



SQL : Манипулација со податоци



Пишување на команди во SQL

- Наредбите во SQL се составени од резервирани или кориснички дефинирани зборови.
 - Резервираните зборови се составен дел од SQL и тие мора да бидат точно напишани како што се бара и не може да се делат со нов ред.
 - Кориснички дефинираните зборови се креирани од страна на корисникот и репрезентираат имиња на различни објекти од базата како што се релации, колони, изведени релации итн.



Пишување на команди во SQL

- Компонентите на една SQL наредба се case sensitive со исклучок на знаковни константи ;
- Се користи напредна форма на Backus Naur Form (BNF) нотацијата:
 - Со големи букви се преставуваат резервирани зборови
 - Со мали букви се преставуваат кориснички дефинирани зборови
 - Знакот | преставува избор помеѓу алтернативи
 - Во големи загради се става бараниот елемент
 - Во квадратни загради се става опционален елемент.
 - (...) претставува опционално повторување

3



Знаковни константи

- Константи кои се користат во SQL наредбите.
- Сите знаковни константи кои не се броеви мора да се ставени во единечни наводници.
(пр. 'London')
- Сите знаковни константи кои не се броеви не се ставаат во наводници.
(пр. 650.00)

4



Наредба SELECT

- Да се прочитаат и прикажат податоците од една или повеќе табели од базата на податоци :

```
SELECT [DISTINCT | ALL]
      { * | [columnExpression [AS newName]] [,...] }
FROM      TableName [alias] [, ...]
[WHERE condition]
[GROUP BY columnList] [HAVING condition]
[ORDER BY columnList]
```

5



Наредба SELECT

- FROM табелите кои се користат
- WHERE филтрира редови
- GROUP BY формира група на редови со иста вредност во колоната
- HAVING филтрира група на објекти со некој зададен услов
- SELECT дефинира кои колони ќе ги добиеме на излез
- ORDER BY дефинира распоред на излезот

6



Наредба SELECT

- Распоредот на клаузите не може да биде променет.
- Само SELECT и FROM се задолжителни.
- SELECT операцијата е затворена: резултатот од прашањето врз некоја табела е друга табела.

7



Пример 5.1 Сите колони, сите редици


- Детална листа на податоци за целиот персонал :

```
SELECT staffNo, fName, lName, address,  
       position, sex, DOB, salary, branchNo  
FROM Staff;
```

Може да се користи * како замена за сите колони

```
SELECT *  
FROM Staff;
```

8



Пример 5.1

Сите колони, сите редови

Table 5.1 Result table for Example 5.1.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005



Пример 5.2

Одредени колони, сите редици

- Листа на плата на целиот персонал, прикажувајќи само шифра на вработен, име и презиме и плата :

```
SELECT staffNo, fName, lName, salary  
FROM Staff;
```



Пример 5.2

Одредени колони, сите редици

Table 5.2 Result table for Example 5.2.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

11



Пример 5.3

Употреба на DISTINCT

- Листа на сите шифри на имоти кои биле разгледани:

```
SELECT propertyNo
FROM Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

12



Пример 5.3

Употреба на DISTINCT

- Со користење на DISTINCT ги елиминираме дупликатите во листата :

```
SELECT DISTINCT propertyNo  
FROM Viewing;
```

propertyNo
PA14
PG4
PG36

13




Пример 5.4

Резултатни полиња

- Листа на месечна плата за целиот персонал, прикажувајќи шифра на вработен, име и презиме и плата :

```
SELECT staffNo, fName, lName, salary/12  
FROM Staff;
```

14




Пример 5.4

Резултатни полиња

Table 5.4 Result table for Example 5.4.

staffNo	fName	lName	col4
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

15



Пример 5.4

Резултатни полиња

- За именување на колона може да се користи AS клаузата :

```
SELECT staffNo, fName, lName, salary/12
      AS monthlySalary
FROM Staff;
```

16



СЕЛЕКЦИЈА НА РЕДОВИ (WHERE)

- Предикати што се користат со WHERE:
 - За споредба
 - За опсег
 - За членство во множество
 - За совпаѓање со примерок (pattern match)
 - За тестирање на непознати вредности (Null)

17



Пример 5.5 Пребарување со услов за споредување

- Листа на сите вработени со плата поголема од 10.000:

```
SELECT staffNo, fName, lName, position, salary  
FROM Staff  
WHERE salary > 10000;
```

Table 5.5 Result table for Example 5.5.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00


18



Оператори за споредба

- Во SQL, се користат следниве едноставни оператори за споредба :
 - = еднакво
 - < > нееднакво (ISO standard)
 - < помало од
 - > поголемо од
 - != нееднакво
 - <= помало или еднакво
 - >= поголемо или еднакво

19



Пример 5.6 Сложено пребарување со услов за споредување

- Листа на адресите на сите преставништва во Лондон и Глазгов :

```
SELECT *  
FROM Branch  
WHERE city = 'London' OR city = 'Glasgow';
```

Table 5.6 Result table for Example 5.6.

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

20



Пример 5.7 Пребарување во опсег

- Листа на вработени со плата помеѓу 20.000 и 30.000 :

```
SELECT staffNo, fName, lName, position, salary  
FROM Staff  
WHERE salary BETWEEN 20000 AND 30000;
```

- Тестот BETWEEN ги дефинира крајните точки на опсегот.

21



Пример 5.7 Пребарување во опсег

Table 5.7 Result table for Example 5.7.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

22



Пример 5.7

Пребарување во опсег

- Постои и негација NOT BETWEEN
- BETWEEN може и да се замени на следниов начин :

