МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

А. В. Аксенов

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ. ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА SQL-ЗАПРОСОВ, УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ

Лабораторный практикум

ГУАП
Санкт-Петербург
2020

УДК
ББК
А

Рецензент:

Аксенов, А.В.

А Реляционные базы данных. Проектирование, разработка SQL-запросов, управление транзакциями: лаб. практикум / А. В. Аксенов. – СПб.: ГУАП, 2020. – с.

Содержит теоретические и практические сведения по проектированию структуры баз данных и работе с ними, а также сведения о порядке выполнения лабораторных работ

Предназначены для студентов дневной, очно-заочной и заочной форм обучения направления 09.03.01 «Информатика и вычислительная техника».

УДК
ББК

Лабораторная работа № 1.
ИНФОЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Цель работы

Освоение методов инфологического моделирования при проектировании схемы данных для реляционной базы данных.

1.1. Теоретические сведения

1.1.1. Общие понятия и принципы инфологического моделирования

Инфологическое моделирование – способ разработки структуры базы данных, который опирается на семантику (смысл) данных. Входными данными является словесное описание предметной области, полученное от экспертов, выходными – формализованная модель, как правило, представленная в виде ER-диаграммы.

ER-диаграмма (Entity-Relationship diagram, диаграмма Сущность-Связь) описывает предметную область как набор сущностей, семантически связанных между собой.

Следует различать понятия «сущность» и «экземпляр сущности».

Сущность – набор однотипных объектов или фактов, о которых требуется хранить какую-либо информацию. Иными словами, сущность – все, что можно представить списком.

Сущности желательно именовать существительными единственного числа.

Примеры:

«Товар» – множество всех товаров; «Клиент» – множество всех клиентов; «Покупка» – множество всех фактов покупки тем или иным покупателем того или иного товара.

Экземпляр сущности – один объект (факт) набора, один элемент списка.

Сущность определяет набор **атрибутов** – свойств, значения которых требуется хранить про каждый экземпляр.

Важнейшая черта атрибутов сущности – атомарность.

Атомарность данных (atomicity) – неделимость. Атомарные данные не представляют собой множества значений или списка значений. Атомарность данных – неоднозначная характеристика, и должна определяться с точки зрения семантики данных и предполагаемых методов работы с ними. Так, атрибут «Габариты» сущности «Товар» может быть классифицирован как атомарный, если при работе с моделью будет иметь смысл использовать габариты товара только в совокупности (например, выводить на экран в спецификации), и не придется рассматривать ширину, глубину и высоту товара по отдельности (например, искать товар с шириной не более 1,5 метров). В противном случае атрибут «Габариты» будет неатомарным и требовать пересмотра модели.

Рассмотрим способы борьбы с неатомарностью.

1) Атрибут может быть разбит на несколько атрибутов (в случае если атрибут представляет собой набор разных по смыслу значений, рис. 1.1).

2) Атрибут может быть преобразован в атомарный путем пересмотра того, что будет экземпляром сущности (в случае если атри-

Рис. 1.1. Удовлетворение критерия атомарности путем разбиения атрибута

Рис. 1.2. Удовлетворение критерия атомарности путем пересмотра семантики сущности

бут представляет собой список однотипных по смыслу значений, о которых не требуется хранить дополнительных сведений, и значение атрибута индивидуально у каждого экземпляра, рис. 1.2).

В примере на рис. 1.2 экземпляром сущности был человек, а атрибутами – его фамилия и номера телефонов. После преобразования экземпляром сущности стал факт того, что человек имеет тот или иной номер телефона, а атрибутами факта – фамилия человека и номер телефона.

3) Атрибут может быть вынесен в отдельную сущность (в случае если атрибут представляет собой список одинаковых по смыслу значений, о которых требуется хранить дополнительные сведения, или значение атрибута может повторяться у нескольких экземпляров). В этом случае вновь сформированная сущность будет связана с исходной (рис. 1.3).

Связи между сущностями проводятся в том случае, если экземпляры сущностей связаны друг с другом семантически. Экземпляры связей также несут информацию и устанавливают ассоциации между конкретными экземплярами сущностей. Сущность может быть связана сама с собой, в случае если между собой ассоциированы ее отдельные экземпляры.

Связи различаются по **кратности** (мощности): «один-к-одному», «один-ко-многим», «многие-ко-многим».

Тип связи определяется по отдельности на обоих ее концах. Чтобы определить тип связи на одном из концов, необходимо зафиксировать одиночный экземпляр одной из связанных сущностей

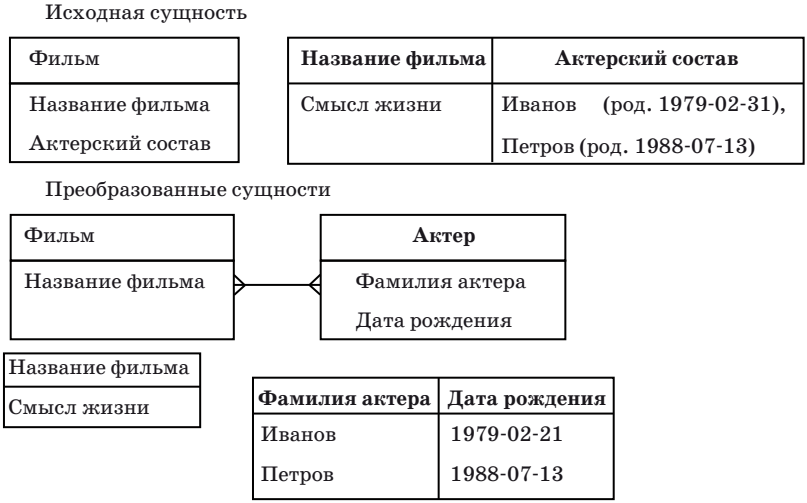

Рис. 1.3. Удовлетворение критерия атомарности путем создания новой сущности

и оценить, сколько экземпляров второй сущности с ним может быть логически связано: только один или несколько. Затем аналогичные действия производятся для другого конца связи.

Примеры

Сущность «Сеанс» связана с сущностью «Билет» связью «один-ко-многим» (на один сеанс продается много билетов; один билет действителен только на один сеанс). Сущность «Группа» связана с сущностью «Студент» связью «один-ко-многим», которая обозначает, что студент учится в группе (в группе учится много студентов, студент может учиться только в одной группе, рис. 1.4). Сущность «Группа» связана с сущностью «Студент» связью «один-к-одному», которая обозначает, что студент является старостой группы (у группы только один староста, студент может быть старостой только одной группы, рис. 1.4). Сущность «Водитель» связана с сущностью «Автобус» связью «многие-ко-многим» (один водитель может водить несколько автобусов, один автобус может управляться несколькими водителями, рис. 1.5).

