ОПЕРАЦИИ С ДАННИТЕ В РЕЛАЦИОННИ БАЗИ ДАННИ

dimitrova@tu-sofia.bg
pct.tu-sofia.bg/dd/pik3

ВИДОВЕ ОПЕРАЦИИ

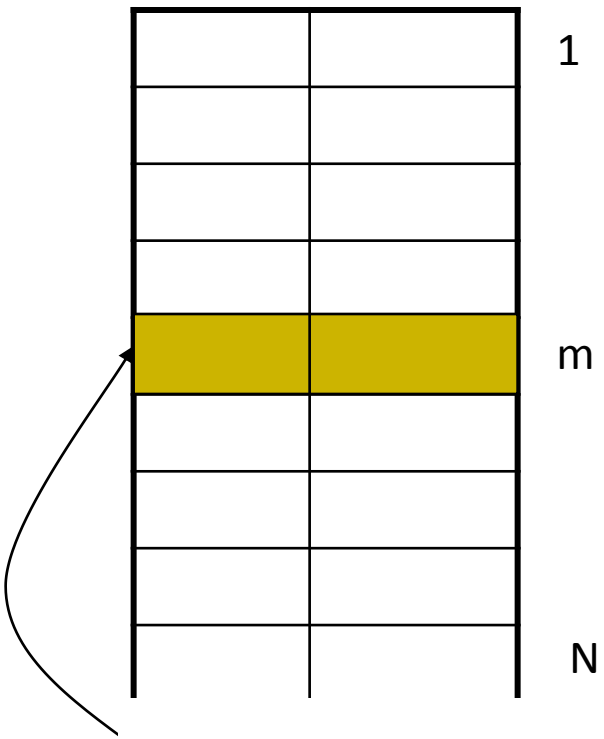
Операции

- Резултатът от операции с релации е нова релация
- Три нива на представяне : три вида операции
  - външно ниво: tables – операции за поддържане на данните и извличане на информация
  - концептуално: sets – релационна алгебра
  - вътрешно: files – търсене и локализация на данни

Операции на външно ниво

- Поддържане на данните - в актуално състояние
  - добавяне на нови днни
  - триене на излишни данни
  - промяна на съществуващи данни, без промяна на количеството им
- Търсене на данни
  - локализиране на определени елементи от данните
  - селектиране на данни, отговарящи на определено условие
- Агрегиране и генерализация
  - групиране, игнориране на различията

Добавяне и изтриване



Добавяне на ред

N:= N+1;

tab[N] := *newrow*;

Изтриване на ред *tab[m]*

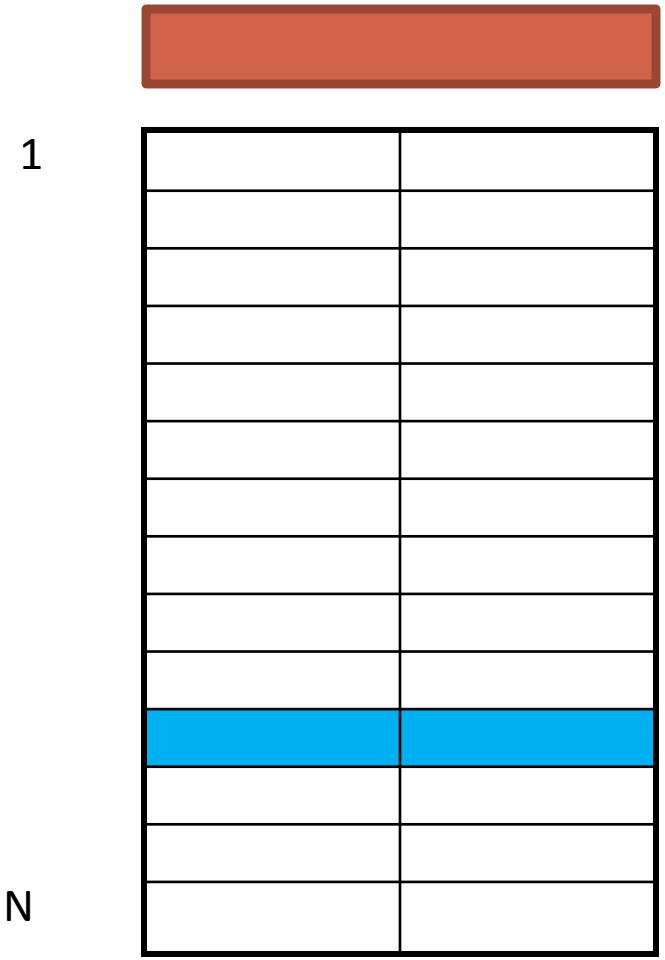
tab[m] := tab[N];

N:= N-1;

Операции на вътрешно ниво

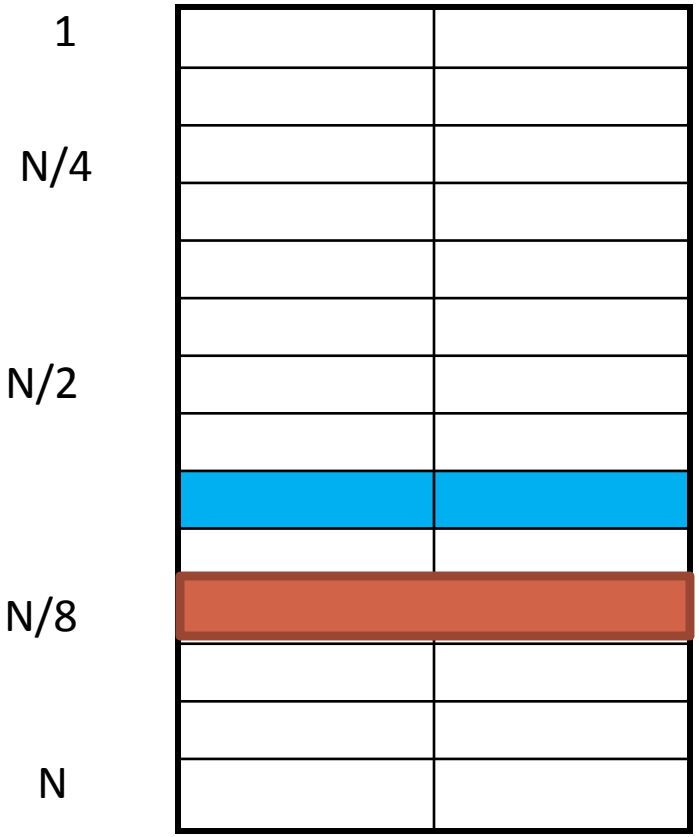
- Навигация
  - индексиране
  - ключове
- Сортиране и търсене
- Методи и алгоритми за търсене
  - последователно
  - двоично
  - блоково
  - индексно

Последователно търсене в неподредена таблица



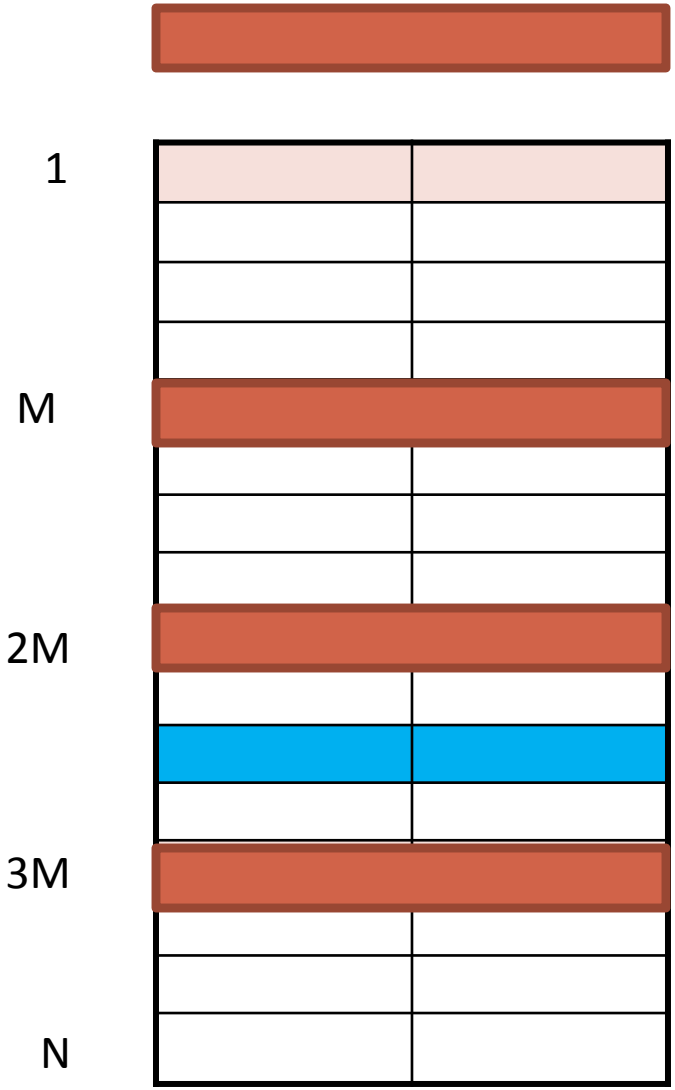
Сложност O(N)

