

Министерство образования и науки Российской Федерации

Калужский филиал  
федерального государственного бюджетного образовательного  
учреждения высшего образования  
**«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»**  
(КФ МГТУ им. Н.Э. Баумана)

**С.А. Глебов**

**ВЗАИМОДЕЙСТВИЕ ТРАНЗАКЦИЙ**  
Методические указания по выполнению лабораторной работы  
по курсу «Базы данных»

Калуга – 2018

УДК 004.65  
ББК 32.972.134  
Г53

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий и прикладной математики».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий и прикладной математики» (ФН1-КФ) протокол № 7 от « 21 » февраля 2018 г.


И.о. зав. кафедрой ФН1-КФ  к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ФНК протокол № 2 от « 28 » 02 2018 г.

Председатель методической комиссии факультета ФНК  к.х.н., доцент К.Л. Анфилов

- Методической комиссией КФ МГТУ им.Н.Э. Баумана протокол № 2 от « 06 » 03 2018 г.

Председатель методической комиссии КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:  
к.т.н., доцент кафедры ЭИУ6-КФ

 А.Б. Лачихина

Авторы  
к.ф.-м.н., доцент кафедры ФН1-КФ

 С.А. Глебов

#### Аннотация

Методические указания по выполнению лабораторной работы по курсу «Базы данных» содержат руководство по настройке транзакций баз данных и задание на выполнение лабораторной работы.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

© Калужский филиал МГТУ им. Н.Э. Баумана, 2018 г.  
© С.А. Глебов, 2018 г.

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
ТРАНЗАКЦИИ.....	6
УРОВНИ ИЗОЛЯЦИИ ТРАНЗАКЦИЙ .....	10
ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ .....	15
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ .....	15
ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ .....	15
ОСНОВНАЯ ЛИТЕРАТУРА.....	16
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА .....	16

## **ВВЕДЕНИЕ**

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Базы данных» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат руководство по настройке транзакций баз данных.

## **ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ**

Целью выполнения лабораторной работы является сформировать практические навыки практические навыки анализа и выявления путей бесконфликтного взаимодействия транзакций.

Основными задачами выполнения лабораторной работы являются:

- создать приложение, использующее явный механизм установки параметров, старта, подтверждения и отката транзакций.
- создать конфликт двух транзакций, сделать выводы о причинах возникновения конфликта и способах его недопущения

Результатами работы являются:

- Приложение, использующее явный механизм установки параметров, старта, подтверждения и отката транзакций
- Подготовленный отчет.

## ТРАНЗАКЦИИ

Любая современная СУБД предназначена для работы в многопользовательском режиме. При этом очевидны конфликтные ситуации между пользователями и СУБД должна корректно из этих ситуаций выходить. Это выполняется при помощи механизма транзакций.

Под транзакцией понимается неделимая с точки зрения воздействия на БД логическая последовательность операторов манипулирования данными (чтения|SELECT, удаления|DELETE, вставки|INSERT, модификации|UPDATE) такая что возможны два итога:

- результаты всех операторов, входящих в транзакцию, соответствующим образом отображаются в БД;
- воздействие всех этих операторов полностью отсутствует.

Классическим примером транзакции, переводящей БД из одного целостного состояния в другое является бухгалтерская проводка, когда некоторая сумма S должна быть списана со счета А и занесена на счет В. Только успешное выполнение этих двух операций гарантирует целостность информации в БД. Если сумма S снимется со счета А, затем произойдет сбой, то сумма не будет записана на счет В. Поэтому, в этом случае, результаты первой операции должны быть отменены.

Существуют некоторые свойства, которыми должна обладать любая из транзакции. Ниже представлены четыре основных свойства (ACID — аббревиатура, составленная из первых букв их английских названий).

- **Атомарность (atomicity).** Это свойство типа "все или ничего". Любая транзакция представляет собой неделимую единицу работы, которая может быть либо выполнена вся целиком, либо не выполнена вовсе.
- **Согласованность (consistency).** Каждая транзакция должна переводить базу данных из одного согласованного состояния в другое согласованное состояние.

- **Изолированность (isolation).** Все транзакции выполняются независимо одна от другой и промежуточные результаты незавершенной транзакции не доступны другим транзакциям.
- **Продолжительность (durability).** Результаты успешно завершенной (зафиксированной) транзакции должны сохраняться в базе данных постоянно и не должны быть утеряны в результате последующих сбоев.

