Politechnika Śląska
Instytut Informatyki

Język SQL

instrukcja laboratoryjna

laboratorium Bazy Danych

przygotowali:
mgr inż. Paweł Kasprowski
(Kasprowski@zti.iinf.polsl.gliwice.pl)
mgr inż. Bożena Małysiak
(bozena@ivp.iinf.polsl.gliwice.pl)

Wstęp

Język SQL (Structured Query Language) jest najbardziej znanym językiem zapytań, zaimplementowanym w praktycznie wszystkich istniejących na rynku systemach relacyjnych baz danych.

SQL jest używany jako samodzielny język służący do interakcyjnych zapytań, tworzenia i aktualizacji relacyjnej bazy danych (i w ten sposób będzie wykorzystywany podczas ćwiczeń laboratoryjnych). Może być również zanurzany (ang. embedded) w klasyczne języki programowania. W językach 4GL (np. INFORMIX 4GL) jest zintegrowany ze środowiskiem 4GL.

Instrukcje języka SQL podzielić można na kilka typów:

- instrukcje wyszukiwania danych - SELECT
- instrukcje tworzenia bazy (Data Description Language) – np.: CREATE TABLE
- instrukcje modyfikacji danych (Data Modification Language) – np.: UPDATE
- instrukcje do określania praw dostępu i więzów integralności (Data Control Language) – np.: GRANT

Podczas laboratorium SQL1 i SQL2 wykorzystywać się będzie głównie instrukcję wyszukiwania SELECT.

Instrukcja SELECT

Instrukcja wyszukiwania SELECT służy do generowania zapytań do bazy danych. Po jej wykonaniu powstaje tablica wynikowa zawierająca żądane atrybuty pobrane z wierszy spełniających podane warunki. Jej najprostsza forma to:

```
SELECT <atrybuty>
FROM <tabela>
WHERE <warunek>
```

Przykładowo zapytanie:

```
SELECT Nazwisko FROM Pracownicy
```

powoduje pobranie wartości atrybutu Nazwisko z wszystkich rekordów tablicy Pracownicy. Aby uzyskać wszystkie atrybuty tablicy w miejsce listy atrybutów użyć należy znaku '*':

```
SELECT * FROM Pracownicy
```

Powyższe zapytanie zwraca jako wynik całą tablicę Pracownicy.

Aby pobrać tylko określone rekordy użyć należy frazy WHERE:

```
SELECT * FROM Pracownicy WHERE NumerPrac>10
```

Powyższe zapytanie pobiera z tablicy tylko rekordy spełniające podany warunek.

Przy porównywaniu wartości tekstowych z podanym wzorcem można we wzorcu używać znaków specjalnych '_' – odpowiednik jednego dowolnego znaku oraz '%' odpowiednik dowolnego ciągu znaków. W niektórych systemach przyjmuje się symbolikę pochodzącą z systemu DOS, odpowiednio '?' i '*'. Do porównań wartości tekstowych służy fraza LIKE.

```
SELECT * FROM Pracownicy WHERE Nazwisko LIKE „Kowal%"
```

Inne operatory używane po frazie WHERE to na przykład operator IN (zbiór wartości)

```
SELECT * FROM Pracownicy WHERE NumerPrac IN (100,101,200)
```

lub operator BETWEEN x AND y

```
SELECT * FROM Pracownicy WHERE NumerPrac BETWEEN 100 AND 200
```

Sortowanie

Do sortowania danych używa się frazy ORDER BY. Występuje ona zawsze na końcu zapytania:

```
SELECT * FROM Pracownicy WHERE NumerPrac > 10
ORDER BY Nazwisko
```

Zapytanie powyższe zwróci tablicę wartości posortowaną alfabetycznie według nazwisk. Można sortować tablicę wartości według więcej niż jednego atrybutu

```
SELECT * FROM Pracownicy WHERE NumerPrac > 10
ORDER BY NumerZesp, Nazwisko
```

W tym przypadku tablica posortowana zostanie według numerów zespołów a w ramach jednego zespołu według nazwisk. Do sortowania w odwrotnej kolejności służy znacznik DESC.

```
SELECT * FROM Pracownicy WHERE NumerPrac > 10
ORDER BY NumerPrac DESC
```

Łączenie tablic

W zapytaniu dane pobierać można z więcej niż jednej tablicy jednocześnie. Należy wtedy zdefiniować warunek łączący te tabele.

```
SELECT Nazwisko, NazwaZesp FROM Pracownicy P, Zespoły Z
WHERE P.NumerZesp=Z.NumerZesp
```