Двоично търсене в подредена таблица



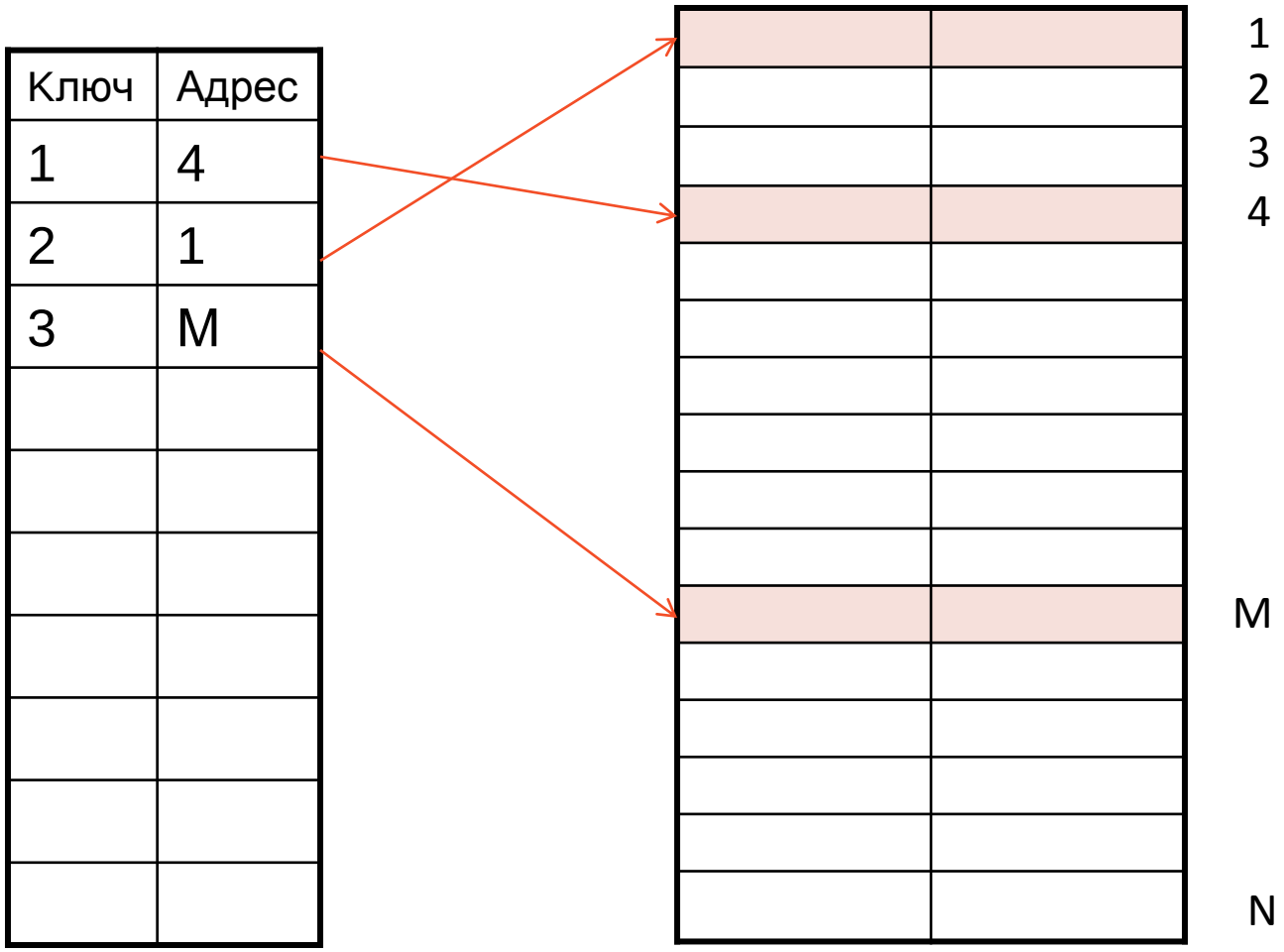
Сложност $O(\log_2 N)$

Блоково търсене в подредена таблица



Сложност O(N/M)

Индексно търсене



Операции на концептуално ниво

- Операции с множества (Relational Algebra)
- Множество
  - колекция от подобни, но добре различими елементи
- Резултат от операция с множества е множество
- Описва се резултатното множество

- Операция: оператори и операнди

Унарни операции

- Операции с един операнд

селекция – намаляване на размера на релацията

where
relational operators < = >
set operators AND OR NOT

$$\sigma_{\texttt{lastname='Донев'}}\ (\texttt{Customer})$$

проекция – намаляване на броя на атрибутите

$$\Pi_{\texttt{lastname, firstname}}\ (\texttt{Customer})$$

Селекция

T1

A	B	C
a	b	c
d	a	f
c	b	d

T2

A	B	C
a	b	c
c	b	d

Проекция

T1

A	B	C
a	b	c
d	a	f
c	b	d

T2

C	A
c	a
f	d
d	c

Бинарни операции

- Операции с два операнда

Обединение Union

$\pi_{lnameID}$ (Customer1) U $\pi_{lnameID}$ (Customer2)

Сечение Intersection

$\pi_{lnameID}$ (Customer1) ∩ $\pi_{lnameID}$ (Customer2)

Разлика Difference

$\pi_{lnameID}$ (Customer1) - $\pi_{lnameID}$ (Customer2)

Умножение JOIN

елемент по елемент

```
inner join (natural join)- only once
left join
right join
outer join
```

Деление

намиране на подмножество, което изцяло съдържа второто множество

Сечение

T1	T2	T1 ∩ T2
a b c	b g a	d a f
d a f	d a f	
c b d		

Обединение

T1	T2	T1 U T2
a b c	b g a	a b c
d a f	d a f	d a f
c b d		c b d
		b g a

Разлика

T1	T2	T1 -T2
a b c	b g a	a b c
d a f	d a f	c b d
c b d		

Умножение

T1	T2	T1 x T2
a b c	b g a	a b c b g a
d a f	d a f a	b c d a f
c b d		d a f b g a
		d a f d a f
		c b d b g a
		c b d d a f

Агрегиране и генерализация

Агрегиране – произвеждане на една стойност от много редове

- описателна статистика - `SUM, AVG, COUNT`
- сравняване - `MIN, MAX`

Генерализация

- групиране + агрегиране

SQL

SQL structured query language

SQL = DDL + DML + DCL

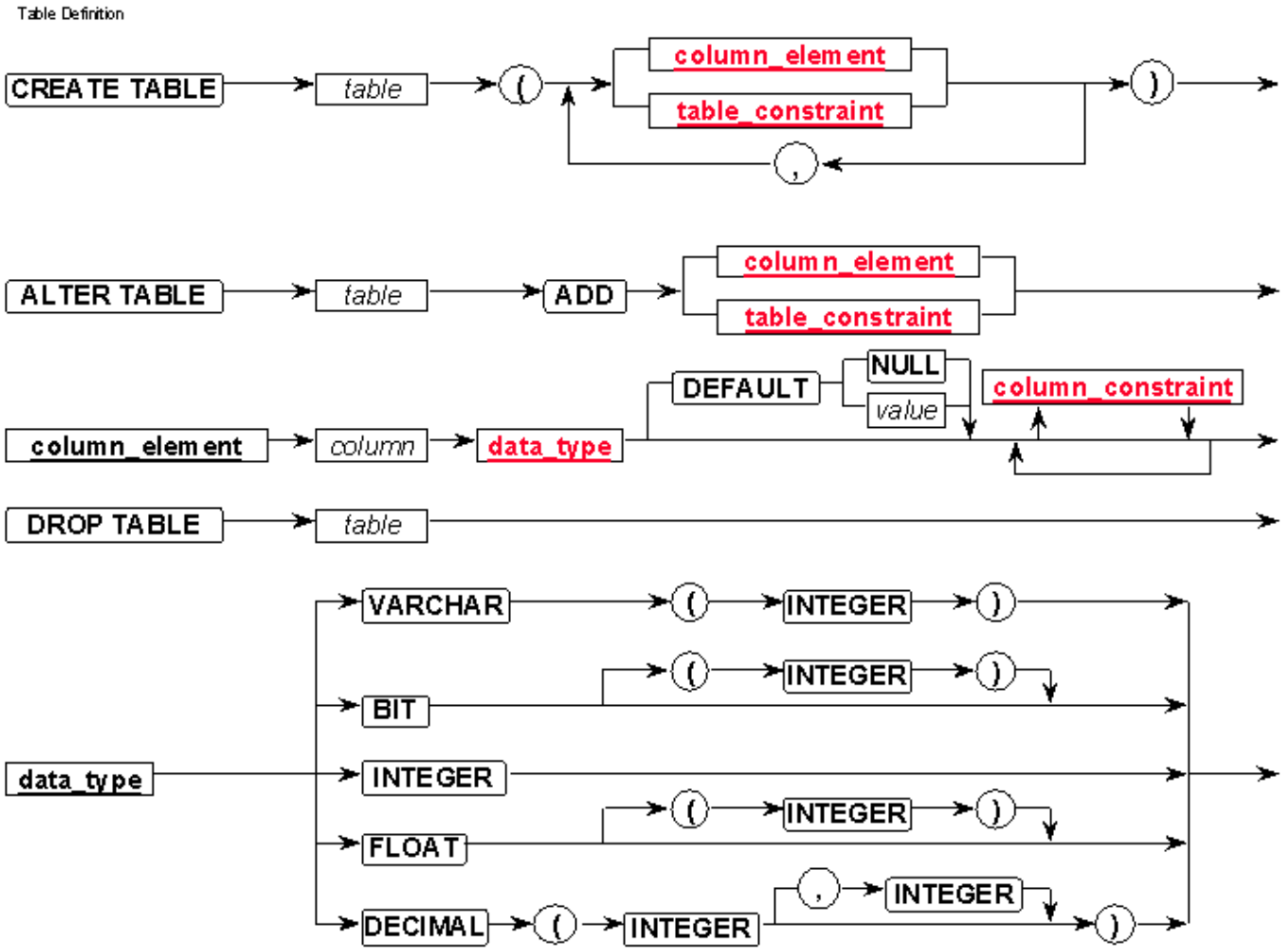
DDL	data definition language
DML	data manipulation language
DCL	data control language