Все операции в БД осуществляются в рамках транзакции, как минимум одной. Транзакция начитается оператором `start transaction` и заканчивается либо подтверждением (`commit`), либо откатом (`rollback`). При этом решение о том или ином исходе никак не связано с успешностью или неуспешностью операций внутри транзакции. Ничто не может помешать пользователю откатить результаты успешных действий исходя из своих высших соображений и наоборот, подтвердить результаты неуспешных. Согласно стандарту оператор `start transaction` может быть выполнен автоматически при поступлении первого SQL-оператора, способного инициировать транзакцию

Даже если разработчик клиента БД и не использует транзакции явно, это лишь означает, что работу по управлению транзакциями берет на себя инструмент разработки. В простейшем случае транзакция запускается одновременно с запуском приложения-клента и подтверждается с его завершением. Т.о. все операции, выполняемые приложением считаются выполняющимися в рамках одной транзакции.. Поэтому, при аварийном завершении программы (потеря соединения с сервером БД) происходил откат транзакции и потеря всех изменений. Поэтому грамотная настройка механизма транзакций — залог стабильной работы и сохранения всех изменений.

Определить какие именно операции должны входить в состав транзакции, т.е. определить моменты начала и окончания транзакции в состоянии только само приложение-клиент или пользователь этого приложения исходя из бизнес-логики (см. пример).

Понятие транзакции имеет непосредственную связь с понятием целостности БД. Очень часто БД может обладать такими ограничениями целостности, которые просто невозможно не нарушить, выполняя изменение только одним оператором.

Для поддержания подобных ограничений целостности допускается их нарушение внутри транзакции с тем условием, чтобы к моменту завершения транзакции условия целостности должны быть соблюдены. Но, здесь вступает в силу свойство изолированности транзакций, говорящее, что изменения внутри транзакции (т.е. до их подтверждения) не видны в других транзакциях.

Реализация механизма транзакций различна. В INTERBASE реализована многоверсионная архитектура (MGA — Multi Generation Architecture). Суть ее в том, что любые действия над записью производятся не над самой записью, а над ее копией (версией). Когда транзакция изменяет запись, для нее создается копия над которой и совершаются изменения в рамках этой транзакции. При этом изменять данные может только одна транзакция, остальные могут только читать. Когда изменяющая транзакция подтверждается, исходная версия записи помечается для удаления, а текущая версия становится основной.

Каждая транзакция при запуске получает уникальный идентификатор (transaction ID | TID), при помощи которого все транзакции учитываются на странице учета транзакций (Transaction Inventory Page | TIP). Таким образом, для каждой версии записи указывается какой транзакции она принадлежит, а также указатель на следующую версию той же записи.

Транзакция может быть в одном из четырех состояний: активная (active), подтвержденная (committed), отмененная (rolledback) и с неопределенным статусом (limbo /при двухфазном подтверждении/).

Когда, транзакция committed, то в БД не происходит никаких изменений, а лишь изменяется состояние транзакции с активной на подтвержденную на TIP. Поэтому читающие транзакции, прежде чем прочесть запись, должны пройти по ее существующим версиям и взять из подтвержденных версий ту, TID которой больше (т.е. последнюю).

Если же транзакция rolledback то, ее версия записи помечается для удаления и считается мусором (garbage) и подлежат удалению, но удаляются они не той транзакцией, которая породила мусор, а следующей за ней, которая ищет последнюю подтвержденную версию



записи и во время поиска удаляет все старые версии записи. Этот процесс называется "сборкой мусора" и является кооперативным, т.е. им занимаются все транзакции, в т.ч. и читающие.

При этом если, транзакция не выполнила никаких изменений (по флагу Update Flag) и завершается отменой, то вместо rollback вызывается commit, чтобы не нагружать сервер.

Когда транзакция удаляет запись, запись лишь помечается для удаления и после подтверждения транзакции станет мусором.

Таким образом, мусором являются:

- старые версии записей, когда подтвердилась изменяющая транзакция и сделала свою версию основной;
- отмененные изменения, когда изменяющая или добавляющая транзакция откатилась, оставив свою версию записи;
- удаленные записи.

Транзакция может быть запущена в двух режимах: **чтения/записи** (write) или только чтения (read). Первый режим устанавливается по умолчанию. Транзакции, которые не изменяют данных и запущены в режиме чтения меньше нагружают сервер, т.к. не создают лишних версий записей.

Существует такое понятие как конфликт. Н-р, когда две транзакции пытаются изменить одну и ту же запись. Поэтому транзакция, которая обратилась к записи первой, блокирует ее. Тогда вторая транзакция должна будет либо немедленно возбудить исключение, либо подождать некоторое время: вдруг первая транзакция завершится и разблокирует доступ к записи. Отсюда вытекают **два режима блокировки**: wait (по умолчанию) и nowait.

Как говорилось выше, транзакции обладают свойством изолированности, т.е. происходящие в рамках транзакции изменения не видны из вне (т.е. другими транзакциями) до подтверждения транзакции.

