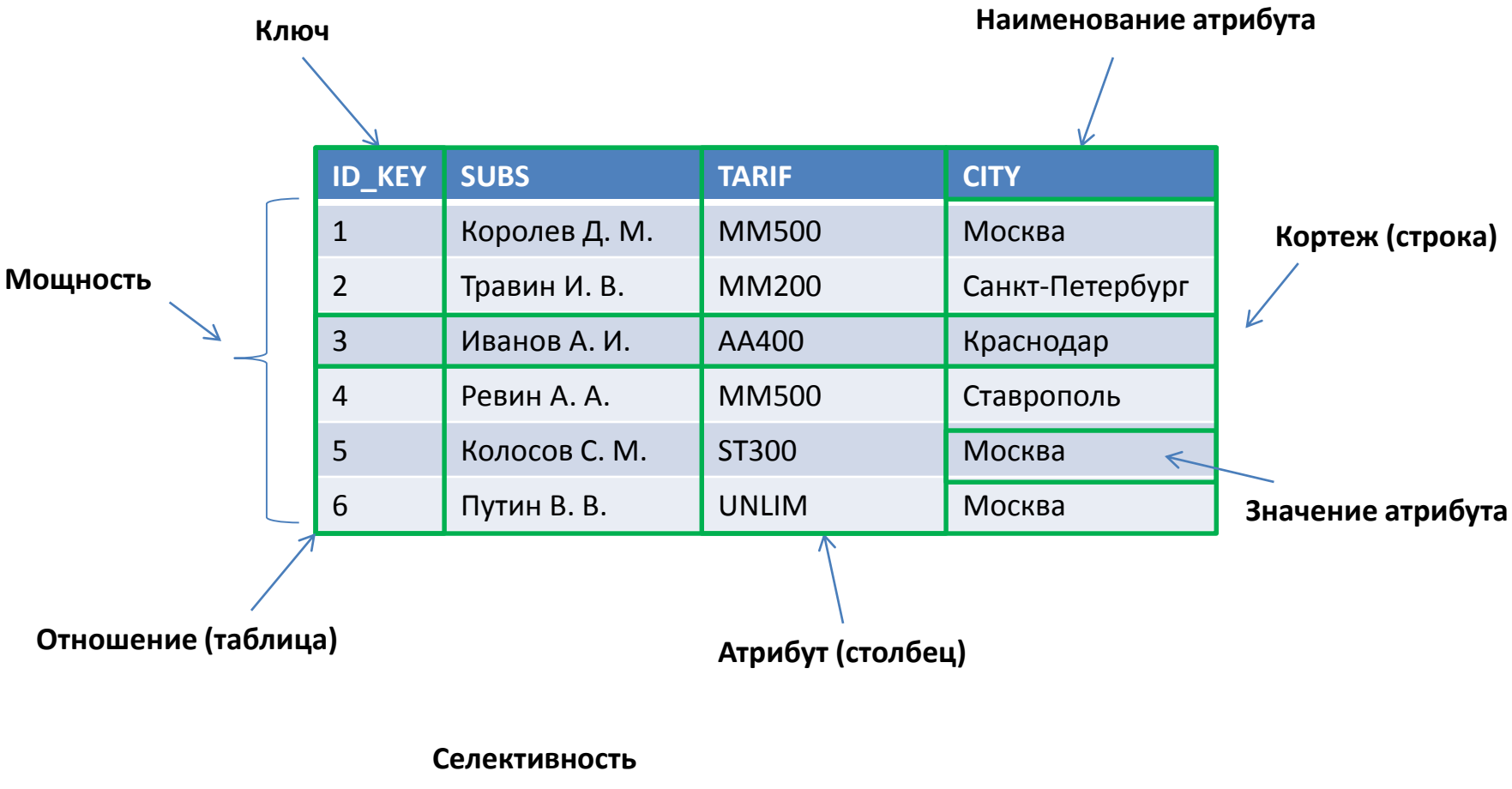
SQL. Основы.

