Основы построения защищенных баз данных

Ваша команда по спасению компьютерной безопасности

31 мая 2020 г.

Самый надежный в мире алгоритм
Всегда делать исключительно то, что горит
В самый прекрасный последний момент
Ведь для самых важных дел в принципе лучше времени нет

Anacondaz - Факап

Содержание

1 Механизмы обеспечения целостности СУБД

1.1 Угрозы целостности СУБД

Задача обеспечения целостности предусматривает комплекс мер по предотвращению непреднамеренного изменения или уничтожения информации, используемой информационной системой управления или системой поддержки принятия решений. Изменение или уничтожение данных может быть следствием неблагоприятного стечения обстоятельств и состояния внешней среды (стихийные бедствия, пожары и т. п.), неадекватных действий пользователей (ошибки при вводе данных, ошибки операторов и т. п.) и проблем, возникающих при многопользовательской обработке данных (Лихоносов А. Г. 2011).

Например, с помощью SQL-операторов UPDATE, INSERT и DELETE можно изменить данные в СУБД. Опасность заключается в том, что пользователь, обладающий соответствующими привилегиями, может модифицировать все записи в таблице (Утебов Д. Р. 2008).

Основные виды и причины возникновения угроз целостности

Перечислим основные угрозы целостности информации (Пирогов 2009):

1. **Отказ пользователей**. Под пользователями в данном случае мы понимаем широкий круг персонала, работающего с системой: операторы, програм- мисты, администраторы и т. д.

   - Непреднамеренные ошибки. Непреднамеренная ошибка может вызвать непосредственно порчу данных или средств доступа, либо создать условия для реализации другой угрозы, например вторжения злоумышленника.
   - Нежелание работать с информационной системой. Причиной нежелания может, например, быть необходимость освоения новых возможностей системы или несоответствие системы запросам пользователей.
   - Невозможность работать с системой. Причиной невозможности работать с системой может быть как отсутствие соответствующей подготовки персонала, так и отсутствие необходимой документации по системе.

2. **Внутренние отказы информационной системы**. Основными источниками внутренних отказов могут быть:

   - случайное или умышленное отступление от правил эксплуатации. Например, правила могут предусматривать определенный набор параметров сервера (объем памяти, производительность процессора, объем дискового пространства, версия операционной системы и т. п.), на котором предполагается использовать ИС;
   - выход системы из штатного режима эксплуатации в силу случайных или преднамеренных действий пользователей;
   - ошибки конфигурирования системы. В сложных системах конфигурирование выполняется при установке и настройке системы. При неправильной настройке могут возникнуть проблемы в эксплуатации;
   - отказ программного обеспечения. Программное обеспечение может содержать ошибки, в том числе и такие, которые могут привести к серьезным повреждениям данных. Кроме этого преднамеренно может быть изменен алгоритм программы. Таким образом, следует защищать программное обеспечение информационной системы и от случайного повреждения, и от исправления непосредственно исполняемых модулей;
   - разрушение данных (возможно, преднамеренное). На заре компьютерной революции, когда не слишком заботились о безопасности данных, часто сталкивались с ситуацией, когда не совсем компетентный пользователь просто стирал важные (но плохо защищенные) данные с диска.

3. **Внешние источники возможного нарушения доступа к данным**. Дан- ные источники могут быть вызваны и злонамеренными действиями людей, и стихийными бедствиями или авариями.

   - Отказ, повреждение или разрушение аппаратных средств (носителей информации, компьютеров, каналов связи). При интенсивном использовании отказ аппаратных средств случается совсем не редко, и меры безопасности должны учитывать такую возможность.

- Нарушение условий работы (системы связи, электропитание, отопление и т. п.). Отключение электричества совсем недавно было серьезной проблемой функционирования компьютерных систем. Источники бесперебойного питания должны защищать не только сами компьютеры, но все устройства в сети.
- Разрушение или повреждение помещений. Конечно, такая ситуация на первый взгляд кажется мало возможной, но это вполне вероятно в регионах, например, с сейсмической неустойчивостью.
- Невозможность или отказ обслуживающего персонала выполнять свои обязанности (стихийные бедствия, волнения, забастовка и т. п.). Отказ персонала выполнять свои обязанности для некоторых систем может привести к катастрофическим последствиям.
- Сетевые атаки, вирусные программы и другое вредоносное программное обеспечение. Сетевые атаки последнее время стали сильнейшим фактором риска информационных систем, работающих в Интернете.
- Разрушение информации намеренными действиями человека (диверсия). В данном случае речь идет о действиях людей, не являющихся обслуживающим персоналом данной системы.

Способы противодействия

Основными средствами защиты целостности информации в ИС являются (Пирогов 2009):

- транзакционные механизмы, позволяющие восстановить целостность дан- ных в случае незначительных сбоев;
- контроль ввода данных. Много ошибок можно было бы избежать, если программы не пропускали бы заведомо противоречивые данные;
- использование средств защиты целостности СУБД;
- резервное копирование данных;
- периодическое тестирование системы на предмет нарушения целостности.

1.2 Метаданные и словарь данных

Метаданные – Это данные, описывающие другие данные. Это важный элемент хранилища данных, который предоставляет возможность показывать пользователю предметно-ориентированную структуру, а не набор абстрактно-связанных таблиц. Метаданные предназначены для хранения информации о происхождении данных, о любых изменениях данных, о расположении данных, об ограничениях на данные, о соответствии данных тем или иным объектам предметной области и т. д. (Пирогов 2009)

Назначение словаря данных

Согласно канонам реляционной модели данных описание структуры базы данных (описание объектов, входящих в базу данных) также должно храниться в таблицах. Такая структура называется словарем данных или системным каталогом. По сути, такие данные следует назвать метаданными, т. е. данными, описывающими структуру других данных. В 1992 году в стандарте языка SQL было дано описание такого системного каталога. Но к этому времени в различных СУБД были приняты свои принципы хранения метаданных, отказаться от которых не так-то просто. В результате в одних СУБД вообще отсутствуют стандартные подходы к хранению метаданных, в других эти подходы в значительной степени дополнены своими особенностями. (Пирогов 2009)

Доступ к словарю данных

По 4 правилу Кодда (Dynamic On-Line Catalog Based on the Relational Model) словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств.

