MySQL – это система управления базами данных (СУБД) с открытым кодом. Это высо-

копроизводительная и масштабируемая СУБД с множеством программных интерфейсов.

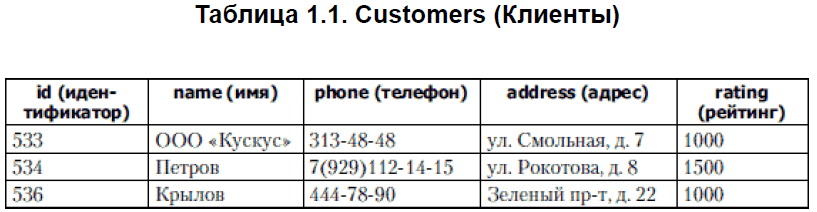
Она обладает огромными функциональными возможностями и подходит для решения самых

разных задач.

**Таблицы**

Реляционная база данных существует в виде таблиц, имеющих свои имена. На пересе-

чении каждого столбца и каждой строки располагается одно значение.



Строки таблицы могут храниться в произвольной последовательности и не должны

повторяться.

Каждый столбец таблицы имеет имя и *тип данных,* которому соответствуют все значе-

ния в столбце. Так, в нашем примере столбцы с именами id и rating – числовые, а с именами

name, phone и address – символьные.

По существу, таблица реляционной базы данных представляет собой набор информа-

ции об однотипных объектах. При этом каждая строка содержит сведения об одном объекте,

а каждый столбец – значения некоторого атрибута этих объектов. Например, строка с иден-

тификационным номером 533 содержит информацию об объекте, у которого атрибут name

(имя) имеет значение ООО «Кускус», атрибут phone (телефон) – значение 313-48-48 и т. д.

**Первичный ключ**

*Первичный ключ* – это минимальный набор столбцов, совокупность значений которых

однозначно определяет строку. Это означает, что в таблице не должно быть строк, у которых

значения во всех столбцах первичного ключа совпадают, при этом ни один столбец нельзя

исключить из первичного ключа, иначе это условие нарушится.

На практике первичным ключом служит специальный столбец, значения которого

автоматически задает СУБД. Например, в таблице Customers (Клиенты) (см. табл 1.1) это

столбец id (идентификатор). Использовать такой искусственный первичный ключ значи-

тельно проще, чем естественный (основанный на атрибутах объекта). Например, в таблице

Customers столбец name (имя) не может быть первичным ключом, так как имена клиен-

тов могут совпадать; а первичный ключ из столбцов name (имя) и phone (телефон) был

бы слишком громоздким. Дополнительными преимуществами искусственного ключа явля-

ются гарантированная уникальность значений (ее обеспечивает СУБД), постоянство значе-

ний (может меняться значение атрибута, но не значение искусственного ключа), а также

числовой тип данных (поиск по числовым значениям выполняется намного быстрее, чем по

символьным).

**Связи между таблицами. Внешний ключ**

Реляционная база данных – это не просто набор таблиц. Объединить разрознен-

ные фрагменты информации в единую структуру данных позволяют *связи* между табли-

цами, посредством которых строка одной таблицы сопоставляется строке (строкам) дру-

гой таблицы. Благодаря связям можно извлекать информацию одновременно из нескольких

таблиц (например, выводить с помощью одного запроса и сведения о клиенте, и сведе-

ния о его заказах), избегать дублирования информации (не требуется в каждом заказе хра-

нить адрес клиента), поддерживать полноту информации (не хранить сведения о заказанном

товаре, если в базе данных отсутствует его описание) и многое другое.

Существует три типа связей, устанавливаемых между таблицами в базе данных.

***• Связь «один ко многим».***

Этот тип связи используется чаще всего. В этом случае одна или несколько строк

таблицы *A* ссылаются на одну из строк таблицы *B.*

Для установки связи между таблицами в дочернюю таблицу добавляется *внешний ключ*

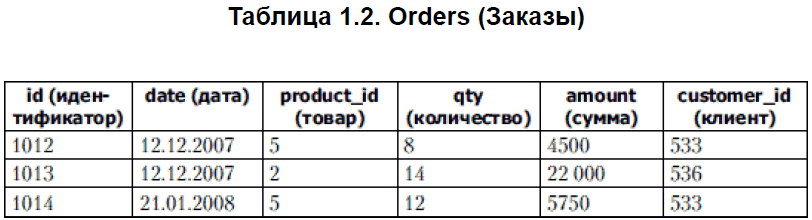
(foreign key) – один или несколько столбцов, содержащих значения первичного ключа роди-

тельской таблицы (иными словами, во внешнем ключе хранятся *ссылки* на строки родитель-

ской таблицы).

Рассмотрим таблицу, которая содержит сведения о заказах, сделанных клиентами, и

является дочерней по отношению к таблице Customers (Клиенты) (табл. 1.2).



В таблице Orders внешним ключом является столбец customer\_id (клиент), в кото-

ром содержатся номера клиентов из таблицы Customers (Клиенты). Таким образом, каждая

строка таблицы Orders ссылается на одну из строк таблицы Customers. Например, строка с

идентификационным номером 1012 содержит в столбце customer\_id (клиент) значение 533:

это означает, что заказ № 1012 сделан клиентом ООО «Кускус».

Столбец product\_id таблицы Orders также является внешним ключом – он содержит

номера товаров из столбца id (идентификатор) таблицы Products (Товары). Таким образом,

таблица Orders является дочерней по отношению к таблицам Customers и Products.

***• Связь «один к одному».***

Такая связь между таблицами означает, что каждой строке одной таблицы соответ-

ствует одна строка другой таблицы, и наоборот. Например, если требуется хранить паспорт-

ные данные клиентов, можно создать таблицу Passports (Паспорта), связанную отношением

«один к одному» с таблицей Customers (Клиенты).

