Школа программирования
DECODE

# SQL basics.
# Урок 1

Преподаватель: Данияр Султан

# Что изучим на курсе?

- **Введение в SQL:** концепции, реляционная модель, инсталляция SQL Server, создание БД, таблиц,
  виды отношений, типы данных
- **Простые выборки:** SELECT, DISTINCT, COUNT, WHERE, AND / OR, BETWEEN, IN, ORDER BY, MIN/MAX/AVG, LIKE, TOP, GROUP BY, HAVING, UNION/INTERSECT/EXCEPT, проверки на NULL
- **Соединения:** INNER, LEFT, RIGHT, SELF
- **Подзапросы:** WHERE EXISTS, ANY, ALL
- **DDL:** создание базы данных, таблиц, управление ключами (PK, FK), ограничения, INSERT, UPDATE/DELETE
- **Проектирование БД:** основы, рекомендации, нормальные формы (НФ)
- **Представления (Views):** основы, создание, обновления через views, опция check
- **Логика** с CASE WHEN, COALESCE и NULLIF
- Познакомимся с индексами, операциями DCL и TCL

# Что такое база данных?

**База данных** – организованная структура для хранения и обработки данных. Характеристики:

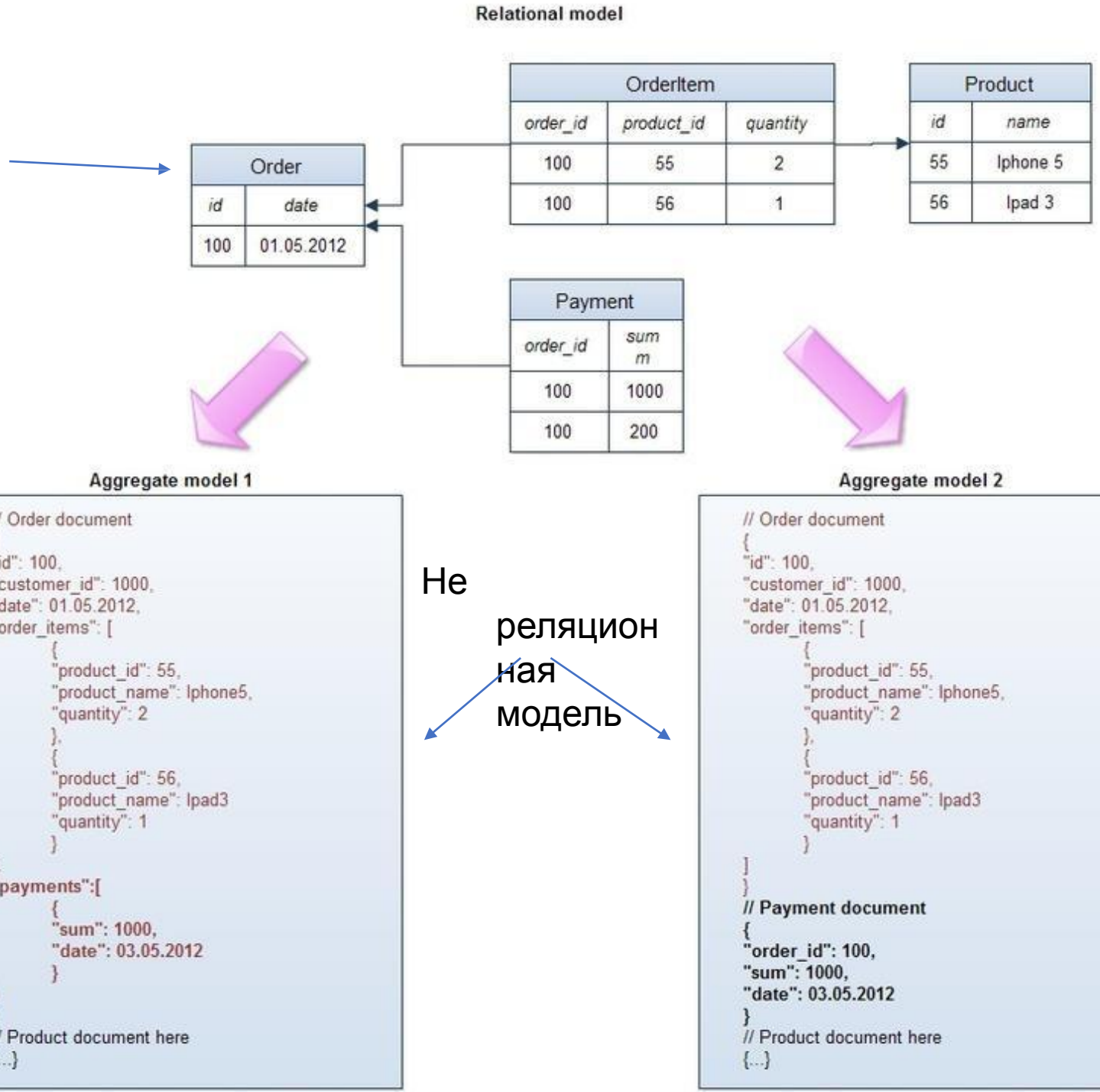
- Созданы для хранения миллионов строк (в отличии от spreadsheets)
- Лимитированы (ограничены) объемом памяти на жестком диске компьютера
- Оптимизированы таким образом, чтобы использовать всю память компьютера для улучшения performance