Для доступа к словарю данных используются инструкции SQL. Так как словарь данных доступен только для чтения, допускается выполнять только запросы таблиц и представлений. Можно запрашивать представления словаря, которые основаны на таблицах словаря, чтобы найти сведения, такие как (Sql-oracle б.г.):

- определения всех объектов схемы в словаре (таблицы, представления, индексы, синонимы, последовательности, процедуры, функции, пакеты, триггеры и так далее);
- значения по умолчанию для столбцов;
- сведения об ограничениях целостности;
- имена пользователей Oracle;
- привилегии и роли, предоставленные каждому пользователю;
- другие общие сведения о базе данных.

Состав словаря

В состав словаря данных базы данных входят *базовые таблицы* и *доступные пользователю представления*. Основу словаря данных составляет совокупность базовых таблиц, хранящих информацию о базе данных. Эти таблицы читаются и пишутся на системном уровне; они редко используются непосредственно пользователями. Словарь данных содержит доступные пользователю представления, которые суммируют и отображают в удобном формате информацию из базовых таблиц словаря. Эти представления декодируют информацию базовых таблиц, представляя ее в полезном виде, таком как имена пользователей или таблиц, и используют соединения и фразы WHERE, чтобы упростить информацию. Большинство пользователей имеют доступ к этим представлениям вместо базовых таблиц словаря. Словарь данных базы данных Oracle имеет два основных применения (Кирилов 2009):

- Oracle обращается к словарю данных каждый раз, когда выдается предложение DDL;
- каждый пользователь Oracle может использовать словарь данных как только читаемый справочник по базе данных.

Словарь данных всегда доступен при открытой базе данных. Он размещается в табличном пространстве SYSTEM, которое всегда находится в состоянии online, когда база данных открыта.

Представления словаря

Словарь данных состоит из нескольких наборов представлений. Во многих случаях такой набор состоит из трех представлений, содержащих аналогичную информацию и отличающихся друг от друга своими префиксами (Кирилов 2009):

- USER — представление для пользователя;
- ALL — расширенное представление для пользователя;
- DBA — представление администратора.

Столбцы в каждом представлении набора идентичны, но имеются исключения. В представлениях с префиксом USER обычно нет столбца с именем OWNER (владелец); в представлениях USER под владельцем подразумевается пользователь, выдавший запрос. Некоторые представления DBA имеют дополнительные столбцы, которые содержат информацию, полезную для АБД.

Представления с префиксом USER:

- отражают окружение пользователя в базе данных, включая информацию о созданных им объектах, предоставленных им грантах и т. д.;
- выдают только строки, имеющие отношение к пользователю;
- имеют столбцы, идентичные с другими представлениями, с тем исключением, что столбец OWNER подразумевается (текущий пользователь);
- возвращают подмножество информации, предоставляемой представлениями ALL;
- могут иметь сокращенные общие синонимы для удобства.

Представления с префиксом ALL отражают общее представление о базе данных со стороны пользователя. Эти представления возвращают информацию об объектах, к которым пользователь имеет доступ через общие или явные гранты, помимо тех объектов, которыми владеет этот пользователь.

Представления с префиксом DBA показывают общее представление о базе данных, и предназначены для администраторов базы данных. Во время своей работы Oracle поддерживает набор "виртуальных"таблиц, в которых регистрируется текущая информация о базе данных. Эти таблицы называются динамическими таблицами производительности. Так как динамические таблицы

производительности не являются истинными таблицами, большинство пользователей не должно обращаться к ним. Динамические таблицы производительности принадлежат схеме SYS, а их имена начинаются с V_$. По этим таблицам создаются представления, а для представлений создаются синонимы, имена которых начинаются с V$.

1.3 Понятие транзакции

> Последовательность действий над данными, обрабатываемая СУБД как единая операция, будем называть **транзакцией**.

Из определения ясно, что транзакция может состоять из одной или нескольких команд языка SQL. При этом команды языка SQL могут перемежаться командами, не выполняющими непосредственно операций над данными (не читающими данные, не изменяющими данные, не изменяющими структуру данных). Таким образом, в качестве транзакции может быть выбрана любая часть программы, содержащая команды чтения или записи данных. В частности, транзакция может состоять всего из одной команды, например INSERT, или из сотен команд, изменяющих или не изменяющих данные. Понятие транзакции чрезвычайно емкое. Оно в действительности тесно связано с концептуальной моделью данных и всей информационной системы. Дело в том, что на уровне пользователя операции над данными носят предметный характер: начислить зарплату, уволить работника, перевести деньги на другой счет и т. п. Для выполнения таких операций часто необходимо выполнить десятки, и даже сотни команд языка SQL. Однако вся эта последовательность команд должна быть подчинена одной цели: операция уровня пользователя должна быть обязательно выполнена. Что будет, если при выполнении такой цепочки команд произойдет сбой? Как рассматривать тогда выполняемую операцию — как частично выполненную? Но как в дальнейшем система узнает, что операция была только частично выполнена, и как ее закончить? Все эти вопросы, в конце концов, и приводят к понятию **"транзакция"** (Пирогов 2009).

