Bazy danych i systemy zarządzania

Wykład V

Elementy języka SQL

Część I

Wykaz literatury

1. Celko, J.: *SQL Zaawansowane techniki programowania.* Mikom, Warszawa, 1999. ISBN 83-7158-221-8.

2. Connan, S.J., G.A.M. Otten: *SQL – The Standard Handbook.* (based on the new SQL standard (ISO 9075:1992(E)). McGraw-Hill Book Company, London, 1993.

3. Gruber, M.: SQL. Znakomity podręcznik opisujący najnowszy standard SQL-a. Wydawnictwo Helion, Gliwice, 1996. ISBN 83-86718-32-3.

4. Harrington, J.L.: *SQL dla każdego.* EDU-MIKOM, Warszawa, 1998. ISBN 83-87102-55-5.

5. Kloc, A.: *Informix.* Wydawnictwo PLJ, Warszawa, 1991.

6. Lustig, D.: Język SQL dla relacyjnych bazy danych ORACLE. KSW Technimex, Wrocław, 1992.

7. *SQL Język relacyjnych baz danych.* Wellesley Software. WNT, W-wa, 1992/95. ISBN 83-204-1806-2.

8. Stephens, R.K. et al.: *SQL w 3 tygodnie.* LT&P, Warszawa, 1999. ISBN 83-7158-221-8.

Strony internetowe:

`http://galaxy.uci.agh.edu.pl/ chwastek/lectures/db/dbtitle.html`
`http://www.ia.pw.edu.pl/%7Ettraczyk/`
`http://baszta.iie.ae.wroc.pl/index.html`
`http://www.cs.put.poznan.pl/kjankiewicz/oracle/sql/index.htm`
`http://www.cs.put.poznan.pl/rwrembel/courses/sbd.htm`

Czym jest SQL

Definicja

SQL := Structured Query Language; database sub-language (niepełny język obsługi baz danych (bez kontroli sterowania)).

SQL jest językiem obsługi baz danych (RBD) zaimplementowanym w systemach zarządzania bazami danych (SZDB), przeznaczonym do definiowania struktur danych, wyszukiwania danych oraz operacji na danych. Posiada on akceptację ANSI oraz standard ISO. W praktyce jest *standardowym językiem zapytań* dla relacyjnych baz danych.

Cechy języka SQL

- jest językiem wysokiego poziomu (4GL), opartym na słownictwie języka angielskiego; jego wyrażenia mają określoną strukturę,
- jest językiem deklaratywnym (nieproceduralnym); zorientowanym na wynik (użytkownik definiuje co chce otrzymać, ale nie pisze jak),
- nie posiada instrukcji sterujących wykonaniem programu,
- nie dopuszcza rekurencji,
- zawiera logikę trójwartościową,
- umożliwia definiowanie struktur danych, wyszukiwanie danych, oraz opracje na danych.

Historia SQL-a

Etapy powstawania SQL-a

- 1970: E.F. Codd, IBM – Relacyjne Bazy Danych,
- 1974: Chamberlain, IBM, San Jose – Structured English Query Language SEQUEL (prototyp SQL),
- 1976-7: SEQUEL/2,
- koniec lat 70-tych: ORACLE (Relational Software Inc.) – pierwsza implementacja praktyczna (komercyjna),
- 1981: IBM – SQL/DS (SZBD), poprzednik DB/2 (1983),
- 1982: ANSI: RDL (Relationla Data Language),
- 1983: ISO – definicja SQL,
- 1986: ANSI – pierwszy standard SQL (SQL-86),
- 1987: ISO – pierwszy standard SQL: ISO 9075: 1987 (E),
- 1989: ISO – następny standard SQL: ISO 9076: 1989 (E) (SQL-89),
- 1992: ISO – kolejna, wzbogacona wersja: ISO 9075: 1992 (E) (**SQL 2**),
- 1999: SQL 3 (?)
- OQL (?)

Strukturai wykorzystanie języka SQL

Komponenty języka SQL

- DDL (Data Definition Language) – język definiowania struktur danych (`CREATE`),
- DQL (Data Query Language) – język definiowania zapytań dla wyszukiwania danych, (`SELECT`),
- DML (Data Manipulation Language – język operacji na danych (`SELECT, INSERT, UPDATE, DELETE`),
- Instrukcje sterowania danymi – kontrola uprawnień użytkowników (`GRANT, REVOKE`).

