BAZY DANYCH
wprowadzenie do języka SQL

Opracował:

dr inż. Piotr Suchomski

Wprowadzenie

- Język SQL używany jest do pracy z relacyjną bazą danych. Jest to język nieproceduralny, należący do grupy języków deklaratywnych. Semantyka SQL wyraża, co ma być zrobione, a nie jak. Problem „jak” przeniesiony został na poziom systemu zarządzania bazą danych.
- Historia SQL’a rozpoczęła się w zeszłym stuleciu. W ramach prac w firmie IBM nad językiem do obsługi baz danych powstał język SEQUEL (Structured English Query Language). Język następnie został rozwinięty i nazwany jako SQL (Structured Query Language - Strukturalny Język Zapytań)

Standardy języka

- Istnieje wiele różnych dialektów języka SQL. Najważniejsze z nich to ANSI SQL oraz zmodyfikowany w 1992 standard SQL-92 (SQL2), później SQL-99(SQL3) . Jedną z ostatnich wersji standardu jest SQL:2003 (ISO/IEC 9075-X:2003).
- Ponadto istnieje wiele wersji języka SQL implementowane przez wiodących producentów systemów baz danych. Wszystkie zgodne są ze standardem ANSI. Oprócz tego mają wiele dodatkowych funkcji.

Struktura języka

- Język definiowania danych (DDL – Data Definition Language)
  - CREATE TABLE – tworzy nowe tabele
  - CREATE VIEW – tworzenie widoku
  - CREATE INDEX – tworzy indeks
  - ALTER TABLE – modyfikuje (zmienia) tabele
  - DROP TABLE – usuwa tabele
  - DROP VIEW – usuwa widok
  - DROP INDEX – usuwa indeks

Struktura języka – cd.

- Język manipulowania danymi (DML – Data Manipulation Language)
  - SELECT – pozwala wydobyć dane z bazy danych
  - UPDATE – uaktualnia/modyfikuje dane w bazie danych
  - DELETE – usuwa dane z bazy danych
  - INSERT INTO – wstawia nowe dane do tabeli bazy danych
- Język kontroli danych (DCL – Data Control Language)
  - GRANT – nadawanie uprawnień
  - REVOKE – odbieranie uprawnień

Tworzenie tablic

- Aby utworzyć nową tablicę w bazie danych należy wykonać polecenie CREATE TABLE, w którym należy określić:
  - Nazwę tablicy (unikalną w całej bazie danych),
  - Nazwy kolumn (unikalne w tablicy),
  - Typ danych każdej kolumny.

```
CREATE TABLE <nazwa tablicy>
    {<nazwa kolumny> <typ danych>[opcje],
     [<nazwa kolumny> <typ danych> [opcje]…]
     [UNIQUE {<nazwa kolumny>,[<nazwa
    kolumny>]}]…}
```

Tworzenie tablic - opcje

- NOT NULL – w kolumnie zawsze musi być wartość,
- UNIQUE – wartości zapisywane w kolumnie muszą być unikatowe dla każdego wiersza,
- PRIMARY KEY – ma znaczenie jak połączenie NOT NULL i UNIQUE,
- CHECK(warunek) – w czasie wprowadzania wartości sprawdza zadany warunek
- DEFAULT wartość – możliwość zdefiniowania wartości domyślnej,
- REFERENCES – określenie ograniczeń dla klucza obcego, wymaga aby wskazywana wartość była unikatowa.

Tworzenie tablic - przykład

```
CREATE TABLE Zamówienia
{
  zam_id serial,
  klient_id integer NOT NULL,
  data_zam date NOT NULL,
  data_wys date,
  CONSTRAINT zam_info PRIMARY KEY(zam_id),
  CONSTRAINT zam_klient_info
  FOREIGN KEY(klien_id) REFERENCES
  Klienci(klient_id)
};
```

Zmiana schematu tablicy

- Niestety zazwyczaj nie udaje się zaprojektować tak schematu bazy danych aby nie wymagała późniejszej modyfikacji. Modyfikacja struktury tabeli jest możliwa za pomocą ALTER TABLE:

```
ALTER TABLE nazwa tabeli ADD COLUMN nazwa kolmny typ kolumny
ALTER TABLE nazwa tabeli DROP COLUMN nazwa kolumny
ALTER TABLE nazwa tabeli RENAME stara_nazwa_kolumny TO nowa nazwa
ALTER TABLE stara_nazwa_tabeli RENAME TO nowa_nazwa_tabeli
```

- W nowych wersjach języka można praktycznie modyfikować każdy parametr.
- Należy pamiętać, że taka przebudowa jest często kosztowną i ryzykowną operacją.