Виды БД:

-реляционные (SQL) – логическая модель данных, описывающая структуры данных в виде таблиц.

-не реляционные (NoSQL) - не используется табличная схема строк и столбцов (JSON).

Реляционная модель



# Что такое СУБД?

- **СУБД – система управления базами данных.** Это комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать).
- Виды:

-файл-серверные (Microsoft access) файлы данных располагаются централизовано на файл-
сервере, а СУБД на каждом клиентском компьютере.

-клиент-серверные (MySQL, PostgreSQL, Oracle, MS SQL) и СУБД и файлы данных располагаются на  сервере

-встраиваемые: SQLite

-реляционные (MS SQL, PostgreSQL, Oracle и др.) / не реляционные (MongoDB и др.)

- Для взаимодействия с СУБД используется программа/GUI (graphical user interface), как  например SQL Server Management Studio, Workbench

# На этом курсе мы будем изучать:

- Реляционную базу данных
- СУБД: MS SQL SERVER
- GUI SQL server management studio
- Transact-SQL

# Реляционная модель

- Сущность – клиенты, заказы, сотрудники, поставщики
  - Таблица – отношение
  - Столбец –атрибуты
  - Строка/запись – кортеж
  - Результоующий набор (result set)

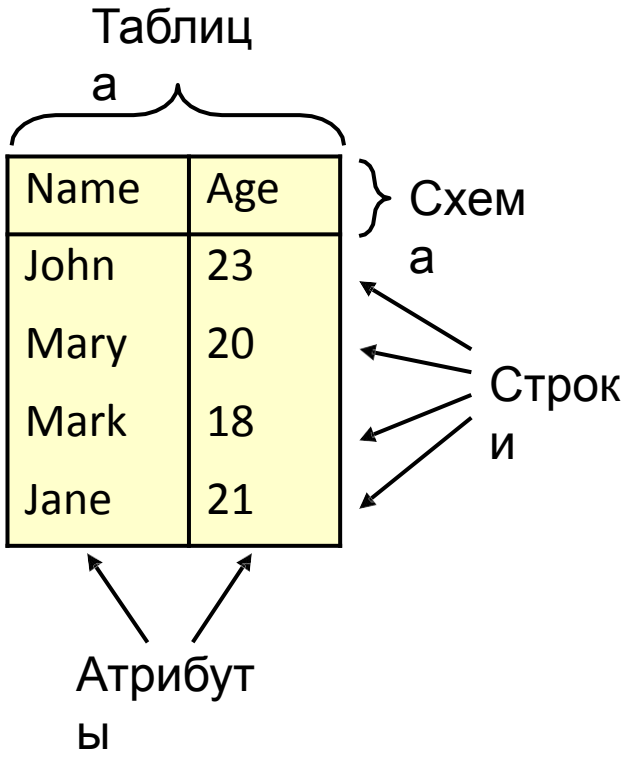
Результат запроса SQL:

```
SELECT TOP(13) contact_name, address, city
FROM customers
```