Тренер: Ольховский Никита

Основные понятия реляционных баз данных



ID_KEY	SUBS	TARIF	CITY
1	Королев Д. М.	MM500	Москва
2	Травин И. В.	MM200	Санкт-Петербург
3	Иванов А. И.	AA400	Краснодар
4	Ревин А. А.	MM500	Ставрополь
5	Колосов С. М.	ST300	Москва
6	Путин В. В.	UNLIM	Москва

Типы данных

CHAR(n) – Символьная строка фиксированной длины, имеющая максимальную длину *n* символов. Длина по умолчанию – 1, максимальная – 2000

VARCHAR2(n) – Символьная строка переменной длины, имеющая максимальную длину n байт. Максимальное количество – 4000 байт

INT – Масштабируемое целое

NUMBER(n, m) – Масштабируемое целое с плавающей точкой. n – общее количество знаков, m – количество знаков после запятой

DATE – Дата и время

Неопределенное значение NULL

- **NULL** – значение, которое недоступно, не присвоено, неизвестно или неприменимо
- Это не ноль и не пробел
- **NVL**(поле, значение) – функция, которая при нахождение NULL в поле, возвращает значение

COMMIT, ROLLBACK и SAVEPOINT

Команда	Описание
COMMIT	Завершает текущую транзакцию, делая все незафиксированные изменения постоянными
SAVEPOINT *имя*	Создает точку сохранения в текущей транзакции.
ROLLBACK	Команда ROLLBACK прекращает текущую транзакцию, отменяя все произведенные изменения в данных.
ROLLBACK TO *SAVEPOINT имя*	Команда ROLLBACK TO *SAVEPOINT имя* отменяет все произведенные изменения до точки сохранения <*имя*>.Если опущено предложение TO SAVEPOINT, команда ROLLBACK отменяет все изменения транзакции. Так как точки сохранения создаются логически, нельзя просмотреть список созданных точек сохранения.

Создание таблицы

CREATE TABLE имя_таблицы (*элемент описания, элемент описания*)

Элемент описания – описание столбца, ограничение таблицы

Описание столбца – имя столбца, тип данных

DESC [RIBE]– вывод структуры таблицы

```
CREATE TABLE PERSON
(PERSON_ID INTEGER PRIMARY KEY,
FNAME VARCHAR2(20),
LNAME VARCHAR2(20),
GENDER CHAR(1),
BIRTH_DATE DATE,
ADRESS VARCHAR2(30),
CITY VARCHAR2(20))
;
```

Ограничения на столбец

- **NOT NULL** – в столбце не может быть значений NULL

CITY VARCHAR2(40) **NOT NULL**

- **UNIQUE** – значение должно быть уникальным в пределах данной таблицы

NAME VARCHAR2(40) **UNIQUE**

- **PRIMARY KEY** – в столбце не может быть значений NULL и все значения уникальны

PERSON_ID INT **PRIMARY KEY**;

- **CHECK** – проверяемые ограничения

CITY VARCHAR2(30) **CHECK** (CITY **IN** ('Харьков','Полтава'))

- **REFERENCES (FOREIGN KEY)** – внешний ключ

UNIV_ID INT **REFERENCES** UNIVERSITY (UNIV_ID)

Изменение таблицы

- **ALTER TABLE** – используется для модификации структуры и параметров существующей таблицы
- **ALTER TABLE** имя_таблицы ADD (*имя _столбца тип данных размер*)
- **ALTER TABLE** имя_таблицы MODIFY (*имя _столбца тип данных размер*)

- **DROP TABLE** имя_таблицы – используется для удаления таблицы

- **TRUNCATE** TABLE имя_таблицы – используется для очистки таблицы

ALTER TABLE STUDENT **ADD** (CITY VARCHAR2(40));

ALTER TABLE LECTURER **MODIFY** (SURNAME CHAR(40));

DROP TABLE STUDENT;

TRUNCATE TABLE STUDENT;

Оператор SELECT

- **SELECT [DISTINCT]** список атрибутов [псевдоним]
- **FROM** список таблиц
- [**WHERE** условие выборки]
- [**ORDER BY** список атрибутов]
- [**GROUP BY** список атрибутов]
- [**HAVING** условие]
- [**UNION** выражение с оператором **SELECT]**;

```
SELECT SURNAME, NAME
FROM STUDENT
WHERE SURNAME = 'Петров';
```

STUDENT	
SURNAME	**NAME**
Петров	Антон
Петров	Илья

Операторы для работы с данными в таблице

- **INSERT** – осуществляет вставку в таблицу новой строки.
- **INSERT INTO** имя_таблицы [поля таблицы] **VALUES** *(значение, значение);*