```
SELECT staffNo, fName, lName, position, salary  
FROM Staff  
WHERE salary >= 20000 AND salary <= 30000;
```

- Сепак, тестот BETWEEN е поедноставен начин да се прикаже одреден опсег.

23



Пример 5.8

Пребарување со дадено множество на вредности

- Постои негација (NOT IN).
- Изразот IN може да се замени на следниов начин :

```
SELECT staffNo, fName, lName, position  
FROM Staff  
WHERE position = 'Manager' OR position = 'Supervisor';
```

- Изразот IN се користи кога множеството има повеќе вредности.
- IN – тестира дали одредена вредност се совпаѓа со некоја вредност од даденото множество.

24

Пример 5.8

Пребарување со дадено множество на вредности

- Листа на сите менаџери и супервизори :

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position IN ('Manager', 'Supervisor');
```

Table 5.8 Result table for Example 5.8.

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

25

Пример 5.9

Пребарување со споредба на даден стринг

- SQL има два специјални знаци за споредба со стринг
 - % претставува секвенца од нули или повеќе карактери (wildcard)
 - _ (долна црта) претставува било кој единечен карактер.

26



Пример 5.9

Пребарување со споредба на даден стринг

- Најди ги сите сопственици со стрингот 'Glasgow' во нивната адреса :

```
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE '%Glasgow%';
```

Table 5.9 Result table for Example 5.9.

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

27



Пример 5.10

Пребарување со споредба со NULL

- Детална листа на сите посети на имотот PG4 каде што не е внесено коментар :

```
SELECT clientNo, viewDate
FROM Viewing
WHERE propertyNo = 'PG4' AND
      comment IS NULL;
```

28



Пример 5.10

Пребарување со споредба со NULL

clientNo	viewDate
CR56	26-May-04

- Негацијата (IS NOT NULL) е тест за non-null вредности.

29



Пример 5.11

Сортирање на колона

- Листа на плати на целиот персонал, сортирани во опаѓачки редослед според плата :

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```

30



Пример 5.11 Сортирање на колона

Table 5.11 Result table for Example 5.11.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

31



Пример 5.12 Сортирање на повеќе колони

- Скратена листа на имоти сортирани по тип на имот (една колона):

```
SELECT propertyNo, type, rooms, rent  
FROM PropertyForRent  
ORDER BY type;
```

32



Пример 5.12

Сортирање на повеќе колони

Table 5.12(a) Result table for Example 5.12 with one sort key.

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

33




Пример 5.12

Сортирање на повеќе колони

- Сортирање по тип на имот по азбучен редослед и опаѓачки редослед на колоната наемнина (по две колони):

```
SELECT propertyNo, type, rooms, rent
FROM PropertyForRent
ORDER BY type, rent DESC;
```

34



Пример 5.12

Сортирање на повеќе колони

Table 5.12(b) Result table for Example 5.12 with two sort keys.

propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

35



Наредба SELECT - Агрегати

- ISO стандардот дефинира пет агрегати :

COUNT го враќа бројот на вредности во дадена колона

SUM ја враќа сумата на вредности во дадена колона

AVG ја враќа средната вредност од вредностите во дадена колона

MIN ја враќа минималната вредност во дадена колона

MAX ја враќа максималната вредност во дадена колона

36



Наредба SELECT - Агрегати

- Секој агрегат делува на една колона од табелата и враќа една вредност.
- COUNT, MIN, and MAX може да се користат кај нумерички и ненумерички полиња, но SUM и AVG може да се користат само кај нумерички полиња.
- Освен COUNT(*), секоја функција прво ги елиминира null вредностите и оперира со останатите non-null вредности.

37



Наредба SELECT - Агрегати

- COUNT(*) ги брои сите редови во табелата, без разлика дали имаме дупликат или полиња со null вредност.
- Може да се користи DISTINCT пред името на колоната за да се елиминираат дупликати.
- DISTINCT нема ефект кај агрегатите MIN и MAX но може да се користи кај SUM и AVG.

38



Наредба SELECT - Агрегати

- Агрегатните функции може да се користат само кај SELECT наредбата како и во HAVING клауза.
- Ако SELECT наредбата содржи агрегатна функција но нема GROUP BY клауза, SELECT наредбата не може да референцира колона која не е аргумент на агрегатната функција. Следниов пример е неточен :

```
SELECT staffNo, COUNT(salary)
FROM Staff;
```

39



Пример 5.13 Употреба на COUNT(*)

- Колку недвижности чинат повеќе од £350 месечна наемнина :

```
SELECT COUNT(*) AS myCount
FROM PropertyForRent
WHERE rent > 350;
```

myCount
5

40



Пример 5.14

Употреба на COUNT(DISTINCT)

- Колку недвижности се разгледани во месец мај 2004?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount
FROM Viewing
WHERE viewDate BETWEEN '1-May-04'
AND '31-May-04';
```

myCount
2

41



Пример 5.15

Употреба на COUNT и SUM

- Најди го бројот на менаџерите и сумата на нивните плати :

```
SELECT COUNT(staffNo) AS myCount,
       SUM(salary) AS mySum
FROM Staff
WHERE position = 'Manager';
```

myCount	mySum
2	54000.00

2



Употреба на MIN, MAX и AVG

- Најди ја минималната, максималната и средната плата на персоналот :

```
SELECT MIN(salary) AS myMin,  
       MAX(salary) AS myMax,  
       AVG(salary) AS myAvg  
FROM Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

43



Наредба SELECT - Групирање

- GROUP BY клаузата се користи за добивање подгрупи.
- SELECT and GROUP BY се блиску поврзани: секој објект во SELECT наредбата мора да има единечна вредност по група и SELECT клаузата може да содржи само:
 - имиња на колони
 - агрегатни функции
 - константи
 - изрази кои се комбинации од горенаведеното.