| | contact_name character varying (30) | address character varying (60) | city character varying (15) |
|---|---|---|---|
| 1 | Maria Anders | Obere Str. 57 | Berlin |
| 2 | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. |
| 3 | Antonio Moreno | Mataderos 2312 | México D.F. |
| 4 | Thomas Hardy | 120 Hanover Sq. | London |
| 5 | Christina Berglund | Berguvsvägen 8 | Luleå |
| 6 | Hanna Moos | Forsterstr. 57 | Mannheim |
| 7 | Frédérique Citeaux | 24, place Kléber | Strasbourg |
| 8 | Martín Sommer | C/ Araquil, 67 | Madrid |
| 9 | Laurence Lebihan | 12, rue des Bouchers | Marseille |
| 10 | Elizabeth Lincoln | 23 Tsawassen Blvd. | Tsawassen |
| 11 | Victoria Ashworth | Fauntleroy Circus | London |
| 12 | Patricio Simpson | Cerrito 333 | Buenos Aires |
| 13 | Francisco Chang | Sierras de Granada 9993 | México D.F. |

# Реляционная модель:

- Данные хранятся в таблицах
- Таблицы состоят из атрибутов (столбцов)
- Данные имеют форму строки
- Порядок строк не важен
- Не должно быть повторяющихся строк
- У каждой таблицы есть схема (иногда заголовком)
- Схема определяет атрибуты таблицы (столбцы).



# Пример таблицы:

Атрибуты: ID, Name, Salary and Department

Степень(degree) таблицы: 4

| ID | Name | Salary | Department |
|---|---|---|---|
| M139 | John Smith | 18,000 | Marketing |
| M140 | Mary Jones | 22,000 | Marketing |
| A368 | Jane Brown | 22,000 | Accounts |
| P222 | Mark Brown | 24,000 | Personnel |
| A367 | David Jones | 20,000 | Accounts |

Схема:{ ID, Name, Salary, Department }

Строки:
{ (ID, A368),
(Name, Jane Brown),
(Salary, 22,000),
(Department, Accounts)}

Уникальность данных (cardinality) таблицы: 5

# Атрибуты и домены

- Домен дается для каждого атрибута
- В домене перечислены возможные значения атрибута

Примеры:

- "Возраст" может быть получен из набора целых чисел от 0 до 150.
- «Отдел» должен храниться в виде varchar, не более 250 символов.
- Атрибут «Комментарии» ограничений не имеет

# Потенциальные ключи

Набор атрибутов в таблице либо атрибут является **потенциальным ключом** тогда и только
тогда, когда:

- Каждая строка имеет уникальное значение для этого набора атрибутов: уникальность
- Минимализм
- Среди них выбирается первичный ключ

***ения?***

| officeID | Name | Country | Postcode/Zip | Phone |
|---|---|---|---|---|
| O1001 | Headquarters | England | W1 1AA | 0044 20 1545 3241 |
| O1002 | R&D Labs | England | W1 1AA | 0044 20 1545 4984 |
| O1003 | US West | USA | 94130 | 001 415 665981 |
| O1004 | US East | USA | 10201 | 001 212 448731 |
| O1005 | Telemarketing | England | NE5 2GE | 0044 1909 559862 |
| O1006 | Telemarketing | USA | 84754 | 001 385 994763 |

# Определить потенциальные ключи

Потенциальные ключи: {OfficeID}, {Phone} and {Name, Postcode/Zip}

| officeID | Name | Country | Postcode/Zip | Phone |
|---|---|---|---|---|
| O1001 | Headquarters | England | W1 1AA | 0044 20 1545 3241 |
| O1002 | R&D Labs | England | W1 1AA | 0044 20 1545 4984 |
| O1003 | US West | USA | 94130 | 001 415 665981 |
| O1004 | US East | USA | 10201 | 001 212 448731 |
| O1005 | Telemarketing | England | NE5 2GE | 0044 1909 559862 |
| O1006 | Telemarketing | USA | 84754 | 001 385 994763 |