Бывают ситуации, когда между сущностями есть несколько связей. Например, между сущностями «Группа» и «Студент», как показано в примерах выше.

В большинстве случаев, когда между сущностями существует единственная связь кратности «один-к-одному», данные сущности

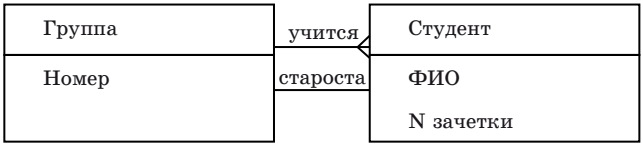

Рис. 1.4. Связи «один-ко-многим» и «один-к-одному»

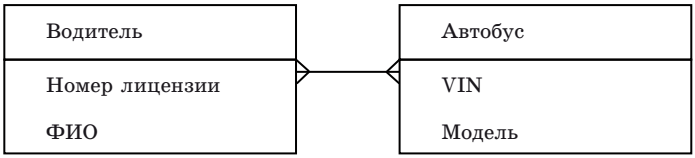

Рис. 1.5. Связь «многие-ко-многим»

могут быть отождествлены с точки зрения данных и слиты в одну. Например, сущности «Товар» и «Товар на складе» могут быть отождествлены в том случае, если склад единственен, и одному товару соответствует ровно один факт его наличия на этом складе.

Для каждой сущности должен быть определен ключ.

Ключ сущности – минимальный набор атрибутов и связей, комбинация значений которых гарантированно уникальна для всех экземпляров данной сущности (иными словами, гарантированно различается у любых двух ее экземпляров). Ключ может быть естественным (т.е. быть продиктованным семантикой предметной области) или суррогатным (т.е. введенным в модель только из-за необходимости наличия ключа при неудобстве использования или неинтуитивности естественного ключа).

В лабораторной работе в академических целях предпочтение необходимо отдавать естественным ключам.

В случае, если ключ составляет одиночный атрибут, его значение не может повториться ни у одного экземпляра сущности. В случае, если ключ состоит из нескольких атрибутов, по отдельности значение каждого из них может совпадать у двух экземпляров, а требование уникальности накладывается на комбинацию их значений.

Нередки случаи, когда частью естественного ключа для сущности является связь (такая связь называется **ключевой**). Это значит, что два экземпляра данной сущности при совпадении значений остальных частей ключа будут гарантированно связаны с различными экземплярами второй сущности.

Пример

Ключ сущности «Этап проекта» будет составным: (атрибут «Номер этапа», связь с сущностью «Проект»). Так, номера этапов, ассоциированных с одним и тем же проектом, будут гарантированно различными, а если у двух этапов номера совпадают, то они гарантированно относятся к различным проектам.

1.1.2. Переход к реляционной модели

После составления ER-модели необходимо, руководствуясь нижеприведенным алгоритмом, перейти к реляционной модели той же самой предметной области. В дальнейшем в СУБД используется именно она.

Реляционная модель является менее абстрактной и уточняет некоторые детали, такие как домены для атрибутов, специфические для СУБД названия отношений и атрибутов, ограничения целостности и др.

Алгоритм перехода от ER-модели к реляционной модели:

1. Каждой сущности ставится в соответствие отношение. Атрибуты сущности становятся атрибутами отношения. Если атрибут был ключевым, он становится частью потенциального ключа отношения.

2. Для каждой связи «один-ко-многим» ключевые атрибуты со стороны «один» копируются на сторону «многие». Если связь была ключевой (и только в этом случае), скопированные атрибуты становятся частью ключа.

3. Для каждой связи «многие-ко-многим» создается промежуточное отношение, в которое копируются ключевые атрибуты связанных сущностей. Все они образуют составной ключ. Полученное отношение (т.н. «отношение пересечения») связывается с исходными связями «многие-к-одному».

1.2. Индивидуальные варианты

Номер варианта соответствует порядковому номеру в журнале сдачи лабораторных работ.

1. Социальная сеть.
2. ВУЗ: учебный план и расписание.
3. ВУЗ: текущая успеваемость.
4. Справочник по сериалам.
5. Сеть ресторанов.

6. Рекламная фирма.
7. Книжный магазин.
8. Театр: расписание.
9. Железная дорога.
10. Театр: продажа билетов.
11. Агентство трудоустройства.
12. Химчистка.
13. Зоологический музей.
14. Проведение социологических опросов.
15. Справочник по фильмам.
16. ВУЗ: научная деятельность.
17. Турфирма.
18. Магазин мебели.
19. Музыкальный справочник.
20. Поликлиника.

Словесное описание предметной области можно получить у преподавателя.

1.3. Порядок выполнения

1. Ознакомиться с неформализованным словесным описанием предметной области, при необходимости уточнив детали у преподавателя.
2. Разработать ER-модель в соответствии с индивидуальным вариантом задания.
3. Осуществить переход к реляционной модели.
4. Оформить отчет по проделанной работе.

1.4. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Индивидуальный вариант задания.
4. Разработанная ER-модель.
5. Синтезированная реляционная схема.
6. Выводы.

1.5. Контрольные вопросы

1. В чем отличия инфологической и даталогической моделей?
2. Что является экземпляром сущности X?

3. Объясните семантику (смысл) связи между сущностями X и Y.

4. Объясните кратность (вид) связи между сущностями X и Y.

5. Что такое ключ в ER-модели?

6. Что такое ключевая связь?

7. Объясните выбор ключа для сущности X.

8. Что такое потенциальный ключ в реляционной модели?

9. В чем отличие первичного ключа от потенциального ключа?

10. В чем отличие двух потенциальных ключей a и b от составного потенциального ключа (a, b)?

11. Что такое внешний ключ в реляционной модели?

12. В чем отличие двух внешних ключей a и b от составного внешнего ключа (a, b)?

Лабораторная работа № 2.
СОЗДАНИЕ БАЗЫ ДАННЫХ В СУБД MICROSOFT ACCESS

Цель работы

Ознакомление с методами проектирования схемы и управления данными в графическом интерфейсе СУБД Microsoft Access.

2.1. Теоретические сведения

Microsoft Access – реляционная система управления базами данных (СУБД), входящая в пакет Microsoft Office. Не является клиент-серверной СУБД: реляционный движок Microsoft Jet встроен в приложение Access, что позволяет ему работать с файлами специального формата как с базой данных.

Access располагает компонентами для проектирования схемы базы данных, работы с данными в графическом интерфейсе, построения экранных форм, формирования отчетов. Несмотря на то, что функционал приложения как СУБД существенно ограничен и во многом не соответствует стандарту SQL, простота установки и настройки, а также удобный графический интерфейс пользователя делают его подходящим для изучения основ работы с реляционными базами данных.

