Raiffeisen BANK

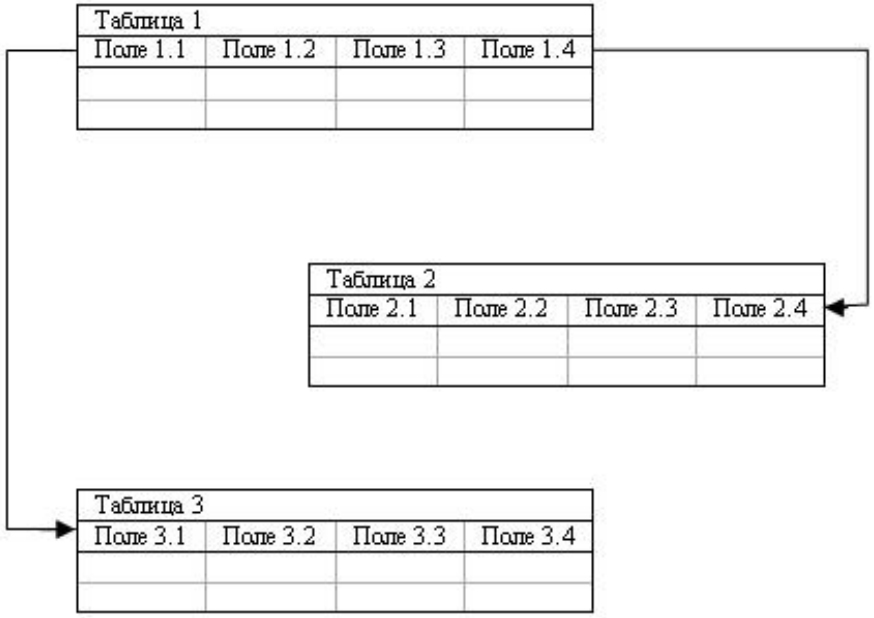
Основы SQL

Понятие БД

База данных - комплекс данных (информации), которые структурированы и взаимосвязаны между собой.

Реляционная модель данных

Реляционные базы данных (БД) имеют табличную форму организации т.е. реализованы в виде таблиц, связанных между собой отношениями с помощью кодов (ключей).



Особенности реляционных БД:

- таблице присваивается имя, уникальное в пределах БД
- каждому столбцу присваивается имя уникальное в пределах таблицы
- каждый столбец таблицы содержит данные одного типа
- в таблице нет одинаковых строк
- строки таблицы не имеют имен (но имеют уникальный идентификатор)

Таблица БД

Таблица представляет из себя набор столбцов. Столбцы таблицы могут называть ***полями или колонками***, все эти слова будут использоваться как синонимы

Строки, записи – тоже синонимы

Поле или столбец

Запись или строка

Код	Фамилия	Имя	Отчество	Дата рождения
1	Иванов	Иван	Петрович	20.07.2000
2	Петров	Илья	Иванович	12.11.1988
3	Сидорова	Анна	Сергеевна	20.06.1989
4	Смирнов	Лев	Владимирович	01.01.2000

Первичный ключ (PRIMARY KEY)

PRIMARY KEY(первичный ключ) – это поле (или комбинация полей), которое однозначно идентифицирует запись.

В таблице **не может** быть двух записей с одинаковым значением ключа.

PRIMARY KEY должен:
- содержать уникальные значения
- не может содержать NULL значения.

Таблица может иметь только один первичный ключ

Могут ли эти данные быть ключом?

- фамилия
- номер и серия паспорта
- номер дома
- регистрационный номер автомобиля

Внешний ключ

Внешний ключ (FOREIGN KEY) атрибут (или группа атрибутов), значение которого может повторяться для нескольких записей.
Содержит ссылку на поле первичного ключа в другой таблице.

Таблица содержащая внешний ключ называется дочерней, содержащая первичный ключ – родительской

«Person» таблица

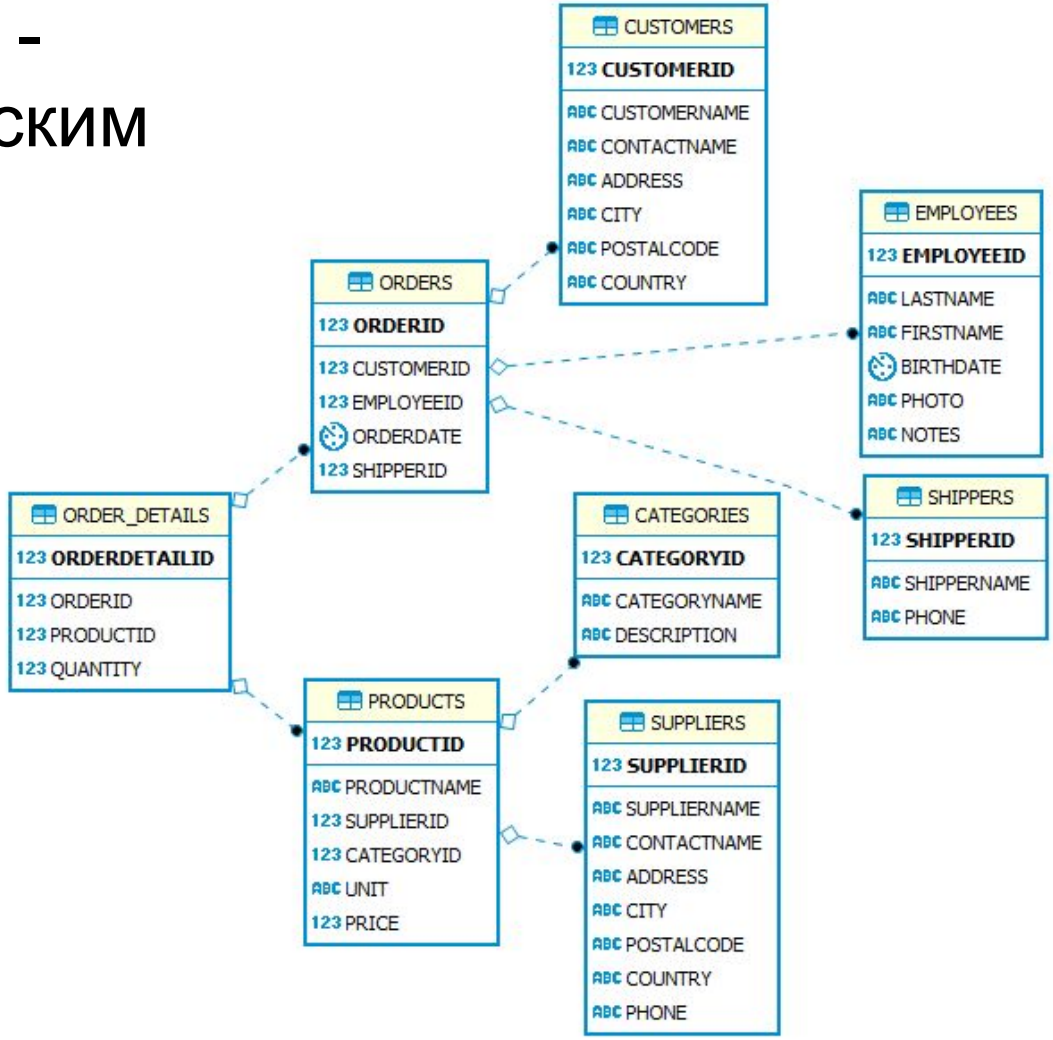
PersonID	Last Name	First Name	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

«Orders» таблица

OrderID	Order Number	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Схема БД

Схема БД (схема) - является графическим образом БД



Типы связей между таблицами

Существует три типа связей между таблицами:

- Связь “**один-ко-многим**”
- Связь “**один-к-одному**”
- Связь “**многие–ко–многим**”

Связи между таблицами

Один к одному («1-1») – одной записи в первой таблице соответствует ровно одна запись во второй.