Следующий ключ тоже уникален {Name, Country, Phone} ,
но не подходит по критерию МИНИМАЛИЗМ

# Первичный ключ (Primary key)

- Обычно выбирается один потенциальный ключ для идентификации строк в таблицу.
- Это называется первичным ключом.
- Часто в качестве первичного ключа используется специальный идентификатор

| ID | First | Last |
|---|---|---|
| S139 | Alan | Carr |
| S140 | Jo | Brand |
| S141 | Alan | Davies |
| S142 | Jimmy | Carr |

Первичный ключ:
{ID}

# NULLs и первичные ключи

- Отсутствующая информация  может быть представлена с помощью NULL.
- NULL указывает на отсутствующее или неизвестное  значение.

- *Целостность данных*

  Первичные ключи не  могут содержать значения NULL

# Внешние ключи

- Внешние ключи используются для связывания данных в  таблицах.
- Атрибут в ссылающейся таблице является внешним  ключом, если его значение:
  - Соответствует значению Первичного ключа во второй таблице
  - Это называется ссылочной целостностью данных
  - Может быть неуникальным и иметь NULL значения

# Пример внешнего ключа

Департамент

| DID | DName |
|---|---|
| 13 | Marketing |
| 14 | Accounts |
| 15 | Personnel |

{DID} это первичный ключ
таблицы Департамент
– Каждая строка имеет уникальное значение {DID}

Сотрудн

| EID | EName | DID |
|---|---|---|
| 15 | John Smith | 13 |
| 16 | Mary Brown | 14 |
| 17 | Mark Jones | 13 |
| 18 | Jane Smith | NULL |

{DID} - это внешний ключ в таблице Сотрудники
- значение DID каждого сотрудника либо NULL, либо соответствует значению атрибута DID в таблице Департамент. Это связывает каждого сотрудника не более чем с одним департаментом

# Типы связей между таблицами

Связи делятся на:

- **Многие ко многим -** реализуется в том случае, когда нескольким  объектам из таблицы А может соответствовать несколько объектов из  таблицы Б, и в тоже время нескольким объектам из таблицы Б
  соответствует несколько объектов из таблицы А.

- **Один ко многим** -реализуется тогда, когда объекту А может принадлежать или же соответствовать несколько объектов Б, но  объекту Б может соответствовать только один объект А

- **Один к одному** – самая редко встречаемая связь между таблицами.  Если вы видите такую связь, то можно объединить

# SQL –structured query language

# SQL

- Structured query language – структурированных язык запросов
- Непроцедурный язык и не язык общего назначения
- Если необходимо реализовать процедурную логику – нужен другой язык (Python, java, c++…)
- ANSI SLQ-92
- У каждого СУБД свой диалект (T-SQL в SQL server, PL в Oracle)
- Результатом SQL запроса является результирующий набор (как правило – таблица)

# Подмножества SQL

Запрос типа «выбор»:

- DML ( Data Manipulation Language) – позволяет запрашивать и манипулировать данными (SELECT, INSERT, UPDATE, DELETE*, MERGE)

Запрос типа «действие»:

- DDL (Data Definition Language) – позволяет создавать и изменять объекты в базе (CREATE, ALTER,
  DROP, а также TRUNCATE, USE)
- DCL ( Data Control Language) – позволяет контролировать доступ к базе данных (Grant, Revoke).
- TCL (Transaction Control Language) – обозначает начало и конец транзакции (BEGIN TRANSACTION, COMMIT, ROLLBACK)

# Операторы

# DDL

- DDL –язык описания данных. Предназначен для работы с  объектами базы данных, для изменения структуры БД.

| ОПЕРАТОР | ЗНАЧЕНИЕ |
|---|---|
| CREATE | Создание объекта (БД, таблицы, view, триггера) |
| ALTER | Изменение структуры объекта |
| DROP | Удаление объекта |
| USE | Для выбора нужной БД |

# DDL

- CREATE DATABASE test;  - создание новой базы test
- USE test – Пишется в начале запроса, чтобы указать с какой именно  базой работаем в текущем запросе.
- CREATE TABLE table_name (

  Имя_атрибута типа_данных [NOT NULL][UNIQUE][DEFAULT value],

  Имя_атрибута типа_данных [NOT NULL][UNIQUE][DEFAULT value]

  ….