Powyższe zapytanie zwróci listę nazwisk pracowników i nazw zespołów w których pracują. Zapytanie tworzy najpierw iloczyn kartezjański dwóch tablic według podanego warunku a następnie dokonuje odpowiednich selekcji i projekcji. Ponieważ pole NumerZesp występuje w obydwu tablicach, dla każdego wystąpienia w zapytaniu należy określić z której tablicy ma być pobrane. W przykładzie skorzystano z możliwości tworzenia tzw. aliasów – identyfikatorów tabel. Zapis ‘Pracownik P’ powoduje, że tablica Pracownik jest w dalszej części zapytania widziana pod identyfikatorem ‘P’.

Użycie aliasów pozwala na otwarcie więcej niż jednej instancji tej samej tablicy. Przykładowo, aby znaleźć pracowników należących do zespołów o numerze zespołu większym niż pracownik o numerze pracowniczym 5 można zastosować zapytanie:

```
SELECT P.NumerPrac, P.Nazwisko
FROM Pracownicy P, Pracownicy P1
WHERE P1.NumerPrac=5  AND P.NumerZesp>P1.NumerZesp
```

Funkcje agregujące

Po frazie SELECT można w miejsce atrybutów użyć tzw. funkcji agregujących. Są to funkcje SUM(), AVG(), COUNT(), MAX() i MIN().

```
SELECT SUM(Pensja) FROM Pracownicy
```

Powyższe zapytanie zwróci jeden rekord z jednym polem zawierającym sumę zawartości pola Pensja we wszystkich rekordach tabeli Pracownicy.

```
SELECT COUNT(*) FROM Pracownicy
```

Powyższe zapytanie zwróci ilość rekordów w tabeli Pracownicy.

Wartości unikalne

Aby w zapytaniu uzyskać jedynie wartości unikalne danego atrybutu lub grupy atrybutów należy użyć frazy DISTINCT. Frazę tę użyć można na dwa sposoby. Pierwszy sposób to umieszczenie jej bezpośrednio za słowem SELECT:

```
SELECT DISTINCT Nazwisko FROM Pracownicy
```

Zapytanie powyższe zwróci zbiór nazwisk pracowników bez powtórzeń dla tych samych nazwisk.

```
SELECT DISTINCT Imię, Nazwisko FROM Pracownicy
```

Zapytanie powyższe zwróci zbiór unikalnych par imię - nazwisko. Dla tego samego nazwiska i różnych imion pojawią się dwa rekordy, np.: Jan Kowalski i Piotr Kowalski.

Inne zastosowanie frazy DISTINCT to użycie jej wewnątrz funkcji agregujacej:

```
SELECT SUM(DISTINCT Pensja) FROM Pracownicy
```

Zapytanie powyższe zwróci sumę różnych pensji. Na przykład dla zbioru 100,200,300,200,300 zapytanie to zwróci wartość 600 podczas, gdy zapytanie bez frazy DISTINCT zwróciłoby wartość 1100.

Grupowanie

Za pomocą instrukcji SELECT można także grupować dane. Aby uzyskać informacje na temat sum dochodów pracowników w poszczególnych zespołach można zmodyfikować poprzednie zapytanie dodając frazę GROUP BY:

```
SELECT SUM(Pensja) FROM Pracownicy GROUP BY NumerZesp
```

Powyższe zapytanie zwróci jeden rekord dla każdej wartości pola NumerZesp. W rekordzie tym

będzie suma pola Pensja dla wszystkich rekordów tablicy o danej zawartości NumerZesp.

```
SELECT P.NumerZesp, NazwaZesp, Sum(Pensja)
FROM  Pracownicy P, Zespoły Z
WHERE P.NumerZesp=Z.NumerZesp AND NazwaZesp<>”Piece”
GROUP BY P.NumerZesp, NazwaZesp
```

Powyższe zapytanie zwróci tablicę z trzema kolumnami: numer zespołu, nazwa zespołu, suma zarobków w zespole. Zauważyć należy, że wszystkie atrybuty występujące bezpośrednio po frazie SELECT na których nie dokonuje się agregacji muszą znaleźć się we frazie GROUP BY. W tym przykładzie zasada ta dotyczy pola NazwaZesp.

W języku SQL istnieje możliwość nałożenia warunku na całą grupę za pomocą frazy HAVING. Przykładowo zapytanie:

```
SELECT P.NumerZesp, Sum(Pensja)
FROM  Pracownicy P
GROUP BY P.NumerZesp
HAVING P.NumerZesp>10
```

zwróci sumy zarobków tylko w zespołach, których numer jest większy niż 10.