Применение: выделение часто используемых данных.

1 — 1

Код	Фамилия	Имя
1	Иванов	Кузьма
2	Петров	Василий
…		

Код	Год рожд.	Адрес
1	1992	Суворовский, д.20, кв. 6
2	1993	Кирочная, д. 30, кв 18
…		

Один ко многим («1-∞») – одной записи в первой таблице соответствует сколько угодно записей во второй.

товары: 1 — ∞ :прайс-лист

Код	Название
1	Монитор
2	Винчестер
…	

Код	Код товара	Цена
123	1	10 999
345	1	11 999
…		

Связи между таблицами

Многие ко многим («∞ - ∞») – одной записи в первой таблице соответствует сколько угодно записей во второй, **и наоборот**.

учителя

Код	Фамилия
1	Иванов
2	Петров
…	

∞ ∞

предметы

Код	Название
1	История
2	География
3	Биология
…	

Реализация – через третью таблицу и две связи «1-∞».

1 ∞ ∞ 1

Код	Фамилия
1	Иванов
2	Петров
…	

расписание

Код	Код учителя	Код предмета	Класс
1	1	1	9-А
2	1	2	8-Б
3	2	3	7-В
…			

Код	Название
1	История
2	География
3	Биология
…	

Немного истории

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная
реляционная СУБДэкспериментальная
реляционная СУБД IBM System R,
для которой был создан специальный язык SEQUEL, позволявший управлять данными в этой СУБД.

Аббревиатура **SEQUEL** расшифровывалась как *Structured English Query Language* — «структурированный английский язык запросов».
Позже язык *SEQUEL* был переименован в SQL (Structured Query Language)

Создатели SQL

Дональд Д.Чемберлин и Раймонд Ф.Бойс

С помощью чего можно выполнить SQL запрос

Raiffeisen BANK

Все современные СУБД содержат в своем составе утилиты, позволяющие выполнить SQL запрос и посмотреть его результат.

Например:

- Oracle содержит утилиту SQL Plus
- Microsoft SQL Server - утилиту SQL Query Analyzer

Структура языка SQL

Операторы определения данных
(DDL Data Definition Language)

- CREATE TABLE
- DROP TABLE
- ALTER TABLE
- CREATE VIEW
- ALTER VIEW
- DROP VIEW
- CREATE INDEX
- DROP INDEX

Операторы манипулирования данными
(DML Data Manipulation Lanquaqe)

- DELETE
- INSERT
- UPDATE
- SELECT

Средства администрирования данных
(DCL Data Control Language)

- GRANT
- REVOKE
- DENY

Средства управления транзакциями
(TCL Transaction Control Language)

- COMMIT
- ROLLBACK
- SAVEPOINT

Рейтинг популярности СУБД

Raiffeisen BANK

	СУБД	Год	Разработчик	Диалект SQL
	MySQL	1995	Oracle	**T-SQL**
	PostgreSQL	1995	сообщество	**PLpg/SQL**
	MS SQL Server	1988	Microsoft	**T-SQL**
	SQLite	2000	Hwaci, сообщество	**SQL**
	Oracle Database	1979	Oracle	**PL/SQL**
	Firebird	2000	Сообщество	**PSQL**
	DB2	1995	IBM	**SQL**
	MariaDB	2009	MariaDB Corporation Ab, MariaDB Foundation, сообщество	**T-SQL**

SELECT

Raiffeisen BANK

Используется для выбора данных из базы данных (БД). Данные, возвращаемые в результате запроса, называются множеством результатов.

SELECT column_list
FROM table_name
[*WHERE* условие]
[*GROUP BY* условие]
[*HAVING* условие]
[*ORDER BY* условие]

SELECT - ключевое слово, которое сообщает базе данных о том, что оператор является запросом. Все запросы начинаются с этого слова, за ним следует пробел.

Column_list - список столбцов таблицы, которые выбираются запросом. Столбцы, не указанные в операторе, не будут включены в результат.

SELECT column1, column2, ...FROM table_name

Если необходимо вывести данные всех столбцов, можно использовать (*).

*SELECT * FROM table_name*

FROM - ключевое слово, которое должно присутствовать в каждом запросе. После него через пробел указывается имя таблицы (**table_name**), являющейся источником данных.

Операторы в скобках являются не обязательными в SELECT.

SQL является регистронезависимым.

Задачи:

1. Вывести все поля и записи таблицы Клиенты (Customers)

2. Необходимо вывести список клиентов *(таблица Customers)*, отобразить их имена (*CustomerName*), *город (City), страну (Country).*

CUSTOMERS (Клиенты)
CustomerID
CustomerName
ContactName
Address
City
PostalCode
Country

Задачи:

1. Вывести все поля и записи таблицы Customers

   *SELECT * FROM Customers*

2. Необходимо вывести список клиентов *(таблица Customers)*, отобразить их имена (*CustomerName*), *город (City), страну (Country).*

   SELECT CustomerName, City, Country
   FROM Customers

CUSTOMERS (Клиенты)
CustomerID
CustomerName
ContactName
Address
City
PostalCode
Country

DISTINCT

Raiffeisen BANK

Столбец внутри таблицы часто содержит много повторяющихся значений. Иногда необходимо получить перечень значений без повторов.

SELECT DISTINCT возвращает только различные значения.

DISTINCT Syntax:

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

Задачи:

DISTINCT Syntax:

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

1. Вывести только различающиеся значения из столбца «Страна» (Country) таблицы Клиенты (Customers)

2. Вывести только различающиеся значения из столбца «Город» (City) таблицы Клиенты (Customers

Задачи:

1. Вывести только различающиеся значения из столбца «Страна» (Country) таблицы Клиенты (Customers)

   SELECT DISTINCT Country FROM Customers

2. Вывести только различающиеся значения из столбца «Город» (City)

   SELECT DISTINCT City FROM Customers

WHERE

Raiffeisen BANK

Предложение WHERE используется для извлечения только тех записей, которые удовлетворяют заданному условию (фильтрация записей). **WHERE** также используется в конструкциях UPDATE, DELETE

WHERE Syntax:

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

SQL требует ***одинарные кавычки для текстовых значений*** (в большинстве систем БД также допустимы двойные кавычки). Числовые значения не должны быть заключены в кавычки.