## УРОВНИ ИЗОЛЯЦИИ ТРАНЗАКЦИЙ

Уровень изоляции транзакции определяет какие изменения, сделанные в других транзакциях будут видны в данной транзакции.

Установка параметров транзакций (уровня изоляции, режима записи, режима блокировки) осуществляется при помощи перечисления набора констант:

Параметры	Константа	Описание
<b>Уровень изоляции</b>	read_committed rec_version	Возможность читать подтвержденные записи других транзакций, в том числе и те которые заблокированы (разумеется только последнюю подтвержденную версию)
	read_committed no_rec_version	Также видит все подтвержденные изменения, сделанные другими транзакциями, но НЕ может читать записи, имеющие неподтвержденные версии (т.е. заблокированные).
	concurrency	Видит состояние БД только на момент запуска. Не видит никаких изменений, сделанных другими транзакциями.
	consistency	Аналогично уровню concurrency, но таблица (целиком) блокируется на запись другими транзакциями.
<b>Режим доступа</b>	read	Транзакция может только читать
	write	Разрешает запись в рамках транзакции
<b>Режим блокировки</b>	wait	Отсроченное разрешение конфликтов
	nowait	Не ждать разрешения конфликта

В Interbase таких уровня три:

**1. READ COMMITTED** (читать подтвержденные данные).

Транзакция с данным уровнем изоляции видит все результаты всех подтвержденных транзакций. Этот уровень используется для

получения самого «свежего» состояния БД. Существует две разновидности такого чтения:

- **rec\_version**. Этот вариант используется по умолчанию и означает, что при чтении записи считывается последняя версия каждой записи независимо есть неподтвержденные версии или нет.
- **no\_rec\_version**. Этот вариант требует, чтобы на момент чтения записи у нее не существовало неподтвержденных версий. При чтении записи в такой транзакции производится проверка не существует ли у этой записи неподтвержденной версии. Если существует, то наша транзакция ждет, пока не завершится транзакция, изменяющая эту запись, при условии что наша запущена в режиме wait. Если же в nowait, то немедленно возникнет ошибка Deadlock.
- Очевидно, что такой уровень может привести к частым конфликтам, поэтому использовать его нужно с большой осторожностью.

**2. SNAPSHOT.** Задается параметром **concurrency**. Транзакция с данным уровнем изоляции не видит никаких изменений (кроме своих конечно), видит только состояние БД на момент своего запуска (как бы моментальный снимок БД). При попытке в этой транзакции изменить данные, измененные другими транзакциями уже после ее запуска возникает конфликт. Этот режим обычно используется для длительных по времени запросов, для генерации отчетов.

**3. SNAPSHOT TABLE STABILITY.** Задается параметром [consistency](#). Уровень изоляции почти такой же что и SNAPSHOT, но дополнительно блокирует таблицу на запись. Т.е. если транзакция с уровнем consistency производит изменения в таблице, то другие транзакции смогут только читать эту таблицу, а транзакции с уровнем consistency не смогут даже читать, т.к. этот уровень потребует блокировки таблицы, которая уже заблокирована. Обычно этот уровень используют только для коротких обновляющих транзакций, которые запускаются, проводят очень короткое по времени изменение и сразу завершаются. Другие транзакции в зависимости от режима блокировки wait|nowait либо ждут, либо возбуждают исключение.

**Практические аспекты использования** транзакций нужно решать для каждой задачи индивидуально. Обычно все запросы к БД можно разделить на:

- запросы на чтение самого «свежего» состояния БД;
- запросы на текущие изменения;
- запросы на чтение вспомогательных таблиц;
- запросы на чтение данных для построения отчетов и т.д.

Пусть, для рассматриваемого нами примера ФИЛЬМЫ, имеется форма с сеткой DBGrid, где просматривается таблица MOVIE.

Параметры транзакции по умолчанию будут: write, nowait, read\_committed rec\_version. В контексте этой транзакции пользователь имеет право изменять записи (write), но эти изменения не будут видны другим транзакциям (пользователям), до ее подтверждения. Более того, попытка других пользователей изменить измененные записи приведет к блокировке, т.к. транзакция не завершилась.

Наиболее оптимальным решением будет использование двух транзакций. Первая будет предназначена только для просмотра данных в сетке, поэтому параметры этой транзакции следует выбрать:

- read
- read\_committed
- rec\_version
- nowait

При таких параметрах производится чтение всех подтвержденных другими транзакциями записей причем без конфликтов в другими читающими или пишущими транзакциями. Такую транзакцию можно долго держать открытой — сервер не нагружается версиями записей.

