доц. д-р Цветанка Георгиева-Трифонова

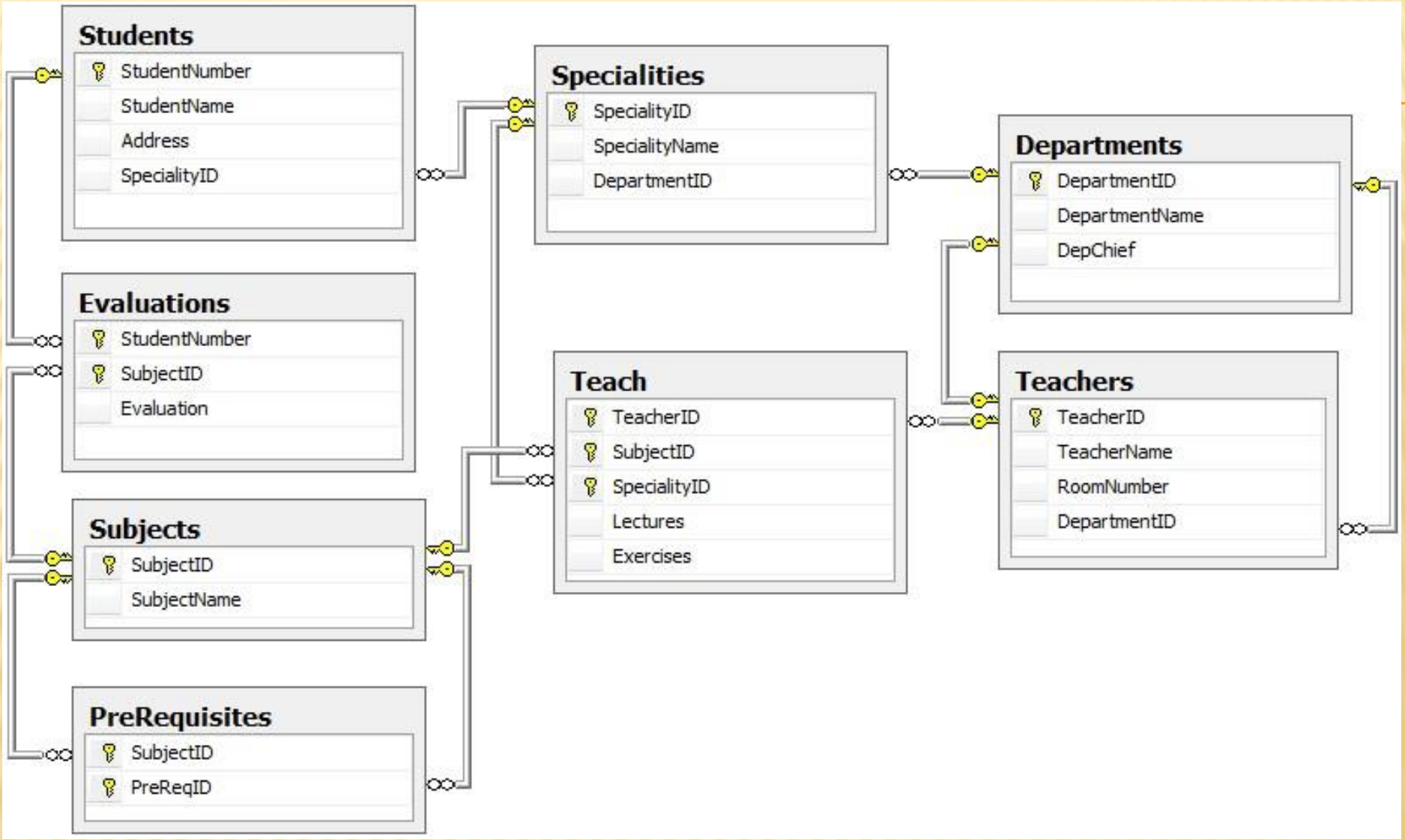
ЕЗИКЪТ SQL

СЪДЪРЖАНИЕ

- SQL заявки (*is null*, *between*, *in*, *like*, *order by*)
- Обобщаване на данните с помощта на агрегатни функции (*sum*, *avg*, *count*, *min*, *max*, *group by*, *having*)
- Дефиниране на данни със SQL
- Модифициране на данни със SQL

SQL

- SQL (*Structured Query Language – език за структурирани заявки*)
  - стандартен език за дефиниране, обновяване и извличане на данни от релационните бази от данни.
- Използват се неформалните понятия за обектите в релационния модел:
  - таблица (релация);
  - колона (атрибут);
  - ред (кортеж).



Базата от данни *StudentsDB*

SQL ЗАЯВКИ

SELECT *column_list*

FROM *table_list*

[WHERE *search_conditions*]

[GROUP BY *group_by_column_list*]

[HAVING *search_conditions*]

[ORDER BY *order_column_list* [ASC | DESC]]

SQL ЗАЯВКИ (2)

Нека релацията R се състои от атрибутите a_1, a_2, ..., a_n.

$\pi_{a_1,a_2,\ldots,a_l}(R)$	SELECT a_1, a_2, …, a_l FROM R
$\sigma_C(R)$	SELECT * FROM R WHERE C
$R \times S$	SELECT * FROM R, S
$R \rhd\!\lhd_\theta S$	SELECT * FROM R, S WHERE θ

SQL ЗАЯВКИ (3)

Нека релацията R се състои от атрибутите $a_1, a_2, \ldots, a_n$ и $a_1, a_2, \ldots, a_l$ са всички атрибути, дефинирани едновременно в схемите на релациите R и S.

$R \triangleright\!\triangleleft S$

```
SELECT *
FROM R, S
WHERE R.a1 = S.a1 AND R.a2 = S.a2 AND … AND R.al = S.al
```

$R \triangleright\!\triangleleft S$

```
SELECT *
FROM R
INNER JOIN S
   ON R.a1 = S.a1 AND R.a2 = S.a2 AND … AND R.al = S.al
```

$R \overset{\circ}{\triangleright\!\triangleleft} S$

```
SELECT *
FROM R
FULL OUTER JOIN S
   ON R.a1 = S.a1 AND R.a2 = S.a2 AND … AND R.al = S.al
```

SQL ЗАЯВКИ (4)

Нека релацията R се състои от атрибутите $a_1, a_2, \ldots, a_n$ и $a_1, a_2, \ldots, a_l$ са всички атрибути, дефинирани едновременно в схемите на релациите R и S.

$R \overset{\circ}{\rhd\lhd}_L S$

```
SELECT *
FROM R
LEFT OUTER JOIN S
    ON R.a1 = S.a1 AND R.a2 = S.a2 AND … AND R.al = S.al
```

$R \overset{\circ}{\rhd\lhd}_R S$

```
SELECT *
FROM R
RIGHT OUTER JOIN S
    ON R.a1 = S.a1 AND R.a2 = S.a2 AND … AND R.al = S.al
```

SQL ЗАЯВКИ (5)

$\tau_L(R)$

```
SELECT *
FROM R
ORDER BY L
```

$\delta(R)$

```
SELECT DISTINCT a1, a2, …, an
FROM R
```

$\rho_{S(b_1,b_2,\ldots b_n)}(R)$

```
SELECT a1 AS b1, a2 AS b2, …, an AS bn
FROM R AS S
```

SQL ЗАЯВКИ (6)

$$\gamma_{StudentNumber,AVG(Evaluation)\rightarrow Average,\ COUNT(SubjectID)\rightarrow ctSubject}(Evaluations)$$

```
SELECT StudentNumber,
       AVG(Evaluation) AS Average,
       COUNT(SubjectID) AS ctSubject
FROM Evaluations
GROUP BY StudentNumber
```

$$\sigma_{ctSubject\geq 3}(\gamma_{StudentNumber,AVG(Evaluation)\rightarrow Average,\ COUNT(SubjectID)\rightarrow ctSubject}(Evaluations))$$

```
SELECT StudentNumber,
       AVG(Evaluation) AS Average,
       COUNT(SubjectID) AS ctSubject
FROM Evaluations
GROUP BY StudentNumber
HAVING COUNT(SubjectID) >= 3
```