Фиксация транзакции

Согласно требованиям ACID транзакция должна быть устойчивой. После своего завершения она сохраняется в системе, которую ничто не может вернуть в исходное (до начала транзакции). состояние, т. е. происходит фиксация транзакции, означающая, что ее действие постоянно даже при сбое системы. При этом подразумевается некая форма хранения информации в постоянной памяти как часть транзакции. При выполнении отдельных операций транзакции могут быть нарушены какие-либо требования целостности данных (в первую очередь имеются в виду корпоративные правила целостности, см. главу 2). Важно только то, что по окончании выполнения транзакции (фиксация транзакции) все правила целостности базы данных должны быть соблюдены. Согласно стандарту транзакция начинается с первой команды, которая обращается к данным. После этого транзакция продолжается до тех пор, пока не будет выполнена команда COMMIT WORK (или просто COMMIT) либо не будет закрыто соединение к базе данных. При выполнении команды COMMIT WORK происходит фиксация транзакции, другими словами, после этой команды откат транзакции уже будет невозможен (Пирогов 2009).

Прокрутки вперед и назад

В результате сбоя СУБД могут возникнуть две потенциальные ситуации (Карпова 2009):

- Блоки, содержащие подтверждённые модификации, не были записаны в файлы данных, так что эти изменения отражены лишь в журнале транзакций. Следовательно, журнал транзакций содержит подтверждённые данные, которые должны быть переписаны в файлы данных.
- Журнал транзакций и блоки данных содержат изменения, которые не были подтверждены. Изменения, внесенные неподтверждёнными транзакциями, во время восстановления БД должны быть удалены из файлов данных.

Для того чтобы разрешить эти ситуации, СУБД автоматически выполняет два этапа при восстановлении после сбоев: прокрутку вперед и прокрутку назад.

1. Прокрутка вперед заключается в применении к файлам данных всех изме-нений, зарегистрированных в журнале транзакций. После прокрутки вперед файлы данных содержат все как подтверждённые, так и неподтверждённые изменения, которые были зарегистрированы в журнале транзакций.
2. Прокрутка назад заключается в отмене всех изменений, которые не были подтверждены. Для этого используются журнал транзакций и сегменты отката, информация из которых позволяет определить и отменить те транзакции, которые не были подтверждены, хотя и попали на диск в файлы БД.

После выполнения этих этапов восстановления БД находится в согласованном состоянии и с ней можно работать.

Контрольная точка

Критическим моментом в отказе системы является потеря содержимого основной (оперативной) памяти (в частности, буферов базы данных). Поскольку точное состояние любой выполнявшейся в момент отказа системы транзакции остается неизвестным, такая транзакция не может быть успешно завершена. Поэтому при перезапуске системы любая такая транзакция будет отменена (т.е. будет выполнен ее откат). Более того, при перезапуске системы, возможно, потребуется повторно выполнить некоторые транзакции, которые были успешно завершены до аварийного останова, но выполненные ими обновления еще не были перенесены из буферов оперативной памяти в физическую базу данных во вторичной памяти. Возникает очевидный вопрос: как система определяет в процессе перезапуска, какую транзакцию следует отменить, а какую выполнить повторно? Ответ заключается в том, что система автоматически создает контрольные точки с некоторым наперед заданным интервалом (обычно, когда в журнале накапливается определенное число записей). Для создания контрольной точки требуется, во-первых, выполнить принудительное сохранение содержимого буферов оперативной памяти в физической базе данных, и, во-вторых, осуществить принудительное сохранение специальной записи контрольной точки в журнале на физическом носителе. Запись контрольной точки содержит список всех транзакций, выполняемых в тот момент, когда создавалась контрольная точка (Дейт 2005).

Откат

Для управления транзакциями в системах, поддерживающих механизм транзакций и язык SQL, используется оператор отката транзакции (отмены изменений): ROLLBACK [WORK]. Для фиксации или отката транзакции система создаёт неявные точки фиксации и отката. По команде rollback

система откатит транзакцию на начало (на неявную точку отката). Для обеспечения целостности транзакции СУБД может откладывать запись изменений в БД до момента успешного выполнения всех операций, входящих в транзакцию, и получения команды подтверждения транзакции (commit). Но чаще используется другой подход: система записывает изменения в БД, не дожидаясь завершения транзакции, а старые значения данных сохраняет на время выполнения транзакции в сегментах отката. Сегмент отката (rollback segment, RBS) – это специальная область памяти на диске, в которую записывается информация обо всех текущих (незавершённых) изменениях. Обычно записывается "старое"и "новое"содержимое изменённых записей, чтобы можно было восстановить прежнее состояние БД при откате транзакции (по команде rollback) или при откате текущей операции (в случае возникновения ошибки). Данные в RBS хранятся до тех пор, пока транзакция, изменяющая эти данные, не будет завершена. Потом они могут быть перезаписаны данными более поздних транзакций. Команда savepoint запоминает промежуточную "текущую копию"состояния базы данных для того, чтобы при необходимости можно было вернуться к состоянию БД в точке сохранения: откатить работу от текущего момента до точки сохранения (rollback to <имя_точки>) или зафиксировать работу от начала транзакции до точки сохранения (commit to <имя_точки>). На одну транзакцию может быть несколько точек сохранения (ограничение на их количество зависит от СУБД). Для сохранения сведений о транзакциях СУБД ведёт журнал транзакций. Журнал транзакций — это часть БД, в которую поступают данные обо всех изменениях всех объектов БД. Журнал недоступен пользователям СУБД и поддерживается особо тщательно (иногда ведутся две копии журнала, хранимые на разных физических носителях). Форма записи в журнал изменений зависит от СУБД. Но обычно там фиксируется следующее:

- номер транзакции (номера присваиваются автоматически по возрастанию);
- состояние транзакции (завершена фиксацией или откатом, не завершена, находится в состоянии ожидания);
- точки сохранения (явные и неявные);
- команды, составляющие транзакцию, и проч.