Операторы в **WHERE**

Raiffeisen BANK

Следующие операторы могут быть использованы в предложении **WHERE**:

Оператор	**Описание**
=	Равно
<>	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
BETWEEN	Диапазон значений
LIKE	Поиск по шаблону
IN	Для указания нескольких возможных значений столбца

Примеры:

CustomerID - числовое поле

Запрос:

*SELECT * FROM Customers WHERE CustomerID=1*

Country, PostalCode -Текстовые поля

Запросы:

*SELECT * FROM Customers WHERE PostalCode= '67000'*

*SELECT * FROM Customers WHERE Country='France'*

Задачи:

1. Выбрать из таблицы «Клиенты» (Customers) клиентов из города «Париж» (Paris)

2. Вывести продукты (таблица Products) с ценой (Price) больше 55

Задачи:

1. Выбрать из таблицы «Клиенты» (Customers) клиентов из города «Париж» (Paris)

   SELECT * FROM Customers

   WHERE City='Paris'

2. Вывести продукты (таблица Products) с ценой (Price) больше 55

   SELECT * FROM PRODUCTS WHERE Price>55

AND, OR и NOT Операторы

Предложение WHERE может быть объединено с AND, OR и NOT операторами. OR и AND операторы используются для фильтрации записей основанной на более чем одном условии:

- AND отображает запись, если все условия, разделенные «AND» ИСТИНА
- OR отображает запись, если какое-либо из условий разделенных «OR» ИСТИНА
- NOT отображается запись, если условие не соответствует действительности

AND OR NOT Syntax

AND Syntax

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

OR Syntax

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

NOT Syntax

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

Задачи:

1. Выбрать строки из таблицы «Customers» (Клиенты), где страна (поле Country)=*'Germany'* и Город (поле City)=*'Berlin'*

2. Выбрать строки из таблицы «Customers» (Клиенты), где город (поле City)=*'Berlin'* или *‘London'*

3. Выбрать строки из таблицы «Customers» (Клиенты), где страна (поле Country) не *'France'*

Задачи:

1. Выбрать строки из таблицы «Customers» (Клиенты), где страна (поле Country)*'Germany'* и Город (поле City)=*'Berlin‘*

SELECT * FROM Customers WHERE Country='Germany‘ AND City='Berlin'

2. Выбрать строки из таблицы «Customers» (Клиенты), где город (поле City)=*'Berlin'* или *'London'*

SELECT * FROM Customers WHERE City='Berlin' OR City='London'

3. Выбрать строки из таблицы «Customers» (Клиенты), где страна (поле Country) не *'France‘*

SELECT * FROM Customers WHERE NOT Country='France'

NULL

Raiffeisen BANK

Поле со значением ***NULL*** представляет собой поле без значения. Если поле в таблице не является обязательным, то можно вставить новую запись или обновить без добавления значения в поле, оно будет сохранено со значением NULL.

Значение NULL отличается от нулевого значения (0) или поля, которое содержит пробелы (« »).

Невозможно проверить значения NULL операторами сравнения, такими как =, < или <>.

Нужно использовать IS NULL или IS NOT NULL

IS NULL / IS NOT NULL

IS NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL
```

IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL
```

Задачи:

IS NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL
```

IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL
```

1. Найти записи в таблице Клиенты (Customers), в которых не указан адрес, но указана контактная информация (поле ContactName)

Задачи:

1. Найти записи в таблице Клиенты (Customers), в которых не указан адрес, но указана контактная информация (поле ContactName)

SELECT * FROM Customers WHERE Address IS NULL and ContactName IS NOT NULL

ORDER BY

ORDER BY используется для сортировки множества результатов в порядке возрастания или убывания. ORDER BY сортирует записи в возрастающем порядке по умолчанию. Чтобы отсортировать записи в порядке убывания, используется DESC.

ORDER BY Syntax

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

Задачи:

ORDER BY Syntax

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

1. Отсортировать строки таблицы «Customers» (Клиенты) по полю Country (страна) по возрастанию

2. Отсортировать данные таблицы «Customers» (Клиенты) по 2м полям: Страна по возрастанию, Город по убыванию

Задачи:

1. Отсортировать строки таблицы «Customers» (Клиенты) по полю Country (страна)

   *SELECT * FROM Customers ORDER BY Country*

2. Отсортировать данные таблицы «Customers» (Клиенты) по 2м полям: Страна по возрастанию, Город по убыванию

   *SELECT * FROM Customers ORDER BY Country, City DESC*

SELECT TOP

Raiffeisen BANK

SQL Server / MS Access Syntax:

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

MySQL Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

Oracle Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

Задачи

(*SQL Server / MS Access Syntax*):

1. Написать запрос, который выведет 5 любых записей из таблицы Customers

   *SELECT TOP 5 * FROM Customers*

2. Написать запрос, который выведет 10 % записей из таблицы Customers

   *SELECT TOP 10 PERCENT * FROM Customers*

SQL MIN() and MAX() Functions

Функция MIN () возвращает наименьшее значение в выбранном столбце.

Функция MAX () возвращает наибольшее значение в выбранном столбце.

MIN() Syntax

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

MAX() Syntax

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

Задачи:

1. Вывести максимальное значение из поля Price таблицы Products

   SELECT MAX(Price) FROM Products

2. Вывести минимальное значение из поля Price таблицы Products

   SELECT MIN(Price) FROM Products

3. Вывести одним запросом максимальное и минимальное значение по столбцу Price

   SELECT MIN(Price), MAX(Price) FROM Products

Raiffeisen BANK

COUNT(), AVG() and SUM() Functions

- Функция **COUNT()** возвращает количество строк, которое соответствует указанным критериям
- Функция **AVG()** возвращает среднее значение числового столбца
- Функция **SUM()** возвращает общую сумму по числовому столбцу.

COUNT(), AVG(), SUM() Syntax

COUNT() Syntax

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

AVG() Syntax

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

SUM() Syntax

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

Задачи:

1. Посчитать кол-во записей с помощью Count в таблице *Products*
   SELECT COUNT(*) FROM Products

2. Посчитать среднюю цену товаров в таблице Products с категорией 2 и 5 (столбец CategoryID)
   SELECT AVG(Price) FROM Products where CategoryID=2 or CategoryID=5

LIKE

Raiffeisen BANK

Оператор LIKE используется в предложении WHERE для поиска указанного шаблона в столбце.

LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern
```

Подстановочные знаки, используемые в сочетании с LIKE:

Символ	пояснение
%	Соответствует любой строке любой длины (в том числе нулевой длины)
_	Соответствует одному символу

ак процента и подчеркивание могут быть спользованы в комбинации!

LIKE с подстановочными знаками «_» и «%»

LIKE	Описание
LIKE 'a%'	находит любые значения, которые начинаются с " a"
LIKE '%a'	находит любые значения, которые заканчиваются на "a"
LIKE '%or%'	будут найдены все значения "or" в любом положении
LIKE '_r%'	находит любые значения, которые имеют "r" во второй позиции
LIKE 'a__%'	находит любые значения, которые начинаются с "a" и имеют длину не менее 3 символов
LIKE 'a%o'	находит любые значения, которые начинаются с "a" и заканчивается на "o"

Задачи:

1. Найти всех клиентов с именем клиента (CustomerName), начинающиеся с “M":

   *SELECT * FROM Customers WHERE CustomerName LIKE ‘M%‘*

2. Найти клиентов с именем клиента (CustomerName), заканчивающиеся на "et":

   *SELECT * FROM Customers WHERE CustomerName LIKE '%et‘*

3. Найти клиентов, у которых в имени (CustomerName) в любой позиции содержится "gou":

   *SELECT * FROM Customers WHERE CustomerName LIKE '%gou%‘*

4. Выбрать клиентов с CUSTOMERNAME, которое начинается с «W» и имеет длину 6 символов

   *SELECT * FROM Customers WHERE CustomerName LIKE ‘W_____'*

IN

Оператор IN позволяет указать несколько значений в предложении WHERE.
Оператор IN является сокращением для нескольких условий OR

IN Syntax

```
SELECT column_name(s)
    FROM table_name
    WHERE column_name IN (value1, value2, ...);
```

or:

```
SELECT column_name(s)
    FROM table_name
    WHERE column_name IN (SELECT STATEMENT);
```

Задачи:

IN Syntax

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* IN (*value1*, *value2*, ...);

1. Выбрать всех клиентов (*Customers)*, которые находятся в «Бразилии», «Мексике» и «Канаде» *(Brazil, Mexico, Canada)* :

2. Выбрать всех клиентов (*Customers*), которые НЕ находятся в «Бразилии», «Мексике» и «Канаде» *(Brazil, Mexico, Canada)*

Задачи:

1. Выбрать всех клиентов (таблица *Customers*), которые находятся в странах «*Brazil*», «*Mexico*» и «*Canada*» (столбец Country):

   *SELECT * FROM Customers*
   WHERE Country IN ('Brazil', 'Mexico', 'Canada')

2. Выбрать всех клиентов, которые НЕ находятся в «Бразилии», «Мексики» и «Канады»:

   *SELECT * FROM Customers*
   WHERE Country not IN ('Brazil', 'Mexico', 'Canada')

BETWEEN

Оператор BETWEEN выбирает значения в заданном диапазоне. Значения могут быть числами, текстом или датами.

BETWEEN Syntax