Oprócz zwykłych warunków w warunkach frazy HAVING można stosować także funkcje agregujące.

```
SELECT P.NumerZesp, Sum(Pensja)
FROM  Pracownicy P
GROUP BY P.NumerZesp
HAVING Sum(P.Pensja)>1000
```

Zapytanie zwróci tylko zespoły w których suma pensji jest większa od 1000.

```
SELECT P.NumerZesp, Sum(Pensja)
FROM  Pracownicy P
```

```
GROUP BY P.NumerZesp
HAVING count(*)>5
```

Zapytanie zwróci tylko zespoły dla których ilość rekordów (ilość pracowników) jest większa od 5.

Nazywanie kolumn

Nazwy kolumn zapytania nie muszą być identyczne z nazwami atrybutów bazy. Można zdefiniować własne nazwy za pomocą frazy AS.

```
SELECT P.NumerZesp AS Zespół, Sum(Pensja) as Zarobki
FROM  Pracownicy P
GROUP BY P.NumerZesp
```

Zagnieżdżanie instrukcji

Instrukcja SELECT może być zagnieżdżona w obrębie innej instrukcji SELECT np.

```
SELECT Nazwisko
FROM Pracownicy
WHERE NumerZesp IN
    (SELECT NumerZesp
    FROM Zespoły
    WHERE Kierownik= „MISIURA”);
```

Zapytanie zagnieżdżone ujęte w nawiasy może być traktowane jako zbiór wartości. W takim przypadku powinno zawracać tylko jedną kolumnę danych. Odwoływać się do niego można przez porównanie z którymś z atrybutów zapytania zewnętrznego:

atrybut IN (zapytanie) – wartość znajduje się w tablicy wyników zapytania.

atrybut > ALL (zapytanie) – wartość jest większa od wszystkich elementów tablicy wyników zapytania.

atrybut > SOME|ANY (zapytanie) – wartość jest większa od przynajmniej jednego elementu tablicy wyników zapytania.

Zamiast operatora '>' można użyć innych operatorów porównania.

Innym sposobem użycia zagnieżdżonej instrukcji SELECT jest zastosowanie jej we frazie EXISTS (zapytanie) lub NOT EXISTS (zapytanie). Frazy takie użyte bezpośrednio jako warunki są prawdziwe jeśli odpowiednio istnieje lub nie istnieje chociaż jeden rekord w tabeli wynikowej zapytania zagnieżdżonego.

```
SELECT Nazwisko
FROM Pracownicy P
WHERE EXISTS
    (SELECT NumerZesp
    FROM Zespoły Z
    WHERE Z.NumerZesp>P.NumerZesp)
```

Zapytanie to zwraca nazwiska pracowników, dla których istnieją zespoły o numerach większych niż numer zespołu do którego należą.

Złączenie zewnętrzne

Zwykłe złączenie tablic przez podanie warunku łączącego jest nie zawsze wystarczające. Dla przykładu zapytanie

```
SELECT P.NumerPrac, Nazwisko, Imię, NazwaZwiązku
FROM Pracownicy P, Związki Z
WHERE P.NumerPrac=Z.NumerPrac
```

da w wyniku numery pracownicze tylko pracowników należących do związku. Aby uzyskać listę wszystkich pracowników z polem NazwaZwiązku wypełnionym w zależności od tego czy pracownik do jakiegoś należy, należy zastosować złączenie zewnętrzne tablic. Jest kilka „standardów" tworzenia złączeń zewnętrznych. Na laboratorium używa się serwera SQL Base firmy Centura i w tym przypadku (podobnie jest dla Oracle'a) złączenie zewnętrzne definiuje

się bezpośrednio przy podaniu warunku złączenia:

```
SELECT P.NumerPrac, Nazwisko, Imię, NazwaZwiązku
FROM Pracownicy P, Związki Z
WHERE P.NumerPrac=Z.NumerPrac(+)
```

Zapis t1.pole=t2.pole(+) oznacza, że do tablicy wynikowej dołączane będą tylko rekordy z tablicy t2 spełniające podany warunek. W pozostałych miejscach zamiast pól z t2 znajdą się wartości NULL.

