7 Data Manipulation Language (DML)

Основна синтакса

SQL-от главно служи за поставување на прашања до базата на податоци. Освен ова, со помош на SQL изразите може да се генерираат и извршат конструкти кои служат за администрација на базата на податоци (креирање на табели, внесување на податоци во табелите, менување/бришење на податоци од табелите, бришење на табели и сл.)
Основната SQL синтакса за поставување на прашање до базата на податоци по кое треба да се добие листа на пронајдени резултати кои ги задоволуваат условите на поставеното прашање е следната:

```
SELECT lista na atributi
FROM lista na tabeli
[WHERE uslovi]
```

Во изразот делот во кој се поставуваат условите е опционален и во одредени случаи може да се испушти.

ЗАДАЧА 1. Дадени се следните релации:

ISP (ISP#, IME_I, STATUS, GRAD_I) / испорачатели
DELOVI (DEL#, IME_D, BOJA, TEZINA) / на делови што се изработуваат
PROIZVODITELI (PR#, IME_P, GRAD_P) / од разни производители
PONUDI (ISP#*, DEL#*, PR#*, KOLICINA, CENA)

Да се напишат SQL изрази за добивање на следниве информации:

а) сите податоци за производителите од Скопје и Битола

SELECT * FROM PROIZVODITELI WHERE GRAD_P = 'Skopje' OR
GRAD_P = 'BITOLA'

б) Шифрите на испорачатели што имаат доставено понуда за дел бр.5 од производител бр.1.

SELECT ISP# FROM PONUDI WHERE DEL#=5 AND PR# = 1

в) Бои на делови за кои што има понуда од испорачателот бр. 8.

SELECT BOJA FROM DELOVI, PONUDI WHERE DELOVI.DEL# = PONUDI.DEL #
AND PONUDI.ISP# = 8

г) броеви на делови што се испорачуваат за производи што се изработуваат во Битола.

```
SELECT PONUDI.DEL# FROM PROIZVODITELI, PONUDI
WHERE GRAD_P = 'BITOLA' AND PROIZVODITELI.PR# = PONUDI.PR#
```

д) Броеви на делови за кои има понуда, произведени од било кој производител, но и испорачателот да е од истиот град како и производителот.

```
SELECT DEL# FROM ISP, PROIZVODITELI, PONUDI
WHERE PONUDI.PR# = PROIZVODITELI.PR# AND PONUDI.ISP# = ISP.ISP#
AND ISP.GRAD_I = PROIZVODITELI.GRAD_P
```

ЗАДАЧА 2. Дадени се следните релации:

SEKTORI (SEKTOR# , IME_SEKTOR, GRAD)
VRABOTENI (VRAB# , IME_VRAB, KVALIFIKACIJA, PLATA, SEKTOR#*)
PROEKTI (PROEKT#, IME_PR, SREDSTVA)
UCESTVA (VRAB#*, PROEKT#* , FUNKCIJA)

Да се напишат SQL прашањата кои ќе ги даваат следните информации:

а) Имиња на сектори и вработени, што се ВКВ и кои имаат плата > 10000, сортирани по името на вработените.

```
SELECT IME_VRAB, IME_SEKTOR FROM VRABOTENI, SEKTORI
WHERE KVALIFIKACIJA = 'VKV' AND PLATA > 10000
AND VRABOTENI.SEKTOR#=SEKTORI.SEKTOR#
ORDER BY  IME_VRAB
```

б) Имиња на вработени кои се вработени во Скопје

```
SELECT VRABOTENI.IME_VRAB FROM VRABOTENI, SEKTORI
WHERE SEKTORI.GRAD = 'SKOPJE'
AND VRABOTENI.SEKTOR# = SEKTORI.SEKTOR#

SELECT IME_VRAB FROM VRABOTENI WHERE SEKTOR#  IN
(SELECT SEKTOR#  FROM SECTORI WHERE GRAD = 'SKOPJE' )
```

в) Преглед на квалификации по сектор

Кога сакаме да избереме некои колони од некоја табела, без притоа да се повторуваат торките, се користи функцијата DISTINCT (различно, посебно).

```
SELECT DISTINCT IME_SEKTOR, KVALIFIKACIJA
FROM SEKTORI, VRABOTENI
WHERE SEKTORI.SEKTOR# = VRABOTENI.SEKTOR#
ORDER BY IME_SEKTOR
```

г) Имиња на вработени што работат во сектор 4 или 7

SELECT IME_V FROM VRABOTENI WHERE SECTOR# = 4 OR SEKTOR# = 7

(Или: WHERE SEKTOR# IN (4, 7))

Операторите minus, union и intersect се однесуваат на множествата издвоени (селектирани) податоци. Колоните што се селектираат (во овој пример SEKTOR#) мора да бидат исти и во двете SELECT наредби.

