Paweł Rajba
pawel@ii.uni.wroc.pl
http://www.itcourses.eu/

Podstawy SQL

Agenda

- Wprowadzenie
- Historia i standardy
- Podstawy relacyjności
- Typy danych
- DDL
  - tabele, widoki, sekwencje
  - zmiana struktury
- DML
- DQL
  - Podstawy, złączenia, podzapytania, grupowanie, operacje na zbiorach

Wprowadzenie

- SQL, czyli Structured Query Language
- Umożliwia
  - Definiowanie struktur bazy danych
    - DDL – Data Definition Language
  - Operowanie na danych: dodawanie, udostępnianie, modyfikowanie i usuwanie
    - DML – Data Manipulation Language
  - Zarządzanie dostępem do danych
    - DCL – Data Control Language
  - Definiowanie zapytań
    - DQL – Data Query Language

Historia i standardy

- Język pojawił się oficjalnie w 1974 jako Structured English Query Language – SEQUEL
  - Twórcy: Donald D. Chamberlin i Raymond F. Boyce
- 1987: SQL-86 jako oficjalny standard ISO/ANSI
- Kolejne standardy:
  - SQL-89, SQL-92, SQL:1999
  - SQL:2003, SQL:2006, SQL:2008, SQL:2011
    - http://en.wikipedia.org/wiki/SQL#Standardization
- Przyjrzymy się językowi SQL w wersji Oracle

Podstawy relacyjności

- Relacyjna baza danych składa się z:
  - Tabelek (a.k.a. encji, relacji)
  - Powiązań między nimi
  - … i mnóstwem innych obiektów, ale podstawa to tabelki
- Tabela
  - Kolumny, wiersze
  - Klucz główny
  - Więzy integralności
- Powiązania między tabelami, czyli klucz obcy
  - Ksiazka(isbn, tytul, autor, rok_wydania, cena)
  - Egzemplarz(sygnatura, isbn)
    → {Egzemplarz.isbn} ⊂ {Ksiazka.isbn}

Typy danych

- Wbudowane typy danych
  - VARCHAR2(size), NVARCHAR2(size)
  - NUMBER [(p [, s])], FLOAT [(p)]
  - LONG, INT
  - DATE
  - BINARY_FLOAT, BINARY_DOUBLE
  - TIMESTAMP
  - RAW(size), LONG RAW
  - ROWID
  - CHAR [(size)], NCHAR[(size)]
  - CLOB, NCLOB
  - BLOB, BFILE
    - Więcej: http://docs.oracle.com/database/121/SQLRF/sql_elements001.htm#SQLRF30020
- Można także tworzyć własne typy dane

DDL: Tabele

```
CREATE TABLE <nazwa>
(
    <kolumna1> <typ1> [<więzy kolumny 1>],
    …
    <kolumnaN> <typN> [<więzy kolumny N>],
    [<więzy tabeli>]
)
```

DDL: Tabele

- Więzy kolumny

```
[CONSTRAINT <nazwa więzu>]
    NOT NULL |
    DEFAULT <wartość domyślna> |
    PRIMARY KEY |
    UNIQUE |
    CHECK (<warunek>) |
    REFERENCES <tabela> [(<kolumna>)]
            [ON DELETE {SET NULL | CASCADE}]
```

- Więzy tabeli

```
[CONSTRAINT <nazwa więzu>]
    PRIMARY KEY (<lista kolumn>) |
    UNIQUE (<lista kolumn>) |
    FOREIGN KEY (<lista kolumn>)
            REFERENCES <tabela>(<lista kolumn>)
                    [ON DELETE {SET NULL | CASCADE}]|
    CHECK (<warunek>)
```

DDL: Tabele

```
CREATE TABLE KSIAZKA
( KSIAZKA_ID INT
, ISBN VARCHAR2(20)
, TYTUL VARCHAR2(300)
, AUTOR VARCHAR2(200)
, ROK_WYDANIA NUMBER(4)
, CENA NUMBER(10,2)
, CONSTRAINT KSIAZKA_PK PRIMARY KEY (KSIAZKA_ID)
, CONSTRAINT KSIAZKA_UK_ISBN UNIQUE (ISBN)
);

CREATE TABLE EGZEMPLARZ
( EGZEMPLARZ_ID INT
, SYGNATURA CHAR(8)
, KSIAZKA_ID INT
, CONSTRAINT PK_KSIAZKA PRIMARY KEY (KSIAZKA_ID)
, CONSTRAINT EGZEMPLARZ_UK_SYGNATURA UNIQUE (SYGNATURA)
, CONSTRAINT KSIAZKA_FK FOREIGN KEY (KSIAZKA_ID)
    REFERENCES KSIAZKA (KSIAZKA_ID) ON DELETE CASCADE
);
```

