Язык SQL (DML)

Язык DML включает оператор выборки данных (**SELECT**) и операторы модификации данных (**INSERT, UPDATE, DELETE**).

Для отбора строк и столбцов таблиц базы данных используется инструкция ***SELECT***. **Синтаксис:**

SELECT [**ALL|DISTINCT**] *набор_атрибутов*
FROM *набор_отношений*
[***WHERE*** *условие_отбора_строк*]
[***GROUP BY*** *спецификация_группировки*]
[***HAVING*** *спецификация_выбора_групп*]
[ORDER BY *спецификация_сортировки*]

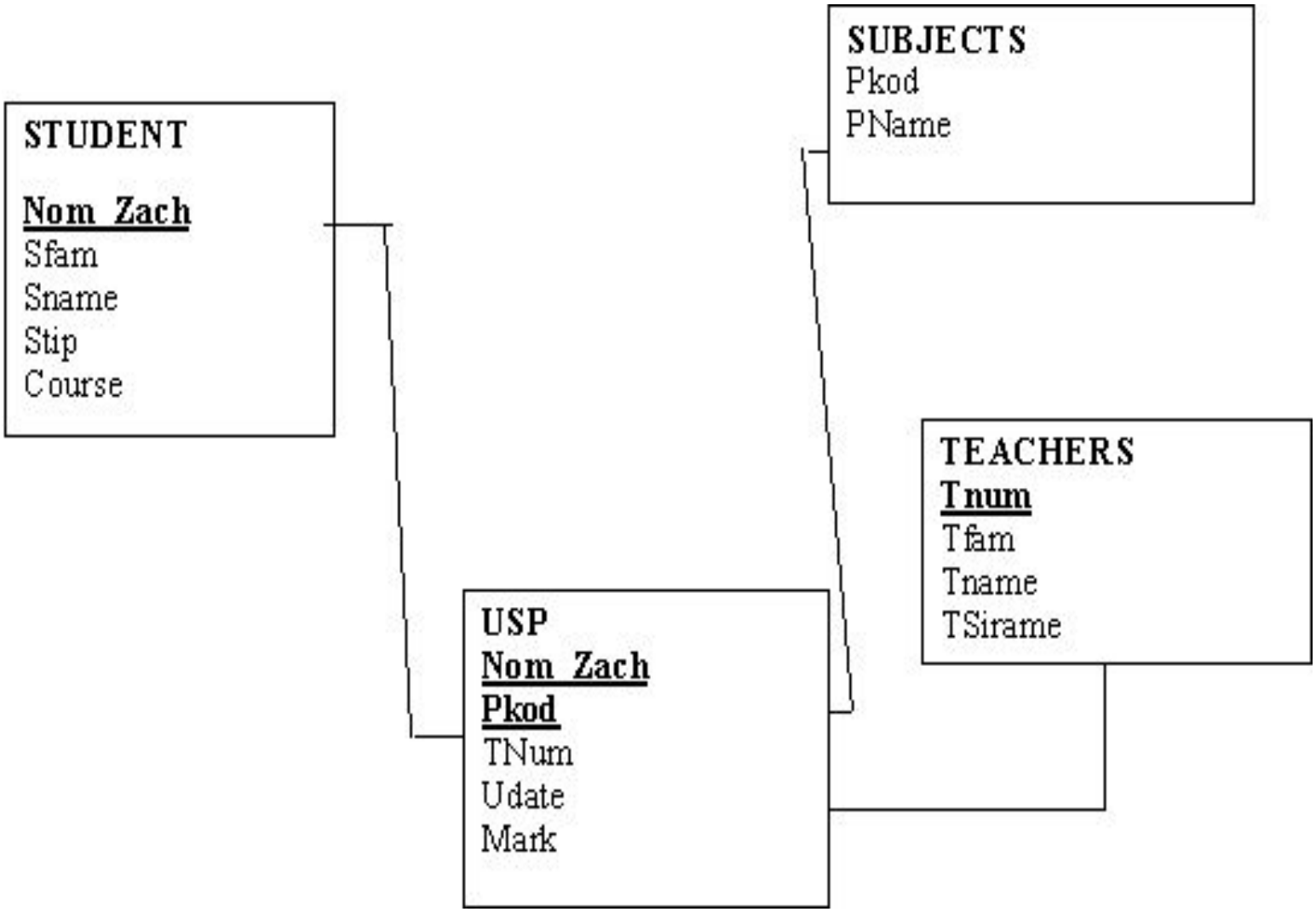
Ключевое слово ***ALL*** - в результирующий набор строк включаются все строки, удовлетворяющие условиям запроса, в том числе и строки-дубликаты.

Ключевое слово ***DISTINCT*** – в результирующий запрос включаются только различные строки.

В разделе ***SELECT*** атрибуты могут указываться с помощью (*).

Например **X.*** обозначает совокупность всех атрибутов отношения X,

изолированная * – совокупность всех атрибутов всех отношений, фигурирующих в разделе ***FROM*** для создания запроса.


SUBJECTS
Pkod
PName
STUDENT
Nom_Zach
Sfam
Sname
Stip
Course
TEACHERS
Tnum
Tfam
Tname
TSirame
USP
Nom_Zach
Pkod
TNum
Udate
Mark

SELECT * FROM STUDENT

Таблицам могут быть присвоены имена – псевдонимы, что бывает полезно при соединении таблицы с самой собою или для доступа из вложенного подзапроса к текущей записи внешнего запроса. Псевдонимы задаются с помощью ключевого слова ***AS***, которое может быть опущено.

SELECT * FROM STUDENT S

Раздел **FROM**

Раздел **FROM** определяет **таблицы** или **запросы**, служащие источником данных. В случае если указано более одного имени таблицы, по умолчанию предполагается, что над перечисленными таблицами будет выполнена операция декартова произведения. Например, запрос

SELECT * FROM STUDENT, USP

соответствует декартову произведению отношений ***STUDENT*** и ***USP***.

Для задания типа соединения таблиц в единый набор записей, из которого будет выбираться необходимая информация, в разделе **FROM** используются ключевые слова ***JOIN*** и ***ON***.

Ключевое слово ***JOIN*** и его параметры указывают соединяемые таблицы и методы соединения.

Ключевое слово ***ON*** указывает общие для таблиц поля.