SQL ЗАЯВКИ (7)

$$\gamma_{StudentNumber,AVG(Evaluation)\rightarrow Average,\ COUNT(SubjectID)\rightarrow ctSubject}(Evaluations)$$

```
SELECT Stu
       AVG
       COU
FROM Evalu
GROUP BY S
```

StudentNumber	Average	ctSubject
13987	4.833333	6
15111	4.500000	2
15321	4.166666	6
17111	4.500000	6
17123	4.666666	6
17654	4.666666	6

$$\sigma_{ctSubject\geq 3}(\gamma_{StudentNumber,AVG(Evaluation)\rightarrow Average,\ COUNT(SubjectID)\rightarrow ctSubject}(Evaluations))$$

```
SELECT St
       AV
       CO
FROM Eval
GROUP BY
HAVING COU
```

StudentNumber	Average	ctSubject
13987	4.833333	6
15321	4.166666	6
17111	4.500000	6
17123	4.666666	6
17654	4.666666	6

SQL ЗАЯВКИ (8)

$R \cup S$

```
SELECT * FROM R
UNION
SELECT * FROM S
```

$R \cap S$

```
SELECT * FROM R
INTERSECT
SELECT * FROM S
```

$R \setminus S$

```
SELECT * FROM R
EXCEPT
SELECT * FROM S
```

```
SELECT *
FROM R
WHERE C
```

StudentNumber	StudentName	Address
15222	Петя Маринова Иванова	NULL
17222	Соня Стоянова Петрова	NULL

- Критериите за избор могат да включват стандартни оператори +, –, *, / , % (остатък при деление), >, <, >=, <=, =, <>, както и някои специални SQL оператори:
  - IS NULL проверява дали на колоната е зададена стойност. За проверка дали стойността не е NULL се използва *column_name* IS NOT NULL.

```
SELECT StudentNumber, StudentName, Address
FROM Students
WHERE Address IS NULL
```

СТОЙНОСТ NULL

- SQL използва тризначната логика (three-valued logic)
  - TRUE, FALSE и UNKNOWN

Израз	Логическа стойност на израза
NOT TRUE	FALSE
NOT FALSE	TRUE
NOT UNKNOWN	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

СТОЙНОСТ NULL – ПРИМ

CompanyName	City
Денница Стар ЕООД	Варна
Мики-92 ЕООД	София

```
SELECT CompanyName, City FROM Customers
WHERE City <> 'Велико Търново'
```

- `NULL <> 'Велико Търново'`
  - стойност `UNKNOWN` – не се извежда от `WHERE`

```
SELECT CompanyName, City FROM Customers
WHERE City <> 'Велико Търново' OR City IS NULL
```

CompanyName	City
Денница Стар ЕООД	Варна
МИЛСТОР 90 ЕООД	NULL
Мики-92 ЕООД	София

```
SELECT *
FROM R
WHERE C
```

StudentNumber	StudentName	Address	SpecialityID
17111	Мартин Иванов Петров	ул. Христо Ботев, 12	2
17123	Иван Иванов Иванов	ул. В. Левски, 7	2
17222	Соня Стоянова Петрова	NULL	2

- BETWEEN *min_value* AND *max_value* проверява дали стойността е в зададения интервал.

```
SELECT *
FROM Students
WHERE StudentNumber BETWEEN '17111' AND '17555'
```

- IN (*value1*, *value2*, ..., *valueN*) връща TRUE, ако стойността е в зададения списък от стойности.

```
SELECT *
FROM Students
WHERE StudentNumber IN ('17111', '17555',
'17635')
```

StudentNumber	StudentName	Address	SpecialityID
17111	Мартин Иванов Петров	ул. Христо Ботев, 12	2

- LIKE '*шаблон на символен низ*' сравнява символните низове със зададения шаблон.
  - _ – един произволен символ;

```
SELECT * FROM Students
WHERE StudentNumber LIKE '17___'
```

  - % – последователност от произволни символи;

```
SELECT * FROM Students
WHERE StudentName LIKE 'П%'
```

  - [] – един знак, намиращ се в указаната област;

```
SELECT * FROM Students
WHERE StudentName LIKE '[С-Я]%[ав]'
```

  - [^] – един знак, намиращ се извън указаната област;

```
SELECT * FROM Students
WHERE StudentName LIKE '[^С-Я]%[^ав]'
```

- LIKE '*шаблон на си*
  със зададения ша

  - _ – един произв

```
SELECT * FROM
WHERE StudentNumber LIKE '17___'
```

StudentNumber	StudentName	Address	SpecialityID
17111	Мартин Иванов Петров	ул. Христо Ботев, 12	2
17123	Иван Иванов Иванов	ул. В. Левски, 7	2
17222	Соня Стоянова Петрова	NULL	2
17654	Георги Иванов Георгиев	ул. Иван Вазов, 12	2

  - % – последовате

```
SELECT * FROM
WHERE StudentName LIKE 'П%'
```

StudentNumber	StudentName	Address	SpecialityID
15222	Петя Маринова Иванова	NULL	1
15321	Петър Иванов Петров	бул. България, 10	1

  - [] – един знак, на

```
SELECT * FROM
WHERE StudentName LIKE '[С-Я]%[ав]'
```

StudentNumber	StudentName	Address	SpecialityID
15111	Христо Иванов Иванов	ул. В. Левски, 32	1
17222	Соня Стоянова Петрова	NULL	2

  - [^] – един знак, намиращ се

```
SELECT * FROM Students
WHERE StudentName LIKE '[^С-Я]%[^ав]'
```

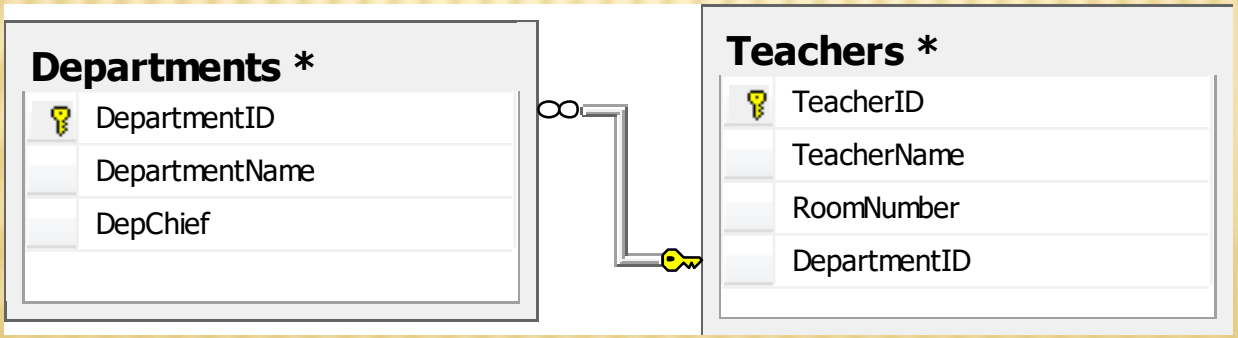
StudentNumber	StudentName	Address	SpecialityID

SQL ЗАЯВКИ – ПРИМЕРИ

- **Пример 1** Да се намерят имената на преподавателите, които са от факултет „Математика и информатика”.

