Дисциплина
«Базы и банки данных»

Маркова Ирина Васильевна,
начальник управления информатизации
markova@miit.ru

Транзакция (определение)

Транзакция - последовательность операции над БД, рассматриваемая СУБД как логическая единица работы и обладающая свойствами:

- неразрывность (atomicity);
- согласованность (consistency);
- изолированность (isolation);
- устойчивость (durability).

Неразрывность предполагает атомарность (неделимость) действий (т.е. «все или ничего»).

Согласованность гарантирует, что транзакция переводит базу данных из одного согласованного состояния в другое, т. е. транзакция обеспечивает непротиворечивость базы данных при изменениях (ошибки логики, ошибки данных и т.д.).

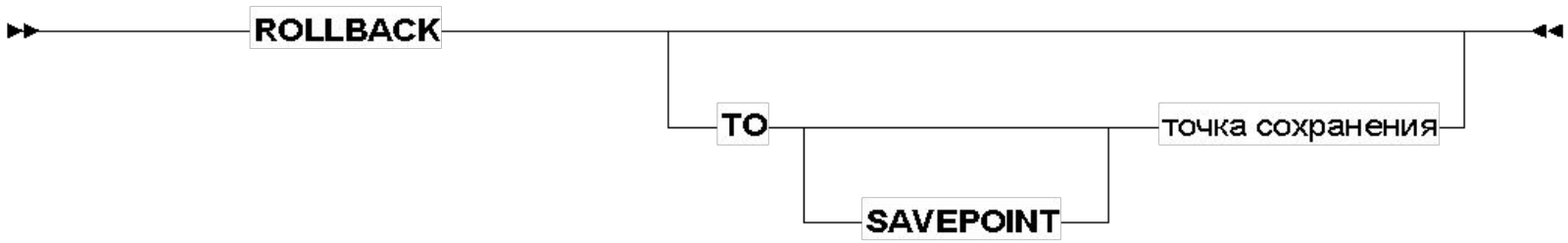
Изолированность означает, что транзакции выполняются независимо друг от друга, т.е. промежуточные результаты незавершённой транзакции не должны быть доступны другой транзакции.

Устойчивость предполагает, что результат успешно завершенной транзакции должен храниться в базе данных постоянно и не должен потеряться в результате последующих сбоев.

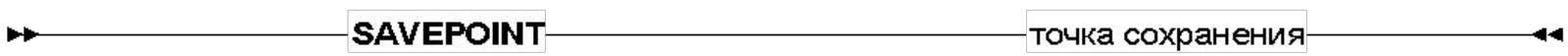
Команды управления транзакциями

Команда фиксации транзакции

Команда отката транзакции



Команда установки точки сохранения



Модели транзакций

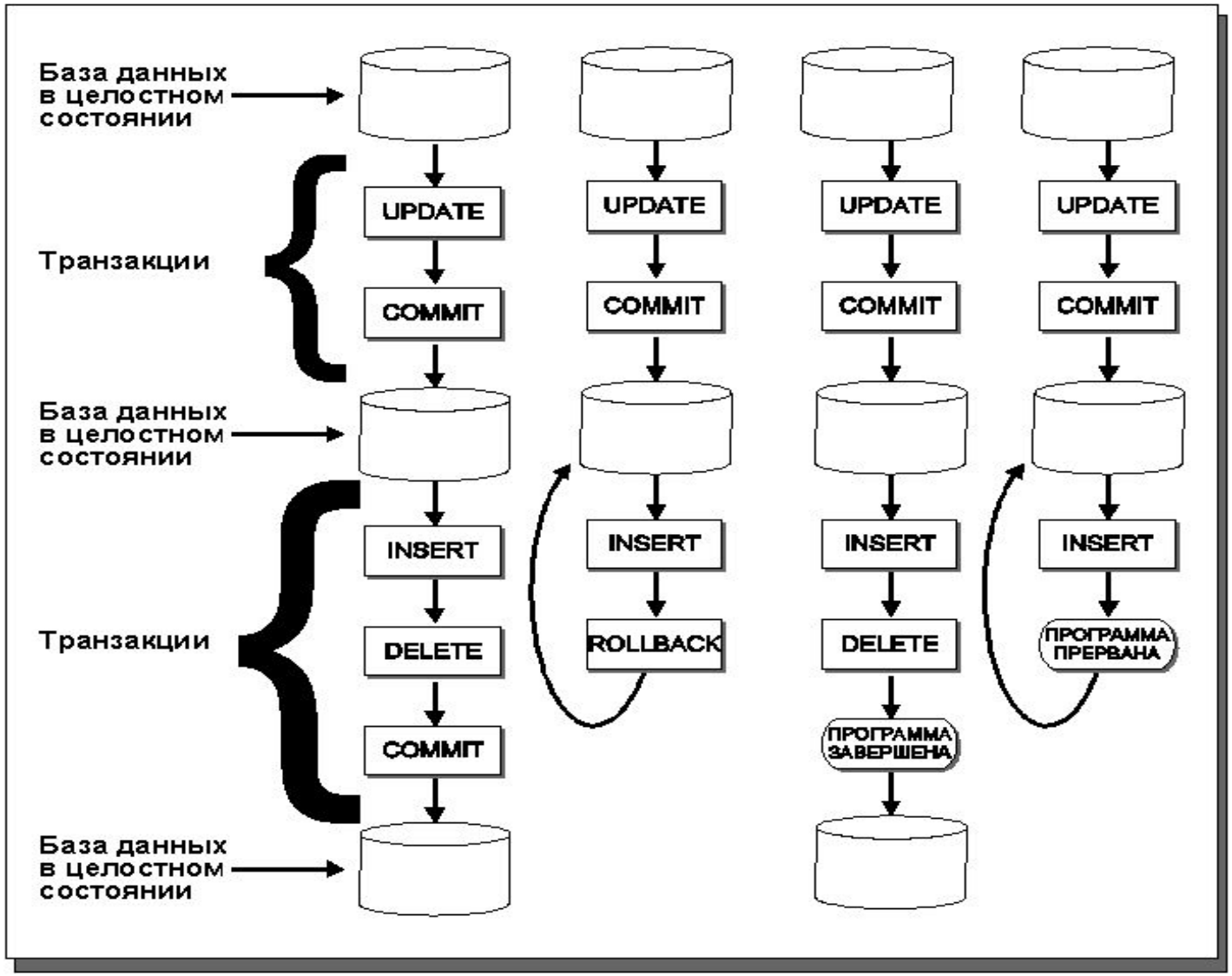
Стандарт ANSI/ISO определена модель транзакций, согласно, которой транзакция начинается