При внутреннем соединении таблиц (***INNER JOIN)*** сравниваются значения общих полей этих таблиц. В окончательный набор возвращаются*я записи, у которых эти значения совпадают.*

```
SELECT *
FROM STUDENTS INNER JOIN USP
 ON
TUDENTS.NOM_ZACH=USP.NOM_ZACH
```

```
SELECT *

FROM STUDENTS,  USP

WHERE STUDENTS.NOM_ZACH=

USP.NOM_ZACH
```

Операция ***LEFT JOIN*** возвращает все строки из первой таблицы, соединённые с теми строками второй, для которых выполняется условие соединения.

Если во второй таблице таких строк нет, возвращаются ***NULL*** значения для атрибутов второй таблицы.

Операция ***RIGHT JOIN*** возвращает все строки второй таблицы, соединённые с теми строками первой, для которых выполняется условие соединения.

Операции ***LEFT JOIN*** или ***RIGHT JOIN*** могут быть вложены в операцию ***INNER JOIN***, но операция ***INNER JOIN*** не может быть вложена в операцию ***LEFT JOIN*** или ***RIGHT JOIN***.

Телефоны

Номер	Владелец	Адрес
61-32-72	Степанова	Чкалова 18-18
55-55-55	Иванов	Чкалова 15-15

Звонки

Номер	Дата	Город	Продолжит.
61-32-72	15.11.2002	Москва	5
61-32-72	12.11.2002	Токио	35

После **применения** к отношениям операции соединения будет получено отношение

Номер	Владелец	Адрес	Дата	Город	Продолж.
61-32-72	Степанова	Чкалова 18-18	15.11.2002	Москва	5
61-32-72	Степанова	Чкалова 18-18	12.11.2000	Токио	35
55-55-55	Иванов	Чкалова 15-15	NULL	NULL	NULL

```
SELECT Телефоны.*,Звонки.Дата,
Звонки.Город, Звонки.
Продолжительность
FROM Телефоны LEFT JOIN Звонки
ON Телефоны.Номер_телефона
= Звонки.Номер_телефона
```

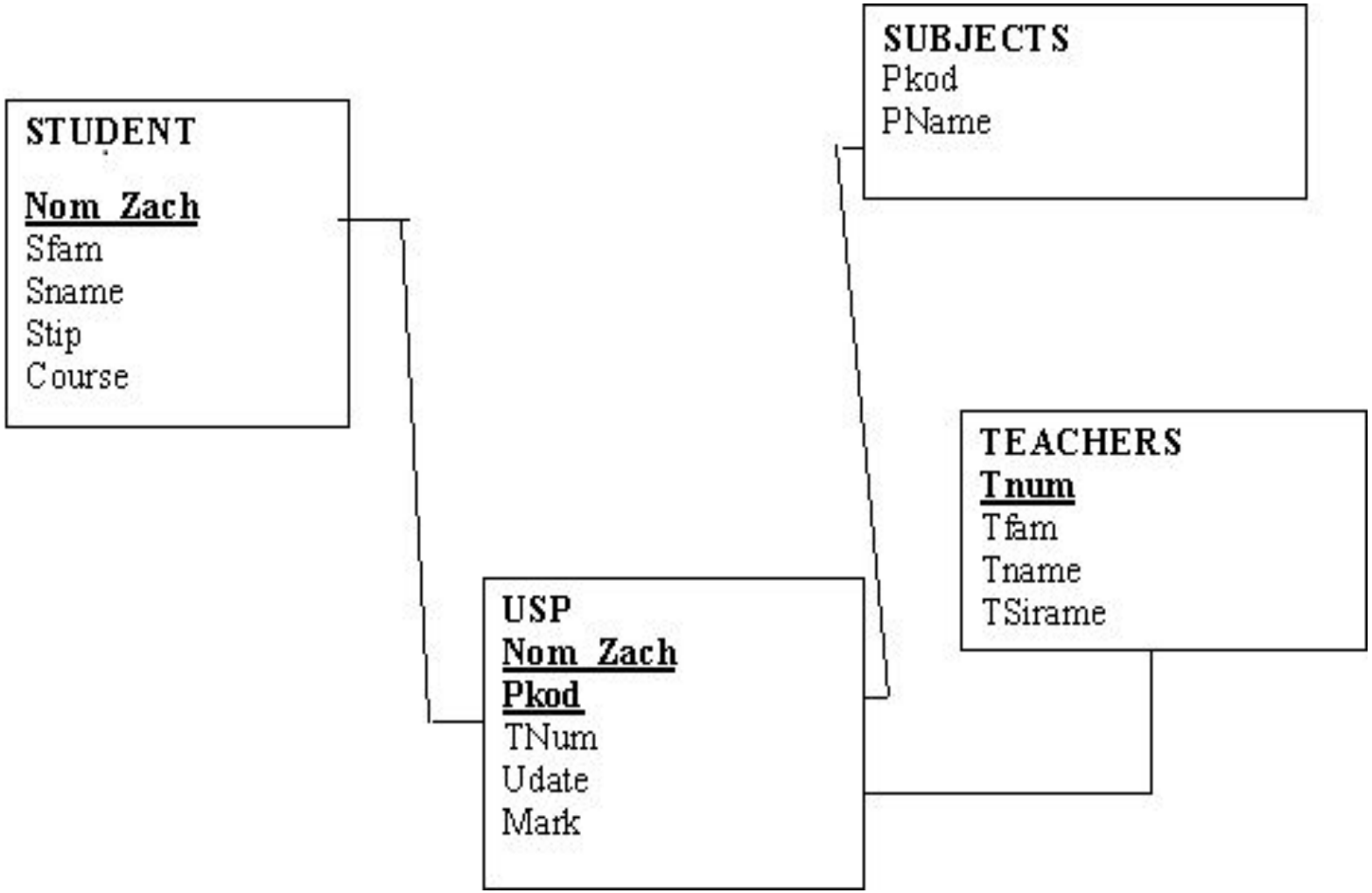
```
SELECT Телефоны.*, Звонки.Дата, Звонки.Город,

Звонки.Продолжительность

FROM Телефоны RIGHT JOIN Звонки ON

Телефоны.Номер_телефона =

Звонки.Номер_телефона;
```


SUBJECTS
Pkod
PName
STUDENT
Nom_Zach
Sfam
Sname
Stip
Course
TEACHERS
Tnum
Tfam
Tname
TSirame
USP
Nom_Zach
Pkod
TNum
Udate
Mark

