

## 10. Транзакции

Транзакцией называется последовательность операций, производимых над БД и переводящих базу данных из одного непротиворечивого (согласованного) состояния в другое непротиворечивое (согласованное) состояние. Транзакция рассматривается как некоторое неделимое действие над БД, осмысленное с точки зрения пользователя. В то же время это логическая единица работы системы.

Рассмотрим пример, связанный с принятием заказа в фирме на изготовление компьютера. Компьютер состоит из комплектующих, которые сразу резервируются за данным заказом в момент его формирования. Тогда транзакцией будет вся последовательность операций, включающая следующие этапы:

- 1) ввод нового заказа со всеми реквизитами заказчика;
- 2) изменение состояния для всех выбранных комплектующих на складе на «занято» с привязкой к определенному заказу;
- 3) подсчет стоимости заказа с формированием платежного документа;
- 4) включение нового заказа в производство.

С точки зрения работника, это единая последовательность операций; если она будет прервана, то БД потеряет свое целостное состояние.

Существуют различные модели транзакций, которые могут быть классифицированы на основании различных свойств, включающих структуру транзакции, параллельность внутри транзакции, продолжительность и т.д.

В настоящий момент выделяют следующие типы транзакций: классические (традиционные) транзакции, цепочечные транзакции и вложенные транзакции.

Традиционные транзакции характеризуются четырьмя основными свойствами: атомарности, согласованности, изолированности, долговечности (прочности) — ACID (*Atomicity, Consistency, Isolation, Durability*). Иногда традиционные транзакции называют ACID-транзакциями. Упомянутые выше свойства означают следующее:

1. *Свойство атомарности* (Atomicity) выражается в том, что транзакция должна быть выполнена в целом или не выполнена вовсе.
2. *Свойство согласованности* (Consistency) гарантирует, что по мере выполнения транзакций данные переходят из одного согласованного состояния в другое — транзакция не разрушает взаимной согласованности данных.

3. *Свойство изолированности (Isolation)* означает, что конкурирующие за доступ к базе данных транзакции физически обрабатываются последовательно, изолированно друг от друга, но для пользователей это выглядит так, как будто они выполняются параллельно.
4. *Свойство долговечности (Durability)* трактуется следующим образом: если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах (даже в случае последующих ошибок).

Возможны два варианта завершения транзакции. Если все операторы выполнены успешно и в процессе выполнения транзакции не произошло никаких сбоев программного или аппаратного обеспечения, транзакция фиксируется.

**Фиксация транзакции** — это действие, обеспечивающее запись на диск изменений в базе данных, которые были сделаны в процессе выполнения транзакции.

До тех пор, пока транзакция не зафиксирована, возможно аннулирование этих изменений, восстановление базы данных в то состояние, в котором она была на момент начала транзакции. Фиксация транзакции означает, что все результаты выполнения транзакции становятся постоянными. Они станут видимыми другим транзакциям только после того, как текущая транзакция будет зафиксирована. До этого момента все данные, затрагиваемые транзакцией, будут «видны» пользователю в состоянии на начало текущей транзакции.

Если в процессе выполнения транзакции случилось нечто такое, что делает невозможным ее нормальное завершение, база данных должна быть возвращена в исходное состояние. **Откат транзакции** — это действие, обеспечивающее аннулирование всех изменений данных, которые были сделаны операторами SQL в теле текущей незавершенной транзакции.

Каждый оператор в транзакции выполняет свою часть работы, но для успешного завершения всей работы в целом требуется безусловное завершение всех их операторов. Группирование операторов в транзакции сообщает СУБД, что вся эта группа должна быть выполнена как единое целое, причем такое выполнение должно поддерживаться автоматически.

В стандарте ANSI/ISO SQL определены модель транзакций и функции операторов COMMIT и ROLLBACK. Стандарт определяет, что транзакция начинается с первого SQL-оператора, инициируемого пользователем или содержащегося в программе. Все последующие SQL-операторы составляют тело транзакции. Транзакция завершается одним из четырех возможных путей:

- 1) оператор COMMIT означает успешное завершение транзакции; его использование делает постоянными изменения, внесенные в базу данных в рамках текущей транзакции;

- 2) оператор ROLLBACK прерывает транзакцию, отменяя изменения, сделанные в базе данных в рамках этой транзакции; новая транзакция начинается непосредственно после использования ROLLBACK;
- 3) успешное завершение программы, в которой была инициирована текущая транзакция, означает успешное завершение транзакции (как будто был использован оператор COMMIT);
- 4) ошибочное завершение программы прерывает транзакцию (как будто был использован оператор ROLLBACK).