Начало транзакции соответствует появлению первого исполняемого SQLоператора. При этом в журнале появляется запись об этой транзакции (Карпова 2009).

Транзакции как средство изолированности пользователей

Поддержание механизма транзакций —- показатель уровня развитости СУБД и основа обеспечения целостности базы данных. Транзакции также составляют основу изолированности в многопользовательских системах, где с одной базой данных параллельно могут работать несколько пользователей и (или) прикладных программ. Одна из основных задач СУБД —- обеспечение изолированности, т. е. создание такого режима функционирования, при котором каждому пользователю казалось бы, что база данных доступна только ему. Такую задачу СУБД принято называть параллелизмом транзакций. Большинство выполняемых действий производится в теле транзакций. По умолчанию каждая команда выполняется как самостоятельная транзакция. Как было показано ранее, при необходимости пользователь может явно указать ее начало и конец, чтобы иметь возможность включить в нее несколько команд. Решение проблемы параллельной обработки базы данных

заключается в том, что строки таблиц блокируются, а последующие транзакции, модифицирующие эти строки, отвергаются и переводятся в режим ожидания. В связи со свойством сохранения целостности базы данных транзакции являются подходящими единицами изолированности пользователей. Действительно, если каждый сеанс взаимодействия с базой данных реализуется транзакцией, то пользователь начинает с того, что обращается к согласованному состоянию базы данных —- состоянию, в котором она могла бы находиться, даже если бы пользователь работал с ней в одиночку. Если бы в СУБД не были реализованы механизмы блокирования, то при одновременном чтении и изменении одних и тех же данных несколькими пользователями могли бы возникнуть следующие проблемы одновременного доступа:

- проблема последнего изменения возникает, когда несколько пользователей изменяют одну и ту же строку, основываясь на ее начальном значении; тогда часть данных будет потеряна, т. к. каждая последующая тран- закция перезапишет изменения, сделанные предыдущей. Выход из этой ситуации заключается в последовательном внесении изменений;

- проблема "грязного"чтения возможна в том случае, если пользователь выполняет сложные операции обработки данных, требующие множественного изменения данных перед тем, как они обретут логически верное состояние. Если во время изменения данных другой пользователь будет считывать их, то может оказаться, что он получит логически неверную информацию. Для исключения подобных проблем необходимо производить считывание данных после окончания всех изменений;

- проблема неповторяемого чтения является следствием неоднократного считывания транзакцией одних и тех же данных. Во время выполнения первой транзакции другая может внести в данные изменения, поэтому при повторном чтении первая транзакция получит уже иной набор данных, что приведет к нарушению их целостности или логической несогласованности;

- проблема чтения фантомов появляется после того, как одна транзакция выбирает данные из таблицы, а другая вставляет или удаляет строки до завершения первой. Выбранные из таблицы значения будут некорректны.

Сериализация транзакций

Для того, чтобы добиться изолированности транзакций, в СУБД должны использоваться какие-либо методы регулирования совместного выполнения транзакций.

План (способ) выполнения набора транзакций называется сериальным, если результат совместного выполнения транзакций эквивалентен результату некоторого последовательного выполнения этих же транзакций.

Сериализация транзакций – это механизм их выполнения по некоторому сериальному плану. Обеспечение такого механизма является основной функцией компонента СУБД, ответственного за управление транзакциями. Система, в которой поддерживается сериализация транзакций, обеспечивает реальную изолированность пользователей.

Основная реализационная проблема состоит в выборе метода сериализации набора транзакций, который не слишком ограничивал бы их параллельность. Приходящим на ум тривиальным

решением является действительно последовательное выполнение транзакций. Но существуют ситуации, в которых можно выполнять операторы разных транзакций в любом порядке с сохранением сериальности. Примерами могут служить только читающие транзакции, а также транзакции, не конфликтующие по доступу к объектам базы данных.

Между транзакциями могут существовать следующие виды конфликтов:

- W-W - транзакция 2 пытается изменять объект, измененный не закончившейся транзакцией 1;
- R-W - транзакция 2 пытается изменять объект, прочитанный не закончившейся транзакцией 1;
- W-R - транзакция 2 пытается читать объект, измененный не закончившейся транзакцией 1.

Практические методы сериализации транзакций основываются на учете этих конфликтов (Citforum 2020b).

Методы сериализации транзакций

Существуют два базовых подхода к сериализации транзакций – основанный на синхронизационных захватах объектов базы данных и на использовании временных меток. Суть обоих подходов состоит в обнаружении конфликтов транзакций и их устранении. Предварительно заметим, что для каждого из подходов имеются две разновидности – пессимистическая и оптимистическая. При применении пессимистических методов, ориентированных на ситуации, когда конфликты возникают часто, конфликты распознаются и разрешаются немедленно при их возникновении. Оптимистические методы основываются на том, что результаты всех операций модификации базы данных сохраняются в рабочей памяти транзакций. Реальная модификация базы данных производится только на стадии фиксации транзакции. Тогда же проверяется, не возникают ли конфликты с другими транзакциями.

Пессимистические методы сравнительно просто трансформируются в свои оптимистические варианты (Citforum 2020b).

1.4 Блокировки

Блокировки, называемые также синхронизационными захватами объектов, могут быть применены к разному типу объектов. Наибольшим объектом блокировки может быть вся БД, однако этот вид блокировки сделает БД недоступной для всех приложений, которые работают с данной БД. Следующий тип объекта блокировки —- это таблицы. Транзакция, которая работает с таблицей, блокирует ее на все время выполнения транзакции. Этот вид блокировки предпочтительнее предыдущего, потому что позволяет параллельно выполнять транзакции, которые работают с другими таблицами.