USP

n_zach	Pkod	mark
a	01	5
a	05	5
b	01	6

SUBJECTS

ŴŶĂŵĞ	Ŵ Ě
ls ' ' ^dzs	66
> '	6ϱ
' h	6ϳ

SELECT * FROM USP, SUBJECTS

n_zach	USP.Pkod	mark	SUBJECTS. Pkod	SUBJECTS.Pname
a	01	5	01	МДиСУБД
a	05	5	01	МДиСУБД
b	01	6	01	МДиСУБД
a	01	5	05	СТП
a	05	5	05	СТП
b	01	6	05	СТП
a	01	5	03	МПИ
a	05	5	03	МПИ
b	01	6	03	МПИ

SELECT N_Zach, PNAME, mark

FROM USP INNER JOIN SUBJECTS

ON USP.Pkod= SUBJECTS.Pkod

N_Zach	PNAME	Mark
a	МД и СУБД	5
b	МД и СУБД	6
a	СТП	5

```
SELECT n_zach, pname, mark
FROM usp, subjects
where usp.pkod=subjects.pkod
```

N_Zach	PNAME	Mark
a	МД и СУБД	5
a	МД и СУБД	5
a	СТП	5
NULL	МПИ	NULL

```
SELECT n_zach, SUBJECTS.pname,
  mark

  FROM USP RIGHT JOIN SUBJECTS

   ON MARKS.pkod=SUBJECTS.pkod
```

```
SELECT UPPER(SFAM)
FROM STUDENTS

SELECT UCASE(SFAM)
FROM STUDENTS
```

Раздел WHERE

Раздел ***WHERE*** задаёт условия отбора строк.

Имена атрибутов, входящие в предложение ***WHERE*** могут не входить в набор атрибутов, перечисленных в предложении ***SELECT***.

В выражении условий раздела ***WHERE*** могут быть использованы следующие предикаты

- Предикаты сравнения **{=, >, <, >=, <=, <>.** }.
- Предикат ***BETWEEN*** **A** ***AND*** **B**. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона.

SELECT * FROM USP WHERE mark BETWEEN 4 AND 6

- Предикат вхождения во множество

IN (множество) истинен тогда, когда сравниваемое значение входит во множество заданных значений. Выражение **IN("A","B","C")** означает то же, что и

"A" OR "B" OR "C"

SELECT * FROM USP where mark in (3,4,5)

При этом множество может быть задано простым перечислением или встроенным подзапросом. Одновременно существует противоположный предикат **NOT IN** (множество).

- Предикаты сравнения с образцом

  LIKE и ***NOT LIKE***

Предикат ***LIKE*** требует задания шаблона, с которым сравнивается заданное значение.

В образец поиска можно включать символы шаблона: **%, _ , ^.**

Допустимый диапазон заключается в квадратные скобки.

- Предикат сравнения с неопределённым значением ***IS NULL***.

Неопределённое значение интерпретируется в реляционной модели как значение, неизвестное в данный момент времени.

NOM_Zach	PNAME	Mark
a	МД и СУБД	5
b	МД и СУБД	6
a	МД и СУБД	5
a	СТП	5
NULL	МПИ	NULL

Регулярное выражение это конструкции позволяющие вести поиск в тексте по различным условиям.

Регулярные выражения очень широко используется как для поиска (в случае SQL), так и для ограничений ввода, к примеру, при вводе номера телефона, телефон должен соответствовать маске XXX-XX-XX, где X- это число.

Регулярное выражение содержит один и более метасимволов.

Подстановочные знаки SQL 92

- **%** - соответствует любому количеству знаков.

wh% — поиск слов what, white и why.

- _ - соответствует любому текстовому символу.

(**B_ll** - поиск слов Ball, Bell и Bill)

- [ ] - соответствует одному любому знаку из заключенных в скобки.

(**B[ae]**ll — поиск слов Ball и Bell, но не Bill)

- ^ - соответствует одному любому знаку, кроме заключенных в скобки.

(b[^ae]ll — поиск слов bill и bull, но не bell или ball)

- - соответствует любому знаку из диапазона. Необходимо указывать этот диапазон по возрастанию:

- (**b[a-c]d** — поиск слов bad, bbd и bcd.)

Для поиска символов '%‘ и '_' можно задать ***ESCAPE*** символ -

символ, помещаемый перед символом-шаблоном, чтобы символ-шаблон рассматривался как обычный символ, а не как шаблон.

Например, для поиска строк, начинающихся символами '13%' можно задать

LIKE ‘13#%’ ESCAPE ‘#’

Вывести список студентов, у которых в поле ***FNAME*** содержится символ "_"

```
SELECT*
FROM STUDENT
 WHERE FNAME LIKE ‘%#_%’
  ESCAPE ‘#’
```

Найти все издания, которые содержат в заголовке текст "10%".

Предложение **WHERE** в инструкции SQL будет иметь следующий вид:

WHERE TITLE LIKE '%10#%%' ESCAPE '#'

Найти товары, коды которых начинаются с четырёх букв "A%BC

SELECT PRODUCT FROM ORDERS WHERE PRODUCT LIKE 'A$%BC%' ESCAPE '$'

Первый символ процента в шаблоне, следующий за символом пропуска, считается литералом, второй — подстановочным знаком

Чтобы использовать символ-шаблон в качестве литерала, его можно заключить в скобки.

Вывести список студентов, у которых в поле ***FNAME*** содержится символ "_"

```
SELECT*
FROM STUDENT
 WHERE FNAME LIKE ‘%[_]%’
```

Когда запрос включает предложение ***WHERE***, СУБД просматривает всю таблицу по одной записи, чтобы определить является ли предикат истинным.

Предикат может включать неограниченное число условий, содержащих булевы операторы.

Например, создать запросы для вывода сведений о студентах, чьи фамилии начинаются на букву "С" или "К" и заканчиваются буквой "й”

```
SELECT *
FROM Student
WHERE Sfam LIKE ‘[СК]%й’
```

- Сведений о студентах, чьи фамилии начинаются на любую букву, исключая "Н" и состоят из восьми букв.

SELECT * FROM STUDENTS WHERE SFam LIKE "[^H]_____"

Проверка на равенство значению NULL (оператор IS NULL)

Значения null обеспечивают возможность применения трехзначной логики в условиях отбора.

Для любой заданной строки результат применения условия отбора может быть **true**, **false** или **null** (в случае, когда в одном из столбцов содержится значение null).

Трехзначная логика

Значения **NULL** влияют на результаты сравнений.

При сравнении двух значений **x** и **y**, если **x** или **y** имеет значение **NULL**, то результатом некоторых логических сравнений будет значение **UNKNOWN**, а не **TRUE** или **FALSE**.

