В каждой компании имеются данные, которые необходимо хранить в каких-либо ресурсах. Например, библиотека хранит списки посетителей, книг, авторов, сроков сдачи этих книг и др. В компании хранится информация о работниках, подразделениях и зарплатах. Эти наборы информации называются данными (**data**).

Организация может хранить данные в различных форматах, таких как hard-copy документ, или данные, которые хранятся в электронных таблицах (spreadsheets), или БД.

**Database** is an organized collection of information.

**Database Management System** is s program that stores, retrieves and modifies data in the database on request.

**Table** is the basic storage structure of DBMS.

**Row** represents all data for a particular object. Каждая строка в таблице может идентифицироваться ключами, для которых не допускается дублирование: **Первичный ключ** (**Primary Key**, может быть только один на таблицу и не допускает пустых (NULL) значений) или **Уникальный ключ** (**Unique Key**, или просто **Key**, может быть несколько в таблице, и может содержать пустые (NULL) значения). Порядок строк в таблице не важен.

**Column** represents one kind of data in a table. Колонка может содержать ключ, по которому идентифицируются данные в строках таблицы, а также может содержать **Внешний ключ** (**Foreign key**, являющийся первичным или обычным ключом из этой же или другой таблицы), который определяет, как таблицы взаимосвязаны одна с другой. Порядок колонок не важен для хранения данных.

Поле (**field** или **cell**) может вовсе не содержать значений, тогда оно называется нулевым (NULL) значением.

**SELECT**

Чтобы получить данные из БД, необходимо использовать выражение **SELECT**, синтаксис которого следующий:

SELECT \*|{[DISTINCT] *column*|*expression* [alias], …} FROM *table*; (знак ‘|’ обозначает выбор из возможных вариантов, квадратные скобки – необязательное значение)

SELECT identifies *what* columns

FROM identifies *which* table

Синтаксис:

SELECT - список одной или нескольких колонок

«\*» - выбрать все колонки

DISTINCT (или UNIQUE в некоторых диалектах SQL) – ликвидирует повторения записей в выборке

*Column*|*expression* – выбирает названную колонку или выражение

*Alias –* дает выбранным колонкам разные заголовки

FROM table – определяет таблицу, содержащую колонки

**Selecting All Columns**

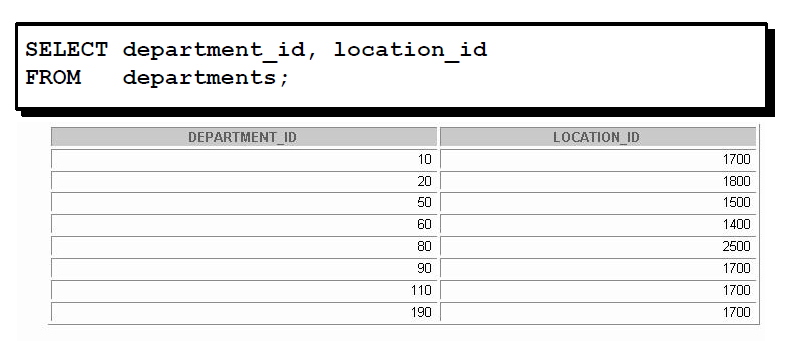
SELECT \*

FROM departments; - select all columns of all rows

**Selecting Specific Columns**

SELECT department\_id, location\_id

FROM departments;