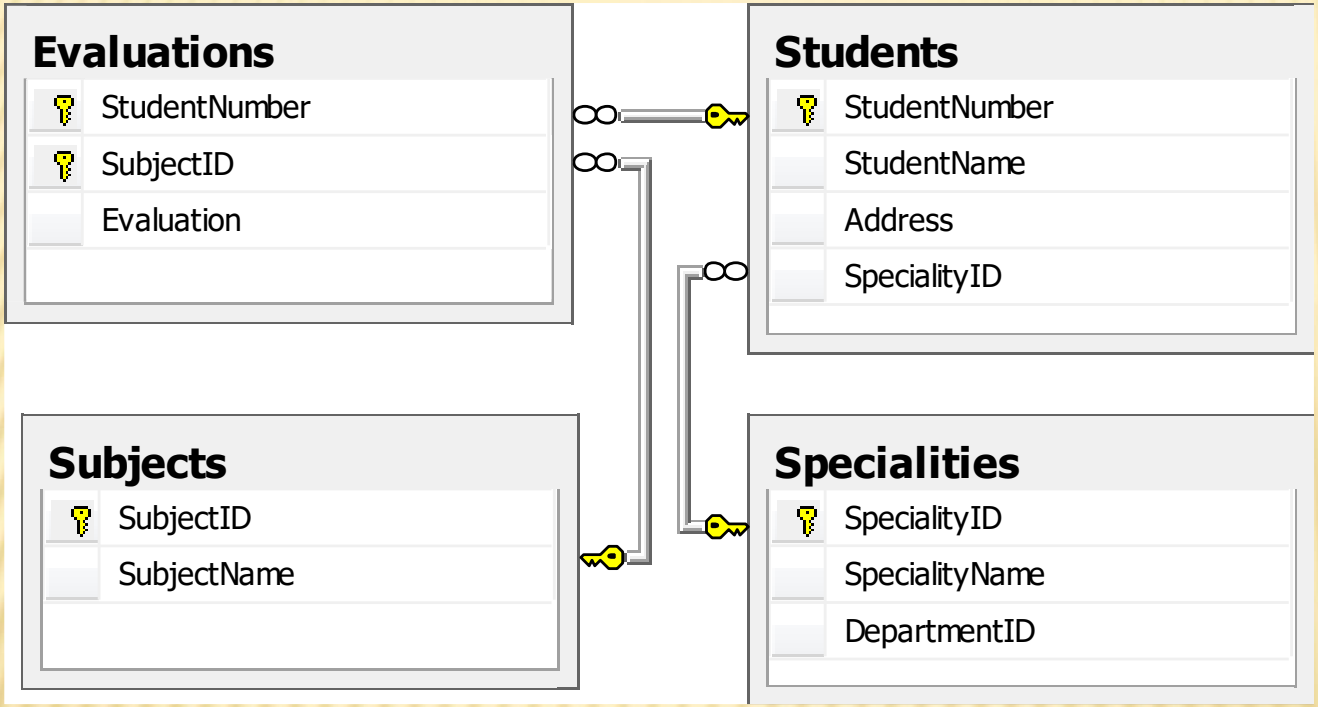
RA $\pi_{TeacherName}(\sigma_{DepartmentName='Математика\ и\ информатика'}(Teachers \triangleright\!\triangleleft Departments))$

SQL
```
SELECT TeacherName
FROM Teachers
INNER JOIN Departments
   ON Teachers.DepartmentID = Departments.DepartmentID
WHERE DepartmentName = 'Математика и информатика'
```



TeacherName
Иван Димитров
Димитър Александров
Красимир Петров

- **Пример 2** Да се намерят факултетните номера, имената и оценките на студентите от специалност „Информатика” по „Бази от данни”, сортирани по факултетни номера.



StudentNumber	StudentName	Evaluation
17111	Мартин Иванов Петров	6.00
17123	Иван Иванов Иванов	4.00
17654	Георги Иванов Георгиев	6.00

- **Пример 2** Да се намерят факултетните номера, имената и оценките на студентите от специалност „Информатика” по „Бази от данни”, сортирани по факултетни номера.

RA

$$\tau_{StudentNumber}(\pi_{StudentNumber,StudentName,Evaluation}(Students \triangleright\triangleleft Evaluations \triangleright\triangleleft \sigma_{SubjectName='Бази\ от\ данни'}(Subjects) \triangleright\triangleleft \sigma_{SpecialityName='Информатика'}(Specialities)))$$

SQL

```
SELECT Students.StudentNumber,
       StudentName, Evaluation
FROM Students
INNER JOIN Evaluations
   ON Students.StudentNumber = Evaluations.StudentNumber
INNER JOIN Subjects
   ON Evaluations.SubjectID = Subjects.SubjectID
INNER JOIN Specialities
   ON Students.SpecialityID = Specialities.SpecialityID
WHERE SubjectName = 'Бази от данни' AND
      SpecialityName = 'Информатика'
ORDER BY Students.StudentNumber ASC
```

SQL ЗАЯВКИ – ПРИМЕРИ

- **Пример 3** Да се намери средния успех на студент с факултетен номер 17123.

RA $\gamma_{AVG(Evaluation)\rightarrow Average}(\sigma_{StudentNumber='17123'}(Evaluations))$

SQL
```
SELECT AVG(Evaluation) AS Average
FROM Evaluations
WHERE StudentNumber = '17123'
```

Average
4.666666

SQL ЗАЯВКИ – ПРИМЕРИ

- **Пример 4** Да се намери броят на студентите от специалност „Информатика”.

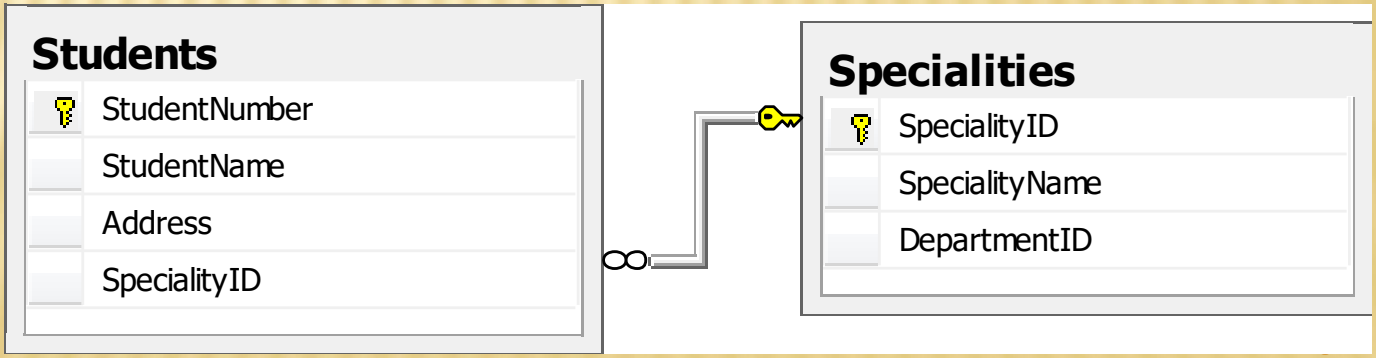
RA

$$\gamma_{COUNT(StudentNumber)\rightarrow StudentsCount}(Students \triangleright\!\triangleleft \sigma_{SpecialityName='Информатика'}(Specialities))$$

SQL

```
SELECT COUNT(StudentNumber) AS StudentsCount
FROM Students
INNER JOIN Specialities
   ON Students.SpecialityID =
      Specialities.SpecialityID
WHERE SpecialityName = 'Информатика'
```

StudentsCount
4



SQL ЗАЯВКИ – ПРИМЕРИ

- **Пример 5** Да се намерят наименованията на специалностите и да се изчисли броят на студентите от всяка специалност.

