Реляционные базы данных

Наиболее значимыми элементами реляционной модели являются отношения, которые чаще всего называют **таблицами**.
Отношения — это набор кортежей, или строк в таблице, где каждый кортеж имеет набор атрибутов, или столбцов.
Столбец — это наименьшая организационная структура реляционной базы данных, представляющая различные ячейки, которые определяют записи в таблице. Отсюда происходит более формальное название — **атрибут**. Можно рассматривать каждую строку в качестве уникального экземпляра чего-либо, что может находиться в таблице.

При создании столбцов вы указываете тип данных, определяющий, какие записи могут вноситься в данный столбец. СУБД часто используют собственные уникальные типы данных, которые могут не быть напрямую взаимозаменяемы с аналогичными типами данных из других систем.

Рассмотрим на примере MySQL. Некоторые распространённые типы данных включают даты, строки (char, varchar), целые числа (int), логические значения (bool), бинарные данные (blob) и так далее.
В реляционной модели каждая таблица содержит по крайней мере один столбец, который можно использовать для уникальной идентификации каждой строки. Он называется **первичным ключом**. Это важно, поскольку это означает, что пользователям не нужно знать, где физически хранятся данные на компьютере. Их СУБД может отслеживать каждую запись и возвращать её, в зависимости от конкретной цели.
Если у вас есть две таблицы, которые вы хотите связать друг с другом, можно сделать это с помощью внешнего ключа. **Внешний ключ** — это, по сути, копия основного ключа одной таблицы («предка»), вставленная в столбец другой таблицы («потомка»).

Структурные элементы реляционной модели помогают хранить данные в структурированном виде, но хранение имеет значение только в том случае, если вы можете извлечь эти данные. Для извлечения информации из СУБД можно создать запрос, то есть структурированный запрос на набор информации.

Язык SQL

Большинство реляционных баз данных используют **язык SQL** для управления данными и отправки запросов. SQL позволяет фильтровать результаты и обрабатывать их с помощью различных пунктов, предикатов и выражений, позволяя вам контролировать, какие данные появятся в результате.

Язык SQL поддерживает следующие группы инструкций:

- язык описания данных — DDL (Data Definition Language);
- язык манипулирования данными — DML (Data Manipulation Language);

- язык управления транзакциями.

Инструкции DDL предназначены для создания, изменения и удаления объектов базы данных.

Инструкция	Назначение
CREATE	Создание новых объектов (таблиц, полей, индексов и т. д.)
DROP	Удаление объектов
ALTER	Изменение объектов

Инструкции DML позволяют выбирать данные из таблиц, а также добавлять, удалять и изменять их.

Инструкция	Назначение
SELECT	Выполнение запроса к базе данных с целью отбора записей, удовлетворяющих заданным критериям
INSERT	Добавление записей в таблицы базы данных
UPDATE	Изменение значений отдельных записей и полей
DELETE	Удаление записей из базы данных

DML-операции фигурируют в назначении привилегий для пользователей базы данных (речь идёт о внутренних пользователях базы данных, никак не связанных с пользователями в операционной системе). Это позволяет обеспечить уже упомянутую гибкость в настройках доступа к данным. Пользователю, который использует программу для доступа к базе, мы можем позволить только просматривать информацию в одной таблице, но не добавлять или удалять информацию в другой таблице.

Третьей составной частью SQL является язык управления транзакциями. **Транзакция** — это логически завершённая единица работы, содержащая одну или несколько элементарных операций обработки данных (то есть один или несколько SQL-запросов). Все действия, составляющие транзакцию, должны либо выполниться полностью, либо полностью не выполниться.
Инструкции языка управления транзакциями.

Инструкция	Назначение
COMMIT	Фиксация в базе данных всех изменений, сделанных текущей транзакцией
SAVEPOINT	Установка точки сохранения (начала транзакции)
ROLLBACK	Откат изменений, сделанных с момента начала транзакции

В большинстве СУБД элементарные команды, составляющие тело транзакции, выполняются над некоторой буферной копией данных, и только если удаётся успешно довести их до конца, происходит окончательное обновление основной базы.

Транзакция начинается от точки сохранения, задаваемой инструкцией SAVEPOINT, и может быть завершена по команде COMMIT или прервана по команде ROLLBACK (откат). Также в современных системах управления данными предусмотрены средства автоматического отката транзакций при возникновении системных сбоев. Таким образом, механизм управления транзакциями является важнейшим инструментом поддержания целостности данных.

Преимущества и недостатки реляционных баз

Поддержка транзакций является огромным преимуществом реляционных баз. Она обеспечивает целостность данных, защищая от неконсистентности в случае сбоя.

Реляционную базу данных сложно масштабировать горизонтально из-за того, что она разработана для обеспечения целостности, то есть клиенты, отправляющие запросы в одну и ту же базу данных, всегда будут получать одинаковые данные. Если вы масштабируете реляционную базу данных горизонтально по всем машинам, будет трудно обеспечить целостность, так как клиенты могут вносить данные только в один узел, а не во все.

Ещё одно ограничение, существующее в СУБД, заключается в том, что реляционная модель была разработана для управления структурированными данными или данными, которые соответствуют заранее определённому типу данных или, по крайней мере, каким-либо образом предварительно организованы. Однако с распространением персональных компьютеров и развитием сети Интернет появились неструктурированные данные, такие как электронные сообщения, фотографии, видео и пр.

Реляционные базы данных демонстрируют чрезвычайную гибкость. Они используются для построения широкого спектра различных приложений и продолжают эффективно работать даже с большими объёмами данных. Язык SQL также обладает огромным потенциалом и позволяет вам добавлять или менять данные на лету, а также вносить изменения в структуру схем баз данных и таблиц, не влияя на существующие данные.