Организация многопользовательского доступа к данным

"Кто хочет работать – ищет средства, кто не хочет – причины".

С.П. Королёв, советский ученый и конструктор в области космонавтики

Управление параллельным доступом

Управление параллельным доступом – это процесс организации одновременного выполнения а базе данных различных операций доступа, гарантирующий предотвращение их влияния друг на друга.

Транзакция – это упорядоченная последовательность операторов обработки данных, которая переводит базу данных из одного согласованного состояния в другое.

Транзакция обладает следующими свойствами:

Логическая неделимость (**атомарность**, Atomicity) означает, что выполняются либо все операции (команды), входящие в транзакцию, либо ни одной.

Согласованность (Consistency): транзакция начинается на согласованном множестве данных и после её завершения множество данных согласовано.

Изолированность (Isolation), т.е. отсутствие влияния транзакций друг на друга.

Устойчивость (Durability): результаты завершённой транзакции не могут быть потеряны. Возврат БД в предыдущее состояние может быть достигнут только путём запуска компенсирующей транзакции.

Транзакции, удовлетворяющие этим свойствам, называют ACID-транзакциями (по первым буквам названий свойств).

Команды управления транзакциями

Для управления транзакциями в системах, поддерживающих механизм транзакций и язык SQL, используются следующие операторы:

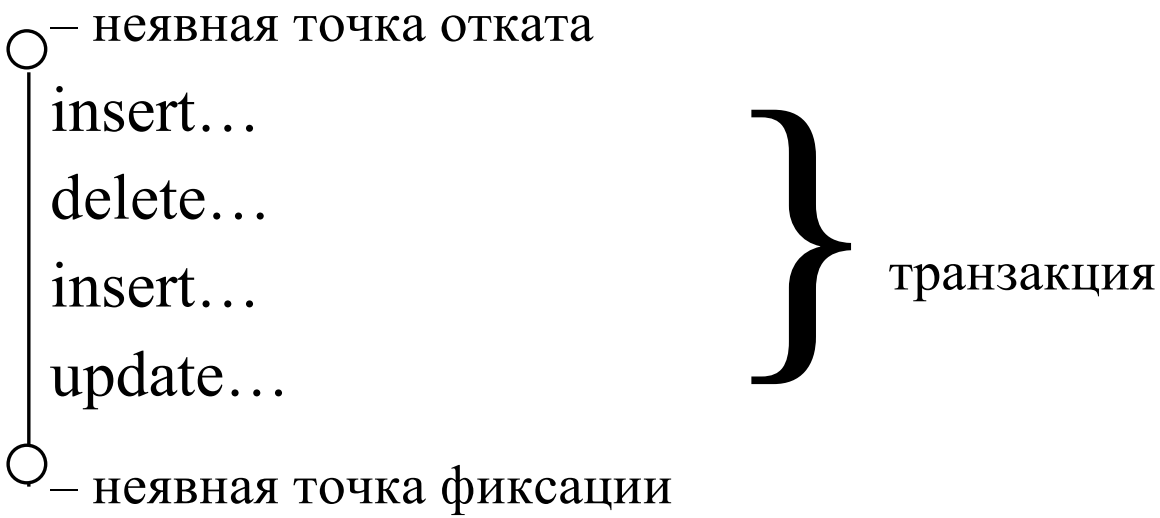
– **фиксация** транзакции (запоминание изменений):

COMMIT [WORK];

– **откат** транзакции (отмена изменений):

ROLLBACK [WORK];

– создание точки сохранения: **SAVEPOINT** <имя_точки_сохранения>;



Механизмы обеспечения работы транзакций

Сегмент отката (rollback segment, RBS) – это специальная область памяти на диске, в которую записывается информация обо всех текущих (незавершённых) изменениях. Данные в RBS хранятся до тех пор, пока транзакция, изменяющая эти данные, не будет завершена. Потом они могут быть перезаписаны данными более поздних транзакций.
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 и завершение транзакции

Начало транзакции соответствует появлению первого исполняемого SQL-оператора. При этом в журнале появляется запись об этой транзакции.

