Раздел 4. Основы языка SQL

Тема 4.4

Средства (операторы) управления транзакциями

Вопросы лекции:

1. **Понятие транзакции и свойства транзакций.**
2. Механизмы СУБД для поддержки транзакций.
3. Параллельные транзакции и блокировки.

Транзакция — это последовательность операций, производимых над БД и переводящая ее из одного непротиворечивого (согласованного) состояния в другое непротиворечивое (согласованное) состояние.

Транзакция — это последовательность предложений (операторов) SQL, которые **рассматриваются как единое целое,** осмысленное с точки зрения пользователя **- логически неделимая единица** работы с БД.

Под этим подразумевается следующее: **либо все предложения**, содержащиеся в данной транзакции, выполнятся успешно, **либо ни одно из них** не будет выполнено (БД вернется в исходное состояние до начала транзакции).

Это свойство **поддерживается СУБД**, **даже если** в процессе выполнения транзакции произойдет **программный или аппаратный сбой**.

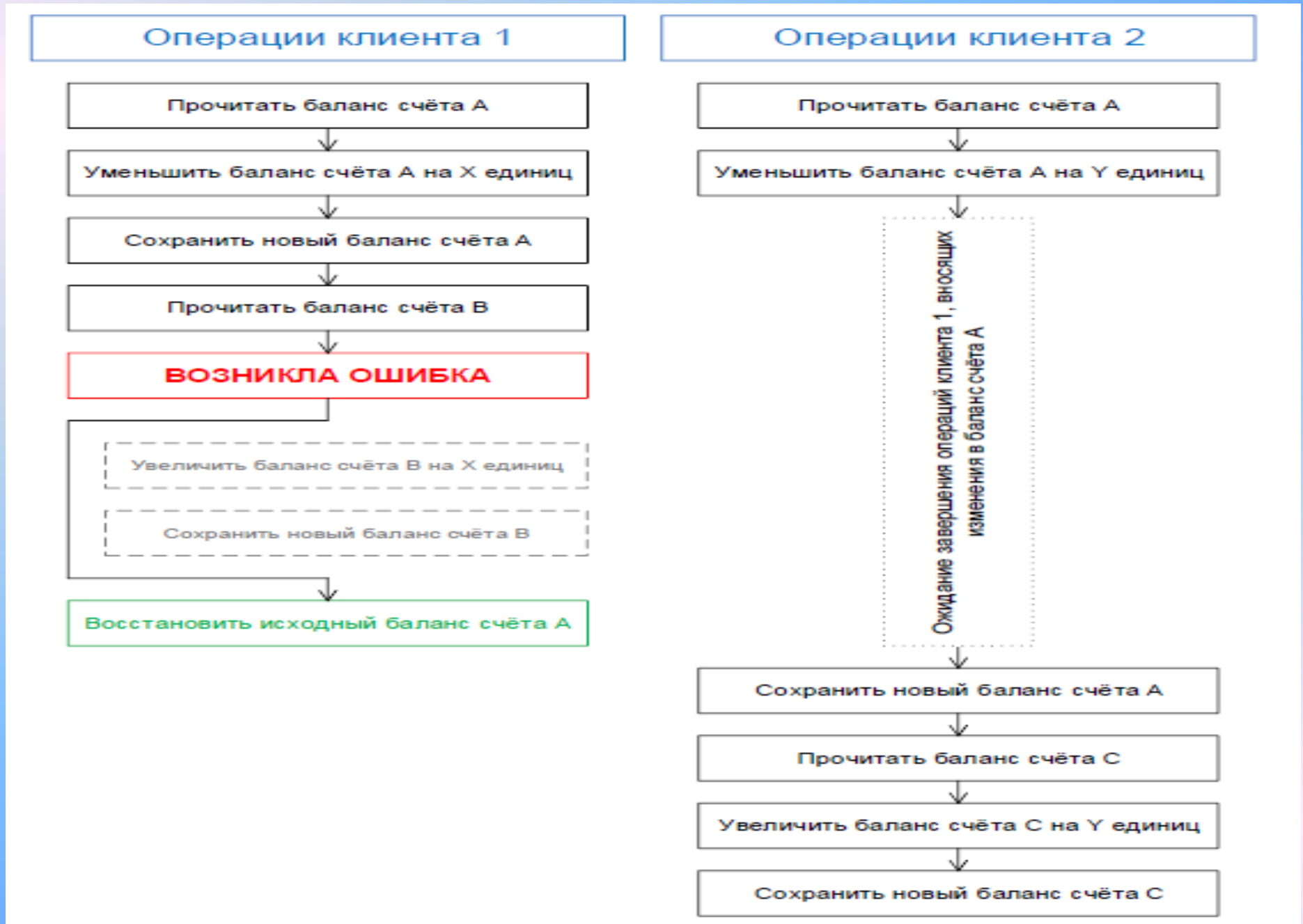
Транзакции преобразуют базу данных **из одного целостного (непротиворечивого) состояния в другое**, но **в процессе выполнения транзакции** допускается нарушение целостности (противоречивость данных).