AND	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	FALSE
NULL	TRUE	FALSE	NULL

NOT	**TRUE**	**FALSE**	**NULL**
	FALSE	TRUE	NULL

Иногда бывает необходимо явно проверять значения столбцов на равенство **NULL**.

Для этого в SQL имеется специальная проверка IS NULL.

Студент

код студента	Фамилия	Имя	Отчество	Группа
2009001	Смирнов	Иван	Иванович	11
2009002	Петров	Иван	Иванович	11
20[illegible]	[illegible]	[illegible]		П1

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	3
2009001	11	3
2009002	10	10
2009002	11	10

Пример. Вывести фамилии студентов, сдававших экзамены.

```
SELECT Студент.Фамилия
FROM Студент, Оценки
WHERE Студент.[код студента]
= Оценки.[Код студента]
```

```
SELECT Студент.Фамилия
FROM Студент INNER JOIN Оценки ON
Студент.[код студента] =
Оценки.[Код студента];
```

Запрос10

Фамилия
Смирнов
Смирнов
Петров
Петров

```
SELECT DISTINCT Студент.Фамилия
FROM Студент INNER JOIN Оценки
ON Студент.[код студента] =
Оценки.[Код студента];
```

Запрос10

Фамилия
Петров
Смирнов

Пример. Вывести список студентов, не сдававших экзамены.

Студент

код студента	Фамилия	Имя	Отчество	Группа
2009001	Смирнов	Иван	Иванович	11
2009002	Петров	Иван	Иванович	11
20	Сидоров	Михаил	Иванович	П1

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	3
2009001	11	3
2009002	10	10
2009002	11	10

```
SELECT Студент.Фамилия,
Код_дисциплины
FROM Студент LEFT JOIN Оценки ON
Студент.[код студента] =
Оценки.[Код студента]
```

Запрос10

Фамилия	Код_дисциплины
Смирнов	10
Смирнов	11
Петров	10
Петров	11
Синицын	NULL

```
SELECT Студент.Фамилия

FROM Студент LEFT JOIN Оценки ON
Студент.[код студента] =
Оценки.[Код студента]

WHERE Оценки.Код_дисциплины Is Null
```

Запрос10

Фамилия
Синицын

Пример. Вывести список студентов, получивших несколько троек:

Оценки

Код студента	Код_дисциплины	Оценка
2009001	10	3
2009001	11	3
2009002	10	10
2009002	11	10

SELECT *
FROM Оценки AS A,
Оценки AS B

Запрос10

A.Код студента	A. Код_дисциплины	A.Оценка	B.Код студента	B. Код_дисциплины	B.Оценка
2009001	10	3	2009001	10	3
2009002	10	10	2009001	10	3
2009001	11	3	2009001	10	3
2009002	11	10	2009001	10	3
2009001	10	3	2009002	10	10
2009002	10	10	2009002	10	10
2009001	11	3	2009002	10	10
2009002	11	10	2009002	10	10
2009001	10	3	2009001	11	3
2009002	10	10	2009001	11	3
2009001	11	3	2009001	11	3
2009002	11	10	2009001	11	3
2009001	10	3	2009002	11	10
2009002	10	10	2009002	11	10
2009001	11	3	2009002	11	10

```
SELECT *
FROM Оценки AS A, Оценки AS B
WHERE A.[Код студента]=B.[Код студента];
```

Запрос10

A.Код студента	A. Код_дисциплины	A. Оценка	B.Код студента	B. Код_дисциплины	B. Оценка
2009001	10	3	2009001	10	3
2009001	11	3	2009001	10	3
2009001	10	3	2009001	11	3
2009001	11	3	2009001	11	3
2009002	10	10	2009002	10	10
2009002	11	10	2009002	10	10
2009002	10	10	2009002	11	10

```
SELECT *
FROM Оценки AS A, Оценки AS B
WHERE A.[Код студента]=B.[Код студента] AND A.
Оценка=3 AND B. Оценка=3;
```

Запрос10

A.Код студента	A. Код_дисциплины	A. Оценка	B.Код студента	B. Код_дисциплины	B.Оценка
2009001	10	3	2009001	10	3
2009001	10	3	2009001	11	3
2009001	11	3	2009001	10	3
2009001	11	3	2009001	11	3

```
SELECT A. [Код студента]
FROM Оценки AS A, Оценки AS B
WHERE A.[Код студента]=B.[Код студента]
AND A.Оценка=3 AND B.Оценка=3 AND A.
Код_дисциплины<>B.Код_Дисциплины;
```

Запрос10

Код студента
2009001
2009001

```
SELECT DISTINCT Студент.Фамилия

FROM Оценки AS A, Оценки AS B, Студент

WHERE A.[Код студента]=B.[Код студента] AND A.Оценка=3 AND
A.Код_дисциплины<>B.Код_Дисциплины AND Студент.[Код студента]=A.[Код студента];
```

- Вывод списка шифров владельцев собственности (***Owner_no***), предлагающих несколько трехкомнатных квартир для продажи:

Property_No	Owner_no	Rooms
2	1	3
3	1	3
4	2	1
5	2	3

a.Property_No	a.Owner_no	a.Rooms	b.Property_No	b.Owner_no	b.Rooms
2	1	3	2	1	3
2	1	3	3	1	3
2	1	3	4	2	1
2	1	3	5	2	3
3	1	3	2	1	3
3	1	3	3	1	3
3	1	3	4	2	1
3	1	3	5	2	3
4	2	1	2	1	3
4	2	1	3	1	3
4	2	1	4	2	1
4	2	1	5	2	3
5	2	3	5	2	1

```
SELECT DISTINCT a.Owner_no
FROM PROPERTY a, PROPERTY b
WHERE a.Owner_no=b.Owner_no AND
a.Property_no<>b.Property_no AND
a.Rooms=3 AND b.Rooms=3;
```

В запросе используются псевдонимы a и b таблицы ***PROPERTY***, так как для выполнения запроса необходимо оценить равенство поля ***Owner_no*** в двух экземплярах одной и той же таблицы.

В результате выполнения оператора **FROM** получаем декартово произведение таблиц a и b, которая содержит все комбинации значений полей двух псевдонимов одной и той же таблицы **PROPERTY**. Если у владельца есть несколько квартир, в таблице будут записи, у которых значения поля **Owner_no** совпадают, **Property_no** отличаются.