**Writing SQL Statements**

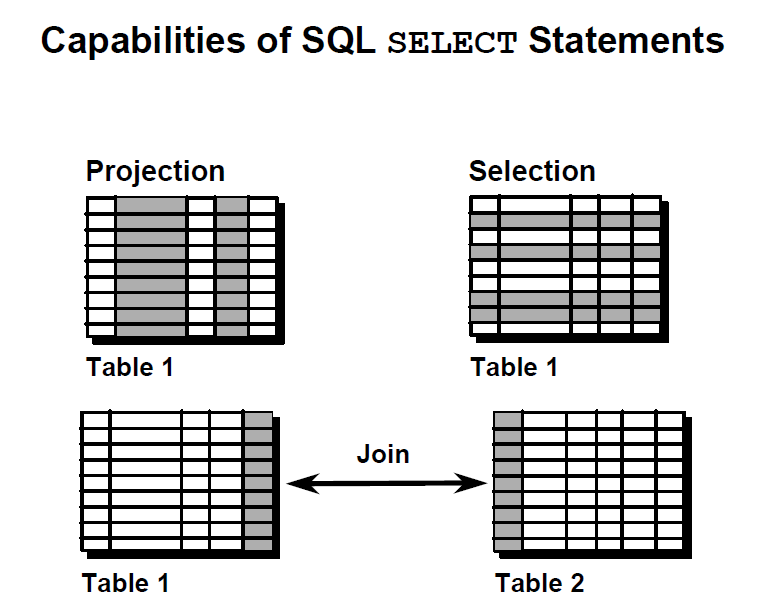
* предложения SQL не чувствительны к регистру;
* могут располагаться на одной или нескольких строках;
* перенос строк разрешен;
* ключевые слова нельзя сокращать, разрывать или переносить на другие строки;
* ключевые слова обычно пишутся в верхнем регистре; все другие слова, например, названия таблиц и колонок обозначаются в нижнем регистре.

SELECT позволяет получить даные из БД, а именно выполянть:

- **Projection**: проекция используется для того, чтобы выбрать колонки таблицы: одну или несколько.

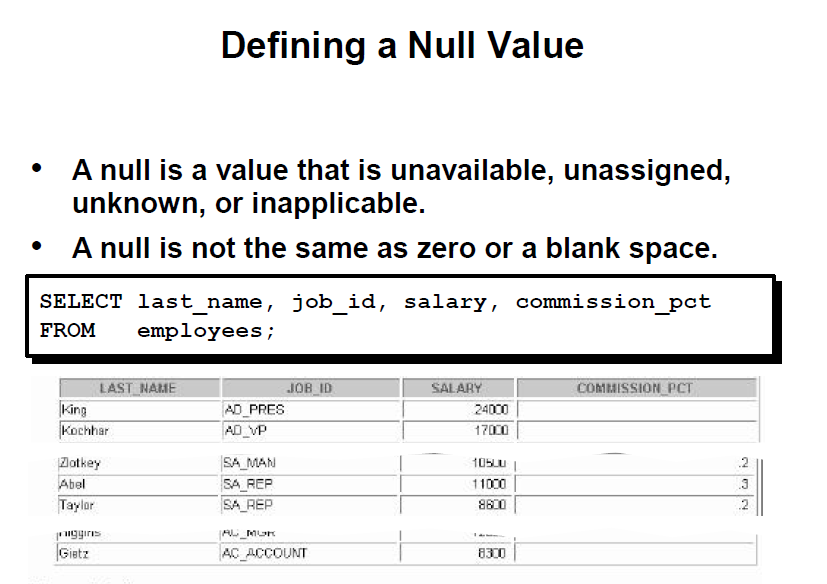
**- Selection**: чтобы выбрать определенные строки, для этого также можно использовать разные критерии выборки.

**- Joining**: чтобы соединять данные из разных таблиц, указывая линки (взаимосвязи) между ними.



**Defining NULL value**

NULL является значением, используемым для обозначения ячеек, не имеющих какого-либо значения. Это не то же самое, что ноль (число), пробел (символ), или пустая строка (строка, не содержащая символов).



В столбце COMMISSION\_PCT таблицы EMPLOYEES только sales manager or sales representative может получить комиссию, а другие нет. Об этом говорит NULL значение.