```
SELECT column_name(s)
   FROM table_name
   WHERE column_name BETWEEN value1 AND
   value2;
```

Задачи:

1. Выбрать продукты(таблица *Products*) с ценой(*Price*) от 50 до 150

   *SELECT * FROM Products WHERE Price BETWEEN 50 AND 150 order by price*

Aliases

Raiffeisen BANK

Псевдонимы используются для присвоения таблице или столбцу в таблице временного имени.
Псевдонимы часто используются, чтобы сделать имена столбцов более удобочитаемыми.
Псевдоним существует только во время выполнения запроса.

Alias Column Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Задачи (MS SQL):

1. Вывести все строки таблицы *Customers* и все столбцы (используя *) с присвоением псевдонима таблице «*Customers*».
   *SELECT * FROM Customers **AS C***

2. Вывести из таблицы *Customers* столбцы CustomerName, Country. Заголовок столбца Country переименовать в «Страна»
   *SELECT CustomerName, Country **AS Страна** FROM Customers*

CustomerName	Страна
Alfreds Futterkiste	Germany
Ana Trujillo Emparedados y helados	Mexico
Antonio Moreno Taquerна	Mexico
Around the Horn	UK

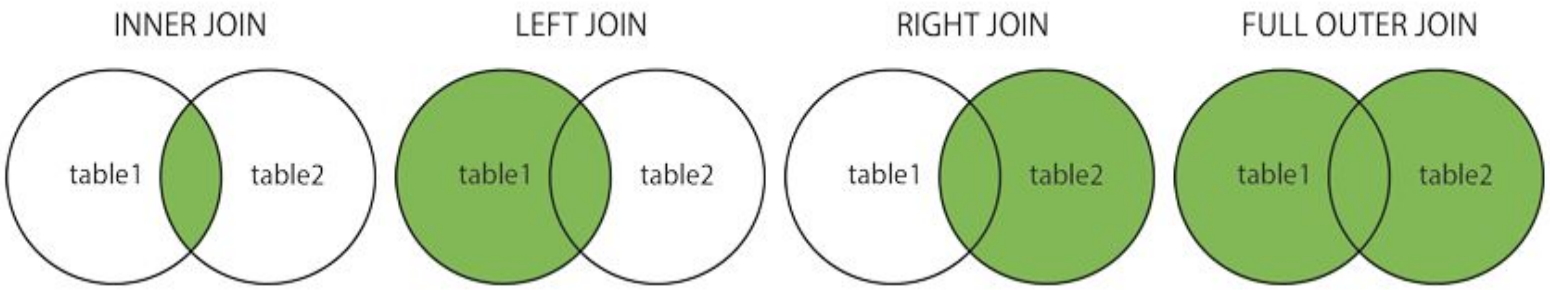
JOIN

JOIN-соединения – операции горизонтального соединения данных. Суть операции объединения - склеить разбитые по таблицам данные ,т.е. привести их обратно в человеческий вид.

JOIN объединяет строки из двух или более таблиц, на основе связи между ними (ключей).

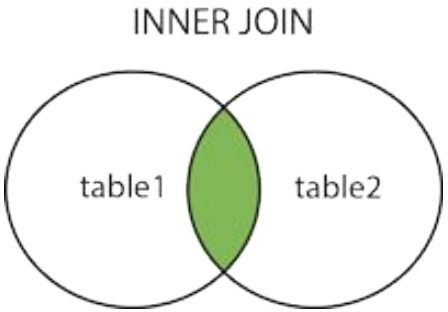
Различные типы SQL JOINS

- **INNER JOIN (ВНУТРЕННИЙ)**: Возвращает записи , которые имеют соответствующие значения в обеих таблицах
- **LEFT (OUTER) JOIN**: Возвращает все записи из левой таблицы и совпавшие записи из таблицы справа
- **RIGHT (OUTER) JOIN**: Возвращает все записи из таблицы справа, и совпавшие записи из левой таблицы
- **FULL (OUTER) JOIN (ПОЛНЫЙ)**: Возвращает все записи , когда есть совпадение в левой или правой таблице



INNER JOIN

INNER JOIN выбирает записи, которые имеют совпадающие значения в обеих таблицах.



INNER JOIN Syntax

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

Примечание:

Внутреннее соединение выбирает все строки из обеих таблиц до тех пор, пока существует соответствие между столбцами.

В большинстве СУБД ключевое слово **INNER** является необязательным, поэтому можно просто не указывать его.

INNER JOIN

INNER JOIN выбирает записи, которые имеют совпадающие значения (по ключу) в обеих таблицах.

Задачи:

1. Выбрать все заказы (таблица Orders) с информацией о клиентах (таблица Customers):

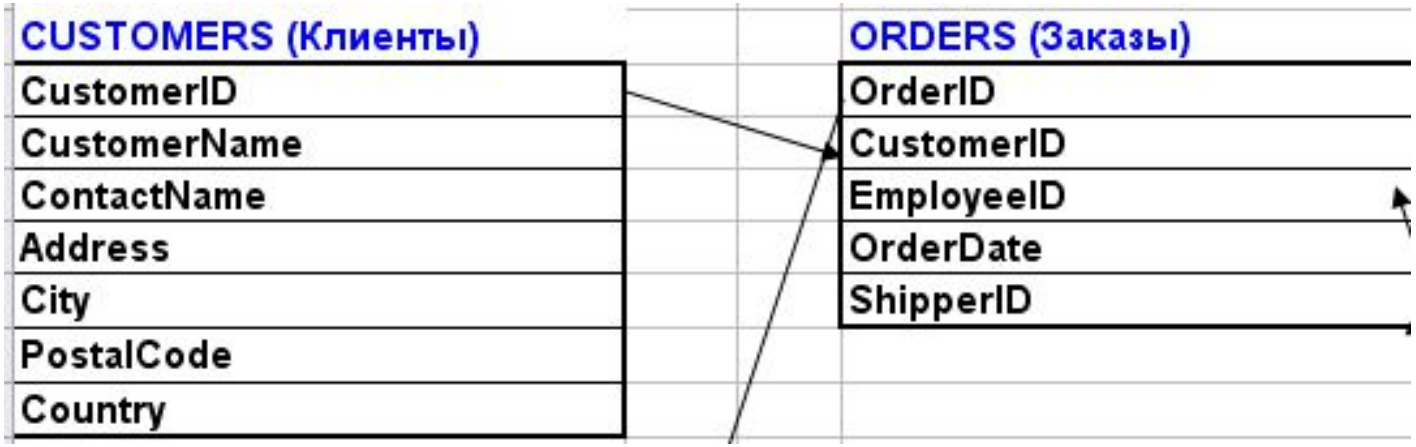
INNER JOIN Syntax

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

Задачи:

1. Показать все заказы (таблица Orders) с информацией о клиентах (таблица Customers):

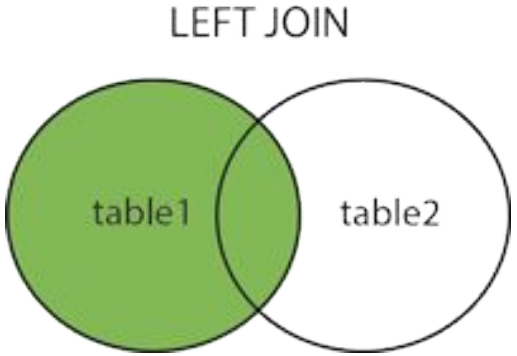
```
SELECT Orders.OrderID, Orders.OrderDate
Customers.CustomerName, Customers.ContactName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID =
Customers.CustomerID
```



LEFT JOIN

LEFT JOIN возвращает все записи из левой таблицы (Table1), а также совпавшие записи из правой таблицы (table2). В результате будет NULL с правой стороны, если нет совпадения.

В некоторых базах данных LEFT JOIN называется LEFT OUTER JOIN



LEFT JOIN Syntax

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

LEFT OUTER JOIN

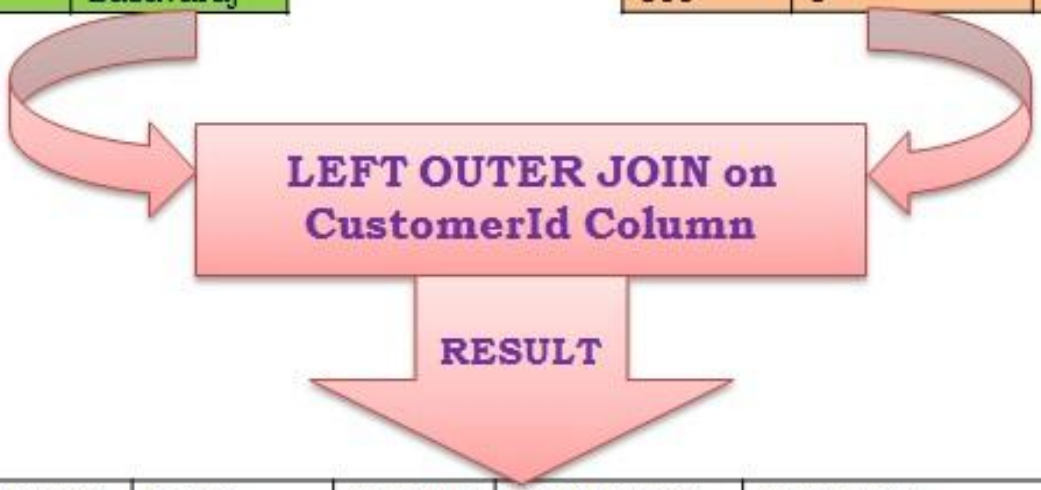
Raiffeisen BANK

Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700



CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	NULL	NULL	NULL
3	Basavaraj	300	3	2014-02-01 23:48:32.853

LEFT JOIN возвращает все записи из левой таблицы даже если нет совпадений в таблице справа

Задачи:

1. Вывести список всех клиентов (таблица Customers) и их заказы (таблица Orders)

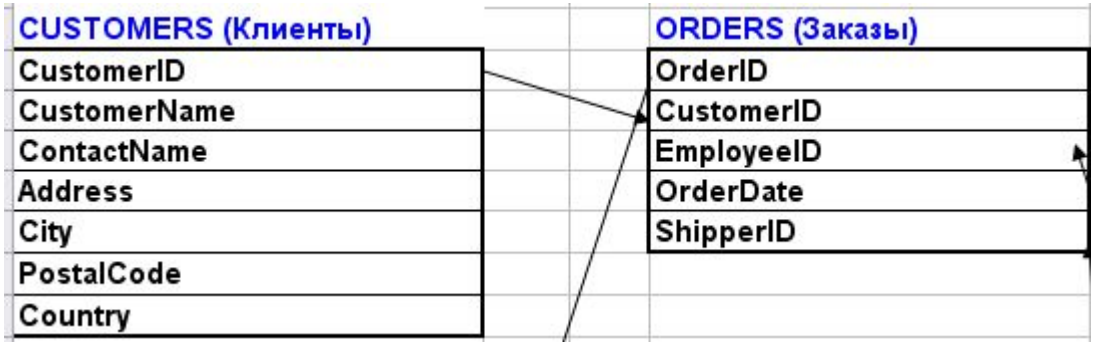
LEFT JOIN Syntax

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

Задачи:

1. Вывести список всех клиентов (таблица Customers) и их заказы (таблица Orders)

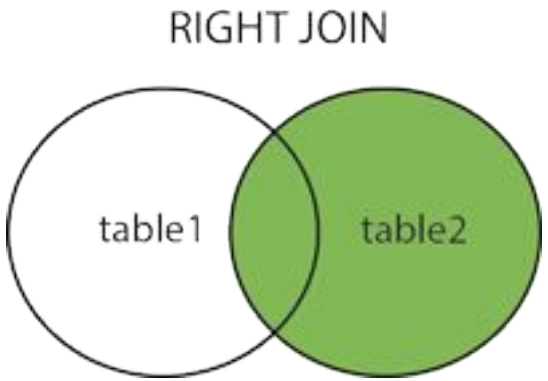
```
SELECT Orders.OrderID, Orders.OrderDate,
Customers.CustomerName, Customers.ContactName
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName
```



RIGHT JOIN

RIGHT JOIN возвращает все записи из правой таблицы (table2) и совпавшие записи из левой таблицы (table1). В результате отображается NULL с левой стороны, когда нет соответствия.

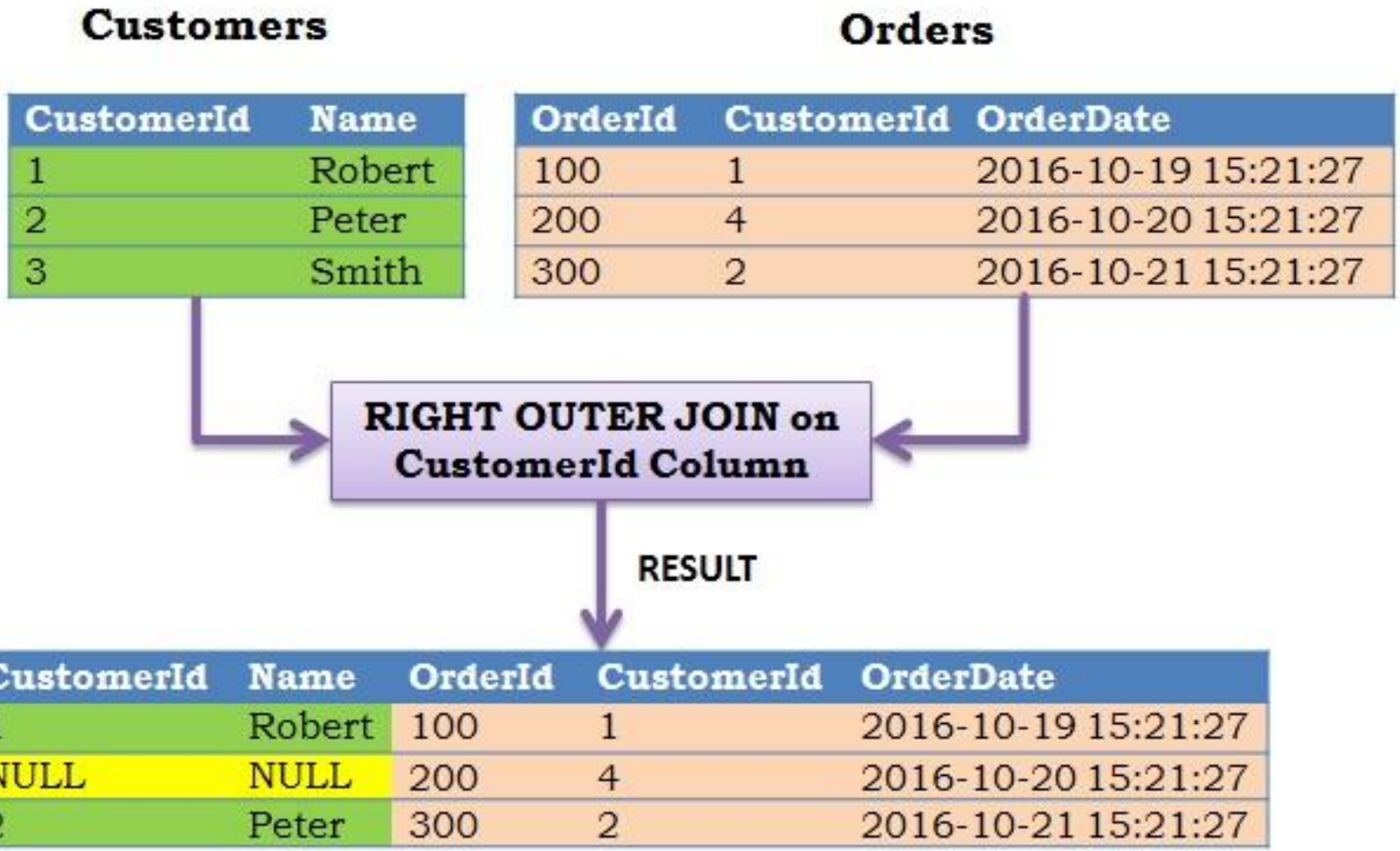
В некоторых базах данных RIGHT JOIN называется RIGHT OUTER JOIN



RIGHT JOIN Syntax

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

RIGHT OUTER JOIN



CustomerId	Name
1	Robert
2	Peter
3	Smith

OrderId	CustomerId	OrderDate
100	1	2016-10-19 15:21:27
200	4	2016-10-20 15:21:27
300	2	2016-10-21 15:21:27

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Robert	100	1	2016-10-19 15:21:27
NULL	NULL	200	4	2016-10-20 15:21:27
2	Peter	300	2	2016-10-21 15:21:27

RIGHT JOIN возвращает все записи из правой таблицы, даже если нет совпадений в таблице слева

Задачи:

1. Вывести всех сотрудников (*Employees*) и заказы (*Orders*), ими оформленные используя RIGHT JOIN

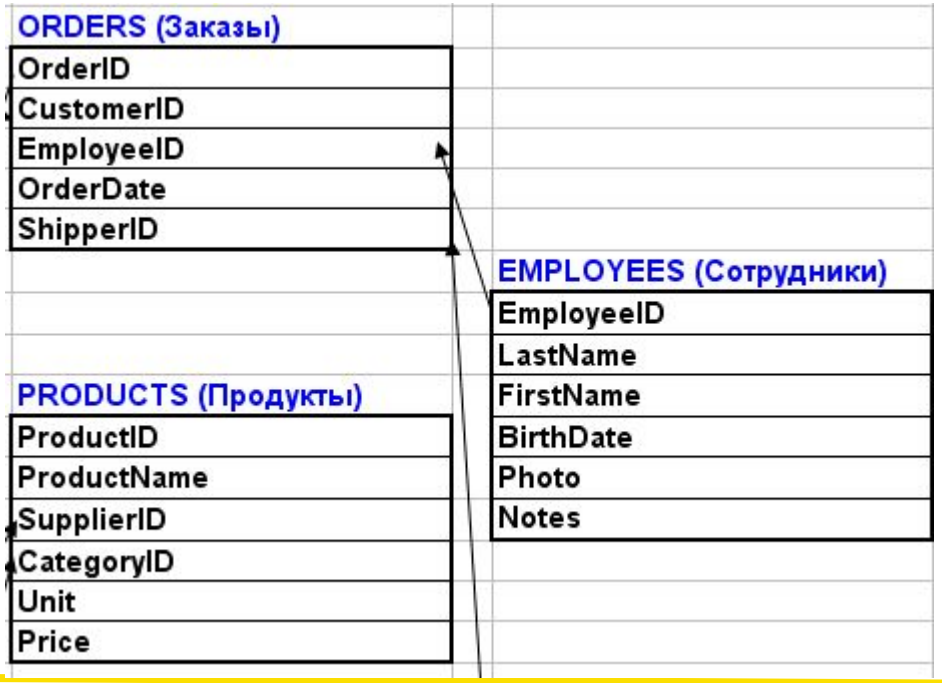
RIGHT JOIN Syntax

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

Задачи:

1. Вывести всех сотрудников (*Employees*) и заказы (*Orders*), ими оформленные используя RIGHT JOIN

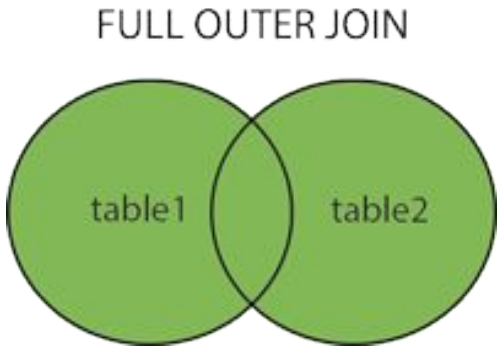
```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID =
Employees.EmployeeID
ORDER BY Orders.OrderID
```



FULL OUTER JOIN

FULL OUTER JOIN возвращает строку из любой таблицы, когда условия выполняются, и возвращает нулевое значение, когда нет совпадения. Порядок таблиц для оператора неважен, поскольку оператор является симметричным.
Объединяет левое внешнее соединение и правое внешнее соединение.

FULL OUTER JOIN потенциально может вернуться очень большие результирующие наборы!



FULL OUTER JOIN Syntax

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

FULL OUTER JOIN

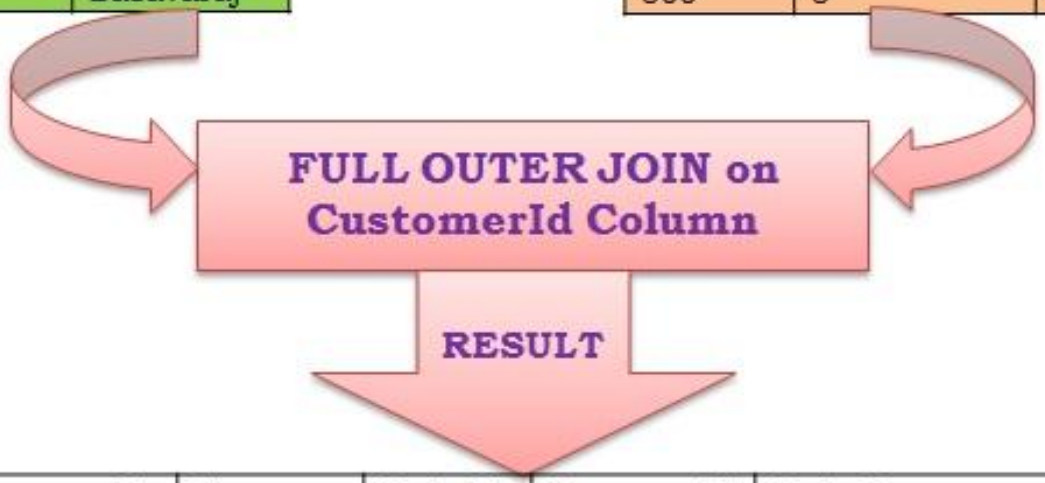
Raiffeisen BANK

Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

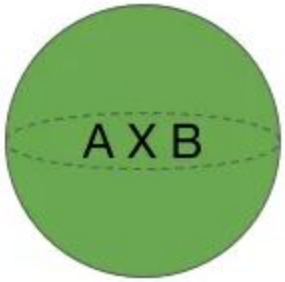


CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	NULL	NULL	NULL
3	Basavaraj	300	3	2014-02-01 23:48:32.853
NULL	NULL	200	4	2014-01-31 23:48:32.853

FULL OUTER JOIN возвращает все строки из левой таблицы *Customers* (Клиенты) и все строки из правой таблицы Orders (Заказы).

CROSS JOIN

CROSS JOIN – это декартово произведение. Результатом такого соединения будет сцепление каждой строки первой таблицы с каждой строкой второй таблицы.
Если в *Table1* содержится N1 записей, а в *Table2* – N2 записей, в результате будет N1 x N2 записей.



CARTESIAN
(CROSS) JOIN

FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
CROSS JOIN  table2
```

```
SELECT column_name(s)
FROM table1,  table2
```

