**Реляционные базы данных**

*Бабаева Александра, ТХ-11*

Реляционные базы данных представляют собой базы данных, которые используются для хранения и предоставления доступа к взаимосвязанным элементам информации. Реляционные [базы данных](https://www.oracle.com/cis/database/what-is-database/) основаны на реляционной модели — интуитивно понятном, наглядном табличном способе представления данных. Каждая строка, содержащая в таблице такой базы данных, представляет собой запись с уникальным идентификатором, который называют ключом. Столбцы таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута, что дает возможность легко устанавливать взаимосвязь между элементами данных.

Пример реляционной базы данных

В качестве примера рассмотрим две таблицы, которые небольшое предприятие использует для обработки заказов продукции. Первая таблица содержит информацию о заказчиках: каждая запись в ней включает в себя имя и адрес заказчика, платежные данные и информацию о доставке, номер телефона и т. д. Каждый элемент информации (атрибут) помещен в отдельный столбец базы данных, которому назначен уникальный идентификатор (ключ) для каждой строки. Во второй таблице (с информацией о заказе) каждая запись содержит идентификатор заказчика, совершившего заказ, название заказанного продукта, его количество, размер или цвет и т. д. Записи в этой таблице не содержат таких данных, как имя заказчика или его контактные данные.

У обеих таблиц есть только один общий элемент — идентификатор столбца (ключ). Благодаря наличию этого общего столбца реляционные базы данных могут устанавливать взаимосвязи между двумя таблицами. Когда приложение для обработки заказов передает заказ в базу данных, база данных обращается к таблице со сведениями о заказах, извлекает сведения о продукции и использует идентификатор заказчика из этой таблицы, чтобы найти сведения об оплате и доставке в таблице с информацией о нем. Затем на складе подбирают нужный продукт, заказчик своевременно получает свой заказ и производит оплату.

Структура реляционных баз данных

Реляционная модель подразумевает логическую структуру данных: таблицы, представления и индексы. Логическая структура отличается от физической структуры хранения. Такое разделение дает возможность администраторам управлять физической системой хранения, не меняя данных, содержащихся в логической структуре. Например, изменение имени файла базы данных не повлияет на хранящиеся в нем таблицы.

Разделение между физическим и логическим уровнем распространяется в том числе на операции, которые представляют собой четко определенные действия с [данными](https://www.oracle.com/cis/database/what-is-data-management/) и структурами базы данных. Логические операции дают возможность приложениям определять требования к необходимому содержанию, в то время как физические операции определяют способ доступа к данным и выполнения задачи.

Чтобы обеспечить точность и доступность данных, в реляционных базах должны соблюдаться определенные правила целостности. Например, в правилах целостности можно запретить использование дубликатов строк в таблицах, чтобы устранить вероятность попадания неправильной информации в базу данных.

Реляционная модель

В первых базах данных данные каждого приложения хранились в отдельной уникальной структуре. Если разработчик хотел создать приложение для использования таких данных, он должен был хорошо знать конкретную структуру, чтобы найти необходимые данные. Такой метод организации был неэффективен, сложен в обслуживании и затруднял оптимизацию эффективности приложений. Реляционная модель была разработана, чтобы устранить потребность в использовании разнообразных структур данных.

Она обеспечила стандартный способ представления данных и отправки запросов, которые могли быть использованы в любых приложениях. Разработчики уяснили, что таблицы являются ключевым преимуществом реляционных баз данных, так как обеспечивают интуитивно понятный, эффективный и гибкий способ хранения структурированной информации и получения к ней доступа.

Со временем, когда разработчики стали использовать язык структурированных запросов (SQL) для записи данных в базу и отправки запросов, стало очевидным и другое преимущество реляционной модели. Вот уже на протяжении многих лет SQL широко используется в качестве языка запросов в базах данных. Он основан на алгоритмах реляционной алгебры и четкой математической структуре, что обеспечивает простоту и эффективность при оптимизации любых запросов к базе данных. Для сравнения: при использовании других подходов приходится создавать отдельные, уникальные запросы.

