

федеральное государственное автономное образовательное учреждение  
высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ  
К ЛАБОРАТОРНЫМ ЗАНЯТИЯМ**

по дисциплине

**БАЗЫ ДАННЫХ**

направление подготовки

**02.03.02 - Фундаментальная информатика и информационные  
технологии**

**Составитель**

к.т.н., доцент кафедры  
программных систем  
Попова-Коварцева  
Дарья Александровна

Самара 2023

## Оглавление

Введение .....	3
Лабораторная работа № 1 .....	6
Лабораторная работа № 2 .....	23
Лабораторная работа № 3 .....	34
Лабораторная работа № 4 .....	42
Лабораторная работа № 5 .....	44
Литература .....	50

## Введение

Данный курс лабораторных работ предназначен для студентов специальности 020302 - фундаментальная информатика и информационные технологии.

**Целями лабораторного практикума** являются:

- приобретение практических навыков анализа и моделирования предметной области;
- ознакомление с работой специализированных CASE-средств проектирования баз данных;
- приобретение практических навыков использования структурированного языка запросов SQL.

## Требования к организации и проведению лабораторных занятий

На лабораторных занятиях студент выполняет персональное (индивидуальное) задание соответствующее варианту, что способствует более эффективному формированию практических умений, навыков и компетенций.

Задания выполняются *строго* в соответствии с утвержденным преподавателем графиком.

***Выполнение всех заданий является необходимым условием допуска студента к экзамену.***

Цикл лабораторных работ необходимо выполнять с использованием СУБД MySQL, в частности с использованием **MySQL Workbench**.

## Программное обеспечение цикла лабораторных работ

**Draw.io** — инструмент для создания диаграмм, блок-схем, интеллект-карт, бизнес-макетов, отношений сущностей, программных блоков и другого. Сервис распространяется на бесплатной основе с открытым исходным кодом. Draw.io обладает богатым набором функций для визуализации большинства задач пользователя.

При входе на сервис пользователь сразу попадает в рабочий интерфейс. У пользователя нет возможности для авторизации или регистрации, есть только опция выбора места для экспорта проекта. Процесс создания проекта выглядит следующим образом: пользователь перетаскивает из левой панели фигуры или элементы на рабочую поверхность, затем изменяет их — изменяет цвет, размер, шрифт текста, свойства фигуры (прозрачность, форма и т. д.). Draw.io позволяет отслеживать и восстанавливать изменения готовых проектов, импортировать и экспортировать в PDF, PNG, XML, VSDX, HTML, а также автоматически публиковать и делиться работами.

Инструмент работает с Google Диск, Google Workspace и Dropbox, глубоко интегрирован и удобен для работы с продуктами Confluence и Jira от Atlassian. Пользователи также могут работать с диаграммами в автономном режиме и сохранять их локально, используя настольное приложение для персональных компьютеров.

Инструмент позволяет создавать: графики, диаграммы, таблицы, презентации, блок-схемы, планы помещений, воронки продаж, ментальные карты, карты сайтов.

### **Особенности Draw.io:**

- Более 500 шаблонов элементов и фигур;
- Облегченный интерфейс, в котором за короткий промежуток времени можно создать готовый проект;
- Поддержка горячих клавиш, задействованных в большинстве графических редакторов;
- Экспорт в форматы: JPG, PNG, SVG, VDSX;
- Возможность совместной работы;
- Наличие различных фоновых тем;
- Мультиязычный интерфейс.

СУБД **MySQL** — это система управления реляционными базами данных. Программное обеспечение MySQL — это ПО с открытым кодом. На веб-сайте MySQL (<http://www.mysql.com>) представлена самая последняя информация о программном обеспечении MySQL. Код написан на C и C++. СУБД может быть установлена на все операционные системы с работающими потоками Posix и компилятором C++. MySQL поддерживает большое число типов данных столбцов, практически все стандарты функций ANSI SQL, а также программный интерфейс доступа к базам данных ODBC. MySQL позволяет управлять очень большими базами данных (более 50 миллионов записей). Система основана на привилегиях и паролях, за счет чего обеспечивается гибкость и безопасность, а также возможность верификации с удаленного компьютера. Пароли являются защищенными, поскольку при передаче по сети при соединении с сервером они шифруются.