Дефиниране на обекти

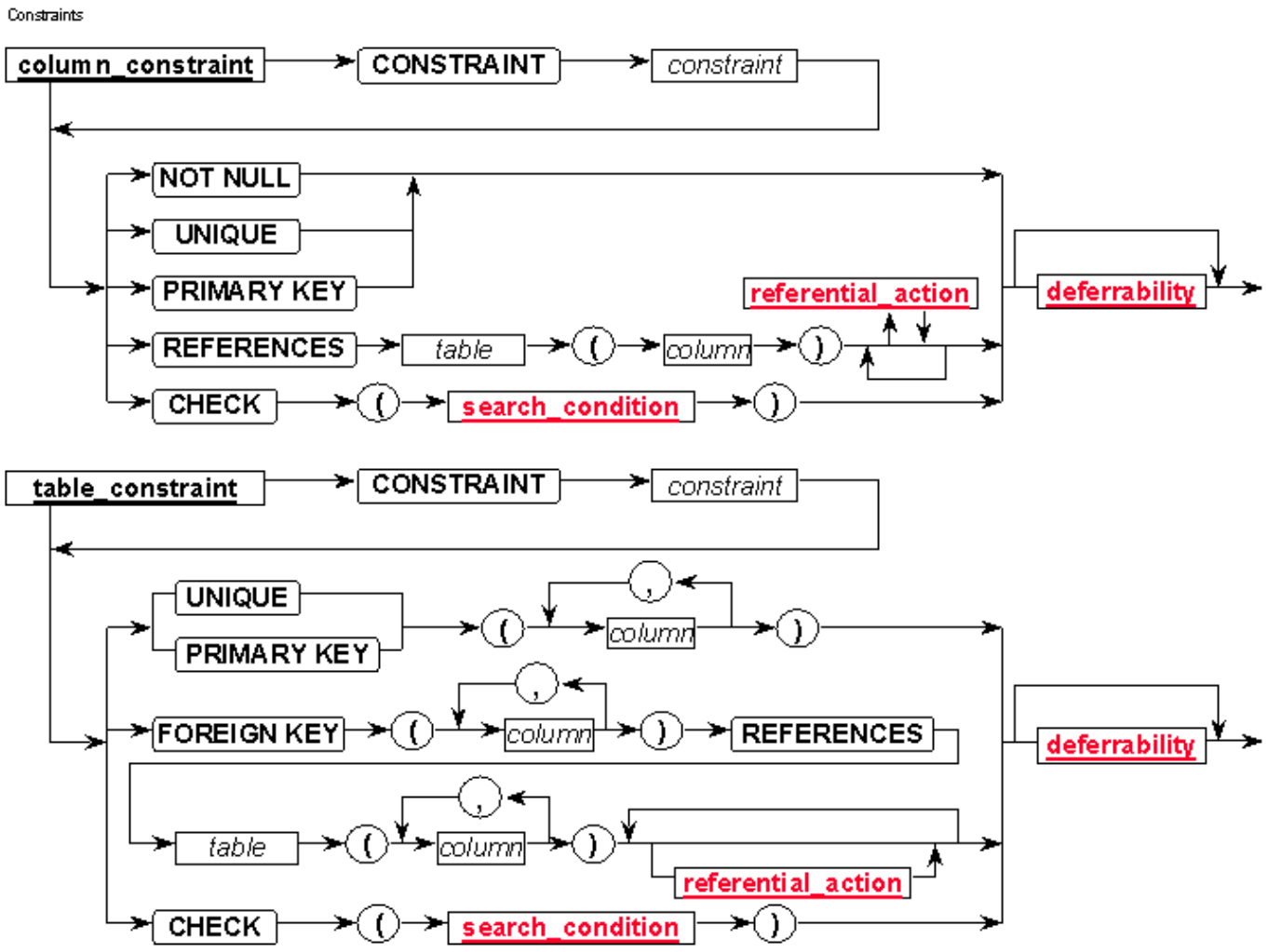
Дефиниране на таблици, атрибути, отношения и индекси

CREATE TABLE	създава нова таблица
ALTER TABLE	променя структурата на съществуваща таблица
DROP TABLE	изтрива таблица
CREATE INDEX	създава индекс за поле или група полета
DROP INDEX	изтрива индекс

Синтактична диаграма: Операции с таблици



Ограничения



constrains

- Правило, асоциирано с колона, което данните в колоната трябва да спазват
- Видове
  - "unique" – не може два или повече записи да имат еднакви стойности в тази колона
  - "not null“ – не може да има празна клетка в тази колона
  - "primary key" - идентификатор на реда, първичен ключ

CREATE TABLE

- Формат

```
create table "tablename"
     ("column1" "data type",
      "column2" "data type",
      "column3" "data type");
create table "tablename"
     ("column1" "data type" [constraint],
      "column2" "data type" [constraint],
      "column3" "data type" [constraint]);
```

Типове данни

- **Символни типове**
  - **Char** за отделни символи.
  - **Char(n)** за низове от *n* символа , **Varchar** (DB2)
  - **Long** (ORACLE) за низове от максимум 65655 символа.
  - MS ACCESS :Text, Memo
- **Числени типове**
  - **number, number(n)** (**float** при SQL/DS) за цели с променлива или фиксирана дължина
  - **number(m,n)** (**Decimal** при SQL/DS) за реални числа с дължина *m* и с *n* цифри след десетичната точка
  - MS ACCESS: **Number**(**Integer**, **LongInteger**, **Byte**, **Single**, **Double**) , **Currency**, **Autonumber**
- **Други типове**
  - **Date** (**Time** в SQL/DS) представя дата като сложна структура от полета
  - **Raw** (ORACLE V6) за двоични данни, вкл. bitmaps
  - **BLOB** (Binary Large OBjects)
  - MS ACCESS: **Date/Time**, **Logical**, **OLE** обекти.

Примери

char(size) Стринг с фиксирана дължина, max 255 B

varchar(size) Стринг с променлива дължина,
max size – в скоби

number(size) Числова стойност с max брой цифри

date Дата

number(size,d) Числова стойност с max брой цифри
общо " size" и след десетичната точка
цифри "d"

Пример

```
CREATE TABLE EMP
(
  EMPNO     INTEGER NOT NULL,
  ENAME     VARCHAR(10) CHARACTER SET ISO8859_1,
  JOB       VARCHAR(9) CHARACTER SET ISO8859_1,
  MGR       INTEGER,
  HIREDATE  TIMESTAMP,
  SAL       NUMERIC(9, 2),
  COMM      NUMERIC(9, 2),
  DEPTNO    INTEGER,
CONSTRAINT PK_EMP PRIMARY KEY (EMPNO)
);
ALTER TABLE EMP ADD CONSTRAINT FK_DEPTNO FOREIGN KEY
    (DEPTNO) REFERENCES DEPT (DEPTNO);
```

Резултат

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17.12.1980	800		20
7499	ALLEN	SALESMAN	7698	20.2.1981	1600	300	30
7521	WARD	SALESMAN	7698	22.2.1981	1250	500	30
7566	JONES	MANAGER	7839	02.4.1981	2975		20
7654	MARTIN	SALESMAN	7698	28.9.1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01.5.1981	2850		30
7782	CLARK	MANAGER	7839	09.6.1981	2450		10
7788	SCOTT	ANALYST	7566	19.4.1987	3000		20
7839	KING	PRESIDENT		17.11.1981	5000		10
7844	TURNER	SALESMAN	7698	08.9.1981	1500		30
7876	ADAMS	CLERK	7788	23.5.1987	1100		20
7900	JAMES	CLERK	7698	03.12.1981	950		30
7902	FORD	ANALYST	7566	03.12.1981	3000		20
7934	MILLER	CLERK	7782	23.1.1982	1300		10

emp(empno,ename, job, mgr, hiredate, sal, comm ,deptno);

Друг пример

```
CREATE TABLE DEPT
(
  DEPTNO   INTEGER NOT NULL,
  DNAME    VARCHAR(14) CHARACTER SET ISO8859_1,
  LOC      VARCHAR(13) CHARACTER SET ISO8859_1,
CONSTRAINT PK_DEPT PRIMARY KEY (DEPTNO)
);
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

dept(deptno,dname,loc);

DROP TABLE

- Модел:

```
drop table "tablename"
```

- Пример:

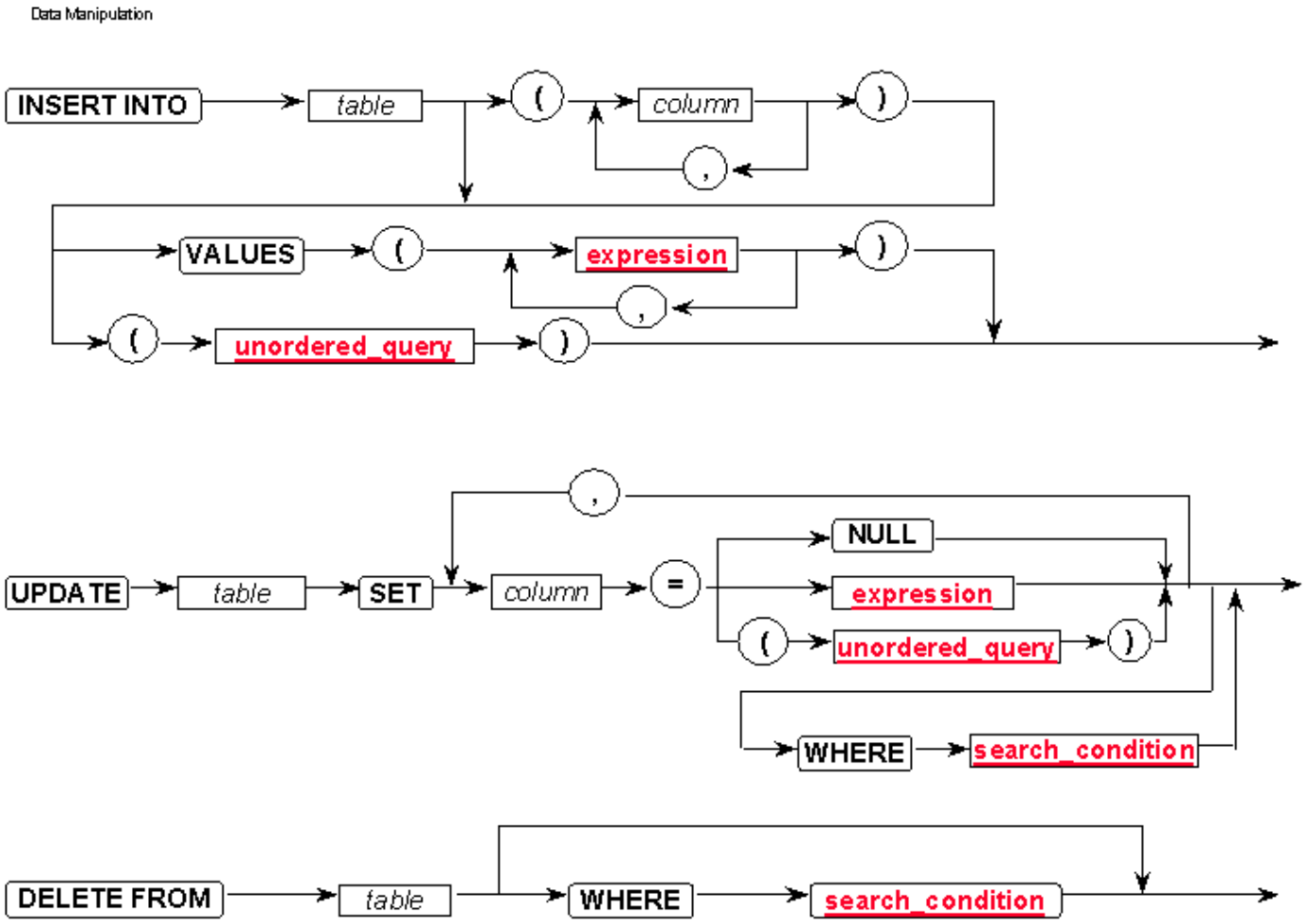
```
drop table myemployees;
```


Манипулиране с обекти

Извличане, вмъкване и изменяне на съдържанието на базата данни

INSERT INTO	добавя записи в таблица
DELETE	изтрива записи от таблица
UPDATE	актуализира записи
SELECT ... INTO	записва резултати в таблица
SELECT... FROM ... WHERE ...	извлича информация

Обработка на данни



INSERT INTO

Модел:

```
insert into "tablename"
  (first_column,...last_column)
values
  (first_value,...last_value);
```

Пример:

```
insert into employee
(first, last, age, address, city, state)
values
('Luke','Duke', 45, '2130 Boars Nest',
  'Hazard Co', 'Georgia');
```

DELETE FROM

- Модел:

```
delete from "tablename"
  where "columnname" OPERATOR "value"
  [and|or "column" OPERATOR "value"];
```

- Примери:

```
delete from employee;

delete from employee where lastname = 'May';

delete from employee
  where firstname = 'Mike'
     or firstname = 'Eric';
```

UPDATE

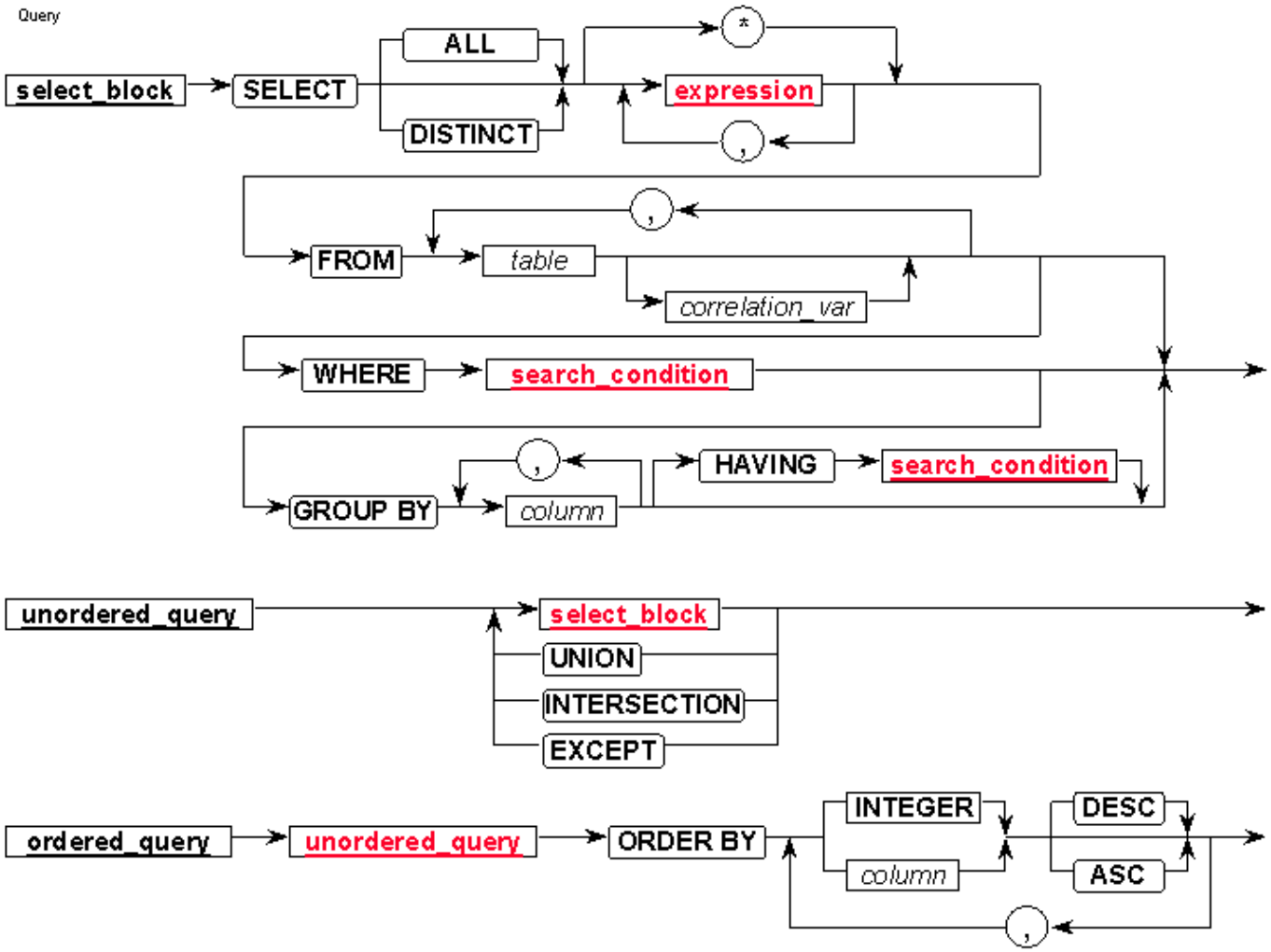
- Модел:

```
update "tablename"
set "columnname" = "newvalue"
[,"nextcolumn" = "newvalue2"...]
where "columnname" OPERATOR "value"
[and|or "column" OPERATOR "value"];
                                    [] = optional
```

- Пример:

```
update phone_book
set area_code = 623 where prefix = 979;

update phone_book
set last_name = 'Smith', prefix=555, suffix=9292
  where last_name = 'Jones';

update employee set age = age+1 where
  first_name='Mary' and last_name='Williams';
```

Селекция и проекция



SELECT

- Модел:

```
select "column1" [,"column2",etc] from
  "tablename" [where "condition"];
```

- Примери:

```
select first, last, city from empinfo
where first LIKE 'Er%';

select first, last from empinfo
where last LIKE '%s';

select * from empinfo
where first = 'Eric';
```

SELECT

- Агрегиране и генерализация

```
SELECT [ALL | DISTINCT] column1[,column2]
  FROM table1[,table2]
  [WHERE "conditions"]
  [GROUP BY "column-list"]
  [HAVING "conditions]
  [ORDER BY "column-list" [ASC|DESC] ]
```

- Ред на изпълнение:
  1. Ако има клауза WHERE се премахват всички кортежи, които не удовлетворяват условието.
  2. Извършва се групиране и изчисляване на агрегатните стойности.
  3. Премахват се всички групи, които не удовлетворяват условието в клаузата HAVING.

Агрегиране

- Функции:
  - MIN – връща най-малката стойност в колоната
  - MAX - връща най-голямата стойност в колоната
  - SUM - връща сумата от числените стойности в колоната
  - AVG - връща средната стойност в колоната
  - COUNT - връща броя стойности в колоната
  - COUNT(*) – връща броя редове в таблицата

- Примери:

```
SELECT AVG(salary) FROM employee;

SELECT AVG(salary) FROM employee
  WHERE title = 'Programmer';

SELECT Count(*) FROM employees;
```

Агрегиране и генерализация

- GROUP BY - групират се резултатите, според даден критерий и кортежите от всяка група да се обработват от агрегатни функции
  - за атрибути, които не са параметри на агрегатна функция
- Примери:

```
SELECT column1, SUM(column2)
  FROM "list-of-tables"
  GROUP BY "column-list";

SELECT max(salary), dept
  FROM employee
  GROUP BY dept;
```

Дефиниране на подмножество

- Модел:

```
SELECT column1, SUM(column2)
  FROM "list-of-tables"
  GROUP BY "column-list"
  HAVING "condition";
```

- Пример:

```
SELECT dept, avg(salary)
  FROM employee
  GROUP BY dept
  HAVING avg(salary) > 20000;
```

SELECT deptno, MIN(sal), MAX (sal)FROM emp
WHERE job = 'CLERK' GROUP BY deptno;

SELECT deptno, MIN(sal), MAX (sal) FROM emp GROUP BY deptno;

Сортиране на селекцията

- ORDER BY {ASC | DESC}

- Модел:

```
SELECT column1, SUM(column2)
  FROM "list-of-tables"
  ORDER BY "column-list" [ASC | DESC];
```

- Примери:

```
SELECT employee_id, dept, name, age, salary
  FROM employee_info
  WHERE dept = 'Sales'
  ORDER BY salary;
```

```
SELECT employee_id, dept, name, age, salary
  FROM employee_info
  WHERE dept = 'Sales'
  ORDER BY salary, age DESC;
```

Селекция, проекция и съединение

- Модел:

```
SELECT "list-of-columns"
  FROM table1,table2
  WHERE "search-condition(s)"
```

- Пример:

```
SELECT customer_info.firstname,
  customer_info.lastname,purchases.item
  FROM customer_info, purchases
  WHERE customer_info.customer_number =
  purchases.customer_number;
```

Селекция, проекция и умножение

- Търсене в две или повече таблици едновременно
- Модел:

```
SELECT "list-of-columns"
  FROM table1 JOIN table2
  ON key1 = key2 WHERE "search-condition(s)"
```

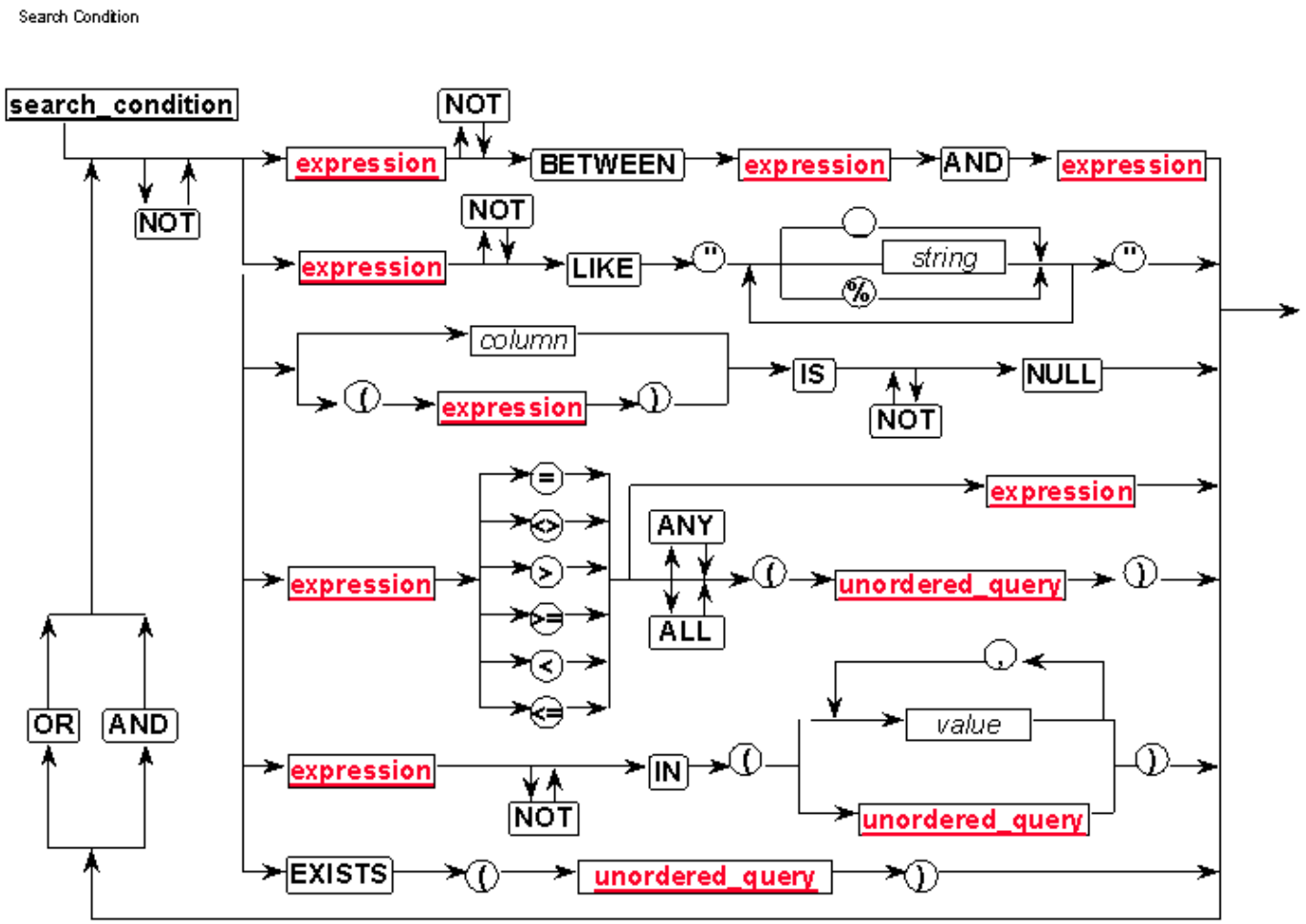
- Примери:

```
SELECT title, artist
  FROM album JOIN track
   ON (album.asin=track.album)
  WHERE song = 'Alison';

SELECT title, COUNT(*)
  FROM album JOIN track ON (asin=album)
 GROUP BY title
```

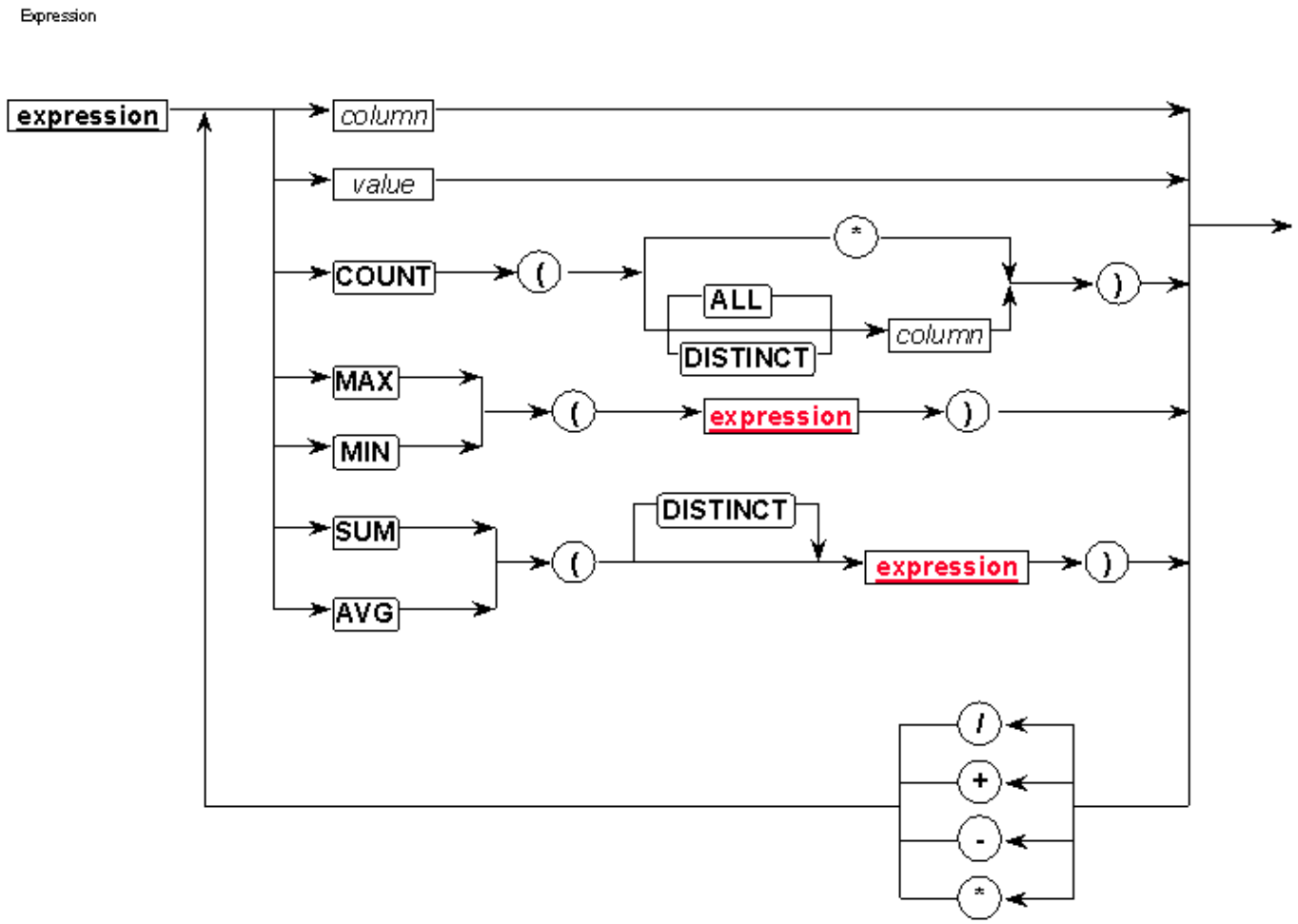
Подзаявки за търсене

```
SELECT ename, sal  FROM emp
WHERE
deptno = 10
AND
sal > ( SELECT MAX(sal) FROM emp WHERE deptno = 20);

SELECT dname, deptno FROM dept
WHERE
EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno);
```

Търсене


Search Condition
search_condition
NOT
expression
BETWEEN
expression
AND
expression
NOT
NOT
expression
LIKE
string
%
column
(
expression
)
IS
NULL
NOT
=
<>
>
>=
<
<=
expression
ANY
ALL
expression
(
unordered_query
)
OR
AND
,
value
expression
NOT
IN
(
)
unordered_query
EXISTS
(
unordered_query
)

Израз


Expression
expression
column
value
COUNT
(
*
ALL
DISTINCT
column
)
MAX
MIN
(
expression
)
SUM
AVG
(
DISTINCT
expression
)
/
+
-
*

Оператори

Код	Операция
+, -	Положително, обратен знак
*, / , %	Умножение, деление
+, -	Събиране, изваждане
|| (&)	конкатенация
^	Степен,
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	сравнения
NOT	отрицание
AND	конюнкция
OR	дизюнкция

Функции

- Математически функции
  - ABS(x) - връща |x|
  - SIGN(x) – връща знака на x as -1, 0, or 1
  - MOD(x,y) modulo – остатъкът от делението на x и y (x%y)
  - FLOOR(x) – най-голямото цяло <= x
  - CEILING(x) или CEIL(x) - най-малкото цяло >= x
  - POWER(x,y) - x на степен y
  - ROUND(x) – закръгление на x до най-близкото цяло
  - ROUND(x,d) – закръгление на x до знак d
  - SQRT(x) – квадратен корен от x

Функции

•Обработка на символи

- ASCII(char) - връща ASCII кода на символа (ASC в ACCESS)
- CHR(n) - връща символа, чийто ASCII код е параметър.
- INITCHAR(string) - връща низа с като прави първата буква главна (липсва в MS ACCESS)
- LENGTH(string) - брой на символите в низ
- LPAD(string, n, char), RPAD(string, n, char) запълва наляво или надясно низа с *n* екзепляра на символа *char* (SPACES(n) връща *n* интервала в MS ACCESS)
- LTRlM(string, char), RTRIM(string, char) премахва всичко, което се намира наляво или надясно от символа char, ако той се намира в низа
- TRANSLATE(string, cl, c2) замества в низа всички появявания на символа cl с c2 (липсва в MS ACCESS)
- SUBSTR(string, pos, length) връща подниз със зададена дължина , започващ от pos (MID$(string, pos, length) в ACCESS)
- UPPER(string), LOWER(string) конвертира низа с главни или малки букви (UCASE, LCASE в ACCESS)
- || е операция за конкатенация на низове (В ACCESS операторът е &)

Функции

Конвертиране на типове

- TO_CHAR(n [,format]) обръща число *n* в символен низ според зададения формат

  (STR$ и FORMAT в ACCESS)

- TO_CHAR(date[, format]) обръща дата в низ

  (STR в ACCESS)

- TO_DATE(string,format) обръща низ в дата
  (CDATE в ACCESS)

- TO_NUMBER(string) обръща низ в цяло или реално число

  (VAL в ACCESS)

Примери

= [Orders]![Freight] * 1.1

PrimeFreight: [Freight] * 1.1= [Subtotal] + [Freight]

FreightPercentage: Sum([Freight])/Sum([Subtotal]) *100

=[RequiredDate]-[ShippedDate]

=[EmployeeTotal]/[CountryTotal]

Логически изрази

Оператори

= Equal
> Greater than
< Less than
>= Greater than or equal
<= Less than or equal
<> Not equal to
LIKE *
AND
OR
NOT

Примери

<>0

0 Or >100

Like "K???"

<#1/1/96#

>=#1/1/97# And <#1/1/98#

Агрегиране

=Count(*)

=Count([OrderID])

FreightPercentage:
Sum([Freight])/Sum([Subtotal]) *100

=Sum([Sales])

=Sum([Quantity]*[Price])

=[Sales]/Sum([Sales])*100

=Avg([Freight])

Операции с дати

=Date()

=Date( )- 30

YearHired:

DatePart("yyyy",[HireDate])

=Format(Now(), "ww")

=DatePart("yyyy", [OrderDate])

=DateAdd("y", -10, [PromisedDate])

=DateDiff("d", [OrderDate],
[ShippedDate])

Операции с текст

="N/A“

FullName: [FirstName] & " " & [LastName]

Address2: [City] & " " & [Region] & " " & [PostalCode]

ProductInitial: Left([ProductName], 1)

TypeCode: Right([AssetCode], 2)

AreaCode: Mid([Phone],2,3)

=Trim([Address])

=IIf(IsNull([Region]), [City] & " " & [PostalCode], [City] & " " & [Region] & " " & [PostalCode])

IN

- Модел:

```
SELECT col1, SUM(col2)
  FROM "list-of-tables"
  WHERE col3 IN (list-of-values);
```

- Пример:

```
SELECT employeeid, lastname, salary
  FROM employee_info
  WHERE lastname IN
  ('Hernandez','Jones','Roberts','Ruiz');
```

= [Country] In ("France", "Italy", "Spain") And
Len([PostalCode])<>5

BETWEEN

- Модел:

```
SELECT col1, SUM(col2)
  FROM "list-of-tables"
  WHERE col3 BETWEEN value1 AND value2;
```

- Примери:

```
SELECT employeeid, age, lastname, salary
  FROM employee_info
  WHERE age BETWEEN 30 AND 40;
```

Алтернативи

=IIf([Confirmed] = "Yes", "Order Confirmed", "Order Not Confirmed")

=IIf(IsNull([Country]), " ", [Country])

=IIf(IsNull([Region]), [City] & " " & [PostalCode], [City] & " " & [Region] & " " & [PostalCode])

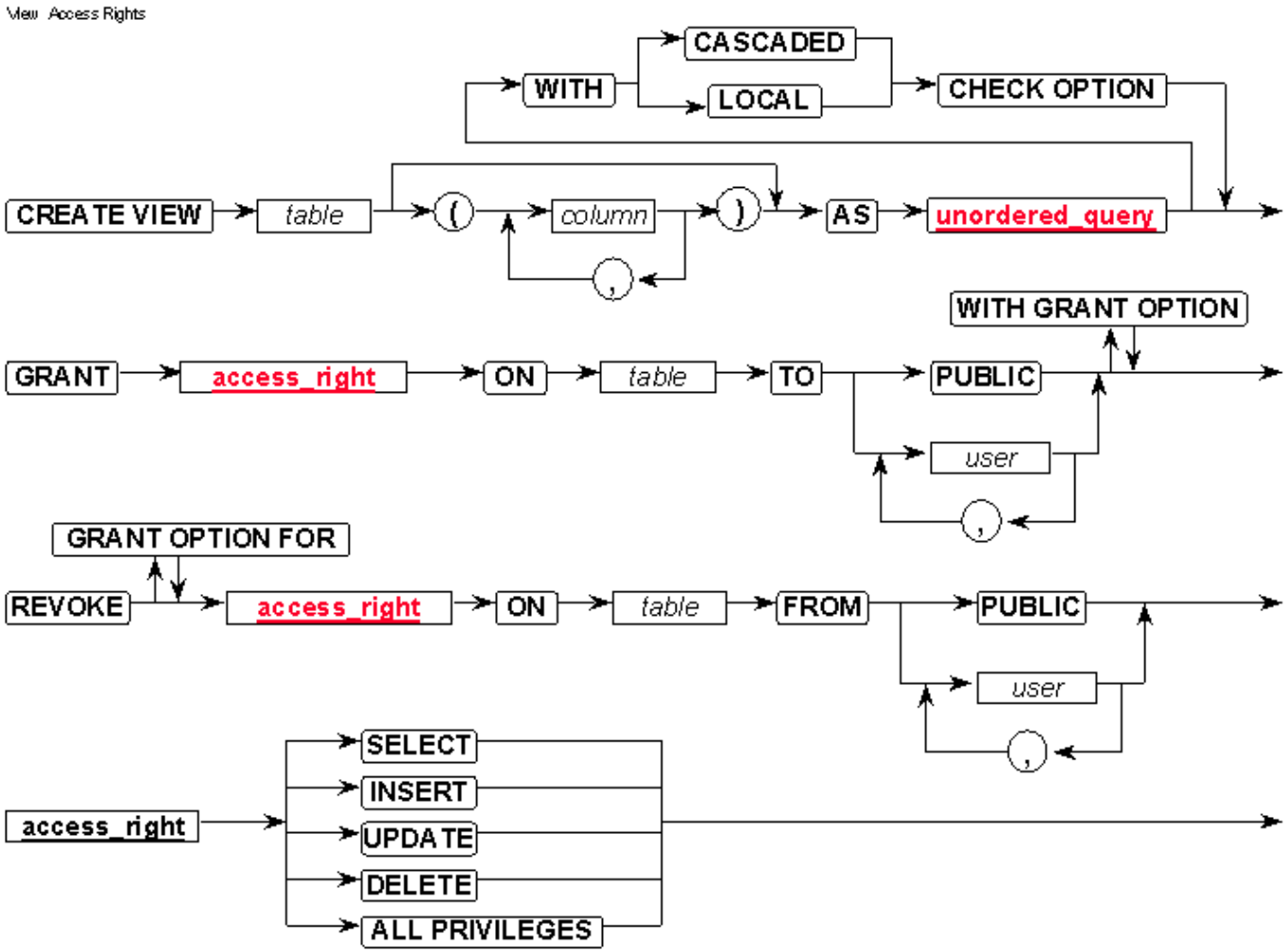
=IIf(IsNull([RequiredDate] - [ShippedDate]), "Check for a missing date", [RequiredDate] - [ShippedDate])

Маски

ShipName	Like "S*"	започва с ...
ShipName	Like "*Import"	завършва с ...
ShipName	Like "[A-D]*"	започва с някоя от ...
ShipName	Like "*ar*"	съдържа ...
ShipName	Like "Maison Dewe?“	последната буква е неизвестна

?	замества един символ
*	замества нула или повече символи
#	замества една цифра
[списък]	произволен символ от списъка
[!списък]	произволен символ извън списъка

Създаване на изгледи и права за достъп



Потребители и права за достъп

- Процедури в зависимост от СУБД
- Нов потребител

```
sp_addlogin 'scott', 'tiger';
CREATE DATABASE scott;
USE scott;
sp_changedbowner scott;
```

- Смяна на парола

```
sp_password @old='tiger', @new='tiger'
```

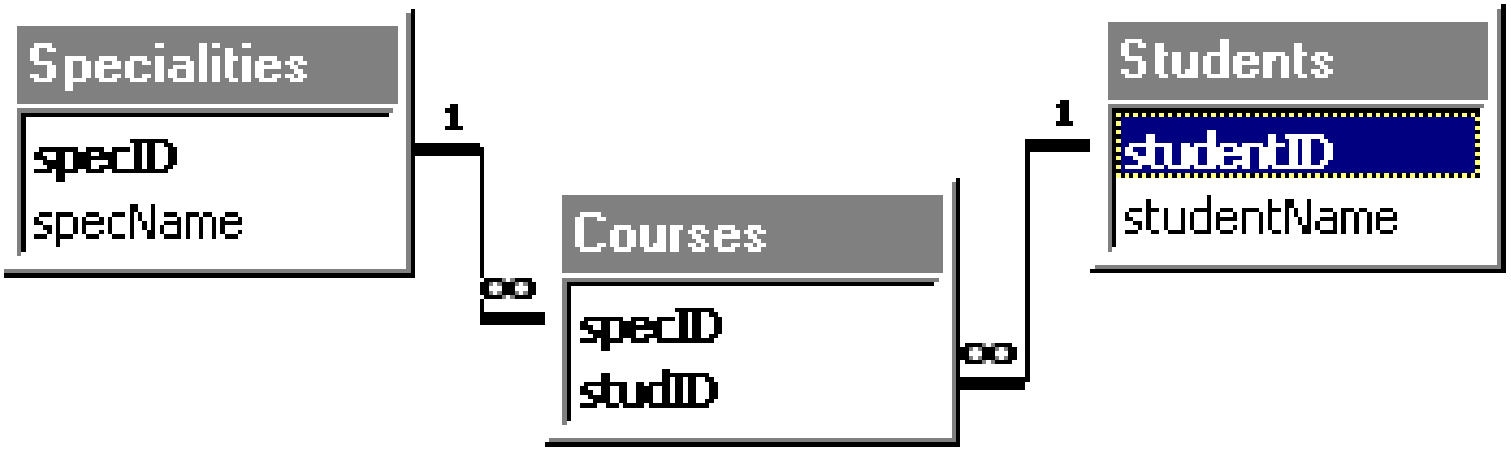
- Идентификация на потребител и собственик (dbo)