Inną metodą definiowania złączeń zewnętrznych jest na przykład stosowany w systemie Informix zapis z użyciem frazy OUTER:

```
SELECT P.NumerPrac, Nazwisko, Imię, NazwaZwiązku
FROM Pracownicy P, OUTER Związki Z
WHERE P.NumerPrac=Z.NumerPrac
```

Perspektywy

Zapytanie przekształcić możemy w tablicę wirtualną – tak zwaną perspektywę lub widok (ang. VIEW). Perspektywę tworzy się za pomocą frazy:

```
CREATE VIEW Nazwa [(atrybuty)] AS (zapytanie)
```

Od tego momentu identyfikator Nazwa będzie traktowany w kolejnych instrukcjach języka SQL jak zwykła tablica. Przykładowo instrukcja:

```
CREATE VIEW DanePers (Nazwisko, Imię, Zespół) AS
    (SELECT Nazwisko, Imię, NazwaZesp
    FROM Pracownicy P, Zespoły Z
    WHERE P.NumerZesp=Z.NumerZesp)
```

tworzy perspektywę do której można się odwoływać zapytaniem:

```
SELECT * FROM DanePers WHERE Imię LIKE „Adam”
```

Podanie nazw atrybutów perspektywy jest konieczne tylko w przypadku, gdy w zapytaniu ją tworzącym wykorzystuje się funkcje agregujące.

Na wykorzystywanym podczas laboratorium serwerze SQL Base nie jest możliwe zdefiniowanie perspektywy z zapytania zawierającego frazę ORDER BY.

Usunięcie perspektywy realizuje się instrukcją:

```
DROP VIEW Nazwa
```

Perspektywy mogą służyć do modyfikacji danych w bazie za pomocą instrukcji INSERT lub UPDATE. Aby jednak było to możliwe perspektywa spełniać musi pewne warunki. Dla przykładu na serwerze SQL Base perspektywa modyfikująca bazę nie może:

- odwoływać się do więcej niż jednej tablicy,
- zawierać funkcji agregujących,
- zawierać frazy GROUP BY
- zawierać frazy DISTINCT.

Użycie perspektyw upraszcza dostęp do danych, pozwala także na lepszą kontrolę i ograniczenie uprawnień użytkowników.

SQL – przykłady:

1. Instrukcja ***select*** na 1 tabeli:

   Wypisać nazwiska i płeć wszystkich pracowników zespołu numer 6:

   ```
   Select nazwisko, kobieta
   From pacowni
   Where nrz = 6;
   ```

2. Instrukcja ***select*** na kilku tabelach:

   Wygenerować zestawienie studentów (mężczyzn) kierunku INFORMATYKA w postaci:
   kierunek, numer, nazwisko:

   ```
   Select k.kierunek, s.student, s.nazwisko
   From studenci s, kier_stu k
   Where s.student = k.student and k.kierunek = 'INFORMATYKA' and s.kobieta = 'F';
   ```

3. Instrukcja ***select*** na kilku tabelach oraz funkcja agregująca na wszystkich elementach zapytania:

   Wypisać liczbę pracowników zespołu 'PIECE'

   ```
   Select z.nazwazesp, count(p.nrp)
   From zespol z, pracowni p
   Where z.nrz = p.nrz and nazwazesp='PIECE';
   ```

4. Instrukcja ***select*** na kilku tabelach oraz funkcja agregująca na pewnych grupach rekordów zapytania:

   Wygenerować zestawienie dla każdego zespołu w postaci:
   nazwa zespołu liczba pracowników:

```
Select z.nazwazesp, count(p.nrp)
From zespol z, pracowni p
Where z.nrz = p.nrz
Group by z.nrz, z.nazwazesp;
```

5. Instrukcja ***select*** na kilku tabelach oraz funkcja agregująca na pewnych grupach rekordów zapytania, spełniających podane kryterium:

   W ramach zespołów, wypisać nazwiska pracowników, których sumaryczne dochody w ramach poszczególnych tematów przekraczają 500 zł:

```
Select z.nrz, z.nazwazesp, p.nrp, p.nazwisko, sum (d.kwota)
From zespol z, pracowni p, dochody d
Where z.nrz = p.nrz and p.nrp = d.nrp
Group by z.nrz, z.nazwazesp, p.nrp, p.nazwisko, d.nrt
Having sum(d.kwota) > 500;
```

6. Zagnieżdżona instrukcja ***select*****,** operator ***in***:

   Podać numery i nazwiska pracowników (kobiet), które uzyskały dochody w jednym z tematów, prowadzonych przez pracownika o nazwisku ‘NOWAK’:

```
Select p.nrp, p.nazwisko
From pracowni p, dochody d
Where p.nrp = d.nrp and p.kobieta = ‘T’ and d.nrt in
        (Select t.nrt
        from temat t, pracowni p
        where p.nrp = t.nrpkt and p.nazwisko = ‘NOWAK’);
```