Очевидно, что в настоящее время создание баз данных невозможно без их тщательного проектирования с применением CASE-средств. Использование современных CASE-средств для проектирования различного рода программных продуктов на сегодняшний день связано не просто с необходимостью облегчить труд разработчика ПО, но и с необходимостью иметь инструмент, позволяющий также стандартизировать процесс разработки, получить наглядные графические модели разрабатываемого ПО, повторно использовать разработки и иметь возможность разделения огромных по объему задач между разработчиками.

При разработке баз данных часто в связи с большим объемом информации,

необходимой для хранения, возникают сложности даже с построением логической модели, не говоря уже о физической модели, в которой необходимо учесть возможности конкретной системы управления баз данных (СУБД). В связи с этим многие СУБД создают свои CASE-средства, позволяющие разработчику в значительной степени автоматизировать процессы моделирования, проектирования и реализации баз данных. В настоящий момент в состав пакета разработчика помимо СУБД MySQL входит CASE-средство Workbench, которое также может быть получено на сайте (<http://www.mysql.com>) и установлено. Его использование в значительной мере упрощает и делает более наглядным процесс проектирования баз данных.

**MySQL Workbench** — инструмент для визуального проектирования баз данных, который позволяет проектировать, моделировать, создавать и эксплуатировать БД. С его помощью можно представить модель базы данных в графическом виде, устанавливая связи между таблицами, восстанавливать структуры уже существующей на сервере БД (обратный инжиниринг), использовать редактор SQL запросов, позволяющий сразу же отправлять их на сервер и получать ответ в виде таблицы, редактировать данные в таблице в визуальном режиме.

Следует заметить, что основные принципы работы с MySQL Workbench остаются неизменными вне зависимости от того, установлена ли СУБД на ОС Windows или ОС Linux. Различия в большей степени касаются коммерческой и свободно распространяемой версий. Некоторые возможности доступны только для коммерческого варианта. Также более новые версии могут незначительно отличаться от предыдущих, например иметь небольшие расхождения в наименовании пунктов меню или экранов. Однако основные подходы разработки и реализации не изменяются.

В цикле лабораторных работ будут рассмотрены основные возможности использования MySQL Workbench для проектирования и реализации баз данных, в том числе реализация баз данных традиционным способом при помощи создания таблиц и связей между ними, а также при помощи построения ER-моделей.

## Лабораторная работа № 1

**Тема:** инфологическое и логическое проектирование базы данных (ER-моделирование).

**Цель:** формирование навыков по исследованию заданной предметной области и умений выделить сущности и атрибуты, созданию ER-модели базы данных в различных нотациях.

### Порядок выполнения работы №1:

1. Ознакомиться с описанием предметной области варианта задания.
2. С помощью инструмент для создания диаграмм Draw.io, создать ER-диаграмму базы данных, соответствующей варианту задания, используя нотацию Питера Чена.
  - a. Определить необходимые сущности и их атрибуты для предметной области.
  - b. Указать первичные ключи сущностей.
  - c. Определить связи между сущностями. При необходимости добавить в проект дополнительные сущности.

Сохранить проект.

3. В рабочей области MySQL Workbench создать ER-модель базы данных в нотации IDEF1X.
  - a. В каждом отношении задать первичный ключ и столбцы- атрибуты предметной области. Определить типы данных столбцов.
  - b. Определить типы и мощности связей между сущностями.
  - c. Создать связи между сущностями. Проверить наличие и расположение внешних ключей в дочерних таблицах.
  - d. Сохранить модель.

### Описание вариантов заданий предметной области

#### Вариант № 1

Предметная область: Деканат (успеваемость студентов).

Основные предметно-значимые сущности: Студенты, Группы студентов, Дисциплины.

