Базы данных и их разновидности. Запросы на выборку и модификацию данных.

www.andersenlab.com

План лекции

1. Базы данных
2. Разновидности баз данных
3. SQL запросы

База данных (БД) - это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов.

Система управления базами данных (СУБД) - это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации.

СУБД основываются на модели базы данных - это специальные структуры предназначенные для работы с данными.

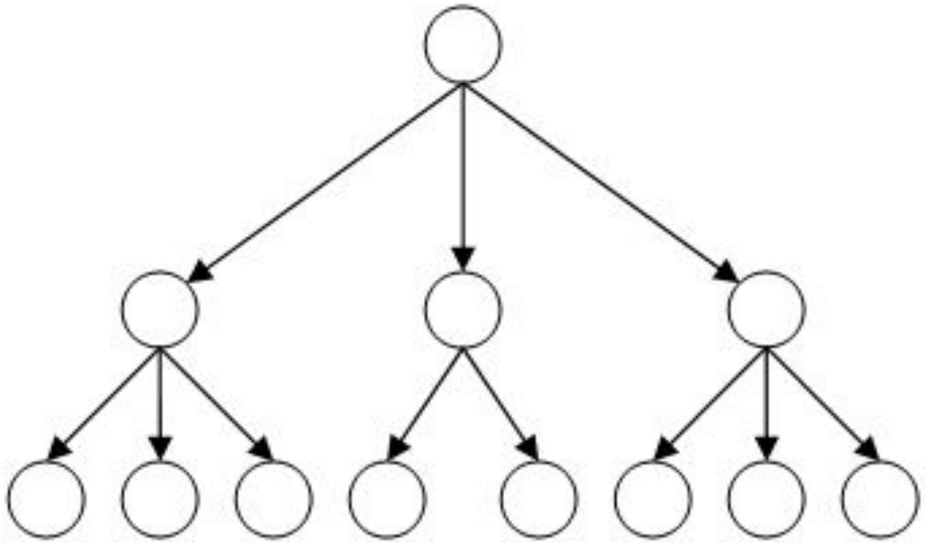
Разновидность баз данных

Виды:

- Иерархическая;
- Объектная и объектно-ориентированная;
- Объектно-реляционная;
- Реляционная;
- Сетевая;
- Функциональная.

Иерархические базы данных

Иерархическая база данных – каждый объект, при таком хранение информации, представляется в виде определенной сущности, то есть у этой сущности могут быть дочерние элементы, родительские элементы, а у тех дочерних могут быть еще дочерние элементы, но есть один объект, с которого все начинается.



Объектные базы данных

Объектные базы данных - это модель работы с объектными данными. Объектная модель идеально подходит для трактовки такого рода объектных данных как изображение, музыка, видео, разного вида текст.

Объектно-ориентированная база данных (ООБД) - база данных, в которой данные моделируются в виде объектов, их атрибутов, методов и классов. Объектно-ориентированные базы данных обычно рекомендованы для тех случаев, когда требуется высокопроизводительная обработка данных, имеющих сложную структуру.