По стандарту ANSI/ISO транзакция завершается при наступлении одного из следующих событий:

- Поступила команда commit (результаты транзакции фиксируются).
- Поступила команда rollback (результаты транзакции откатываются).
- Успешно завершена программа (exit, quit), в рамках которой выполнялась транзакция. В этом случае транзакция фиксируется автоматически.
- Программа, выполняющая транзакцию, завершена аварийно (abort). При этом транзакция автоматически откатывается.

Примечания:

Возможна работа в режиме AUTOCOMMIT, когда каждая команда воспринимается системой как транзакция. В этом режиме пользователи меньше задерживают друг друга, требуется меньше памяти для сегмента отката, зато результаты ошибочно выполненной операции нельзя отменить командой rollback.

В некоторых СУБД реализованы расширенные модели транзакций, в которых существуют дополнительные ситуации фиксации транзакций. Например, в СУБД Oracle команды DDL выполняются в режиме AUTOCOMMIT, т.е. не могут быть откачены.

Запись изменений на диск

Все изменения данных выполняются в оперативной памяти в буфере данных, затем фиксируются в журнале транзакций и в сегменте отката и периодически (при выполнении контрольной точки) переписываются на диск.

Процесс формирования **контрольной точки** (КТ) заключается в синхронизации данных, находящихся на диске (т.е. во вторичной памяти) с теми данными, которые находятся в ОП: все модифицированные данные из ОП переписываются во вторичную память. В разных системах процесс формирования контрольной точки запускается по-разному.

Например, в СУБД Oracle КТ формируется:

- при поступлении команды commit,
- при переполнении буфера данных,
- в момент заполнения очередного файла журнала транзакций,
- через три секунды со времени последней записи на диск.

Внесение изменений в журнал транзакций всегда носит опережающий характер по отношению к записи изменений в основную часть БД (протокол WAL – Write Ahead Log). Эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала транзакций раньше, чем изменённый объект попадёт во внешнюю память основной части БД.

Действия при завершении транзакции

При использовании протокола WAL изменённые данные почти сразу попадают в базу данных, ещё по поступления команды commit.
Поэтому фиксация транзакции чаще всего заключается в следующем:

1. Изменения, внесённые транзакцией, помечаются как постоянные.
2. Уничтожаются все точки сохранения для данной транзакции.
3. Если выполнение транзакций осуществляется с помощью блокировок, то освобождаются объекты, заблокированные транзакцией.
4. В журнале транзакций транзакция помечается как завершённая, уничтожаются системные записи о транзакции в оперативной памяти.

А при откате транзакции вместо п.1 обычно выполняется считывание из сегмента отката прежних значений данных и переписывание их обратно в БД (остальные пункты сохранятся без изменений). Поэтому откат транзакции практически всегда занимает больше времени, чем фиксация.

Взаимовлияние транзакций

Транзакции в многопользовательской БД должны быть изолированы друг от друга, т.е. в идеале каждая из них должна выполняться так, как будто выполняется только она одна. В реальности транзакции выполняются одновременно и могут влиять на результаты друг друга, если они обращаются к одному и тому же набору данных и хотя бы одна из транзакций изменяет данные.

Транзакции не влияют друг на друга, если:
1. они только читают данные;
2. они обращаются к разным данным.

В общем случае взаимовлияние транзакций может проявляться в виде:
1. потери изменений;
2. чернового чтения;
3. неповторяемого чтения;
4. фантомов.

Потеря изменений

Представим, что одновременно начали выполняться две транзакции:

транзакция 1 – UPDATE *СОТРУДНИКИ*
SET *Оклад* = 39200
WHERE ***Номер*** = 1123;

транзакция 2 – UPDATE *СОТРУДНИКИ*
SET *Должность* = "старший экономист"
WHERE ***Номер*** = 1123;