CROSS JOIN

Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	100	1	2014-01-30 23:48:32.850
3	Basavaraj	100	1	2014-01-30 23:48:32.850
1	Shree	200	4	2014-01-31 23:48:32.853
2	Kalpana	200	4	2014-01-31 23:48:32.853
3	Basavaraj	200	4	2014-01-31 23:48:32.853
1	Shree	300	3	2014-02-01 23:48:32.853
2	Kalpana	300	3	2014-02-01 23:48:32.853
3	Basavaraj	300	3	2014-02-01 23:48:32.853

Результатом такого соединения будет сцепление каждой строки первой таблицы с каждой строкой второй таблицы.

GROUP BY

Raiffeisen BANK

Предложение **GROUP BY** используется для определения групп, к которым могут применяться агрегатные функции (**COUNT**, **MIN**, **MAX, AVG** и **SUM**).

Если **Group by** отсутствует, и используются агрегатные функции, то все столбцы с именами, упомянутыми в **SELECT**, должны быть включены в агрегатные функции, и эти функции будут применяться ко всему набору строк. В противном случае все столбцы списка **SELECT**, не вошедшие в агрегатные функции, должны быть указаны в предложении **GROUP BY**. В результате чего все выходные строки запроса разбиваются на группы, характеризуемые одинаковыми комбинациями значений в этих столбцах. После чего к каждой группе будут применены агрегатные функции.

Если при наличии предложения **GROUP BY**, в предложении **SELECT** отсутствуют агрегатные функции, то запрос просто вернет по одной строке из каждой группы..