MINUS	- РАЗЛИКА на множества
UNION	- УНИЈА на множества
INTERSECT	- ПРЕСЕК на множества

д) Броеви на сектори што немаат работници.

SELECT SEKTOR# FROM SEKTORI
MINUS
SELECT SEKTOR # FROM VRABOTENI

ѓ) Броеви на ВКВ вработени што земаат учество во некои проекти.

SELECT VRAB# FROM VRABOTENI WHERE KVALIFIKACIJA = 'VKV'
INTERSECT
SELECT VRAB# FROM UCESTVA

AVG е т.н. агрегатна (збирна, групна) функција и ја дава средната вредност на сите вредности во дадена колона. Други такви функции се:

MAX	- ја дава најголемата вредност во дадена колона
MIN	- ја дава најмалата вредност во дадена колона
SUM	- го дава збирот на вредностите во дадена колона
COUNT	- го дава бројот на редици што се избрани со дадена SELECT наредба

е) Просечен личен доход на ВКВ работници.

SELECT AVG(PLATA) FROM VRABOTENI WHERE KVALIFIKACIJA = ‘VKV‘

ж) Бројот на различни функции што ги обавува во проектите вработен бр. 946.

SELECT COUNT(DISTINCT FUNKCIJA) FROM UCESTVA WHERE VRAB# = 946

Функцијата DISTINCT може да се користи и во комбинација со групните функции спомнати погоре.

ЗАДАЧА 3. Нека ја имаме следната табела (PROIZVOD):

sifra	imePr	edMer	kol	nabCena	rabat	prodCena
L12	Leb T-400	parce	120	20,00	5	20,00
M3	Maslo rastitelno	lit	89	65,01	0	72,00
M8	Maslo od maslinki	lit	34	209,35	0	250,00
K14	Domati	kgr	345	56,00	10	60,00
M16	Maslo bez holesterol	lit	99	70,50	0	75,00
L3	Leb od furna	parce	206	25,00	5	25,00
K22	Piperki	kgr	870	48,89	10	50,00
K30	Jagodi	kgr	200	100,50	10	120,00
L2	Lepinja	parce	189	23,00	3	23,00

Кога се поставуваат прашања се дефинира листата на атрибути кои ќе се излистаат во резултатот, или со * се добиваат сите атрибути. Во одредени случаи може да се јави потреба од воведување на изведени атрибути кои ги нема во ниедна од табелите во кои се поставило SQL прашањето.

1. Излистај ги имињата на производите заедно со набавната цена и продажната цена без ДДВ.

   SELECT imePr, nabCena, prodCena*100/118 FROM PROIZVODI

   Ако сакаме изведената вредност во резултатот да добие некое име, тогаш пишуваме:

   SELECT imePr, nabCena, (prodCena*100/118) prodCenaBezDDV
   FROM PROIZVODI

2. Излистај ги имињата на производите заедно со набавната и продажната цена како и разликата помеѓу продажната и набавната цена.

   SELECT imePr, nabCena nabavna, prodCena prodazna, (prodCena-nabCena) razlika
   FROM PROIZVODI

Кога сакаме да извршиме филтрирање на резултатот според одреден атрибут кој е од тип стринг, тогаш освен стандардните релациони оператори можеме да го користиме и операторот LIKE. Со помош на овој оператор се проверува дали за дадена вредност на атрибутот важи некој регуларен израз. Обично во стрингот кој се бара се користат џокер знаците:

- %, кој заменува повеќе карактери
- _, кој заменува еден карактер

3. Дај ги податоците за производите за кои името им почнува на ‘Maslo’.

   SELECT * FROM PROIZVODI WHERE imePr like ‘Maslo%’

Кога се поставуваат услови за атрибути кои не се стрингови, тогаш може да се користат стандарните аритметички, релациони и логички оператори. Кога се бара

некоја вредност за која однапред се дефинирани вредностите кои треба да ги исполни, SQL прашањето може да се постави на два начини. Исто така на два начини може да се постави SQL прашањето со кое се проверува дали некој атрибут прима вредности во некој интервал.

4. Дај ги производите кои се мерат во литри и килограми.

   SELECT * FROM PROIZVODI WHERE edMer='lit' OR edMer='kgr'

   SELECT * FROM PROIZVODI WHERE edMer IN ('lit', 'kgr')

5. Дај ги прозводите кои имаат цена меѓу 50,00 и 250,00 денари.

