Dariusz Olczyk[1]

DIALEKTY JĘZYKA SQL W IMPLEMENTACJACH ORACLE ORAZ MS SQL SERVER

Streszczenie

W artykule przedstawiono podstawowe informacje na temat relacyjnych baz danych definiując najbardziej istotne pojęcia oraz odwołując się do historii powstawania standardów w tej dziedzinie wiedzy informatycznej. Wychodząc od kryteriów Codda na relacyjną bazę danych omówiono różnice między bazą danych a Systemem Zarządzania Bazą Danych, a także sformułowano główne funkcje SZBD. Po krótkiej klasyfikacji języka SQL służącego do manipulacji bazami danych przedstawiono w części zasadniczej różnice i podobieństwa w tymże języku w różnych implementacjach skupiając się na produktach komercyjnych (MS SQL Server, Oracle) z niewielkimi odwołaniami do systemów typu OpenSource (MySQL, PostgreSQL). Porównania dokonano na trzech płaszczyznach: typy danych, polecenia oraz funkcje języka SQL.
W końcowej części artykułu zaprezentowano prace dyplomowe prowadzone w MILA College (obecnie Warszawska Wyższa Szkoła Informatyki) obejmujące zagadnienia transformacji kodu SQL-owego między poszczególnymi dialektami oraz tworzenia pakietów ułatwiających pracę programistów baz danych w środowisku wieloproduktowym.

Abstract

In the article, basic information about databases are presented, including defining the most crucial notions and referring to the history of emerging standards in that field of IT knowledge. Starting from Codd's database criteria, the differences between a database and DataBase Management System (DBMS) are elaborated on. The main functions of DBMS are also clarified. Upon the short classification of SQL, which is used to manipulate databases, the main differences and similarities between different software are presented, focusing on commercial products (MS SQL Server, Oracle) with some references to Open Source systems (MySQL, PostgreSQL). The comparison was made in three areas: data type, commands and SQL function.
In the final part of the article, dissertations from the MILA College (currently Warsaw School of Computer Science) are presented. They all refer to the transformation of SQL code between particular dialects as well as creating packages to make the job of database makers easier in a multiproduct environment.

[1] Mgr inż. Dariusz Olczyk jest wykładowcą w Warszawskiej Wyższej Szkole Informatyki.

1. WPROWADZENIE

W codziennej pracy informatyka-programisty pojawiają się problemy z różnych dziedzin, od znajomości, których zależy jakość tworzonego oprogramowania, co nota bene moim zdaniem stanowi o atrakcyjności tej profesji. Podobnie rzecz ma się z bazami danych. Osoby zajmujące się projektowaniem oraz oprogramowywaniem baz często funkcjonują w środowisku wieloprodukcyjnym, tzn. muszą mieć wiedzę i umiejętności wykorzystania produktów dostarczanych przez różnych producentów. Najczęściej spotykanymi na rynku i znanymi szerokiej rzeszy użytkowników z pewnością są Oracle, MS SQL Server, MySQL, PostreSQL czy zdobywający ostatnio dużo pochlebnych opinii FireBird (dawniej Interbase). Tak, więc na obecnie nie wystarczy już specjalizować się w rozwiązaniach wyłącznie jednego producenta. Od wykształconego inżyniera-informatyka wymaga się ogólnej znajomości zasad projektowania oraz konkretnych umiejętności w wielu systemach baz danych.

Niniejszy wykład poświęcony jest dwóm czołowym produktom funkcjonującym na rynku baz komercyjnych, z niewielkimi dodatkowymi informacjami o bazach OpenSource-owych. Pokrótce omówione zostaną: modele i architektura relacyjnych baz danych, standard języka SQL, w części zasadniczej zaprezentowane zostaną różnice i podobieństwa języka SQL w bazach Oracle oraz Microsoft SQL Server, a na koniec pokazane zostaną prace z tej dziedziny wykonywane przez studentów MILA College.