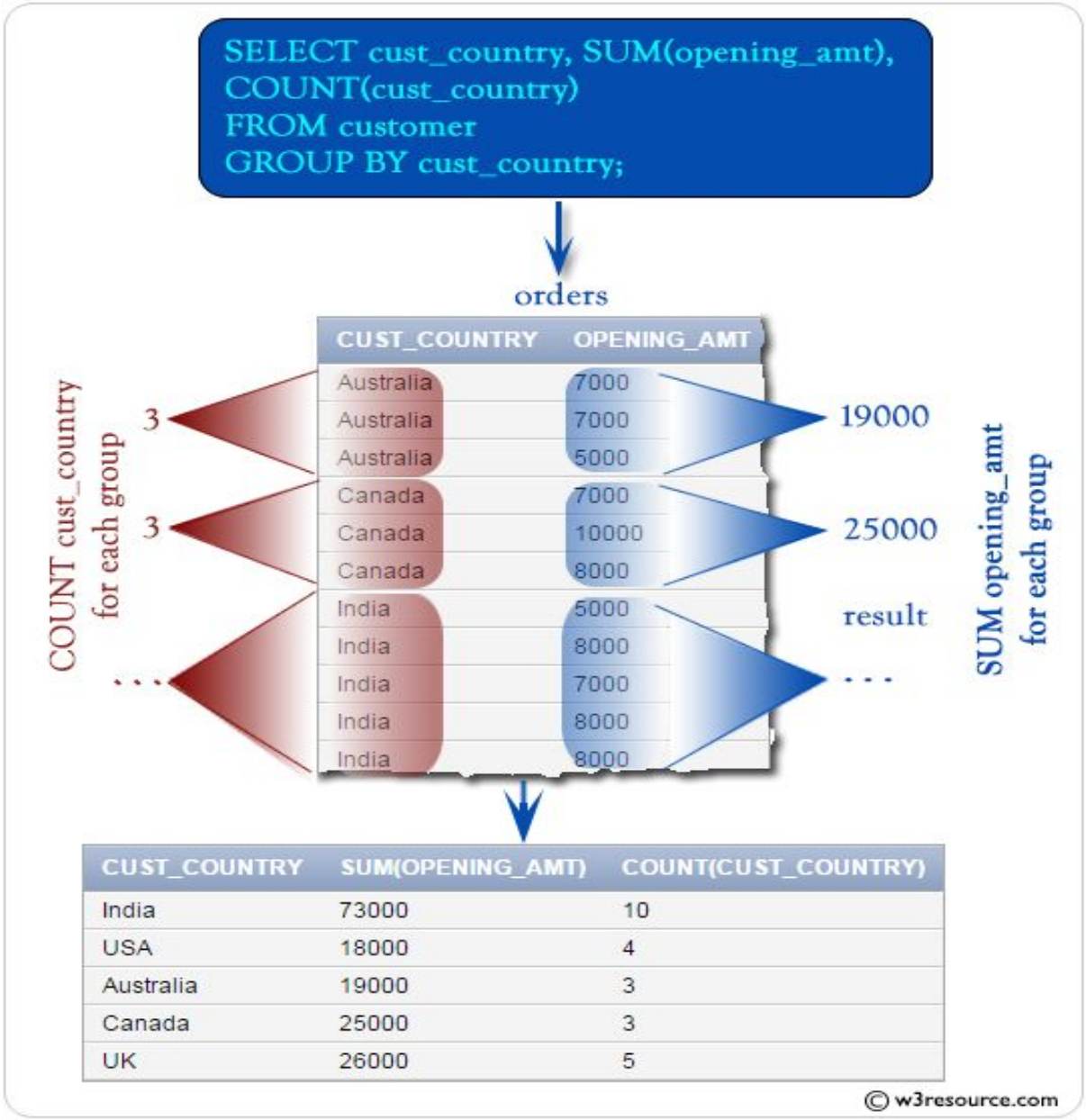
Raiffeisen BANK

GROUP BY

GROUP BY используется для определения групп, к которым могут применяться агрегатные функции (**COUNT**, **MIN**, **MAX, AVG** и **SUM**).

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```


SELECT cust_country, SUM(opening_amt),
COUNT(cust_country)
FROM customer
GROUP BY cust_country;
orders
CUST_COUNTRY
OPENING_AMT
Australia 7000
Australia 7000
Australia 5000
Canada 7000
Canada 10000
Canada 8000
India 5000
India 8000
India 7000
India 8000
India 8000
COUNT cust_country for each group
3
3
. . .
19000
25000
result
. . .
SUM opening_amt for each group
CUST_COUNTRY
SUM(OPENING_AMT)
COUNT(CUST_COUNTRY)
India 73000 10
USA 18000 4
Australia 19000 3
Canada 25000 3
UK 26000 5
© w3resource.com

Задачи:

1. Посчитать количество клиентов (таблица *Customers*) в каждой стране, результаты отсортировать по убыванию столбца с количеством

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC
```

CUSTOMERS (Клиенты)
CustomerID
CustomerName
ContactName
Address
City
PostalCode
Country

HAVING

Raiffeisen BANK

HAVING пункт был добавлен в SQL, потому что ключевое слово WHERE не может быть использовано с агрегатными функциями. WHERE используют относительно всех выбираемых данных, а HAVING - только по отношению к группам, определенным в параметре GROUP BY

HAVING позволяет определять условия выборки уже сгруппированных некоторым образом данных

- используется исключительно в связке с параметром GROUP BY и указывается сразу же после него и перед ORDER BY.
- в условии этого параметра можно использовать только агрегатные функции и поля, указанные в параметре GROUP BY.

HAVING Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Задачи:

1. Вычислить количество зарегистрированных клиентов в каждой стране, показать только страны, где зарегистрировано больше 5 клиентов

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
```

CUSTOMERS (Клиенты)
CustomerID
CustomerName
ContactName
Address
City
PostalCode
Country

UPDATE

UPDATE используется для изменения существующих записей в таблице.

Будьте осторожны при обновлении записей в таблице! Обратите внимание на пункт WHERE в операторе UPDATE. Предложение WHERE определяет, какая запись/записи, которые должны быть обновлены. Если опустить предложение WHERE, все записи в таблице будут обновлены!

UPDATE Syntax

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

Задачи:

1. Обновить поле *City* клиента, у которого ключевое поле CustomerID=85 новым городом= ‘Paris’

```
UPDATE Customers
SET  City='Paris'
WHERE CustomerID=85
```

CUSTOMERS (Клиенты)
CustomerID
CustomerName
ContactName
Address
City
PostalCode
Country

DELETE

Будьте осторожны при удалении записей в таблице!
Предложение WHERE определяет , какие записи должны быть удалены. Если опустить предложение WHERE, все записи в таблице будут удалены!

DELETE Syntax

```
DELETE FROM table_name
WHERE condition;
```

Пример:

Следующая конструкция удаляет клиента «Alfreds Futterkiste» из таблицы «Клиенты»:

```
DELETE FROM Customers
    WHERE CustomerName='Alfreds Futterkiste';
```

INSERT INTO

Добавляет строку в таблицу

Syntax:

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

Пример:

INSERT INTO Customers
(CustomerName, ContactName, Address, City, PostalCode)
VALUES ('Cardinal', 'Tom B', 'Skagen21', ‘Paris', '4006')

INSERT INTO SELECT

Копирует данные из одной таблицы и вставляет их в другую. Для вставки требуется, чтобы типы данных в столбцах исходной и конечной таблиц совпадали.

Syntax:

INSERT INTO *table2* (*column1*, *column2*, *column3*, ...)
SELECT *column1*, *column2*, *column3*, ...

FROM *table1* WHERE *condition*;

Пример:

INSERT INTO Customers (CustomerName, City, Country)

SELECT SupplierName, City, Country FROM Suppliers;

Raiffeisen BANK

Рекомендуемые сайты по SQL

1. http://www.sql-ex.ru/
2. https://www.codecademy.com/
3. https://www.udemy.com/
4. https://www.w3schools.com/
5. www.sql-tutorial.ru

Raiffeisen BANK

Рекомендуемые книги по SQL

Алан Бьюли «Изучаем SQL»
Мартин Грайбер "Понимание SQL"
Крис Фиайли «SQL»