Транзакция также может гарантировать, что в процессе ее выполнения промежуточное **противоречивое состояние** базы данных **будет невидимым** для других пользователей.

Пример транзакции – оформление заказа на выпуск изделия

Последовательность операций:

1. Ввод нового заказа со всеми реквизитами заказчика,
2. Изменения состояния для всех выбранных комплектующих на складе на «занято» с привязкой их к определенному заказу;
3. Подсчет стоимости заказа с формированием платежного документа типа выставляемого счета к оплате,
4. Включение нового заказа в производство – фиксация в БД работы с заказом.



Логика работы транзакций

В настоящий момент выделяют следующие **типы транзакций**:

- плоские или классические транзакции,
- цепочечные транзакции и
- вложенные транзакции.

Поддерживаются не всеми СУБД

Плоские, или традиционные, транзакции, характеризуются четырьмя классическими свойствами:

- атомарности,
- согласованности (целостности),
- изолированности,
- долговечности (прочности).

Сокращенно — **ACID** (Atomicity, Consistency, Isolation, Durability).

Иногда традиционные транзакции называют ACID-транзакциями.

Свойство атомарности (**Atomicity**) выражается в том, что транзакция является неделимой (*как раньше считался атом*), она должна быть выполнена в целом или не выполнена вовсе.

Если транзакция прерывается не дойдя до конца, то база данных должна остаться в том состоянии, которое она имела до начала транзакции.

Свойство согласованности (**Consistency**) гарантирует, что по мере выполнения транзакций данные переходят из одного согласованного (целостного) состояния в другое, также целостное — транзакция не разрушает взаимной согласованности данных.

Важно – в процессе выполнения транзакции БД может временно пребывать в нецелостном состоянии, но оно не должно быть видно другим пользователям БД.

Очень многие правила целостности БД таковы, что их просто невозможно не нарушить, выполняя каждую одиночную команду SQL как отдельную транзакцию. Чтобы не потерять целостность БД (ее смысловое соответствие реальной предметной области), требуется объединение нескольких команд в **единый блок**.

Свойство согласованности

Пример транзакции:

Процесс удаления студента из основной БД с переносом данных о нем и всех его оценок в архивную БД.

На каждом этапе выполнения отдельных команд все промежуточные состояния БД являются несогласованными и при любом сбое будет возврат в исходное состояние (**откат, инициированный сервером - СУБД**).

После завершающей команды БД уже в согласованном состоянии, но **откат еще возможен (по команде ROLLBACK пользователя)**.

После фиксации откат невозможен, все вернуть можно новой транзакцией.

Свойство изолированности (**Isolation**) означает, что в многопользовательской среде при работе с БД 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**.

Псевдо-одновременно выполняющиеся транзакции не наложатся и не исказят результаты друг друга. Такой режим работы серверов БД называется сериальным и является режимом по умолчанию для многопользовательских БД.

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 и гарантированно сохраняются в БД.

В случае последующих ошибок и аварий в БД можно восстановить все зафиксированные транзакции.

Раздел 4. Основы языка SQL

Тема 4.4

Средства (операторы) управления транзакциями

Вопросы лекции:

1. Понятие транзакции и свойства транзакций.
2. **Механизмы СУБД для поддержки транзакций.**
3. Параллельные транзакции и блокировки.

Механизмы СУБД предусматривают **два варианта завершения транзакции – фиксация транзакции или откат транзакции**.

1. Если все операторы, входящие в транзакцию, 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 этого момента все данные, затрагиваемые транзакцией, будут «видны» любому пользователю в состоянии «на начало» текущей транзакции.