В этой модели каждый оператор, который изменяет состояние БД, рассматривается как транзакция, поэтому при успешном завершении этого оператора БД переходит в новое устойчивое состояние.

Реализация в СУБД принципа сохранения промежуточных состояний, подтверждения или отката транзакций обеспечивается специальным механизмом, называемым *журналом транзакций*.

Обеспечение надежного хранения данных в БД предполагает, в частности, возможность восстановления согласованного состояния базы данных после любого рода аппаратных и программных сбоев. Очевидно, что для выполнения восстановлений необходима некоторая дополнительная информация. В подавляющем большинстве современных реляционных СУБД такая избыточная информация поддерживается в виде журнала изменений базы данных и называемого журналом транзакций.

Итак, общей целью журнализации изменений БД является обеспечение возможности восстановления согласованного состояния БД после любого сбоя. Поскольку основой поддержания целостного состояния БД является механизм транзакций, журнализация и восстановление тесно связаны с понятием транзакции.

Общими принципами восстановления являются следующие:

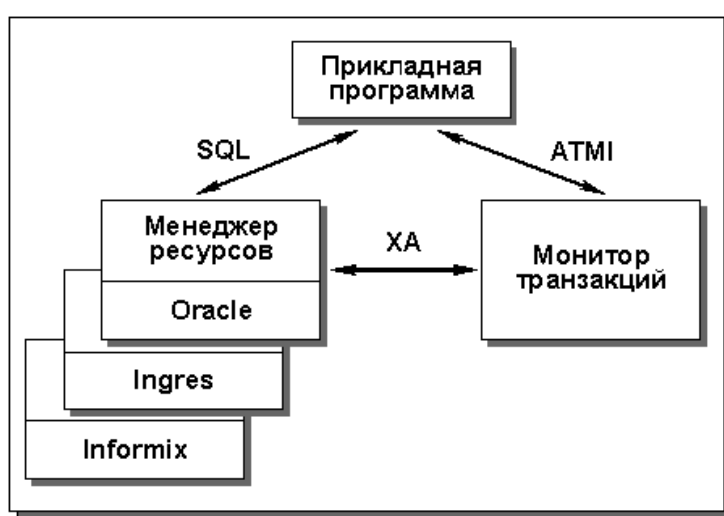
- результаты зафиксированных транзакций должны быть сохранены в восстановленном состоянии БД;
- результаты незафиксированных транзакций должны отсутствовать в восстановленном состоянии БД.

Это означает, что восстанавливается последнее по времени согласованное состояние БД.

Для оперативной обработки распределенных транзакций в неоднородных вычислительных системах (содержащих компьютеры, работающие под различными операционными системами) были созданы *мониторы обработки транзакций* (*Transaction Processing Monitor* — TPM), или мониторы транзакций. TPM опираются на трехзвенную модель клиент-сервер (модель сервера приложений или AS-модель).

Приложение ТРМ позволяет выполнять масштабирование системы, поддерживать функциональную полноту и целостность приложений а также повысить производительность обработки данных при невысокой стоимости накладных расходов.

Принципы организации обработки информации с помощью монитора транзакций описываются **моделью монитора транзакций X/Open DTP** (*Distributed Transaction Processing* — обработка распределенных транзакций). Эта модель (рисунок 1) включает в себя три объекта: прикладную программу, менеджер ресурсов (*Resource Manager* — RM) и монитор, или менеджер, транзакций (*Transaction Manager* — TM).



**Рисунок 1** — Модель обработки транзакций X/Open DTP

В качестве прикладной программы может выступать произвольная программа-клиент. RM выполняет функции сервера ресурса некоторого вида. Прикладная программа взаимодействует с RM с помощью набора специальных функций либо посредством операторов SQL (когда сервером является сервер БД).