44



Наредба SELECT - Групирање

- Сите имиња на колони во наредбата SELECT мора да ги има во GROUP BY клаузата освен ако името не се користи во агрегатна функција.
- Ако се користи WHERE со GROUP BY, прво се извршува WHERE а потоа се формираат групите од останатите редови кои го задоволуваат предикатот.
- ISO стандардот дефинира две null вредности да бидат еднакви за намената на GROUP BY.

45



Пример 5.17 Употреба на GROUP BY

- Најди го бројот на персоналот и нивната вкупна плата во секое преставништво :

```
SELECT      branchNo,  
            COUNT(staffNo) AS myCount,  
            SUM(salary) AS mySum  
FROM Staff  
GROUP BY branchNo  
ORDER BY branchNo;  
Секоја колона (што не е во агрегатна ф-ја во  
SELECT) мора да ја има и во GROUP BY.
```

46



Пример 5.17

Употреба на GROUP BY

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

47



Ограничени Групирања - HAVING клауза

- HAVING е дизајнирана да се користи во комбинација со GROUP BY со цел да се ограничат групите кои се појавуваат во крајната резултатна табела.
- Слично е со WHERE, но WHERE филтрира посебни редови додека HAVING филтрира групи.
- Имињата на колоните во HAVING клаузата мора да ги има и во GROUP BY или во некоја агрегатна функција.

48



Пример 5.18 Употреба на HAVING

- За секое претставништво со повеќе од еден член на персонал, најди го бројот на персонал во секое претставништво и сумата на нивните плати :

```
SELECT branchNo,  
       COUNT(staffNo) AS myCount,  
       SUM(salary) AS mySum  
FROM Staff  
GROUP BY branchNo  
HAVING COUNT(staffNo) > 1  
ORDER BY branchNo;
```

49



Пример 5.18 Употреба на HAVING

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

50



Подпрашања (Subqueries)

- Некои SQL наредби содржат вгнезден SELECT во нив.
- Вгнездениот селект може да се користи во WHERE и HAVING клаузата на надворешен SELECT каде претставува подпрашање или вгнездено прашање.
- Вгнездениот селект може да се користи и кај INSERT, UPDATE, и DELETE наредбите.

51



Пример 5.19 Подпрашање со еднаквост

- Листа на целиот персонал што работи во претставништвото на улицата '163 Main St' :

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
    (SELECT branchNo
     FROM Branch
     WHERE street = '163 Main St');
```

52



Пример 5.19

Подпрашање со еднаквост

- Внатрешниот SELECT ја наоѓа шифрата на претставништвото на улицата '163 Main St' која е ('B003').
- Надворешниот SELECT потоа ги чита деталите за целиот персонал кој работи во претставништвото
- Надворешниот SELECT тогаш станува :

```
SELECT staffNo, fName, lName, position  
FROM Staff  
WHERE branchNo = 'B003';
```

53



Пример 5.19

Подпрашање со еднаквост

Table 5.19 Result table for Example 5.19.

staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

54



Пример 5.20

Прашање со агрегат

- Листа на целиот персонал чија плата е поголема од просечната плата и прикажано за колку :

```
SELECT staffNo, fName, lName, position,  
       salary – (SELECT AVG(salary) FROM Staff) As SalDiff  
FROM Staff  
WHERE salary >  
       (SELECT AVG(salary)  
        FROM Staff);
```

55



Пример 5.20

Прашање со агрегат

- Не може да се напише 'WHERE salary > AVG(salary)' зошто агрегатните ф-ии не може да бидат во WHERE
- Наместо тоа, користиме подпрашање да ја најдеме просечната плата (17000), и потоа со надворешниот SELECT ги наоѓаме вработените со поголема плата од просечната :

```
SELECT staffNo, fName, lName, position,  
       salary – 17000 As salDiff  
FROM Staff  
WHERE salary > 17000;
```

56



Пример 5.20 Прашање со агрегат

Table 5.20 Result table for Example 5.20.

staffNo	fName	lName	position	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00

57



Правила за подпрашања

- ORDER BY клаузата не може да се користи во подпрашање.
- SELECT листата на подпрашањето мора да содржи колони со единствено име или израз, со исклучок на подпрашања кои користат EXISTS.
- Имињата на колоните се однесуваат на името на табелата во FROM клаузата на подпрашањето. Може да се однесуваат на табела во FROM преку некој алиас.

58



Правила за подпрашања

- Кога подпрашање е еден од операндите инволвирани во споредба, подпрашањето мора да фигурира од десната страна од споредбата.
- Подпрашање не може да се користи како операнд во израз.

59



Правила за подпрашања

- SELECT staffNo,
- FROM Staff
- WHERE branchNo = (SELECT branchNo
FROM Branch
WHERE street= '163
Main st')

Резултат: Внатрешното прашање го наоѓа branchNo= B003, а потоа надв. ги наоѓа вработените во тој branch.

60

Пример 5.21 Употреба на IN

- Листа на недвижности менаџирани од персоналот на улица '163 Main St' : IN се користи наместо = зошто може да има повеќе недвижности придружени за секој вработен во тој branch.