Автоматически с выполнения пользователем или программой первого оператора SQL.

Последовательное выполнение остальных операторов SQL происходит до тех пор, пока транзакция не завершится одним из следующих 4-х способов:

- COMMIT завершает текущую транзакцию (произведенные изменения становятся постоянными, новая транзакция начинается непосредственно после COMMIT);
- ROOLLBACK отменяет текущую транзакцию (произведенные изменения отменяются, новая транзакция начинается непосредственно после ROOLLBACK);
- успешное выполнение программы (аналог COMMIT) завершает текущую транзакцию (новая транзакция не начинается);
- неуспешное выполнение программы (аналог ROLLBACK) отменяет текущую транзакцию (новая транзакция не начинается).

Модели транзакций (продолжение)



Модели транзакций (продолжение)

Особенности модели транзакций ANSI/ISO:

- для начала транзакции не требуется выполнение специальных действий;
- транзакция начинается автоматически вместе с первым оператором SQL или непосредственно после окончания предыдущей транзакции.

Например, DB\2.

Модели транзакций (продолжение)

Модель транзакций, отличная от предлагаемой ANSI/ISO, использует 4 оператора:

- BEGIN TRANSACTION - сообщает о начале транзакции;
- COMMIT TRANSACTION - сообщает об успешном завершении транзакции;
- SAVE TRANSACTION - создает внутри транзакции точку сохранения (SAVEPOINT);
- ROLLBACK - либо отменяет изменения после SAVEPOINT, возвращая транзакцию к месту, где был выполнен SAVE TRANSACTION, либо отменяет все изменения после BEGIN TRANCTION.

Модели транзакций (продолжение)

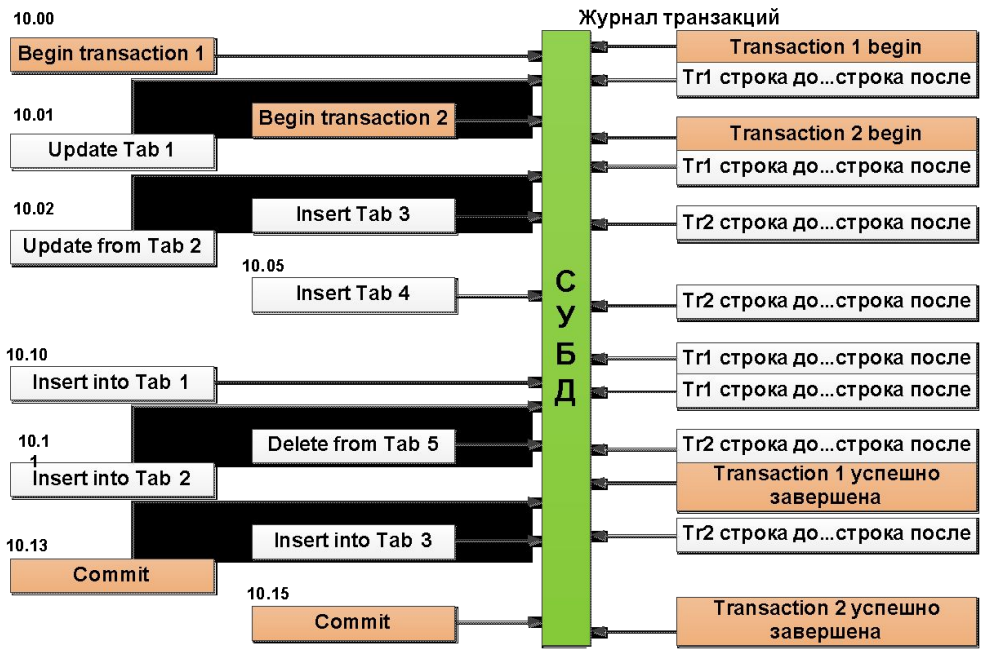
Особенности модели транзакций типа Sybase:

- возможность использовать при очень сложной логике в длинных транзакциях;
- возможность разделить сложную транзакцию на более простые части;
- операторы SQL, которые выполняются вне транзакции, обрабатываются в режиме «автовыполнения» (отменить успешно выполненный оператор невозможно);
- запрещает использование в транзакциях команд изменения структуры.

Например, SQLServer, Sybase и др.

Журнал транзакций

Журнал транзакций – это специальным образом организованный последовательный файл, в котором фиксируются все изменения, выполняемые всеми транзакциями.



Общие принципы восстановления:

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

Журнал транзакций и контрольные точки

Контрольная точка является моментом синхронизации между БД во вторичной памяти и журналом транзакций.

Контрольная точка образуется через определенный интервал времени и предусматривает:

- перенос всех имеющихся в оперативной памяти записей журнала во вторичную память;
- запись всех модифицированных блоков из буферов базы данных во вторичную память;
- помещение в файл журнала транзакций записи контрольной точки (эта запись содержит идентификаторы всех активных транзакций в момент контрольной точки).

Журнал транзакций и восстановление данных

В случае системного сбоя с помощью специальной утилиты восстановления можно просмотреть журнал транзакций, отыскать незавершенные транзакции и отменить их.

Особенности использования журнала транзакций:

- использование журнала транзакций увеличивает время на изменение данных и может уменьшать производительность;
- в промышленных базах данных журнал транзакций необходим для восстановления данных и откатов транзакций.
- с целью снижения нагрузки на жёсткий диск при многопользовательском режиме работы с оперативными данными, журнальные файлы хранятся отдельно на высокоскоростном диске.