Таблицы, соединенные связью «один к одному», можно объединить в одну. Две

таблицы вместо одной используют по соображениям конфиденциальности (например,

можно ограничить доступ пользователей к таблице Passports), для удобства (если в единой

таблице слишком много столбцов), для экономии дискового пространства (в дополнитель-

ную таблицу выносят те столбцы, которые часто бывают пустыми, тогда дополнительная

таблица содержит значительно меньше строк, чем основная, и обе они занимают меньше

места, чем единая таблица).

Связь «один к одному» может быть организована так же, как связь «один ко многим», –

с помощью первичного ключа родительской таблицы и внешнего ключа дочерней. Другой

вариант – связь посредством первичных ключей обеих таблиц, при этом связанные строки

имеют одинаковое значение первичного ключа.

***• Связь «многие ко многим».***

Этот тип связи в реляционной базе данных реализуется только с помощью вспомога-

тельной таблицы. Например, если потребуется включить в заказ несколько наименований

товаров, связь «многие ко многим» между таблицами Orders (Заказы) и Products (Товары)

можно организовать с помощью вспомогательной таблицы Items (Позиции заказа), содержа-

щей столбцы product\_ id (номер товара из таблицы Products), qty (количество товаров дан-

ного наименования в заказе) и order\_id (номер заказа из таблицы Orders). При этом столбцы

product\_id и qty из таблицы Orders исключаются. Таким образом, таблица Items будет дочер-

ней по отношению к таблицам Orders и Products и каждая строка таблицы Items будет соот-

ветствовать одному наименованию товара в заказе.

**Целостность данных**

*Целостностью данных,* хранимых в СУБД, называется их корректность и непротиво-

речивость.

Базовыми требованиями целостности, которые должны выполняться в любой реляци-

онной базе данных, являются *целостность сущностей* и *целостность связей* (ссылочная

целостность).

Целостность сущностей означает, что в каждой таблице есть первичный ключ – уни-

кальный идентификатор строки. Первичный ключ не должен содержать повторяющихся и

неопределенных значений. Например, если в таблицу Customers (Клиенты) добавить еще

одну строку с идентификатором 533 (притом что одна строка с таким идентификатором уже

существует в таблице), то целостность сущностей будет нарушена и невозможно будет опре-

делить, кому из этих двух клиентов с одинаковыми идентификаторами принадлежат заказы

№№ 1012 и 1014.

Целостность связей означает, что внешний ключ в дочерней таблице не содержит

значения, отсутствующие в первичном ключе родительской таблицы. Иными словами,

строка дочерней таблицы не должна ссылаться на несуществующую строку родительской

таблицы.

Например, в таблицу Orders

(Заказы) добавлена строка, содержащая в столбце customer\_id значение 999. Здесь нару-

шится целостность связи между таблицами Customers и Orders: с одной стороны, заказ не

является «ничьим», так как в этом случае в столбце customer\_id было бы установлено зна-

чение NULL, с другой стороны, невозможно выяснить имя и адрес клиента, сделавшего этот

заказ.

Как видно из приведенных примеров, если целостность данных нарушена, то с

ними невозможно нормально работать. Поэтому поддержание целостности данных является

одной из основных функций любой СУБД.

Помимо проверки корректности значения внешнего ключа при

добавлении и изменении строк дочерней таблицы, необходимо также предотвратить нару-

шение ссылочной целостности при удалении и изменении строк родительской таблицы. Для

этого существует несколько способов.

• ***Запрет (RESTRICT)***: если на строку родительской таблицы ссылается хотя бы одна

строка дочерней таблицы, то удаление родительской строки и изменение значения первич-

ного ключа в такой строке запрещаются. Например, не допускается удаление информации

о клиенте из таблицы Customers (Клиенты), если у этого клиента есть зарегистрированные

заказы, то есть строки в таблице Orders (Заказы), которые ссылаются на строку со сведени-

ями об этом клиенте.

• ***Каскадное удаление/обновление (CASCADE):*** при удалении строки из родительской

таблицы автоматически удаляются все ссылающиеся на нее строки дочерней таблицы; при

изменении значения первичного ключа в строке родительской таблицы автоматически обно-

вляется значение внешнего ключа в ссылающихся на нее строках дочерней таблицы.

Например, при удалении записи о клиенте из таблицы Customers (Клиенты) автома-

тически удаляются сведения о заказах этого клиента, то есть соответствующие строки в

таблице Orders (Заказы).

• ***Обнуление (SET NULL):*** при удалении строки и при изменении значения первичного

ключа в строке значение внешнего ключа во всех строках, ссылающихся на данную, автома-

тически становится неопределенным (NULL). Например, при удалении записи о клиенте из

таблицы Customers (Клиенты) заказы этого клиента автоматически становятся «ничьими»,

то есть в соответствующих строках таблицы Orders (Заказы) в столбце customer\_id (клиент)

устанавливается значение NULL.

С понятием целостности данных тесно связано понятие *транзакции.* Транзакцией

называется группа связанных операций, которые должны быть либо все выполнены, либо

все отменены. Если при выполнении одной из операций происходит ошибка или сбой, то

транзакция отменяется. При этом все уже внесенные другими операциями изменения авто-

матически аннулируются и восстанавливается исходное состояние базы данных. Важнейшее

применение транзакций – это объединение тех операций, которые, будучи выполнены по

отдельности, могут нарушить целостность данных. Например, рассмотренная выше опера-

ция каскадного удаления выполняется как единая транзакция: строка родительской таблицы

должна быть удалена вместе со всеми ссылающимися на нее строками дочерней таблицы, а

если по каким-либо причинам одну из этих строк удалить невозможно, то не будет удалена

ни одна из строк.