```
SELECT propertyNo, street, city, postcode, type, rooms, rent
FROM PropertyForRent
WHERE staffNo IN
  (SELECT staffNo
   FROM Staff
   WHERE branchNo =
     (SELECT branchNo
      FROM Branch
      WHERE street = '163 Main St'));
```

61

Пример 5.21 Употреба на IN

Table 5.21 Result table for Example 5.21.

propertyNo	street	city	postcode	type	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375
PG21	18 Dale Rd	Glasgow	G12	House	5	600

62



ANY и ALL

- ANY и ALL може да се користат кај подпрашања кои продуцираат единечна колона на броеви.
- Со ALL, условот ќе биде вистинит ако е задоволен за сите вредности продуцирани од подпрашањето.
- Со ANY, условот ќе биде вистинит ако е задоволен за било која вредност продуцирана од подпрашањето.
- Ако подпрашањето е празно, ALL враќа TRUE додека ANY враќа FALSE .
- SOME може да се користи наместо ANY.

63



Пример 5.22 Употреба на ANY/SOME

- Најди го целиот персонал чија плата е поголема од платата на барем еден член на персоналот во претставништвото со шифра B003 :

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > SOME
  (SELECT salary
   FROM Staff
   WHERE branchNo = 'B003');
```

64



Пример 5.22 Употреба на ANY/SOME

- Внатрешното прашање продуцира множество {12000, 18000, 24000} и надворешното прашање го селектира персоналот чија плата е поголема од било која вредност во множеството.

Table 5.22 Result table for Example 5.22.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

65



Пример 5.23 Употреба на ALL

- Најди го целиот персонал чија плата е поголема од платата на секој член на персоналот во претставништвото со шифра B003 :

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > ALL
      (SELECT salary
       FROM Staff
       WHERE branchNo = 'B003');
```

66



Пример 5.23 Употреба на ALL

Table 5.23 Result table for Example 5.23.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00

67



Прашања со повеќе табели (Multi-Table Queries)

- Во претходните примери постоеше битно ограничување: колоните во резултантната табела од некое прашање припаѓаат на една табела.
- Ако резултатите се продуцираат од повеќе табели мора да се користи JOIN.
- За да се направи JOIN, вклучуваме повеќе од една табела во FROM клаузата.
- Како сепаратор се користи запирка како и WHERE клауза која ги специфицира JOIN колоните.

68



Прашања со повеќе табели (Multi-Table Queries)

- Можно е да се користи алиас за табела во FROM клаузата.
- Алиасот е одвоен од името на табелата со празно место.
- Алиасот може да се користи да означува имиња на колони кои се недвосмислени.

69



Пример 5.24 Едноставен JOIN

- Листа на имињата на сите клиенти кои посетиле недвижност заедно со коментарот :

```
SELECT c.clientNo, fName, lName,  
       propertyNo, comment  
FROM Client c, Viewing v  
WHERE c.clientNo = v.clientNo;
```

70



Пример 5.24 Едноставен JOIN

- Само оние редови од двете табели кои имаат исти вредности во clientNo колоната (c.clientNo = v.clientNo).
- Исто како EQUI-JOIN во релациона алгебра.

Table 5.24 Result table for Example 5.24.

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

71



Алтернативни JOIN конструкции

- SQL овозможува алтернативен начин за дефинирање на JOIN:

```
FROM Client c JOIN Viewing v ON c.clientNo = v.clientNo
FROM Client JOIN Viewing USING clientNo
FROM Client NATURAL JOIN Viewing
```

- Во секој од трите случаи, FROM ги заменува оригиналниот FROM и WHERE. Но разликата е тоа што првиот продуцира табела со две идентични колони clientNo.

72



Пример 5.25 Сортирање на JOIN

- За секое претставништво, излистај ги шифрите и имињата на персоналот кој менаџираат со недвижностите како и недвижностите кои ги менаџираат :

```
SELECT s.branchNo, s.staffNo, fName, lName,
       propertyNo
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo
ORDER BY s.branchNo, s.staffNo, propertyNo;
```

73



Пример 5.25 Сортирање на JOIN

Table 5.25 Result table for Example 5.25.

branchNo	staffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14

74

Пример 5.26 JOIN на три табели

- За секое претставништво, излистај го персоналот кој менаџира со недвижности, вклучувајќи го градот каде што се наоѓа претставништвото и недвижностите кои ги менаџираат :

```
SELECT b.branchNo, b.city, s.staffNo, fName,
       lName, propertyNo
FROM Branch b, Staff s, PropertyForRent p
WHERE b.branchNo = s.branchNo AND
       s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

75

Пример 5.26 JOIN на три табели

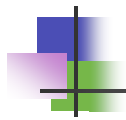
Table 5.26 Result table for Example 5.26.