2.2. Рекомендации по выполнению лабораторной работы

Для выполнения работы рекомендуется использовать Access версий 2013, 2016 или 2019.

При выполнении работы необходимо выбрать создание пустой базы данных, а затем по очереди создать все заданные в модели таблицы в режиме конструктора. В этом режиме для каждой колонки указывается имя, тип данных, а также ряд других параметров (размерность, маска ввода, обязательность, пользовательское ограничение).

Пользовательские ограничения (согласно индивидуальному варианту) задаются в поле «Правило проверки».

Следует различать ограничения уровня колонки (задаваемые в окне «Свойства поля», рис. 2.1), которые проверяются только при занесении или изменении значений в данной колонке, и ограничения уровня таблицы (задаваемые в окне «Свойства таблицы», рис. 2.2), которые проверяются при занесении или изменении значений сразу в нескольких колонках.

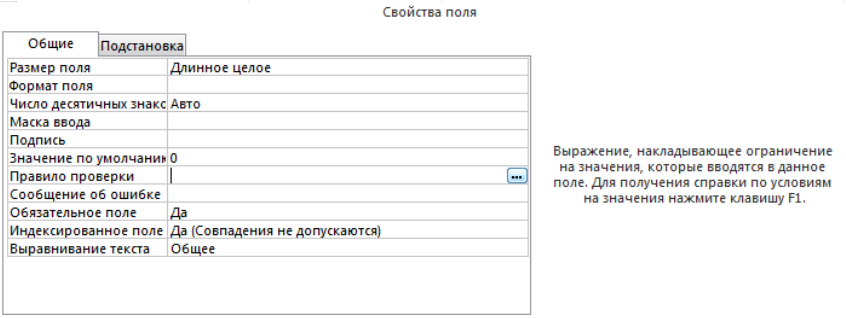

Рис. 2.1. Ограничение уровня колонки

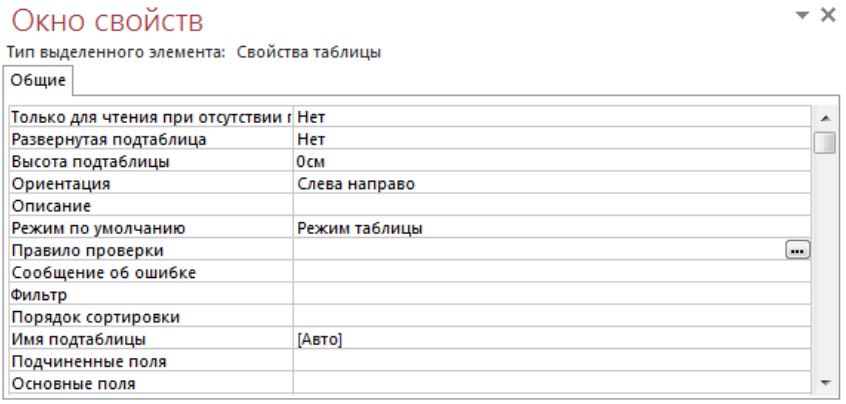

Рис. 2.2. Ограничение уровня таблицы

После создания таблиц необходимо перейти к созданию связей между ними, осуществляющих проверку ссылочной целостности. Следует помнить, что ссылаться можно только на колонку, имеющую тот же тип данных, и в целом ссылка (в том числе составная) должна идти на полный ключ.

Для каждой связи помимо указания связанных атрибутов необходимо включать проверку целостности (в противном случае ограничения по внешнему ключу не будут форсироваться), а также (по желанию) каскадное обновление и удаление связанных записей (рис. 2.3).

Включение каскадного обновления (удаления) активирует режим, когда при изменении (удалении) строки в главной таблице из-

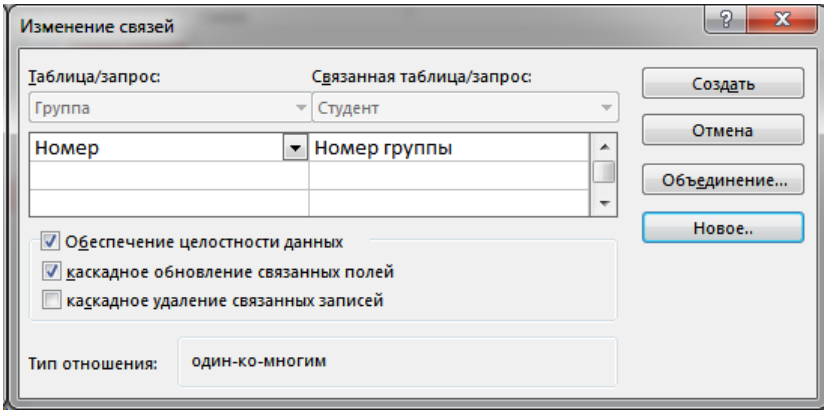

Рис. 2.3. Свойства связи

меняются (удаляются) строки в подчиненной таблице, ссылающуюся на изменяемую (удаляемую) строку.

После завершения работы над схемой данных, можно переходить к заполнению базы данных данными. Нужно помнить, что менять схему после занесения в таблицы данных может быть затруднительно. Всего необходимо добавить в таблицы по 5–10 строк, которые правдоподобно и полно отражали бы множественные взаимосвязи между данными.

Также необходимо проверить попыткой внесения некорректных данных, работают ли ограничения ссылочной целостности, заданные связями между таблицами, и пользовательские ограничения.

2.2. Индивидуальные варианты

Номер варианта соответствует порядковому номеру в журнале сдачи лабораторных работ.

Индивидуальные варианты соответствуют вариантам ЛР1.

2.3. Порядок выполнения

1. Реализовать реляционную схему, полученную в ЛР1, визуальными средствами Microsoft Access.

2. Осуществить проверку ограничений в соответствии с индивидуальным вариантом средствами Microsoft Access.

3. Заполнить базу данных данными, позволяющими удостовериться в работоспособности ограничений реляционной модели и пользовательских ограничений.

4. Оформить отчет по проделанной работе.

2.4. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Индивидуальный вариант задания.
4. Снимок экрана реляционной схемы в Microsoft Access.
5. Снимки экранов всех заполненных таблиц.
6. Снимки экранов результатов работы ограничений с данными, вызвавшими их срабатывание.
7. Выводы.