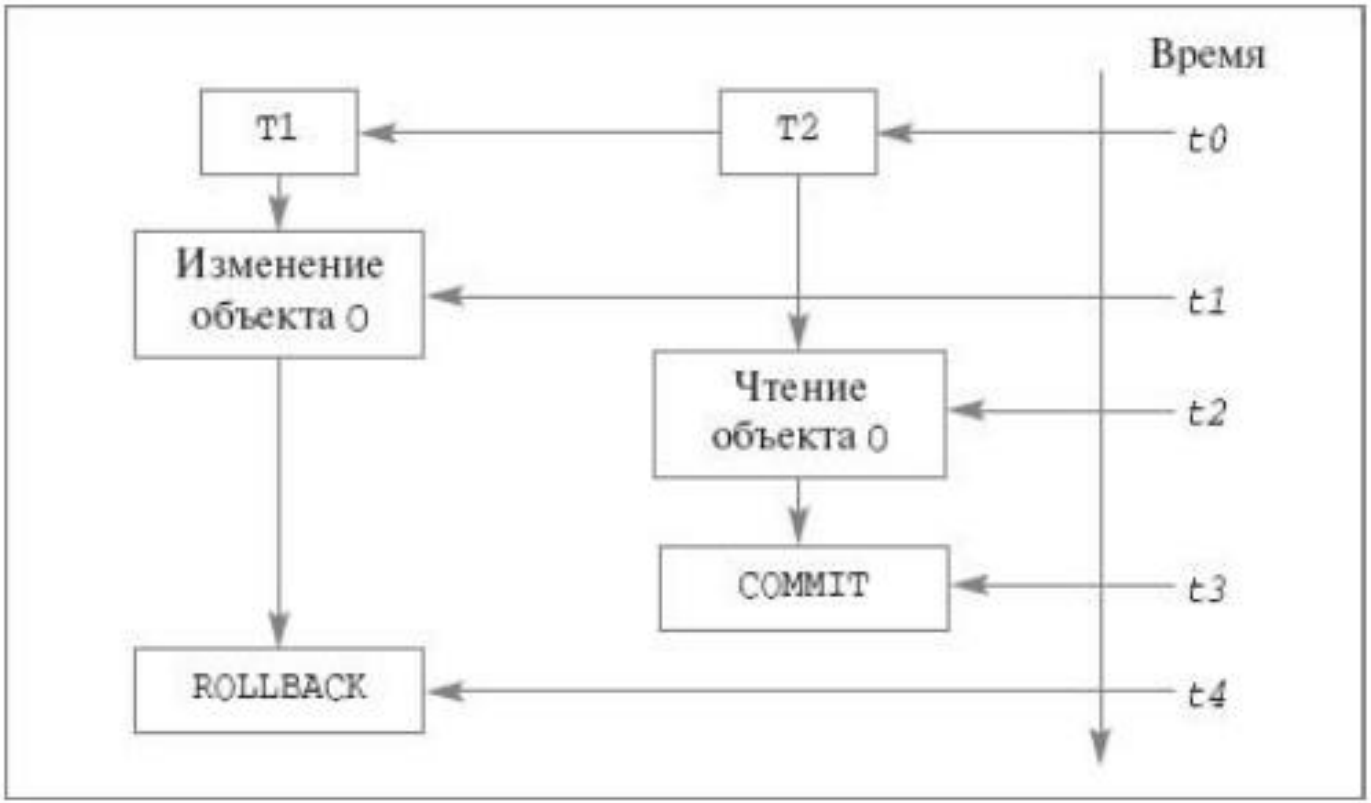
Отношение "Сотрудники"

Номер	ФИО	Должность	Оклад	
1123	Рудин В.П.	экономист	28300	
1123	Рудин В.П.	экономист	**39200**	Транзакция 1
1123	Рудин В.П.	**старший экономист**	28300	Транзакция 2

СУБД не допускает такого взаимовлияния транзакций, при котором возможна потеря изменений.