Фиксация транзакции заключается в следующем:

1. Изменения, внесённые транзакцией, делаются постоянными.
2. Уничтожаются все точки сохранения для данной транзакции.
3. Завершается транзакция (уничтожаются системные записи о транзакции в оперативной памяти).
4. Если выполнение транзакций осуществляется с помощью блокировок, то освобождаются объекты, заблокированные транзакцией.

Механизмы СУБД предусматривают **два варианта завершения транзакции – фиксация транзакции или откат транзакции** ***(продолжение)***.

2. Если в процессе выполнения транзакции случилось нечто такое, что делает невозможным ее нормальное завершение, а также по требованию пользователя - 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.

Для обеспечения механизма отката в СУБД реализуется **журнализация** хода выполнения всех транзакций.

Как правило, для каждой транзакции ведется отдельный журнал, который при успешном ее завершении может быть удален или переписан в **единый журнал работы СУБД** для возможности восстановления БД из ее ранее созданных архивных копий.

Реализация в СУБД принципа сохранения промежуточных состояний, подтверждения или отката транзакции **обеспечивается специальным механизмом**, для поддержки которого создается некоторая системная структура, называемая ***Журналом транзакций***.

Ведение журналов транзакций одновременно преследует две цели:

1) Возможность отката отдельных транзакций;
2) Восстановление информации и согласованного состояния БД в случае аварийных ситуаций или программных или аппаратных сбоев.

Как правило серверами ведутся 2 вида журналов:

- **Undo**-журналы – для отката каждой отдельной транзакции, удаляются после операций COMMIT или ROLLBACK;
- **Redo**-журнал – единый системный журнал для обеспечения **повторного выполнения** всех транзакций.

Этот механизм полностью обеспечивает выполнение свойства «долговечность».
Обычно реализуется правило **упреждающей записи** в Redo-журнал и затем в БД.

Возможны следующие **три ситуации,** при которых требуется производить восстановление состояния базы данных.

1) Индивидуальный откат транзакции с помощью Undo-журнала**.**
Этот откат должен быть применен в следующих случаях:

- стандартной ситуацией отката транзакции является ее явное завершение оператором ROLLBACK, введенным пользователем;

- аварийное завершение работы прикладной программы, которое логически эквивалентно выполнению оператора ROLLBACK, но физически имеет иной механизм выполнения;

- принудительный откат транзакции в случае взаимной блокировки при параллельном выполнении транзакций. В подобном случае для выхода из тупика данная транзакция может быть выбрана в качестве «жертвы» и принудительно прекращено ее выполнение **ядром СУБД**.

Возможны следующие **ситуации, при которых требуется производить восстановление** состояния базы данных.

2) Восстановление после внезапной потери содержимого **оперативной памяти** (**мягкий сбой**) – используется Redo-журнал и имеющаяся активная БД.

Такая ситуация может возникнуть в следующих случаях:

- при аварийном выключении электрического питания;
- при возникновении неустранимого сбоя процессора (например, срабатывании контроля оперативной памяти) и т. д. Ситуация характеризуется потерей той части базы данных, которая к моменту сбоя содержалась в буферах оперативной памяти.

3) Восстановление после поломки основного внешнего носителя базы данных (**жесткий сбой**) – также используется Redo-журнал + архивная копия БД.

Общими **принципами восстановления** являются следующие:

- **результаты зафиксированных транзакций** должны быть сохранены в восстановленном состоянии базы данных;

- **результаты незафиксированных транзакций** должны отсутствовать в восстановленном состоянии базы данных.

Это, собственно, и означает, что восстанавливается последнее по времени согласованное состояние базы данных.

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.

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

- Стандартная модель ANSI / ISO;
- Расширенная модель.

В стандарте ANSI/ISO SQL определены модель транзакций и функции операторов **COMMIT** и **ROLLBACK**.

Стандарт определяет, что транзакция **неявно начинается** с первого SQL-оператора, инициируемого пользователем или содержащегося в программе, изменяющего текущее состояние базы данных.

Все последующие SQL-операторы составляют **тело транзакции**.