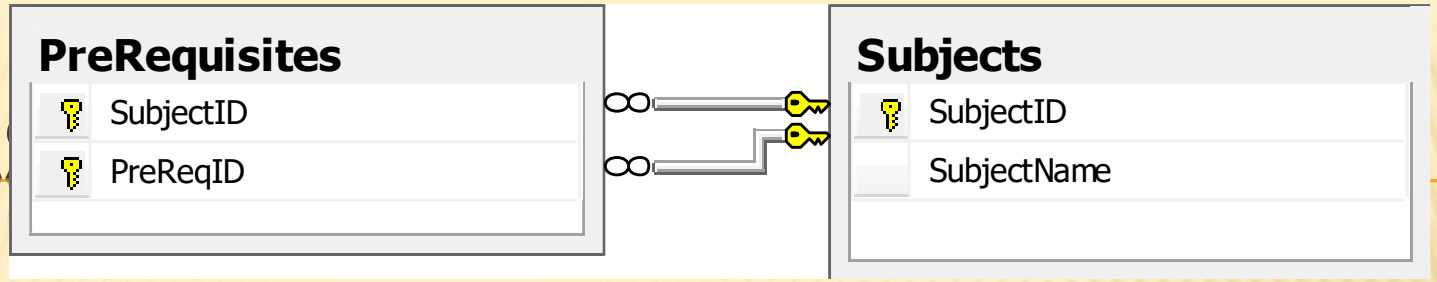
RA $\gamma_{SpecialityName,\ COUNT(StudentNumber)\rightarrow StudentsCount}(Students \triangleright\!\triangleleft Specialities)$

SQL

```
SELECT SpecialityName,
       COUNT(StudentNumber) AS StudentsCount
FROM Students
INNER JOIN Specialities
   ON Students.SpecialityID = Specialities.SpecialityID
GROUP BY SpecialityName
```

SpecialityName	StudentsCount
Информатика	4
Компютърни науки	1
Математика и информатика	3

SQL За



- **Пример 6** Да се намерят наименованията на учебните предмети, за които е необходимо предварително изучаване на „Бази от данни”.

RA

$$\pi_{Subjects.SubjectName}(Subjects \triangleright\!\triangleleft PreRequisites \underset{PreReqID=Pre.SubjectID}{\triangleright\!\triangleleft} \rho_{Pre}(\sigma_{SubjectName='Бази\ от\ данни'}(Subjects)))$$

SQL

```
SELECT Subjects.SubjectName
FROM Subjects
INNER JOIN PreRequisites
   ON Subjects.SubjectID = PreRequisites.SubjectID
INNER JOIN Subjects AS Pre
   ON PreRequisites.PreReqID = Pre.SubjectID
WHERE Pre.SubjectName = 'Бази от данни'
```

SubjectName
Информационни системи

SQL ЗАЯВКИ – ПРИМЕРИ

- **Пример 7** Да се намерят различните номера на кабинети на преподаватели.

RA | $\delta(\pi_{RoomNumber}(Teachers))$

SQL |
```
SELECT DISTINCT RoomNumber
FROM Teachers
```

RoomNumber
123
321

SQL ЗАЯВКИ – ПРИМЕРИ

- **Пример 8** Да се намерят наименованията на всички учебни предмети и наименованията на учебните предмети, които са необходими преди тяхното изучаване. В резултата да се включат и учебните предмети, които не изискват предварително изучаване на други предмети.

RA

$$\pi_{SubjectName,Pre.SubjectName}(Subjects \overset{\circ}{\triangleright\!\triangleleft}_{L} PreRequisites \overset{\circ}{\triangleright\!\triangleleft}_{PreReqID=Pre.SubjectID}{}_{L} \rho_{Pre}(Subjects))$$

SQL

```
SELECT Subjects.SubjectName, Pre.SubjectName
FROM Subjects
LEFT JOIN PreRequisites
   ON Subjects.SubjectID = PreRequisites.SubjectID
LEFT JOIN Subjects Pre
   ON PreRequisites.PreReqID = Pre.SubjectID
```

SQL ЗАЯВКИ – ПРИМЕРИ

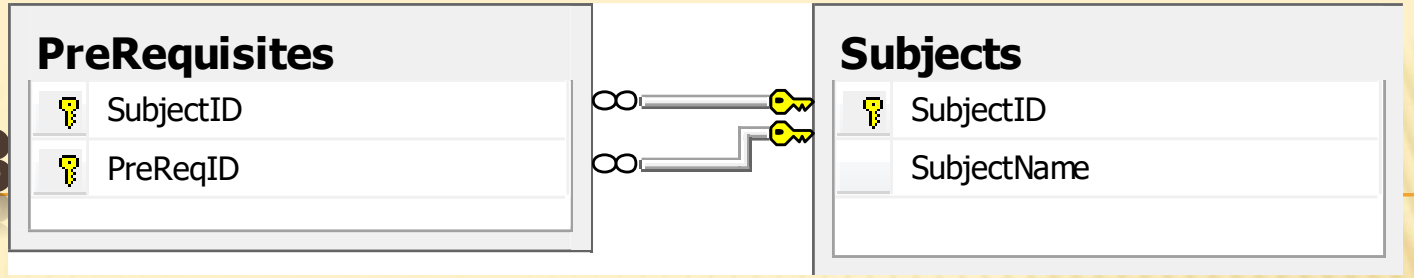
- **Пример 8** Да се намерят наи
предмети и наименованията
необходими преди тяхното из
включат и учебните предмети
предварително изучаване на

RA $\pi_{SubjectName,Pre.SubjectName}(Subjects \overset{\circ}{\triangleright\!\triangleleft}_L$)

SQL
```
SELECT Subjects.SubjectNa
FROM Subjects
LEFT JOIN PreRequisites
   ON Subjects.SubjectID
LEFT JOIN Subjects Pre
   ON PreRequisites.PreRe
```

SubjectName	SubjectName
DHTML	NULL
XML	NULL
Аналитична геометрия	NULL
Бази от данни	Програмиране I
Бази от данни	Програмиране II
Дискретна математика	NULL
Информационни системи	Програмиране I
Информационни системи	Бази от данни
Кодиране	Програмиране I
Кодиране	Линейна алгебра
Кодиране	Аналитична геометрия
Компютърна графика	Линейна алгебра
Компютърна графика	Аналитична геометрия
Компютърна графика	Дискретна математика
Линейна алгебра	NULL
Програмиране I	Програмиране II
Програмиране II	NULL
Теория на вероятностите	NULL

SQL ЗАЯВ



- **Пример 9** Да се намерят наименованията на учебните предмети, които не изискват предварително изучаване на други учебни предмети.

RA | $\pi_{SubjectName}(Subjects) \setminus \pi_{SubjectName}(Sub$ ... $tes)$

SubjectName
DHTML
XML
Аналитична геометрия
Дискретна математика
Линейна алгебра
Програмиране II
Теория на вероятностите

SQL

```
SELECT SubjectName
FROM Subjects
EXCEPT
SELECT SubjectName
FROM Subjects
INNER JOIN PreRequisites
   ON Subjects.SubjectID = PreRequisites.SubjectID
```

SQL ЗАЯВКИ – ПРИМЕРИ

- Пример 9 – втори начин

RA | $\pi_{SubjectName}(\sigma_{SubjectID\ NOT\ IN\ \pi_{SubjectID}(PreRequisites)}(Subjects))$

SQL |
```
SELECT SubjectName
FROM Subjects
WHERE SubjectID NOT IN
    (SELECT SubjectID
     FROM PreRequisites)
```