Основные предметно-значимые атрибуты сущностей:

- студенты – фамилия, имя, отчество, пол, дата рождения, адрес прописки;
- группы студентов – название, курс, семестр;
- дисциплины – название.

Основные требования к функциям системы:

- 1) выбрать успеваемость студента по дисциплинам с указанием общего количества часов и вида контроля;
- 2) выбрать успеваемость студентов по группам и дисциплинам;
- 3) выбрать группы, в которых у студентов в настоящее время наибольшая успеваемость;
- 4) выбрать дисциплины, по которым у студентов в настоящее время наибольшая успеваемость;
- 5) выбрать возраст студентов, у которых в настоящее время наибольшая успеваемость;
- 6) выбрать дисциплины, изучаемые группой студентов на определенном курсе или определенном семестре;
- 7) выбрать дисциплины, которые не изучаются студентами в настоящий момент.

## **Вариант № 2**

Предметная область: Отдел кадров (контингент сотрудников).

Основные предметно-значимые сущности: Сотрудники, Подразделения.

Основные предметно-значимые атрибуты сущностей:

- сотрудники – фамилия, имя, отчество, пол, дата рождения, адрес прописки;
- подразделения – название, вид подразделения.

Основные требования к функциям системы:

- 1) выбрать список сотрудников по подразделениям или определенному подразделению;
- 2) подсчитать средний возраст работающих сотрудников по подразделениям;
- 3) выбрать должности работающих сотрудников по подразделениям;
- 4) подсчитать количество работающих сотрудников по подразделениям и должностям;
- 5) выбрать подразделения с наибольшим количеством работающих сотрудников;
- 6) выбрать подразделения, в которых за последние полгода уволилось больше всего сотрудников;
- 7) выбрать список сотрудников по составу (профессорско-преподавательский состав, учебно-вспомогательный состав, административно-хозяйственный состав и т.п.).

## **Вариант № 3**

Предметная область: Учебно-методическое управление (профессорско-преподавательский состав).

Основные предметно-значимые сущности: Сотрудники, Подразделения, Дисциплины.

Основные предметно-значимые атрибуты сущностей:

- сотрудники – фамилия, имя, отчество, пол, дата рождения, адрес прописки;
- подразделения – название;
- дисциплины – название.

**Основные требования к функциям системы:**

- 1) выбрать дисциплины, читаемые сотрудниками или определенным сотрудником;
- 2) выбрать список сотрудников по подразделениям или определенному подразделению;
- 3) подсчитать количество сотрудников по подразделениям;
- 4) подсчитать средний возраст сотрудников по подразделениям;
- 5) выбрать дисциплины, читаемые сотрудниками по подразделениям или определенному подразделению;
- 6) подсчитать количество дисциплин по подразделениям;
- 7) вывести подразделения, сотрудники которых в настоящий момент не участвуют в учебном процессе.

**Вариант № 4**

**Предметная область:** Учебно-методическое управление (учет площади помещений).

**Основные предметно-значимые сущности:** Помещения, Подразделения.

**Основные предметно-значимые атрибуты сущностей:**

- помещения – название или номер помещения, вид помещения (аудитория, кабинет и т.п.), площадь, количество посадочных мест;
- подразделения – название, вид подразделения.

**Основные требования к функциям системы:**

- 1) выбрать названия или номера помещений по подразделениям;
- 2) вывести подразделения, которые не имеют посадочных мест для студентов;
- 3) вывести количество посадочных мест по подразделениям;
- 4) подсчитать общую площадь учебных аудиторий по подразделениям и в целом по учебному заведению;
- 5) вывести количество посадочных мест для студентов по аудиториям;
- 6) подсчитать общее количество посадочных мест для сотрудников;
- 7) подсчитать общее количество посадочных мест для сотрудников по подразделениям.

**Вариант № 5**

**Предметная область:** Поликлиника (учет пациентов).

**Основные предметно-значимые сущности:** Пациенты, Врачи.

**Основные предметно-значимые атрибуты сущностей:**

- пациенты – фамилия, имя, отчество, дата рождения;



- врачи – фамилия, имя, отчество, дата рождения, должность, специализация.

