Выборка данных

Оператор SELECT

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    [ * | expression [ [ AS ] output_name ] [, ...] ]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
    [ HAVING condition ]
    [ WINDOW window_name AS ( window_definition ) [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS { FIRST | LAST } ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start [ ROW | ROWS ] ]
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES } ]
    [ FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE } [ OF table_name [, ...] ] [ NOWAIT | SKIP LOCKED ]
```

Однотабличные запросы

```
SELECT col1,col2,col3
  FROM table_name
 WHERE expression
```

Выборка из таблицы

```
SELECT ID,name,countrycode,pulation
   FROM city;
```

Арифметика в столбцах

```
SELECT id*200, name, countrycode+10
   FROM city;
```

Синонимы для столбцов

```
SELECT  2+3 sign, countrycode cc
  FROM  city;
```

Предикат условия **where**

```
SELECT * FROM city WHERE id = 123;


#конструкция для набора IN
SELECT *
  FROM city
 WHERE id in (123,7,5);


#выбор по шаблону LIKE
SELECT * FROM city
 WHERE name like '%Petersburg%';
```

LIMIT и OFFSET

```
SELECT список_выборки
    FROM табличное_выражение
    [ ORDER BY ... ]
    [ LIMIT { число | ALL } ]
[ OFFSET число ]
```

Если указывается число **LIMIT**, в результате возвращается не больше заданного числа строк

OFFSET указывает пропустить указанное число строк, прежде чем начать выдавать строки. OFFSET 0 равносильно отсутствию указания OFFSET, как и OFFSET с аргументом NULL.

LIMIT и OFFSET

Пример:

```
SELECT *
   FROM sputnik
 ORDER BY NAME DESC
 LIMIT 2 OFFSET 4
```

AND | OR

```
SELECT  2+3 sign, countrycode cc
  FROM  city c
 WHERE  c.id > 10
   AND  cc <= 100
    OR  cc >= 10;
```

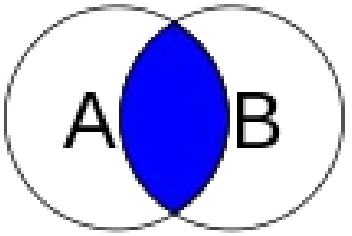
AND | OR