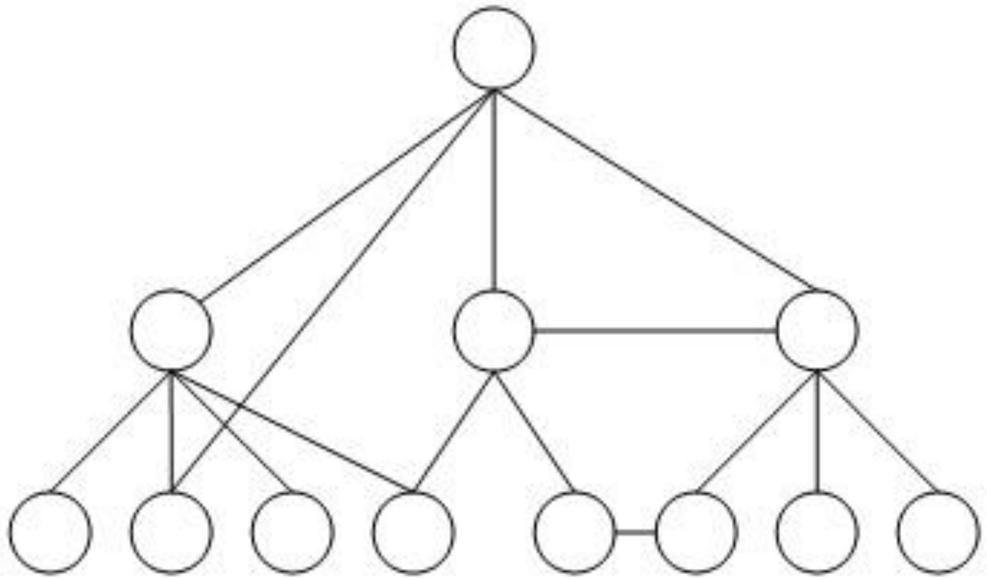
Объектно-реляционные СУБД

Объектно-реляционные СУБД объединяют в себе черты реляционной и объектной моделей. Их возникновение объясняется тем, что реляционные базы данных хорошо работают со встроенными типами данных и гораздо хуже - с пользовательскими, нестандартными. Когда появляется новый важный тип данных, приходится либо включать его поддержку в СУБД, либо заставлять программиста самостоятельно управлять данными в приложении.

Реляционная(или табличная) БД содержит перечень объектов одного типа, т.е. объектов с одинаковым набором свойств. Такую базу удобно представлять в виде двумерной таблицы (или, чаще всего, нескольких связанных между собой таблиц).

Примером такой таблицы может служить БД "Учащиеся", представляющая собой перечень объектов (учеников), каждый из которых имеет фамилию, имя, отчество, дату рождения, класс, номер личного дела и др.

Сетевые базы данных являются своеобразной модификацией иерархических баз данных. Сетевые базы данных отличаются от иерархических тем, что у дочернего элемента может быть несколько предков, то есть элементов стоящих выше него.



Функциональные базы данных используются для решения аналитических задач: финансовое моделирование и управление производительностью.

Функциональная модель является частью категории оперативной аналитической обработки (OLAP электронной таблице,), поскольку она включает многомерное иерархическое объединение. Но она выходит за рамки OLAP, требуя ориентирования ячейки, подобно тому, где ячейки могут быть введены или рассчитаны как функции других ячеек. Также, как и в электронных таблицах, данная модель поддерживает интерактивные вычисления, в которых значения всех зависимых ячеек автоматически обновляются каждый раз, когда изменяется значение ячейки.

SQL

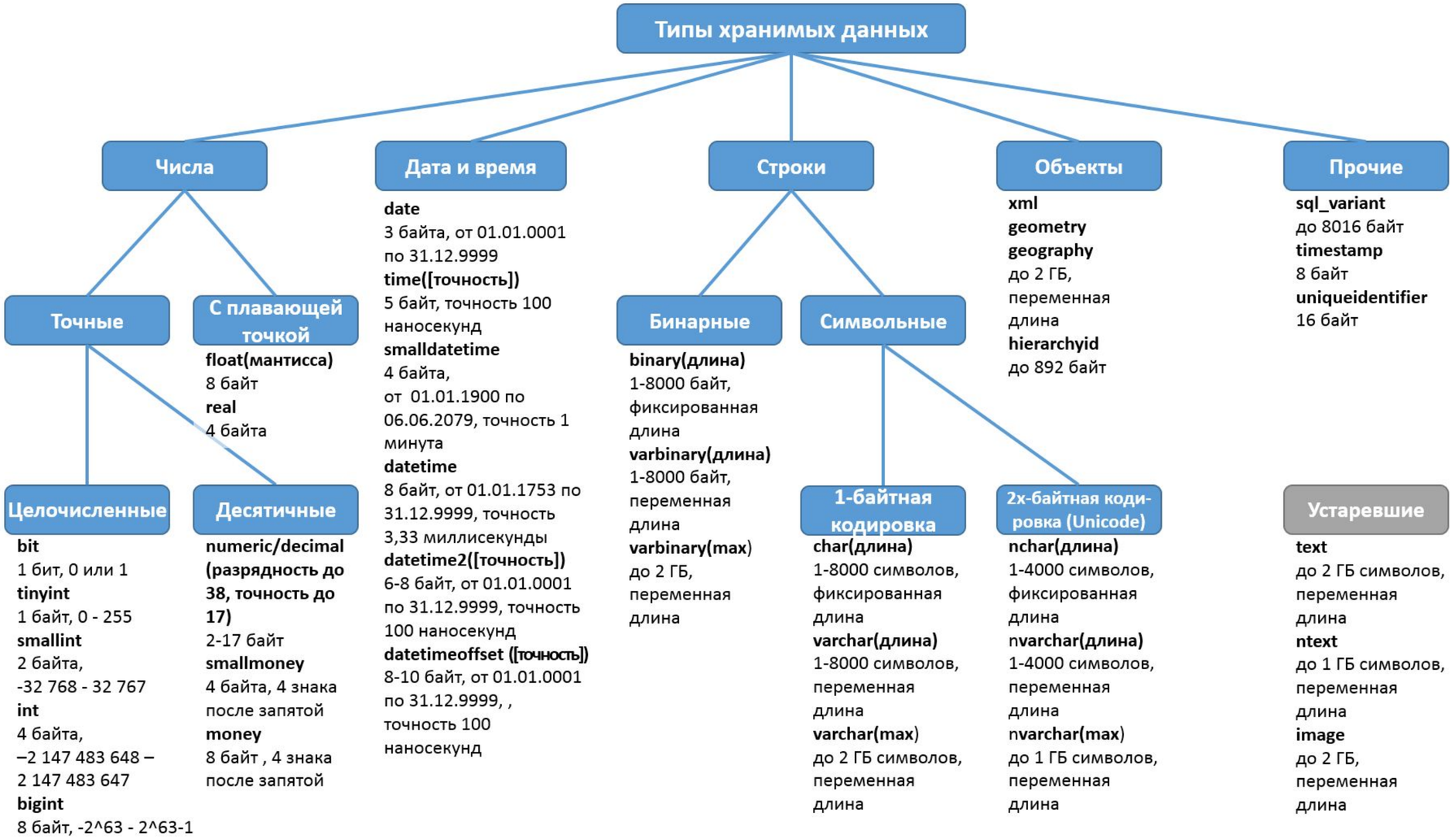
SQL - это универсальный язык структурированных запросов, применяемый для создания, модификации и управления данными в реляционных базах данных.
Structured Query Language - язык структурированных данных. Поддерживается всеми современными СУБД. Язык SQL основан на запросах (query) - инструкциях оформленных определенным образом.

