При проектировании баз данных, каждая таблица относится к описанию одной сущности. Большой задачей является нормализация данных при представлении в таблицах. В результате проектирования часто возникает необходимость отдельные сущности хранить в разных таблицах, но между таблицами при этом существуют связи. Простым примером здесь могут служить сущность «Студенты» (ФИО, дата рождения, принадлежность группе) и сущность «Группы» (номер группы, факультет). Связь устанавливается через номер группы – понятно, что каждый студент учится в какой-то группе, которая в свою очередь находится на определенном факультете. Но в самой таблице «Студенты» вовсе не нужно для каждого студента прописывать принадлежность к факультету, достаточно установить связь, к какой группе он относится, и потом уже с помощью запросов особого типа получить сведения по группе. Подробную информацию о нормализации данных при проектировании реляционных таблиц можно получить из учебного пособия А.А. Рузакова «Управление данными» [17, с. 32].

Связи между таблицами MySQL выполняются в результате запроса JOIN. Пусть имеются две таблицы:

Students (id, fio, birthday, groupid);

Groups (id, name, faculty).

Связь устанавливается через соединение поля groupid таблицы Students и поля id таблицы Groups. Задача запроса – найти одинаковые значения в данных полях в обеих таблицах. Чтобы это произошло, значения id в таблице Groups должны быть уникальными, а вот значения groupid могут повторяться.

Таблица 4

Таблица Students

id	fio	birthday	groupid
1	Иванов Иван Иванович	01.01.1990	1
2	Петров Петр Петрович	02.02.1990	2
3	Сидорова Мария Владимировна	03.03.1989	1
4	Никитина Елена Викторовна	04.04.1990	3

Таблица 5

Таблица Groups

id	name	faculty
1	213-092-5-1	Математики, физики, информатики
2	409-053-5-1	Филологический
3	302-041-5-1	Исторический
4	511-066-5-1	Иностранных языков

Чтобы вывести, в какой группе и на каком факультете учится каждый студент, запрос MySQL будет выглядеть следующим образом:

```
SELECT * FROM Students, Groups WHERE Students.groupid = Groups.id
```

В результате запроса получим следующие сведения (рисунок 29).

+ Параметры

id	fio	birthday	groupid	id	name	faculty
1	Иванов Иван Иванович	1990-01-01	1	1	213-092-5-1	Математики, физики, информатики
3	Сидорова Мария Владимировна	1989-03-03	1	1	213-092-5-1	Математики, физики, информатики
2	Петров Петр Петрович	1990-02-02	2	2	409-053-5-1	Филологический
4	Никитина Елена Викторовна	1990-04-04	3	3	302-041-5-1	Исторический

Рис. 29. Результат запроса на соединение

Алгоритм здесь несложный: берётся первая запись из таблицы Students. Далее берётся её id и анализируются все записи из таблицы Groups, добавляя в результат те, у которых groupid равен id из таблицы Groups. Таким образом на первой итерации собираются все сведения у первого студента. На второй итерации собираются все сведения у второго студента и так далее.

Оператор JOIN выполняет полное объединение. Операторы JOIN, INNER JOIN и «,» (запятая) с указанием условий объединения дают пересечение для всех совпадающих значений из условия.

```
SELECT t1.*, t2.* FROM t1, t2 ON t1.i1 = t2.i2
```

Аналогично можно использовать и такую запись

```
SELECT t1.*, t2.* FROM t1 INNER JOIN t2 ON t1.i1 = t2.i2
```

Операторы LEFT JOIN, RIGHT JOIN выполняют, соответственно, левое и правое объединения таблиц.

При использовании оператора LEFT JOIN выбираются все значения из левой таблицы (первой указанной), после чего к ним подбираются соответствующие значения из правой, если таких нет, то все значения для правой таблицы в этой строке равны NULL.

```
SELECT t1.*, t2.* FROM t1 LEFT JOIN t2 ON t1.i1 = t2.i2
```

RIGHT JOIN противоположен LEFT JOIN: всё абсолютно также, но выбираются все значения для правой таблицы из выражения.

При указании любого из операторов JOIN можно использовать ключевое слово USING() вместо ON, если названия столбцов в обеих таблицах одинаковы.