- **DELETE** – удаляет строки из таблицы.
- **DELETE FROM** имя_таблицы

- **UPDATE** – обновляет значения полей.
- **UPDATE** имя_таблицы **SET** изменение

```
UPDATE STUDENT
SET STIPEND=450
WHERE STIPEND<450;

INSERT INTO STUDENT (NAME, SURNAME) VALUES('Никита', 'Ольховский');

DELETE FROM STUDENT
WHERE STIPEND<300;
```

Псевдоним (алиас) столбца или таблицы

- **Псевдоним (алиас) столбца** – альтернативный заголовок столбца, удобен при вычислениях.

```
SELECT S.NAME AS ИМЯ, S.SURNAME  AS ФАМИЛИЯ, U.UNIV_NAME УНИВЕР
FROM STUDENT S, UNIVERSITY U
WHERE S.CITY=U.CITY;
```

ИМЯ	ФАМИЛИЯ	УНИВЕР
Никита	Сергеев	Гарвард
Андрей	Степанов	Кэмбридж
Антон	Третьяков	МГУ
Сергей	Кислицын	МГИМО
Кирилл	Антонов	Стэнфорд

Операторы сравнения

Оператор	Значение
<	Меньше, чем
>	Больше, чем
=	Равно
>=	Больше или равно
<=	Меньше или равно
<>	Не равно
!=	Не равно

Арифметические выражения

Обозначение	Оператор
*	Умножение
/	Деление
+	Сложение
-	Вычитание

SELECT NAME, STIPEND, STIPEND +300
FROM STUDENT

STUDENT		
NAME	**STIPEND**	**STIPEND +300**
Никита	340	640
Алексей	470	770
Антон	290	590

Операторы сравнения с множеством значений IN, ANY, All

Операторы	
IN	*Равно любому из значений, полученных во вну*треннем запросе
NOT IN	*Не равно ни одному из значений, полученных* во внутреннем запросе
= ANY	Соответствует логическому оператору OR
> ANY, > = ANY	*Больше, чем (либо больше или равно) любое по*лученное число.
< ANY, < = ANY	*Меньше, чем (либо меньше или равно) любое* полученное число
= ALL	Равно всем полученным значениям
< ALL, < = ALL	*Меньше, чем (либо меньше или равно) все полу*ченные числа
> ALL, > = ALL	*Больше, чем (либо больше или равно) все полу*ченные числа

Операторы IN, BETWEEN, LIKE, is NULL

IN – равен любому из списка
NOT IN – не равен ни одному из списка
BETWEEN – проверка условия вхождения значения поля в заданный интервал

```
SELECT *
FROM EXAM_MARKS
WHERE MARK IN (4, 5 );

SELECT *
FROM SUBJECT
WHERE HOUR BETWEEN 30 AND 40;
```

Операторы IN, BETWEEN, LIKE, is NULL

LIKE – проверка соответствия строковых полей
NOT LIKE – проверка несоответствия строковых полей
% - любое количество любых символов
_ - один символ
ESCAPE ‘символ’ – исключающий символ
IS NULL – поле является пустым
IS NOT NULL – поле не является пустым

```
SELECT *
FROM STUDENT
WHERE SURNAME LIKE 'P%';
```

```
SELECT *
FROM STUDENT
WHERE SURNAME LIKE 'Сергей/_/_/_255' ESCAPE '/';
```

Конкатенация строк

- **Конкатенация строк** – «склеивание»значений двух или более столбцов символьного типа или символьных констант в одну строку.
- значимое_символьное_выражение || значимое_символьное_выражение
- **WM_CONCAT**(поле) – функция, переводящая столбцы в строку

```
SELECT SURNAME ||'_'|| NAME FIO, STIPEND
FROM STUDENT
WHERE KURS = 4 AND STIPEND > 0;
```

STUDENT	
FIO	**STIPEND**
Петров_Сергей	150
Иванов_Петр	300
Сергеев_Андрей	670
Коротков_Артем	300

Функции преобразования символов в строке

- **LOWER** — перевод в строчные символы (нижний регистр)
- **LOWER** (строка)
- **UPPER** — перевод в прописные символы (верхний регистр)
- **UPPER** (строка)
- **INITCAP** — перевод первой буквы каждого слова строки в заглавную
- **INITCAP** (строка)

```
SELECT LOWER (SURNAME), UPPER (NAME)
FROM STUDENT
WHERE KURS = 4 AND STIPEND > 0;
```