Основные требования к функциям системы:

- 1) выбрать все диагнозы по пациентам или определенному пациенту;
- 2) выбрать всех пациентов записанных к определенному врачу на определенную дату;
- 3) выбрать всех здоровых на настоящее время пациентов;
- 4) подсчитать количество приемов пациентов по врачам за последний месяц;
- 5) выбрать наиболее распространенные болезни за последний месяц;
- 6) подсчитать количество посещений по пациентам за последний месяц;
- 7) выбрать всех врачей, к которым записан определенный пациент.

**Вариант № 6**

Предметная область: Бухгалтерия (учет материальных ценностей).

Основные предметно-значимые сущности: Оборудование, Подразделения, Материально ответственные лица.

Основные предметно-значимые атрибуты сущностей:

- оборудование – название, стоимость;
- подразделения – название, вид подразделения;
- материально ответственные лица – фамилия, имя, отчество.

Основные требования к функциям системы:

- 1) выбрать все не списанное оборудование по материально-ответственным лицам или определенному лицу;
- 2) подсчитать остаточную стоимость не списанного оборудования по материально ответственным лицам или определенному лицу;
- 3) выбрать все не списанное оборудование по подразделениям или определенному подразделению;
- 4) выбрать подразделения, не имеющие в настоящее время на балансе никакого не списанного оборудования;
- 5) выбрать списанное оборудование по подразделениям и материально-ответственным лицам;
- 6) выбрать материально-ответственных лиц с наибольшей суммой остаточной стоимости оборудования;
- 7) подсчитать общую стоимость не списанного оборудования по подразделениям.

**Вариант № 7**

Предметная область: Студенческое общежитие.

Основные предметно-значимые сущности: Студенты, Общежития, Комнаты.

Основные предметно-значимые атрибуты сущностей:

- студенты – фамилия, имя, отчество;
- общежития – название или номер общежития, адрес;
- комнаты – название или номер комнаты, этаж.

**Основные требования к функциям системы:**

- 1) выбрать всех студентов, проживающих в общежитии, с указанием комнаты по общежитиям или определенному общежитию;
- 2) выбрать всех студентов, проживающих в общежитии, с указанием комнаты по группам студентов или определенной группе;
- 3) выбрать группы студентов, в которых все студенты проживают в общежитии;
- 4) выбрать комнаты, в которых в настоящее время есть свободное место;
- 5) подсчитать количество студентов по группам, проживающих в настоящий момент в общежитии;
- 6) выбрать группы студентов, в которых все студенты не проживают в общежитии;
- 7) подсчитать количество проживающих студентов по комнатам с указанием общежития.

**Вариант №8**

**Предметная область:** Приемная комиссия (абитуриенты).

**Основные предметно-значимые сущности:** Абитуриенты, Специальности, Предметы.

**Основные предметно-значимые атрибуты сущностей:**

- абитуриенты – фамилия, имя, отчество, пол, дата рождения;
- специальности – название специальности;
- предметы – название предмета.

**Основные требования к функциям системы:**

- 1) выбрать всех абитуриентов по специальностям или определенной специальности;
- 2) подсчитать количество абитуриентов по специальностям или определенной специальности;
- 3) выбрать всех абитуриентов, не явившихся на вступительные экзамены;
- 4) подсчитать рейтинг (сумма баллов по всем сданным предметам) всех абитуриентов, сдавших вступительные экзамены, по специальностям или определенной специальности;
- 5) выбрать специальности с наибольшим количеством абитуриентов;
- 6) выбрать предметы, которые абитуриенты сдавали наиболее успешно;
- 7) подсчитать средний балл по предметам и специальностям.

## **Вариант № 9**

Предметная область: Бухгалтерия (расчет стипендии).

Основные предметно-значимые сущности: Студенты, Группы студентов, Результаты сдачи сессии.

Основные предметно-значимые атрибуты сущностей:

- студенты – фамилия, имя, отчество;
- группы студентов – название или номер группы;
- результаты сдачи сессии – студент, семестр, название категории (не сдал, сдал на 3, сдал на 4-5, сдал на 5).