DDL: Widoki

CREATE VIEW <nazwa widoku> AS
    <select statement>

http://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_8004.htm#SQLRF01504

DDL: Sekwencje

```
CREATE SEQUENCE KSIAZKA_PKSEQ
        INCREMENT BY 1
        START WITH 1
        NOMAXVALUE
        MINVALUE 1
        CYCLE
        CACHE 20
        ORDER;
```

DDL: zmiana struktury

- ALTER TABLE <nazwa tabeli>
  ADD COLUMN <kol> <typ>
  RENAME COLUMN <kol1> TO <kol2>
  DROP COLUMN <nazwa kolumny>
  ADD CONSTRAINT <nazwa> CHECK (<wiez>)
  …
- DROP TABLE <nazwa tabeli>
- DROP SEQUENCE <nazwa>
- …

DML

INSERT INTO t (k1,k2,…) VALUES(v1,v2,…)
UPDATE t SET k1=v1,k2=v2,… WHERE <warunek>
DELETE FROM t WHERE <warunek>

CRUD=

- CREATE
- RETRIEVE
- UPDATE
- DELETE

DQL: Podstawy

- Podstawowa składnia
  - SELECT kolumna1, kolumna2, …
    FROM tabela1
    WHERE warunki
    ORDER BY 1,2,…
- Jakie mogą być warunki?
  - Operatory: =, <, >, <=, >=, IN (s1,s2,…), LIKE napis, BETWEEN … AND …
    - Cena < 1000,00
    - Nazwisko LIKE 'A%'
    - Gatunek IN ('muzyka', 'sf', 'biografie')
    - Wiek BETWEEN 30 AND 40

DQL: Podstawy

- Funkcje agregujące
  - COUNT, AVG, SUM, MIN, MAX
- Znaczenie słowa DISTINCT
- Znaczenie „tabeli” DUAL
  - SELECT 2+3 as wynik FROM dual

DQL: Podstawy

- Lista książek o cenie do 60 zł, które mają w tytule SQL Server
  - SELECT *
    FROM Ksiazka
    WHERE Tytul LIKE '%SQL Server%' AND Cena<60
- Najdroższa i najtańsza książka
  - SELECT MIN(Cena) AS Najtańsza, MAX(Cena) AS Najdroższa
    FROM Ksiazka
- Średni czas wypożyczenia
  - SELECT AVG(LICZBA_DNI) AS "Średni czas wypożyczenia"
    FROM Wypozyczenie

DQL: Złączenia

- Złączenie polega na wyświetleniu w wynikowej tabelce danych z wielu tabel
- Jak ta tabelka jest budowana? Mamy kilka rodzajów złączeń:
  - Pełne
  - Wewnętrzne
  - Zewnętrzne lewe i prawe

DQL: Złączenia

- Pełne
  - SELECT k.*, e.*
    FROM Ksiazka k, Egzemplarz e
- Wewnętrzne
  - SELECT k.*, e.*
    FROM Ksiazka k, Egzemplarz e
    WHERE k.ksiazka_id=e.ksiazka_id
  - SELECT k.*, e.*
    FROM Ksiazka k INNER JOIN Egzemplarz e
    ON k.ksiazka_id=e.ksiazka_id
- Zewnętrzne
  - SELECT k.*, e.*
    FROM Ksiazka k LEFT JOIN Egzemplarz e
    ON k.ksiazka_id=e.ksiazka_id

DQL: Złączenia

- Tytuły i liczby dni, na które książki były wypożyczane
  - SELECT k.Tytuł, w.dni
    FROM Wypozyczenie w, Egzemplarz e, Ksiazka k
    WHERE w.egzemplarz_id = e.egzemplarz_id AND
    k.ksiazka_id = e.ksiazka_id
  - SELECT k.tytul, w.liczba_dni
    FROM Wypozyczenie w JOIN Egzemplarz e
    ON w.egzemplarz_id = e.egzemplarz_id JOIN Ksiazka k
    ON k.ksiazka_id = e.ksiazka_id
- Cena wszystkich książek (10)
  - SELECT SUM(Cena)
    FROM Ksiazka k JOIN Egzemplarz e
    ON k.ksiazka_id = e.ksiazka_id

DQL: Złączenia

- Inny przykład
  - Nośnik reklamowy można rezerwować dokładnością do 10 minut
  - Rezerwację na zakres pewnej liczby dni chcemy modelować listą slotów 10 minutowych
    - Z której możemy potem usuwać pojedyncze sloty
- Jak to zrobić?