В ряде СУБД реализована блокировка на уровне страниц. В этом случае СУБД блокирует только отдельные страницы на диске, когда транзакция обращается к ним. Этот вид блокировки

еще более мягок и позволяет разным транзакциям работать даже с одной и той же таблицей, если они обращаются к разным страницам данных.

В некоторых СУБД возможна блокировка на уровне строк, однако такой механизм блокировки требует дополнительных затрат на поддержку этого вида блокировки.

В настоящее время проблема блокировок является предметом большого числа исследований (Intuit 2020a).

Режимы блокировок

Рассматривают два типа блокировок (синхронизационных захватов) (Intuit 2020a):

- совместный режим блокировки — нежесткая, или разделяемая, блокировка, обозначаемая как S (Shared). Этот режим обозначает разделяемый захват объекта и требуется для выполнения операции чтения объекта. Объекты, заблокированные таким образом, не изменяются в ходе выполнения транзакции и доступны другим транзакциям также, но только в режиме чтения;
- монопольный режим блокировки — жесткая, или эксклюзивная, блокировка, обозначаемая как X (eXclusive). Данный режим блокировки предполагает монопольный захват объекта и требуется для выполнения операций занесения, удаления и модификации. Объекты, заблокированные данным типом блокировки, фактически остаются в монопольном режиме обработки и недоступны для других транзакций до момента окончания работы данной транзакции.

Правила согласования блокировок

Захваты объектов несколькими транзакциями по чтению совместимы, то есть нескольким транзакциям допускается читать один и тот же объект, захват объекта одной транзакцией по чтению не совместим с захватом другой транзакцией того же объекта по записи, и захваты одного объекта разными транзакциями по записи не совместимы. В примере, представленном на рис. 1 считается, что первой блокирует объект транзакция A, а потом пытается получить к нему доступ транзакция B.

		Транзакция B		
		Разблокирована	Нежесткая блокировка	Жесткая блокировка
Транзакция A	Разблокирована	Да	Да	Да
	Нежесткая блокировка	Да	Да	Нет
	Жесткая блокировка	Да	Нет	Нет

Рис. 1: Правила применения жесткой и нежесткой блокировок транзакций

Двухфазный протокол синхронизационных блокировок

В базах данных и обработке транзакций двухфазная блокировка (2PL) — это метод управления параллелизмом, который гарантирует сериализуемость. Это также имя результирующего набора графиков транзакций базы данных (истории). Протокол использует блокировки, применяемые транзакцией к данным, которые могут блокировать (интерпретировать как сигналы для остановки) другие транзакции от доступа к тем же данным в течение жизни транзакции.

По протоколу 2PL блокировки (locks) применяются и удаляются в два этапа:

1. Фаза расширения: блокировки берутся и ни одна блокировка не освобождается.
2. Фаза сокращения: блокировки освобождаются и ни одна блокировка не берётся.

В базовом протоколе используются два типа блокировок: Shared и Exclusive locks. Уточнения базового протокола могут использовать больше типов блокировок. Используя блокировки, блокирующие процессы, 2PL могут подвергаться взаимоблокировкам, которые являются результатом взаимной блокировки двух или более транзакций (Wikipedia 2020a).

Тупиковые ситуации, их распознавание и разрушение

Одним из наиболее чувствительных недостатков метода сериализации транзакций на основе синхронизационных захватов является возможность возникновение тупиков (deadlocks) между транзакциями.

Вот простой пример возникновения тупика между транзакциями T1 и T2:

1. транзакции T1 и T2 установили монопольные захваты объектов r1 и r2 соответственно;
2. после этого T1 требуется совместный захват r2, а T2 - совместный захват r1;
3. ни одна из транзакций не может продолжаться, следовательно, монопольные захваты не будут сняты, а совместные - не будут удовлетворены.

Поскольку тупики возможны, и никакого естественного выхода из тупиковой ситуации не существует, то эти ситуации необходимо обнаруживать и искусственно устранять.

Основой обнаружения тупиковых ситуаций является построение (или постоянное поддержание) графа ожидания транзакций. Граф ожидания транзакций - это ориентированный двудольный граф, в котором существует два типа вершин - вершины, соответствующие транзакциям, и вершины, соответствующие объектам захвата. В этом графе существует дуга, ведущая из вершины-транзакции к вершине-объекту, если для этой транзакции существует удовлетворенный захват объекта. В графе существует дуга из вершины-объекта к вершине-транзакции, если транзакция ожидает удовлетворения захвата объекта.

Легко показать, что в системе существует ситуация тупика, если в графе ожидания транзакций имеется хотя бы один цикл.

Для распознавание тупика периодически производится построение графа ожидания транзакций (как уже отмечалось, иногда граф ожидания поддерживается постоянно), и в этом графе ищутся циклы. Традиционной техникой (для которой существует множество разновидностей) нахождения циклов в ориентированном графе является редукция графа.

Не вдаваясь в детали, редукция состоит в том, что прежде всего из графа ожидания удаляются все дуги, исходящие из вершин-транзакций, в которые не входят дуги из вершин-объектов. (Это как

бы соответствует той ситуации, что транзакции, не ожидающие удовлетворения захватов, успешно завершились и освободили захваты). Для тех вершин-объектов, для которых не осталось входящих дуг, но существуют исходящие, ориентация исходящих дуг изменяется на противоположную (это моделирует удовлетворение захватов). После этого снова срабатывает первый шаг и так до тех пор, пока на первом шаге не прекратится удаление дуг. Если в графе остались дуги, то они обязательно образуют цикл.

Предположим, что нам удалось найти цикл в графе ожидания транзакций. Что делать теперь? Нужно каким-то образом обеспечить возможность продолжения работы хотя бы для части транзакций, попавших в тупик. Разрушение тупика начинается с выбора в цикле транзакций так называемой транзакции-жертвы, т.е. транзакции, которой решено пожертвовать, чтобы обеспечить возможность продолжения работы других транзакций.