2. FIRST STEP …

Każde rozważania dotyczące szczegółowych i specjalistycznych rozwiązań muszą rozpocząć się od kilku informacji wprowadzających w dziedzinę problemową i umożliwiających uchwycenie szerszego kontekstu omawianego tematu. Tak, więc nasz wykład rozpoczniemy od definicji bazy danych. Istnieje pewne nieporozumienie dotyczące tego terminu, a wynikające z przyjętego uogólnienia. W języku potocznym bardzo często mówiąc o bazie danych mamy na myśli cały produkt oferowany przez producenta, podczas gdy w rzeczywistości jest ona tylko jego podzbiorem. W odróżnieniu od Systemu Zarządzania Bazą Danych (ang. Database Management System – DBMS), który jest zestawem struktur w pamięci operacyjnej oraz procesów ją obsługujących, **baza danych jest zbiorem danych zgromadzonych na nośniku istniejącym przez długi czas**. Już w latach sześćdziesiątych ubiegłego stulecia w USA pojawiły się pierwsze zwiastuny systemów bazodanowych w postaci rozwiązań związanych z sektorem bankowym oraz rezerwacją lotniczą. Zastosowane składowanie danych w plikach dało wymierną korzyść w postaci umożliwienia przechowywania

dużej ilości danych oraz prostoty implementacji, ale przyniosło również zagrożenia związane z brakiem gwarancji trwałości danych, brakiem sformalizowanego języka zapytań oraz brakiem bezkolizyjności w dostępie do tych danych. Szybko okazało się, że konieczne jest opracowanie bardziej złożonego modelu gromadzenia i przetwarzania informacji. W kolejnych latach pojawiły się, dziś już archaiczne, modele hierarchiczne oraz sieciowe, które rzeczywiście rozwiązały część problemów. Nadal jednak brak było języka wysokiego poziomu dającego możliwość tworzenia struktur bazy oraz manipulacji danymi w niej zawartymi. Jeszcze w latach 60. opracowana została przez E. F. Codda relacyjna teoria danych, a sam autor opracował listę kryteriów, które muszą być spełnione, aby system uznano za relacyjny (tab. 1). System ten opiera się na założeniu, że użytkownicy widzą dane w postaci zbioru tabel powiązanych ze sobą poprzez wspólne wartości danych. Dane są przechowywane w relacjach (tabelach), zaś te zbudowane są z atrybutów (kolumn) i krotek (rekordów, wierszy). I chociaż wydawało się, iż trudno będzie zaimplementować system relacyjny, już w latach 70. IBM zaprezentował produkt oparty na tym modelu oraz język, którego dotąd brakowało. Język ten otrzymał nazwę SEQUEL (Structured English Query Language). W kolejnych latach w raz z pojawianiem się kolejnych wersji języka zmieniono również nazwę na SQL, przy czym za ciekawostkę można uznać fakt, iż wiele osób nadal wymawia nowy skrót w starym brzmieniu (głównie w Stanach Zjednoczonych), czyli „siquel".

Tab. 1. Kryteria Codda dla prawdziwie relacyjnych systemów baz danych

1.	Informacje są reprezentowane logicznie w tabelach
2.	Dane są logicznie dostępne poprzez podanie nazwy tabeli, wartości klucza podstawowego i nazwy kolumny
3.	Wartości null są traktowane w jednolity sposób jako „brakujące informacje". Nie mogą być traktowane jako puste łańcuch znaków, puste miejsca, czy zera
4.	Metadane są umieszczone w bazie danych dokładnie tak, jak zwykłe dane
5.	Język obsługi danych ma możliwość definiowania danych i perspektyw, więzów integralności, przeprowadzenia autoryzacji, obsługi transakcji i manipulacji danymi
6.	Perspektywy reagują na zmiany swoich tabel bazowych
7.	Istnieją pojedyncze operacje pozwalające na wyszukanie, wstawienie, uaktualnienie i usunięcie danych
8.	Operacje użytkownika są logicznie oddzielone od fizycznych danych i metod dostępu

9.	Operacje użytkownika pozwalają na zmianę struktury bazy danych bez konieczności tworzenia od nowa bazy czy aplikacji ją obsługującej
10.	Więzy integralności są umieszczone i dostępne w metadanych, a nie w programie obsługi bazy danych
11.	Język manipulacji danymi powinien działać bez względu na to, jak i gdzie są rozmieszczone fizyczne dane
12.	Operacje na pojedynczych rekordach przeprowadzane w systemie podlegają tym samym zasadom i więzom, co operacje na zbiorach danych

Patrząc na kryteria relacyjności bazy danych przedstawione przez Codda trudno oprzeć się wrażeniu, iż nawet niektóre obecnie funkcjonujące na rynku produkty nie są według tych kryteriów w pełni relacyjne. Czego zatem oczekujemy od współczesnych baz danych, jakie cechy powinny one posiadać i jakie w związku z tym stawiamy wobec nich wymagania?