2.5. Контрольные вопросы

1. Объясните нотацию реляционной схемы, применяемую в Access.
2. В чем отличие проверок условия на значение на уровне колонки и на уровне таблицы?
3. Объясните принцип работы ограничения X.
4. Что такое потенциальный ключ в реляционной модели?
5. В чем отличие первичного ключа от потенциального ключа?
6. В чем отличие двух потенциальных ключей a и b от составного потенциального ключа (a, b)?
7. Как задать несколько ключей для таблицы в Access?
8. Что такое внешний ключ в реляционной модели?
9. В чем отличие двух внешних ключей a и b от составного внешнего ключа (a, b)?
10. Что означает свойство связи «Обеспечение целостности данных»?
11. Что означает свойство связи «Каскадное обновление связанных полей»?
12. Что означает свойство связи «Каскадное удаление связанных полей»?

Лабораторная работа № 3.
СОЗДАНИЕ БАЗЫ ДАННЫХ В СУБД SQLITE

Цель работы

Освоение синтаксиса операторов DDL и DML. Приобретение навыков использование команд языка SQL для создания базы данных с заданной схемой и управления данными в ней.

3.1. Теоретические сведения

3.1.1. Средства работы со схемой данных и добавления данных в языке SQL

SQL (Structured Query Language, структурированный язык запросов) – язык управления данными, используемый в реляционных системах управления базами данных. Предусматривает инструментарий для управления схемой данных (DDL, Data Definition Language, язык определения данных), осуществления операций с данными (DML, Data Manipulation Language, язык манипуляции данными) и выполнения запросов на выборку (DQL, Data Query Language, язык запросов к данным).

В языке SQL таблицы создаются с помощью оператора `CREATE TABLE`, после которого идет имя новой таблицы, а далее в скобках через запятую указывается список ее колонок с указанием типов данных, которые будут в них храниться. Помимо этого, для каждой колонки можно задать следующие ограничения:

`NOT NULL` – колонка является обязательной (не может содержать пустых ячеек);

`PRIMARY KEY` – колонка будет первичным ключом (при этом автоматически накладывается ограничение `NOT NULL`). В таблице не может быть двух строк с одинаковым значением первичного ключа, таким образом, зная первичный ключ, можно однозначно найти строку в таблице;

`UNIQUE` – колонка не может содержать повторяющихся значений (является дополнительным ключом);

`FOREIGN KEY ... REFERENCES` – колонка может содержать только те значения, которые находятся в некоторой колонке другой таблицы (является внешним ключом);

`DEFAULT` – значение по умолчанию.

Для добавления данных в существующую таблицу используется оператор `INSERT INTO`. После него указывается ключевое слово `VALUES`,

а затем, опять в скобках, – список значений, которые подлежат добавлению. Порядок значений должен строго соответствовать порядку колонок в таблице.

3.1.2. Общие сведения об SQLite

SQLite – легковесный реляционный движок; представляет собой встраиваемую в приложение библиотеку, и так же как Microsoft Jet, не работает по схеме «клиент-сервер». В отличие от Microsoft Jet, который используется только в Access, SQLite широко используется в прикладных приложениях, в том числе на мобильных платформах, в качестве средства хранения данных. С учетом легкости установки, отсутствия необходимости настройки и неплохой поддержки стандарта SQL, данная СУБД хорошо подходит для использования в учебных целях.

3.2. Рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в программе SQLiteSpy, но допустимо использование других клиентов.

Особенностью движка SQLite является выключенное по умолчанию ограничение ссылочной целостности (не производится форсирование внешних ключей). Это объясняется стремлением сохранить обратную совместимость с приложениями, написанными для более старых версий библиотеки (поддержка внешних ключей была реализована в SQLite начиная с версии 3.6.19 в 2009 году). В академических целях необходимо в самом начале скрипта принудительно включить форсирование внешних ключей командой

```
PRAGMA foreign_keys = ON;
```

Скрипт лабораторной работы представляет собой одиночный текстовый файл, структурированный следующим образом:

– команда включения внешних ключей;
– команды очистки схемы данных (DROP TABLE);
– команды создания схемы данных (CREATE TABLE);
– команды заполнения базы данных данными (INSERT).

Таблицы желательно заполнять данными с учетом ограничений ссылочной целостности, т.е. сначала главные, а затем подчиненные. Иными словами, вначале заполняются данными таблицы, на которые указывают ссылки, а затем – таблицы, которые содержат ссылки. В этом случае ссылки при добавлении строк в подчиненные таблицы будут валидными и не вызовут нарушений целостности.

Команды очистки схемы данных необходимы для удобства исполнения скрипта целиком в случае, если в схему данных были внесены изменения и надо перегенерировать ее. При этом нужно помнить, что если в таблицах уже содержатся данные, при удалении таблиц будут форсироваться ограничения ссылочной целостности (не будет позволено удалить таблицу, если существуют данные, которые ссылаются на строки из удаляемой таблицы). Чтобы гарантированно перегенерировать схему и данные, следует придерживаться определенного порядка при удалении таблиц, а именно противоположного их созданию и заполнению: сначала имеет смысл удалять подчиненные таблицы, содержащие ссылки, и лишь затем – главные таблицы, на которые те ссылались.

Перед сдачей лабораторной работы следует убедиться, что скрипт исполняется полностью многократно и не вызывает ошибок.

3.3. Индивидуальные варианты

Номер варианта соответствует порядковому номеру в журнале сдачи лабораторных работ.

Индивидуальные варианты соответствуют вариантам ЛР1.

3.4. Порядок выполнения

1. Реализовать реляционную схему, полученную в ЛР1, операторами DDL языка SQL в SQLite (с помощью SQLiteSpy или другого приложения, использующего движок SQLite).
2. Командой PRAGMA включить проверку ограничения ссылочной целостности.
3. Осуществить проверку ограничений в соответствии с индивидуальным вариантом ЛР2 средствами языка SQL.
4. Операторами INSERT языка SQL заполнить базу данных данными, позволяющими удостовериться в работоспособности ограничений реляционной модели и пользовательских ограничений.
5. Оформить отчет.

3.5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Индивидуальный вариант задания.
4. Снимок экрана реляционной схемы в Microsoft Access.

5. Листинг скрипта SQL создания и заполнения базы данных.

6. Выводы.

Примечание: листинг скрипта должен быть приведен с использованием моноширинного шрифта для удобочитаемости.

3.6. Контрольные вопросы

1. Объясните синтаксис и семантику оператора CREATE TABLE.

2. Объясните синтаксис и семантику оператора INSERT.

3. В чем проявляется то, что SQLite проверяет ссылочную целостность данных?

4. В чем отличие двух потенциальных ключей a и b от составного потенциального ключа (a, b)?

5. В чем отличие двух внешних ключей a и b от составного внешнего ключа (a, b)?

6. Объясните принцип работы ограничения X.