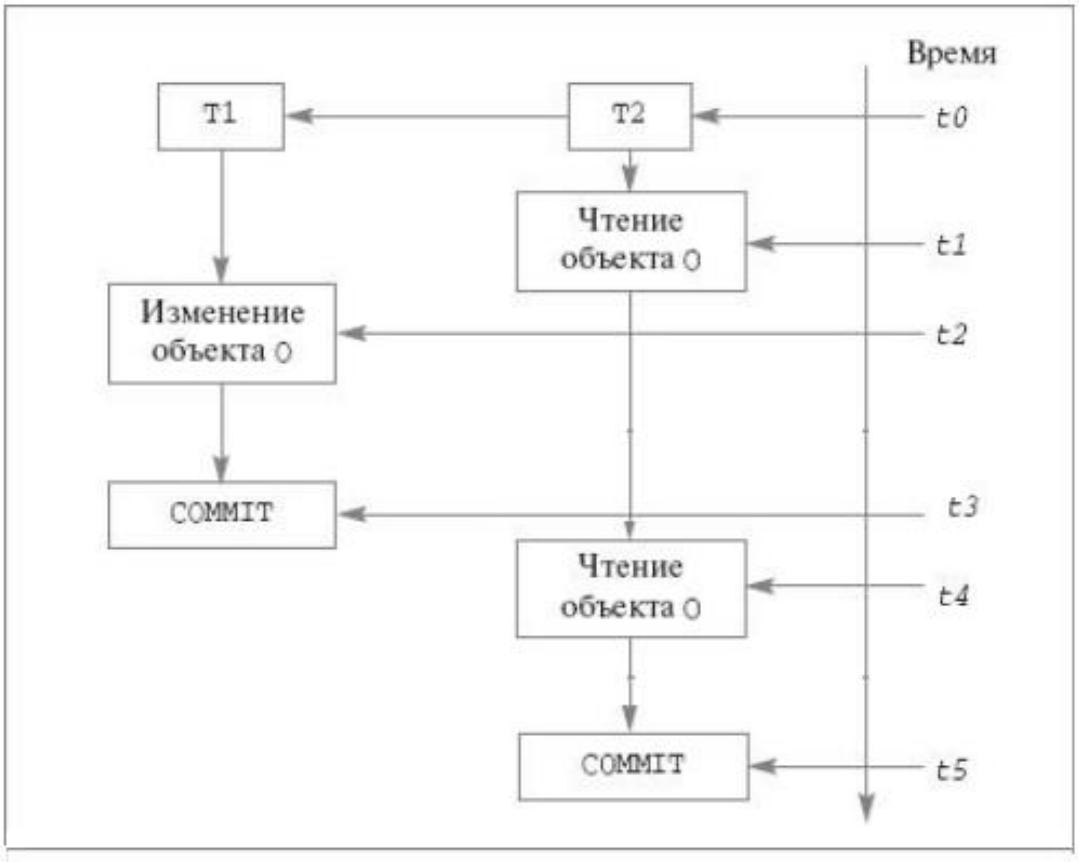
Черновое чтение

Ситуация **чернового чтения** возникает, когда транзакция считывает изменения, вносимые другой (незавершённой) транзакцией.