Wprowadzanie danych do tabeli

- Do wstawiania nowego wiersza danych do tablicy służy polecenie INSERT INTO:

```
INSERT INTO nazwa_tabeli VALUES(lista
  wartości)
INSERT INTO nazwa_tabeli(lista nazw kolumn)
  VALUES(lista wartości)
```

- Druga forma mimo, że jest dłuższa, to jest bezpieczniejsza.

```
INSERT INTO Nauczyciele VALUES(30389,
  'Piotr', 'Suchomski','dr','multimedia');
INSERT INTO Nauczyciele(id, nazwisko,
  stopien) VALUES(30389,'Suchomski', 'dr');
```

Zapytania w języku SQL

- Zapytania w SQL służą do wybierania krotek pewnej relacji, które spełniają określony w zapytaniu warunek. Zapytania są realizacją operacji selekcji w algebrze relacyjnej.
- Konstrukcja typowego zapytania składa się z sekwencji trzech słów kluczowych SELECT, FROM,WHERE.
-

Zapytania w języku SQL

- Klauzula FROM służy do określenia relacji, których zapytanie dotyczy,
- Klauzula WHERE określa warunek, który odpowiada warunkowi selekcji w algebrze relacyjnej. Definiuje on ograniczenia, jakie muszą spełniać krotki, aby zostały wybrane w danym zapytaniu,
- W klauzuli SELECT zostaje określona lista atrybutów, których wartości z krotek spełniających warunek zapytania są dołączone do odpowiedzi (znak * oznacza wszystkie atrybuty relacji, cała krotka jest umieszczona w odpowiedzi)

Sposób czytania/pisania zapytania

- Na ogół jest łatwiej sprawdzać zapytania select-from-where, zaczynając od klauzuli FROM, dzięki czemu dowiadujemy się, których relacji dotyczy dane zapytanie.
- Następnie sprawdzamy klauzulę WHERE, aby wiedzieć, które krotki są w zapytaniu istotnie.
- Dopiero na samym końcu przeglądamy klauzulę SELECT, gdzie określa się wyjście.
- Tak samo postępuje się w czasie pisania zapytań

Składnia SELECT

SELECT[DISTINCT] <lista kolumn/wyrażeń>

FROM <lista tablic>

[WHERE <warunek>]

[GROUP BY <lista kolumn>]

[HAVING <warunek>]

[UNION <instrukcja SELECT>]

[ORDER BY <lista kolumn>

Rzutowanie w języku SQL

- W języku SQL projekcja jest realizowana przez wskazanie, które kolumny maja być pokazane w wyniku. Znak * oznacza, że w wyniku mają być wyświetlone wszystkie kolumny analizowanych tablic.

```
SELECT id, nazwisko FROM Nauczyciele WHRE id = 30890;
```

- Można zażądać aby nazwy wyświetlanych kolumn były inne używając konstrukcji ‚nazwa kolumny> AS <nazwa wyświetlana>

```
SELECT id AS numer ewidencyjny, nazwisko FROM Nuaczyciele
```

Selekcja w języku SQL

- Klauzula WHERE ma nieco szersze znaczenie niż w algebrze relacyjnej. Można w języku SQL używać wyrażeń znanych z języków programowania.
- Porównując wartości można użyć jednego z sześciu operatorów: =, <>, <, >, <=, >=.
- W wyrażeniach można używać operatorów logicznych jak AND, OR, NOT.
- W wyrażeniu warunku mogą występować stałe wartości jak i atrybuty tych relacji, które są wymienione w klauzuli FROM.