Для изменения данных будет служить отдельная форма и отдельная транзакция, на которую вынесены компоненты для редактирования. Запрос на изменение одной записи должен быть очень коротким, чтобы свести к минимуму возможность конфликтов, которые можно отслеживать как исключения при помощи блока try... except. Параметры такой транзакции будут такими:

- write
- consistency

- nowait

Нам не нужно в рамках этой транзакции самых свежих данных, что только замедляет выполнение. Также мы сможем сразу узнать о конфликте, если запись заблокирована и запретить изменение/удаление записи другими транзакциями, в которых возникнет ошибка.

Уровень consistency приведет к тому, что во время такой короткой транзакции нельзя будет изменить другие записи (таблица заблокирована) и тогда разумным будет использование режима блокировки wait. Если использовать уровень изоляции concurrency, но, тогда конфликт будет обнаружен не при старте транзакции, а при выполнении оператора update,

Для запросов, которые строят отчеты (особенно для тех, которые делают несколько проходов по таблице) необходимо видеть только те данные, которые существовали на момент старта запроса и будет неизменными за все время его выполнения. Параметры:

- read
- concurrency
- nowait

Как уже говорилось, все действия в БД осуществляются в рамках транзакций, однако существуют объекты, которые находятся вне контекста транзакций. Это генераторы и внешние таблицы.

Генераторы располагаются на специально для них выделенной странице и все транзакции видят значения генераторов в любой момент времени. Это позволяет организовать бесконфликтные конкурентные вставки в параллельно выполняющихся транзакциях.

Внешние таблицы представляют собой файлы, находящиеся за пределами основного файла БД. Над ними разрешены только операции INSERT/SELECT. Отсутствие обновлений и позволяет отказаться от версионной структуры и, следовательно, от механизма транзакций.

Существует такое понятие как **двухфазное подтверждение транзакций**. Interbase позволяет организовать распределенные транзакции между разными БД и даже между разными серверами. Т.е. клиент может запустить транзакцию сразу на двух серверах. При этом

возможна ситуация, когда в рамках одной БД транзакция способна подтвердиться, а на другой — нет. Чтобы синхронизировать этот процесс вводится особое состояние Prepared, которое говорит о том, что транзакция готова подтвердиться, когда на другом сервере транзакция тоже будет в состоянии Prepared. Если же одна из транзакций откатится, то и транзакция из состояния Prepared также откатится.

Отсюда возникают транзакции limbo. Если между серверами разорвется соединение в тот момент, когда одна транзакция перешла в состояние Prepared и готова подтвердиться, то сервер не может решить подтвердить или откатить ее изменения этой транзакции, поэтому не следует использовать двухфазное подтверждение на медленных каналах связи.

## **ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

Для базы данных, созданной в прошлой лабораторной создать приложение, использующее явный механизм установки параметров, старта, подтверждения и отката транзакций. Запустить два таких приложения, создать конфликт двух транзакций, сделать выводы о причинах возникновения конфликта и способах его недопущения.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ**

1. Раскройте понятие «транзакция».
2. Перечислите основные свойства транзакций.
3. Раскройте сущность атомарности.
4. Раскройте сущность согласованности.
5. Раскройте сущность изолированности.
6. Раскройте сущность продолжительности.
7. Дайте определение конфликтной ситуации в базе данных.
8. Раскройте сущность уровня изолированности транзакций.
9. Перечислите уровни изоляции транзакций.
10. Опишите механизм синхронизации распределенных транзакций.

## **ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ**

На выполнение лабораторной работы отводится 3 занятия (6 академических часов: 5 часов на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), ход выполнения работы, результаты выполнения работы, выводы.

## ОСНОВНАЯ ЛИТЕРАТУРА

1. Карпова, Т.С. Базы данных: модели, разработка, реализация : учебное пособие / Т.С. Карпова. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 241 с. : ил. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=429003>
2. Давыдова, Е.М. Базы данных [Электронный ресурс] : учеб. пособие / Е.М. Давыдова, Н.А. Новгородова. — Электрон. дан. — Москва : ТУСУР, 2007. — 166 с. — Режим доступа: <https://e.lanbook.com/book/11636>. — Загл. с экрана.
3. Харрингтон, Д. Проектирование объектно ориентированных баз данных [Электронный ресурс] — Электрон. дан. — Москва : ДМК Пресс, 2007. — 272 с. — Режим доступа: <https://e.lanbook.com/book/1231>. — Загл. с экрана.

## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: учеб. пособие. – М.: Форум:Инфра-М, 2007.
5. Гагарин Ю.Е. Применение языка SQL в MS Access: учебно-методическое пособие. – М.: МГТУ им. Н.Э. Баумана, 2012.

### Электронные ресурсы:

1. Научная электронная библиотека <http://eLIBRARY.RU>
2. Электронно-библиотечная система <http://e.lanbook.com>
3. Электронно-библиотечная система «Университетская библиотека онлайн» <http://biblioclub.ru>
4. Электронно-библиотечная система IPRBook <http://www.iprbookshop.ru>