САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 13
Целостность данных

Агафонов Антон Александрович
д.т.н., профессор кафедры ГИиИБ

Самара

План лекции

- Целостность данных
- Транзакции
  - Требования к транзакциям
  - Уровни изоляции транзакций;
  - Проблемы совместного доступа к данным.
  - Блокировки и MVCC
- Защита данных встроенными средствами СУБД
  - Представления
  - Хранимые процедуры и функции
  - Триггеры
- Итоги курса

Целостность данных

Целостность данных – соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным ограничениям. Ограничения должны быть формально описаны, после чего СУБД должна обеспечивать их выполнение, контролируя те операции модификации данных, которые могут нарушить эти ограничения, и запрещая те операции, которые их действительно нарушают

Не следует путать целостность (непротиворечивость) БД с истинностью (достоверностью) хранимых данных. Достоверность есть соответствие фактов, хранящихся в базе данных, реальному миру.

Целостность БД не гарантирует достоверности (истинности) содержащейся в ней информации, но обеспечивает по крайней мере правдоподобность этой информации, отвергая заведомо невозможные значения.

Целостность данных

Обеспечение целостности данных можно рассматривать как комплексную проблему, которая подразумевает выполнение ряда принципов:

1) определение ограничений целостности на уровне модели данных;
2) модификация данных только посредством корректных транзакций;
3) работа с данными только авторизованных пользователей;
4) применение принципа минимальных привилегий;
5) управление передачей привилегий;
6) разграничение функциональных обязанностей;
7) аудит событий.

Типы ограничений целостности

В отношении БД разделяют следующие типы ограничений целостности:

- **Целостность сущностей** подразумевает что каждая таблица должна иметь первичный ключ и что столбец или столбцы, выбранные в качестве первичного ключа, должны быть уникальными и не могут содержать значений NULL.
- **Целостность ссылок** подразумевает что значение внешнего ключа может быть только в одном из двух состояний. Либо значение внешнего ключа должно содержать значение, являющееся первичным ключом главной таблицы, либо может содержать значение NULL – в этом случае подразумевается, что между объектами, представленными в БД, нет связи или эта связь неизвестна.
- **Целостность домена** подразумевает, что все столбцы в реляционной БД должны быть определены на определенном домене.
- **Пользовательские правила целостности.** К этой категории относится набор правил, указанных пользователем, которые не принадлежат к категориям целостности сущности, домена и ссылочной целостности. Сюда можно отнести ограничения значения по умолчанию (DEFAULT), уникальности (UNIQUE), запрет NULL значений (NOT NULL), специальные условия на значения (CHECK).

Транзакции

Транзакция - это выполняемая от имени определенного пользователя или процесса последовательность операторов манипулирования данными, переводящая базу данных из одного целостного состояния в другое целостное состояние.

Требования к транзакционной системе, обеспечивающие наиболее надежную и согласованную ее работу (**ACID**):

- **A**tomicity — Атомарность;
- **C**onsistency — Согласованность;
- **I**solation — Изолированность;
- **D**urability — Долговечность.

Свойства транзакций

Атомарность (Atomicity). Транзакция выполняется как атомарная операция - либо выполняется вся транзакция целиком, либо она целиком не выполняется. Транзакция не может быть выполнена частично, если в процессе выполнения транзакции возникает какой-то сбой, то все выполненные до этого момента изменения данных должны быть отменены. Транзакция считается успешно выполненной, если в процессе исполнения не возникла какая-либо аварийная ситуация, и все проверки ограничений целостности дали положительный результат.

Согласованность (Consistency). Транзакция переводит базу данных из одного согласованного (целостного) состояния в другое согласованное (целостное) состояние. Внутри транзакции согласованность базы данных может нарушаться. Согласованность достигается вследствие атомарности, т.к. завершенная транзакция гарантирует согласованность данных, а транзакция, нарушающая ограничения целостности или не завершенная из-за сбоя, не изменяет данные.

Свойства транзакций

Изолированность (Isolation). Транзакции недоступны промежуточные результаты других параллельно исполняемых транзакций, а также её промежуточные результаты недоступны другим транзакциям. Другими словами, изолированная транзакция *A* может получить доступ к объекту, обрабатываемому транзакцией *B*, только после завершения B, и наоборот. Таким образом, при одновременной работе двух пользователей с одними и теми же данными они не замечают друг друга, как если бы они работали с этими данными строго по очереди.

Долговечность (Durability). Если транзакция выполнена, то результаты ее работы должны сохраниться в базе данных и не могут быть утеряны или отменены ни при каких обстоятельствах. В случае успешного выполнения транзакции все изменения гарантировано фиксируются в постоянной памяти, даже если в следующий момент произойдет сбой системы. В случае неуспешного завершения транзакции по любой причине - гарантировано отсутствие каких-либо связанных с ней изменений во внешней памяти.