Лабораторная работа № 4.
СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ НА ОСНОВАНИИ ЗАПРОСОВ НА ВЫБОРКУ В СУБД SQLITE. ЗАПРОСЫ НА МОДИФИКАЦИЮ И УДАЛЕНИЕ ДАННЫХ В СУБД SQLITE

Цель работы

Освоение синтаксиса операторов SELECT, UPDATE и DELETE при выполнении запросов на выборку, модификацию и удаление данных. Приобретение навыков использования средств языка SQL для выполнения типовых запросов к данным в реляционной модели. Знакомство с принципами работы представлений в реляционных СУБД.

4.1. Теоретические сведения

Для выполнения запросов на выборку данных из базы используется оператор `SELECT`. Его синтаксис таков:

```
SELECT список_колонок_результата
FROM список_таблиц
WHERE условие_отбора_для_строк
GROUP BY список_колонок_для_группировки
HAVING условие_отбора_для_групп_строк
ORDER BY порядок_сортировки
LIMIT количество_строк_в_результате_запроса
```

Из 7 секций оператора `SELECT` обязательной является только секция `SELECT`. При этом при наличии остальных секций, они должны идти в указанном порядке и не повторяться.

Выполнение запроса происходит в другом порядке, нежели его запись.

1) декартово произведение таблиц, указанных в секции `FROM`;

2) фильтрация строк удовлетворяющих условию, указанному в секции `WHERE`;

3) группировка строк, имеющих одинаковое значение в колонке, указанной в `GROUP BY`;

4) отбор полученных групп в соответствии с условием в секции `HAVING`;

5) сортировка полученных строк / групп строк в соответствии с указаниями секции `ORDER BY`;

6) отбор только верхние строки результата, количество которых указано в `LIMIT`;

7) отбор колонок, указанных в секции `SELECT`.

Для изменения данных используется оператор UPDATE. После ключевого слова UPDATE указывается имя таблицы, подлежащей изменению, затем – ключевое слово SET, после которого через запятую названия изменяемых колонок и после знака «=» значения, которые в них записываются. Если необходимо изменить только некоторые строки, то указывается условие фильтрации строк после слова WHERE. В противном случае изменяются все строки таблицы.

Для удаления строк из таблицы используется оператор DELETE. Если не указано слово WHERE и условие отбора удаляемых строк, то удаляются все строки таблицы. В противном случае удаляются только строки, соответствующие указанному условию.

4.2 Рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в программе SQLiteSpy, но допустимо использование других клиентов.

Скрипт лабораторной работы представляет собой одиночный текстовый файл, структурированный следующим образом:

– команды удаления представлений (DROP VIEW);

– команды создания представлений (CREATE VIEW);

– команда на модификацию данных (UPDATE, должна быть закомментирована);

– команда на удаление данных (DELETE, должна быть закомментирована).

Команда создания представления должна сопровождаться SQL-комментарием, объясняющим назначение этого представления.

Следует отметить, что представление должно создаваться одиночным SQL-запросом (при этом допускается использование вложенных запросов) и не обращаться к другим представлениям в схеме. И вообще, создание иных представлений помимо тех, которые фигурируют в задании, должно быть согласовано с преподавателем.

Команды модификации и удаления данных следует закомментировать, чтобы они не выполнялись при выполнении скрипта целиком.

Перед сдачей лабораторной работы следует убедиться, что скрипт исполняется полностью многократно и не вызывает ошибок.

4.3. Индивидуальные варианты

Индивидуальные задания можно получить у преподавателя.

4.4. Порядок выполнения

1. Реализовать для базы данных, полученной в ЛР3, представления, а также запросы на модификацию и удаление на основании задания, полученного от преподавателя.
2. Оформить отчет.

4.5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Индивидуальный вариант задания.
4. Снимок экрана реляционной схемы в Microsoft Access.
5. Листинг скрипта SQL создания представлений, а также запросов на модификацию и удаление.
6. Выводы.

Примечание: листинг скрипта должен быть приведен с использованием моноширинного шрифта для удобочитаемости.

4.6. Контрольные вопросы

1. Объясните синтаксис и семантику оператора SELECT.
2. Что такое представления и зачем они нужны?
3. Объясните синтаксис и семантику оператора UPDATE.
4. Объясните синтаксис и семантику оператора DELETE.
5. Объясните принцип работы запроса X.

Лабораторная работа № 5.
СОЗДАНИЕ ТРИГГЕРОВ В СУБД SQLITE

Цель работы

Изучение принципов работы триггеров в реляционных СУБД. Приобретение навыков создания триггеров на добавление, изменение и удаление данных для создания пользовательских ограничений целостности данных.

5.1. Теоретические сведения

Триггер – объект базы данных, который представляют собой набор действий, выполняемых автоматически при наступлении определенных условий. Как правило, триггер привязан к конкретной таблице и запускается при выполнении над ней определенной операции (удаление, добавление, изменение строк).

Синтаксис создания триггера:

```
CREATE TRIGGER имя_триггера
[{BEFORE|AFTER}] [{INSERT|UPDATE|DELETE} [OF имя_колонки]]
ON имя_таблицы
[WHEN условие]
BEGIN
 действия
END;
```

В триггерах есть 2 специальные переменные:

`OLD` – строка, которая только что была удалена из таблицы (доступна в триггерах, отслеживающих удаление и изменение строк).

`NEW` – строка, которая только что была добавлена в таблицу (доступна в триггерах, отслеживающих добавление и изменение строк).

5.2. Рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в программе SQLiteSpy, но допустимо использование других клиентов.

Скрипт лабораторной работы представляет собой одиночный текстовый файл, структурированный следующим образом:

– команды удаления триггеров (DROP TRIGGER);

– команды создания триггеров (CREATE TRIGGER);

– команды проверки работы триггеров (должны быть закомментированы).

Команда создания триггера должна сопровождаться SQL-комментарием, объясняющим назначение этого триггера.

Команды проверки работы триггеров представляют собой DML-операторы, которые должны добавлять/изменять/удалять данные таким образом, чтобы спровоцировать выполнение соответствующего триггера на тех данных, которые имеются в базе данных после выполнения генерирующего скрипта.

Перед сдачей лабораторной работы следует убедиться, что скрипт исполняется полностью многократно и не вызывает ошибок.

5.3. Индивидуальные варианты

Индивидуальные задания можно получить у преподавателя.

5.4. Порядок выполнения

1. Реализовать для базы данных, полученной в ЛР3, триггеры на основании задания, полученного от преподавателя.
2. Оформить отчет.

5.5. Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Индивидуальный вариант задания.
4. Снимок экрана реляционной схемы в Microsoft Access.
5. Листинг скрипта SQL создания триггеров.
6. Примеры срабатывания.
7. Выводы.