Inne przydatne wyrażenia

- IN – sprawdza czy wartość kolumny znajduje się na zadanej liście wartości.

```
SELECT nazwisko, miasto FROM Osoby WHERE
  miasto IN(„Gdańsk”, „Wrocław”, „Kraków”);
```

- BETWEEN – sprawdza czy wartość w kolumnie znajduje się w zadanym przedziale.

```
SELECT nazwisko, wiek FROM Osoba WHERE
  wiek BETWEEN 18 AND 30
```

- W sQL dostępne są również operatory pozwalające porównywać z wartością pustą NULL i NOT NULL.

Funkcje agregujące

- COUNT() – oblicza ilość wierszy w kolumnie dla wartości różnych od NULL.
- MIN() – wyznacza najmniejszą wartość w kolumnie,
- MAX() – wyznacza wartość maksymalną w kolumnie,
- SUM() – oblicza sumę wszystkich wartości w kolumnie,
- AVG() – oblicza wartość średniej arytmetycznej wszystkich wartości w danej kolumnie.
- Zarówno SUM() jak AVG() mogą być użyte z konstrukcja DISTINCT.

Porządkowanie wyniku

- Krotki będące wynikiem zapytania można posortować za pomocą klauzuli ORDER BY.
- Domyślnie porządkowanie jest rosnące (ascending ASC), ale można zmienić na porządkowanie malejące (descending DESC.

```
SELECT imie, nazwisko, wiek FROM Osoby
ORDER BY nazwisko, imie, wiek DESC
```

- W wyniku rzutowania i selekcji w wyniku zapytania mogą pojawić się powtórzenia krotek, aby tego uniknąć można użyć konstrukcji SELECT DISTINCT, ale w przypadku dużych zbiorów danych jest to mało efektywna metoda.

Iloczyn kartezjański i złączenie

- W języku SQL w łatwy sposób można tworzyć zapytania dotyczące wielu relacji. Wystarczy nazwy tych relacji umieścić po klauzuli FROM. Wówczas klauzule SELECT i WHERE mogą odnosić się do dowolnej relacji umieszczonej po słowie FROM.
- Analogicznie jak w algebrze relacji niejednoznaczność nazw kolumn można rozwiązać przez stosowanie zapisu z nazwą tablicy i nazwą kolumny po kropce.

Suma, iloczyn, różnica

- Jeśli w wyniku zapytań SQL powstają relacje o takich samych zbiorach atrybutów, to można do wyników tych zapytań stosować operatory, które odpowiadają teoriomnogościowym operatorom algebry relacyjnej.
- W języku SQL do oznaczenia tych operatorów używa się odpowiednio słów kluczowych: UNION (suma), INTERSECT (przecięcie) i EXCEPT (różnica).

```
(SELECT nazwisko, miasto FROM Osoby)

INTERSECT

(SELECT nazwisko, miasto FROM Studenci)
```

Grupowanie

- Grupowanie polega na łączeniu ze sobą krotek wynikowych z instrukcji SELECT w grupy wierszy, w których wskazane we frazie GROUP BY kolumny mają tą samą wartośc. Następnie każda gupa redukowana jest do jednego wiersza, w którym występują kolumny z frazy GROUP BY oraz ewentualnie wyniki funkcji agregujących. Na koniec eliminowane są grupy nie spełniające warunku HAVING.

GROUP BY <nazwa kolumny> [<nazwa kolumny>]

[HAVING <warunek>]

Grupowanie – cd.

- Funkcje agregujące działają dla każdej grupy osobno.
- Każda kolumna występująca w frazie GROUP BY musi wystąpić w klauzuli SELECT i na odwrót.
- W warunku HAVING może wystąpić funkcja agregująca

Kolejność klauzul

- Generalnie w zapytaniach SQL może występować sześć klauzul : SELECT, FROM, WHERE, GRUP BY, HAVING, ORDER BY.
- Tylko dwie pierwsze klauzule są potrzebne do poprawnego zbudowania zapytania. Jeśli w zapytaniu występują pozostałe klauzule to muszą one być w podanej wyżej kolejności.