```
USE scott;
SELECT user, user_id();
USE gisq;
SELECT user, user_id()
```

Управление на правата на достъп

- **Общи права на достъп**
  - GRANT CONNECT | RESOURCE | ROLE | DBA(ADMIN) TO потребител IDENTIFIED BY парола;
  - CONNECT, RESOURCE са предефинирани роли в СУБД Oracle. Тяхното използване не се препоръчва в последната версия.
  - ROLE - именувано множество от привилегии ,които могат да се дават като едно цяло. То се обработва от командите:
    - CREATE ROLE
    - ALTER ROLE
    - SET ROLE
- Достъп до обекти (с евентуално право за препредаване) :
  - GRANT SELECT | INSERT | DELETE | UPDATE | ALTER | INDEX | CLUSTER ON таблица или изглед TO потребител или роля [WITH GRANT OPTION] ;
  - GRANT OPTION дава на потребителя право да предава собствените си права на други потребители
- Отнемане на права
  - REVOKE отнема права на потребителите
  - REVOKE право ON таблица или изглед FROM потребител

Още примери

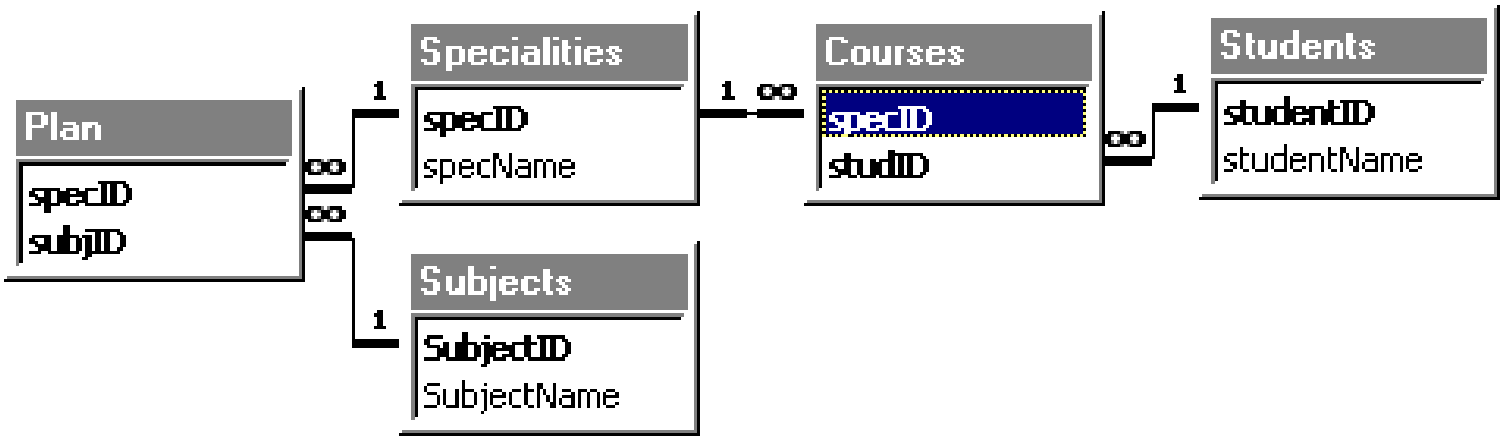


Specialities (specID, specName)

Students (studentID, studentName)

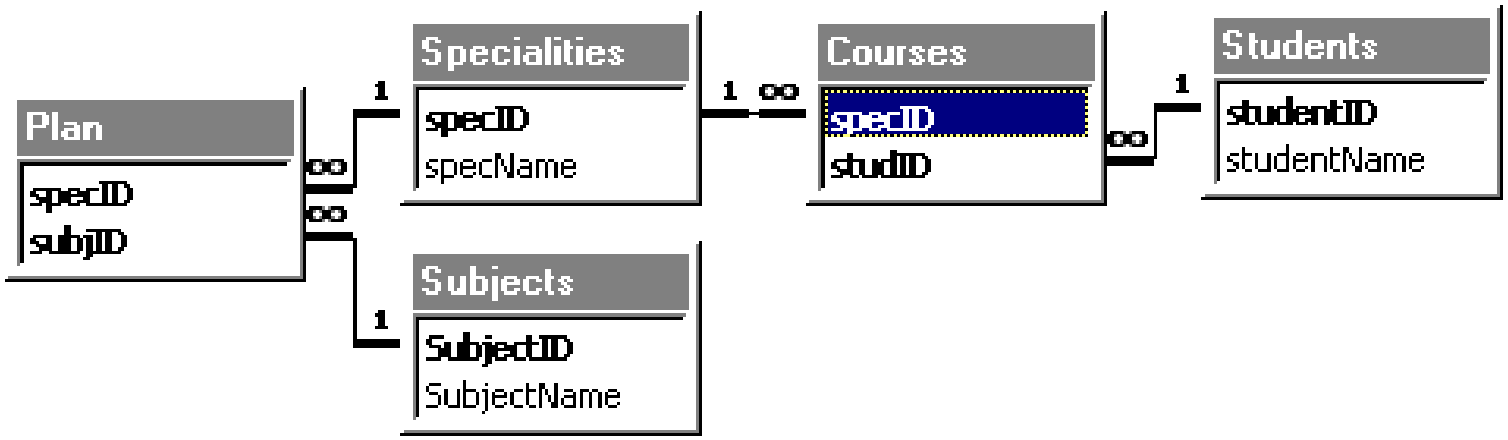
Courses (specID, studID)

Създаване на таблица



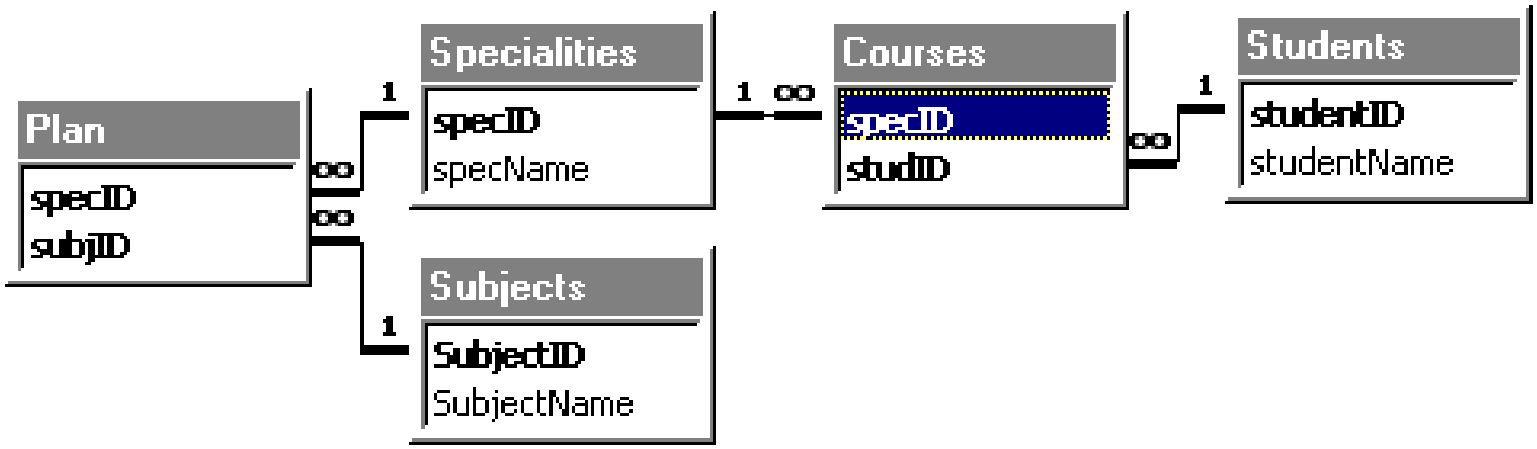

```
CREATE TABLE Plan
(
        specID integer,
        subjID integer,
        constraint tc1 PRIMARY KEY (specID, subjID),
        constraint tc2 FOREIGN KEY (specID) references Specialities,
        constraint tc3 FOREIGN KEY (subjID) references Subjects
);
```


Добавяне в таблица



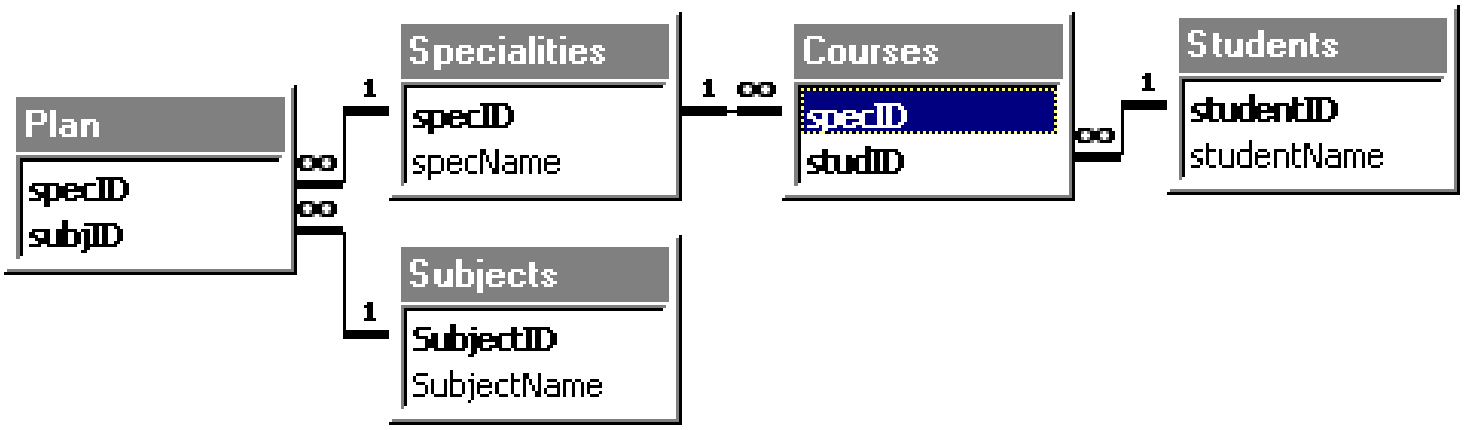

```
INSERT INTO TABLE Plan  (specID, subjID) VALUES (2, 2);