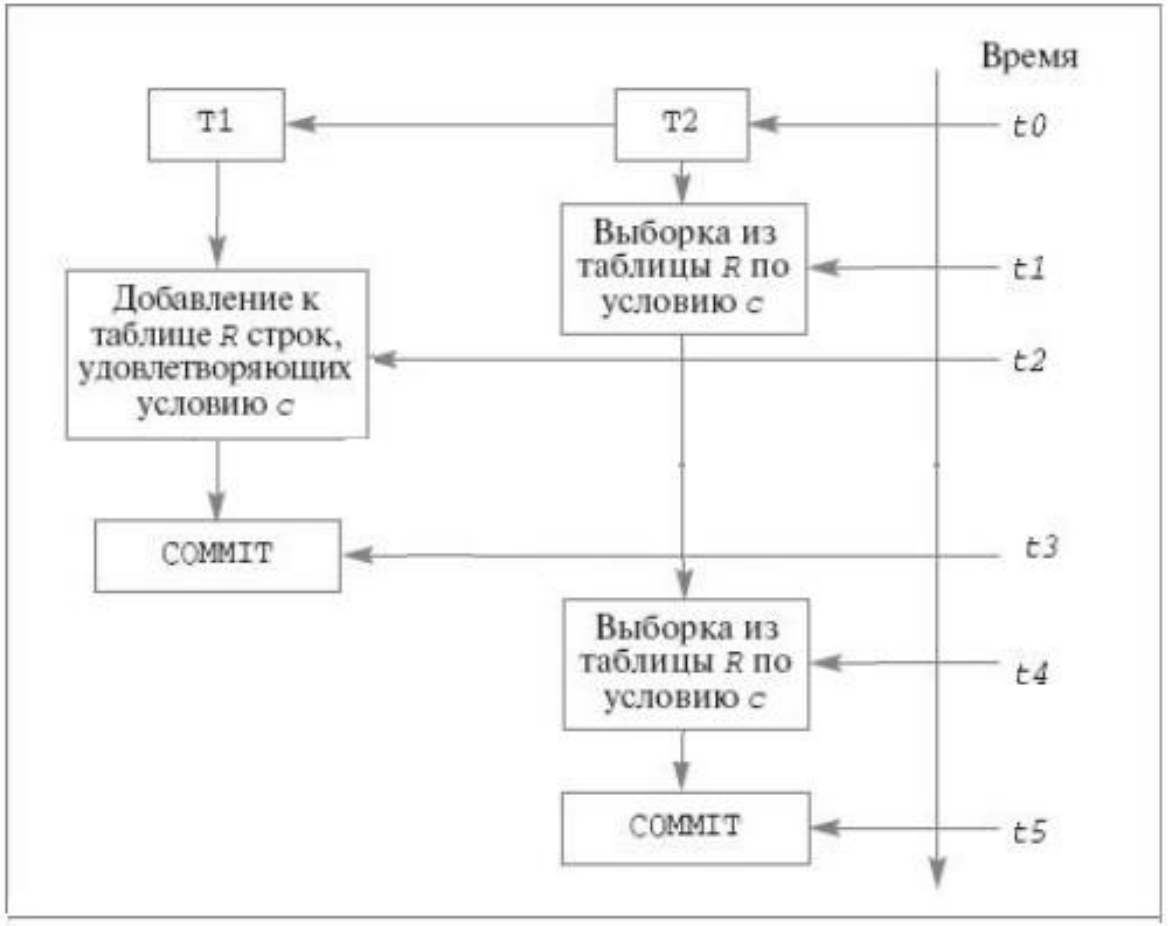
Неповторяемое чтение

Неповторяемое чтение является противоположностью повторяемого, т.е. транзакция "видит" изменения существующих данных, внесённые другими (завершёнными!) транзакциями.



Фантомы

Фантомы возникают, если транзакция "видит" новые данные, добавленные другими (завершёнными!) транзакциями.



Уровни изоляции транзакций

Уровень изоляции	Черновое чтение	Неповторяемое чтение	Фантомы
Read Uncommited – чтение незавершённых транзакций	да	да	да
Read Commited – чтение завершённых транзакций	нет	да	да
Repeatable Read – повторяемое чтение	нет	нет	да
Serializable – последовательное чтение	нет	нет	нет

Управление параллельным доступом

Управление параллельным доступом – это процесс организации одновременного выполнения а базе данных различных операций доступа, гарантирующий предотвращение их влияния друг на друга.
Существуют два основных метода управления параллельным доступом, позволяющих организовать одновременное безопасное выполнение транзакций при соблюдении определенных ограничений: ***метод блокировки*** и ***метод временных отметок***.
Оба этих метода являются *консервативными* (или *пессимистическими)* подходами, поскольку они откладывают выполнение транзакций, способных в будущем в тот или иной момент времени войти в конфликт с другими транзакциями.
Оптимистические методы строятся на предположении, что вероятность конфликта невысока, поэтому они допускают асинхронное выполнение транзакций, а проверка на наличие конфликта откладывается на момент их завершения и фиксации в базе данных.

Блокировки

Блокировка – это временное ограничение доступа к данным, участвующим в транзакции, со стороны других транзакций.

Различают следующие типы блокировок:

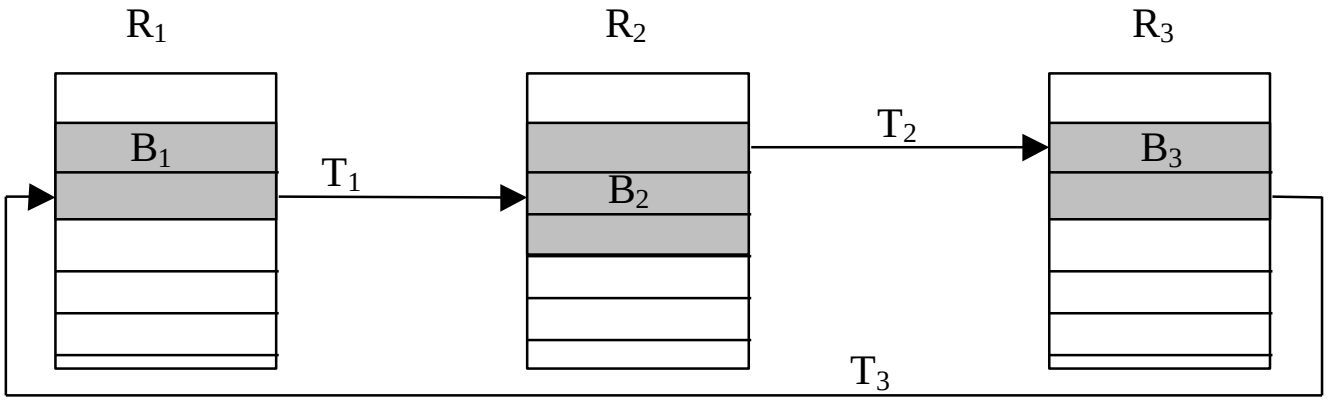
- по степени доступности данных: разделяемые и исключающие;
- по множеству блокируемых данных: блокировка может накладываться на всю базу данных, на файл, на таблицу, на страницу (блок) памяти, на запись таблицы, на отдельное поле записи;
- по способу установки: автоматические и явные.