  PRIMARY KEY (Имя_атрибута),

  FOREIGN KEY (Имя_атрибута) REFERENCES имя_таблицы

  (Имя_атрибута)  ON UPDATE action [restrict][cascade]

  ON DELETE action[restrict][cascade]

- )

# CREATE

# пример

```
CREATE TABLE Employees (
    ID int,
    Name
    nvarchar(255),
    Birthday date,
    Email nvarchar(30),
    Position nvarchar(30),
    Department
    nvarchar(30)
)
```

# DROP удаление объекта из БД

| ОПЕРАТОР | ЗНАЧЕНИЕ |
|---|---|
| **DROP DATABASE** Test | Удаление Базы данных |
| **DROP TABLE** Employees, Customers, Orders | Удаление таблицы |
| **ALTER TABLE** Employees **DROP COLUMN** Age | Удаление столбца из таблицы |
| **ALTER TABLE** Employees **DROP CONSTRAINT** PK_Employees | Удаление ограничения из таблицы |

- DROP TABLE не сработает, если вы пытаетесь удалить таблицу, в которой есть хотя бы одно поле на которое ссылается другая таблица с помощью FOREIGN KEY. Сначала нужно будет удалить все referencing FOREIGN KEY в других таблицах, и только потом вы сможете удалить таблицу.

# DML. INSERT –добавление данных

**INSERT INTO** Название_таблицы (Столбец1, Столбец2, Столбец3, ...)

**VALUES** (Данные1, Данные2, Данные3, ...);

**INSERT INTO** Employees(ID, Name, Birthday, Email, Position, Department)

**VALUES** (100, ‘Arman’, ’01-01-1990’, ‘arman@gmail.com’, ‘Director’, ‘HR’);

* Порядок значений обязательно должен совпадать с порядком колонок

# SELECT – оператор DML для получения данных из БД

```
SELECT [DISTINCT] select_expr, ...
FROM table_references
[WHERE where_condition]
[ORDER BY col_name [ASC | DESC]]
[LIMIT row_count]
```

-После SELECT пишутся столбцы, которые мы хотим вытащить из таблиц
-SELECT * означает вытащить все столбцы из таблиц
-после FROM пишем название таблицы, где хранятся данные
-после WHERE пишем условия, которые фильтруют результат запроса
-ORDER BY сортирует запрос по определенному столбцу по возрастанию/убыванию
-LIMIT – позволяет ставить ограничения в выгрузке

# Пример запроса SELECT

```
USE [HR ] --пишем запрос внутри БД HR
SELECT * --вытаскиваем все данные из таблицы, все столбцы
FROM employees --таблица employees
```

Результат запроса:

Results | Messages

| | employee_id | first_name | last_name | email | phone_number | hire_date | job_id | salary | commission_pct | manager_id | department_id |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | Steven | King | SKING | 515.123.4567 | 17-JUN-1987 | AD_PRES | 24000.00 | NULL | NULL | 90 |
| 2 | 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 21-SEP-1989 | AD_VP | 17000.00 | NULL | 100 | 90 |
| 3 | 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 13-JAN-1993 | AD_VP | 17000.00 | NULL | 100 | 90 |
| 4 | 103 | Alexander | Hunold | AHUNOLD | 590.423.4567 | 03-JAN-1990 | IT_PROG | 9000.00 | NULL | 102 | 60 |
| 5 | 104 | Bruce | Ernst | BERNST | 590.423.4568 | 21-MAY-1991 | IT_PROG | 6000.00 | NULL | 103 | 60 |
| 6 | 105 | David | Austin | DAUSTIN | 590.423.4569 | 25-JUN-1997 | IT_PROG | 4800.00 | NULL | 103 | 60 |
| 7 | 106 | Valli | Pataballa | VPATABAL | 590.423.4560 | 05-FEB-1998 | IT_PROG | 4800.00 | NULL | 103 | 60 |
| 8 | 107 | Diana | Lorentz | DLORENTZ | 590.423.5567 | 07-FEB-1999 | IT_PROG | 4200.00 | NULL | 103 | 60 |
| 9 | 108 | Nancy | Greenberg | NGREENBE | 515.124.4569 | 17-AUG-1994 | FI_MGR | 12000.00 | NULL | 101 | 100 |
| 10 | 109 | Daniel | Faviet | DFAVIET | 515.124.4169 | 16-AUG-1994 | FI_ACCOUNT | 9000.00 | NULL | 108 | 100 |
| 11 | 110 | John | Chen | JCHEN | 515.124.4269 | 28-SEP-1997 | FI_ACCOUNT | 8200.00 | NULL | 108 | 100 |
| 12 | 111 | Ismael | Sciarra | ISCIARRA | 515.124.4369 | 30-SEP-1997 | FI_ACCOUNT | 7700.00 | NULL | 108 | 100 |