STUDENT	
SURNAME	**NAME**
иванов	ПЕТР
синицын	ПАВЕЛ
горбатенко	ИЛЬЯ

Строковые функции

- **LPAD —** дополнение строки слева, строка дополняется слева заданной в подстроке последовательностью символов до указанной длины.
- LPAD (строка, длина[,подстрока])

- **RPAD** — дополнение строки справа, строка дополняется слева заданной в подстроке последовательностью символов до указанной длины.
- RPAD (строка, длина[,подстрока])

- **LTRIM** — удаление левых граничных символов, из строки удаляются слева символы, указанные в подстроке.
- LTRIM (строка, подстрока)

- **RTRIM** — удаление правых граничных символов, из строки удаляются справа символы, указанные в подстроке.
- RTRIM (строка, подстрока)

- **TRIM** – удаление граничных символов

Строковые функции

- **SUBSTR** — выделение подстроки, из строки выбирается заданное количество символов, начиная с указанной параметром начало позиции в строке
- SUBSTR (<строка>,<начало>[,<количество>])

- **INSTR** — поиск подстроки, начало поиска задает начальную позицию в строке для поиска подстроки. Тип возвращаемого значения — INT, функция возвращает позицию найденной подстроки.
- INSTR (<строка>,<подстрока>[,<начало поиска> [,<номер вхождения>]])

- **LENGTH** — определение длины строки
- LENGTH (<строка>)

Примеры запросов, использующих строковые функции

```
SELECT LPAD (SURNAME, 10, '@'), RPAD (NAME, 10, '$')
FROM STUDENT
WHERE KURS = 3 AND STIPEND > 0;
```

STUDENT	
LPAD (SURNAME, 10, '@')	**RPAD (NAME, 10, '$')**
@@@Смертин	Петр$$$$$$
@@@@Петров	Павел$$$$$
@@@@@@Соев	Илья$$$$$$

```
SELECT SUBSTR (NAME, 1, 1) ||'.' || SURNAME FIO, CITY, LENGTH (CITY)
FROM STUDENT
WHERE KURS IN(2, 3, 4)AND STIPEND > 0;
```

STUDENT		
FIO	**CITY**	**LENGTH (CITY)**
П.Смертин	Москва	6
П.Петров	Тверь	5
И.Соев	Ставрополь	10

Числовые функции

- **ROUND** – округляет столбец, выражение или значение до заданной точности
- **ROUND** (столбец(выражение), n)

- **TRUNC** – усекается столбец, выражение или значение до заданного количества десятичных знаков
- **TRUNC** (столбец(выражение), n)

- **MOD** – возвращает остаток от деления
- **MOD** (m, n)

SELECT ROUND (45.2455, 2), **ROUND** (45.2455, 0), **TRUNC** (45.2455, 2)
FROM DUAL

ROUND (45.2455, 2)	ROUND (45.2455, 0)	TRUNC (45.2455, 2)
45.25	45	45.24

Функции преобразования значений

- **TO_CHAR** (<значимое выражение>[, <символьный формат>])
- **TO_NUMBER** (<значимое символьное выражение >)
- **TO_DATE** (<значимое символьное выражение >, <символьный формат>)

```
SELECT SURNAME,NAME,BIRTHDAY,
TO_CHAR (BIRTHDAY, 'DD-MON-YYYY') TO_CHAR,
TO_CHAR (BIRTHDAY, 'DD.MM.YY') TO_CHAR2
FROM STUDENT;
```

SURNAME	NAME	BIRTHDAY	TO_CHAR	TO_CHAR2
Иванов	Петр	3/12/1982	3-дек-1982	3.12.82
Сергеев	Антон	1/12/1980	1 -дек- 1980	1.12.80
Копытин	Олег	7/06/1979	7-июн-1979	7.06.79
Живаев	Кирилл	1/08/1981	1-авг-1981	1.08.81

Работа с датами

- **SYSDATE** – это функция, возвращающая текущую дату и время сервера базы данных.