*Арифметическое выражение, содержащее значение NULL, возвращает тоже NULL.*

**Defining a Column Alias (Синоним)**

* переименовывает название столбца;
* полезно использовать совместно с вычисляемыми столбцами в выборке;
* стоит сразу после названия столбца или вычисляемого выражения, также может использоваться ключевое слово AS для визуального разделения колонки и alias-а;
* если нужно поставить пробел или другие специальные символы в синониме, или учитывать регистр, то его нужно брать в двойные кавычки.

**Using Column Aliases**

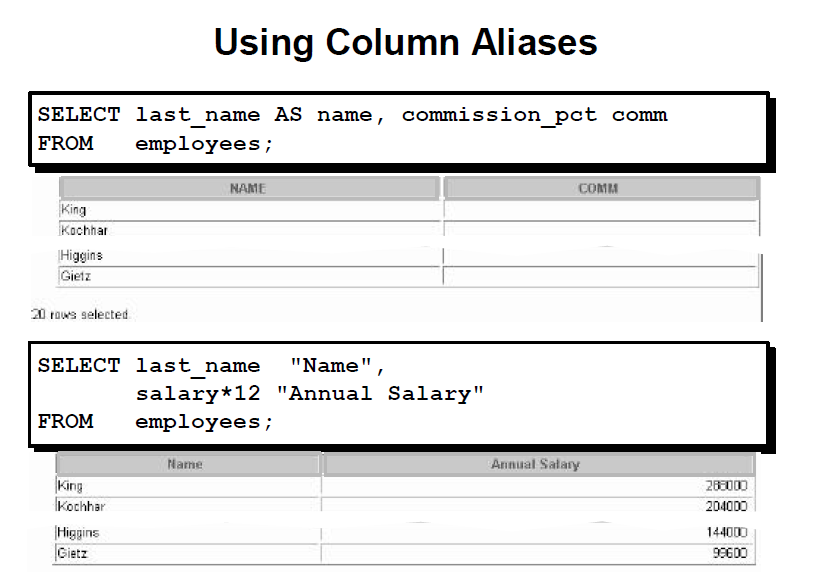
SELECT last\_name AS name, commission\_ptc comm

FROM employees;

SELECT last\_name “Name”

Salary\*12 “Annual Salary”

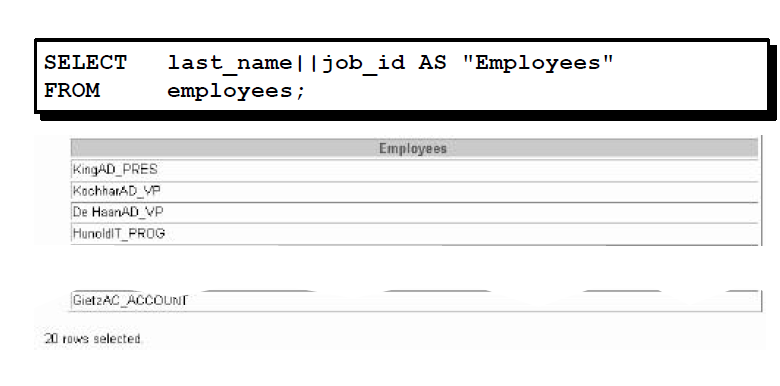
FROM employees;



**Concatenation Operator**

SELECT last\_name||job\_id AS “Employees”

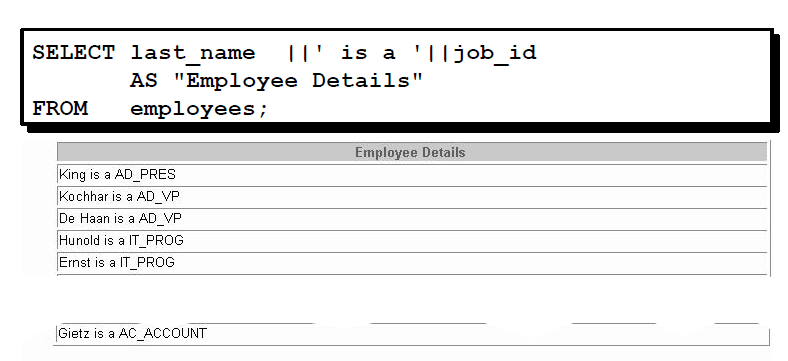
FROM employees;



**Literal Character Strings**

* Literal – это символ (character), число (number), или дата (date), включенный в список SELECT.
* Значения Date и Character литералов должны быть заключены в одинарные кавычки.