DQL: Złączenia

- Rozwiązanie
  - Tworzymy dwie tabele:
    - Slownik_Dni
    - Slownik_Slotow_Na_Dzien
  - Dane: dzien_od, dzien_do
  - Wstawianie rezerwacji:
    - INSERT INTO rezerwacja (…)
      SELECT … FROM Slownik_Dni, Slownik_Slotow_Na_Dzien
      WHERE Slownik_Dni.dzien
      BETWEEN dzien_od and dzien_do

DQL: Podzapytania

- Kiedy występuje podzapytanie?
- Rodzaje podzapytań
  - Nieskorelowane
    - SELECT czytelnik_id, nazwisko FROM czytelnik
      WHERE czytelnik_id IN
      (SELECT DISTINCT czytelnik_id
      FROM WYPOZYCZENIE);
  - Skorelowane
    - SELECT c.czytelnik_id, c.nazwisko FROM czytelnik c
      WHERE EXISTS
      (SELECT 1 FROM WYPOZYCZENIE w
      WHERE w.czytelnik_id=c.czytelnik_id);

DQL: Podzapytania

- Inny przykład
  - SELECT * FROM
    (SELECT * FROM Ksiazka
    WHERE ROK_WYDANIA>2008)

DQL: Podzapytania

- Operatory
  - ANY/SOME, ALL
    - substr(nazwisko,1,1) = ANY('A', 'B','C')
    - pensja > ALL(SELECT pensja FROM pracownik)
  - EXISTS, NOT EXISTS
    - EXISTS(SELECT * FROM Zamowienia WHERE …)
  - IN, NOT IN
    - substr(nazwisko,1,1) IN ('A', 'B','C')
    - telefon NOT IN (SELECT telefon FROM pracownik)

http://stackoverflow.com/questions/2298550/oracle-any-vs-in
http://awads.net/wp/2005/08/09/any-some-and-all-in-oracle/

DQL: Grupowanie

- Wynik zapytania dzielimy na grupy
- Dla każdej grupy wykonujemy agregację
- Przykłady
  - Proste podliczenie liczby egzemplarzy
    - SELECT ksiazka_id, COUNT(egzemplarz_id) as "Liczba egzemplarzy"
      FROM egzemplarz
      GROUP BY ksiazka_id;
    - SELECT k.tytul, COUNT(e.egzemplarz_id) as "Liczba egzemplarzy"
      FROM ksiazka k JOIN egzemplarz e ON k.ksiazka_id = e.ksiazka_id
      GROUP BY k.tytul;
      - Jaka jest różnica?
      - Czy w drugim przypadku ten JOIN jest najlepszym pomysłem?

DQL: Grupowanie

- Klauzula HAVING umożliwia określanie grup, które mają się znaleźć w wyniku
- Jaka jest zależność pomiędzy WHERE i HAVING?
- Przykłady
  - SELECT k.tytul, COUNT(e.egzemplarz_id) as "Liczba egzemplarzy"
    FROM ksiazka k JOIN egzemplarz e ON k.ksiazka_id = e.ksiazka_id
    GROUP BY k.tytul
    HAVING COUNT(e.egzemplarz_id)>=3
  - SELECT k.tytul, COUNT(e.egzemplarz_id) as "Liczba egzemplarzy"
    FROM ksiazka k JOIN egzemplarz e ON k.ksiazka_id = e.ksiazka_id
    WHERE e.egzemplarz_id NOT IN
      (SELECT egzemplarz_id FROM wypozyczenie)
    GROUP BY k.tytul
    HAVING COUNT(e.egzemplarz_id)>=3

DQL: Operacje na zbiorach

- UNION [ALL], INTERSECT, MINUS
  - SELECT tytul FROM ksiazka WHERE substr(tytul, 1, 1)='S'
  - UNION | UNION ALL | INTERSECT | MINUS
  - SELECT tytul FROM ksiazka WHERE substr(tytul, -1) in ('w','a','k')