Примечание: листинг скрипта должен быть приведен с использованием моноширинного шрифта для удобочитаемости.

5.6. Контрольные вопросы

1. Объясните назначение и принцип работы триггеров.
2. В чем разница между тем или иным видом триггеров?
3. Объясните синтаксис и семантику оператора CREATE TRIGGER.
4. Объясните принцип работы триггера X.

Лабораторная работа № 6.
УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ В СУБД MYSQL

Цель работы

Ознакомление с механизмами изоляции транзакций при параллельном выполнении запросов к одним и тем же данным на различных уровнях изоляции с использованием блокировок и многоверсионности. Получение навыков работы с СУБД MySQL.

6.1. Теоретические сведения

6.1.1. Понятие транзакции

Транзакция – группа операторов, осуществляющих действия над данными, которая обладает ACID-свойствами, т.е.

– атомарностью (не может быть выполнена наполовину);

– целостностью (переводит данные из одного корректного состояния в другое);

– изоляцией (допускает одновременное выполнение других транзакций над теми же данными);

– долговечностью (гарантирует сохранность результатов в случае своего успешного выполнения).

Большинство современных реляционных СУБД являются транзакционными (т.н. ACID-совместимыми), и это свойство лежит в основе гарантий, предоставляемых ими в отношении надежности хранения данных, обеспечения параллельного доступа многих пользователей к одним и тем же данным, а также поддержания целостности (т.е. непротиворечивости) данных при выполнении сложных операций над ними.

Свойство изоляции может быть достигнуто несколькими способами, самые распространенные из которых – многоверсионность и использование блокировок.

6.1.2. Многоверсионность (на примере MySQL)

Идея многоверсионности (MVCC, Multi-Versioned Concurrency Control) как метода достижения изоляции основана на том факте, что процесс, выполняющий чтение данных (оператор SELECT) заинтересован не столько в их свежести, сколько в их целостности. Следовательно, можно избавиться от необходимости блокировок на чтение, вместо этого предоставляя читающим транзакциям копию

(слепок) последнего, возможно, не самого свежего, но целостного набора запрашиваемых данных.

На уровне REPEATABLE READ каждой транзакции присваивается временная отметка, после чего она читает свой личный слепок целостных данных из журнала транзакций. Добавления, изменения или удаления данных, произведенные параллельными транзакциями с более поздними временными отметками, не видны всем операторам текущей транзакции.

На уровне READ COMMITTED каждый оператор SELECT работает со наиболее свежим на момент его старта целостным слепком набора данных. При этом если какая-то другая транзакция после этого изменяет этот набор данных и фиксируется, следующий оператор SELECT получит его новую версию в качестве целостного слепка.

Следует понимать, что многоверсионность проявляется только при чтении данных (т.е. отсутствуют S-блокировки, не блокируются зависимости W-R и R-W). При этом запись осуществляется непосредственно в БД, а не в слепок, и если возникает зависимость W-W, используются блокировки.

На уровне READ UNCOMMITTED многоверсионность не используется: каждый оператор SELECT читает последнюю (возможно, «грязную») версию данных непосредственно из БД.

Уровень SERIALIZABLE в MySQL эквивалентен уровню REPEATABLE READ с неявным принудительным включением разделяемых блокировок в операторах SELECT.

6.1.3. Принудительное включение блокировок

В MySQL есть возможность принудительного включения блокировок на чтение. Это достигается конструкцией SELECT ... LOCK IN SHARE MODE.

При этом вне зависимости от уровня изоляции, на существующие в таблице строки, удовлетворяющие условию WHERE оператора SELECT текущей транзакции, накладываются разделяемые блокировки, что:

– не позволяет текущей транзакции читать данные, измененные другой транзакцией, которая еще не была зафиксирована («грязное чтение», W-R);

– не позволяет другим транзакциям изменять данные, прочитанные текущей транзакцией (неповторяемое чтение, R-W).

Разница между уровнями изоляции проявляется лишь в способах наложения блокировок. На уровнях READ COMMITTED

и READ UNCOMMITTED блокируются только существующие строки, удовлетворяющие условию фильтрации. На уровне REPEATABLE READ используются диапазонные блокировки, что исключает возможность появления фантомов.

6.1.4. Общие сведения о MySQL

MySQL – клиент-серверная система управления базами данных. Это означает, что СУБД выполняется в виде процесса и имеет полный доступ к базе данных, тогда как другие приложения при необходимости работать с данными обязаны формировать запросы и отправлять их процессу СУБД.

MySQL имеет в распоряжении несколько движков: InnoDB, MyISAM, XtraDB и другие.

В контексте рассмотрения механизмов управления транзакциями, наибольший интерес представляет ACID-совместимый движок InnoDB, который является который с 2010 года (MySQL 5.5) является движком по умолчанию для таблиц.

6.2. Рекомендации по выполнению лабораторной работы

Задания выполняются с использованием сервера MySQL версии 5.5 или выше. В качестве клиента можно использовать либо поставляемую с сервером утилиту командной строки mysql, либо официальный инструментарий MySQL Workbench, либо стороннее ПО, например, dbForge Studio for MySQL или HeidiSQL. При использовании ПО с графическим интерфейсом пользователя задания выполняются в окне редактора SQL-запросов.

В лабораторной работе необходимо отключить интерактивный режим выполнения запросов, в котором работают все клиенты MySQL. В этом режиме каждый оператор SQL, обрабатываемый клиентом, облекается в отдельную транзакцию, которая автоматически фиксируется после его выполнения (т.н. режим автофискации, autocommit). В лабораторной работе исследуются методы разрешения зависимостей между параллельно (т.е. одновременно) выполняющимися транзакциями, обращающимися к одним и тем же данным, поэтому транзакции должны состоять из нескольких операторов и выполняться продолжительное время.

В интерфейсе командной строки или редакторе SQL начать транзакцию без автофиксации можно выполнив команду BEGIN.

В MySQL Workbench можно достичь того же, отключив режим автофиксации транзакций (Auto-Commit Transactions), после чего все выполняемые операторы будут объединены в одну транзакцию, а также станут доступны команды фиксации и отката текущей транзакции (рис. 6.1).

Те же элементы управления продублированы на панели инструментов (рис. 6.2).

В dbForge Studio for MySQL по умолчанию также включен режим автофиксации. Можно отключив его, нажав на кнопку начала транзакции или выбрав соответствующий элемент контекстного меню соединения (рис. 6.3).

После этого становятся доступны команды фиксации и отката текущей транзакции (рис. 6.4).