Грубо говоря, критерием выбора является стоимость транзакции; жертвой выбирается самая дешевая транзакция. Стоимость транзакции определяется на основе многофакторная оценка, в которую с разными весами входят время выполнения, число накопленных захватов, приоритет.

После выбора транзакции-жертвы выполняется откат этой транзакции, который может носить полный или частичный характер. При этом, естественно, освобождаются захваты и может быть продолжено выполнение других транзакций.

Естественно, такое насильственное устранение тупиковых ситуаций является нарушением принципа изолированности пользователей, которого невозможно избежать.

Заметим, что в централизованных системах стоимость построения графа ожидания сравнительно невелика, но она становится слишком большой в по-настоящему распределенных СУБД, в которых транзакции могут выполняться в разных узлах сети. Поэтому в таких системах обычно используются другие методы сериализации транзакций.

Еще одно замечание. Чтобы минимизировать число конфликтов между транзакциями, в некоторых СУБД (например, в Oracle) используется следующее развитие подхода. Монопольный захват объекта блокирует только изменяющие транзакции. После выполнении операции модификации предыдущая версия объекта остается доступной для чтения в других транзакциях. Кратковременная блокировка чтения требуется только на период фиксации изменяющей транзакции, когда обновленные объекты становятся текущими (Citforum 2020b)

1.5 Ссылочная целостность

Ссылочная целостность. Ссылочная целостность относится непосредственно к связям между таблицами. Если кратко, то ссылочная целостность должна отвечать на вопрос: что будет со строками одной таблицы, если в связанной таблице выполняется какая-либо операция модификации? Для того чтобы понять логику ссылочной целостности, будем считать таблицу главной в паре связанных таблиц, если она содержит первичный ключ, с помощью которого осуществляется связь. Вторую таблицу будем считать подчиненной таблицей. Теперь выделим три вида операции над связанными таблицами: удаление из главной таблицы, обновление строк главной таблицы, вставка в подчиненную таблицу.

1. Удаление строк из главной таблицы. Если удаляемая строка не связана со строками другой таблицы, то удалять можно без всяких последствий. Но если удаляемая строка связана со строками другой таблицы, то надо подумать, что же будет с этими строками. Просто оста-

вить их без изменения нельзя, т. к. не понятно, как быть со значениями внешних ключей. В принципе, возможны следующие четыре сценария, поддерживаемые основными СУБД:

- строки из подчиненной таблицы должны быть удалены вместе со связанными строками из подчиненной таблицы. Такой механизм называется каскадированием;
- если удаляемая строка из главной таблицы связана со сроками из подчиненной таблицы, то такая операция удаления должна быть отвергнута. Данный механизм наиболее безопасен и предпочтителен при построении информационной системы;
- если строка в подчиненной таблице связана с удаляемой строкой в главной таблице, то внешнему ключу следует присвоить значение NULL;
- если строка в подчиненной таблице связана с удаляемой строкой в главной таблице, то внешнему ключу следует присвоить значение, принятое по умолчанию.

2. Обновление строк из главной таблицы. Если обновляемая строка не связана со строками другой таблицы или обновляются столбцы, не от- носящиеся к первичному ключу, то обновлять можно без всяких по- следствий. Но если обновляемая строка связана со строками другой таблицы и обновляется первичный ключ, то здесь, как и в предыдущем случае, возможны четыре сценария:

   - первичные ключи из главной таблицы обновляются вместе с внеш- ними ключами подчиненной таблицы. Как и в случае с подобной операцией удаления, этот механизм называется каскадированием;
   - если обновляемая строка связана с какой-либо строкой подчиненной таблицы, то операция обновления должна быть отвергнута;
   - если строка в подчиненной таблице связана с обновляемой строкой в главной таблице, то внешнему ключу следует присвоить значение NULL;
   - если строка в подчиненной таблице связана с обновляемой строкой в главной таблице, то внешнему ключу следует присвоить значение, принятое по умолчанию.

3. Вставка строк в подчиненную таблицу. Здесь возможны следующие механизмы.

   - Строка в подчиненной таблице вставляется вместе со строкой в главной таблице. Здесь важно иметь в виду, что в главной таблице для всех столбцов должны быть определены значения по умолча- нию. Последовательность добавления такая: вначале добавляется строка в главную таблицу и определяется значение первичного клю- ча. Затем добавляется строка в подчиненную таблицу, в которой значению внешнего ключа присваивается значение первичного клю- ча в главной таблице.
   - Строка в подчиненную таблицу добавляется только при условии, что соответствующая ей строка в главной таблице уже существует.
   - При добавлении строки в подчиненную таблицу внешнему ключу присваивается значение NULL.
   - При добавлении строки в подчиненную таблицу внешнему ключу присваивается значение по умолчанию.

В некоторых простых СУБД отсутствует возможность устанавливать связи между таблицами и, таким образом, поддерживать ссылочную целостность. В этом случае поддержание ссылочной целостности полностью ложится на плечи программиста. Другими словами, связь между таблицами должна быть реализована на уровне прикладного программного обеспечения.

Декларативная и процедурная ссылочные целостности
Различают два способа реализации ограничений целостности:

- Декларативная поддержка ограничений целостности.
- Процедурная поддержка ограничений целостности.

> **Декларативная поддержка ограничений целостности** заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности (если они имеются в СУБД) определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений (ASSERTION).