Операция	Результат	Описание
Дата + Число	Дата	Прибавляет количество дней к дате
Дата – Число	Дата	Вычитает количество дней из даты
Дата – Дата	Количество дней	Вычитает одну дату из другой
Дата +Число/24	Дата	Прибавляет часы к дате

```
SELECT SYSDATE AS TODAY, SYSDATE+7 AS NEXT_WEEK
FROM DUAL;
```

TODAY	NEXT_WEEK
28.09.2010	05.10.2010

Функции для работы с датами

- **MONTHS_BETWEEN** – вычисляет количество месяцев между *датой1* и *датой2*
- **MONTHS_BETWEEN** (дата1, дата2)

- **ADD_MONTHS** – прибавляет *n* месяцев к дате
- **ADD_MONTHS** (дата, n)

- **NEXT_DAY** – возвращает дату, когда наступит заданный день недели
- **NEXT_DAY** (дата, ‘char’)

- **LAST_DAY** – возвращает последнюю дату месяца, к которому принадлежит дата
- **LAST_DAY** (дата)

- **ROUND** – возвращает дату, округленную до единицы, заданной моделью формата
- **ROUND** (дата [, формат])

- **TRUNC** – возвращает дату, в которой время усечено до единицы, заданной моделью формата
- **TRUNC** (дата [, формат])

Форматы даты

- **‘DD-Mon-YY’**
- **‘DD-Mon-YYYY’**
- **‘MM/DD/YY’**
- **‘MM/DD/YYYY’**
- **‘DD.MM.YY’**
- **‘DD.MM.YYYY’**
- **‘HH24’**
- **‘HH24:MI’**
- **‘HH24:MI:SS’**
- **‘HH24:MI:SS.FF’**

DD – формат дня
MON – формат месяца в текстовой форме
MM – формат месяца в числовой форме
YY – год в числовой форме
HH – час в числовой форме
MI – минуты в числовой форме
SS – секунды в числовой форме
FF – доли секунд в числовой форме

Агрегирующие функции

- • **COUNT –** определяет количество строк или значений поля, выбранных посредством запроса и не являющихся NULL-значениями;
- • **SUM** – вычисляет арифметическую сумму всех выбранных значений данного поля;
- • **AVG –** вычисляет среднее значение для всех выбранных значений данного поля;
- • **MAX** – вычисляет наибольшее из всех выбранных значений данного поля;
- • **MIN** – вычисляет наименьшее из всех выбранных значений данного поля.

```
SELECT AVG (MARK)
FROM EXAM_MARKS;
```

```
SELECT COUNT (DISTINCT SUBJ_ID)
FROM EXAM_MARKS;
```

Агрегирующие функции

- **GROUP BY –** позволяет группировать записи в подмножества, определяемые значениями какого-либо поля, и применять агрегирующие функции уже не ко всем записям таблицы, а раздельно к каждой сформированной группе.
- **HAVING –** определяет критерий, по которому группы следует включать в выходные данные, по аналогии с предложением WHERE, которое осуществляет это для отдельных строк.

```
SELECT STUDENT_ID, SUBJ_ID, MAX (MARK)
FROM EXAM_MARKS
GROUP BY STUDENT_ID, SUBJ_ID;
```

```
SELECT SUBJ_NAME, MAX(HOUR)
FROM SUBJECT
GROUP BY SUBJ_NAME
HAVING MAX (HOUR)>= 72;
```

Сортировка

- **ORDER BY** –позволяет упорядочивать выводимые записи в соответствии со значениями одного или нескольких выбранных столбцов. При этом можно задать возрастающую (**ASC**) или убывающую (**DESC**) последовательность сортировки для каждого из столбцов

```
SELECT SUBJ_NAME, SEMESTER, MAX(HOUR)
FROM SUBJECT
GROUP BY SEMESTER, SUBJ_NAME
ORDER BY SEMESTER DESC, SUBJ_NAME;
```

Сортировка очень сильно нагружает сервер. Использовать нужно только в случае крайней необходимости

Подзапросы

```
SELECT MARK
FROM EXAM_MARKS
WHERE SUBJ_ID =
(SELECT SUBJ_ID
FROM SUBJECT
WHERE SUBJ_NAME= ‘Экология');
```

Как работает запрос SQL со связанным подзапросом?

- Выбирается строка из таблицы, имя которой указано во внешнем запросе.
- Выполняется подзапрос и полученное значение применяется для анализа этой строки в условии предложения WHERE внешнего запроса.
- По результату оценки этого условия принимается решение о включении или не включении строки в состав выходных данных.
- Процедура повторяется для следующей строки таблицы внешнего запроса.