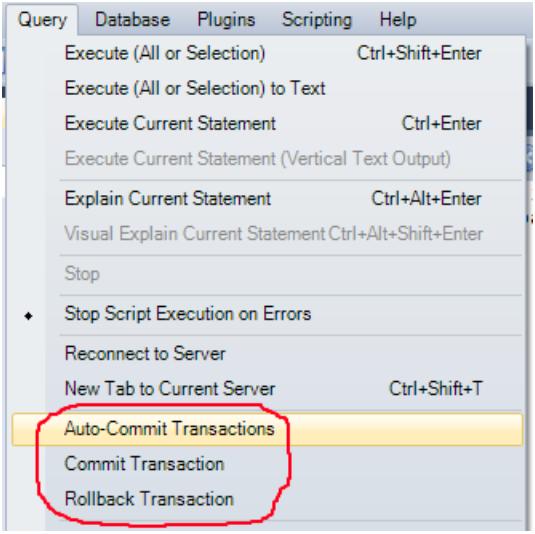

Рис. 6.1. Команды управления транзакциями в меню MySQL Workbench

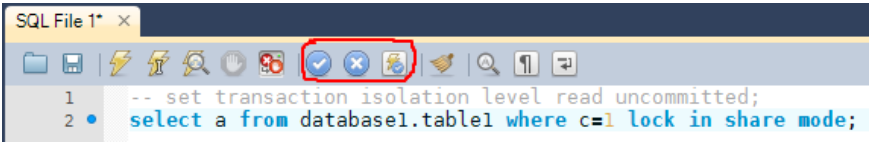

Рис. 6.2. Команды управления транзакциями на панели инструментов MySQL Workbench

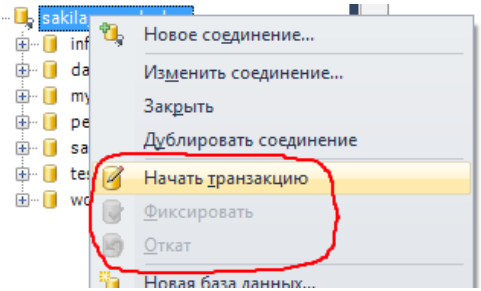

Рис. 6.3. Команды управления транзакциями в меню dbForge Studio for MySQL

Рис. 6.4. Команды управления транзакциями на панели инструментов dbForge Studio for MySQL

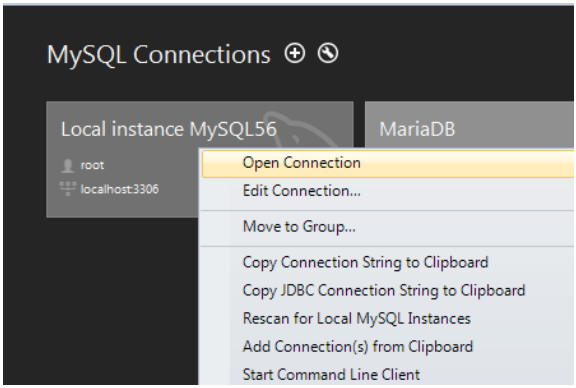

Рис. 6.5. Команда открытия нового подключения в MySQL Workbench

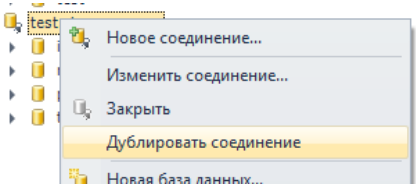

Рис. 6.6. Команда открытия нового подключения в dbForge Studio for MySQL

Для заданий задан уровень изоляции (RU – READ UNCOMMITTED, RC – READ COMMITTED, RR – REPEATABLE READ). Необходимо установить в подключении соответствующий уровень изоляции:

```
-- для следующей транзакции:
SET TRANSACTION ISOLATION LEVEL уровень;
-- для всех последующих транзакций данного подключения:
SET SESSION TRANSACTION ISOLATION LEVEL уровень
```

Затем необходимо запустить второе подключение к той же БД. Уровень изоляции можно оставить по умолчанию (в InnoDB это REPEATABLE READ, если не изменялись настройки).

В MySQL Workbench для открытия второго подключения необходимо повторно выполнить команду Open Connection для созданного подключения к базе данных (рис. 6.5). Оно создастся в соседней вкладке.

В dbForge Studio for MySQL второе подключение открывается командой дублирования соединения (рис. 6.6).

Следует отметить, что в MySQL Workbench по умолчанию включен так называемый «безопасный режим» обновлений и удалений, который заключается в том, что выполняются только те команды UPDATE и DELETE, которые в секции WHERE содержат условие фильтрации по ключевому полю. Считается, что это позволит избежать ошибочного изменения или удаления многих строк сразу неопытными пользователями. Поскольку для выполнения работы иногда необходимо фильтровать запросы не по ключевому полю, данный режим следует отключить. Для этого необходимо зайти в меню настроек среды (Edit -> Preferences...) и на вкладке «SQL Editor» снять галочку «Safe Updates» (рис. 6.7).

Параллельно запускаются две транзакции из двух открытых подключений. Первая транзакция T1 (из подключения с уровнем изоляции согласно варианту) производит операции с данными, после чего вторая транзакция T2 также работает с этими данными и так далее. Список действий должен полностью представлять реакцию системы на зависимости между транзакциями.

Для каждого варианта необходимо проанализировать:

1) Зависимость потерянного обновления

T1	T2
read(x)	
	write(x)
write(x)	

2) Зависимость грязного чтения

T1	T2
	write(x)
read(x)	
	write(x)

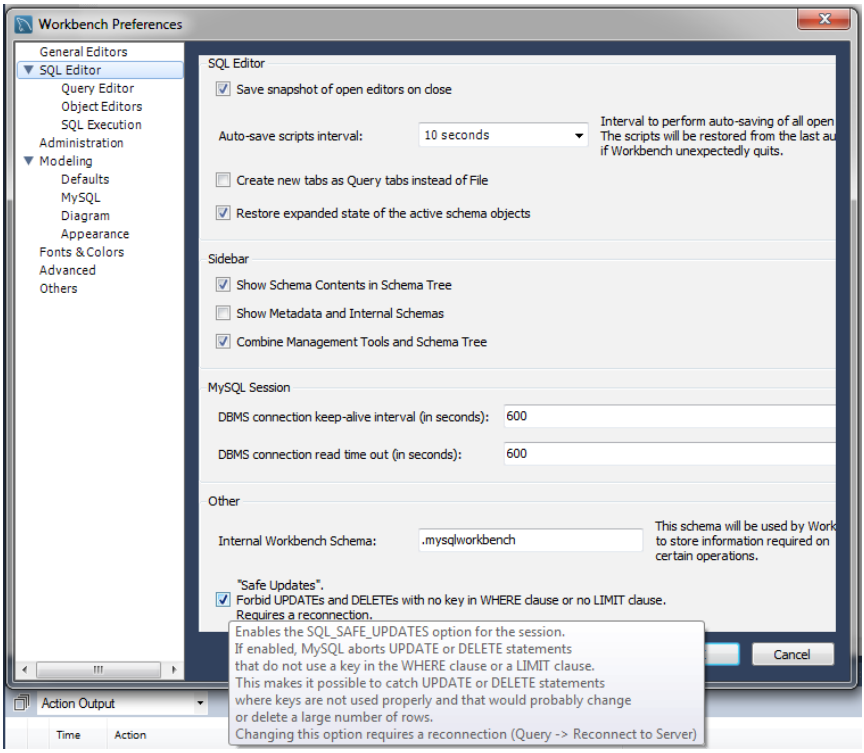

Рис. 6.7. Управление «безопасным режимом» в MySQL Workbench

3) Зависимость неповторяемого чтения

T1	T2
read(x)	
	write(x)
read(x)	