```
SELECT employee_id, last_name, first_name
   FROM employees
 WHERE (last_name = 'Ivanov')
        OR (last_name = 'Petrov' AND state = 'Florida')
        OR (last_name = 'Sidorov' AND status = 'Active' AND
state = 'Nevada');
```

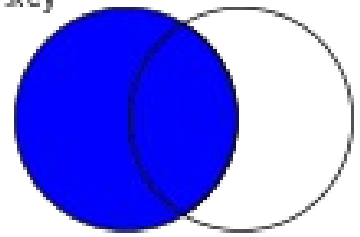
Многотабличные запросы

SQL JOINS

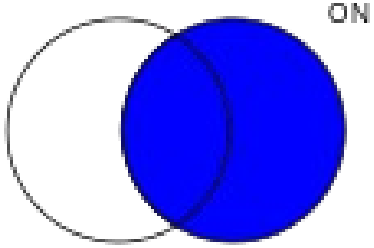
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key



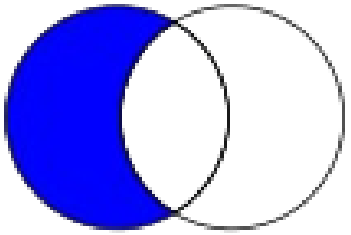
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key



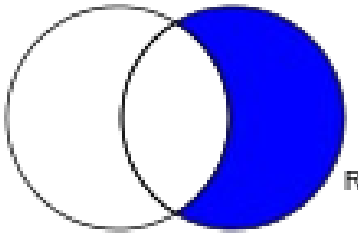
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key



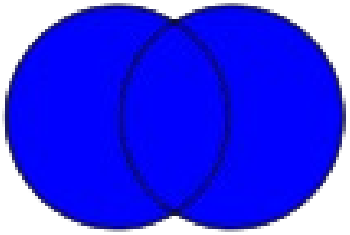
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL



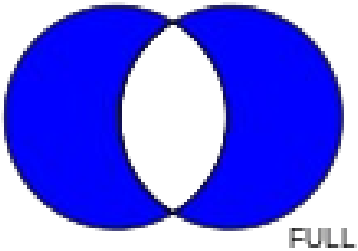
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key



SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL



This work is licensed under a Creative Commons Attribution 3.0 Unported License.
Author: http://commons.wikimedia.org/wiki/User:Arbeck

Многотабличные запросы

CROSS JOIN: Перекрестное соединение, возвращает комбинации каждой записи первой таблицы с каждой записью второй таблицы.
INNER JOIN: Внутренним соединением называется перекрестное соединение,из результатов которого часть записей исключена по условию запроса.
LEFT JOIN: В левом внешнем соединении для КАЖДОЙ ЗАПИСИ ЛЕВОЙ таблицы ищется соответствие среди записей правой таблицы.
RIGHT JOIN: Правое внешнее соединение ищет в левой таблице соответствия для правой таблицы.

INNER JOIN

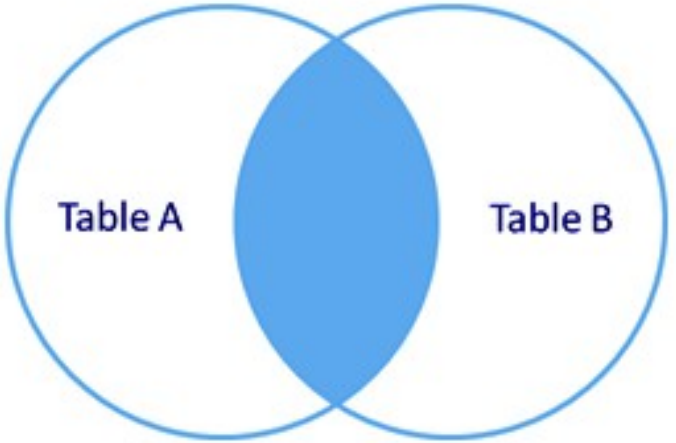
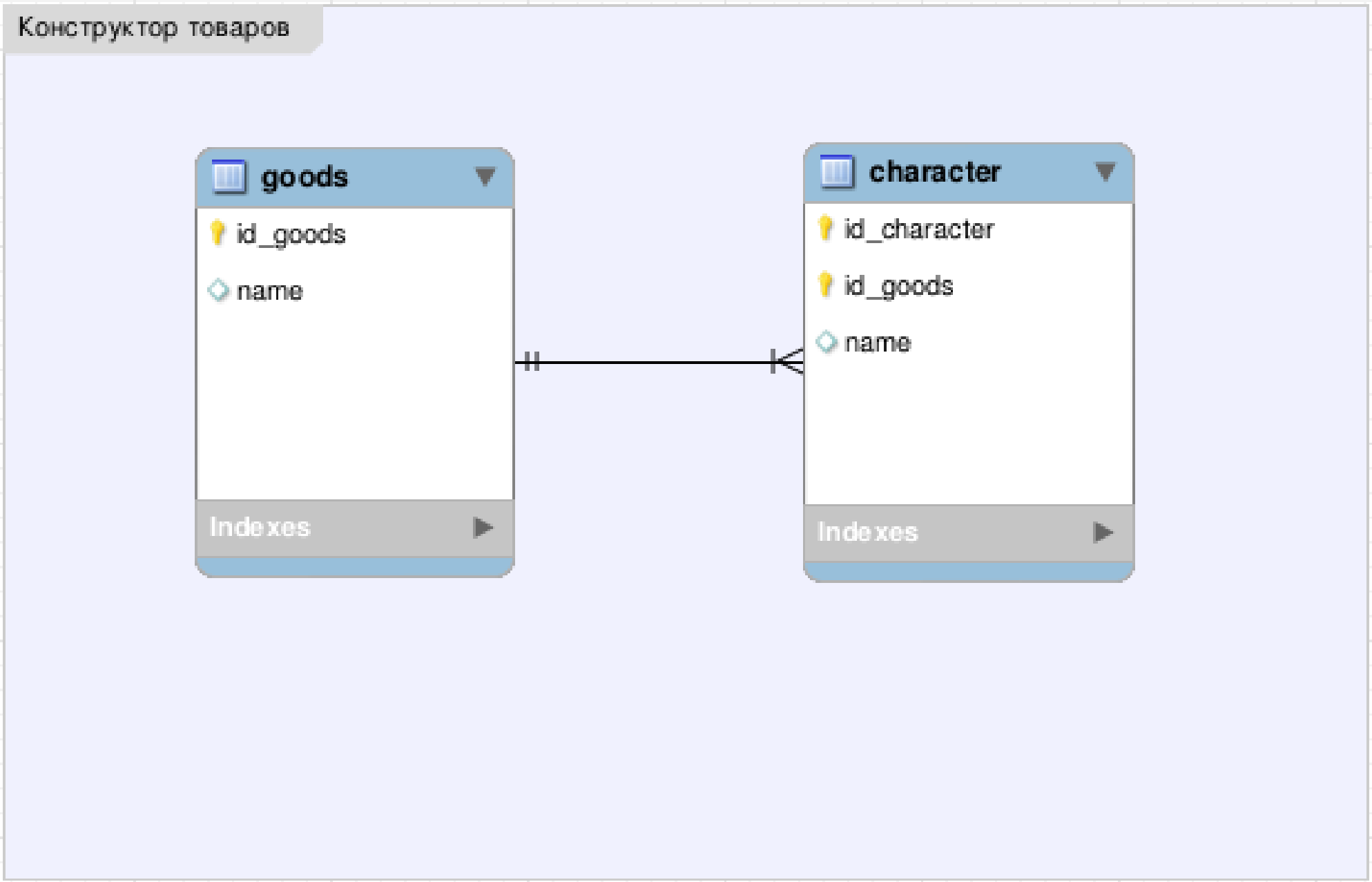


рис. Inner join

Шаблон запроса:

```
SELECT  a.name , b.value
  FROM  table1 a, table2 b
 WHERE  a.id = b.id
```

Модель


Конструктор товаров
goods
id_goods
name
Indexes
character
id_character
id_goods
name
Indexes

Данные

GOODS

ID_GOODS	NAME
1	Книга
2	Жесткий диск
3	Системный блок
4	Монитор

CHARACTER

ID_CHARACTER	ID_GOODS	NAME
1	1	Автор
2	1	Кол-во страниц
3	1	Издательство
4	1	Год выпуска
5	2	Объем
6	2	**Скорость вращения шпинделя**
7	2	Интерфейс
8	3	Жесткий диск
9	4	Процессор

Запрос
(связь по ключу PK = FK)

```
SELECT g.name, ch.name
  FROM goods g, character ch
 WHERE g.id_goods = ch.id_goods
   AND g.id_goods = 1
```

Какой результат получим ?

CROSS JOIN

```
SELECT g.name, ch.name
  FROM goods g, character ch
```

LEFT JOIN

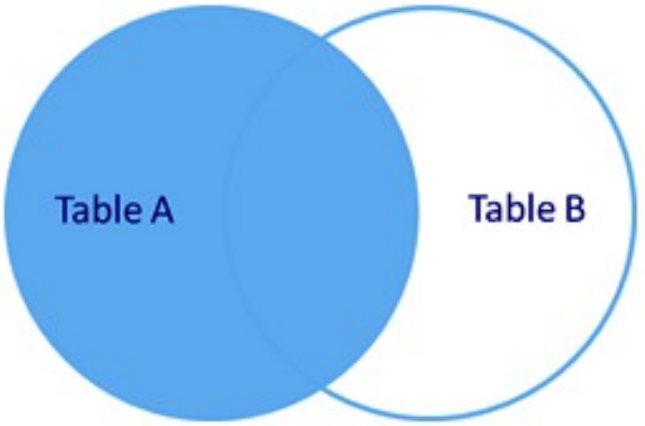


рис. Left join

получим все товары у которых заданы и не заданы характеристики.

```
SELECT a.name, b.name
 FROM goods a left join characters b
          on a.id_goods = b.id_goods;
```

фильтр is not null

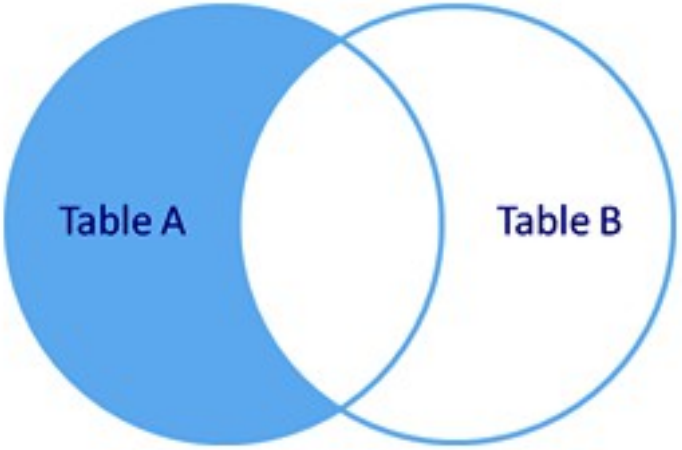
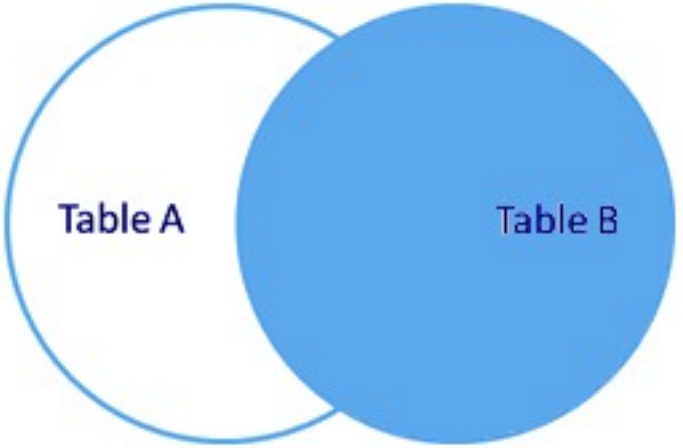


рис. Left outer join с фильтрацией по полю

Получим только те товары у которых не задана характеристика.

```
SELECT a.name, b.name
  FROM goods a
       left join characters b
    on a.id_goods = b.id_goods
 WHERE b.name IS NULL ;
```

RIGHT JOIN



получим все характеристики у которых заданы и не заданы товары.

```
SELECT a.name, b.name
  FROM goods a
       right join characters b
    on a.id_goods = b.id_goods;
```

Полное объединение

Союзы