3. BAZA DANYCH A SYSTEM ZARZĄDZANIA BAZĄ DANYCH

Jeśli chodzi o cechy charakterystyczne koniecznie wymienić należy trzy podstawowe: trwałość danych, duży wolumen danych oraz bogata semantyka danych. Każdą z tych cech możemy opisać poprzez podanie kilku własności składowych:

Trwałość danych

- Długi czas życia – kilka, kilkadziesiąt, kilkaset lat;
- Niezależność od działania aplikacji;
- Niezależność od platformy sprzętowo-programowej;
- Niezależność od eksploatowania aplikacji.

Duży wolumen danych

- Tysiące, miliony, miliardy danych;
- Dane nie mieszczą się w pamięci operacyjnej – wymagana pamięć zewnętrzna;
- Danych jest zbyt dużo dla ich liniowego przeglądania przez użytkowników.

Bogata semantyka danych

- Dane różnią się strukturą i semantyką;
- Na dane są nałożone różne ograniczenia;
- Istnieją złożone zależności między danymi.

Zakładając, iż od współczesnych baz danych oczekujemy, że będą one posiadały wyżej wymienione cechy, wymagamy jednocześnie, aby dla dużej ilości danych systemy zarządzania nimi gwarantowały ich poprawność, tzn. odporność na błędy, awarie i inne anormalne sytuacje wynikające z zawodności środowisk sprzętowo-progra-

mowych, współbieżnego dostępu czy rozproszenia bazy. Wymagamy również, aby dane składowane były efektywnie, a SZBD dostarczały efektywnych metod dostępu do nich oraz optymalizacji zapytań. Na koniec, choć to rzecz oczywista, chcemy, aby pomimo różnorodności świata rzeczywistego modele poprawnie odwzorowywały ten świat. Mając na uwadze fakt, iż baza danych to po prostu „suche" dane widzimy jak ważne znaczenie dla jakości systemu bazy danych ma system zarządzania. To właśnie ten element decyduje o tym, że o jednym produkcie mówimy, że jest dobry, a o innym, że nie, że preferujemy stosowanie w praktyce produktów konkretnych producentów.

System zarządzenia bazą danych musi spełniać następujące funkcje:

- Możliwość tworzenia i utrzymania baz danych za pomocą specjalizowanego języka definiowania struktur danych (ang. DDL – Data Definition Language);
- Możliwość dostępu do bazy danych (zapytań na bazie danych i modyfikacji bazy danych) za pomocą języka dostępu da bazy danych (ang. DML – Data Manipulation Language);
- Efektywne składowanie i przetwarzanie dużego wolumenu danych;
- Strojenie efektywności przetwarzania i składowania danych poza środowiskiem aplikacji;
- Zapewnienie bezpieczeństwa danych zagrożonego awaryjnością środowiska sprzętowo-programowego i przez niepowołanych użytkowników;
- Synchronizacja współbieżnością dostępu do danych.

W jaki sposób odbywa się przetwarzanie danych w systemie bazodanowym zbudowanym w oparciu o model relacyjny? Mając do dyspozycji specjalizowaną aplikację do edycji poleceń lub jakikolwiek inny program umożliwiający kontakt z bazą danych użytkownik konstruuje na przykład polecenie wyszukania określonych danych. Weźmy pod uwagę następujące zadanie:

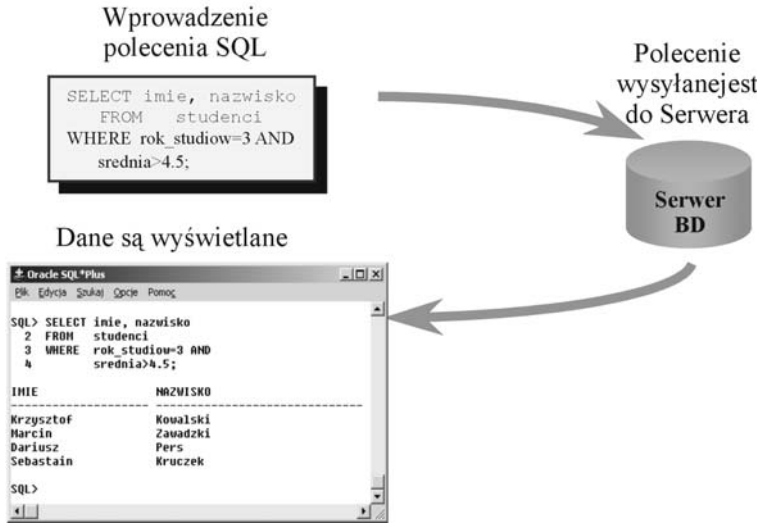
Wybierz z bazy danych informacje o studentach z trzeciego roku (imię i nazwisko), którzy uzyskali średnią powyżej 4,5.

Posługując się językiem SQL polecenie realizujące to zadanie będzie wyglądało następująco:

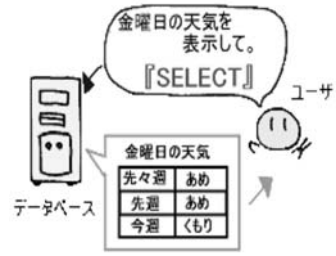
```
SELECT   imie, nazwisko
FROM     studenci
WHERE    rok_studiów=3 AND srednia>4.5
```

Po klauzuli SELECT podajemy listę poszukiwanych atrybutów, po klauzuli FROM tabelę, w której te dane się znajdują, a następnie po klauzuli WHERE precyzujemy warunki wyszukiwania. Zwraca uwagę fakt, iż programista piszący to polecenie nie musi się zastanawiać, w jaki sposób i gdzie składowane są fizycznie dane.

Takie polecenie jest wysyłane do serwera bazy danych i tam następuje proces obsługi żądania klienta. System zarządzania bazą danych dokonuje analizy polecenia pod względem formalnym, następnie na podstawie zaszytych formuł optymalizatora zapytań przygotowuje plan realizacji zapytania (decyduje np. jakie zastosować indeksy, kiedy łączyć tabele itp.). Końcowa faza to realizacja – pobranie i transfer danych do aplikacji klienckiej (rys. 1)



Rys. 1. Komunikacja z SZBD



Zaprezentowane powyżej polecenie zostało napisane w najpopularniejszym języku bazodanowym dostępnym we wszystkich funkcjonujących na rynku produktach – wspomnianym wcześniej języku SQL. Zaczerpnięta z serwisu internetowego WIKIPEDIA definicja tego języka mówi, iż **SQL jest strukturalnym i deklaratywnym językiem zapytań wysokiego poziomu przeznaczonym do manipulacji relacyjnymi bazami danych**. Język ten rozwijał się począwszy od początku lat 70 i pracy dr. E. F. Codda przybierając różne postacie w zależności od firm wprowadzających ten język do swoich systemów. Każdy z producentów w walce o rynek starał się szybko wprowadzić takie elementy przetwarzania danych, które dałyby mu palmę pierwszeństwa. W ten sposób programiści znajdowali się w bardzo trudnej sytuacji. Brak jasno określonego standardu zapewne nie wpływał korzystnie na efekty końcowe, a ich praca mogła przypominać czytanie chińskiego.

4. SQL WCZORAJ I DZIŚ

Wraz z pojawieniem się w 1986 roku SQL-a zaakceptowanego przez Amerykański Instytut Normalizacyjny (ang. ANSI – American National Standards Institute) rozpoczyna się proces jego standaryzacji. Kolejne aktualizacje standardu to SQL'92 (SQL2) oraz zatwierdzony również przez Międzynarodową Organizację Normalizacji ISO – SQL'99 (SQL3). Wprowadzenie standardu języka rozwiązało również problem wymogów, jakie musi spełniać produkt, aby uznano iż implementuje on określony standard. Wprowadzono poziomy zgodności dla SQL'92: wstępny, przejściowy (dołożony przez NIST – National Institute of Standard and Technology), pośredni i pełny oraz dla SQL'99: podstawowy i rozszerzony. Dostawcy muszą osiągnąć poziom wstępny dla SLQ'92 lub podstawowy dla SQL'99, aby produkt spełniał wymogi standardu. Zdefiniowano również zestawy dodatkowych funkcji dla poziomu rozszerzonego (tab. 2). Na uwagę zasługuje fakt, iż wystarczy, aby dostawca w swoim produkcie zaimplementował jedną z dziewięciu grup funkcji, aby uznał on zgodność ze standardem „rozszerzony SQL'99".