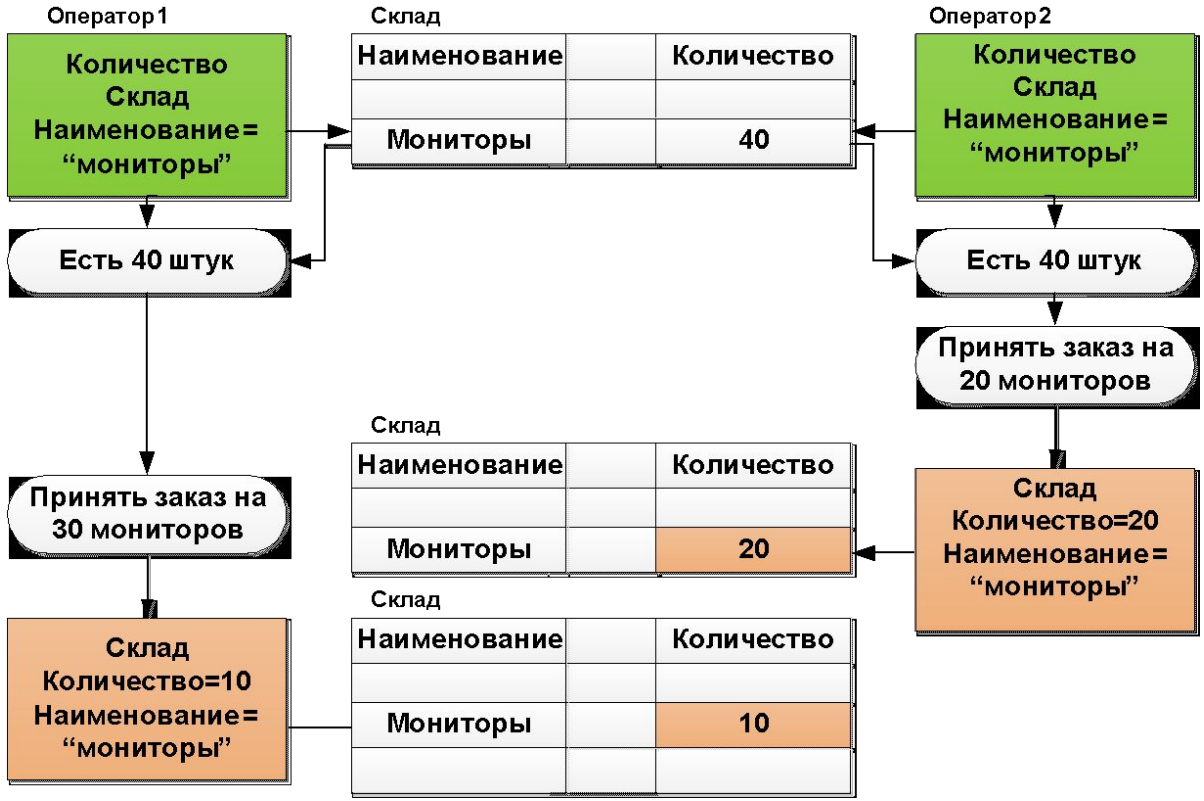
Работа в многопользовательском режиме

Работа в многопользовательском – это высокий уровень изолированности транзакций.

Основные проблемы, возникающие при неправильной обработке параллельных транзакций:

- проблема пропавшего обновления;
- «грязное» чтение;
- проблема несогласованных данных;
- возникновение «строк-призраков».

Проблема пропавшего обновления



Результат:

- заказы приняты;
- продано больше, чем имеется;
- есть остаток на складе, что не соответствует действительности;

Вывод: результаты выполнения одной успешной транзакции могут быть перекрыты результатом выполнения другой успешной транзакции.

Проблема промежуточных данных («грязное» чтение)

Результат:

- отказ в заказе при достаточном количестве на складе;
- заказ лишних изделий;
- возможно восстановление количества изделий на складе, хотя часть их может быть продана другим оператором.

Вывод: одна транзакция получила доступ к промежуточным результатам выполнения другой транзакции до того, как они были зафиксированы.

Проблема несогласованных данных и «строки-призраки»

Результат:

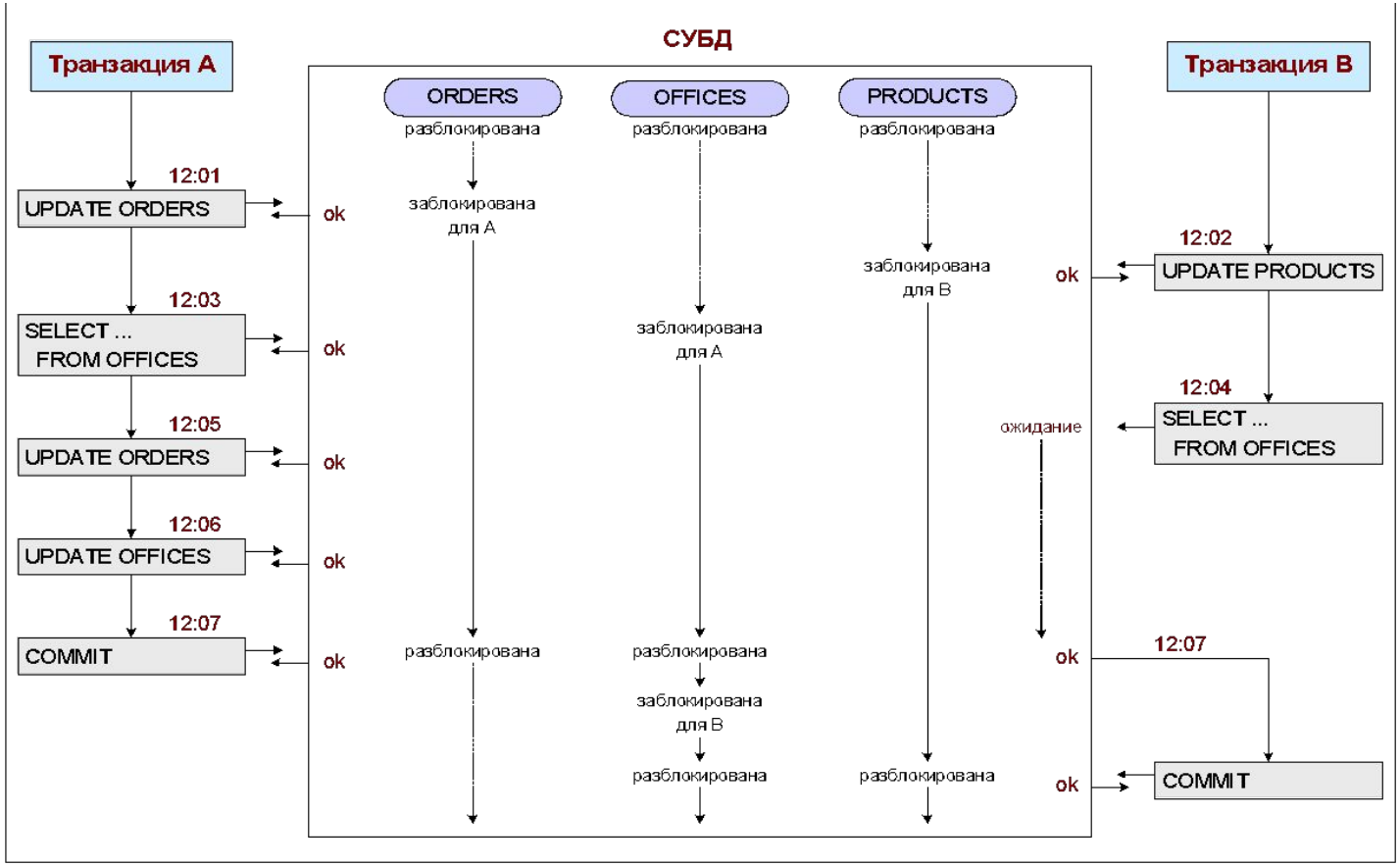
- отказ в заказе при достаточном количестве на складе.

