, i na co musimy się zdecydować);
- określenie zmiennych (poddawanych testom) elementów środowiska;
- analiza rynku SZBD;
- analiza wyników znanych benchmarków (na przykład TPC);
- zebranie podstawowych własności i ograniczeń najpopularniejszych DBMS;
- określenie kryteriów porównawczych;
- zebranie lub stworzenie narzędzi testujących oraz dokonujących pomiarów środowiska;
- zdefiniowanie schematu bazy danych;
- zdefiniowania procedury ładującej dane do bazy danych;
- zdefiniowanie procedury migracji bazy do innych systemów zarządzania;
- testy;
- analiza otrzymanych rezultatów.

Przedmiotem osobnej analizy mogłoby być przeprowadzenie testów porównawczych dla różnych systemów operacyjnych. Jednak w takim przypadku ograniczeniem są możliwości produktu firmy Microsoft (który w analizowanych przypadkach znalazł się wśród zalecanych rozwiązań), pracującego jedynie w środowisku Windows.

Dalsze możliwości stoją również przed narzędziem DBTester, które można rozbudować o możliwość czytania zapisanych danych z narzędzi monitorujących (na przykład Profiler w MS SQL Server) oraz uruchamianie zapytań w przechwyconych przez narzędzie odstępach czasowych. Pozwoliłoby to na otrzymanie obrazu pracy jednego użytkownika i umożliwiło dokładniejsze testy związane z wydajnością przy skalowaniu ilości połączeń.

Literatura

1. [ALAPATI 2005] „Expert Oracle Database 10g Administration”, Sam R. Alapati Apress, 2005.
2. [CODD1 1970] „A Relational Model of Data for Large Shared Data Banks”, E. F. Codd, ACM, 1970.
3. [CODD2 1990] „The Relational Model for Database Management”, E. F. Codd, Addison Wesley Publishing Company, 1990, Wyd. II.
4. [CONN 2004] „Systemy baz danych”, T. Connolly, C. Begg, RM, 2004.
5. [DATE 2000] „Wprowadzenie do systemów baz danych”, Date C.J., WNT Warszawa, 2000, Wyd. II.
6. [GARTNER1 2006]„Gartner Study on DBMS Identifies Spending and Deployment Trends” Gartner 2006.
7. [GARTNER2 2006] „Database Management Systems Technology Trends”, Donald Feinberg, Gartner 2006.
8. [GRINKE 2002] „Bazy danych. Skrypt”, Leonard Grinke, ATH , 2002.
9. [GRUBER 2000] „SQL”, Gruber M., HELION Gliwice, 2000, Wyd. II.
10. [HECTOR 2006] „Systemy baz danych. Pełny wykład”, Garcia-Molina Hector, Jeffrey D. Ullman, Jennifer Widom, WNT, Warszawa, 2006, Wyd. I.
11. [HVB 2006] „History and Comparison of Relational Database Management Systems”, TechnoCircle HVB Information Services, 2006.
12. [ITZIK 2006] „Inside Microsoft SQL Server 2005: T-SQL Programming”, Itzik Ben-Gan, Dejan Sarka, Roger Wolter, Microsoft Press, 2006.
13. [KLINE 2002] „SQL Alamanach“, Kline K., Kline D., Helion Gliwice, 2002.
14. [KOFLER 2005] „The Definitive Guide to MySQL5”, Michael Kofler, Apress, 2005, Third Edition.
15. [KOZ 2007] „Bazy Danych: Nowe Technologie”, Kozielski S., Małysiak B., Kasprowski P., Mrozek D., Politechnika Śląska, Gliwice, 2007.
16. [LACH 2006] „Projekt elektronicznego dziennika lekcyjnego”, praca inżynierska, Mirosław Lach, 2006.
17. [LADANYI 2000] „SQL Księga eksperta”, Ladanyi H., Helion Gliwice, 2000.
18. [MATTHEW 2005] „Beginning Databases with PostgresSQL. From Novice to Professional”, Apress, 2005, Second Edition.

19. [MCCARTY 2002] „Debian GNU/Linux”, Bill McCarty, HELION Gliwice, 2002.
20. [MS2790A 2006] „Troubleshooting and Optimizing Database Servers Using Microsoft SQL Server 2005”, Microsoft Official Workshop, 2006.
21. [NOWAK 2005] „Rozproszone bazy danych - wykład”, A. Nowak, ATH, 2005.
22. [SCOTT 2005] „IBM DB2 Universal IBM DB2 Universal Database, Cloudscape, Database, Cloudscape, and Apache and Apache Derby", Dan Scoot, 2005.
23. [SUBIETA 1997] „Obiektowe Bazy Danych kontra Relacyjne Bazy Danych”, Subieta K., Instytut Podstaw Informatyki PAN, Warszawa, 1997.
24. [SUBIETA 1999] „Słownik terminów z zakresu obiektowości”, Subieta K., Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1999
25. [ULLMAN 1999] „Podstawowy wykład z systemów baz danych”, Ullman J. D., Widom J., WNT, Warszawa, 1999, Wyd. I.
26. [URMAN 2003] „Oracle9i – Programowanie w języku PL/SQL”, Scott Urman, Helion, 2003.
27. [VISSER 2003] „Sams Teach Yourself DB2 Universal Database in 21 Days”, Susan Visser, Bill Wong, Sams, 2003, Second Edition.
28. [WIRTH 1989] „Algorytmy + struktury danych = programy”, Wirth N., WNT, Warszawa, 1989, Wyd. II.
29. [WWWAM 2008] http://www.adventnet.com, 2008.
30. [WWWBD 2008] http://www.bazydanych.com.pl, 2008.
31. [WWWGARTNER 2008] http://www.gartner.com, 2008.
32. [WWWIBM 2008] http://www.ibm.com, 2008.
33. [WWWLINUX 2008] http://linux.die.net, 2008.
34. [WWWMS 2008] http://www.microsoft.com, 2008.
35. [WWWMSI 2008] http://www.msipolska.pl, 2008.
36. [WWWMYSQL 2008] http://www.mysql.com, 2008.
37. [WWWNDB 2008] http://hoslab.cs.helsinki.fi/savane/projects/ndbbenchmark, 2008.
38. [WWWORACLE 2008] http://www.oracle.com, 2008.
39. [WWWPOSTGRE 2008] http://www.postgresql.com, 2008.
40. [WWWSWAG 2008] http://www.swag.uwaterloo.ca, 2008.
41. [WWWTM1 2008] http://www.soliddb.com/tm1, 2008.
42. [WWWTPC 2008] http://www.tpc.org, 2008.
43. [WWWWIKI 2008] http://www.wikipedia.org, 2008.