Преимущества системы управления реляционными базами данных

Компании всех типов и размеров используют простую, но функциональную реляционную модель для обслуживания разнообразных информационных потребностей. Реляционные базы данных применяются для отслеживания товарных запасов, обработки торговых транзакций через Интернет, управления большими объемами критически важных данных заказчиков и т. д. Реляционные базы данных можно рекомендовать для обслуживания любых информационных потребностей, где элементы данных связаны между собой и необходимо обеспечивать безопасное и надежное управление ими на основе правил целостности.

Реляционные базы данных появились в 1970-х годах. На сегодняшний день преимущества реляционного подхода сделали его самой распространенной моделью для баз данных в мире.

Реляционная модель и целостность данных

Реляционная модель наиболее эффективно поддерживает целостность данных во всех приложениях и копиях (экземплярах) базы данных. Например, когда заказчик кладет деньги на счет с помощью банкомата, а затем проверяет баланс на мобильном телефоне, он ожидает, что поступившие средства сразу же отобразятся на счете. Реляционные базы данных отлично подходят для обеспечения целостности данных в различных экземплярах базы в одно и то же время.

Другие типы баз данных не могут одновременно поддерживать целостность больших объемов данных. Некоторые современные типы баз данных, такие как NoSQL, обеспечивают только так называемую «окончательную целостность». Это значит, что, когда выполняется масштабирование данных или несколько пользователей одновременно используют одни и те же данные, необходимо некоторое время на внесение изменений. В некоторых случаях окончательная целостность вполне приемлема (например, для обновления позиций в товарном каталоге), однако для критически важной операционной деятельности бизнеса (например, транзакций с использованием корзины) реляционные базы по-прежнему являются «золотым» стандартом.

Фиксация изменений и атомарность

В реляционных базах данных используются очень детальные и строгие бизнес-правила и политики в отношении фиксации изменений в базе данных (то есть сохранения изменений в данных на постоянной основе). Рассмотрим для примера складскую базу данных, в которой отслеживаются три запчасти, всегда использующиеся в комплекте. Когда одну из них извлекают из товарных запасов, две другие также должны извлекаться. Если одна из трех запчастей недоступна, две другие также не могут быть проданы отдельно, то есть, чтобы в базу данных можно было внести изменения, должны быть доступны все три запчасти. Реляционная база данных не разрешит сохранять изменения, если они не касаются всех трех запчастей. Эту особенность реляционных баз данных называют атомарностью или неразрывностью. Неразрывность необходима для сохранения точности данных в базе и обеспечения соответствия с правилами, нормативными положениями и бизнес-политиками.

Свойства ACID и СУРБД

Транзакции в реляционной базе данных имеют четыре важные характеристики: неразрывность (atomicity), целостность (consistency), изолированность (isolation) и неизменность (durability). Это сочетание получило название ACID.

Неразрывность определяет все элементы, которые необходимы для совершения транзакции в базе данных.

Согласованность или целостность определяет правила сохранения состояния данных после выполнения транзакции.

Изолированность гарантирует, что во избежание путаницы транзакция не повлияет на другие элементы до окончательного сохранения изменений.

Неизменность обеспечивает неизменность данных после сохранения изменений в результате транзакции.

Хранимые процедуры и реляционные базы данных

Доступ к данным включает в себя множество повторяющихся действий. Например, иногда для получения нужного результата простой запрос для получения информации из таблицы необходимо повторить сотню или тысячу раз. Для таких сценариев доступа к базе данных необходимо что-то вроде программного кода. Разработчикам каждый раз писать стандартный код доступа к данным для нового приложения было бы утомительно. К счастью, реляционные базы данных поддерживают хранимые процедуры, представляющие собой блоки кода, к которым можно получить доступ с помощью обычного вызова со стороны кода приложения. Например, одну и ту же хранимую процедуру можно использовать для последовательной маркировки записей в целях удобства пользователей для различных приложений. Хранимые процедуры также помогают разработчикам убедиться в правильной реализации определенных функций данных в приложении.

Блокировки базы данных и параллельный доступ

Когда несколько пользователей или приложений пытаются одновременно изменить одни и те же данные, это может вести к возникновению конфликта в базе. Блокировки и параллельный доступ снижают вероятность конфликтов и способствуют сохранению целостности данных.