# Пример запроса SELECT

```
USE [HR] --пишем запрос внутри БД HR
SELECT first_name, last_name, salary --вытаскиваем только перечисленные столбцы
FROM employees --таблица employees
```

Результата запроса:

Results | Messages

| | first_name | last_name | salary |
|---|---|---|---|
| 1 | Steven | King | 24000.00 |
| 2 | Neena | Kochhar | 17000.00 |
| 3 | Lex | De Haan | 17000.00 |
| 4 | Alexander | Hunold | 9000.00 |
| 5 | Bruce | Ernst | 6000.00 |
| 6 | David | Austin | 4800.00 |
| 7 | Valli | Pataballa | 4800.00 |
| 8 | Diana | Lorentz | 4200.00 |
| 9 | Nancy | Greenberg | 12000.00 |
| 10 | Daniel | Faviet | 9000.00 |
| 11 | John | Chen | 8200.00 |
| 12 | Ismael | Sciarra | 7700.00 |

# Пример запроса SELECT с WHERE

```
SELECT first_name, last_name, salary
FROM employees

WHERE department_id = 90 --вытащим инфо о сотрудникам, которые
работают
в отделе с ID=90
```

Р

| | first_name | last_name | salary | department_id |
|---|---|---|---|---|
| 1 | Steven | King | 24000.00 | 90 |
| 2 | Neena | Kochhar | 17000.00 | 90 |
| 3 | Lex | De Haan | 17000.00 | 90 |

# WHERE и операторы сравнения

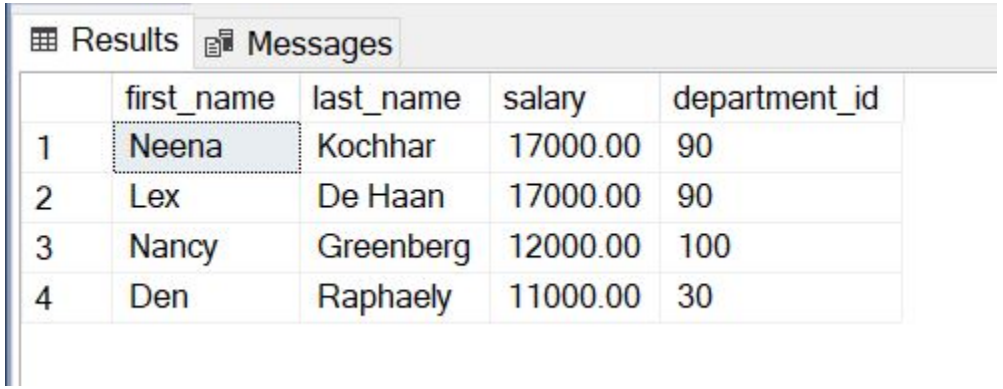
| Оператор | Описание |
|---|---|
| = | Равно |
| <> или != | Не равно |
| < | Меньше |
| > | Больше |
| <= | Меньше или равно |
| >= | Больше или равно |
| ANY | Сравнивает значение с любыми значениями из списка |
| SOME | Идентично оператору ANY; используется реже, чем ANY |
| ALL | Сравнивает значение со всеми значениями в списке. |