Типы данных в SQL

Типы данных SQL разделяются на три группы:

- Строковые;
- С плавающей точкой (дробные числа);
- Целые числа, дата и время.

Типы данных в SQL



CREATE - создает объект БД (базу, таблицу, представление, пользователя и т. д.).

Пример:

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

ALTER - изменяет объект.

Пример:

ALTER TABLE table_name
ADD column_name datatype;
ALTER TABLE classics RENAME new1name; - переименовать таблицу
ALTER TABLE classics ADD pages SMALLINT UNSIGNED; - добавляет новый столбец по имени pages
ALTER TABLE classics DROP pages; - удаление столбца.
ALTER TABLE classics CHANGE type category VARCHAR(16); - изменит имя столбца с type на category.

Операторы SQL. DROP

DROP - удаляет объект.

Пример:

```
DROP TABLE table_name;
```

DELETE - удаляет данные.

Пример:

```
DELETE FROM table_name
WHERE condition.
```

INSERT INTO - добавление новых данных в таблицу.

Пример:

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

Операторы SQL. UPDATE

UPDATE - изменение существующих данных.

Пример:

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

SELECT - считывает данные, удовлетворяющие заданным условиям.

Пример:
SELECT column1, column2, ...
FROM table_name;
SELECT * FROM table_name;
SELECT COUNT(*) FROM classics; подсчет количества строк.
SELECT DISTINCT author FROM classics; исключает множество записей, имеющих одинаковые данные, т. е если автор встречается 2р, то выведется один раз.

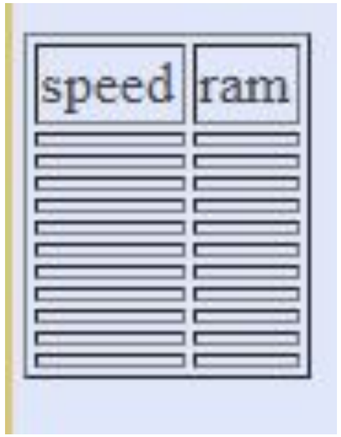
Чтобы упорядочить поля результирующего набора, их следует перечислить через запятую в нужном порядке после слова SELECT:

SELECT price, speed, hd, ram, cd, model, code FROM Pc.