Блокировка не разрешает другим пользователям и приложениям получать доступ к данным во время их обновления. В некоторых базах данных блокировка может применяться к целой таблице, что негативно отражается на эффективности приложения. В других типах баз данных, например реляционных базах Oracle, блокировка выполняется на уровне одной записи, оставляя другие записи в таблице доступными. Такой подход помогает сохранить эффективность приложения.

Инструмент параллельного доступа используется, когда несколько пользователей или приложений пытаются одновременно выполнить запросы к одной базе данных. Он обеспечивает доступ пользователей и приложений к базе данных в соответствии с политиками контроля.

Характеристики, на которые следует обратить внимание при выборе реляционной базы данных

Программное обеспечение, которое используется для сохранения, контроля и извлечения данных в базе, а также выполнения к ней запросов, называют системой управления реляционной базой данных (СУРБД). СУРБД обеспечивает интерфейс между пользователями и приложениями и базой данных, а также административные функции для управления хранением данных, их эффективностью и доступом к ним.

При выборе типа базы данных и продуктов на основе реляционных баз данных необходимо учитывать несколько факторов. Выбор СУРБД зависит от потребностей Вашей компании. Задайте себе следующие вопросы.

Каковы наши требования к точности данных? Будем ли мы использовать бизнес-логику для хранения и обеспечения точности данных? Предъявляются ли к нашим данным более строгие требования в отношении точности (например, если Вы работаете с финансовыми данными и отчетностью)?

Нужна ли нам масштабируемость? Какими объемами данных требуется управлять и каков прогнозируемый рост этих объемов? Должна ли модель базы данных поддерживать зеркальные копии (как отдельные экземпляры) в целях масштабирования? Если да, сможем ли мы обеспечивать целостность данных в этих экземплярах?

Насколько важно наличие параллельного доступа? Потребуется ли пользователям и приложениям одновременный доступ к данным? Поддерживает ли ПО базы данных параллельный доступ без ущерба для безопасности?

Каковы наши потребности в эффективности и надежности баз данных? Требуется ли нам высокоэффективная и надежная система? Каковы требования к скорости выполнения запросов? Какие гарантии дает вендор услуг в соответствии с соглашением об уровне обслуживания (SLA) или на случай незапланированного простоя?

Реляционная база данных будущего: автономная база данных

На протяжении последних лет реляционные базы данных улучшали свою производительность, надежность и безопасность и становились проще в обслуживании. Однако их структура становилась все более сложной, и, как следствие, администрирование такой базы данных начало требовать немалых усилий. Вместо того, чтобы использовать свои навыки для разработки инновационных приложений, которые будут приносить прибыль компании, разработчики вынуждены посвящать львиную долю времени управлению базой данных для оптимизации ее эффективности.

Сегодня [автономные технологии](https://www.oracle.com/cis/autonomous-database/) строятся на возможностях реляционной модели, [технологии облачной базы данных](https://www.oracle.com/cis/database/what-is-a-cloud-database/) и [машинном обучении](https://www.oracle.com/cis/data-science/machine-learning/what-is-machine-learning/), чтобы можно было создать реляционную базу данных нового типа. Самоуправляемая база данных (которую также называют автономной) сохраняет все преимущества и возможности реляционной модели и добавляет к ним средства на основе искусственного интеллекта, машинного обучения и автоматизации для мониторинга и оптимизации скорости выполнения запросов и управления. Например, чтобы улучшить скорость выполнения запросов, автономная база данных строит прогнозы и проверяет индексы, а затем применяет лучшие результаты на практике, и все это без участия администратора. Самоуправляемые базы данных постоянно вносят такие улучшения в собственную работу без человеческого вмешательства.

[Автономные](https://www.oracle.com/cis/database/what-is-autonomous-database.html) технологии дают возможность разработчикам больше не тратить время на рутинные задачи обслуживания. Например, больше не нужно заблаговременно определять требования к инфраструктуре. При использовании решения IaaS Вы арендуете ресурсы, например вычислительные мощности или хранилище, получаете доступ к нужным ресурсам по мере необходимости и платите только за те из них, которые использует Ваша компания. Разработчики могут создавать автономные реляционные базы данных всего за несколько шагов, ускоряя процесс разработки приложений.