Телефоны

Номер	Владелец	Адрес
61-32-72	Степанова	Чкалова 18-18
55-55-55	Иванов	Чкалова 15-15

Звонки

Номер	Дата	Город	Продолжит.
61-32-72	15.11.2002	Москва	5
61-32-72	12.11.2002	Токио	35

После **применения** к отношениям операции соединения будет получено отношение

Номер	Владелец	Адрес	Дата	Город	Продолж.
61-32-72	Степанова	Чкалова 18-18	15.11.2002	Москва	5
61-32-72	Степанова	Чкалова 18-18	12.11.2000	Токио	35
55-55-55	Иванов	Чкалова 15-15	NULL	NULL	NULL

1)

SELECT DISTINCT Владелец

FROM телефоны LEFT JOIN звонки ON телефоны.номер_телефона =звонки. номер_телефона

WHERE Дата IS Null;

Вывести номера телефонов абонентов звонивших по межгороду более одного раза

```
SELECT DISTINCT a.
   Номер_телефона
FROM Звонки a, Звонки b
WHERE
a.Номер_телефона=
b.Номер_телефона
and
a.Дата<>b.Дата
```

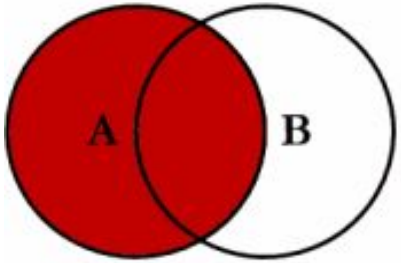
Вывести список всех звонков за январь

```
SELECT Номер_телефона,
   Город, Дата, FROM
   Звонки
WHERE Month(дата)=1
```

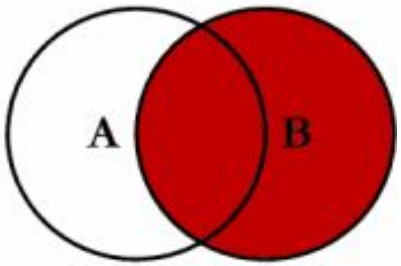
Рассчитать плату за разоворы

```
SELECT Звонки.Номер_телефона,
   IIf(Year(Дата)=2014, Звонки.
   Продолжительность*100,
 Звонки.Продолжительность*200)
   Плата
FROM Звонки
```

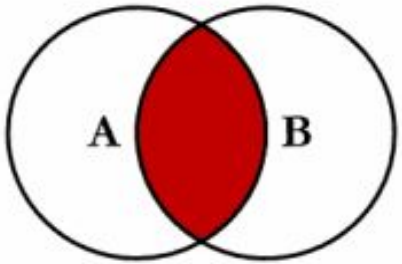
SQL JOINS



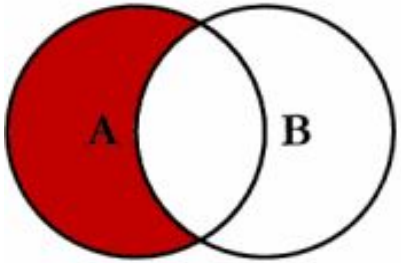

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



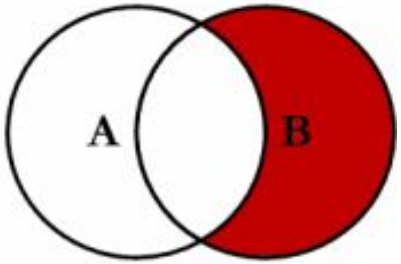

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



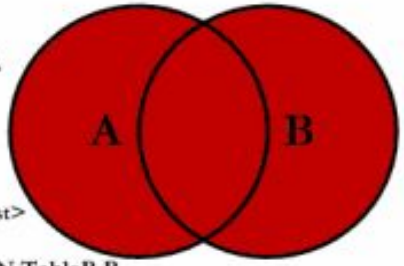

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



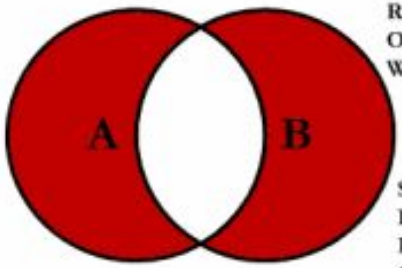

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

FULL [OUTER]

Указывает, что в результирующий набор включаются строки как из левой, так и из правой таблицы, несоответствующие условиям соединения, а выходные столбцы, соответствующие оставшейся таблице, устанавливаются в значение NULL. Этим дополняются все строки, обычно возвращаемые при помощи INNER JOIN.

Раздел *GROUP BY*

Раздел ***GROUP BY*** используется для создания итоговых запросов.

В предложении ***SELECT*** таких запросов используется, по крайней мере, одна агрегатная функция (***AVG***, ***COUNT*** (количество непустых значений в данном столбце), ***SUM, MIN, MAX.***

С функциями ***SUM*** и ***AVG*** могут использоваться только числовые поля.

Синтаксис:

GROUP BY < имя_столбца>

Имя столбца – имя любого столбца из любой из упомянутой в разделе ***FROM*** таблицы.

Если ***GROUP BY*** расположено после ***WHERE*** создаются группы из строк, выбранных после применения раздела ***WHERE***.

При включении раздела ***GROUP BY*** в инструкцию ***SELECT*** список отбираемых полей может содержать **имена полей, указанные в разделе *GROUP BY* и итоговые функций SQL.**

В раздел ***GROUP BY*** должны быть включены все атрибуты, входящие в раздел ***SELECT***.

В предложении ***GROUP BY*** могут быть указаны одновременно несколько столбцов. Группы при этом определяются слева направо.

Предложение ***GROUP BY*** автоматически устанавливает сортировку по возрастанию (если надо по убыванию – задать в ***ORDER BY).***

Студент

код студента	Фамилия	Имя	Отчество	Группа
2009001	Смирнов	Иван	Иванович	11
2009002	Петров	Иван	Иванович	11
200				П1

Оценки

Код студента	Код_дисциплины	Оценка
2009001	10	10
2009001	11	7
2009002	10	10
2009002	11	10

Примеры

- вычислить средний балл каждого студента:

SELECT Студент.Фамилия,AVG(оценка) AS Средняя

FROM Студент INNER JOIN Оценки ON Студент.[код студента] = Оценки.[Код студента]

GROUP BY Студент.Фамилия;

Запрос12

Фамилия	Средняя
Петров	10
Синицын	8
Смирнов	8,5

```
SELECT Студент.Фамилия,
AVG(оценка)

FROM Студент INNER JOIN Оценки ON
Студент.[код студента] =
Оценки.[код студента]

GROUP BY Студент.Фамилия
```

```
SELECT Студент.Фамилия,
Студент.[код студента],
 AVG(оценка)

FROM Студент INNER JOIN Оценки ON
Студент.[код студента] = Оценки.[код
студента]

GROUP BY Студент.Фамилия
```

```
SELECT Студент.Фамилия,
Студент.[код студента],
 AVG(оценка)

FROM Студент INNER JOIN Оценки
ON Студент.[код студента] =
Оценки.[код студента]

GROUP BY Студент.Фамилия,
Студент.[код студента]
```

SELECT Студент.Фамилия,
~~Студент.[код студента],~~
AVG(оценка)

FROM Студент INNER JOIN Оценки ON Студент.[код студента] = Оценки.[код студента]

GROUP BY Студент.Фамилия, Студент.[код студента]

Вычислить количество оценок 2, 3, 4, 5…, полученных на экзаменах.