7. Zagnieżdżona instrukcja ***select*****,** operator ***all***:

   Podać numery i nazwiska studentów (mężczyzn) kierunku ‘INFORMATYKA’, którzy są młodsi od każdej studentki kierunku ‘ELEKTRONIKA’.

```
Select s.student, s.nazwisko
From studenci s, kier_stu k
Where s.student = k.student and k.kierunek = 'INFORMATYKA' and s.kobieta = 'F'
and  s.data_ur > all
(select s.data_ur
From studenci s, kier_stu k
Where s.student = k.student and k.kierunek = 'ELEKTRONIKA'
        and s.kobieta = 'T');
```

8. Zagnieżdżona instrukcja ***select*****,** operator ***any***

Podać numery i nazwiska pracowników zespołu 'PIECE', których pojedyncza wypłata jest większa od co najmniej jednej z wypłat pracowników zespołu 'BUDOWA':

```
Select p.nrp, p.nazwisko
From pracowni p, zespol z, dochody d
Where p.nrz = z.nrz and d.nrp = p.nrp and z.nazwazesp = 'PIECE' and d.kwota > any
(select d.kwota
        from pracowni p, zespol z, dochody d
        where p.nrz = z.nrz and d.nrp = p.nrp and z.nazwazesp = 'BUDOWA');
```

9. Zagnieżdżona instrukcja ***select*****,** operator ***exists***:

Podać nazwy i numery tematów, w których nie uzyskała dochodu żadna kobieta:

```
Select t.nazwatemat ,t.nrt
From temat t
Where not exists
        (select *
        from pracowni p, dochody d
        where p.nrp = d.nrp and p.kobieta = 'T' and d.nrt = t.nrt);
```

10. Tworzenie perspektyw - instrukcja ***view***:

Wykonać zestawienie dochodów pracowników w postaci:
Nazwisko, łączna kwota zarobiona przez pracownika

```
Create view zestawienie (nazwisko, sumaryczne_dochody) as
(Select p.nazwisko, sum (d.kwota)
From pracowni p, dochody d
Where p.nrp = d.nrp
Group by p.nrp, p.nazwisko);
```

Usunięcie perspektywy - polecenie:

```
Drop view zestawienie;
```

11. Wykorzystanie ***złączeń zewnętrznych***:

Wykonać zestawienie kierunków z liczbą studentów zapisanych na ten kierunek, uwzględnić kierunki, na które nie zapisał się jeszcze żaden student.

```
Select k.kierunek, count (distinct ks.student)
From kierunki k, kier_stu ks
Where ks.kierunek (+) = k.kierunek
Group by k.kierunek;
```

SCHEMAT BAZY DANYCH "BAZAUNI" – wykorzystany w przykładach

RELACJA	ATRYBUT	TYP
DOCHODY	NRT	INTEGER
	KWOTA	DECIMAL
	NRP	SMALLINT
INST	INSTYTUT	CHAR
	NRZ	SMALLINT
KIER_STU	KIERUNEK	CHAR
	STUDENT	INTEGER
KIERUNKI	INSTYTUT	CHAR
	KIERUNEK	CHAR
OCENY	PRZEDMIOT	CHAR
	STUDENT	INTEGER
	OCENA	SMALLINT
	DATA_ZAL	TIMESTMP
PRACOWNI	NRP	SMALLINT
	KOBIETA	CHAR
	DATA_UR	TIMESTMP
	NAZWISKO	CHAR
	NRZ	SMALLINT
ROZKLAD	SALA	SMALLINT
	GODZINA	SMALLINT
	DZIEN	CHAR
	PRZEDMIOT	CHAR
ROZM_SAL	SALA	SMALLINT
	ROZM_SALI	SMALLINT
	DYDAKTYKA	CHAR
STUDENCI	NAZWISKO	CHAR
	STUDENT	INTEGER
	DATA_UR	TIMESTMP

	KOBIETA	CHAR
TEMAT	NAZWATEMAT	CHAR
	DATA_ODB	TIMESTMP
	NRT	INTEGER
	NRPKT	SMALLINT
WYKLADOW	PRZEDMIOT	CHAR
	NRP	SMALLINT
ZESPOL	NAZWAZESP	CHAR
	NRZ	SMALLINT
	NRPKZ	SMALLINT