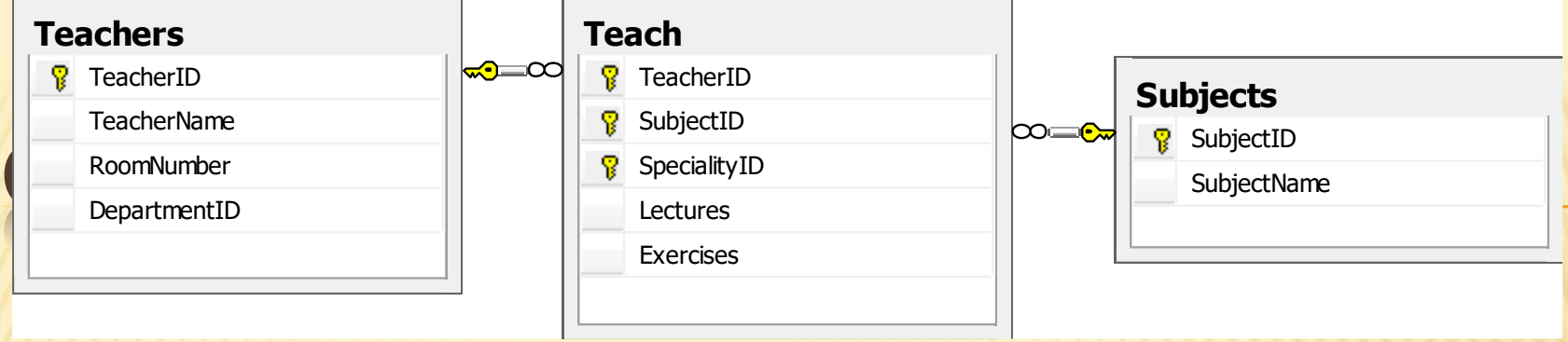
SQL ЗАЯВКИ – ПРИМЕРИ

- Пример 9 – трети начин

RA | $\pi_{SubjectName}(\sigma_{p.SubjectID\ IS\ NULL}(\rho_s(Subjects) \circ \triangleright\triangleleft_L \rho_p(PreRequisites)))$

SQL

```
SELECT SubjectName
FROM Subjects AS s
LEFT OUTER JOIN PreRequisites AS p
  ON s.SubjectID = p.SubjectID
WHERE p.SubjectID IS NULL
```



S

- **Пример 10** Да се намерят имената на преподавателите, които преподават „Бази от данни” или „Информационни системи”.

RA

$$\delta(\pi_{TeacherName}(Teachers \triangleright\triangleleft Teach \triangleright\triangleleft \sigma_{SubjectName='Бази\ от\ данни'\ OR\ SubjectName='Информационни\ системи'}(Subjects)))$$

SQL

```
SELECT DISTINCT TeacherName
FROM Teachers
INNER JOIN Teach
  ON Teachers.TeacherID = Teach.TeacherID
INNER JOIN Subjects
  ON Teach.SubjectID = Subjects.SubjectID
WHERE SubjectName = 'Бази от данни'
   OR SubjectName = 'Информационни системи'
```

TeacherName
Красимир Петров

Пример 10 – втори начин

RA

$$\pi_{TeacherName}(Teachers \rhd\lhd Teach \rhd\lhd \sigma_{SubjectName='Бази\ от\ данни'}(Subjects)) \cup \pi_{TeacherName}(Teachers \rhd\lhd Teach \rhd\lhd \sigma_{SubjectName='Информационни\ системи'}(Subjects))$$

SQL

```
SELECT TeacherName
FROM Teachers
INNER JOIN Teach
  ON Teachers.TeacherID = Teach.TeacherID
INNER JOIN Subjects
  ON Teach.SubjectID = Subjects.SubjectID
WHERE SubjectName = 'Бази от данни'
UNION
SELECT TeacherName
FROM Teachers
INNER JOIN Teach
  ON Teachers.TeacherID = Teach.TeacherID
INNER JOIN Subjects
  ON Teach.SubjectID = Subjects.SubjectID
WHERE SubjectName = 'Информационни системи'
```

- **Пример 11** Да се намерят имената на преподавателите, които преподават „Бази от данни” и „Информационни системи”.

RA

$$\pi_{TeacherName}(Teachers \triangleright\triangleleft Teach \triangleright\triangleleft \sigma_{SubjectName='Бази\ от\ данни'}(Subjects)) \cap \pi_{TeacherName}(Teachers \triangleright\triangleleft Teach \triangleright\triangleleft \sigma_{SubjectName='Информационни\ системи'}(Subjects))$$

SQL

```
SELECT TeacherName
FROM Teachers
INNER JOIN Teach
  ON Teachers.TeacherID = Teach.TeacherID
INNER JOIN Subjects
  ON Teach.SubjectID = Subjects.SubjectID
WHERE SubjectName = 'Бази от данни'
INTERSECT
SELECT TeacherName
FROM Teachers
INNER JOIN Teach
  ON Teachers.TeacherID = Teach.TeacherID
INNER JOIN Subjects
  ON Teach.SubjectID = Subjects.SubjectID
WHERE SubjectName = 'Информационни системи'
```

TeacherName

- **Пример 12** Да се намерят наименованията на учебните предмети и оценките на студент с факултетен номер 17123, но само за тези предмети, за които студентът е получил оценка, не по-малка от средния му успех.

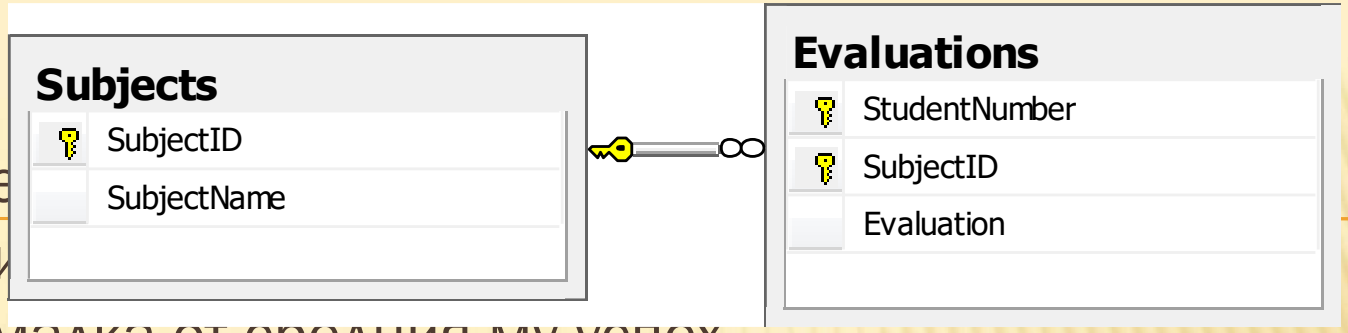
RA

$$\pi_{SubjectName,Evaluation}(Subjects \rhd\lhd \sigma_{StudentNumber='17123'\ AND\ Evaluation \geq \gamma_{AVG(Evaluation)}(\sigma_{StudentNumber='17123'}(Evaluations))}(Evaluations))$$

SQL

```
SELECT SubjectName, Evaluation
FROM Subjects
INNER JOIN Evaluations
   ON Subjects.SubjectID = Evaluations.SubjectID
WHERE StudentNumber = '17123'
  AND Evaluation >=
      (SELECT AVG(Evaluation)
       FROM Evaluations
       WHERE StudentNumber = '17123')
```

- **Пример 12** Да
предмети и оце
но само за тези
оценка, не по-малка от средния му успех.



RA

$$\pi_{SubjectName,Evaluation}(Subjects \triangleright\triangleleft$$

$$\sigma_{StudentNumber='17123'\ AND\ Evaluation \geq \gamma_{AVG(Evaluation)}(\sigma_{Studer}$$

SubjectName	Evaluation
Програмиране I	5.00
Информационни системи	6.00
XML	5.00
DHTML	5.00

SQL

```
SELECT SubjectName, Evaluation
FROM Subjects
INNER JOIN Evaluations
   ON Subjects.SubjectID = Evaluations.SubjectID
WHERE StudentNumber = '17123'
  AND Evaluation >=
      (SELECT AVG(Evaluation)
       FROM Evaluations
       WHERE StudentNumber = '17123')
```

- **Пример 13** Да се намерят факултетните номера и имената на студентите, наименованията на учебните предмети и оценките, но само за тези предмети и студенти, за които получената оценка е не по-малка от средния успех на съответния студент.

