SQL

База данных

База данных (БД) — имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

Данными называют зарегистрированную информацию, представление фактов, понятий или инструкций в форме, которая подходит для передачи, связи, обработки человеком или с помощью машины.

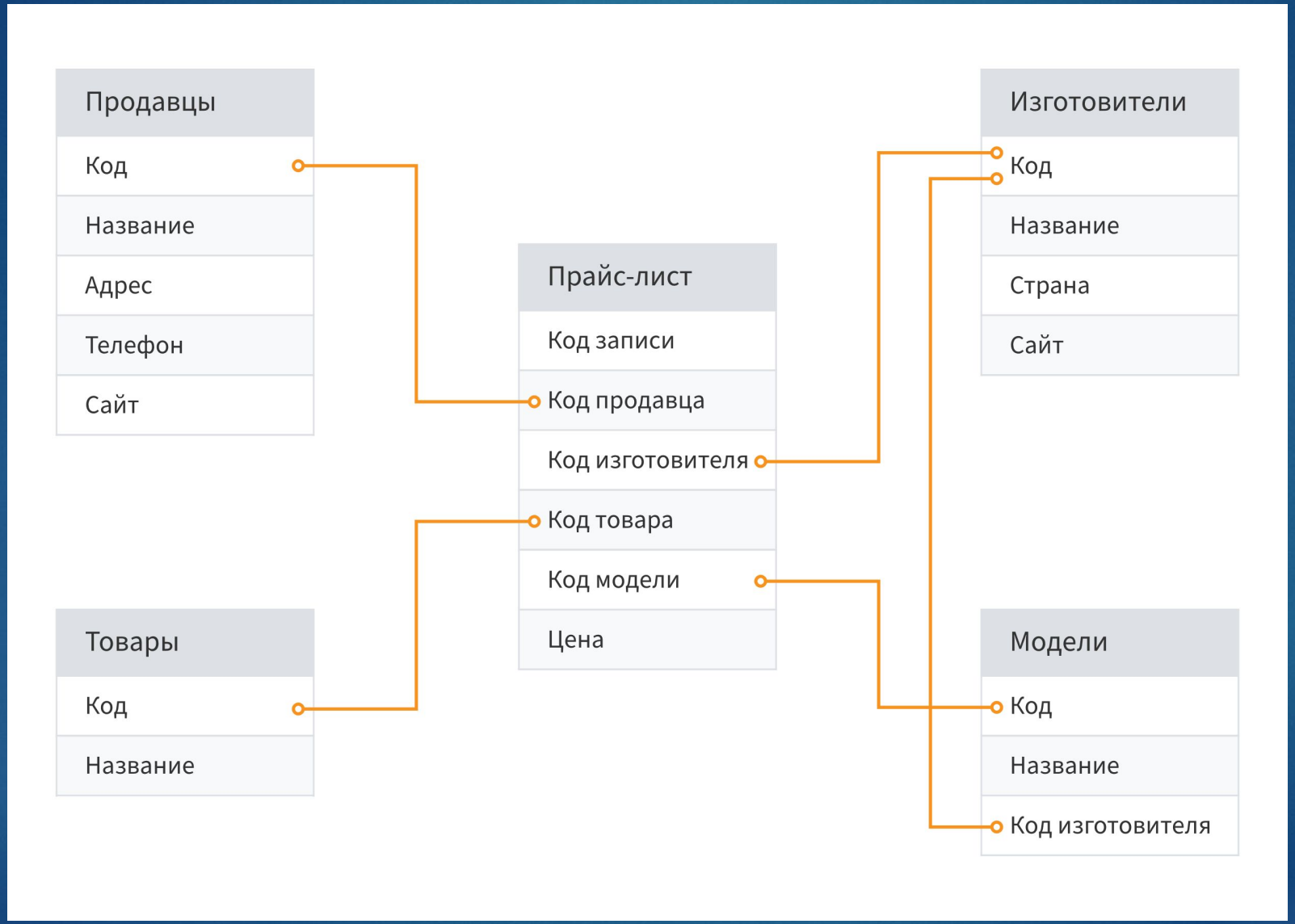
Система управления базами данных

Система управления базами данных (СУБД) — программное обеспечение, которое необходимо для создания, редактирования и обслуживания файлов БД. С его помощью можно упростить процесс работы — от ввода данных до отчетности. Кроме того, система управления базами данных помогает выполнять резервное копирование, поддерживать безопасность, предоставлять общий доступ к БД. СУБД позволяет работать с базами данных одновременно нескольким пользователям.