SELECT last\_name ||’ is a ‘||job\_id AS “Employee Details”

FROM employees;

**Eliminating Duplicate Rows**

Осуществляется с помощью слова DESTINCT в операторе SELECT.

**Limiting the Rows SELECTED.**

Using WHERE clause.

SELECT \*|{[DISTINCT] column|expression [alias],…}

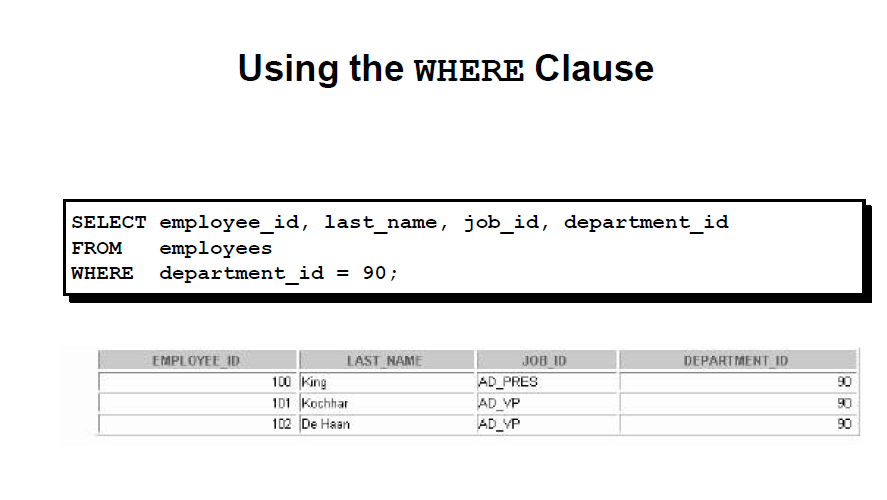
FROM table

[WHERE condition(s)];

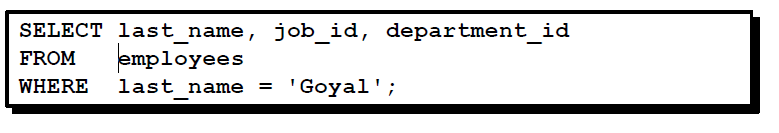
WHERE содержит какое-либо условие, оно следует сразу за FROM. Если условие выполняется, строка, которая состветствует условию, отображается.

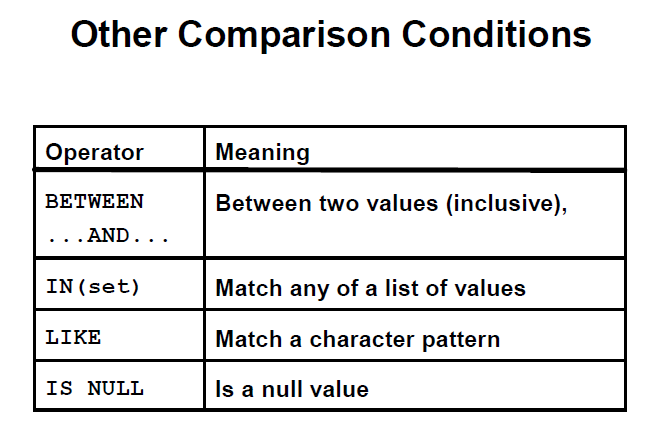
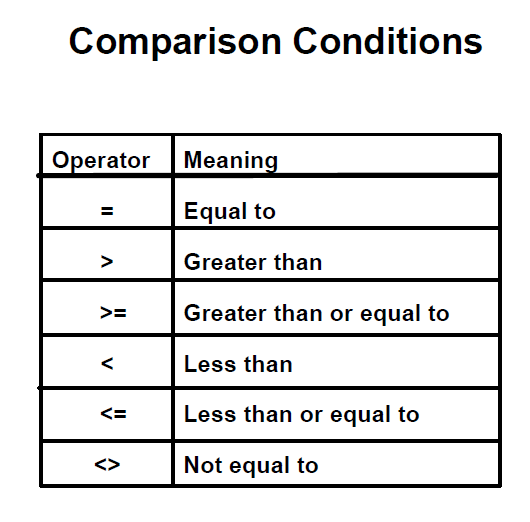
Синтаксис:

WHERE conditions – где «conditions» содержат названия колонок, выражения, константы и операторы сравнения.



* Character strings и date values заключаются в одинарные кавычки;
* Character values чувствительны к регистру, значения dates чувствительны к формату.





Syntax

…WHERE expr operator value

For Example

… WHERE hire\_date=’01-JAN-95’

… WHERE salary>=6000

… WHERE last\_name=’Simth’

Alias не могут быть использованы в WHERE.

Символы != и ^= используются для не равно.

**Using the LIKE condition**

LIKE представляет поиск по шаблону.

Условия поиска могут содержать как буквенные символы, так и численные.

- % - обозначает множество симоволов или ни одного;

- \_ - обозначает один символ.

SELECT first\_name

FROM employees

WHERE first\_name LIKE ‘S%’

Бывает, что вы часто не знаете полностью значение того выражения, которое ищете. Иногда BETWEEN можно заменить LIKE. В следующем примере отображаются last name и hire dates всех employees которых наняли между январем 1995 и декабрем 1995:

SELECT last\_name, hire\_date

FROM employees

WHERE hire\_date LIKE ‘%95’;

Также можно комбинировать «%» и «\_»:

SELECT last\_name

FROM employees

WHERE last\_name LIKE ‘\_o%’

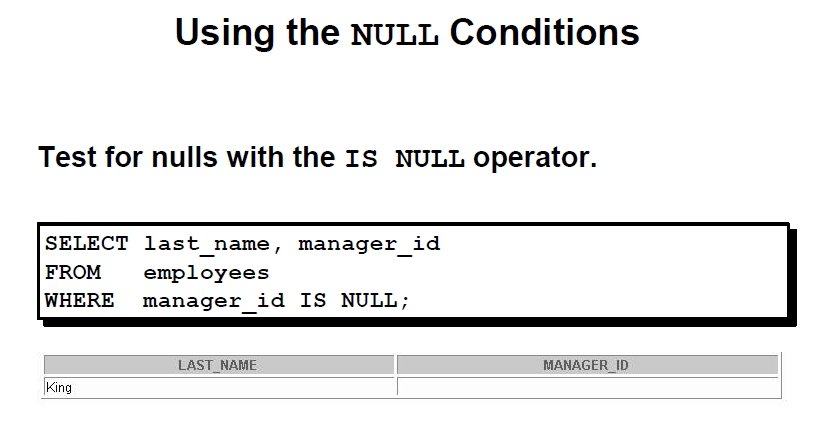
Когда нужно использовать % и \_ как символы, нужно испоьзовать ESCAPE. Эта опция определяет, что нужно считать симоволом escape. Если нужно искать строки, которые содержат ‘SA\_’, вы можете использовать следующее:

SELECT employee\_id, last\_name, job\_id

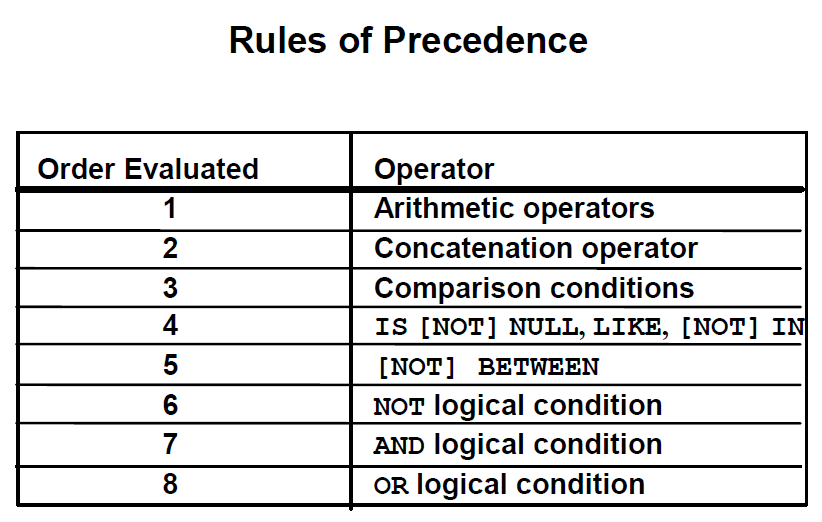
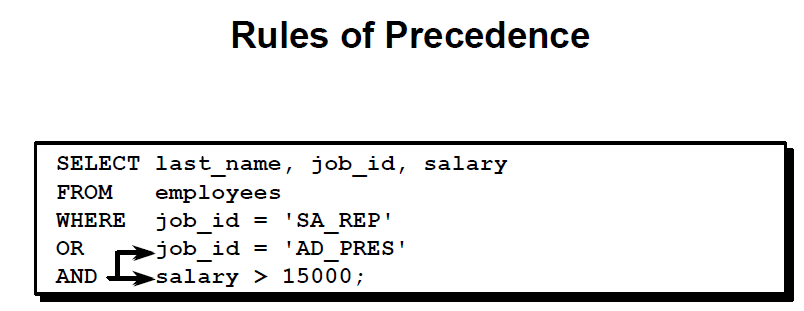
FROM employees

WHERE job\_id LIKE ‘%SA\\_%’ ESCAPE ‘\’;

Команда ESCAPE опеределяет, что (\) является escape character-ом, т.е. символ, следующий за этим escape-символом, будет восприниматься именно, как символ литерала, а не как символ подстановки.



Т.к. результат любой операции со значением NULL тоже будет NULL, то для проверки значений таких ячеек нельзя использовать операции вида “type = NULL”, нужно пользоваться операторами “IS NULL”, “IS NOT NULL”

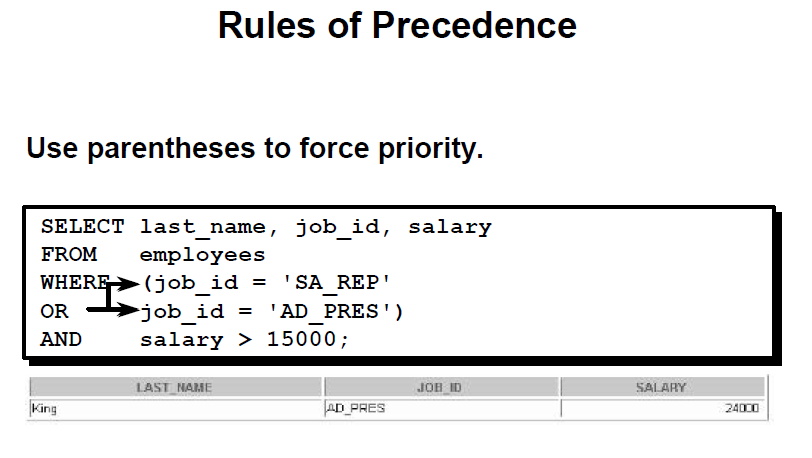


В этом примере есть 2 условия:

- job ID is AD\_PRES AND the salary is greater than 15,000.

- second: job ID is SA\_REP.

Данное условие SELECT читается как следующее: «Select the row if an employee is a president AND earns more than $15,000, OR if the employee is a sales representative.»