Вывод: одна транзакция изменяет данные, которые уже прочитала вторая транзакция (один запрос, выполненный во время выполнения одной транзакции возвращает разный результат).

Блокировки

Основная идея: предотвратить непредсказуемое изменение объекта, пока транзакция выполняет необходимую обработку этого объекта.

Цель– предотвратить конфликты между транзакциями, обеспечивая при этом максимальную степень параллелизма при доступе к базе данных и минимальные затраты на реализацию механизма блокировки.



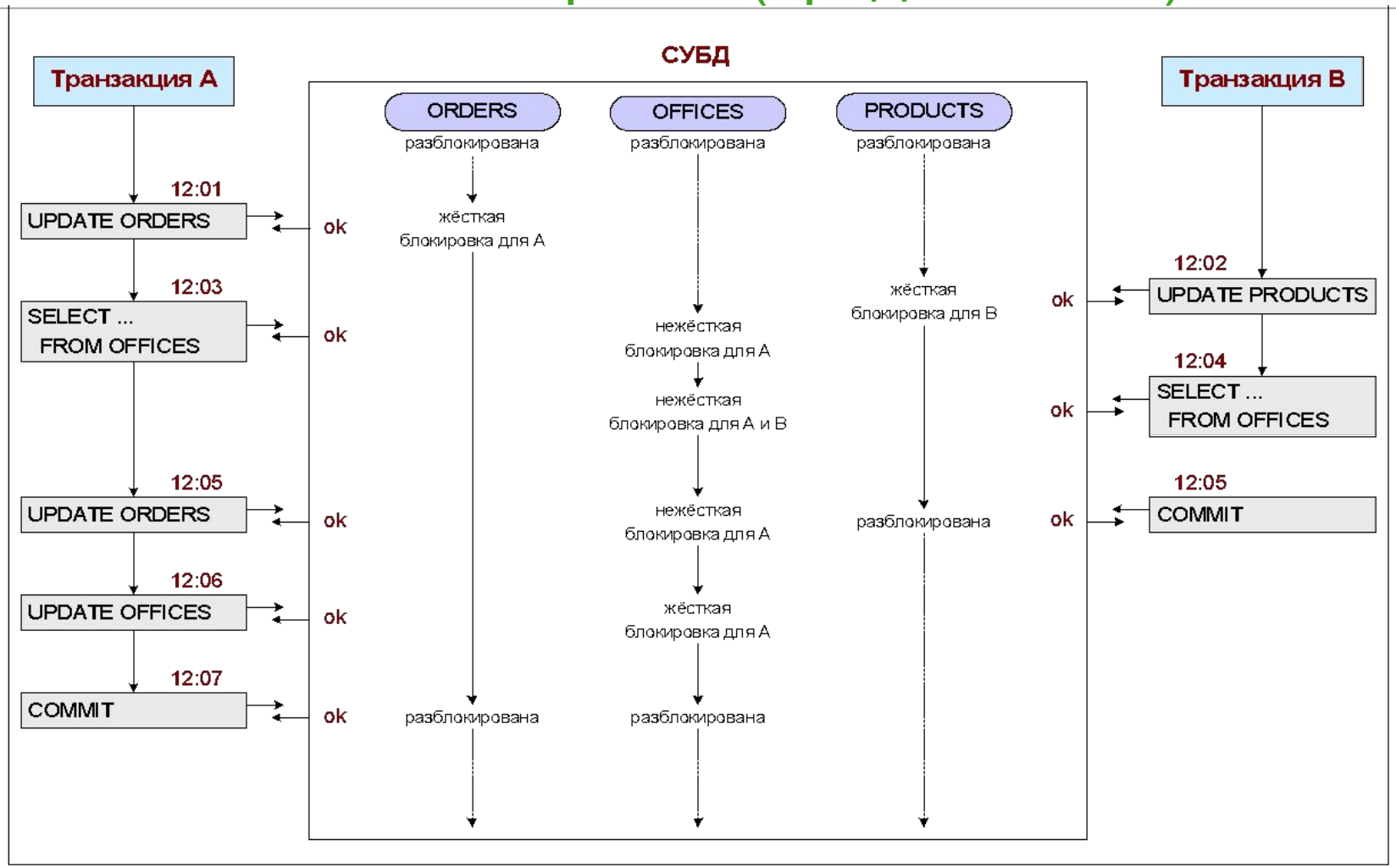
Уровни блокировки

- на уровне базы данных;
- на уровне таблицы;
- на уровне блока;
- на уровне строки.