Wykorzystanie SQL-a

- Interaktywny SQL – bezpośredni dostęp do danych za pomocą interepretera SQL (np. z terminala ASCII),
- Statyczny SQL – stały (predefiniowany) kod w SQL; może to być zanurzony SQL (tzw. embedded SQL) czyli kod znajdujący się wewnątrz innego języka programowania lub modułowy SQL, tzn. samodzielne moduły w języku SQL, które mogą być łączone z modułami innych języków,
- Dynamiczny SQL – kod SQL generowany dynamicznie przez programy użytkowe; często generowany jest za pomocą interfejsów graficznych lub z poziomu WWW,
- Definicyjny SQL – kod w SQL-u generowany przy pomocy narzędzi CASE.

Elementy języka SQL – alfabet i język

Alfabet SQL

Alfabet SQL obejmuje:

- zestaw znaków SQL_TEXT (chrakterystyczny dla implementacji); `A,B,C,...,Z,a,b,c,...,z,0,1,2,...,9` oraz znaki zpecjalne: `. ; ( ) , : % _ ? ' "+ - * / < > = & |` i spacja,
- literały (stałe),
- identyfikatory (nazwy), np. nazwy tabel, kolumn (atrybutów) widoków, schematów, etc.,
- elementy semantyczne języka: nazwy poleceń i funkcji.

Zasady konstrukcji wyrażeń

- nazwy umożliwiają dostęp do obiektów z różnych poziomów; realizuje się to za pomocą tzw. wyrażeń ścieżkowych, np. `CATALOG.Company.Department`, separatorem poziomów jest kropka,
- możliwe jest konstruowanie i operacje porównania na wierszach, np. `(A1, B1, C1) < (A2, B2, C2)`,
- każda instrukcja zaczyna się słowem kluczowym, może zawierać modyfikatory i kończy się średnikiem,
- `*` oznacza wszystkie kolumny (atrybuty) tabeli,
- stałe tekstowe zapisywane są w cudzysłowach, np. `'Warszawa'`.

Struktura i przykłady typowych zapytań

Struktura typowego zapytania

```
SELECT Attribute1, Attribute2,...,Attributen
   FROM Table1, Table2,...,Tablek
   WHERE Condition;
```

Typowe zapytanie pozwala odczytać wartości zadanych atrybutów z wybranej tabeli (lub tabel) – wykonywana jest więc projekcja na wyspecyfikowane atrybuty; warunek zadany po słowie `WHERE` ma charakter formuły logicznej i stanowi kryterium wyboru rekordów – dokonywana jest więc równocześnie selekcja. W przypadku podania więcej niż jednej tabeli wykonywana jest na tych tabelach operacja iloczynu kartezjańskiego. Klauzula `WHERE` nie jest obowiązkowa.

Przykłady prostych typowych zapytań

Wyświetlania zawartości tabeli (wszystkie kolumny):

```
SELECT *
   FROM Dostawcy;
```

Wyświetlania zawartości tabeli (wybrane kolumny):

```
SELECT NazwaDostawcy, TelefonDostawcy
   FROM Dostawcy;
```

Wyświetlanie zawartości tabeli (wybrane kolumny) z usunięciem duplikatów:

```
SELECT DISTINCT NazwaDostawcy
   FROM Dostawcy;
```

Przykłady typowych zapytań

Sortowanie tabeli wynikowej

```
SELECT *
   FROM Dostawcy
   ORDER BY NazwaDostawcy;

SELECT NazwaDostawcy, NazwaTowaru
   FROM Dostawcy
   ORDER BY NazwaTowaru, NazwaDostawcy;
```

Sortowanie tabeli wynikowej w odwrotnej kolejności

```
SELECT Wiek, Nazwisko
   FROM Pracownicy
   ORDER BY Wiek DESC;
```

Sortowanie tabeli wynikowej wg wybranych kryteriów

```
SELECT Wiek, Nazwisko
   FROM Pracownicy
   ORDER BY Wiek DESC, Pracownik ASC;
```

Sortowanie z użyciem numerów kolumn

```
SELECT Wiek, Nazwisko
   FROM Pracownicy
   ORDER BY 3 DESC, 1 ASC;
```

Realizacja selekcji – wybór rekordów

Typowe przykłady operacji selekcji

```
SELECT Nazwisko, Wiek
   FROM Pracownicy
   WHERE Wiek = 65;

SELECT Nazwisko, Wiek
   FROM Pracownicy
   WHERE Nazwisko = 'Kowalski';

SELECT Nazwisko, Wiek
   FROM Pracownicy
   WHERE Nazwisko = 'Kowalski' AND Wiek > 60;

SELECT Nazwisko, Wiek, Stanowisko
   FROM Pracownicy
   WHERE Stanowisko = 'analityk' OR Stanowisko = 'programista';

SELECT Nazwisko, Wiek, Stanowisko
   FROM Pracownicy
   WHERE (Stanowisko = 'analityk' OR stanowisko = 'programista')
         AND Wiek < 25;

SELECT Nazwisko
   FROM Pracownicy
   WHERE (Stanowisko = 'analityk' OR stanowisko = 'programista')
         AND Wiek < 25 AND Jezyk2 IN ('francuski', 'niemiecki')
   ORDER BY Wiek DESC;
```