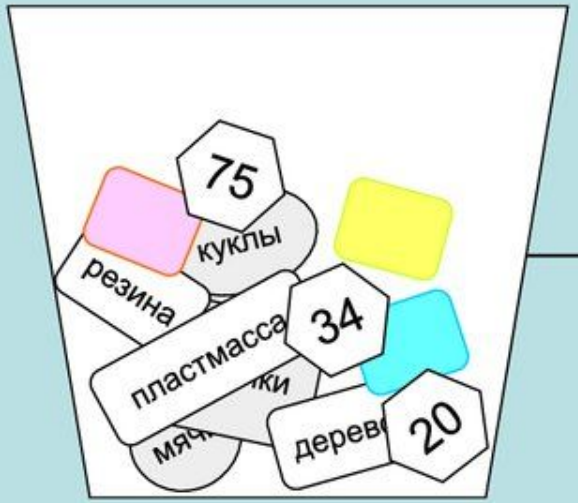
Реляционная база данных

Реляционная база данных – набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей.

Реляционная база данных



Реляционная база данных



Структура данных «Игрушки»

Название	Материал	Цвет	Количество
Мячи	резина	красный	75
Кубики	дерево	голубой	20
Куклы	пластмасса	желтый	34

Объекты

Характеристики (атрибуты) объектов

Поле

Запись

	Имя поля			

Запись	Информация об одном объекте
Поле	Характеристика объекта (атрибут)
Имя поля	Название поля, вынесенное в заголовок

SQL

SQL (Structured Query Language) – язык структурированных запросов, язык запросов для получения из базы данных необходимой информации.

Операторы – определенные слова или символы, которые используются для выполнения конкретной операции. Операторы в SQL делятся на несколько групп в соответствии с задачами, которые они решают:

- DDL (Data Definition Language)
- DML (Data Manipulation Language)
- DCL (Data Control Language)
- TCL (Transaction Control Language)

DDL

DDL (Data Definition Language) – операторы определения данных. Они работают с объектами, то есть с целыми таблицами. Если базу нужно дополнить таблицей с новыми данными или, наоборот, убрать одну из таблиц с ошибочными данными – используется этот набор операторов.

- CREATE — создание объекта в базе данных
- ALTER — изменение объекта
- DROP — удаление объекта

DML

DML (Data Manipulation Language) – операторы манипуляции данными. Эти операторы уже работают с содержимым таблиц – строками, атрибутами и значениями. С их помощью можно вносить изменения в конкретное значение.

- SELECT — выбор данных в соответствии с условием
- INSERT — добавление новых данных
- UPDATE — изменение существующих данных
- DELETE — удаление данных

DCL

DCL (Data Control Language) – оператор определения доступа к данным. Он определяет, кто из пользователей может отправлять запросы к базе, менять объекты и значения. Например, можно отозвать доступ у сотрудника, перешедшего в другой отдел, а также открыть доступ к базе новому маркетологу или разработчику.

- GRANT — предоставление доступа к объекту
- REVOKE — отзыв ранее выданного разрешения
- DENY — запрет, который является приоритетным над разрешением

TCL

TCL (Transaction Control Language) – язык управления транзакциями. Транзакции – это набор команд, которые выполняются поочередно. Если все команды выполнены, транзакция считается успешной, а если где-то произошла ошибка – транзакция откатывается назад, отменяя все выполненные команды. Пример такой транзакции – оплата онлайн, когда банк просит сначала ввести сумму и получателя, затем проверить и подтвердить операцию, а после ввести одноразовый код. На каждом из этих этапов оплату можно отменить и транзакция откатится назад.

Типы данных SQL

При создании таблицы SQL должен решить, какой тип данных будет храниться в каждом столбце. Тип данных является ориентиром для SQL, чтобы понять, какой тип данных ожидается внутри каждого столбца, а также определяет, как SQL будет взаимодействовать с хранимыми данными.

Строковые типы данных

Типы данных	Описание
CHAR(size)	Строки фиксированной длиной (могут содержать буквы, цифры и специальные символы). Фиксированный размер указан в скобках. Можно записать до 255 символов
VARCHAR(size)	Может хранить не более 255 символов.
TINYTEXT	Может хранить не более 255 символов.
TEXT	Может хранить не более 65 535 символов.
BLOB	Может хранить не более 65 535 символов.
MEDIUMTEXT	Может хранить не более 16 777 215 символов.
MEDIUMBLOB	Может хранить не более 16 777 215 символов.
LONGTEXT	Может хранить не более 4 294 967 295 символов.
LONGBLOB	Может хранить не более 4 294 967 295 символов.
ENUM(x,y,z,etc.)	Позволяет вводить список допустимых значений. Можно ввести до 65535 значений в **SQL Тип данных ENUM** список. Если при вставке значения не будет присутствовать в списке ENUM, то мы получим пустое значение. Ввести возможные значения можно в таком формате: ENUM ('X', 'Y', 'Z')
SET(x,y,z,etc.)	**SQL Тип данных SET** напоминает ENUM за исключением того, что SET может содержать до 64 значений.