Tab. 2. Dodatkowe zestawy funkcji w SQL'99

ID	Nazwa	Cechy
PKG001	Rozszerzone funkcje daty i czasu	Typ danych INTERVAL; specyfikacje stref czasowych; pełna obsługa datetime
PKG002	Rozszerzone zarządzanie-integralnością	Referencyjne operacje usuwania i zmiany wartości; zarządzanie więzami; wyzwalacze; podzapytania w klauzuli CHECK
PKG003	Możliwości OLAP	CUBE i ROLLUP; operator INTERSECT, FULL OUTER JOIN
PKG004	Trwale składowane moduły SQL	Polecenia języka proceduralnego: CASE, IF, WHILE, REPEAT, LOOP i FOR; składowane moduły; perspektywa INFORMATION_SCHEMA
PKG005	SQL Call-Level Interface	Interfejs API powalający na wykonywanie operacji SQL podobnych do standardu ODBC
PKG006	Podstawowa obsługa obiektów	Dziedziczenie w typach definiowanych przez użytkownika; referencje pól i atrybutów

PKG007	Rozszerzona obsługa obiektów	ALTER TABLE, ADD; podtabele; funkcje SQL; ONLY w zapytaniach; predykat type
PKG008	Cechy aktywnych baz danych	Wyzwalacze
PKG009	Obsługa multimediów	Obsługa multimedialnych danych strumieniowych oraz dużych i złożonych danych audio i wideo

Standard SQL wprowadził również klasyfikacje poleceń ze względu na ich przeznaczenie. I tak SQL'92 podzielił wszystkie polecenia języka na trzy grupy: DML – polecenia modyfikujące zawartość bazy danych, DDL – umożliwiające tworzenie struktur bazy oraz DCL – kontroli uprawnień użytkowników. Standard SQL'99 wprowadza siedem grup nie nadając im specyficznych nazw:

- Instrukcje obsługi połączeń SQL – otwarcie i zamknięcie połączenia z klientem (CONNECT, DISCONNECT);
- Instrukcje kontrolne SQL – kontrola wykonywania zbiorów instrukcji (CALL, RETURN);
- Instrukcje obsługi danych SQL – manipulacja danymi (SELECT, INSERT, UPDATE, DELETE);
- Instrukcje diagnostyczne SQL – dostarczenie informacji diagnostycznych (GET DIAGNOSTICS);
- Instrukcje obsługi schematów SQL – tworzenie obiektów bazy danych (CREATE, ALTER, DROP);
- Instrukcje obsługi sesji SQL – kontrola domyślnych zachowań i parametrów sesji (SET);
- Instrukcje obsługi transakcji SQL – ustalenie zakresu transakcji (COMMIT, ROLLBACK).

5. MICROSOFT KONTRA ORACLE

Biorąc za punkt odniesienia standardy języka SQL'92 oraz SQL'99 niezwykle ciekawym zadaniem staje się porównanie możliwości implementacji tego języka w produktach renomowanych firm, jakimi są Microsoft i Oracle. Rozważania te podzielone zostały na trzy zasadnicze działy: typy danych, polecenia i funkcje. Całość zagadnienia można, zatem w przybliżeniu sformułować jako określenie możliwości przenoszenia kodu SQL-owego pomiędzy systemami zbudowanymi na bazach Oracle oraz MS SQL Server.

Typy danych

Poniższe zestawienie prezentuje podstawowe typy danych obsługiwane w obu rozpatrywanych systemach oraz typy uznane w standardzie.