Konstruowanie warunku w klauzuli WHERE

Operatory relacyjne

`=, <, >, <=, >=, != (<>)` służą do porównywania liczb, dat, napisów; napisy muszą być zapisane z użyciem apostrofów. Zapis dat i godzin musi być zgodny z formatem stosowanym w SZBD.

Operatory logiczne

`AND, OR, NOT` wraz z nawiasami służą do konstrukcji złożonych warunków logicznych (algebraicznie – odpowiadających iloczynowi, sumie i dopełnieniu). Wyznacznie wartości logiczne przebiega od lewej do prawej (o ile nie ma nawiasów).

Operatory specjalne

`BETWEEN, LIKE, IN, IS NULL` – służą do definiowania warunków złożonych selekcji. Operator `LIKE` pozwla naporównywanie łańcuchów z użyciem symboli specjalnych `%` (dowolny ciąg znaków) oraz `_` (pojedynczy symbol). Wszystkie te operatory mogą być negowane (`NOT`).
Przykłady:

```
DataZatrudnienia BETWEEN '10/12/99' AND '17/01/00'
Nazwisko LIKE 'Kowal%'
StawkaVAT IN (0, 7, 22)
Grzech IN ('pycha', 'chciwość', 'nieczystość', 'zadrość',
'nieumiarkowanie w jedzeniu i piciu', 'gniew', 'lenistwo')
Telefon IS NULL
Telefon IS NOT NULL
```

Zastosowanie obliczeń w zapytaniach

W zapytaniach można umieścić wyrażenia definiujące standardowe operacje arytmetyczne oraz wykorzystujące funkcje.

```
SELECT NumerZam, ISBN, Ilosc, CenaJednost, (CenaJednost * Ilosc)
   FROM ZamowioneKsiazki
   WHERE NumerZam = 3;

SELECT Pracownik, (Zarobki * 12 + Prowizja) / 12
   FROM Pracownicy
   WHERE Stanowisko = 'Sprzedawca';

SELECT Pracownik, Zarobki, Prowizja
   FROM Pracownicy
   WHERE Prowizja < .25 * Zarobki;

SELECT Pracownik, Zarobki, 0.75 * (Zarobki + 550)
   FROM Pracownicy
   WHERE Stanowisko = 'kierownik'
     AND (Zarobki + 550) * 0.75 > 2500
   ORDER BY 3;

SELECT Nazwisko || ', ' || Imie
   FROM Osoby
   ORDER BY Nazwisko, Imie;

SELECT SUBSTRING (Imie FROM 1 FOR 1) || '. ' || Nazwisko
   FROM Osoby;

SELECT CURRENT_DATE - DataZamowienia DAY
   FROM Zamowienia
   WHERE Klient = 'Kowalski';
```

Operacje grupowania

Opcje `GROUP BY` oraz `HAVING` umożliwiają grupowanie wybranych rekordów (tzw. agregację). Możliwe jest użycie typowych funkcji agregujących: `SUM`, `AVG`, `MIN`, `MAX`, `COUNT`.

```
SELECT Stanowisko, AVG(Zarobki), COUNT(*)
   FROM Pracownicy
   WHERE Stanowisko != 'prezes'
   GROUP BY Stanowisko;
```

```
SELECT Stanowisko, AVG(Zarobki), '= Srednia zarobkow',
       COUNT(*), '= # pracownikow na danym satnowisku'
   FROM Pracownicy
   WHERE Stanowisko != 'prezes'
   GROUP BY Stanowisko
   HAVING AVG(Zarobki) < 2500;
```

```
SELECT Stanowisko, NumerDzialu, Count(*)
   FROM Pracownicy
   WHERE Stanowisko != 'prezes'
   GROUP BY Stanowisko, NumerDzialu
   HAVING Count(*) >1;
```

```
SELECT Klient, COUNT(*), SUM(Kwota)
   FROM Zamowienia
   GROUP BY Klient
   HAVING SUM(Kwota) > 1000;
```

```
SELECT CenaJednostkowa, COUNT(*)
   FROM Zamowienia
   WHERE CenaJednostkowa > 17
   GROUP BY CenaJednostkowa;
```