Блокировки

Блокировка – временное ограничение на выполнение некоторых операций обработки данных. Блокировка может быть наложена как на отдельную строку таблицы, так и на всю базу данных. Управлением блокировками на сервере занимается менеджер блокировок, контролирующий их применение и разрешение конфликтов.

Блокировка представляет собой метод управления параллельными процессами, при котором объект БД не может быть модифицирован без ведома транзакции, т.е. происходит блокирование доступа к объекту со стороны других транзакций, чем исключается непредсказуемое изменение объекта.

Различают два вида блокировки:

- блокировка записи – транзакция блокирует строки в таблицах таким образом, что запрос другой транзакции на блокировку этих строк будет отклонен;
- блокировка чтения – транзакция блокирует строки в таблицах таким образом, что запрос другой транзакции на блокировку записи этих строк будет отклонен, а на блокировку чтения – принят.

Блокировки

В СУБД используют протокол доступа к данным, позволяющий избежать проблемы параллелизма:

- транзакция, результатом действия которой является извлечение строки таблицы, обязана наложить блокировку чтения на эту строку;
- транзакция, предназначенная для модификации строки данных, накладывает на нее блокировку записи;
- если запрашиваемая блокировка на строку отклоняется из-за уже имеющейся блокировки, то транзакция переводится в режим ожидания до тех пор, пока блокировка не будет снята;
- блокировка записи сохраняется вплоть до конца выполнения транзакции.

Решение проблемы параллельной обработки БД заключается в том, что строки таблиц блокируются, а последующие транзакции, модифицирующие эти строки, отклоняются и переводятся в режим ожидания.

Если в системе управления базами данных не реализованы механизмы блокирования, то при одновременном чтении и изменении одних и тех же данных несколькими пользователями могут возникнуть проблемы одновременного доступа.

Проблемы совместного доступа к данным

При параллельном выполнении транзакций возможны следующие проблемы:

- **потерянное обновление** (lost update) — при одновременном изменении одного блока данных разными транзакциями теряются все изменения, кроме последнего;
- **«грязное» чтение** (dirty read) — чтение незафиксированных изменений, осуществленных другой транзакцией. В случае перезаписи этих промежуточных значений или отката первой транзакции незафиксированные изменения могут быть отменены, а прочитавшая их транзакция с этого момента станет работать с неверными данными;
- **неповторяющееся чтение** (non-repeatable read) — при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными;
- **фантомное чтение** (phantom reads) — одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. Другая транзакция в интервалах между этими выборками добавляет строки или изменяет столбцы некоторых строк, используемых в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк.

Уровни изоляции транзакций

Под «уровнем изоляции транзакций» понимается степень обеспечиваемой внутренними механизмами СУБД защиты от всех или некоторых видов несогласованности данных, возникающих при параллельном выполнении транзакций.

Стандарт SQL-92 определяет шкалу из четырёх уровней изоляции.

- **READ UNCOMMITTED** (чтение незафиксированных данных) – наименее защищенный уровень изоляции, при котором транзакции способны читать незафиксированные изменения, сделанные другими транзакциями;
- **READ COMMITTED** (чтение фиксированных данных) – исключается «грязное» чтение, транзакция увидит только изменения, зафиксированные другими транзакциями;
- **REPEATABLE READ** (повторяющееся чтение) – накладывает блокировки на обрабатываемые транзакцией строки и не допускает их изменения другими транзакциями. В результате транзакция видит только те строки, которые были зафиксированы на момент ее запуска. Основной недостаток повторяемого чтения – высокая вероятность появления строк-фантомов;
- **SERIALIZABLE** (сериализуемость) – самый надежный уровень изоляции, полностью исключающий взаимное влияние транзакций.

Уровни изоляции транзакций

Уровень изоляции	Потерянное обновление	«Грязное» чтение	Неповторяющееся чтение	Фантомное чтение	Аномалии сериализации
Read uncommited	Исключено	Возможно	Возможно	Возможно	Возможно
Read commited	Исключено	Исключено	Возможно	Возможно	Возможно
Repeatable read	Исключено	Исключено	Исключено	Возможно	Возможно
Serializable	Исключено	Исключено	Исключено	Исключено	Исключено

«Мертвые» блокировки

«Мертвая» блокировка возникает, когда две транзакции блокируют два блока данных и для завершения любой из них нужен доступ к данным, заблокированным ранее другой транзакцией. Для завершения каждой транзакции необходимо дождаться, пока блокированная другой транзакцией часть данных будет разблокирована. Но это невозможно, так как вторая транзакция ожидает разблокирования ресурсов, используемых первой.