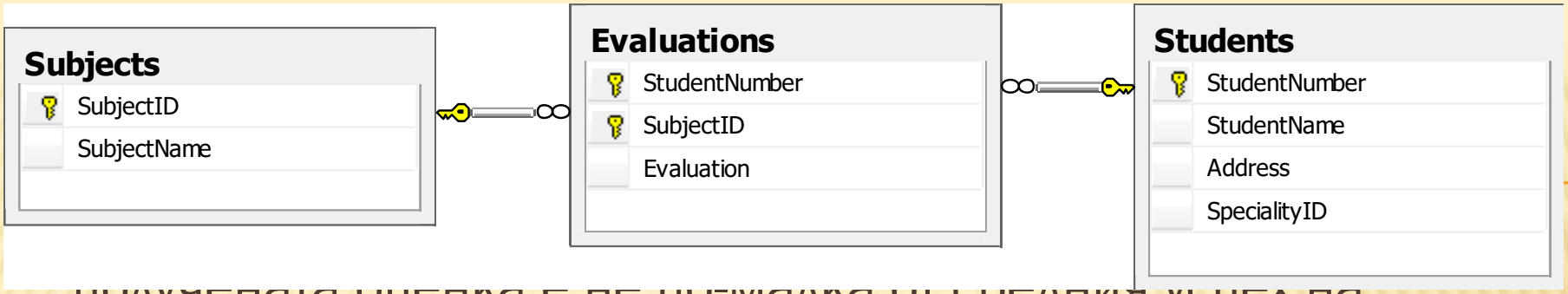
RA

$$\pi_{st.StudentNumber,st.StudentName,sb.SubjectName,e.Evaluation}(\rho_{st}(Students) \triangleright\triangleleft$$

$$\sigma_{Evaluation \geq \gamma_{AVG(Evaluation)}(\sigma_{e1.StudentNumber=st.StudentNumber}(\rho_{e1}(Evaluations)))}(\rho_{e}(Evaluations)) \triangleright\triangleleft \rho_{sb}(Subjects))$$

SQL

```
SELECT st.StudentNumber, st.StudentName,
       sb.SubjectName, e.Evaluation
FROM Subjects sb
INNER JOIN Evaluations e
   ON sb.SubjectID = e.SubjectID
INNER JOIN Students st
   ON e.StudentNumber = st.StudentNumber
WHERE e.Evaluation >=
     (SELECT AVG(e1.Evaluation)
      FROM Evaluations e1
      WHERE e1.StudentNumber = st.StudentNumber)
```



получената оценка е не по-малка от средния успех на съответния сту

RA

$$\pi_{st.StudentNumber,st.Stud}$$

$$\sigma_{Evaluation \geq \gamma_{AVG(Evaluation)}(\sigma_{e}}$$

SQL

```
SELECT st.Stu
       sb.Sub
FROM Subjects
INNER JOIN Ev
   ON sb.Subj
INNER JOIN St
   ON e.Stude
WHERE e.Evalu
      (SELECT
       FROM Ev
       WHERE e
```

StudentNumber	StudentName	SubjectName	Evaluation
13987	Иван Петров Георгиев	Програмиране I	6.00
13987	Иван Петров Георгиев	Бази от данни	5.00
13987	Иван Петров Георгиев	Информационни системи	5.00
13987	Иван Петров Георгиев	XML	6.00
15111	Христо Иванов Иванов	Програмиране II	5.00
15321	Петър Иванов Петров	Програмиране II	6.00
17111	Мартин Иванов Петров	Програмиране I	5.00
17111	Мартин Иванов Петров	Бази от данни	6.00
17111	Мартин Иванов Петров	XML	6.00
17123	Иван Иванов Иванов	Програмиране I	5.00
17123	Иван Иванов Иванов	Информационни системи	6.00
17123	Иван Иванов Иванов	XML	5.00
17123	Иван Иванов Иванов	DHTML	5.00
17654	Георги Иванов Георгиев	Програмиране I	6.00
17654	Георги Иванов Георгиев	Бази от данни	6.00
17654	Георги Иванов Георгиев	XML	6.00

ДЕФИНИРАНЕ НА ДАННИ СЪС SQL

- *Език за дефиниране на данни* (*Data Definition Language – DDL*) – създаване, променяне и изтриване на схемите на съхраняваните в базата от данни релации.
- Типове данни
  - символни низове с постоянна или променлива дължина – char(*n*), varchar(*n*);
  - двоични низове с постоянна или променлива дължина – binary(*n*), varbinary(*n*);
  - логически стойности – boolean, bit;
  - целочислени стойности – int (integer), shortint (smallint), др.;
  - числови стойности с плаваща запетая – float, real;
  - стойности за дати и часове – date, time, datetime.

ИЗПОЛЗВАНЕ НА SQL ЗА ДЕФИНИРАНЕ НА ДАННИ

- Създаване на таблица

```
CREATE TABLE table_name
( { col_name datatype [null_option] }
    [, …] )
```

- Пример

```
CREATE TABLE Students
( StudentNumber varchar(10) NOT NULL,
  FirstName varchar(25) NOT NULL,
  MiddleInitial char(1) NOT NULL,
  LastName varchar(25) NOT NULL,
  BirthDate datetime NULL )
```

ИЗПОЛЗВАНЕ НА SQL ЗА ДЕФИНИРАНЕ НА ДАННИ (2)

- Променяне на вече създадена таблица

```
ALTER TABLE table_name
{
 ALTER COLUMN col_name new_datatype
  [null_option]
 | ADD { col_name datatype [null_option] } [,
  …]
 | DROP { COLUMN col_name } [, …]
}
```

ПРОМЕНЯНЕ НА ВЕЧЕ СЪЗДАДЕНА ТАБЛИЦА – ПРИМЕРИ

- Добавяне на колона в таблицата *Students*:

```
ALTER TABLE Students
      ADD EGN char(10) NULL
```

- Променяне на съществуващата колона *MiddleInitial*, така че да може да приема стойност NULL:

```
ALTER TABLE Students
      ALTER COLUMN MiddleInitial char(1) NULL
```

- Изтриване на съществуващата колона *BirthDate*:

```
ALTER TABLE Students
      DROP COLUMN BirthDate
```

ИЗПОЛЗВАНЕ НА SQL ЗА ДЕФИНИРАНЕ НА ДАННИ

- Изтриване на таблица

```
DROP TABLE table_name
```

- Пример

```
DROP TABLE Students
```

ОГРАНИЧАВАНЕ НА СТОЙНОСТИТЕ НА ДАННИТЕ

- При създаване на таблица :

```
CREATE TABLE table_name
(
 {col_name datatype [null_option]
  [col_constraint[,…]]
 |table_constraint
 } [, …]
)
```

ОГРАНИЧАВАНЕ НА СТОЙНОСТИТЕ НА ДАННИТЕ (2)

- При променяне на таблица :

```
ALTER TABLE table_name
{   ALTER COLUMN col_name new_datatype
   [null_option]
  | ADD {col_name datatype [null_option]
                          [col_constraint [,…]]
        | table_constraint
        } [, …]
  | DROP {COLUMN col_name |
         [CONSTRAINT] constraint_name}[,…]
}
```

ОГРАНИЧЕНИЕ ПЪРВИЧЕН КЛЮЧ

```
[CONSTRAINT pk_constraint_name]
PRIMARY KEY
[(col_name1 [, col_name2 [, …, col_nameN]])]
```

- **Пример** Създаване на таблица *Students* с ограничение първичен ключ върху колоната *StudentNumber*:

```
CREATE TABLE Students
( StudentNumber varchar(10) NOT NULL PRIMARY KEY,
  StudentName varchar(50) NOT NULL,
  Address varchar(100) NULL,
  BirthDate datetime NULL )
```