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

- Алтернативна формулација на FROM и WHERE:

```
FROM (Branch b JOIN Staff s USING branchNo) AS
     bs JOIN PropertyForRent p USING staffNo
```

76



Пример 5.27

Повеќекратно групирање на колони

- Број на недвижности менаџирани од секој член на персоналот :

```
SELECT s.branchNo, s.staffNo, COUNT(*) AS myCount
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo
GROUP BY s.branchNo, s.staffNo
ORDER BY s.branchNo, s.staffNo;
```

77



Пример 5.27

Повеќекратно групирање на колони

branchNo	staffNo	myCount
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

78



Надворешен JOIN

- Пр.: 2 табели со по 2 колони и 3 реда
- Branch1 (branchNo, bCity)
- PropertyForRent1 (propertyNo, pCity)

branchNo	bcity
B003	Glasgow
B004	Bristol
B002	London

propertyNo	pcity
PA14	Aberdeen
PG4	Glasgow
PL94	London

79



Пример 5.28 Надворешен JOIN од лево

- Листа на претставништвата и недвижностите кои се во ист град заедно со било кое претставништво кое не се совпаѓа :

```
SELECT b.*, p.*  
FROM Branch1 b LEFT JOIN  
PropertyForRent1 p ON b.bCity = p.pCity;
```

80



Пример 5.28

Надворешен JOIN од лево

- Ги вклучува оние редови првата (левата) табела кои не се совпаѓаат со редови од втората (десната) табела.
- Колоните од втората табела се полнат со NULLs.

Table 5.28 Result table for Example 5.28.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

81



Пример 5.29

Надворешен JOIN од десно

- Листа на преставништвата и недвижностите кои се во ист град и било кои недвижности кои не се совпаѓаат :

```
SELECT b.*, p.*  
FROM Branch1 b RIGHT JOIN  
PropertyForRent1 p ON b.bCity = p.pCity;
```

82

Пример 5.29 Надворешен JOIN од десно

- Ги вклучува оние редови втората(десната) табела кои не се совпаѓаат со редови од првата(левата) табела.
- Колоните од првата табела се полнат со NULLs.

Table 5.29 Result table for Example 5.29.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London


83

Пример 5.30 Надворешен целосен JOIN

- Листа на претставништва и недвижности во ист град заедно со претставништва и недвижности кои не се совпаѓаат :

```
SELECT b.*, p.*
FROM Branch1 b FULL JOIN
PropertyForRent1 p ON b.bCity = p.pCity;
```

84



Пример 5.30

Надворешен целосен JOIN

- Вклучени се редовите кои не се совпаѓаат во двете табели.
- Колоните кои не се совпаѓаат се полнат со NULLs.

Table 5.30 Result table for Example 5.30.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

85



EXISTS и NOT EXISTS

- EXISTS и NOT EXISTS се користат само за подпрашања(subqueries).
- Продуцираат едноставен true/false резултат .
- TRUE ако и само ако постои барем еден ред во резултатската табела.
- FALSE ако подпрашањето врати празна табела.
- NOT EXISTS е спротивност на EXISTS.

86



EXISTS и NOT EXISTS

- (NOT) EXISTS го проверува постоењето или непостоењето на редови во резултатската табела од подпрашањето така што подпрашањето може да содржи произволен број на колони.
- Обично подпрашањата кои содржат (NOT) EXISTS се во форма:
(SELECT * ...)

87



Пример 5.31 Подпрашања со EXISTS

- Најди го целиот персонал кој работи во претставништвото во Лондон :

```
SELECT staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS
  (SELECT *
   FROM Branch b
   WHERE s.branchNo = b.branchNo AND
         city = 'London');
```

88



Пример 5.31 Подпрашања со EXISTS

Table 5.31 Result table for Example 5.31.

staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant

89



Пример 5.31 Подпрашања со EXISTS

- Ова подпрашање може да се напише и со JOIN конструкција :

```
SELECT staffNo, fName, lName, position
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo AND
      city = 'London';
```

90



Union, Intersect, и Difference (Except)

- Може да се користи множество од операции составени од Union, Intersection, и Difference за да се комбинираат резултатите од две или повеќе прашања во единечна резултатска табела.
- Union на две табели, A и B, е табела која ги содржи сите редови од или A или B или двете табели заедно.
- Intersection е табела која ги содржи сите заеднички редови во A и B.
- Difference е табела која ги содржи сите редови од A но не и од B.
- Двете табели мора да бидат union компатибилни.

91

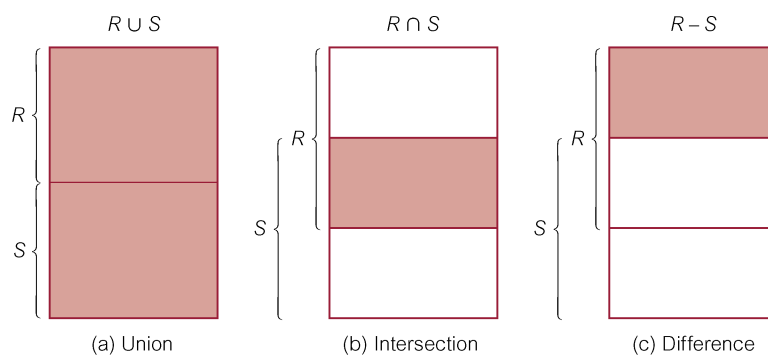


Union, Intersect, и Difference (Except)

- Форматот на множество на операции клаузата во секој случај изгледа вака :
`оп [ALL] [CORRESPONDING [BY {column1 [, ...]}]]`
- Ако CORRESPONDING BY е специфицирано, тогаш множеството на операции се извршува на именуваните колони.
- Ако CORRESPONDING е специфицирано но без BY клаузата, операциите се извршуваат на заедничките колони.
- Ако ALL е специфицирано, резултатот може да содржи дупликат редови.