```
SELECT n from numders1;
 UNION
SELECT n from mumbers2;
```

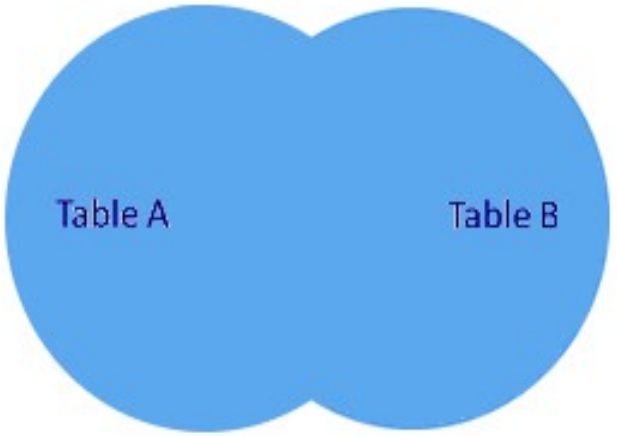
UNION – Объединяет в одну таблицу результаты 2-х и более запросов.

UNION ALL – Для получения списка со всеми дубликатами.

INTERSECT – Возвращает пересечениерезультатов нескольких запросов.

EXCEPT – Возвращает исключение результатов второго запроса из первого.

FULL OUTER JOIN



получим полное пересечение соединений

```
SELECT a.name, b.name
  FROM goods a
  left join characters b
  ON a.id_goods=b.id_goods
UNION
SELECT a.name, b.name
  FROM goods a
  right join characters b
  ON a.id_goods=b.id_goods;
```

Задание.

Для разрабатываемой модели построить многотабличные запросы исходя из бизнес-логики.

Вложенные подзапросы

- Уровень вложенности
- Конструкция NOT EXISTS
- Конструкция ANY, ALL
- Запрос в виде таблице
- Запрос в столбце
- Создание таблицы с помощью запроса

Конструкция **NOT EXISTS, EXISTS**

В результате проверки на существование (EXISTS) можно выяснить, содержится ли в таблице результатов вложенного запроса хотя бы одна строка.

```
SELECT login
  FROM user AS u
 WHERE u.dt_create BETWEEN '2017-01-01' AND '2017-07-01'
   NOT EXISTS (SELECT 1 FROM grp g
                WHERE g.id_user = u.id_user);
```

Конструкция **ANY**

В проверки ANY используется один из шести операторов сравнения (=, <>, <, <=, >, >=) для того чтобы сравнить одно проверяемое значение со столбцом данных, возвращенным вложеным запросом. Проверяемое значение поочередно сравнивается с каждым значением, содержащимся в столбце. Если любое из этих сравнений дает результат TRUE, то проверка ANY возвращает значение TRUE

```
SELECT login
  FROM user AS u
 WHERE u.dt_create BETWEEN '2017-01-01' AND '2017-07-01'
   AND 10 < ANY (SELECT id_group
                   FROM group g
                  WHERE g.id_user = u.id_user );
```

Конструкция **ALL**

В проверки ALL как и в ANY используется один из шести операторов (=, <>, <, <=, >, >=) для сравнения одного проверяемое значения со столбцом данных, возвращенным вложеным запросом. Проверяемое значение поочередно сравнивается с каждым значением, содержащимся в столбце. Если все сравнения дают результат TRUE, то проверка ANY возвращает значение TRUE

```
SELECT login
  FROM user AS u
 WHERE u.dt_create BETWEEN '2017-01-01' AND '2017-07-01'
   AND 10 < ALL (SELECT id_group
                   FROM group g
                  WHERE g.id_user = u.id_user);
```

Запрос в виде таблицы

Результат выполнения запроса можно оформить в виде таблицы задав ее в конструкции FROM.

```
SELECT login
  FROM user a, (SELECT id_user
                  FROM group
                 WHERE id_group in(1,3))b
 WHERE a.id_user = b.id_user
```

Запрос вместо столбца

Результат выполнения запроса можно вставить в качестве столбца.

```
SELECT id_user,(SELECT max(id_user) FROM group) max_id
  FROM user
```

Таблица из запроса

Новую таблицу можно создать на базе подзапроса из существующей таблице или множества таблиц.

```
CREATE TABLE new_table
  AS (SELECT * FROM old_table)
```

```
CREATE TABLE new_table
  AS (SELECT column1, column2, ... column_n  FROM
old_table1, old_table2, ... old_table_n);
```