Транзакция завершается одним из четырех возможных путей :

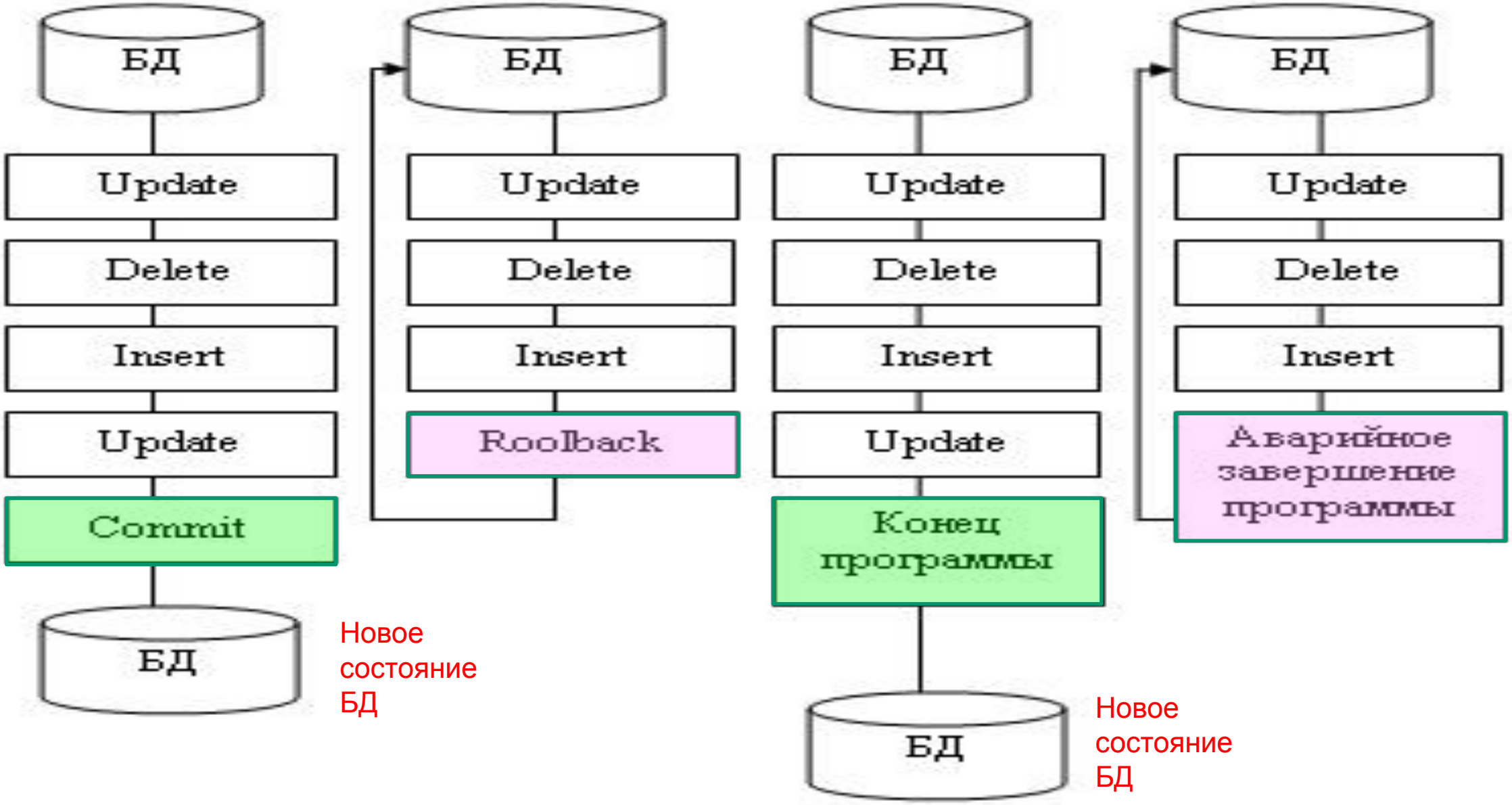
- оператор **COMMIT** означает **успешное завершение транзакции**; его использование делает постоянными изменения, внесенные в базу данных в рамках текущей транзакции;

- оператор **ROLLBACK** **прерывает транзакцию**, отменяя изменения, сделанные в базе данных в рамках этой транзакции; новая транзакция начинается непосредственно после использования ROLLBACK;

- **успешное** **завершение** программы, в которой была инициирована текущая транзакция, **означает успешное завершение транзакции** (как будто был использован оператор COMMIT);

-**ошибочное** **завершение** программы **прерывает транзакцию** (как будто был использован оператор ROLLBACK).

В некоторых СУБД каждый оператор, который изменяет состояние БД, **может рассматривться как транзакция**, поэтому при успешном завершении этого оператора **БД переходит в новое устойчивое состояние**.



Стандартная модель транзакций ANSI/ISO

Расширенная модель транзакций, определенная **стандартом SQL-3**, включает еще ряд дополнительных операций.