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	10
2009001	11	7
2009002	10	10
2009002	11	10

Оценка	Количество
7	1
10	3

```
SELECT Оценка, COUNT(*) AS Количество
FROM Оценки
GROUP BY Оценка
```

Вычислить количество оценок 2, 3, 4, 5…, полученных на экзаменах по каждой дисциплине.

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	10
2009001	11	7
2009002	10	10
2009002	11	10

Дисциплины

Код_Дисциплины	Название_дисциплины
10	ЭВМ и программирование
11	Геометрия

Запрос12

Название_дисциплины	Оценка	Количество
Геометрия	7	1
Геометрия	10	1
ЭВМ и программмирование	8	1
ЭВМ и программмирование	10	2

```
SELECT Название_дисциплины,
Оценка, Count(*) AS Количество
FROM Оценки INNER JOIN Дисциплины
ON Оценки.[Код_ дисциплины] =
Дисциплины.Код_Дисциплины
GROUP BY Название_дисциплины,
Оценка;
```

Сколько десяток получил каждый студент

Студент

код студента	Фамилия	Имя	Отчество	Группа
2009001	Смирнов	Иван	Иванович	11
2009002	Петров	Иван	Иванович	11
20	Сидоров	Михаил	Иванович	П1

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	3
2009001	11	3
2009002	10	10
2009002	11	10

Запрос2

Фамилия	Количество 10
Петров	2

```
SELECT Фамилия, Count(Оценки.
Оценка) AS [Количество 10]

FROM Студент INNER JOIN Оценки
ON Студент.[код студента] =
Оценки.[Код студента]

WHERE Оценка=10

GROUP BY Студент.Фамилия;
```

1) Сколько экзаменов сдал каждый студент?

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	10
2009001	11	7
2009002	10	10
2009002	11	10

Запрос16

код студента	сдал
2009001	2
2009002	2

```
SELECT [код студента] , Count(*) AS [сдал]
FROM оценки
GROUP BY [код студента]
```

2) Сколько студентов в каждой группе

Студент

код студента	Фамилия	Имя	Отчество	Группа
2009001	Смирнов	Иван	Иванович	11
2009002	Петров	Иван	Иванович	11
2009003	Синицын	Михаил	Иванович	П1

Запрос16

Группа	всего студентов
11	2
П1	1

```
SELECT [группа] , Count(*) AS [всего студентов]
FROM студент
GROUP BY [группа]
```

Раздел ***HAVING*** задает условие отбора групп строк, которые включаются в таблицу, определяемую инструкцией ***SELECT***.

Условия отбора применяется к **столбцам, указанным в разделе *GROUP BY*, к столбцам итоговых функций** или к **выражениям, содержащим итоговые функции**. Если некоторая группа не удовлетворяет условию отбора, она не попадает в набор записей**.**

Синтаксис:

HAVING < *условие_отбора*>.

Разница между ***HAVING*** и ***WHERE*** заключается в том, что условие отбора, заданное в разделе ***WHERE*** применяется к **отдельным записям**, **перед их группировкой**, а условие отбора раздела ***HAVING*** применяется **к группам строк**.

Агрегатные функции могут применяться как в выражении вывода результатов строки ***SELECT***, так и в выражении обработки сформированных групп ***HAVING***.

Возможно использование агрегатной функции в выражении ***HAVING*** без включения ее в список вывода ***SELECT.***

Ключевое слово *HAVING* можно использовать только совместно с *GROUP BY*.

Допустимо, чтобы условие ***HAVING*** содержало **ссылку на любое поле в списке выборки,** включая агрегатные функции.

(Выражение ***WHERE*** не может содержать ссылки на агрегатные функции).

Сколько экзаменов сдал студент (Код студента=2009002)

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	10
2009001	11	7
2009002	10	10
2009002	11	10
2009003	10	8

Запрос2

Код студента	Количество
2009002	2

```
SELECT [Код студента], Count(Оценки.
Оценка) AS [Количество оценок]

FROM Оценки

WHERE [Код студента]="2009002«

GROUP BY [Код студента];
```

```
SELECT [Код студента], Count(Оценки.
Оценка) AS [Количество]

FROM Оценки

GROUP BY [Код студента]

HAVING  [Код студента]="2009002"
```

Запрос2

Код студента	Количество оценок
2009001	2
2009002	2
2009003	1

Запрос2

Код студента	Количество оценок
2009002	2

Количество десяток, полученных каждым студентом.

Оценки

Код студента	Код_ дисциплины	Оценка
2009001	10	10
2009001	11	7
2009002	10	10
2009002	11	10
2009003	10	8

Запрос7

код студента	Фамилия	Оценка	количество 10
2009001	Смирнов	10	1
2009002	Петров	10	2

```
SELECT Студент.[код студента],
Студент.Фамилия, Оценка,
Count(Оценки.Оценка) AS [количество
10]
FROM Студент INNER JOIN Оценки ON
Студент.[Код студента] =
Оценки.[Код студента]

GROUP BY Студент.[код студента],
Студент.Фамилия, Оценка

HAVING Оценка=10;
```

Запрос7

код студента	Фамилия	количество 10
2009001	Смирнов	1
2009002	Петров	2

SELECT Студент.[код студента], Студент.Фамилия, Count(Оценки.Оценка) AS [количество 10]

FROM Студент INNER JOIN Оценки ON Студент.[Код студента] = Оценки.[Код студента]