С плавающей точкой (дробные числа) и целые числа

Типы данных	Описание
TINYINT(size)	Целое число. Может хранить числа от -128 до 127
SMALLINT(size)	Целое число. Диапазон от -32 768 до 32 767
MEDIUMINT(size)	Целое число. Диапазон от -8 388 608 до 8 388 607
INT(size)	Целое число. Диапазон от -2 147 483 648 до 2 147 483 647
BIGINT(size)	Целое число. Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
FLOAT(size,d)	Число с плавающей точкой небольшой точности.
DOUBLE(size,d)	Число с плавающей запятой нормального размера. Общее количество цифр указывается в *size*. Количество знаков после запятой указывается в параметре d
DECIMAL(size,d)	Дробное число, хранящееся в виде строки.

Типы данных SQL — Дата и время

Типы данных	Описание
DATE()	Дата в формате ГГГГ-ММ-ДД
DATETIME()	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP()	Дата и время в формате timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIME()	Время в формате ЧЧ:ММ:СС
YEAR()	Год в двух значной или в четырехзначном формате.

Введение в SQL

- SELECT – извлекает данные из базы данных
- UPDATE – обновляет данные в базе
- DELETE – удаляет данные из базы данных
- INSERT INTO – вставляет новые данные в базу данных
- CREATE DATABASE – создает новую базу данных
- ALTER DATABASE – изменяет базу данных
- CREATE TABLE – создает новую таблицу
- ALTER TABLE – изменяет таблицу
- DROP TABLE – удаляет таблицу
- CREATE INDEX – создает индекс (ключ поиска)
- DROP INDEX – – удаляет индекс

CREATE DATABASE – Создать БД

CREATE DATABASE *databasename*;

Пример:

CREATE DATABASE *TuranColledge*

create database *Colledge_Turan*

!!! Операторы SQL НЕ чувствительны к регистру: create то же самое, что и CREATE

CREATE TABLE – Создать таблицу

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Пример:

```
CREATE TABLE Students (
    StudentID int,
```

Constraints – Ограничения

Ограничения можно указать при создании таблицы с помощью оператора CREATE TABLE или после создания таблицы с помощью оператора ALTER TABLE.

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Constraints – Ограничения

- NOT NULL - гарантирует, чтобы столбец имел значение
- UNIQUE - гарантирует, что все значения в столбце различны
- PRIMARY KEY - первичный ключ, А комбинация NOT NULL и UNIQUE. Уникально идентифицирует каждую строку в таблице
- FOREIGN KEY - внешний ключ, используется для ограничения по ссылкам
- CHECK - гарантирует, что значения в столбце удовлетворяют определенному условию
- DEFAULT - устанавливает значение по умолчанию для столбца, если значение не указано.
- CREATE INDEX - используется для быстрого создания и извлечения данных из базы данных.

NOT NULL

При создании таблицы:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

Изменить уже существующую таблицу:

```
ALTER TABLE Persons
ALTER COLUMN Age int NOT NULL;
```

UNIQUE

При создании таблицы:

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Изменить уже существующую таблицу:

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

PRIMARY KEY

При создании таблицы:

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Добавить первичный ключ к уже существующую таблицу:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

Удалить ключ:

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

FOREIGN KEY

При создании таблицы:

```
CREATE TABLE Students (
    StudenID int NOT NULL PRIMARY KEY,
    Name varchar(255) NOT NULL,
    GroupID int FOREIGN KEY REFERENCES Groups (GroupID)
);
```

Добавить вторичный ключ к уже существующую таблицу:

```
ALTER TABLE Students
ADD FOREGIN KEY (GroupID ) REFERENCES Groups(GroupID );
```

Удалить ключ:

```
ALTER TABLE Persons
DROP FOREIGN KEY GroupID;
```

INSERT INTO

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

INSERT INTO *Customers* (*CustomerName*, *City*, *Country*)
VALUES (Arlan', Astana', 'Kazakhstan'),
('Nurlan', 'LA', 'USA'),
('Temirlan', 'Dubai', 'UAE'),
;