4) Зависимость фантомов

T1	T2
filter(t)	
	add(t)
filter(t)	

Задания MVCC связаны с исследованием механизма многоверсионности для движка InnoDB. При этом транзакцией T1 производятся целостные неблокирующие чтения (в InnoDB – обычный оператор SELECT).

Задания LOCK связаны с исследованием реализации принудительных разделяемых блокировок в движке InnoDB. При этом транзакцией T1 производятся блокирующие чтения (в InnoDB – оператор SELECT ... LOCK IN SHARE MODE).

Пример 1

MVCC, уровень READ UNCOMMITTED, зависимость потерянного обновления.

T1	T2
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED; BEGIN; SELECT * FROM t WHERE i = 3; +------+ | i | +------+ | 3 | +------+ 1 row in set (0.00 sec)	
	BEGIN; UPDATE t SET i=3 WHERE i=2; Query OK, 1 row affected (0.05 sec) Rows matched: 1 Changed: 1 Warnings: 0
UPDATE t SET i=5 WHERE i=2; ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction	
	COMMIT;

Вывод: потерянное обновление не допускается, транзакция T1, пытающаяся изменить данные, на которые наложена X-блокировка транзакцией T2, ждет, а затем завершается по таймауту.

Пример 2

LOCK, уровень READ COMMITTED, зависимость фантомов.

T1	T2
SET TRANSACTION ISOLATION LEVEL READ COMMITTED; BEGIN; SELECT * FROM t WHERE i = 3 LOCK IN SHARE MODE;	

```
+------+
| i    |
+------+
|    3 |
+------+
1 row in set (0.00 sec)
                                  BEGIN;
                                  INSERT INTO t VALUES (3);
                                  Query OK, 1 row affected (0.00 sec)
SELECT * FROM t WHERE i = 3 LOCK
IN SHARE MODE;
ожидание…
                                  COMMIT;
+------+
| i    |
+------+
|    3 |
|    3 |
+------+
2 rows in set (0.00 sec)
```

Вывод: фантомы допускаются, повторное выполнение запроса транзакцией T1 получает больший набор строк.

6.3. Индивидуальные варианты

Номер варианта вычисляется как

$$((N - 1) \bmod 6) + 1,$$

где N – индивидуальный номер, полученный у преподавателя.

Например, индивидуальный номер N = 35. Тогда номер варианта:

$$((N - 1) \bmod 6) + 1 = ((35 - 1) \bmod 6) + 1 = (34 \bmod 6) + 1 = 4 + 1 = 5$$

В рамках лабораторной работы необходимо исследовать зависимости, отмеченные в таблице значком «+» в строке соответствующего варианта:

Вариант	RU MVCC	RC MVCC	RR MVCC	RC LOCK	RR LOCK
1	+			+	
2		+			+
3			+	+	
4	+				+
5		+		+	
6			+		+

6.4. Порядок выполнения

1. Создать с помощью скрипта, полученного в ЛР3, базу данных на сервере MySQL.

2. В двух параллельно запущенных сеансах выполнить набор операторов SQL, позволяющий изучить взаимодействие одновременно выполняющихся транзакций при доступе к одним и тем же данным на заданных в индивидуальном варианте уровнях изоляции.

3. Оформить отчет.

6.5. Содержание отчета

1. Титульный лист.

2. Цель работы.

3. Индивидуальный вариант задания.

4. Снимок экрана реляционной схемы в Microsoft Access.

5. Листинги скриптов SQL пар транзакций, анализирующих наличие/отсутствие проблем параллельного выполнения транзакций на разных уровнях изоляции с использованием разделяемых блокировок или многоверсионностью.

6. Выводы для каждого из листингов.

6.6. Контрольные вопросы

1. Что такое транзакция?

2. Что такое ACID-свойства?

3. Какие существуют методы достижения изоляции?

4. Опишите различия между уровнями изоляции X и Y.

5. В чем заключается принцип многоверсионности?

6. Объясните разницу между эксклюзивными и разделяемыми блокировками.

7. Поясните листинг A.

Библиографический список

1. *Преснякова, Г. В.* Проектирование интегрированных реляционных баз данных / Г. В. Преснякова. – М.: КДУ; СПб.: Петроглиф, 2007. – 224 с.

2. *Фуфаев, Э. В.* Базы данных: учебное пособие для учреждений СПО / Э. В. Фуфаев, Д. Э. Фуфаев. – 4-е изд., стер. – М.: Академия, 2008. – 320 с.

3. *Хомоненко, А. Д.* Базы данных: учебник для высших учебных заведени й/ А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; ред. А. Д. Хомоненко. – 6-е изд., доп. и перераб. – СПб.: КОРОНА-Век, 2010. – 736 с.

4. *Илюшечкин, В. М.* Основы использования и проектирования баз данных: учеб. пособие / В. М. Илюшечкин. – М.: Юрайт, 2011. – 213 с.

5. *Советов, Б. Я.* Базы данных: теория и практика: учебник / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М.: Юрайт, 2012. – 464 с.

6. *Ульман, Дж.* Основы реляционных баз данных / Дж. Ульман, Дж. Уидом; пер. П. Быстров. – М.: Лори, 2006. – 382 с.

СОДЕРЖАНИЕ

Учебное издание

Аксенов Алексей Владимирович

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ. ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА SQL-ЗАПРОСОВ, УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ

Лабораторный практикум

Публикуется в авторской редакции.
Компьютерная верстка *Н. Н. Караваевой*

Подписано к печати 28.08.20. Формат 60×84 1/16.
Усл. печ. л. 2,1. Уч.-изд. л. 2,3. Тираж 50 экз. Заказ № .

Редакционно-издательский центр ГУАП
190000, Санкт-Петербург, Б. Морская ул., 67