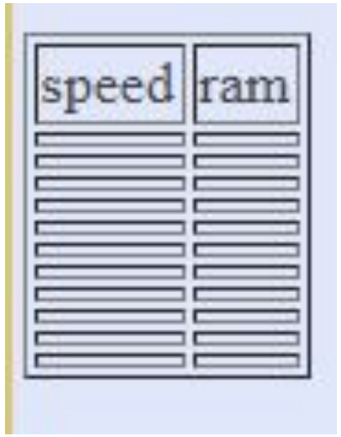
В таблице приводится результат выполнения этого запроса. (Запрос SELECT)

price	speed	hd	ram	cd	model	code
600.0				12x		
850.0				40x		
600.0				12x		
850.0				40x		
850.0				40x		
950.0				50x		
400.0				12x		
350.0				24x		
350.0				24x		
350.0				12x		
980.0				40x		

Вертикальную проекцию таблицы PC можно получить, если перечислить только необходимые поля. Например, чтобы получить информацию только о скорости процессора и объеме оперативной памяти компьютеров, следует выполнить запрос: **SELECT speed, ram FROM PC;** который вернет следующие данные (Запрос SELECT speed.):



Следует отметить, что вертикальная выборка может содержать дубликаты строк в том случае, если она не содержит потенциального ключа, однозначно определяющего запись. В таблице PC потенциальным ключом является поле code, которое выбрано в качестве первичного ключа таблицы. Поскольку это поле отсутствует в запросе, в приведенном выше результирующем наборе имеются дубликаты строк (например, строки 1 и 3). Если требуется получить уникальные строки (скажем, нас интересуют только различные комбинации скорости процессора и объема памяти, а не характеристики всех имеющихся компьютеров), то можно использовать ключевое слово DISTINCT:
SELECT DISTINCT speed, ram FROM Pc; что даст такой результат (1-й результат запроса SELECT DISTINCT speed.):

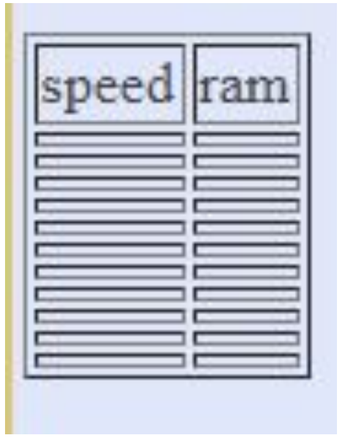


Также может использоваться ключевое слово ALL, которое принимается по умолчанию. Чтобы упорядочить строки результирующего набора, можно выполнить сортировку по любому количеству полей. Для этого используется ORDER BY, являющееся всегда последним предложением в операторе SELECT. В списке полей могут указываться как имена полей, так и их порядковые позиции. Так, упорядочить результирующий набор по объему оперативной памяти в порядке убывания, можно так (2-й результат запроса SELECT DISTINCT speed.):

```
SELECT DISTINCT speed, ram
FROM Pc
ORDER BY ram DESC
```

Или

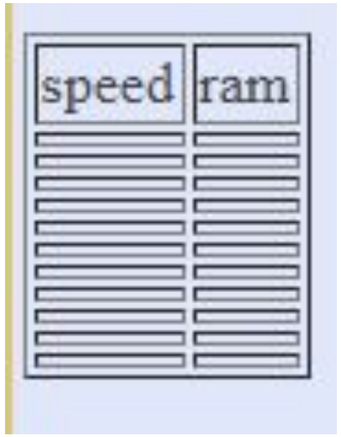
```
SELECT DISTINCT speed, ram
FROM Pc
ORDER BY 2 DESC
```



Сортировку можно проводить по возрастанию (параметр ASC принимается по умолчанию) или по убыванию (параметр DESC). Сортировка по двум полям:

```
SELECT DISTINCT speed, ram
FROM Pc
ORDER BY ram DESC, speed DESC
```

даст следующий результат (3-й результат запроса SELECT DISTINCT speed.):



Операторы SQL. LIKE

LIKE - определяет, совпадает ли данная символьная строка с указанным шаблоном.
Шаблон включает специальные символы:

- % - соответствует любой строке любой длины (в том числе нулевой);
- _ - соответствует одному символу

Пример:

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

Операторы SQL. LIKE

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

ORDER BY - используется для сортировки записей для набора результатов SELECT запроса.