ОГРАНИЧЕНИЕ ПЪРВИЧЕН КЛЮЧ – ПРИМЕРИ

- **Пример** Добавяне на ограничение първичен ключ върху колоната *StudentNumber*:

```
ALTER TABLE Students
ADD CONSTRAINT PK_Students
    PRIMARY KEY (StudentNumber)
```

- **Пример** Създаване на таблица *Evaluations* с първичен ключ *PK_Evaluations*, съставен от две колони *StudentNumber* и *SubjectID*:

```
CREATE TABLE Evaluations
( StudentNumber varchar(10) NOT NULL,
  SubjectID int NOT NULL,
  Evaluation decimal(3,2) NOT NULL,
    CONSTRAINT PK_Evaluations
    PRIMARY KEY (StudentNumber, SubjectID) )
```

ОГРАНИЧЕНИЯ ЗА УНИКАЛНОСТ

```
[CONSTRAINT unique_constraint_name]
UNIQUE
[(col_name1 [, col_name2 [, …, col_nameN]])]
```

- Примери

```
CREATE TABLE Students
( StudentNumber varchar(10) NOT NULL PRIMARY KEY,
  StudentName varchar(50) NOT NULL,
  Address varchar(100) NULL,
  BirthDate datetime NULL,
  EGN char(10) NULL UNIQUE )
```

```
ALTER TABLE Students
ADD EGN char(10) NULL
    CONSTRAINT unique_egn UNIQUE (EGN)
```

ОГРАНИЧЕНИЕ ВЪНШЕН КЛЮЧ

```
[CONSTRAINT fk_constraint_name]
[FOREIGN KEY (col_name1 [, col_name2 [, …,
  col_nameN]])]
REFERENCES referenced_table_name
(ref_col_name1 [, ref_col_name2 [, …,
  ref_col_nameN]])
[ON DELETE { NO ACTION | CASCADE | SET NULL |
  SET DEFAULT }]
[ON UPDATE { NO ACTION | CASCADE | SET NULL |
  SET DEFAULT }]
```

ОГРАНИЧЕНИЕ ВЪНШЕН КЛЮЧ – ПРИМЕРИ

```
CREATE TABLE Evaluations
(
  StudentNumber varchar(10) NOT NULL
        REFERENCES Students (StudentNumber),
  SubjectID int NOT NULL
        REFERENCES Subjects (SubjectID),
  Evaluation decimal(3,2) NOT NULL,
    CONSTRAINT PK_Evaluations
    PRIMARY KEY (StudentNumber, SubjectID)
)
```

ОГРАНИЧЕНИЕ ВЪНШЕН КЛЮЧ – ПРИМЕРИ (2)

```
CREATE TABLE Evaluations
( StudentNumber varchar(10) NOT NULL,
  SubjectID int NOT NULL,
  Evaluation decimal(3,2) NOT NULL,
    CONSTRAINT PK_Evaluations
    PRIMARY KEY (StudentNumber, SubjectID) )

ALTER TABLE Evaluations
ADD CONSTRAINT FK_Evaluation_Student
    FOREIGN KEY (StudentNumber)
    REFERENCES Students (StudentNumber)
      ON DELETE CASCADE
      ON UPDATE CASCADE

ALTER TABLE Evaluations
ADD CONSTRAINT FK_Evaluation_Subject
    FOREIGN KEY (SubjectID)
    REFERENCES Subjects (SubjectID)
```

ОГРАНИЧЕНИЯ ЗА ВАЛИДНОСТ НА ДАННИТЕ

```
[CONSTRAINT check_constraint_name]
CHECK (check_expression)
```

- Пример

```
CREATE TABLE Evaluations
( StudentNumber varchar(10) NOT NULL
    REFERENCES Students (StudentNumber),
  SubjectID int NOT NULL
    REFERENCES Subjects (SubjectID),
  Evaluation decimal(3,2) NOT NULL
    CHECK (Evaluation BETWEEN 2 AND 6),
  PRIMARY KEY (StudentNumber, SubjectID)  )
```

ОГРАНИЧЕНИЯ ПО ПОДРАЗБИРАНЕ

```
[CONSTRAINT default_constraint_name]
DEFAULT default_value
[FOR column_name]
```

- Пример

```
CREATE TABLE Students
( StudentNumber varchar(10) NOT NULL PRIMARY KEY,
  StudentName varchar(50) NOT NULL,
  Address varchar(200) NULL DEFAULT 'неизвестен
  адрес',
  SpecialityID int NOT NULL
    REFERENCES Specialities (SpecialityID) )
```

ИНДЕКСИ

- *Индекс* (*index*) на атрибута *a* на дадена релация *R*
  - структура от данни, която позволява да се повиши ефективността на търсенето на конкретна стойност, съхранявана в атрибута *a* и сортирането по неговите стойности.

```
CREATE [UNIQUE] INDEX index_name
 ON table_name
 ( column_name1 [, column_name2 [,…,
  column_nameN]] )
```

ИНДЕКСИ – ПРИМЕРИ

- **Пример** Създаване на уникален индекс върху колоната *DepartmentName* в таблицата *Departments*:

```
CREATE UNIQUE INDEX DepartmentNameInd
    ON Departments ( DepartmentName )
```

- **Пример** Създаване на индекс върху колоната *RoomNumber* в таблицата *Teachers*:

```
CREATE INDEX RoomNumberInd
    ON Teachers ( RoomNumber )
```

МОДИФИЦИРАНЕ НА ДАННИ СЪС SQL

- Добавяне на нови редове в таблица (INSERT)
- Обновяване на стойности на редове в таблица (UPDATE)
- Изтриване на редове в таблица (DELETE)

ДОБАВЯНЕ НА НОВИ РЕДОВЕ В ТАБЛИЦА

```
INSERT [INTO] table_name [(column_list)]
VALUES (value_list)
```

- Пример

```
INSERT INTO Students
  (StudentNumber, StudentName,
   Address, SpecialityID)
VALUES
   ('17123', 'Иван Иванов Иванов',
    'ул. В. Левски, 7', 2)
```

ДОБАВЯНЕ НА НОВИ РЕДОВЕ В ТАБЛИЦА

```
INSERT [INTO] table_name [(column_list)]
SELECT column_list
FROM other_table_name
...
```

- Пример

```
CREATE TABLE StudentsAverage
( StudentNumber varchar(10) NOT NULL PRIMARY KEY,
  Average decimal (3, 2) NOT NULL )

INSERT INTO StudentsAverage
SELECT StudentNumber, AVG(Evaluation)
FROM Evaluations
GROUP BY StudentNumber
```

ДОБАВЯНЕ НА НОВИ РЕДОВЕ В ТАБЛИЦА (2)

```
SELECT column_list
INTO new_table_name
FROM other_table_name
…
```

- Пример

```
SELECT StudentNumber,
       AVG(Evaluation) AS Average
INTO StudentsAverage
FROM Evaluations
GROUP BY StudentNumber
```

ОБНОВЯВАНЕ НА СТОЙНОСТИ НА РЕДОВЕ В ТАБЛИЦА

```
UPDATE table_name
    SET column_name1 = value1
       [, column_name2 = value2
       [,…, column_nameN = valueN ]]
  [WHERE search_conditions]
```

- Пример

```
UPDATE Subjects
    SET SubjectName = 'Програмиране - I част'
WHERE SubjectName = 'Увод в програмирането'
```