Типы блокировок

- X-блокировка (eXclusive Lock) – жёсткая блокировка;
- S-блокировка (Shared Lock) – нежёсткая блокировка;
- IX-блокировка (Intent Shared Lock) – преднамеренная жёсткая блокировка;
- IS-блокировка (Intent eXclusive Lock) – преднамеренная нежёсткая блокировка;
- SIX-блокировка (Shared Intent eXclusive Lock).

Типы блокировок (продолжение)



Протокол двухфазной блокировки

1. Перед выполнение каких-либо операций с некоторым объектом, транзакция должна заблокировать этот объект.

2. После снятия блокировки, транзакция не должна накладывать никаких других блокировок. Транзакции, используемые в этом протоколе, не различаются по типам и считаются монопольными.

Протоколы доступа к данным с использованием S- и X-блокировок и протокол преднамеренных блокировок являются модификациями протокола двухфазной блокировки для случая, когда блокировки имеют различные типы.

Монопольное (жёсткое) и разделяемое (нежёсткое) блокирование

- **X-блокировка** (eXclusive lock): транзакция T не допускает для других транзакций параллельные запросы к отношению R, при этом сама транзакция T может изменять любые кортежи отношения R. Такую блокировку иногда называют блокировкой без взаимного доступа (монопольной блокировкой), жёсткой блокировкой или блокировкой записи.

- **S-блокировка** (Shared lock): транзакция T допускает для других транзакций параллельные считывания (но не обновления) для отношения R, при этом сама транзакция T также не может обновлять любые кортежи отношения R. Такую блокировку иногда называют блокировкой со взаимным доступом (общей блокировкой), нежёсткой блокировкой или блокировкой чтения.

Преднамеренные блокировки

- **IX-блокировка** (Intent eXclusive locks): транзакция T накладывает X-блокировки на отдельные кортежи отношения R, чтобы гарантировать их стабильность при обработке. Такие блокировки называют также предупреждающие (преднамеренные) блокировки без возможности взаимного доступа.

- **IS-блокировка** (Intent Shared lock): транзакция T накладывает S-блокировки на отдельные кортежи отношения R, чтобы гарантировать их стабильность при обработке. Такие блокировки называют также предупреждающие (преднамеренные) блокировки с возможностью взаимного доступа.

- **SIX-блокировка** (Shared Intent eXclusive locks): транзакция T допускает для других транзакций параллельные считывания (но не обновления) для отношения R, при этом сама транзакция T может обновлять отдельные картежи отношения R.

Правила протокола предупреждающего блокирования

1. чтобы затребовать S-блокировку или X-блокировку любого элемента, следует обратиться к корневой вершине иерархии элементов;

2. если элемент, подлежащий блокированию, достигнут, завершить просмотр иерархии и послать запрос на S- или X-блокировку;

3. если требуемый элемент расположен ниже по дереву иерархии, следует пометить текущую вершину предупреждающей блокировкой – IS- или IX-блокировкой соответственно;

4. как только право на блокирование текущего элемента получено, следует продолжить движение по соответствующим ветвям дерева вплоть до достижения нужного вложенного элемента, руководствуясь правилами пп. 2 и 3.