WHERE Оценка=10

GROUP BY Студент.[код студента], Студент.Фамилия ~~, Оценки.Оценка~~

```
SELECT Студент.[код студента], Студент.Фамилия,

Count(Оценки.Оценка) AS [количество 10]

FROM Студент INNER JOIN Оценки ON Студент.[Код студента] = Оценки.[Код студента]

WHERE Оценка=10

GROUP BY Студент.[код студента], Студент.Фамилия
HAVING Count(*)>1
```

```
SELECT Студент.[код студента], Студент.Фамилия,

~~Count(Оценки.Оценка) AS [количество 10]~~

FROM Студент INNER JOIN Оценки ON Студент.[Код студента] = Оценки.[Код студента]

WHERE Оценка=10

GROUP BY Студент.[код студента], Студент.Фамилия
HAVING Count(*)>1
```

Вывести список групп, в которых обучается более 25 студентов

Студент

код студента	Фамилия	Имя	Отчество	Группа
2009001	Смирнов	Иван	Иванович	11
2009002	Петров	Иван	Иванович	11
2009003	Синицын	Михаил	Иванович	П1
…				

Студент

Группа	Всего студентов
11	**30**

```
SELECT [группа] ,
Count(*) AS [всего
студентов]
FROM студент
GROUP BY [группа]
HAVING Count(*) >25
```

```
SELECT [группа] , Count(*) AS [всего студентов]

FROM студент

WHERE Count(*) >25

GROUP BY [группа]
```

```
SELECT Оценки.[Код студента]
FROM Оценки
GROUP BY Оценки.[Код студента]
HAVING Avg(Оценка)>6;
```

Пусть имеются 2 отношения:

Телефоны

Номер	Владелец	Адрес
61-32-72	Степанова	Чкалова 18-18
55-55-55	Иванов	Чкалова 15-15

Звонки

Номер	Дата	Город	Продолжит.
61-32-72	15.11.2002	Москва	5
61-32-72	12.11.2002	Гокио	35

Вывести список абонентов, говоривших по межгороду > 20 минут

Запрос18

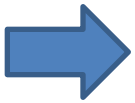
Владелец	Итого_минут
Степанова	40

```
SELECT Телефоны.Владелец,
Sum(Продолжительность) AS
Итого_минут

FROM Телефоны INNER JOIN Звонки ON
Телефоны.Номер_телефона=Звонки.
Номер_телефона

GROUP BY Телефоны.Владелец

HAVING Sum(Продолжительность)>20;
```



SELECT *Владелец*

FROM *Телефоны* ***INNER JOIN*** *Звонки* ***ON*** *Телефоны.Номер_телефона=Звонки.Номер_телефона*

GROUP BY *Телефоны.Владелец*

HAVING Sum*(Продолжительность)>20;*

Запрос18

Владелец

Степанова

Пример. Вывести список студентов, получивших несколько троек:

```
SELECT [Код студента]

FROM Оценки

Where Оценка=3

GROUP BY [Код студента]

HAVING COUNT(*)>1;
```

SELECT *[Код студента],оценка*

FROM *Оценки*

GROUP BY *[Код студента],оценка*

HAVING Оценка=3 and COUNT*(оценка) >1;*

При наличии в запросе раздела **HAVING**, которому не предшествует раздел *GROUP BY*, таблица рассматривается как сгруппированная таблица, состоящая из одной группы строк, без столбцов группирования.

```
SELECT  Count(*) AS [всего десяток]

FROM оценки

 WHERE оценка =10

HAVING Count(*)>3
```

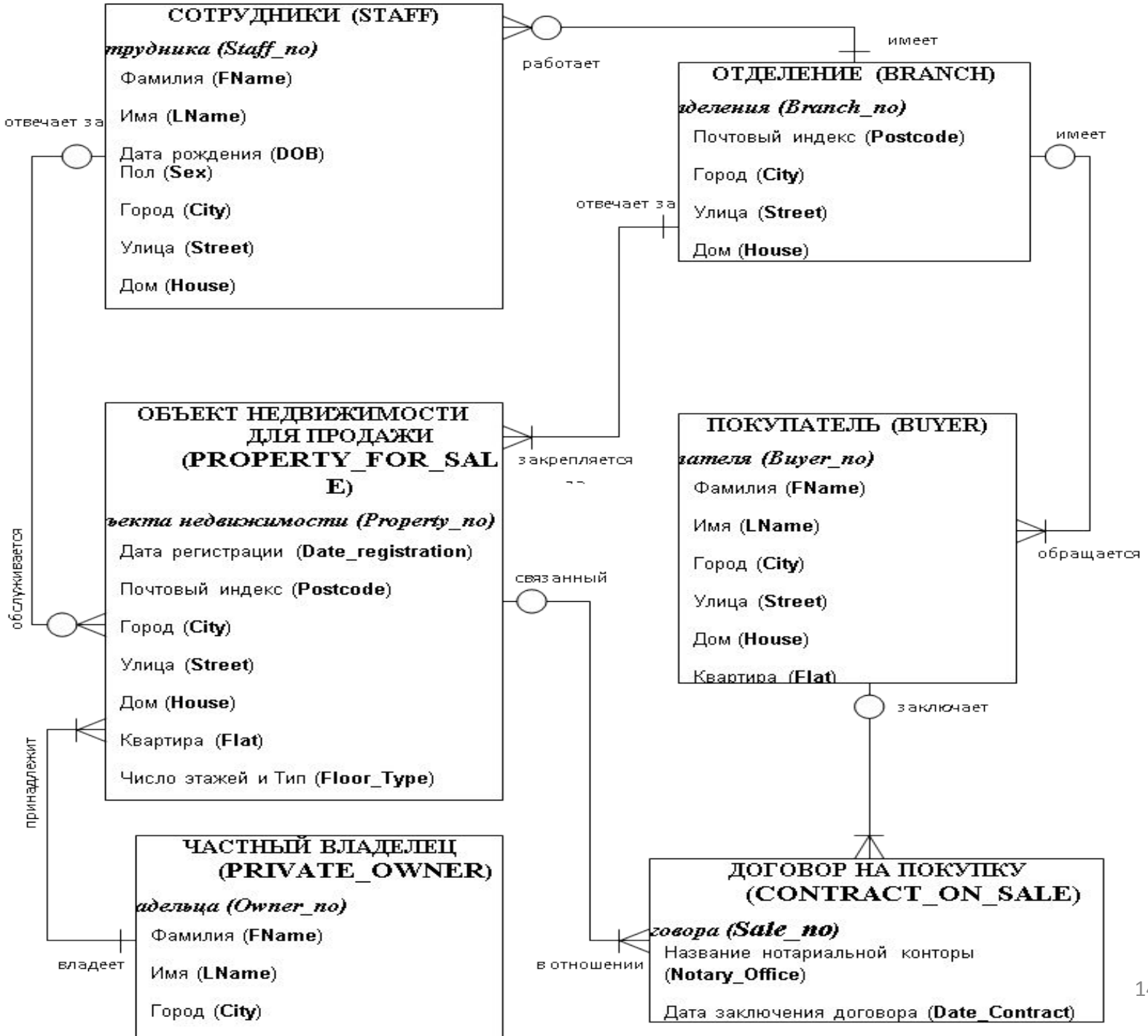
SELECT COUNT(*) AS *[кол-во студентов]*
FROM *Студент*

```
SELECT Группа,COUNT(*) AS [кол-во]
FROM Студент
WHERE Группа LIKE "П*"
GROUP BY Группа
```

```
SELECT count(*) AS [кол-во]
FROM Студент
WHERE Группа LIKE "П*"
```