Tab. 3. Typy danych

SQL92 i SQL99	MS SQL Server	Oracle
-	bigint	-
binary	binary	-
bit	bit	-
charakter	char(n)	char(n)
-	cursor	-
-	-	date
-	datetime	-
decimal	decimal(p,s)	decimal
float	float	float(n)
-	image	-
integer	int	integer(n)
-	money	-
national charakter	nchar(n)	national charakter
-	ntext	-
-	numeric(p,s)	number(p,s)
national charakter varying	nvarchar(n)	national character varying
-	-	nvarchar2(n)
-	-	raw(n)
-	real	real
-	rowversion	-
-	smalldatetime	-
smallint	smallint	smallint
-	smallmoney	-
-	sql_variant	-
-	table	-
-	text	-
-	tinyint	-
-	uniqueidentifier	-
-	-	urowid([(n)]
binary varying	varbinary	-
charakter varying	varchar(n)	varchar
-	-	varchar(2)

Oczywistym staje więc, że możliwości przenoszenia w postaci skryptów poleceń posługujących się typami są dalece ograniczone. Zakładając, iż zasadnicza treść skryptu tworzącego schemat bazy danych opiera się na poleceniach CREATE TABLE, gdzie proces określania typów poszczególnych atrybutów jest niezwykle istotny dla późniejszego działania systemu produkcyjnego, translacja z SQL-a zaimplementowanego w Oracle-u na SQL Microsoft-owy lub odwrotnie będzie musiała podlegać wielu ograniczeniom. Rozwiązaniem jest posługiwanie się najbardziej popularnymi typami znakowymi i numerycznymi jak: char, varchar, float, int lub zgoda na zmianę typu przy założeniu zawężenia lub rozszerzenia zakresu przetwarzanych danych.

Polecenia języka SQL

Kolejna tabela zawiera listę podstawowych poleceń języka z oznaczeniem stopnia, w jakim dane polecenie jest implementowane w wybranym produkcie.

Tab. 4. Typy danych

Polecenie	MS	Oracle	Polecenie	MS	Oracle
ALTER PROCEDURE	OZ	OZ	DROP INDEX	OZ	OZ
ALTER TABLE	OZ	OZ	DROP PROCEDURE	O	O
ALTER TRIGGER	OZ	OZ	DROP ROLE	NO	OZ
ALTER VIEW	OZ	OZ	DROP TABLE	OZ	OZ
CALL	NO	O	DROP TRIGGER	OZ	OZ
CASE	O	NO	DROP VIEW	O	O
CAST	O	NO	FETCH	O	O
CLOSE CURSOR	O	O	GRANT	OZ	OZ
COMMIT	OZ	O	INSERT	OZ	O
TRANSACTION	OZ	O	klauzula JOIN	O	NO
operatory konkatenacji	OO	O	operator LIKE	OZ	OZ
CONNECT	OZ	O	OPEN	O	O
CREATE DATABASE	OZ	OZ	OPERATORS	OZ	OZ
CREATE FUNCTION	OZ	OZ	RETURN	O	O
CREATE INDEX	O	O	REVOKE	OZ	OZ
CREATE PROCEDURE	NO	OZ	ROLLBACK	OZ	O
CREATE ROLE	O	O	SAVEPOINT	OZ	O
CREATE SCHEMA	OZ	OZ	SELECT	OZ	OZ
CREATE TABLE	OZ	OZ	SET CONNECTION	OO	NO
CREATE TRIGGER	OZ	OZ	SET ROLE	NO	OZ
CREATE VIEW	O	O	SET TIME ZONE	NO	OZ
DECLARE CURSOR	OZ	O	SET TRANSACTION	OZ	OO
DELETE	OO	OZ	START TRANSACTION	NO	NO
DISCONNECT	OZ	NO	TRUNCATE TABLE	O	OZ
DROP DATABASE	OZ	OZ	UPDATE	OZ	OZ
DROP FUNCTION					

Legenda
O obsługiwane
OZ wspiera standard, ale używa własnego kodu lub składni
OO obsługuje niektóre funkcje dla danego polecenia
NO nie obsługiwane

I znowu okazuje się, że pomimo założenia, iż oba produkty implementują standard języka SQL, to w rzeczywistości tylko niektóre polecenia mogłyby być przeniesione wprost, większość posługuje się specyficzną dla siebie składnią, a niektóre polecenia w ogóle nie mają swojego odpowiednika w systemie konkurencyjnym. Dla przykładu zaprezentowane zostaną poniżej trzy polecenia SQL.