Например, следующий оператор создает таблицу PERSON и определяет для нее некоторые ограничения целостности:

```
CREATE TABLE PERSON
  (Pers_Id INTEGER PRIMARY KEY,
  Pers_Name CHAR(30) NOT NULL,
  Dept_Id REFERENCES DEPART(Dept_Id) ON UPDATE CASCADE ON DELETE CASCADE);
```

После выполнения оператора для таблицы PERSON будут объявлены следующие ограничения целостности:

- Поле Pers_Id образует потенциальный ключ отношения.
- Поле Pers_Name не может содержать null-значений.
- Поле Dept_Id является внешней ссылкой на родительскую таблицу DEPART, причем, при изменении или удалении строки в родительской таблице каскадно должны быть внесены соответствующие изменения в дочернюю таблицу.

> **Процедурная поддержка ограничений целостности** заключается в использовании триггеров и хранимых процедур.

Не все ограничения целостности можно реализовать декларативно. Примером такого ограничения может служить требование из примера 1, утверждающее, что поле Dept_Kol таблицы DEPART должно содержать количество сотрудников, реально числящихся в подразделении. Для реализации

этого ограничения необходимо создать триггер, запускающийся при вставке, модификации и удалении записей в таблице PERSON, который корректно изменяет значение поля Dept_Kol. Например, при вставке в таблицу PERSON новой строки, триггер увеличивает на единицу значение поля Dept_Kol, а при удалении строки - уменьшает. Заметим, что при модификации записей в таблице PERSON могут потребоваться даже более сложные действия. Действительно, модификация записи в таблице PERSON может заключаться в том, что мы переводим сотрудника из одного отдела в другой, меняя значение в поле Dept_Id. При этом необходимо в старом подразделении уменьшить количество сотрудников, а в новом - увеличить (Citforum 2020a).

Внешний ключ

> **Внешний ключ** – это ограничение целостности, в соответствии с которым множество значений внешнего ключа является подмножеством значений первичного или уникального ключа родительской таблицы (Карпова 2009).

Ограничение целостности по внешнему ключу проверяется в двух случаях (Карпова 2009):

- при добавлении записи в подчинённую таблицу СУБД проверяет, что в родительской таблице есть запись с таким же значением первичного ключа;
- при удалении записи из родительской таблицы СУБД проверяет, что в подчинённой таблице нет записей с таким же значением внешнего ключа.

Способы поддержания ссылочной целостности

СУБД имеют механизм автоматического поддержания ссылочной целостности. Любая операция, изменяющая данные в таблице, вызывает автоматическую проверку ссылочной целостности. При этом (Wikipedia 2020b):

- При операции добавления записи автоматически проверяется, ссылаются ли внешние ключи в этой записи на существующие записи в заявленных при описании связанных таблицах. Если выясняется, что операция приведёт к появлению некорректных ссылок, она не выполняется — система возвращает ошибку.
- При операции редактирования записи проверяется:
  - если изменяется её первичный ключ и на данную запись имеются ссылки, то операция редактирования завершается с ошибкой;
  - если изменяется какой-то из внешних ключей, хранящихся в этой записи, и после изменения внешний ключ будет ссылаться на несуществующую запись, то операция редактирования завершается с ошибкой.
- При операции удаления записи проверяется, нет ли на неё ссылок. Если ссылки имеются, то возможно три варианта дальнейших действий (фактически выполняемый зависит от СУБД и от выбора программиста, который он должен сделать при описании связи):
  - Запрет — удаление блокируется и возвращается ошибка.

– Каскадное удаление — в одной транзакции производится удаление данной записи и всех записей, ссылающихся на данную. Если на удаляемые записи также есть ссылки и настройки также требуют удаления, то каскадное удаление продолжается дальше. Таким образом, после удаления данной записи в базе не остаётся ни одной записи, прямо или косвенно ссылающейся на неё. Если хотя бы одну из ссылающихся записей удалить не получается (либо для неё настроен запрет, либо происходит какая-либо ещё ошибка), то все удаления запрещаются.

– Присвоение NULL — во все внешние ключи записей, ссылающихся на данную, записывается маркер NULL. Если хотя бы для одной из ссылающихся записей это невозможно (например, если поле внешнего ключа описано как NOT NULL), то удаление запрещается.

1.6 Правила(триггеры)

Триггеры являются одной из разновидностей хранимых процедур. Их исполнение происходит при выполнении для таблицы какого-либо оператора языка манипулирования данными (DML). Триггеры используются для проверки целостности данных, а также для отката транзакций.

Триггер представляет собой специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблицах, с которыми триггеры связаны. Каждый триггер привязывается к конкретной таблице. Все производимые им модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушения целостности данных происходит откат этой транзакции. Тем самым внесение изменений запрещается. Отменяются также все изменения, уже сделанные триггером.

Триггер представляет собой весьма полезное и в то же время опасное средство. Так, при неправильной логике его работы можно легко уничтожить целую базу данных, поэтому триггеры необходимо очень тщательно отлаживать.

В отличие от обычной подпрограммы, триггер выполняется неявно в каждом случае возникновения триггерного события, к тому же он не имеет аргументов. Приведение его в действие иногда называют запуском триггера (Intuit 2020b)

Цели использования правил

С помощью триггеров достигаются следующие цели (Intuit 2020b):

- проверка корректности введенных данных и выполнение сложных ограничений целостности данных, которые трудно, если вообще возможно, поддерживать с помощью ограничений целостности, установленных для таблицы;
- выдача предупреждений, напоминающих о необходимости выполнения некоторых действий при обновлении таблицы, реализованном определенным образом;
- накопление аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили;
- поддержка репликации.