- Оператор **BEGIN TRANS [ACTION] –** (в некоторых СУБД - START TRANS) сообщает о начале транзакции.
- Оператор **COMMIT** - сообщает об успешном завершении транзакции.
- Оператор **SAVEPOINT** [NameSavepoint] - создает внутри транзакции точку сохранения, которая соответствует промежуточному состоянию БД, сохраненному на момент выполнения этого оператора. NameSavepoint - имя точки сохранения, если не задано – СУБД присваивает внутреннее имя.
- Оператор **ROLLBACK** - оператор отката, имеет **две модификации**.

Если этот оператор используется без дополнительного параметра, то он интерпретируется как оператор отката всей транзакции, то есть в этом случае он **эквивалентен оператору отката ROLLBACK** в модели ANSI/ISO.

Если же оператор отката имеет параметр и записан в виде ROLLBACK B, то он интерпретируется как **оператор частичного отката** транзакции в точку сохранения B.

Выполнение транзакций в расширенной модели

Раздел 4. Основы языка SQL

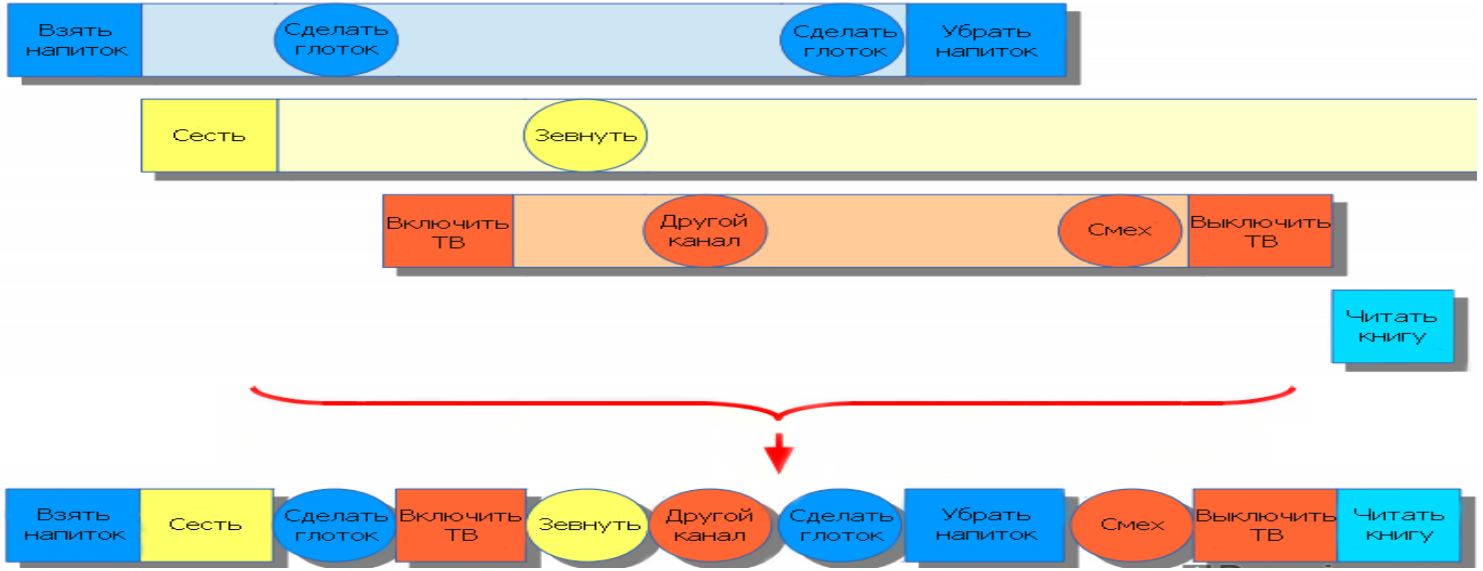
Тема 4.4

Средства (операторы) управления транзакциями

Вопросы лекции:

1. Понятие транзакции и свойства транзакций.
2. Механизмы СУБД для поддержки транзакций.
3. **Параллельные транзакции и блокировки.**

Если с БД работают одновременно несколько пользователей, то обработка транзакций должна рассматриваться с новой точки зрения.



В этом случае СУБД должна

- не только корректно выполнять индивидуальные транзакции и восстанавливать согласованное состояние БД после сбоев, но она призвана
- обеспечить корректную параллельную работу всех пользователей над одними и теми же данными.