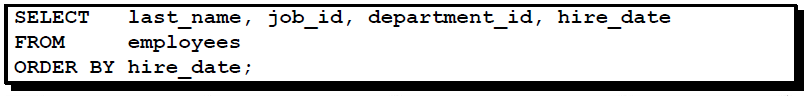


* The first condition is that the job ID is AD\_PRES OR SA\_REP
* The second condition is that salary is greater than $15,000.

Это выражение читают так:

“Select the row if an employee is a president or a sales representative AND if the employee earns more than $15,000”.

**ORDER BY**

* Sort rows with the ORDER BY clause:
* ASC: ascending order;
* DESC: descending order

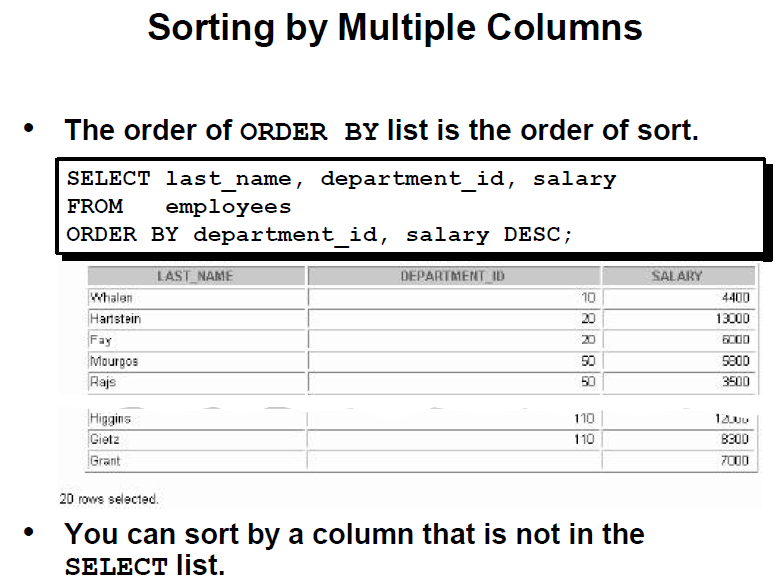
Синтакс:

SELECT expr

FROM table

[WHERE condition(s)]

[ORDER BY {column, expr} [ASC|DESC]];



**SQL Functions**

* Character: на вход подаются символы, могут возврашать как символьные значения, так и численные
* Number functions: на входе численные значения, на выходе тоже численные
* Date functions: оперируют с данными типа Date
* Conversions: конвертируют значение из одного типа в другой

**JOINS**

CROSS\_JOIN - Всевозможные сочетания из обеих таблиц. Условие для этого типа оператора JOIN не указывается.

**INNER JOIN (SIMPLE, EQUIJOIN, NATURAL)**

Объединяет две таблицы, где каждая строка обеих таблиц в точности соответствует условию. Если для строки одной таблицы не найдено соответствия в другой таблице, строка **не включается** в набор.

### OUTER JOIN

Присоединение таблицы с необязательным присутствием записи в таблице.

#### LEFT OUTER JOIN

К левой таблице присоединяются все записи из правой, соответствующие условию (по правилам **inner join**), плюс все не вошедшие записи из левой таблицы, поля правой таблицы заполняются значениями *NULL*.

#### RIGHT OUTER JOIN

Аналогично **left outer join**, но применяется для правой таблицы.

К правой таблице присоединяются все записи из левой, соответствующие условию (по правилам **inner join**), плюс все не вошедшие записи из правой таблицы, поля левой таблицы заполняются значениями *NULL*.

#### FULL OUTER JOIN

К левой таблице присоединяются все записи из правой, соответствующие условию (по правилам **inner join**), плюс все не вошедшие записи из правой таблицы, поля левой таблицы заполняются значениями *NULL* и плюс все не вошедшие записи из левой таблицы, поля правой таблицы заполняются значениями *NULL.*