Способы задания, моменты выполнения

Создает триггер только владелец базы данных. Это ограничение позволяет избежать случайного изменения структуры таблиц, способов связи с ними других объектов и т.п. Основной формат команды CREATE TRIGGER показан ниже:

```
<Определение_триггера>::=
  CREATE TRIGGER имя_триггера
  BEFORE | AFTER <триггерное_событие>
  ON <имя_таблицы>
  [REFERENCING
    <список_старых_или_новых_псевдонимов>]
  [FOR EACH { ROW | STATEMENT}]
  [WHEN(условие_триггера)]
  <тело_триггера>
```

Триггерные события состоят из вставки, удаления и обновления строк в таблице. В последнем случае для триггерного события можно указать конкретные имена столбцов таблицы.

Триггер – это процедура БД, которая привязана к конкретной таблице и вызывается автоматически при наступлении определённого события (добавления, удаления или модификации данных этой таблицы).

В отличие от обычной подпрограммы, триггер выполняется неявно в каждом случае возникновения триггерного события, к тому же он не имеет аргументов. Приведение его в действие иногда называют запуском триггера.

Время запуска триггера определяется с помощью ключевых слов BEFORE (триггер запускается до выполнения связанных с ним событий) или AFTER (после их выполнения).

Выполняемые триггером действия задаются для каждой строки (FOR EACH ROW), охваченной данным событием, или только один раз для каждого события (FOR EACH STATEMENT).

Обозначение <список_старых_или_новых_псевдонимов> относится к таким компонентам, как старая или новая строка (OLD / NEW) либо старая или новая таблица (OLD TABLE / NEW TABLE). Ясно, что старые значения не применимы для событий вставки, а новые – для событий удаления (Intuit 2020b).

1.7 События

Назначение механизма событий

Механизм событий в базе данных (database events) позволяет прикладным программам и серверу базы данных уведомлять другие программы о наступлении в базе данных определенного события и тем самым синхронизировать их работу (Jet Infosystems 1995).

Сигнализаторы событий. Типы уведомлений о происхождении события. Компоненты механизма событий

Операторы языка SQL, обеспечивающие уведомление, часто называют сигнализаторами событий в базе данных (database event alerters). Функции управления событиями целиком ложатся на сервер базы данных.

Механизм событий используется следующим образом. Вначале в базе данных для каждого события создается флажок, состояние которого будет оповещать прикладные программы о том, что некоторое событие имело место (оператор CREATE DBEVENT - СОЗДАТЬ СОБЫТИЕ). Далее во все прикладные программы, на ход выполнения которых может повлиять это событие, включается оператор REGISTER DBEVENT (ЗАРЕГИСТРИРОВАТЬ СОБЫТИЕ), который оповещает сервер базы данных, что данная программа заинтересована в получении сообщения о наступлении события. Теперь любая прикладная программа или процедура базы данных может вызвать событие оператором RAISE DBEVENT (ВЫЗВАТЬ СОБЫТИЕ). Как только событие произошло, каждая зарегистрированная программа может получить его, для чего должна запросить очередное сообщение из очереди событий (оператор GET DBEVENT - ПОЛУЧИТЬ СОБЫТИЕ) и запросить информацию о событии, в частности, его имя (оператор SQL INQUIRE_SQL) (Jet Infosystems 1995).

Список литературы

Jet Infosystems (1995). *Системы управления базами данных - коротко о главном*. URL: https://www.osp.ru/news/articles/1995/0402/13031414#part_4 (дата обр. 30.05.2020).

Дейт (2005). *Введение в системы баз данных*. Вильямс. ISBN: 5-8459-0788-8. URL: http://tc.kpi.ua/content/lib/vvedenie_v_sistemy_baz_dannyh_8izdanie.pdf.

Утебов Д. Р., Белов С. В. (2008). «Классификация угроз в системах управления базами данных». В: *Вестник Астраханского государственного технического университета* 1, с. 87—92. URL: https://cyberleninka.ru/article/n/klassifikatsiya-ugroz-v-sistemah-upravleniya-bazami-dannyh.

Карпова (2009). *Базы данных. Учебное пособие*. Питер. ISBN: 978-5-94157-770-5. URL: https://rucont.ru/file.ashx?guid=a46c217d-4dbd-496b-a229-2ac562197d70.

Кирилов (2009). *Введение в реляционные базы данных*. БХВ-Петербург. ISBN: 978-5-496-00546-3. URL: https://mipt.ru/dnbic/content/db.pdf.

Пирогов (2009). *Информационные системы и базы данных: организация и проектирование*. БХВ-Петербург. ISBN: 978-5-9775-0399-0. URL: https://litmy.ru/knigi/os_bd/107950-informacionnye-sistemy-i-bazy-dannyh-organizaciya-i-proektirovanie.html.

Лихоносов А. Г. (2011). *Интернет-курс по дисциплине Безопасность баз данных*. URL: http://www.e-biblio.ru/book/bib/01_informatika/b_baz_dan/sg.html (дата обр. 01.03.2020).

Citforum (2020a). *Двухфазная блокировка*. URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D1%85%D1%84%D0%B0%D0%B7%D0%BD%D0%B0%D1%8F_%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0 (дата обр. 30.05.2020).

— (2020b). *Транзакции и целостность баз данных*. URL: http://citforum.ru/database/dblearn/dblearn09.shtml (дата обр. 30.05.2020).

Intuit (2020a). *Лекция 11: Модели транзакций*. URL: https://www.intuit.ru/studies/professional_retraining/953/courses/297/lecture/7419?page=4 (дата обр. 30.05.2020).

Intuit (2020b). *Лекция 14: Триггеры: создание и применение*. URL: https://www.intuit.ru/studies/courses/5/5/lecture/148 (дата обр. 30.05.2020).

Wikipedia (2020a). *Двухфазная блокировка*. URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D1%85%D1%84%D0%B0%D0%B7%D0%BD%D0%B0%D1%8F_%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0 (дата обр. 30.05.2020).

— (2020b). *Ссылочная целостность*. URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D1%86%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C (дата обр. 30.05.2020).

Sql-oracle.