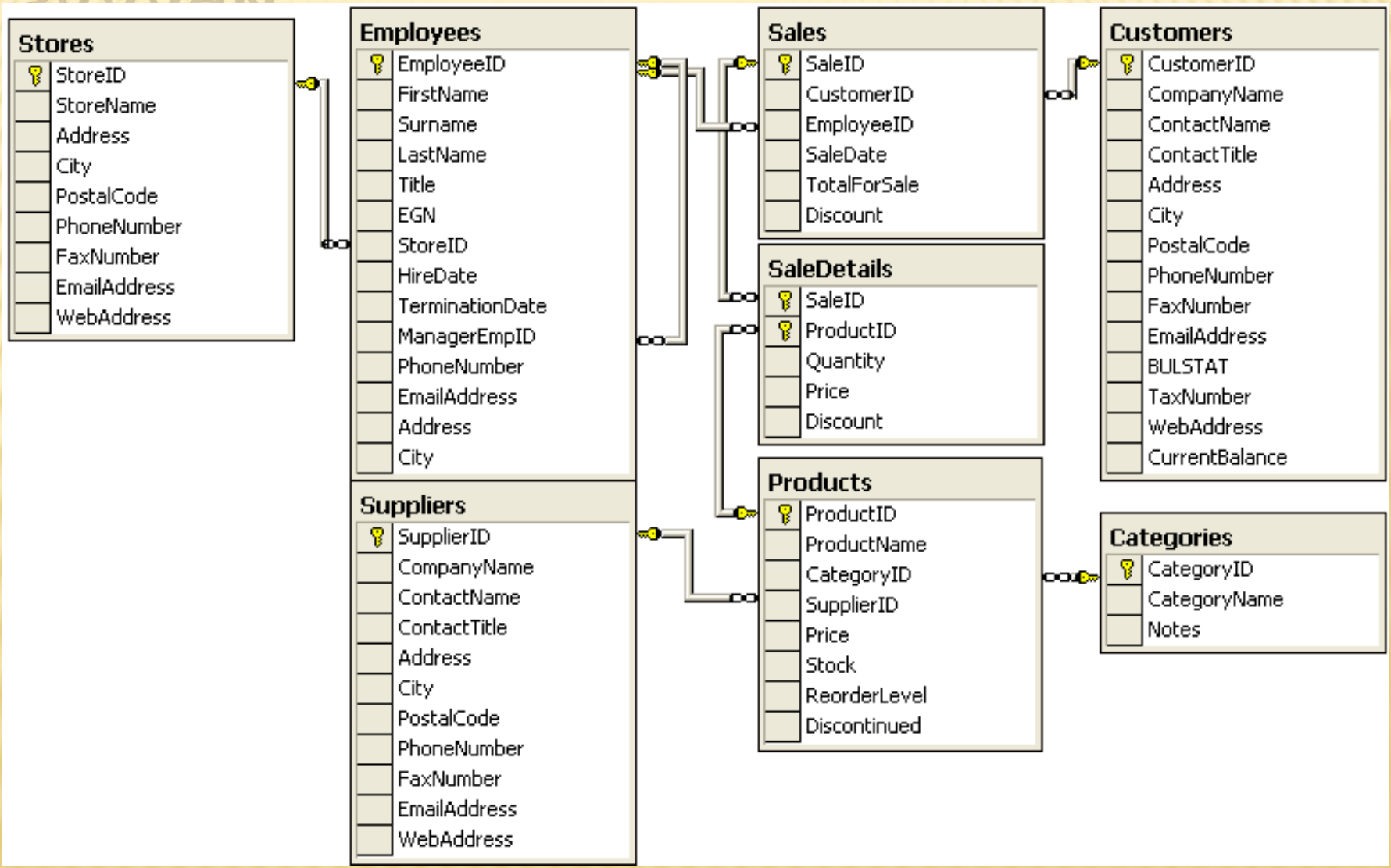
ИЗТРИВАНЕ НА РЕДОВЕ В ТАБЛИЦА

```
DELETE FROM table_name
[WHERE search_conditions]
```

- Пример

```
DELETE FROM Evaluations
WHERE StudentNumber = '17123'
```

ЗАДАЧИ


Stores
StoreID
StoreName
Address
City
PostalCode
PhoneNumber
FaxNumber
EmailAddress
WebAddress
Employees
EmployeeID
FirstName
Surname
LastName
Title
EGN
StoreID
HireDate
TerminationDate
ManagerEmpID
PhoneNumber
EmailAddress
Address
City
Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
PostalCode
PhoneNumber
FaxNumber
EmailAddress
WebAddress
Sales
SaleID
CustomerID
EmployeeID
SaleDate
TotalForSale
Discount
SaleDetails
SaleID
ProductID
Quantity
Price
Discount
Products
ProductID
ProductName
CategoryID
SupplierID
Price
Stock
ReorderLevel
Discontinued
Customers
CustomerID
CompanyName
ContactName
ContactTitle
Address
City
PostalCode
PhoneNumber
FaxNumber
EmailAddress
BULSTAT
TaxNumber
WebAddress
CurrentBalance
Categories
CategoryID
CategoryName
Notes

ЗАДАЧИ

- *1*. Да се напише заявка, която извежда всички колони от таблицата за продажбите, направени от служителите с идентификатори 2, 5, 12 или 90 в периода между 01/10/2003 и 31/12/2003.
- *2*. Да се напише заявка, която извежда данните за продуктите, които имат критично налично количество и продажбата им не е преустановена.
- *3*. Да се напише заявка, която извежда данните за служителите, родени през 1960 година.
- *4*. Да се напише заявка, която извежда данните за служителите, чиито фамилии започват с някоя от буквите A, Y, D÷I или не започват с буквите B, L, M, R÷Z.
- *5*. Да се напише заявка, която извежда данните за клиентите, чиито имена (*CompanyName*) не съдържат цифри.

ЗАДАЧИ

- *6*. Да се напише заявка, която извежда данните за продуктите, които имат имена, състоящи се само от буквите а, б, в, д, и, о, п÷я или интервал (т.е. имената на продуктите да са съставени от всички или част от изброените символи).
- 7. Да се зададе ограничение за валидност:
  - ***7.1.*** на колоната *EGN* в таблицата *Employees*, с което да се гарантира въвеждането на точно 10 цифри;
  - ***7.2.*** на колоната *PhoneNumber* (в таблиците *Employees*, *Stores*, *Suppliers*, *Customers*), с което да се гарантира въвеждане на произволна комбинация от цифри, тирета и интервали.

ЗАДАЧИ (АГРЕГАТНИ ФУНКЦИИ)

- *1*. Да се напише заявка, която извежда броя на продуктите, чиято продажба е преустановена.
- *2*. Да се напише заявка, която извежда идентификатор на продукт и средната продажна цена на съответния продукт.
- *3*. Да се напише заявка, която извежда идентификатор на служител и броя на продажбите, извършените от съответния служител за периода от време между 01 март 2004 г. и 01 април 2004 г. Резултатният набор от редове да бъде сортиран по броя на продажбите в низходящ ред.
- *4*. Да се напише заявка, която извежда броя на различните градове, в които има магазини.
- *5*. Да се напише заявка, която извежда първия клиент по азбучен ред, чието име започва с Л.
- *6*. Да се напише заявка, която извежда продажбата с най-малка стойност за всеки служител.

ЗАДАЧИ (АГРЕГАТНИ ФУНКЦИИ)

- 7. Да се напише заявка, която извежда идентификатор на продукт и сума от продаденото количество за съответния продукт.
- 8. Да се напише заявка, която извежда идентификатор на продукт, най-ниската и най-високата продажна цена на съответния продукт, както и разликата на двете цени.
- 9. Да се напише заявка, която извежда идентификаторите на клиентите и средните суми на общите стойности на покупките на съответните клиенти, по-малки от 50. За получаване на обобщения резултат да се използват само продажбите, направени от служители с идентификатори 2, 5, 56 или 187. Резултатният набор от редове да бъде сортиран по средната сума на покупките в низходящ ред.