   SELECT * FROM PROIZVODI WHERE prodCena>=50 AND prodCena<=250

   SELECT * FROM PROIZVODI WHERE prodCena BETWEEN 50 AND 250.

ЗАДАЧА 4. Нека ги имаме табелите PART и ORDERS со следниве записи:

PARTNUM	DESCRIPTION	PRICE
54	PEDALS	54.25
42	SEATS	24.50
46	TIRES	15.25
23	MOUNTAIN BIKE	350.45
76	ROAD BIKE	530.00
10	TANDEM	1200.00

ORDEREDON	NAME	PARTNUM	QUANTITY	REMARKS
15-MAY-1996	TRUE WHEEL	23	6	PAID
19-MAY-1996	TRUE WHEEL	76	3	PAID
2-SEP-1996	TRUE WHEEL	10	1	PAID
30-JUN-1996	TRUE WHEEL	42	8	PAID
30-JUN-1996	BIKE SPEC	54	10	PAID
30-MAY-1996	BIKE SPEC	10	2	PAID
30-MAY-1996	BIKE SPEC	23	8	PAID
17-JAN-1996	BIKE SPEC	76	11	PAID
17-JAN-1996	LE SHOPPE	76	5	PAID
1-JUN-1996	LE SHOPPE	10	3	PAID
1-JUN-1996	AAA BIKE	10	1	PAID
1-JUL-1996	AAA BIKE	76	4	PAID
1-JUL-1996	AAA BIKE	46	14	PAID
11-JUL-1996	JACKS BIKE	76	14	PAID

Многу често се јавува потреба од вгнездување на SQL изрази еден во друг. На овој начин може да се изврши филтрирање според одреден атрибут за кој би имале многу вредности за споредба или овие вредности за споредба може однапред да не се знаат.

1. Дај ги нарачките во кои се продадени производи кои почнуваат на ROAD.

   SELECT * FROM ORDERS WHERE PARTNUM IN
   (SELECT PARTNUM FROM PART WHERE DESCRIPTION LIKE 'ROAD%')

2. Најди ја просечната цена на нарачките.

   SELECT AVG(O.QUANTITY * P.PRICE) FROM ORDERS O, PART P
   WHERE O.PARTNUM = P.PARTNUM

3. Дај ги имињата на купувачите и датумите на нарачките кои нарачале роба со вредност поголема од просечната цена на сите нарачки.

   SELECT O.NAME, O.ORDEREDON, O.QUANTITY * P.PRICE TOTAL
   FROM ORDERS O, PART P
   WHERE O.PARTNUM = P.PARTNUM AND O.QUANTITY * P.PRICE >
   (SELECT AVG(O.QUANTITY * P.PRICE) FROM ORDERS O, PART P
   WHERE O.PARTNUM = P.PARTNUM)

Многу често се јавува потреба од прашања кои ќе вршат групирање на одредено множество на податоци според одредена вредност. За таа цел се користат клучните зборови GROUP BY опционално во комбинација со HAVING. При тоа доколку за дадена групација сакаме да провериме услов со користење на агрегатна функција за дадената групација на записи, тогаш тоа мора да го проверуваме со HAVING.

4. Дај преглед на просечната нарачана количина по нарачател.

   SELECT NAME, AVG(QUANTITY) FROM ORDERS GROUP BY NAME

5. Ако сакаме 4) да го проширime со следниот услов: просечната нарачана количина по нарачател да биде поголема од просечната нарачана количина за сите нарачки:

   SELECT NAME, AVG(QUANTITY) FROM ORDERS GROUP BY NAME
   HAVING AVG(QUANTITY) > (SELECT AVG(QUANTITY) FROM ORDERS)

6. Излистај ги шифрите на сите производи заедно со вкупната сума остварена од нарачките за тој производи, како и бројот на нарачки за тој производ. Дополнително да се отфрлат сите шифри на производи за кои вкупната сума е помала или еднаква на просечната остварена продажба **за тој производ**.

   SELECT O.PARTNUM, SUM(O.QUANTITY*P.PRICE), COUNT(PARTNUM)
   FROM ORDERS O, PART P WHERE P.PARTNUM = O.PARTNUM
   GROUP BY O.PARTNUM HAVING SUM(O.QUANTITY*P.PRICE) >
   (SELECT AVG(O1.QUANTITY*P1.PRICE) FROM PART P1, ORDERS O1
   WHERE P1.PARTNUM = O1.PARTNUM AND **P1.PARTNUM = O.PARTNUM**)

Резултат:

```
PARTNUM    SUM          COUNT
========= ========== =======
       10     8400.00       4
       23     4906.30       2
       76    19610.00       5
```