CAST (MS – O; Oracle – NO)
Polecenie CAST konwertuje jawnie wyrażenie jednego typu na inny typ danych. Obsługiwane również w PostgreSQL; brak w MySQL

Składnia SQL99:
CAST (wyrażenie AS typ_danych [(długość)])

Przykład zastosowania:

```
SELECT
CAST(roczna_sprzeda¿ AS CHAR(5))+„egzemplarzy”+CAST(tytul AS
      VARCHAR(30))
FROM     tytuly
WHERE    roczna_sprzedaz IS NOT NULL AND roczna sprzedaz > 10000
ORDER BY roczna_sprzedaz DESC
```

Wynik:

```
...
22246 egzemplarzy The Gourment Microwave
18772 egzemplarzy You Can Combat Computer Stress
15096 egzemplarzy Fifty Years in Buckingham Pala
...
```

CREATE ROLE (MS – NO; Oracle – OZ)
Pozwala na stworzenie nazwanego zestawu uprawnień, który można przypisać użytkownikowi bazy danych.
Nie obsługiwane ani w PostgreSQL ani w MySQL.

Składnia SQL99:
CREATE ROLE nazwa_roli [WITH ADMIN {CURRENT_USER | CURRENT_ROLE}]

W Oracle
CREATE ROLE nazwa_roli [NOT IDENTIFIED | IDENTIFIED {BY hasło | EXTERNALLY | GLOBALLY}]

Przykład zastosowania:

```
CREATE ROLE manager;

GRANT ALL ON pracownicy TO manager;
GRANT CREATE SESSION, CREATE DATABASE LINK TO manager;

ALTER ROLE manager IDENTIFIED BY tiger;

GRANT manager TO Tomasz, Zofia;
```

operator LIKE (MS – OZ; Oracle – OZ)

Operator LIKE pozwala na podanie wzorca dopasowania łańcucha znaków w instrukcjach SELECT, INSERT, UPDATE i DELETE. Obsługiwane ze zmianami w PostgreSQL i w MySQL.

Składnia SQL99:
WHERE wyrażenie [NOT] LIKE wzorzec

operatory wieloznaczne

		MS	Oracle
%	zastępuje dowolny ciąg znaków SELECT * FROM autorzy WHERE miasto LIKE '%dolne%';	✓	✓
[ ]	dopasowuje wszystkie wartości z podanego zbioru lub zakresu SELECT * FROM autorzy WHERE au_nazwisko LIKE '[ABC]acki'	✓	-
[^]	dopasowuje dowolny znak nie należący do podanego zbioru lub zakresu SELECT * FROM autorzy WHERE au_nazwisko LIKE '[A-Z^L]ars[eo]n'	✓	-
_	zastępuje dokładnie jeden znak	✓	✓

Funkcje

Ostatnie zestawienie prezentuje funkcje języka SQL. Oczywiście jest to niewielki wycinek z ogólnej liczby funkcji zaimplementowanych w obu produktach.

Tab. 5. Funkcje SQL

MS SQL Server	Oracle
abs	abs
acos	acos
app_name	add_months
ascii	ascii
asin	asin
atan	atan
atan2	atan2
avg	avg
cast	ceil
ceiling	chr
char	concat
coalesce	convert
col_length	corr
col_name	cos
contains	cosh
convert	count
cos	decode
cot	exp
count	floor
current_timestamp	greatest
current_user	grouping
datalength	initcap
dateadd	instr
datediff	last_day
datename	length
datepart	ln
day	log
.....	....

Na pierwszy rzut oka widać, że część funkcji się pokrywa. Powtarzają się naturalnie funkcje trygonometryczne (sin, cos, tg, ctg), matematyczne (abs, exp, ln, log), grupowe (avg, min, max, sum, count) oraz działające na ciągach znaków (convert). Wiele jest jednak rozwiązań charakterystycznych dla danego producenta, których trudno byłoby się dopatrzyć po drugiej stronie, szczególnie jeśli chodzi o obsługę charakterystycznych typów danych (np. MS – datetime, Oracle – date).