Подзапросы

- Возможно использование подзапроса после SELECT. Однако в этом случае подзапрос должен вывести одну строку
- При использовании подзапросов во внутреннем запросе можно ссылаться на таблицу, имя которой указано в предложении FROM внешнего запроса

```
SELECT NAME, SURNAME, (SELECT UNIV_NAME FROM UNIVERSITY U
WHERE  S.UNIV_ID=U.UNIV_ID) NAZV
FROM STUDENT S;

SELECT *
FROM (SELECT NAME, SURNAME FROM STUDENT  ORDER BY STIPEND)
WHERE ROWNUM<=5;
```

Оператор EXISTS

- **EXISTS** – используя подзапросы в качестве аргумента, этот оператор оценивает результат выполнения подзапроса как истинный если этот подзапрос генерирует выходные данные, то есть в случае *существования (возврата) хотя бы одного найденного значения*
- **NOT EXISTS** – используя подзапросы в качестве аргумента, этот оператор оценивает результат выполнения подзапроса как истинный если этот подзапрос **НЕ** генерирует выходных данных, то есть если ничего не найдено.

```
SELECT DISTINCT STUDENT_ID
FROM EXAM_MARKS A
WHERE EXISTS
(SELECT *
FROM EXAM_MARKS B
WHERE MARK<3
AND A.STUDENT_ID = B.STUDENT_ID);
```

Функции DECODE и CASE

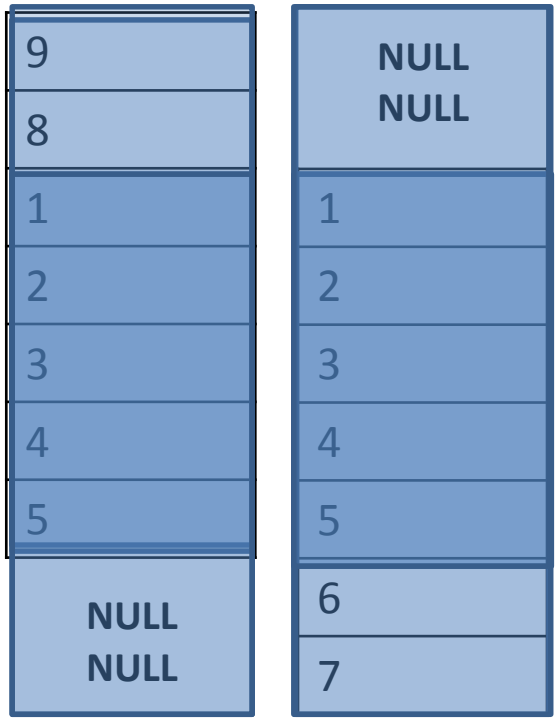
- **DECODE** – сравнивает значения столбца с условием, затем выводит определенный результат
- **DECODE** (столбец, сравниваемое значение столбца, результат1, результат2)

```
SELECT NAME, SURNAME, DECODE (KURS,1, 'ПЕРВОКУРСНИК', 'СТАРШЕКУРСНИК')
FROM STUDENT;
```

- **CASE** – вычисляет некоторое количество условий и возвращает результат, определяемый тем, какое из условий оказалось истинным
- **CASE WHEN** условие **THEN** выражение **ELSE** выражение **END**

```
SELECT NAME, SURNAME, KURS,
CASE
WHEN KURS=1 THEN 'Первокурсник'
WHEN KURS>2 THEN 'Старшекурсник'
ELSE 'Второкурсник'
END Status
FROM STUDENT;
```

Соединения таблиц



JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN

Примеры соединений таблиц

```
SELECT S.NAME, S.SURNAME, J.SUBJ_NAME, E.MARK
FROM STUDENT S, EXAM_MARKS E, SUBJECT J
WHERE S.STUDENT_ID=E.STUDENT_ID (+)
AND E.SUBJ_ID=J.SUBJ_ID
ORDER BY E.MARK DESC;
```