Во SQL синтаксата постојат и клучните зборови EXISTS, ANY и ALL кои се користат во WHERE делот од SQL прашањата.
EXISTS прима под-прашање како аргумент и враќа TRUE ако под-прашањето врати било што, а враќа FALSE ако резултатот од под-прашањето е празно множество.
ANY враќа TRUE ако барем еден резултатот од споредбата на даден атрибут и даден резултат добиен во под-прашањето го задоволува условот.
ALL враќа TRUE само ако сите споредби го задоволуваат условот.

```
SELECT NAME, ORDEREDON FROM ORDERS WHERE EXISTS
(SELECT * FROM ORDERS WHERE NAME ='MOSTLY HARMLESS')

SELECT NAME, ORDEREDON FROM ORDERS WHERE NAME = ANY
(SELECT NAME FROM ORDERS WHERE NAME ='TRUE WHEEL')

SELECT NAME, ORDEREDON FROM ORDERS WHERE NAME > ANY
(SELECT NAME FROM ORDERS WHERE NAME ='JACKS BIKE')

SELECT NAME, ORDEREDON FROM ORDERS WHERE NAME <> ALL
(SELECT NAME FROM ORDERS WHERE NAME ='JACKS BIKE')
```

Користење на погледи

Кога се прават покомплексни прашања, може да се искористат погледите, или виртуелните табели за да се зачува резултатот. Откога ќе се креира поглед кон одредено множество на податоци, истиот може да се третира како друга табела. Откога ќе се изврши промена на податоците во базата на податоци, тогаш и резултатот во погледот се менува. Погледите не завземаат физичко место во базата на податоци како табелите. Општата синтакса за креирање на поглед е следната:

```
CREATE VIEW <view_name> [(column1, column2...)] AS
SELECT <columns'_names>
FROM <tables'_names>
```

ЗАДАЧА 5. Нека ја имаме следната база на податоци составена од следните табели:

FIRMA (ime, adresa, grad, drzava)
SMETKA (SB#, tip, sostojba, imeBanka)
UPLATA (ime*, SB#*, suma, zadocneto)

1. Креирање на едноставен поглед врз табелата UPLATA.

   CREATE VIEW POM_UPLATA AS SELECT * FROM UPLATA;

2. Креирање на поглед за уплатите на сметката со шифра 4.

   CREATE VIEW UPLATI_4 AS
   SELECT * FROM POM_UPLATA WHERE SB# = 4;

3. Креирање на поглед преку кој ќе се печатат поштенските информации за фирмите.

   CREATE VIEW ENVELOPE (FIRMA, POSTENSKA_ADRESA) AS
   SELECT ime, adresa + " " + grad + ", " + drzava FROM FIRMA;

4. Да се прикаже вкупниот број на плаќања извршени по држави, заедно со вкупната сума уплатена со тие плаќања.

   CREATE VIEW PLAKANJA_PO_DRZAVI(DRZAVA,BR_PLAKANJA,VKUPNO) AS
   SELECT FIRMA.drzava, COUNT(UPLATA.*), SUM(UPLATA. suma)
   FROM UPLATA, FIRMA WHERE UPLATA.ime = FIRMA.ime
   GROUP BY FIRMA.drzava;

5. Да се даде поглед кон сите задоцнети плаќања со пресметана камата.

   CREATE VIEW ZADOCNETI_PLAKANJA (ime, vkupno, tipSmetka) AS
   SELECT UPLATA.ime, UPLATA.suma * 1.10, SMETKA.tip
   FROM UPLATA, SMETKA WHERE UPLATA.SB# = SMETKA.SB# and zadocneto;

Користење на индекси

Податоците од базата на податоци може да се превземат на два начини. Првиот начин (метод со секвенцијален пристап) го користи SQL за да ги измине секвенцијално сите записи сместени во базата на податоци и во секој чекор да извршува споредба со она што се бара. Овој начин е неефикасен, меѓутоа е единствениот начин да се лоцира одреден запис. Вториот начин е преку користење на индекси, преку кои SQL прашањето ќе користи директен метод за пристап. На овој начин SQL користи стебловидна структура за чување и пребарување низ индексните податоци. На овој начин се добива во брзина и ефикасност во лоцирањето на одреден запис во базата на податоци.

Општата синтакса за креирање на индекс е следната:

```
CREATE INDEX index_name
ON table_name(column_name1, [column_name2], ...);
```

ПРИМЕР:

1. Ако сакаме да креираме индекс по колоната SB# од табелата UPLATA, тогаш имаме:

   CREATE INDEX smetkaIndeks ON UPLATA(SB#);

2. Уништување на веќе креираниот индекс се прави со наредбата:

   DROP INDEX smetkaIndeks;