92

Union, Intersect, и Difference (Except)



93

Пример 5.32 Употреба на UNION

- Листа на сите градови каде што постои претставништво или недвижност :

```
(SELECT city
FROM Branch
WHERE city IS NOT NULL)
UNION
(SELECT city
FROM PropertyForRent
WHERE city IS NOT NULL);
```

94



Пример 5.32 Употреба на UNION

- или :

```
(SELECT *  
FROM Branch  
WHERE city IS NOT NULL)  
UNION CORRESPONDING BY city  
(SELECT *  
FROM PropertyForRent  
WHERE city IS NOT NULL);
```

95



Пример 5.32 Употреба на UNION

- Резултатските табели се продуцира од двете прашања и се соединуваат во една.

Table 5.32 Result table for Example 5.32.

city
London
Glasgow
Aberdeen
Bristol

96



Пример 5.33 Употреба на INTERSECT

- Листа на сите градови каде има претставништво и недвижност :

```
(SELECT city FROM Branch)
INTERSECT
(SELECT city FROM PropertyForRent);
```

97



Пример 5.33 Употреба на INTERSECT

- или :

```
(SELECT * FROM Branch)
INTERSECT CORRESPONDING BY city
(SELECT * FROM PropertyForRent);
```

Table 5.33 Result table for Example 5.33.

city
Aberdeen
Glasgow
London

98



Пример 5.33 Употреба на INTERSECT

- Ова прашање може да се напише и без INTERSECT :

```
SELECT b.city  
FROM Branch b, PropertyForRent p  
WHERE b.city = p.city;
```

- или :

```
SELECT DISTINCT city FROM Branch b  
WHERE EXISTS  
(SELECT * FROM PropertyForRent p  
WHERE p.city = b.city);
```

99



Пример 5.34 Употреба на EXCEPT

- Листа на сите градови каде има претставништво но нема недвижност :

```
(SELECT city FROM Branch)  
EXCEPT  
(SELECT city FROM PropertyForRent);
```

- или :

```
(SELECT * FROM Branch)  
EXCEPT CORRESPONDING BY city  
(SELECT * FROM PropertyForRent);
```

Table 5.34 Result table for Example 5.34.

city
Bristol

100



Пример 5.34 Употреба на EXCEPT

- Ова прашање може да се напише и без EXCEPT :

```
SELECT DISTINCT city FROM Branch  
WHERE city NOT IN  
(SELECT city FROM PropertyForRent);
```

- или :

```
SELECT DISTINCT city FROM Branch b  
WHERE NOT EXISTS  
(SELECT * FROM PropertyForRent p  
WHERE p.city = b.city);
```

101



Запишување (INSERT)

```
INSERT INTO TableName [ (columnList) ]  
VALUES (dataValueList)
```

- *columnList* е опционално; ако не се користи тогаш SQL користи листа од сите колони во нивниот оригинален CREATE TABLE распоред.
- Секоја изоставена колона мора да биде декларирана како NULL кога се креира табелата освен ако е специфицирано DEFAULT кога се креира колоната.

102



Запишување (INSERT)

- *dataValueList* мора да се совпаѓа со *columnList* и тоа:
 - Бројот на објектите во секоја листа мора да е ист;
 - мора да постои директна коренсподенција помеѓу објектите во двете листи;
 - Типот на податоци во *dataValueList* мора да биде компатибилен со типот на податоци од коренспонтирачката колона.

103



Пример 5.35 INSERT ... VALUES

- Внеси нов ред во табелата Staff пополнувајќи ги сите колони :

```
INSERT INTO Staff  
VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M',  
Date'1957-05-25', 8300, 'B003');
```

104



Пример 5.36 INSERT using Defaults

- Внеси нов ред во табелата Staff пополнувајќи ги сите задолжителни колони :

```
INSERT INTO Staff (staffNo, fName, lName,  
                  position, salary, branchNo)  
VALUES ('SG44', 'Anne', 'Jones',  
        'Assistant', 8100, 'B003');
```

- Или :

```
INSERT INTO Staff  
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL,  
        NULL, 8100, 'B003');
```

105



INSERT ... SELECT

- Втората форма на INSERT овозможува повеќе редови да бидат копирани од една или повеќе табели во друга табела :

```
INSERT INTO TableName [ (columnList) ]  
SELECT ...
```

106



Пример 5.37 INSERT ... SELECT

- Да претпоставиме дека постои табела StaffPropCount која ги содржи имињата на персоналот и бројот на недвижности кој го менаџираат :

StaffPropCount(staffNo, fName, lName, propCnt)

Табелата StaffPropCount настанува од табелите Staff и PropertyForRent табелите.

107



Пример 5.37 INSERT ... SELECT

```
INSERT INTO StaffPropCount
  (SELECT s.staffNo, fName, lName, COUNT(*)
   FROM Staff s, PropertyForRent p
   WHERE s.staffNo = p.staffNo
   GROUP BY s.staffNo, fName, lName)
UNION
  (SELECT staffNo, fName, lName, 0
   FROM Staff
   WHERE staffNo NOT IN
     (SELECT DISTINCT staffNo
      FROM PropertyForRent));
```

108



Пример 5.37 INSERT ... SELECT

Table 5.35 Result table for Example 5.37.

staffNo	fName	lName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

- Ако вториот дел на UNION е изоставен, персоналот што моментално не менаџира недвижности е изоставен.

109



Ажурирање (UPDATE)

```
UPDATE TableName  
SET columnName1 = dataValue1  
  [, columnName2 = dataValue2...]  
[WHERE searchCondition]
```

- TableName е името на базната табела.
- SET клаузата ги специфицира имињата на една или повеќе колони кои треба да бидат ажурирани.