# Between… and –проверяет лежит ли значение в интервале

```
--пишем запрос который вытащит инфо о сотрудниках с зарплатой между 10000 и 23000
SELECT first_name, last_name, salary,
department_id  FROM employees

WHERE salary between 10000 and 23000
--это же условие можно переписать как: salary >= 10000 and salary<= 23000
```

Результата запроса:



Results | Messages

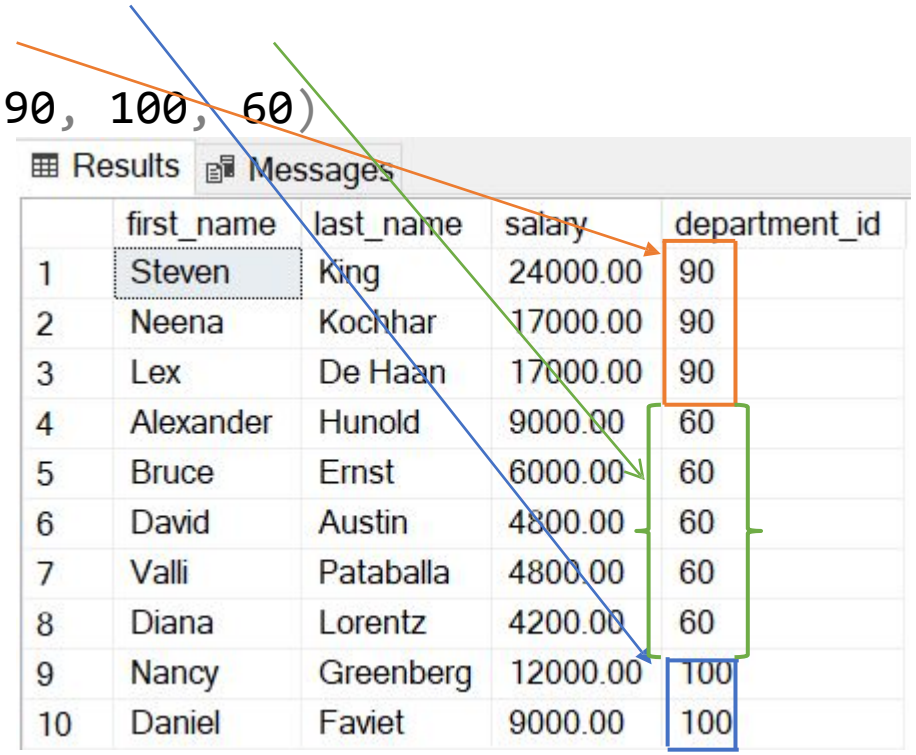
| | first_name | last_name | salary | department_id |
|---|---|---|---|---|
| 1 | Neena | Kochhar | 17000.00 | 90 |
| 2 | Lex | De Haan | 17000.00 | 90 |
| 3 | Nancy | Greenberg | 12000.00 | 100 |
| 4 | Den | Raphaely | 11000.00 | 30 |

# IN–проверяет равен ли одному из значение внутри IN

```
--пишем запрос который вытащит инфо о сотрудниках, которые работают в отделе
с
ID=60, ID=90, ID =100

SELECT first_name, last_name, salary, department_id
FROM employees
WHERE department_id IN (90, 100, 60)
```

Результата запроса:



Results | Messages

| | first_name | last_name | salary | department_id |
|---|---|---|---|---|
| 1 | Steven | King | 24000.00 | 90 |
| 2 | Neena | Kochhar | 17000.00 | 90 |
| 3 | Lex | De Haan | 17000.00 | 90 |
| 4 | Alexander | Hunold | 9000.00 | 60 |
| 5 | Bruce | Ernst | 6000.00 | 60 |
| 6 | David | Austin | 4800.00 | 60 |
| 7 | Valli | Pataballa | 4800.00 | 60 |
| 8 | Diana | Lorentz | 4200.00 | 60 |
| 9 | Nancy | Greenberg | 12000.00 | 100 |
| 10 | Daniel | Faviet | 9000.00 | 100 |