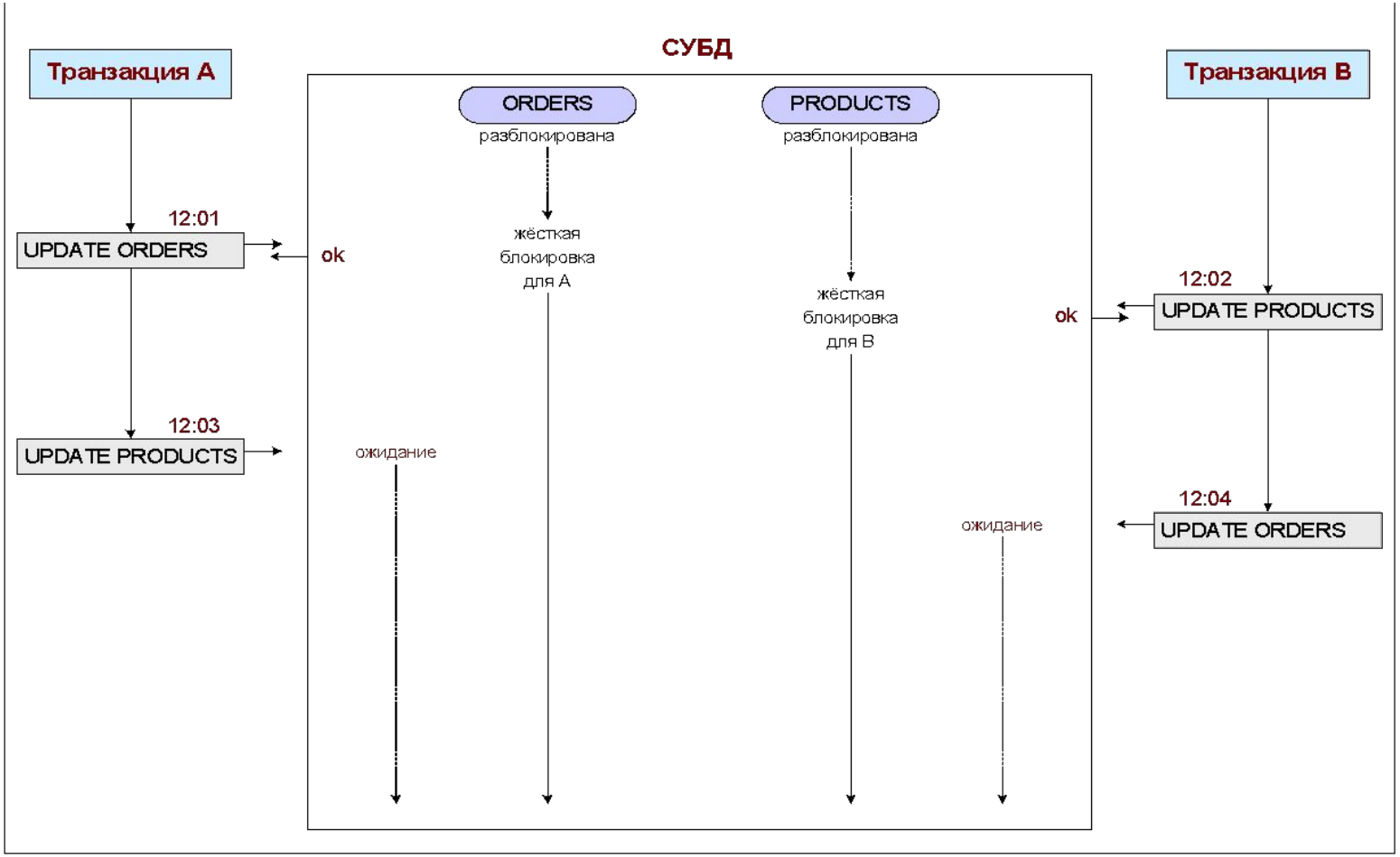
Матрица совместимости блокирования

	Транзакция T2 пытается наложить на таблицу блокировку:				
Транзакция T1 наложила на таблицу блокировку:	IS	S	IX	SIX	X
IS	да	да	да	да	нет
S	да	да	нет	нет	нет
IX	да	нет	да	нет	нет
SIX	да	нет	нет	нет	нет
X	нет	нет	нет	нет	нет

Тупиковые ситуации

Тупик – цикл блокировок, когда каждая транзакция ожидает освобождения данных, заблокированных другой транзакцией.

Без внешнего вмешательства ожидание может быть бесконечным.



Работа транзакций в смеси

Элементарные операции различных транзакций могут выполняться в произвольной очередности. При этом внутри каждой транзакции последовательность элементарных операций этой транзакции является строго определенной.

Например, если есть несколько транзакций T, Q, S, состоящих из последовательности операций элементарных:

$$T = \{T1, T2, T3, \ldots, Tn\},$$
$$Q = \{Q1, Q2, Q3, \ldots, Qm\},$$
$$S = \{ S1, S2, S3, \ldots, Sl \},$$

то реальная последовательность, в которой СУБД выполняет эти транзакции, может быть, например, такой {T1, Q1, T2, S1, T3, S2, S3, Q2,…}.

Работа транзакций в смеси (определения)

Определение 1. Набор из нескольких транзакций, элементарные операции которых чередуются друг с другом, называется смесью транзакций.

Определение 2. Последовательность, в которой выполняются элементарные операции заданного набора транзакций, называется графиком запуска набора транзакций.

Определение 3. График запуска набора транзакций называется последовательным, если транзакции выполняются строго по очереди, т.е. элементарные операции транзакций не чередуются друг с другом.

Определение 4. Если график запуска набора транзакций содержит чередующиеся элементарные операции транзакций, то такой график называется чередующимся.