Operacje grupowania

W klauzuli `SELECT` grupującej dane można używać tylko nazw atrybutów dla których następuje grupowanie. W zależności od struktury tabeli oraz zawartych w niej danych i spodziewanego wyniku zapytania, istnieje możliwość wykorzystania `WHERE` lub `HAVING`; `WHERE` działa przed sformowaniem grup a `HAVING` po – predykat tej opcji musi więc odnosić się do kryteriów wykorzystanych przy tworzeniu grup.

```
SELECT CenaJednostkowa, COUNT(*)
   FROM Zamowienia
   WHERE CenaJednostkowa > 17
   GROUP BY CenaJednostkowa;
```

```
SELECT CenaJednostkowa, COUNT(*)
   FROM Zamowienia
   GROUP BY CenaJednostkowa
   HAVING CenaJednostkowa > 17;
```

Możliwe jest także jednoczesne użycie `WHERE` oraz `HAVING`, np.:

```
SELECT CenaJednostkowa, COUNT(*)
   FROM Zamowienia
   WHERE RokWydania > '1980'
   GROUP BY CenaJednostkowa
   HAVING CenaJednostkowa > 17;
```

Wyniki operacji grupujących mogą być zapamiętywane do dalszego przetwarzania, np.:

```
INSERT INTO DzialSrednia (NumerDzialu, DzialSrednieZarobki)
SELECT NumerDzialu, AVG(Zarobki)
   FROM Pracownicy
   GROUP BY NumerDzialu;
```

Zagnieżdżanie zapytań

Zapytania w SQL mogą operować na wynikach innych zapytań; możliwe jest więc tzw. zagnieżdżanie zapytań. Typowy schemat zagnieżdżania:

```
SELECT Kolumna/Kolumny
   FROM Tabela
   WHERE IN|NOT IN|ANY SELECT Kolumna/Kolumny
                          FROM Tabela
                          WHERE Warunek;
```

```
SELECT DataZlozeniaZamowienia, NumerKlienta
   FROM Zamowienia
   WHERE NumerZamowienia IN (
      SELECT NumerZamowienia
         FROM ZamowioneKsiazki
         WHERE ISBN = '83-204-1806-2'
                           );
```

```
SELECT Nazwisko, Imie
   FROM Klienci
   WHERE NumerKlienta IN (
      SELECT NumerKlienta
         FROM Zamowienia
         WHERE NumerZamowienia = ANY (
            SELECT Numer Zamowienia
               FROM ZamowioneKsiazki
               WHERE Rok = '1999'
 )
      );
```

Operatory `IN` oraz `ANY` pozwalają na przeszukanie pewnego zbioru; różnica pomiędzy nimi polega na możliwości zastosowania różnych operatorów relacyjnych (zamiast "=") dla `ANY`.

Złączenie

W specyfikacji tabeli, po słowie FROM można zdefiniować dowolne złączenie tabel (naturalne, θ-złączenie, iloczyn kartezjański; wewnętrzene, zewnętrzne i typu *union*). Składnia odpowiednich wyrażeń jest następująca.

- złączenie krzyżowe: TablicaA `CROSS JOIN` TablicaB,
- złączenie naturalne: TablicaA `[NATURAL]` [typ złączenia] `JOIN` TablicaB,
- złączenie union: TablicaA `UNION JOIN` TablicaB,
- złączenie przez predykat : TablicaA [typ złączenia] `JOIN` TablicaB `ON` predykat,
- złączenie po zadanej kolumnie: TablicaA [typ złączenia] `JOIN` TablicaB `USING` (nazwa kolumny,...),
- typy złączenia: `INNER`| `{LEFT|RIGHT|FULL} [OUTER]`.

Przykłady:

```
SELECT Imie, Nazwisko, NumerZamowienia, DataZamowienia
   FROM Klienci, Zamowienia
   WHERE Klienci.NumerKlienta = Zamowienia.NumerKlienta
      AND Nazwisko IN ('Kowalski', 'Malinowski', 'Nowak');

SELECT Imie, Nazwisko, NumerZamowienia, DataZamowienia
   FROM Klienci JOIN Zamowienia
   ON Klienci.NumerKlienta = Zamowienia.NumerKlienta;

SELECT Imie, Nazwisko, NumerZamowienia, DataZamowienia
   FROM Klienci T1 LEFT OUTER JOIN Zamowienia T2
   ON (T1.NumerKlienta = T2.NumerKJlienta);
```

QQQ

TTT

QQQ

TTT