Основные требования к функциям системы:

- 1) назначить размер стипендии студентам за последнюю сессию в соответствии с действующими правилами;
- 2) вывести группы, в которых студенты не получают стипендии;
- 3) выбрать всех студентов, обучающихся по целевым договорам, сдавших на 3 или вообще не сдавших последнюю сессию, по группам;
- 4) выбрать всех студентов, сдавших последнюю сессию на 5, по группам;
- 5) подсчитать сумму стипендий студентов по курсам;
- 6) вывести размер назначенной стипендии студентам, обучающихся по целевым договорам, по предприятиям;
- 7) подсчитать сумму стипендий по группам.

## **Теоретические сведения**

### **Общие сведения о ER-моделях**

При построении баз данных CASE-средства чаще всего ориентированы на два уровня представления модели — логический и физический. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация обо всех объектах БД. Поскольку стандартов на объекты БД не существует (например, хотя имеется стандарт основных типов данных, в каждой конкретной СУБД могут использоваться дополнительно и свои типы данных), физическая модель зависит от используемой СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько различных физических моделей.

Создание модели данных, как правило, начинается с проектирования логической модели. Для проектирования могут использоваться различные инструменты, однако в последнее время наиболее часто применяются CASE-средства (Computer Aided

Software Engineering) — средства для автоматизации проектирования программных продуктов. Как правило, CASE-средства предлагают построение ER-моделей.

Для создания ER-диаграмм обычно используют одну из двух наиболее распространенных нотаций.

- **IDEF1X** — усовершенствованная версия IDEF1 (методологии структурного анализа для проектирования сложных ИС, разработанный Т. Рэмей (T. Ramey)) и позволяющая разрабатывать концептуальную модель предметной области системы баз данных в форме одной или нескольких ER-диаграмм, эквивалентных отношениям в третьей нормальной форме. Помимо того, что эта нотация стала федеральным стандартом США, она также является стандартом в ряде международных организаций (например, Международный валютный фонд и др.);
- **Information Engineering (IE)**. Нотация, разработанная Мартином (Martin), Финкельштейном (Finkelstein) и др., используется преимущественно в промышленности.

Заметим, что IDEF1X является методом для разработки реляционных баз данных и использует нотацию, специально разработанную для удобного построения концептуальной схемы — универсального представления структуры данных в рамках рассматриваемой предметной области, независимого от конечной реализации базы данных и аппаратной платформы. Использование метода IDEFIX наиболее целесообразно для построения логической структуры базы данных.

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, является жесткая и строгая стандартизация моделирования.

IDEFIX используется для моделирования реляционных баз данных и имеет в США статус федерального стандарта. Стандарт входит в семейство методологий IDEF (методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем), позволяющих исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем.

MySQL – свободная реляционная система управления базами данных (БД). Полная документация на английском языке доступна на сайте <https://dev.mysql.com/doc/>.

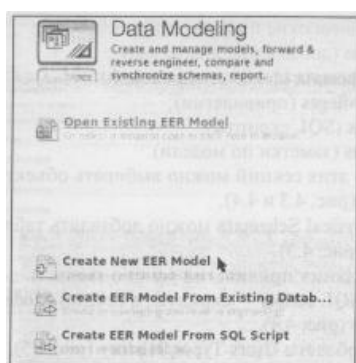
Среда **MySQLWorkbench** предназначена для визуального проектирования баз данных и управления сервером **MySQL**.