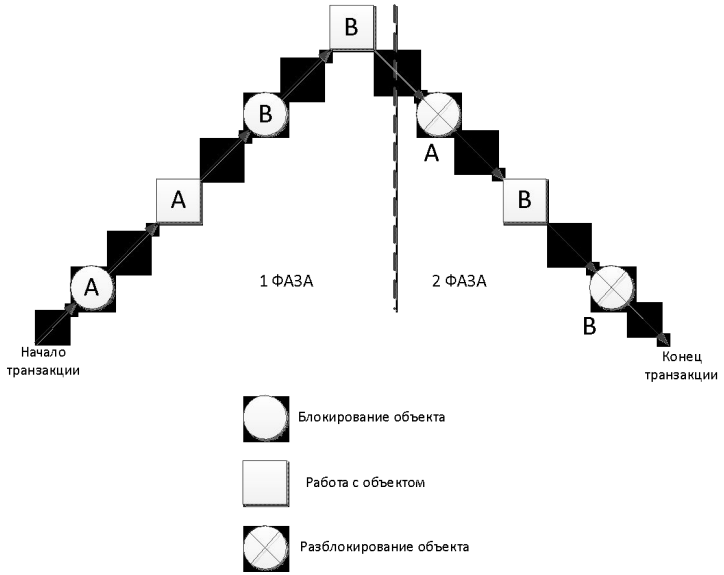
Определение 5. Два графика называются эквивалентными, если при их выполнении будет получен один и тот же результат, независимо от начального состояния базы данных.

Определение 6. График запуска транзакции называется верным (или сериализуемым), если он эквивалентен какому-либо последовательному графику.

Теорема Эсварана

Теорема: если все транзакции в смеси подчиняются протоколу двухфазной блокировки, то для всех возможных чередующихся графиков запуска существует возможность упорядочения.

Работа транзакции по протоколу двухфазной блокировки может выглядеть следующим образом:

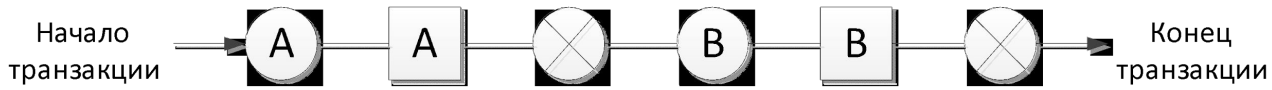


- 1 фаза – нарастание блокировок: во время этой фазы накладываются блокировки, и производится работа с заблокированными объектами;
- 2 фаза – снятие блокировок: во время этой фазы блокировки только снимаются (работа с ранее заблокированными данными может продолжаться).

Вторая фаза – это, как правило, одна операция завершения (или отката) транзакции с одновременным снятием всех блокировок.

Транзакции, не подчиняющиеся протоколу двухфазной блокировки

Пример:



Если некоторая транзакция A не подчиняется протоколу двухфазной блокировки (и, следовательно, состоит не менее, чем из двух операций блокирования и разблокирования), то всегда можно построить другую транзакцию B, которая при чередующемся выполнении вместе с A приводит к графику, не подлежащему упорядочению и неверному.

Защёлки

Защёлка – особый вид блокировки, которая устанавливается на гораздо менее продолжительный период времени, чем обычные блокировки.

Защёлка может быть применена перед выполнением операции чтения или записи страницы
памяти на диск для обеспечения неразрывности операции ввода-вывода. Поскольку защёлки служат только для предотвращения конфликта операций, требующих доступа такого рода,
обычно они не применяются для реализации протокола управления параллельным выполнением наподобие протокола двухфазной блокировки.

Определение прав доступа

Команды определения прав доступа также относятся к языку управления данными (DCL).

Определение прав доступа регламентируется набором разрешённых привилегий:

- *Объектные привилегии* – разрешённые действия над определёнными объектами базы данных для конкретного пользователя. К объектным привилегиям относятся SELECT, INSERT, UPDATE, DELETE, REFERENCES. Права на объекты предоставляют их создатели.
- *Системные привилегии* – привилегии, которые не определяются в терминах отдельных объектов, а относятся к системе в целом, включая право создавать объекты, различать базовые таблицы и представления.

К системным привилегиям относятся CONNECT, RESOURCE и DBA.

- CONNECT: предусматривает пользователю право входить в систему и создавать представления и синонимы (альтернативные имена таблиц).
- RESOURCE: предоставляет пользователю право создавать базовые таблицы,
- DBA: позволяет пользователю распоряжаться базой данных без ограничений на функции.

Назначение привилегий

Назначение объектных привилегий происходит с помощью следующей конструкции:

```
GRANT привилегия ON схема.объект TO пользователь [WITH GRANT
OPTION];
```

Например:

```
GRANT SELECT ON HR.EMPLOYEES TO ALL_ORACLE;
GRANT UPDATE (SALARY) ON HR.EMPLOYEES TO ALL_ORACLE;
GRANT ALL ON HR.REGIONS TO ALL_ORACLE;
```

Отмена привилегий

Отмена привилегий осуществляется каскадно, т.е. привилегия отменяется последовательно для каждого пользователя, получившего привилегию от того, кто их лишается в данный момент:

```
REVOKE [GRANT OPTION FOR] привилегия ON схема.объект FROM
пользователь [CASCADE|RESTRICT];
```

Например:

```
REVOKE INSERT, DELETE ON bonus FROM stud;
```