Например, вывести названия и номера телефонов отделений, которые предлагают более одной трехкомнатной квартиры.

```
SELECT PROPERTY.Branch_no, BRANCH. Btel_no
FROM BRANCH, PROPERTY
WHERE PROPERTY.Branch_no=BRANCH.Branch_no AND
PROPERTY.Rooms=3
GROUP BY PROPERTY.Branch_no
HAVING COUNT(*)>1;
```


СОТРУДНИКИ (STAFF)
трудника (Staff_no)
Фамилия (FName)
Имя (LName)
Дата рождения (DOB)
Пол (Sex)
Город (City)
Улица (Street)
Дом (House)
работает
имеет
ОТДЕЛЕНИЕ (BRANCH)
деления (Branch_no)
Почтовый индекс (Postcode)
Город (City)
Улица (Street)
Дом (House)
имеет
отвечает за
отвечает за
ОБЪЕКТ НЕДВИЖИМОСТИ ДЛЯ ПРОДАЖИ (PROPERTY_FOR_SALE)
ъекта недвижимости (Property_no)
Дата регистрации (Date_registration)
Почтовый индекс (Postcode)
Город (City)
Улица (Street)
Дом (House)
Квартира (Flat)
Число этажей и Тип (Floor_Type)
закрепляется
ПОКУПАТЕЛЬ (BUYER)
пателя (Buyer_no)
Фамилия (FName)
Имя (LName)
Город (City)
Улица (Street)
Дом (House)
Квартира (Flat)
обращается
обслуживается
связанный
заключает
принадлежит
ЧАСТНЫЙ ВЛАДЕЛЕЦ (PRIVATE_OWNER)
адельца (Owner_no)
Фамилия (FName)
Имя (LName)
Город (City)
владеет
в отношении
ДОГОВОР НА ПОКУПКУ (CONTRACT_ON_SALE)
говора (Sale_no)
Название нотариальной конторы (Notary_Office)
Дата заключения договора (Date_Contract)

Вывести список владельцев собственности (Owner_no), предлагающих несколько квартир

Property_no	Owner_no
3000	1
3001	5
3002	7
3003	5
3004	7
3005	6
3006	3
3007	2
3008	7

```
SELECT OWNER_No,
COUNT(*) AS Количество
FROM PROPERTY
GROUP BY OWNER_No
HAVING COUNT(*)>1
```

Owner_no	Количество
5	2
7	3

Сортировка результатов запроса

***ORDER BY** имя_поля **ASC|DESC**;*

Если указывается несколько полей, то столбцы вывода упорядочиваются один внутри другого, при этом можно определить ***ASC*** (возрастание) или ***DESC*** (убывание).

Например, вывести все сведения студентах с упорядочением списка по убыванию номера группы:

```
SELECT *
FROM Студент
ORDER BY группа desc
```

Размещение текста в выводе запроса:

SELECT *имя_поля1+ ‘текст’, имя_поля2 …*

Этот способ можно использовать для маркировки вывода вместе со вставляемыми комментариями.

SELECT Телефоны.Номер_телефона, Владелец + 'проживающий по адресу: ' + Адрес AS Абонент

FROM Телефоны;

Запрос18

Номер_ телефона	Абонент
37-49-75	Степанова проживающий по адресу: Чкалова, 18—18

TOP n [PERCENT]

Возвращает некоторое количество записей, находящихся в числе первых записей диапазона, заданного предложением **ORDER BY**

Например, вывести 10 лучших студентов.

```
SELECT TOP 10[Код студента],
ROUND(AVG(оценка),2)
FROM Оценки

GROUP BY [Код студента]
ORDER BY  ROUND(AVG(оценка),2 )DESC;
```

```
SELECT TOP 1 [Код студента],
ROUND(AVG(оценка),1)

FROM Оценки

GROUP BY [Код студента]

ORDER BY ROUND(AVG(оценка),1) DESC
;
```

Предикат **TOP** не предполагает выбора между равными значениями. В примере, если десятый и одиннадцатый студент имеют одинаковый средний балл, в ответ на запрос будет выведено 11 записей.

Можно также использовать зарезервированное слово **PERCENT** для получения некоторого процента записей, находящихся в числе первых или последних записей диапазона, заданного предложением ORDER BY.

Вывести группу, в которой получен максимальный средний балл

```
SELECT  TOP 1 Группа,ROUND(AVG(оценка),2 )

FROM Студент INNER JOIN Оценки ON
Студент.[код студента] = Оценки.[Код
студента]

GROUP BY Группа

ORDER BY  ROUND(AVG(оценка),2) DESC
```

Запрос на объединение (UNION)

Запрос на объединение позволяет выполнить два запроса независимо друг от друга и объединить их результаты. Запросы должны быть совместимы по объединению, то есть иметь одинаковое количество отбираемых столбцов, типы соответствующих столбцов должны совпадать.

В выходном запросе отсутствуют дублирующие друг друга строки. Если надо оставить все строки в запросе, то после **UNION** следует указать **ALL.**

Пример:

```
SELECT *
FROM Студент1
UNION ALL
SELECT *
FROM Студент2
```

Обычно оператор ***UNION*** используют для объединения данных двух независимых таблиц с одинаковой структурой.

Если используется ключевое слово **ALL**, повторяющиеся строки не удаляются из объединённого набора. Это может существенно повысить скорость обработки запроса, поскольку не нужно выполнять проверку результатов на наличие повторов.

Ключевое слово **ALL** рекомендуется использовать в следующих условиях:

- В результате выполнения запросов на выборку не возникают повторяющиеся строки.
- Наличие повторяющихся строк не имеет значения.
- Нужно просмотреть повторяющиеся строки.