Практика 1


students
student_id
imia
familia
data_rozhdenia
group_id
nomer_tel
address
groups
group_id
nazvanie_grup
specialnost
kurs
kurator_id
predmet_1
predmet_2
predmet_3
teachers
teacher_id
imia
familia
stazh_raboty
nomer_tel
address

Создать таблицу

```
CREATE TABLE teachers (
    teacher_id  int NOT NULL PRIMARY KEY,
    imia varchar(20) NOT NULL,
    familia varchar(20),
    stazh_raboty int,
    nomer_tel varchar(12),
    address varchar(30)          );
```

```
CREATE TABLE groups (
    group_id  int NOT NULL PRIMARY KEY,
    nazvanie_grup varchar(20) NOT NULL,
    speciality varchar(20),
    kurs int,
    kurator_id int FOREGIN KEY REFERENCES teachers (teacher_id ),
     predmet_1 varchar(20),
     predmet_2 varchar(20),
    predmet_3 varchar(20)            );
```

```
CREATE TABLE students (
    student_id  int NOT NULL PRIMARY KEY,
    ima varchar(20) NOT NULL,
    familia varchar(20),
    data_rozhdenia date,
    stazh_raboty int,
    group_id int FOREGIN KEY REFERENCES groups (group_id),
    nomer_tel varchar(12),
    address varchar(30)       );
```

Таблицы

groups							
group_id	nazvanie_grup	specialty	kurs	kurator_id	predmet_1	predmet_2	predmet_3

students						
student_id	imia	familia	data_rozhdenia	group_id	nomer_tel	address

teachers					
teacher_id	imia	familia	stazh_raboty	nomer_tel	address

SELECT

```
SELECT * FROM table_name;
```

```
SELECT column1, column2, ...
FROM table_name;
```

WHERE

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Пример:

```
SELECT * FROM Customers
WHERE Country='USA';
```

AND, OR, NOT

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

ORDER BY

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Пример:

```
SELECT * FROM Customers
ORDER BY Country;
```

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Пример:

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

MIN, MAX

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

Примеры:

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

```
SELECT MAX(column_name)
FROM table_name
```

JOIN

JOIN используется для объединения строк из двух или более таблиц на основе связанного столбца между ними.

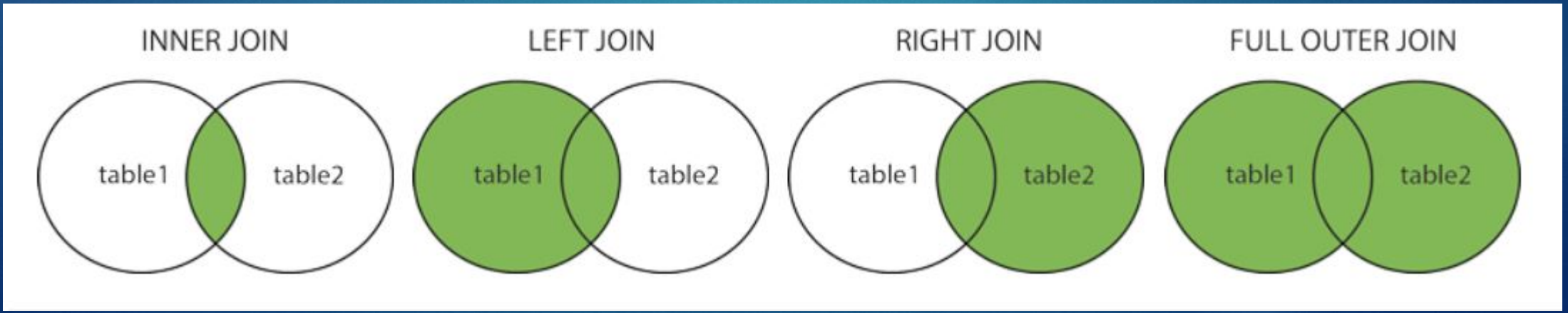
Различные типы JOIN в SQL:

(INNER) JOIN: возвращает записи с совпадающими значениями в обеих таблицах.

LEFT (OUTER) JOIN: возвращает все записи из левой таблицы и соответствующие записи из правой таблицы.

RIGHT (OUTER) JOIN: возвращает все записи из правой таблицы и соответствующие записи из левой таблицы.

FULL (OUTER) JOIN: возвращает все записи, если есть совпадение в левой или правой таблице.



INNER JOIN

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

LEFT JOIN

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

RIGHT JOIN

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

FULL JOIN

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SELF JOIN

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

Спасибо за внимание!