Явная блокировка, накладываемая командой LOCK TABLE языка SQL.
LOCK TABLE <имя таблицы> IN <тип блокировки> [NOWAIT | WAIT];

Явную блокировку также можно наложить с помощью ключевых слов for update, например:

```
SELECT *
    FROM <имя_таблицы>
    WHERE <условие>
    for update;
```

Тупиковые ситуации (deadlocks)



Стратегии разрешения проблемы взаимной блокировки:

1. Вводится **таймаут (**time-out) – максимальное время, в течение которого транзакция может находиться в состоянии ожидания. Если транзакция находится в состоянии ожидания дольше таймаута, считается, что она находится в состоянии тупика, и СУБД инициирует её откат с последующим рестартом через случайный промежуток времени.

Тупиковые ситуации (deadlocks)

Стратегии разрешения проблемы взаимной блокировки:

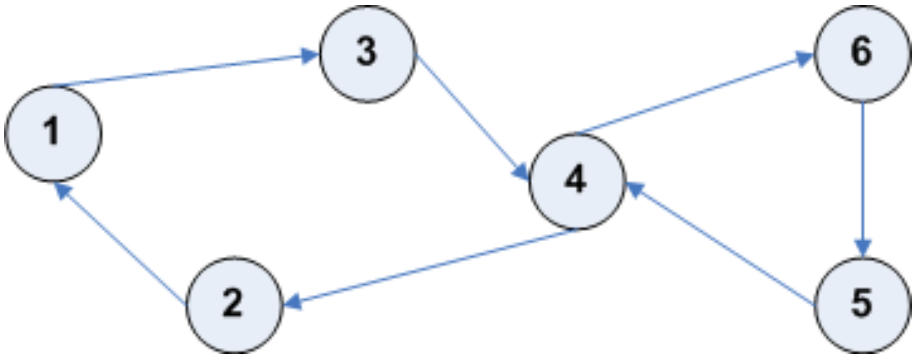
2. Предотвращение взаимных блокировок. Достигается применением одного из двух алгоритмов:
   — ожидание отмены. Предполагается, что старые транзакции отменяются в пользу новых и перезапускаются с новой временной отметкой.
   — отмена ожидания. Предполагает, что следует завершить старые транзакции, перезапустив новые с прежней временной отметкой.
3. Выявление возникающих тупиков и отмена одной из транзакций с её последующим рестартом через случайный промежуток времени. Этот метод требует дополнительных накладных расходов. СУБД обычно строит граф ожидания транзакций, и наличие цикла в графе говорит о взаимной блокировке. Запуск этого алгоритма происходит периодически, причем период увеличивается в случае отсутствия выявленных тупиков и уменьшается при их обнаружении.

Тупиковые ситуации (deadlocks)

При выявлении тупиков необходимо выбрать, какую транзакцию стоит отменить и перезапустить. Критериями выбора могут быть:

1. Минимальная продолжительность выполнения транзакции.
2. Минимальное количество элементов данных, обновленных в транзакции (откат будет занимать мало времени).
3. Минимальное количество элементов данных, которые все еще подлежат обновлению в транзакции (уже почти все сделано).
4. Максимальное количество транзакций, ожидающих завершения данной транзакции (отменив эту транзакцию, можно разорвать сразу несколько циклов, например, транзакцию 4 на рисунке снизу).