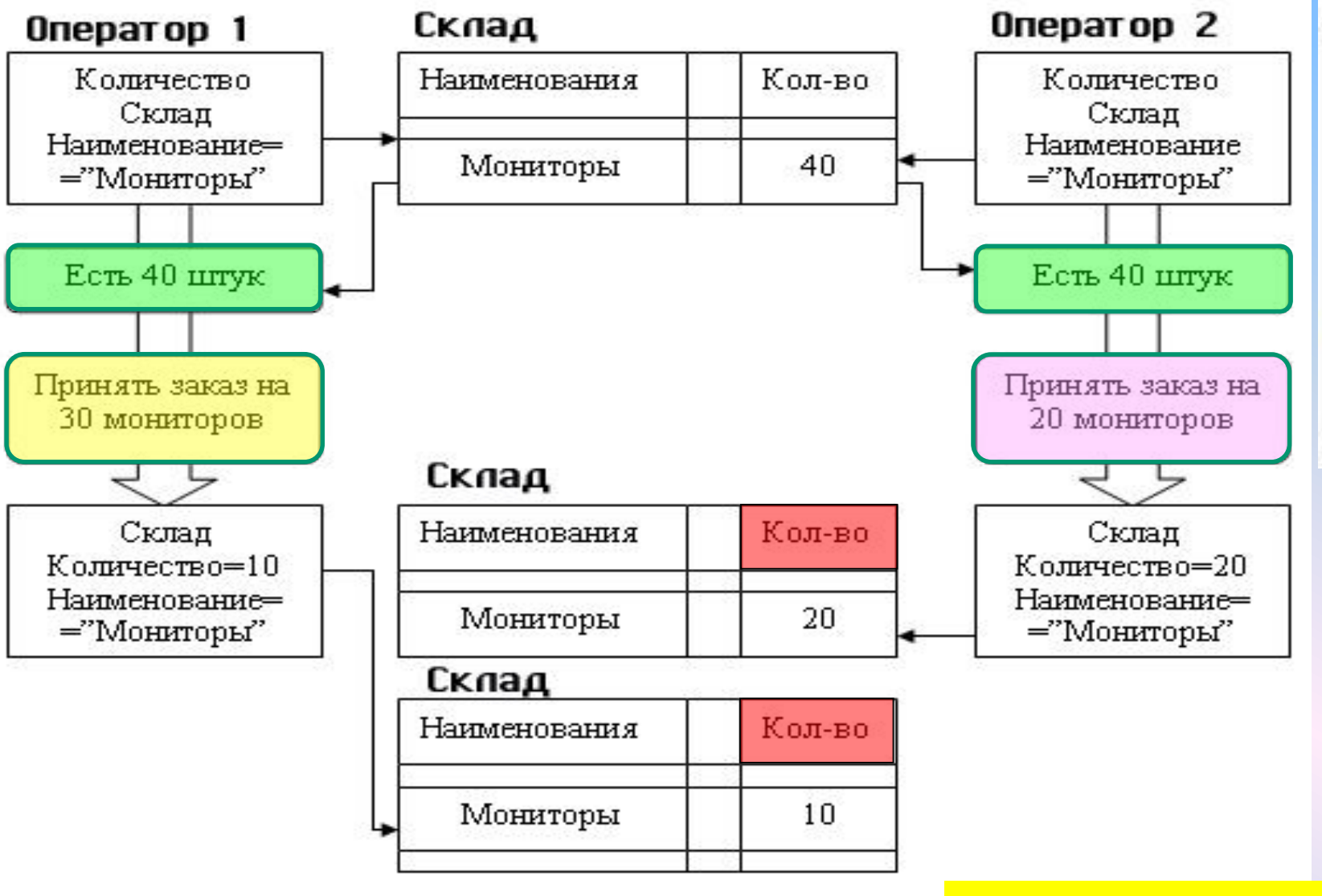
По теории каждый пользователь и каждая транзакция должны обладать свойством **изолированности**, то есть они должны выполняться так, как если бы только один пользователь работал с БД.