110



Ажурирање (UPDATE)

- WHERE клаузата е опционална:
 - Ако не постои, именуваните колони се ажурираат за сите редови во табелата;
 - Ако постои, само оние редови кои го задоволуваат пребарувачкиот услов (*searchCondition*) се ажурираат.
- Новите вредности (*dataValue(s)*) мора да бидат компатибилни со типот на податоци во кореспондирачката колона.

111



Пример 5.38/39 Ажурирање на сите редови

- Зголемување на платата на персоналот за 3%.

```
UPDATE Staff  
SET salary = salary*1.03;
```
- Зголемување на платата на менаџерите за 5%.

```
UPDATE Staff  
SET salary = salary*1.05  
WHERE position = 'Manager';
```

112



Пример 5.40

Ажурирање на повеќе колони

- Унапреди го David Ford (staffNo='SG14') во менаџер и зголеми ја неговата плата на 18000 :

```
UPDATE Staff  
SET position = 'Manager', salary = 18000  
WHERE staffNo = 'SG14';
```

113




Бришење (DELETE)

```
DELETE FROM TableName  
[WHERE searchCondition]
```

- TableName е името на базната табела.
- *searchCondition* е опционално; ако го нема, сите редови се бришат од табелата. Но ова не ја брише самата табела. Ако *search_condition* постои, само оние редови кои го задоволуваат условот кје бидат избришани.

114



Пример 5.41/42

Бришење на специфични редови

- Избриши ги сите посети на недвижноста PG4 :

```
DELETE FROM Viewing  
WHERE propertyNo = 'PG4';
```

- Избриши ги сите рекорди од Viewing табелата :

```
DELETE FROM Viewing;
```

115



Assertions

- Assertion- boolean valued SQL expression – must be true all the times

- **CREATE ASSERTATION** <ass. name>
CHECK (<condition>)

Ex.: ASS guarantying Rich presidents

```
CREATE ASSERTATION Rich Pres CHECK
```

```
(NOT EXISTS
```

```
(SELECT Studio.name
```

```
FROM Studio, MovieExec
```

```
WHERE presC# = cert# AND netWorth < 1000000
```

```
)
```

```
);
```

```
MovieExec(name, add, cert#, networth); Studio(name, add, presC#)
```



TRIGGERS

- Event-Condition_Action (ECA) model
- Ex.: MovieExec (name, address, cert#, netWorth)
- The relation is triggered by updates to the netWorth attribute
- The effect is to avoid lower netWorth of a movie executive



TRIGGERS

- CREATE TRIGGER NetWorthTrigger
- AFTER UPDATE OF netWorth ON Movie Exec
- REFERENCING
 - OLD ROW AS OldTuple,
 - NEW ROW AS NewTuple
- FOR EACH ROW
- WHEN (OldTuple.netWorth > NewTuple.netWorth)
 - UPDATE MovieExec
 - SET netWorth = OldTuple.netWorth
 - WHERE cert# = NewTuple.cert#;

Aktivatori (Triggers)

```
CREATE TRIGGER <ime na
aktivator>
(AFTER | BEFORE) <nastani> ON
<tabela>
[FOR EACH ROW]
[WHEN <uslov>]
<akcii>;

<nastani>::=<nastan> {OR <nastan>}
<nastan>::=INSERT|DELETE|UPDATE [OF <kolona> {,
<kolona>}]
<akcii>::=<PL/SQL block>
```

119

Aktivatori (Triggers)

- Пример: Да се напише активатор кој ќе ја споредува платата на вработениот со платата на неговиот шеф во текот на операциите внесување и промена
- ```
CREATE TRIGGER INFORM_SUPERVISOR
BEFORE INSERT OR UPDATE OF
 SALARY, SUPERVISOR_SSN ON EMPLOYEE
FOR EACH ROW
WHEN
 (NEW.SALARY > (SELECT SALARY
 FROM EMPLOYEE
 WHERE SSN=NEW.SUPERVISOR_SSN)
)
INFORM_SUPERVISOR (NEW.SUPERVISOR_SSN,
NEW.SSN) ;
```

120



## Pogledi (Views)

```
CREATE VIEW <ime> [(<kolona>{,<kolona>})]
AS <SELECT izraz>
```

- име на табелата (погледот)
- можна листа на имиња на атрибути на пример, кога имаме аритметички операции или кога сакаме да ги промениме имињата на атрибутите од базните релации од кои ќе се влече погледот
- прашалник преку кој ќе се специфицира содржината на табелата (погледот)

121



## Pogledi (Views)

- Пример: Специфицирај нова табела WORKS\_ON која ќе ја има истата содржина со старата WORKS\_ON табела, само што наместо SSN на вработениот ќе се чуваат неговите име и презиме

```
CREATE VIEW WORKS_ON_NEW AS
SELECT FNAME, LNAME, PNAME, HOURS
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE SSN=ESSN AND PNO=PNUMBER;
```

122

## ORACLE PL/SQL

- PL/SQL is Oracle procedural extension to SQL
- PL/SQL block has up to three parts:
  - Optional declaration part (variables, constants, cursors and exceptions are defined)
  - Mandatory executable part (variables are manipulated)
  - Optional exception part (for handling any exceptions raised during the execution)