Во избежание постоянной отмены одной и той же транзакции система периодически меняет критерии, учитывая предельно допустимое количество перезапусков транзакции.



Временн***ы***е отметки

Временн***а***я отметка – это уникальный идентификатор, который СУБД создаёт для обозначения относительного момента запуска транзакции. Каждая транзакция ***Ti*** имеет временн***у***ю отметку ***ti***, и каждый элемент данных в БД (запись или блок) имеет две отметки:

tread(x) – временн***а***я отметка транзакции, которая последней считала элемент ***x***,

twrite(x) – временн***а***я отметка транзакции, которая последней записала элемент ***x***.

При выполнении транзакции ***Ti*** система сравнивает отметку ***ti*** и отметки ***tread(x)*** и ***twrite(x)*** элемента ***x*** для обнаружения конфликтов:

для читающей транзакции ***Ti***:

ti < twrite(x).

для пишущей транзакции:

ti < tread(x),

ti < twrite(x).

Во всех случаях обнаружения конфликта система перезапускает текущую транзакцию ***Ti*** с более поздней временн***о***й отметкой.

Оптимистический протокол управления параллельностью

Включает две или три стадии в зависимости от типа транзакции:

• Стадия чтения: от начала транзакции и до момента, предшествующего фиксации результатов. Транзакция считывает из базы данных значения всех необходимых ей элементов данных и помещает их в ОП. Любые обновления применяются только к локальной копии данных в ОП.

• Стадия проверки. Для читающих транзакций проверка состоит в подтверждении того, что использованные транзакцией значения по-прежнему остаются текущими значениями соответствующих элементов данных. Если нарушений нет, транзакция завершает свое выполнение путем фиксации. Если найдены измененные значения, транзакция отменяется и перезапускается. Если в транзакции выполнялось обновление данных, то проверка включает выполнение контроля, сохранится ли база данных в согласованном состоянии после внесения в нее результатов данной транзакции, а также не будут ли нарушены условия упорядочиваемости. В случае обнаружения ошибки транзакция отменяется и перезапускается.

• Стадия записи выполняется после успешного завершения стадии проверки, но только в отношении пишущих транзакций. Все изменения, внесенные в локальные копии данных, переносятся собственно в БД.

Оптимистический протокол управления параллельностью

Каждой транзакции в начале ее выполнения присваивается некоторая временная отметка, *start (T)*. Дополнительные временные отметки присваиваются транзакции в начале стадии проверки (*validation (T)*) и в момент завершения ее выполнения (*finish(T)* – включая и стадию записи). Для успешности проверки должно соблюдаться одно из следующих условий.

1. Все транзакции S с более старыми временными отметками должны быть уже завершены до начала выполнения транзакции T*: finish (S) < start (T).*
2. Если транзакция T начала выполняться до завершения какой-либо из транзакций S, то должны соблюдаться требования:
   а) множество элементов данных, записанных начавшейся ранее транзакцией, не должно совпадать ни с одним из элементов данных, прочитанных указанной транзакцией;
   б) стадия записи начавшейся ранее транзакции должна быть завершена до перехода текущей транзакции на стадию проверки: start (T) < finish(S) < validation(T).

Правило 2,а гарантирует, что данные, записанные начавшейся ранее транзакцией, не считывались текущей транзакцией. Правило 2,б гарантирует, что операции записи выполняются последовательно, а это гарантирует отсутствие конфликтов.

Степень детализации данных

В свете управление параллельностью доступа к данным под элементом данных может пониматься:

- Вся база данных.
- Отдельный файл.
- Отдельная таблица БД.
- Отдельная страница (блок) памяти.
- Отдельная запись таблицы.
- Отдельное поле записи таблицы.

Чем больший объем элемента данных, тем ниже степень параллельности, и наоборот. Но при маленьком объеме элемента увеличиваются издержки на поддержание блокировок (или проверок при временных отметках). Большинство СУБД динамически изменяют размер элемента. В частности, некоторые системы автоматически повышают степень детализации блокировок от уровня отдельной записи до уровня страницы или файла, если определенная транзакция блокирует больше установленного процента записей или страниц некоторого файла. Существующие степени детализации блокируемых элементов можно представить в виде иерархической структуры, и блокировка элемента верхнего уровня означает автоматическую блокировку входящих в него элементов.