Пример:

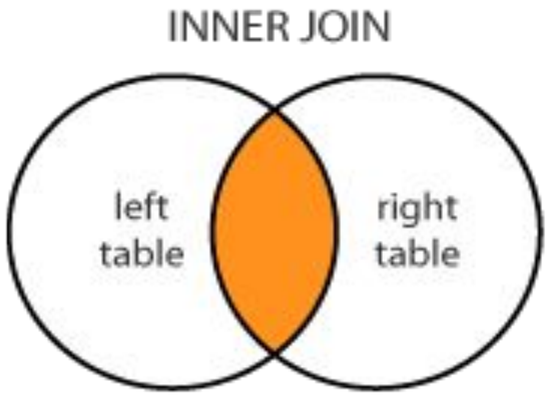
```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

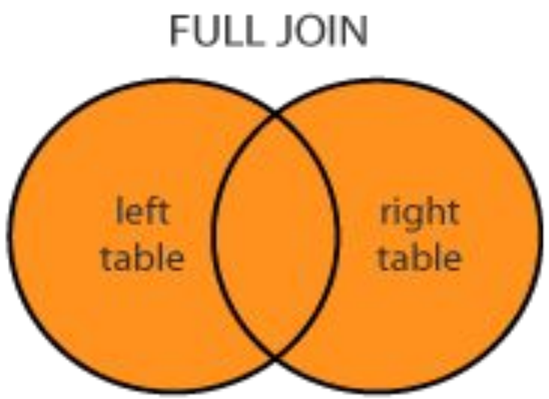
GROUP BY - можно использовать в SELECT для сбора данных по нескольким записям и группировки результатов
одного или нескольких столбцов.

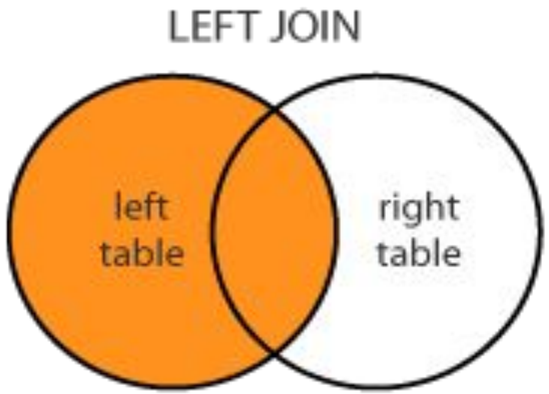
Пример:

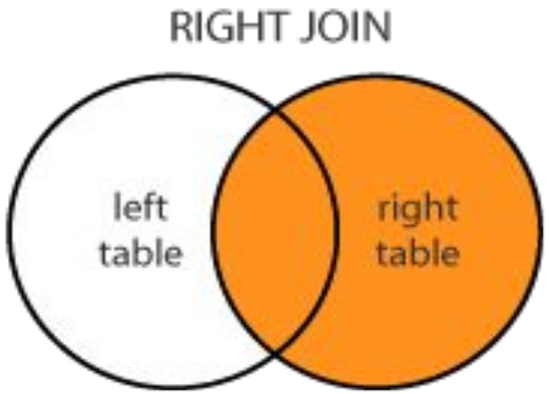
```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

JOINS - используются для извлечения данных из нескольких таблиц.









INNER JOIN - возвращает все строки из нескольких таблиц, где выполняется условия соединения.

Пример:

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

Операторы SQL. LEFT JOIN

LEFT JOIN - этот тип соединения возвращает все строки из таблиц с левосторонним соединением, указанным в
условии ON, и только те строки из другой таблицы, где объединяемые поля равны.

Пример:

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

RIGHT JOIN - этот тип соединения возвращает все строки из таблиц с правосторонним соединением, указанным в
условии ON, и только те строки из другой таблицы, где объединяемые поля равны.

Пример:

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

Операторы SQL. FULL OUTER JOIN

FULL OUTER JOIN - этот тип соединения возвращает все строки из левой таблицы и правой таблицы с NULL - значениями в месте, где условие объединения не выполняется.

Пример:

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

Спасибо за внимание!

www.andersenlab.com

Список литературы

1. https://www.site-do.ru/db/db1.php
2. https://metanit.com/sql/sqlserver/
3.https://intellect.ml/bazy-dannykh-i-znanij-vidy-baz-dannykh-relyatsionnye-i-ne-relyatsionnye-153
4. http://devacademy.ru/posts/sql-nosql/
5. https://www.w3schools.com/sql/sql_join_full.asp