### General structure of PL/SQL block

|                            |           |
|----------------------------|-----------|
| [DECLARE                   | Optional  |
| .....declarations]         |           |
| BEGIN                      | Mandatory |
| .....executable statements |           |
| [EXCEPTION                 | Optional  |
| ....exception handlers]    |           |
| END;                       | Mandatory |



## DECLARATIONS

- vStaffNo **VARCHAR2**(5);
- vRent **NUMBER**(6,2)**NOT NULL:=600;**
- MAX\_PRO **CONSTANT NUMBER:=10;**
  
- Declare a variable to be the same type as in column in the table:
- vStaffNo Staff.staffNo%**Type;**

125



## Executable part of PL/SQL block

### Assignments to variables

- Normal statement by using :=
- As a result of SQL SELECT or FETCH stat.
- Example: set in variable x the No. of properties managed by stafNo SG14

vStaffNo:='SG14';

vRent:=500;

**SELECT COUNT (\*) INTO x FROM**  
PreopertyForRent**WHERE**staffNo= vStaffNo;

126



## Control statements

---

- IF-THEN-ELSE-END-IF;
- LOOP-EXIT WHEN-END LOOP;
- FOR-END LOOP;
- WHILE-END LOOP;
- GO TO;

127



## EXCEPTIONS


---

- An **exception** is an identifier in PL/SQL raised during the execution of a block which terminates its main body of actions
- To handle raised exceptions, separate routines called **exception handlers** are specified

128




## Ex. For exception handling in PL/SQL



```
DECLARE
vpCOUNT NUMBER
vStaffNo PropertyForRent.staffNo%TYPE:='SG14'
...define an exception to prevent a member of staff
...managing more than 100 properties
 e_too_many_properties EXCEPTION;
PRAGMA EXCEPTION _INIT(e_too_many_props,-2000);
BEGIN
 SELECT COUNT(*)INTO vpCount
 from PropertyForRent
 WHERE staffNo =vStaffNo;
 IF vpCount=100
 ...raise an exception to prevent a member of staff
```

129

## Ex. For exception handling in PL/SQL



```
RAISE e_too_many_properties;
 End IF;
UPDATE ProppertyForRent SET
staffNo=vStaffNo WHERE propertyNo='PG4'
EXCEPTION
...handle the exception.....
WHEN e_too_many_properties THEN
dbms_output.put_line 'Member of staff'
||staffNo|| 'already managing 100 properties')
END;
```

130



## CURSORS

---

- SELECT returns normally one row
- To handle query to return an arbitrary number of rows (zero, one or more rows), PL/SQL uses **cursors**
- **Cursors** allow to access one by one row of the query result
- Cursor can be advance by one to access the next row

131



## CURSORS

---

- Cursor must be declared and opened before it can be used, and it must be closed to deactivate it
- The row pointed by the cursor can be retrieved by FETCH statement
- Ex: Use the cursor to determine the properties managed by SG14 (the query can return an arbitrary no. of rows)

132

## CURSORS- example



```
DECLARE
```

```
...
```

```
CURSOR propertyCursor IS
```

```
 SELECT propNo,street,sity,postcode
```

```
 FROM ProperyForRent
```

```
 WHERE staffNo='SG14'
```

```
 ORDER by propertyNo;
```

```
BEGIN
```

```
....open the cursor to start selection, then loop to fetch
each row of the result table
```

```
 OPEN propertyCursor
```

```
 LOOP
```

133

## CURSORS- example



```
...fetch next row of the result table
```

```
 FETCH propertyCursor
```

```
 INTO vPropNo, vStreet, vCity, vPostcode
```

```
 EXIT WHEN propertyCursor%NOT FOUND;
```

```
....Display data....
```

```
 END LOOP;
```

```
 IF propertyCursor%ISOPEN THEN CLOSE
 propertyCursor END IF;
```

```
EXCEPTION.....
```

```
END;
```

134



## Subprograms, Stored Procedures, Functions and Packages

- Subprograms are named PL/SQL blocks that can take parameters and be invoked
- PL/SQL has two types of subprograms:
  - 1. (stored) **procedures (P)**
  - 2. **Functions (F)** – return a single value
- **P, F** can take a set of parameters (given by the calling program), perform a set of actions & return data as a parameter.
- Parameter has a specified name and data type designed as: IN, OUT and IN OUT.

135



## Stored Procedures

- Previous anonymous PL/SQL block transformed in procedure:
- **CREATE OR REPLACE PROCEDURE** PropForStaff  
(**IN** vStaffNo VARCHAR2)  
AS.....
- It can then be executed in SQL\*Plus as:  
SQL>SET SERVEROUTPUT ON;  
SQL>EXECUTE PropForStaff('SG14')

136

## PACKAGES (P)

- A **package (P)** is a collection of procedures, functions, variables and SQL statements that are grouped together and stored as a simple program unit.
- **P** has 2 parts: **specification** (declares all public constructs of **P**) & **body** (defines all public & private constructs of **P**), and so implements the specification.

137

## PACKAGES (P)

- For the previous ex., we can define **P** specification as:  
**CREATE OR REPLACE PACKAGE** StaffPropPackage **AS**  
    **procedure** PropForStaff(vStaffNo **VARCHAR2**);  
**END** StaffPropPackage;
- And then, to create the package body as:  
**CREATE OR REPLACE PACKAGE BODY**  
StaffPropPackage **AS**.....  
**END** StaffPropPackage;
- We can then reference PropForStaff procedure as:  
StaffPropPackage.PropForStaff('SG14');

138