БАЗЫ ДАННЫХ

Сложные запросы

Связывание таблиц

При простых выборках мы выбираем данные только из одной таблицы. В этом случае у нас в операторе FROM указывается одна таблица:

```
SELECT * FROM `users` …;
```

Для выборки из нескольких таблиц их перечисляют через запятую:

```
SELECT * FROM `users`, `comments` …;
```

Связывание таблиц

Однако указание нескольких таблиц само по себе зачастую не дает желаемого результата: данные из таблиц просто «перемножаются» между собой.

Для логической связи между таблицами необходимо добавить условие, в котором будут связаны первичный ключ основной таблицы и внешний ключ ссылающейся таблицы.

Связывание таблиц

```
SELECT * FROM `users`, `comments`
WHERE `users`.`id` = `comments`.`user_id`;
```

Обратите внимание: здесь мы обращаемся к столбцам по полному имени с указанием названия таблицы. Это необходимо для устранения двусмысленности, ведь в обоих таблицах есть поле `id`, а значит нужно указать к какой именно таблице он принадлежит.

Уникальное название поля можно описать только именем. Но, как правило, лучше сразу указывать полное имя для понимания запроса.

Связывание таблиц

Обычно указание полного имени столбца визуально утяжеляет запрос, поэтому часто применяют псевдонимы для таблиц:

```
SELECT *
FROM
    `users` AS `u`,
    `comments` AS `c`
WHERE
    `u`.`id` = `c`.`user_id`;
```

Связывание таблиц

При обращении к столбцу по полному имени можно опускать квотирование, т.к. в этом случае не происходит смешивание имен и операторов (даже если имя выглядит как оператор):

```
SELECT *
FROM
    `users` AS u,
    `comments` AS c
WHERE
    u.id = c.user_id;
```

Связывание таблиц

Обычно помимо логического условия для связывания таблиц могут присутствовать и другие условия для фильтрации:

```
SELECT *
FROM users AS u, comments AS c
WHERE u.id = c.user_id AND c.is_published = 1;
```

Это может визуально усложнить запрос, ведь при редактировании условия можно случайно удалить связующее условие.

Запросы с JOIN

Чтобы изолировать связующие условия можно прибегнуть к запросам через оператор JOIN:

```
SELECT *
FROM users AS u
JOIN comments AS c ON u.id = c.user_id
WHERE c.is_published = 1;
```

Теперь условие для связи всегда находится рядом с соответствующей таблицей.

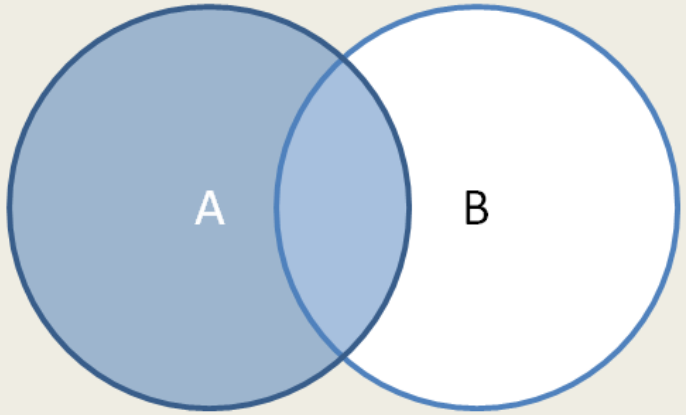
Запросы с JOIN

JOIN-запросы имеют еще и дополнительную функциональность. Мы можем задать алгоритм связи:

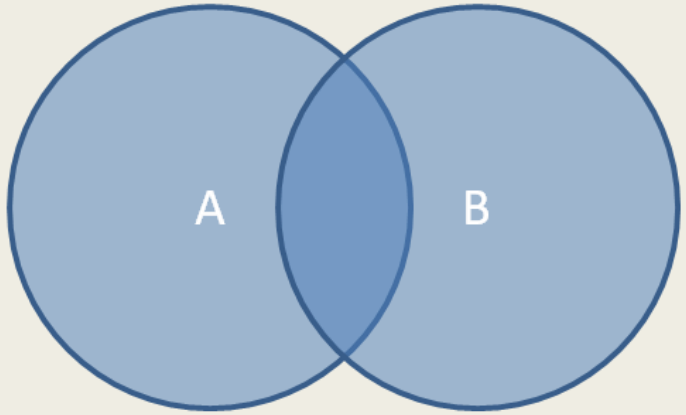
```
INNER JOIN
CROSS JOIN
OUTER JOIN:
    RIGHT OUTER JOIN
    LEFT OUTER JOIN
    FULL OUTER JOIN
```

Запросы с JOIN

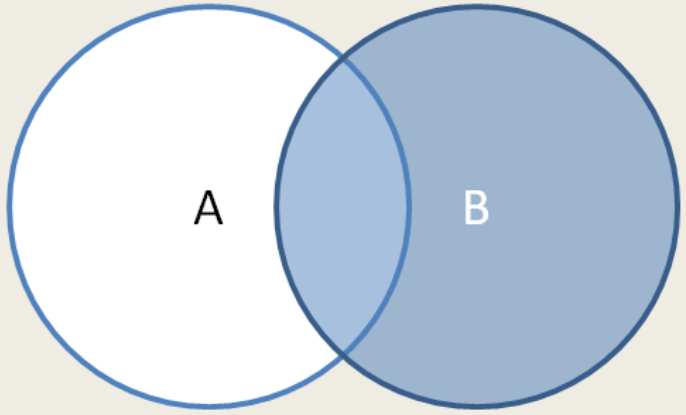
Left Join



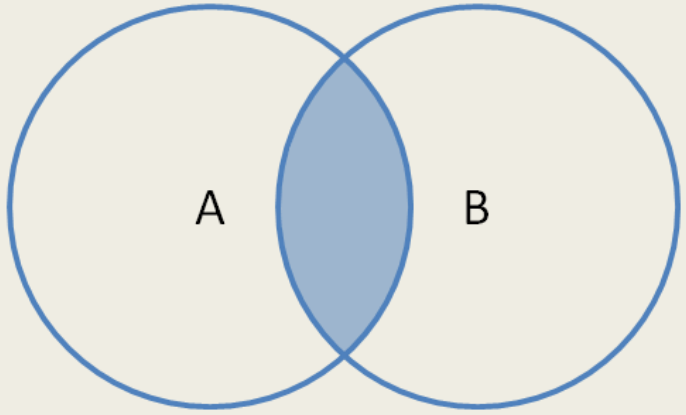
Outer Join / Join



Right Join



Inner Join



Вложенные запросы

Вложенные запросы позволяют соединить 2 запроса в 1:

```
SELECT * FROM `students`
WHERE `id` = ( SELECT `student_id`
               FROM `student_marks`
               GROUP BY `student_id`
               ORDER BY AVG(`mark`) DESC
               LIMIT 1 );
```

Вложенные запросы

Вложенные запросы позволяют соединить 2 запроса в 1:

```
SELECT * FROM `subjects`
WHERE `id` IN ( SELECT `subject_id`
                FROM `subject_schedules`
                WHERE `date` = CURDATE() );
```

Вложенные запросы

Для вложенных запросов характерны следующие особенности:

1. При сравнении с одним значением (через =, LIKE, >, < и их комбинации) необходимо, чтобы подзапрос вернул только 1 строку с 1 столбцом.
2. При сравнении со списком (через IN) подзапрос может вернуть несколько строк, но по-прежнему только с 1 столбцом.