Без применения специальных механизмов обнаружения и снятия «мертвых» блокировок нормальная работа транзакций будет нарушена. Для этих целей сервер снимает одну из блокировок, вызвавших конфликт, и откатывает инициализировавшую ее транзакцию.

Для минимизации возможности образования «мертвых» блокировок при разработке кода транзакции следует придерживаться следующих правил:

- выполнять действия по обработке данных в определенном порядке, чтобы не создавать условия для захвата одних и тех же данных;
- минимизировать длительность транзакций;
- применять как можно более низкий из допустимых уровней изоляции.

MVCC

MVCC (MultiVersion Concurrency Control — управление параллельным доступом посредством многоверсионности) — один из механизмов СУБД для обеспечения параллельного доступа к базам данных, заключающийся в предоставлении каждому пользователю так называемого «снимка» базы, обладающего тем свойством, что вносимые пользователем изменения невидимы другим пользователям до момента фиксации транзакции. Этот способ управления позволяет добиться того, что **пишущие транзакции не блокируют читающих, и читающие транзакции не блокируют пишущих**.

- Разные пользователи могут одновременно работать с одними и теми же данными;
- Каждый пользователь видит свой изолированный срез данных;
- Изменения, вносимые пользователем, никому не видны до завершения транзакции.

Защита данных встроенными средствами СУБД

- Представления
- Хранимые процедуры и функции
- Триггеры

Представления

Представление — виртуальная (логическая) таблица, представляющая собой именованный запрос, который будет подставлен как подзапрос при использовании представления.

Применение представлений позволяет разработчику базы данных создать интерфейс приложения, не зависящий от реально существующих таблиц, а также предоставить каждому пользователю или группе пользователей ограниченный набор данных, скрывая поля и записи с конфиденциальной информацией. Представления позволяют ограничить доступ к данным на уровне записей, дополняя, таким образом, дискреционную модель разграничения доступа.

Разрешение на доступ к представлению должно быть явно предоставлено или отозвано, независимо от прав доступа к базовым таблицам представления, тем самым реализуется принцип минимальных привилегий. Данные в базовой таблице, не включенные в представление, скрыты от пользователей, которые имеют права доступа к представлению, но не к базовой таблице.

Представления

Определяя различные представления и выборочно давая на них права доступа, пользователь (или любая комбинация пользователей) может быть ограничен различными подмножествами данных. С помощью представлений можно ограничить доступ к данным и выдавать только часть:

- Подмножество строк базовой таблицы (подмножество, зависящее от значений).
- Подмножество столбцов базовой таблицы (подмножество, не зависящее от значений).
- Подмножество строк и столбцов базовой таблицы.
- Подмножество столбцов, являющихся соединением более чем одной базовой таблицы.
- Статистическая сводка данных по базовой таблице.
- Подмножество данных другого представления или совокупности некоторых представлений и базовых таблиц.

Использование представлений

- **Повышение защищенности данных**. Возможность гибкой настройки прав доступа к данным за счет того, что права даются не на таблицу, а на представление. Это позволяет назначить права на отдельные строки таблицы или возможность получения не самих данных, а результата каких-то действий над ними.
- **Обеспечение целостности данных**. Если в операторе `CREATE VIEW` будет указана фраза `WITH CHECK OPTION`, то СУБД станет осуществлять контроль за тем, чтобы в исходные таблицы базы данных не была введена ни одна из строк, не удовлетворяющих предложению `WHERE` в определяющем запросе.
- **Разделение логики хранения данных и программного обеспечения**. Можно менять структуру данных, не затрагивая программный код. При изменении схемы БД достаточно создать представления, аналогичные таблицам, к которым раньше обращались приложения. Это удобно, когда нет возможности изменить программный код или к одной базе данных обращаются несколько приложений с различными требованиями к структуре данных.
- **Ограничение доступа к данным**. Можно предоставить пользователям упрощенную модель только необходимых ему данных.

- **Ограниченные возможности обновления**. В большинстве случаев представления не позволяют вносить изменения в содержащиеся в них данные.

Хранимые процедуры (функции)

Хранимые процедуры / функции – объекты базы данных, представляющие собой набор SQL-инструкций, который компилируется и хранится как самостоятельный исполняемый код в системном каталоге БД.

С точки зрения безопасности использование хранимых процедур позволяет определить интерфейс взаимодействия с данными строго определенным образом. Возможно разрешить доступ к данным только через процедуры и функции, не давая доступ непосредственно к таблицам или представлениям, с которыми эти процедуры взаимодействуют.