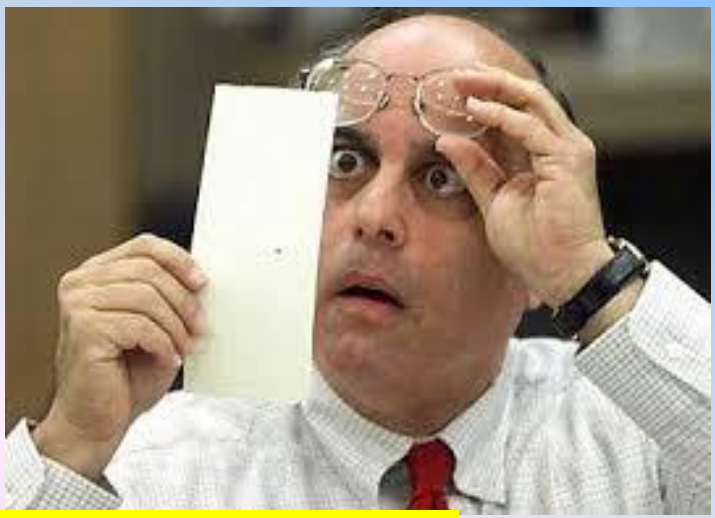
Основные **проблемы**, которые возникают **при параллельном выполнении транзакций**, делятся условно на 4 типа:

- Проблемы пропавших обновлений - **Потеря изменений** - может происходить при **одновременном обновлении** двумя и более транзакциями одного и того же набора данных.

Транзакция, **закончившаяся последней,** перезапишет результаты изменений, внесённых предыдущими транзакциями, и они будут потеряны.

- Проблемы промежуточных данных - **Ситуация чернового чтения** - возникает, когда транзакция считывает изменения, вносимые другой (незавершенной) транзакцией. Если эта вторая транзакция не будет зафиксирована, то данные, полученные в результате чернового чтения, будут некорректными.





Операторы продали 50 мониторов из наличествующих 40 штук, и на складе еще числится 10 подобных мониторов.

Проблема пропавших обновлений

Основные **проблемы**, которые возникают **при параллельном выполнении транзакций**, делятся условно на 4 типа: *(продолжение)*

- Проблемы несогласованных данных - **Неповторяемое чтение** - является противоположностью повторяемого, т.е. транзакция "видит" изменения, внесённые другими (незавершёнными!) транзакциями и многократно использует их на разных этапах запроса (при повторном чтении – новый результат).

Следствием -**несогласованность результатов запроса**, когда часть данных запроса соответствует состоянию БД до внесения изменений, а часть – состоянию БД после внесения другими транзакциями до фиксации изменений.

- Проблемы строк-призраков (**строк-фантомов**). - **Фантомы** – это особый тип неповторяемого чтения. Одна и та же транзакция сначала производит обновление набора данных, а затем считывание этого же набора.

Если считывание данных начинается раньше, чем закончится их обновление, то в результате чтения можно получить несогласованный (не обновлённый или частично обновлённый) набор данных или появятся лишние строки-призраки.

Для того чтобы избежать подобных проблем, требуется выработать некоторую процедуру **согласованного выполнения параллельных транзакций**.

Такая процедура называется **сериализацией транзакций**.

Эта процедура должна удовлетворять следующим правилам:

- В ходе выполнения транзакции пользователь видит только согласованные данные. Пользователь не должен видеть несогласованных промежуточных данных.
- Когда в БД две транзакции выполняются параллельно, то СУБД гарантированно поддерживает **принцип независимого выполнения** транзакций, который гласит, что:
  - результаты выполнения транзакций будут такими же, как если бы вначале выполнялась транзакция 1, а потом транзакция 2,
  - или наоборот, сначала транзакция 2, а потом транзакция 1.

Наиболее распространенным механизмом, который используется коммерческими СУБД **для реализации на практике сериализации транзакций** является **механизм блокировок**.

Рассматривают **два типа блокировок** (синхронизационных захватов):

- совместный режим блокировки — **нежесткая, или разделяемая, блокировка**, обозначаемая как **S (Shared)**.

- монопольный режим блокировки — **жесткая**, или **эксклюзивная, блокировка**, обозначаемая как **X (eXclusive)**.

Блокировки, называемые также синхронизационными захватами объектов, **могут быть применены к разному типу объектов**:

- вся БД (блокировки SD / XD);
- отдельные таблицы (блокировки ST / XT);
- блокировка на уровне страниц (даже для одной таблицы) - (блокировки SP / XP);
- в некоторых СУБД возможна блокировка на уровне строк - (блокировки SR / XR).

- 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**.

Правила совместимости захватов одного объекта разными транзакциями представлены в таблице:

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