Интерфейс ATMI (*Application Transaction Monitor Interface* — интерфейс монитора транзакций приложения) позволяет вызывать функции ТРМ на некотором языке программирования, например, Си.

Функции менеджера ресурсов обычно выполняют серверы БД или СУБД. В задачах организации управления обработкой распределенных транзакций (транзакций, затрагивающих программные объекты вычислительной сети) ТМ взаимодействует с RM, который должен поддерживать протокол двухфазной фиксации транзакций и удовлетворять стандарту X/Open XA. Примерами СУБД, поддерживающих протокол двухфазной фиксации транзакции, являются Oracle 7.0, Open INGRES и Informix-Online 5.0.

Понятие транзакции в ТРМ несколько шире, чем в СУБД, где транзакция включает в себя элементарные действия над данными базы. Здесь транзакция может охватывать и другие необходимые действия: передачу сообщения, запись информации в файл, опрос датчиков и т. д. Это значит, что ТРМ предоставляет более мощные средства управления вычислительным процессом. Транзакции, которые поддерживают ТРМ, называют также *прикладными транзакциями*.

Модель X/Open DTP не раскрывает структуру ТМ в деталях, а определяет состав компонентов распределенной системы обработки информации и как эти компоненты взаимодействуют друг с другом. Практические реализации этой модели, естественно, могут отличаться друг от друга. В числе примеров реализаций мониторов транзакций можно назвать ACMS, CICS, и TUXEDO System.

Для *разработчиков приложений* мониторы обработки транзакций ТРМ предоставляют удобства, связанные с возможностью декомпозиции приложений по нескольким функциональным уровням со стандартными интерфейсами, что позволяет создавать легко модифицируемые ИС со стройной архитектурой.

Прикладные программы становятся практически независимыми от конкретной реализации интерфейса с пользователем и от менеджера ресурсов. Первое означает, что для реализации функций представления можно выбрать любое удобное и привычное для разработчика средство (от языка Си до какой-либо CASE-системы). Независимость от менеджера ресурсов подразумевает возможность легкой замены одного менеджера ресурсов на другой, лишь бы они поддерживали стандарт взаимодействия с прикладной программой (для СУБД — язык SQL).

Если ТРМ поддерживает множество аппаратно-программных платформ, как например, TUXEDO System, то разрабатываемые приложения становятся, кроме того, мобильными.

Сосредоточение всех прикладных функций в серверах приложений и наличие богатых возможностей управления и администрирования существенно упрощает обновление прикладных функций и контроль за их непротиворечивостью. Изменения в прикладных функциях при этом никак не отражаются на программах-клиентах.

Для *пользователей распределенных систем* ТРМ позволяют улучшить показатели пропускной способности и времени отклика, снизить стоимость обработки данных в оперативном режиме на основе эффективной организации вычислительного процесса.

Улучшение показателей функционирования достигается благодаря осуществлению статической и динамической балансировки нагрузки.

Управление загрузкой состоит в запуске или остановке AS-процессов (программных компонентов AS-модели) в зависимости от заранее установленных параметров и текущего состояния системы. При необходимости ТРМ может тиражировать копии AS-процессов на этом или других узлах сети.

*Администраторы* распределенных систем, имея ТРМ, получают возможность легкого масштабирования ИС и увеличения производительности обработки информации. Здесь, кроме вертикального и горизонтального масштабирования, можно обеспечить так называемое матричное масштабирование. Суть его в том, что без остановки серверов приложений в любое время может быть добавлен, например, компьютер или менеджер ресурсов.

Кроме того, администраторы систем получают возможность снизить общую стоимость программного обеспечения систем клиент-сервер. Снижение стоимости можно достигнуть простым уменьшением количества подключений к серверам БД.

Стоимость серверов БД (или СУБД) в сильной степени зависит от числа одновременных подключений к программе. Уменьшение количества подключений к серверу БД достигается путем мультиплексирования запросов или, что то же самое, логического преобразования потока запросов от многих источников к потоку от одного или нескольких источников. Выигрыш в стоимости и производительности при эффективной реализации самих ТРМ может оказаться весьма существенным.