LEFT OUTER JOIN

```
SELECT *
FROM LECTURER L, UNIVERSITY U
WHERE U.CITY(+)=L.CITY;
```

RIGHT OUTER JOIN

```
SELECT DISTINCT L.LECTURER_ID, U.CITY
FROM LECTURER L, UNIVERSITY U
WHERE L.CITY=U.CITY(+)
UNION
SELECT DISTINCT L.LECTURER_ID, U.CITY
FROM LECTURER L, UNIVERSITY U
WHERE L.CITY(+)=U.CITY;
```

FULL OUTER JOIN

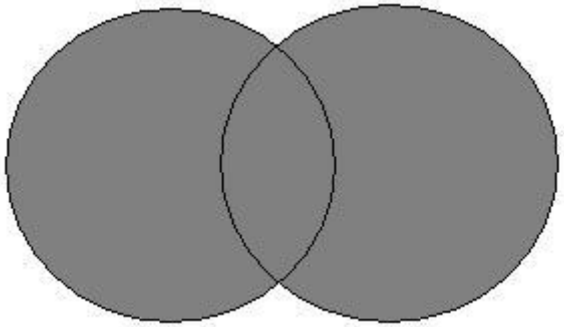
Операторы объединения

- **UNION** – используется для объединения выходных данных двух или более SQL-запросов в единое множество строк и столбцов. Дубликаты не выводит
- **UNION ALL** – используется для объединения выходных данных двух или более SQL-запросов в единое множество строк и столбцов. Дубликаты выводит
- **MINUS** – используется для вывода строк первого запроса, которые не присутствуют во втором запросе
- **INTERSECT** – используется для вывода общих или пересекающихся строк нескольких запросов

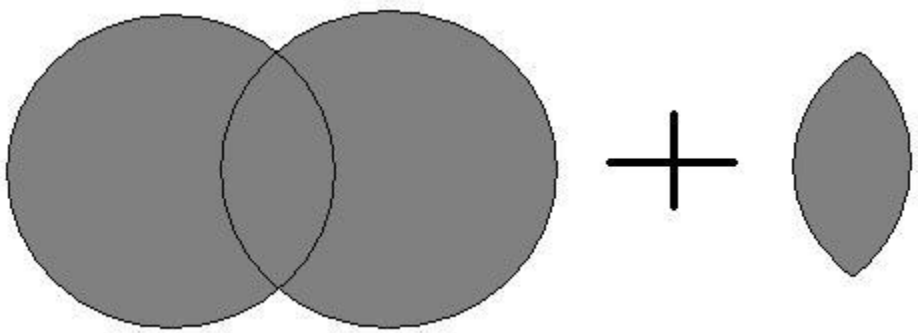
```
SELECT CITY
FROM STUDENT
UNION
SELECT CITY
FROM LECTURER
UNION
SELECT CITY
FROM UNIVERSITY
;
```

Операторы объединения

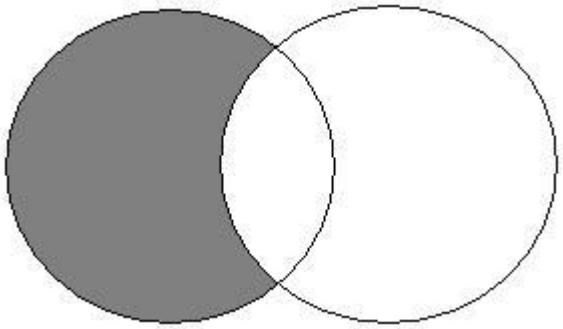
- **UNION**



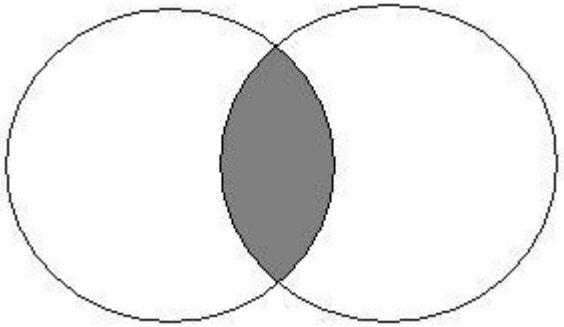
- **UNION ALL**



- **MINUS**



- **INTERSECT**



Нумерация строк

- **ROWNUM** – псевдостолбец, нумерующий строки в возвращаемых результатах.

```
SELECT NAME, STIPEND
FROM (SELECT * FROM STUDENT ORDER BY STIPEND DESC)
WHERE ROWNUM<=5;
```