Dla przykładu pokazano poniżej różne sposoby łączenia ciągów znakowych.

CONCATENATE
Operator konkatenacji łączy dwa łańcuchy znaków w jeden.

Składnia SQL99:
CONCATENATE('łańcuch_1'** || **'łańcuch_2')

Przykład zastosowania (Oracle):

```
SELECT nazwisko||' '||imie
FROM studenci;
```

Przykład zastosowania (Microsoft):

```
SELECT nazwisko+' '+imie
FROM studenci;
```

6. PROJEKTY W MILA COLLEGE

Omawiane powyższej zagadnienia stanowią nielada wyzwanie dla programistów próbujących funkcjonować w środowisku wielkoproduktowym lub stojących przez zadaniem przeniesienia istniejącego systemu bazodanowego na inną platformę. Stąd pojawił się pomysł stworzenia mechanizmów ułatwiających takie działania. W ramach realizowanych w MILA College przez studentów trzeciego roku prac inżynierskich sformułowano prace bezpośrednio związane z tą dziedziną wiedzy.

Pierwszy z tematów zrealizowany przez pana Radosława Tomaszewskiego dotyczył wykonania aplikacji dokonującej automatycznej translacji kodu SQL-owego MS SQL Server na dialekt Oracle-SQL. Ze względu na złożoność problemu program miał w swoim założeniu obsługiwać wyłącznie polecenia DML, co i tak zmusiło studenta do dogłębnego przeanalizowania składni języka, wykonania projektu aplikacji oraz skonstruowania odpowiednich algorytmów translacji. Testy sprawdzające poprawność rozwiązań dały pozytywny wynik, a cała praca oceniona została na ocenę dobrą.

Drugi temat jest obecnie w fazie przygotowawczej, w której student zbiera niezbędne informacje z zakresu pracy, kompletuje literaturę oraz analizuje możliwości realizacji. Pan Piotr Salwowski otrzymał zadanie wykonania dwóch pakietów funkcji dla MS SQL Server 2000 oraz Oracle 9i, które uzupełniać będą funkcje z produktu konkurencyjnego. Stworzenie takich zestawów umożliwi w większym zakresie uniezależnienie się od specyfiki producenta i tworzenie kodu SQL-owego w jednym dialekcie (z możliwością łatwej zamiany na drugi dialekt).

7. ZAKOŃCZENIE

Podsumowując niniejszy wykład należy stwierdzić, iż pomimo funkcjonowania (niewątpliwie użytecznych) standardów języka SQL pogoń czołowych producentów systemów baz danych za nowinkami oraz chęć wzbogacenia ich produktów o elementy wyróżniające z „tłumu" powoduje, iż praca projektanta oraz programisty bazodanowego nie jest łatwa. Z drugiej strony biorąc za przykład naszych studentów sytuacja taka daje szerokie możliwości rozwoju oraz prowadzenia różnego rodzaju działalności naukowej. Z mojej strony, chciałbym zapewnić, iż tworzenie poleceń SQL-owych czy budowanie złożonych struktur bazy danych może dać wiele satysfakcji i być równie interesującym zajęciem jak, tak bardzo popularna ostatnimi czasy animacja komputerowa czy grafika 3D. Możliwości są właściwie ograniczone wyłącznie wyobraźnią studenta.

Literatura

[1] „SQL almanach. Opis poleceń języka"; Kelvin Kline, Daniel Kline; Wydawnictwo Helion; 2002

[2] „SQL. Omówienie standardu języka"; C.J. Date, H. Darwen; Wydawnictwo WNT; Warszawa 2000

[3] „Transact-SQL. Czarna księga"; Marcin Szeliga; Wydawnictwo Helion; 2003

[4] „Oracle 8 PL/SQL Programming"; Scott Urman; Wydawnictwo Oracle Press Osborne; 1997

[5] „Oracle. Polecenia"; G. Harrison; Wydawnictwo „Robomatic"; Warszawa 2002

[6] "Microsoft SQL Server 2000. Księga eksperta"; Ray Rankins, Paul Jensen, Paul Bertucci; Wydawnictwo Helion; 2003

[7] „SQL. Księga eksperta"; Hans Ladanyi; Wydawnictwo Helion; 2000.