INSERT INTO TABLE Plan (specID 2, subjID 4);
```


Изтриване на запис от таблица



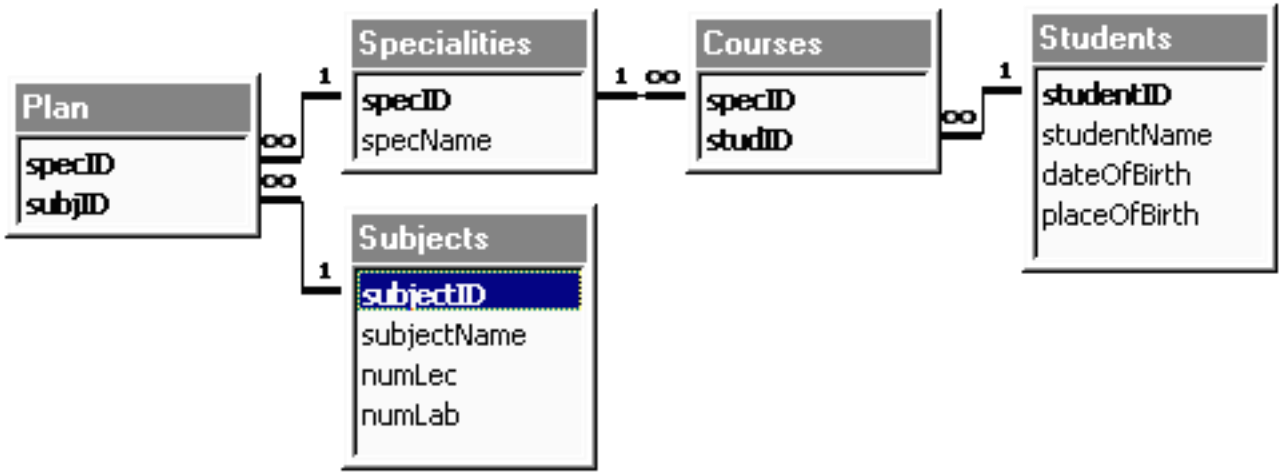

```
DELETE FROM Students WHERE studentName = 'Maria';

UPDATE Students
        SET studentID = studentID + 1
        WHERE studentName = 'Kate';
```

Промяна на структурата на таблица



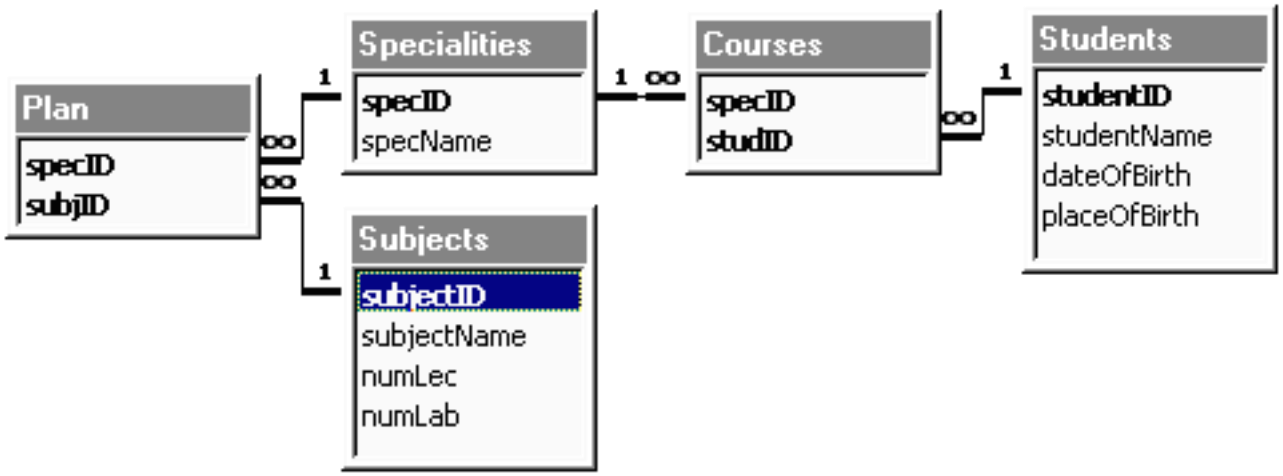

```
ALTER TABLE Students
        ADD COLUMN dateOfBirth,
        ADD COLUMN placeOfBirth,
        ADD FOREIGN KEY (placeOfBirth) REFERENCES Map;

DROP TABLE Subjects;
```


Селекция



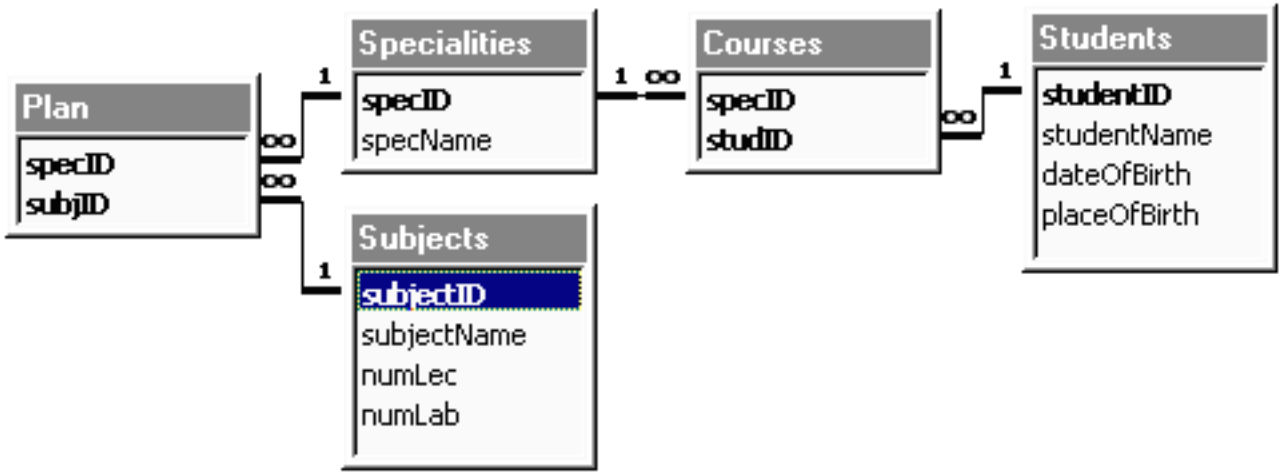

```
SELECT * FROM Students;

SELECT studentName FROM Students;

SELECT * FROM Students WHERE studentID=14;
```


Селекция



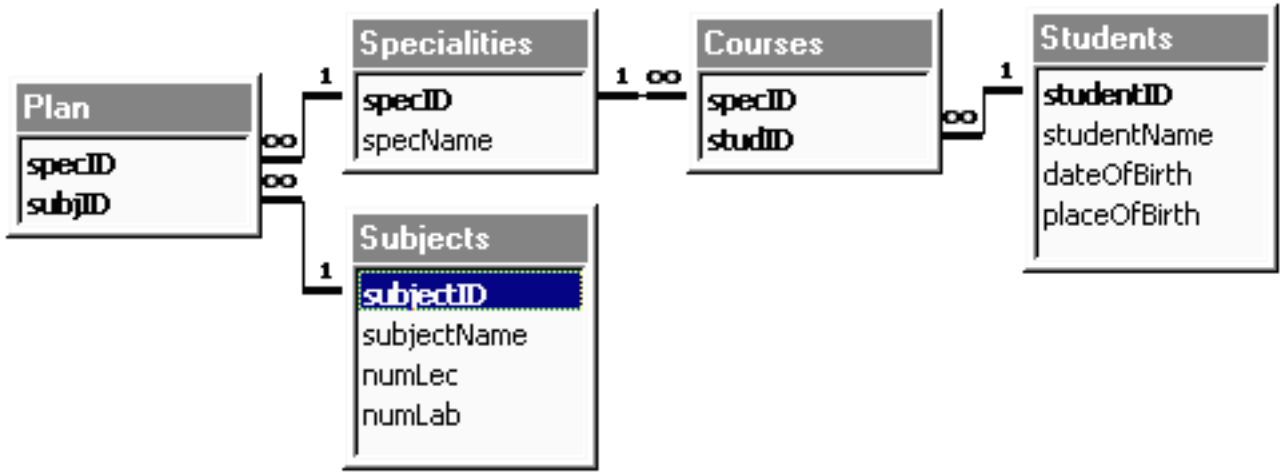

```
SELECT DISTINCT specID FROM Plan;

SELECT studentName FROM Students  WHERE studentName LIKE “K*”;

SELECT * FROM Subjects   WHERE numLec>20 AND numLab>20;
```


Селекция




```
SELECT * FROM Students ORDER BY studentName;

SELECT (numLec + numLab)  AS Horarium FROM Subjects;

SELECT * FROM Students JOIN Courses
        ON Students.studentID = Courses.studID;
```


Селекция и обединение




```
SELECT studentName COUNT (*) FROM Students
                GROUP BY DayPart("yy", dateOfBirth);

SELECT subjectName MIN(numLec + numLab) FROM Subjects
                HAVING numLec >= 10 OR numLab <= 20;

(SELECT * FROM Students )
        UNION / EXCEPT / INTERSECT (SELECT * FROM FootballPlayers);
```


Упражнение

Дадена е релационната схема:

Supplier(SupNo, Name, Country, Address, TelNo)
Article(ArtNo, Name, Unit)
Price(SupNo, ArtNo, PriceUn)
Delivery(LivNo, Date, SupNo)
Detail(LivNo, ArtNo, Quant)

Да се напишат операторите SQL даващи отговор на следните въпроси:

1. Кои са доставчиците с имена, съдържащи “ова ”?

```
SELECT * FROM Supplier WHERE Name LIKE ‘%ова%’
```

2. Какви са минималните, максималните и средните цени за всяка стока?

```
SELECT A.NAME, Max(PriceUn), Min(PriceUn), Avg(PriceUn)
FROM (Supplier s INNER JOIN Price P ON S.SupNo=P.SupNo)
JOIN Article A ON P.ArtNo = A.ArtNo
GROUP BY A.NAME;
```

Упражнение

4. Какъв е броят на стоките доставян от всеки доставчик?

```
SELECT SupNo, S.Name, Count(ArtNo)
FROM Supplier S LEFT JOIN Price P ON S.SupNo = P.SupNo
GROUP BY SupNo, S.Name;
```

5. Месечен отчет по доставчици – всеки доставчик стоки на каква сума е доставил?

```
SELECT S.SupNo, Name, Sum(Quant*PriceUn)
FROM Supplier S, Price P, Delivery D, Detail L
WHERE S.SupNo = P.SupNo and S.SupNo = D.SupNo and D.LivNo= L.LivNo and
P.ArtNo=L.ArtNo and Date Between 1.4.03 and 30.4.03
GROUP BY S.SupNo, Name
ORDER By Name;
```

6. Средна сума на дневните доставки за последния месец

```
CREATE Day AS SELECT Date, Sum(Quant*PriceUn) AS Suma
FROM Price P, Delivery D, Detail L
WHERE  D.LivNo= L.LivNo and P.ArtNo=L.ArtNo
          and Date Between 1.4.03 and 30.4.03
GROUP BY Date;
SELECT AVG(Suma) FROM Day;
```