Когда пользователь вызывает процедуру, она выполняется с привилегиями владельца процедуры. Пользователи, обладающие только привилегиями на выполнение процедуры (но не привилегиями на запросы обновления или удаления непосредственно из таблиц), могут вызвать процедуру, но не могут манипулировать данными таблицы каким-либо другим способом.

Фактически все прямые операции выборки и модификации данных могут быть запрещены и единственным вариантом их выполнения могут быть соответствующие процедуры или функции.

Кроме того, если операции с данными требуют некоторой дополнительной логики, эти проверки также проще реализовать в коде хранимой процедуры, чем другими способами.

Использование хранимых процедур / функций

- **Повышение защищенности данных**. Доступ к данным большинству пользователей может быть предоставлен исключительно через ограниченный набор хранимых процедур, прямой доступ к таблицам предоставляется исключительно узкому кругу пользователей (администраторы). Это серьезно снижает риск возникновения инцидентов. Также хранимые процедуры могут выполнять дополнительную логику при выполнении операций над данными, связанную с проверкой нетривиальных ограничений или формированием аудиторского следа.
- **Повышение производительности**. Хранимые процедуры хранятся в скомпилированном и оптимизированном виде. Как следствие выполнение хранимой процедуры происходит быстрее, чем запуск аналогичного кода динамического SQL.
- **Снижение объема передаваемых данных**. Для вызова хранимой процедуры достаточно указать ее имя и значения параметров, а не передавать полный текст запроса.
- **Повторное использование кода**. Хранимые процедуры, выполняющие однотипные действия, могут переноситься между базами данных с незначительной модификацией.

Триггер – специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблице, с которой триггер связан.

Триггеры в основном предназначены для обеспечения целостности и непротиворечивости данных, а также для отката транзакций:

- триггер гарантированно срабатывает только при наступлении определенного события, обычно связанного с модификацией значений в строке таблицы;
- триггер проверяет условия выполнения операции, вызвавшей его срабатывание;
- если условия верны, то триггер выполняет определенные действия (например, разрешает добавить в таблицу новую строку), а если условия ложны – триггер отвергает операцию.

Каждый триггер привязывается к конкретной таблице и выполняется в составе соответствующей транзакции модификации данных. В случае обнаружения ошибки или нарушения целостности данных в коде триггера можно произвести откат транзакции. Тем самым внесение изменений (событие, вызвавшее триггер) отменяется. Отменяются также все изменения, уже сделанные триггером. В отличие от обычной процедуры, триггер выполняется неявно в каждом случае возникновения триггерного события.

Триггеры

С точки зрения безопасности данных триггеры в первую очередь стоит рассматривать как дополнительный инструмент обеспечения целостности данных. Их используют, когда ограничения целостности и значений по умолчанию не позволяют добиться нужного уровня функциональности. Часто требуется реализовать сложные алгоритмы проверки данных, отслеживать изменения значений таблицы, чтобы нужным образом изменить связанные данные.

Кроме того, триггеры могут использоваться для формирования аудиторского следа, например, при операциях модификации данных записывать в таблицу аудита пользователя, который выполнил изменения, и время изменения.

В ситуациях, когда представления не являются изменяемыми, триггеры для этих представлений могут использоваться для реализации модификации данных.

Использование триггеров

- **Скрытая функциональность**: перенос части функций в базу данных и сохранение их в виде одного или нескольких триггеров иногда приводит к сокрытию от пользователя некоторых функциональных возможностей. Хотя это в определенной степени упрощает работу, но может стать причиной незапланированных, потенциально нежелательных и вредных побочных эффектов, поскольку в этом случае пользователь не в состоянии контролировать все процессы, происходящие в базе данных.
- **Влияние на производительность**: перед выполнением каждой команды по изменению состояния базы данных СУБД должна проверить условие триггера с целью выяснения необходимости запуска триггера для этой команды. Выполнение подобных вычислений сказывается на общей производительности СУБД, а в моменты пиковой нагрузки ее снижение может стать особенно заметным.
- Неправильно написанные триггеры могут привести к серьезным проблемам, таким, например, как появление "мертвых" блокировок. Триггеры способны длительное время блокировать множество ресурсов, поэтому следует обратить особое внимание на сведение к минимуму конфликтов доступа.

Заключение. Рассмотренные темы

- Управления доступом к данным
  - Учетные записи
  - Аутентификация
  - Управления привилегиями
- Доступность данных
  - Резервное копирование
  - Репликация и балансировка нагрузки
  - Секционирование и сегментирование
  - Мониторинг
- Аудит
- Шифрование
- SQL-инъекции
- Целостность данных

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
д.т.н., профессор кафедры ГИиИБ