## Рабочее пространство для проектирования ER-моделей в MySQL Workbench

Рассмотрим основные принципы работы с MySQL Workbench для построения ER-моделей. При работе с ER-моделями Workbench позволяет:

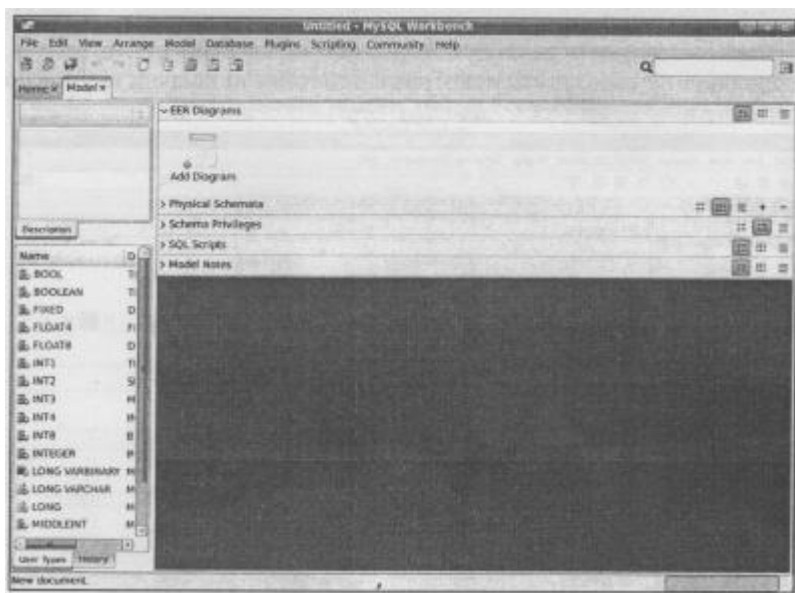
- создавать графическую модель базы данных;
- при помощи прямого инжиниринга создавать из графической модели базу данных;
- при помощи обратного инжиниринга из существующей базы данных получать графическую модель.

Вход в MySQL Workbench осуществляется нажатием соответствующей иконки на рабочем столе или через меню «Пуск». Далее для работы с ER-моделями необходимо выбрать для входа область моделирования данных, расположенную в центре (Data Modeling), и в ней выбрать переход на создание новой модели — Create New EER Model:



Обратите внимание, что в данном случае построение модели ведется в EER (enhanced entity-relationship) редакторе. Основное отличие EER-моделирования от ER-моделирования состоит в том, что EER поддерживает понятия подкласса и суперкласса, а также связанных с ними понятий специализации и обобщения. Однако в большинстве случаев для проектирования баз данных эти возможности являются излишними.

После запуска будет открыто окно редактирования модели:



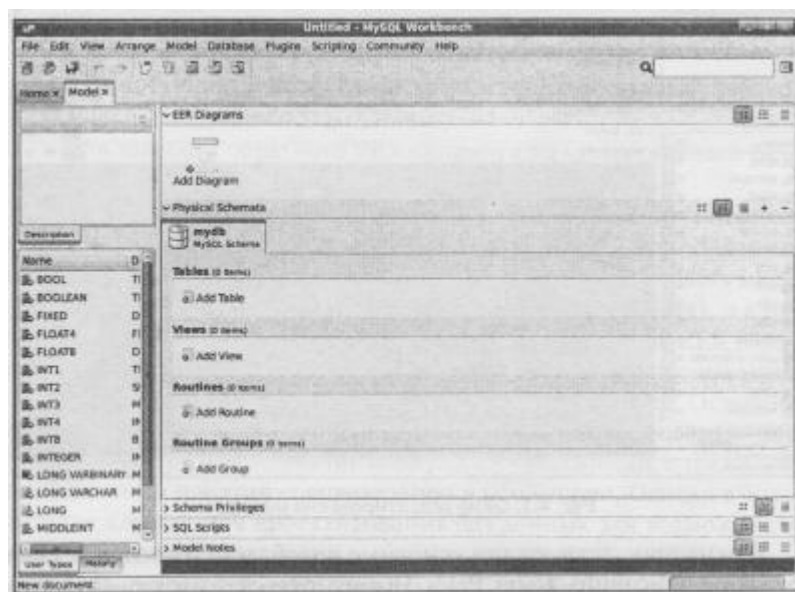
В окне присутствуют три основные подобласти: Description Editor (редактор описаний), Users Types/History (окно для пользовательских типов данных/истории модели) и основное окно.

В этом основном окне присутствуют вкладки:

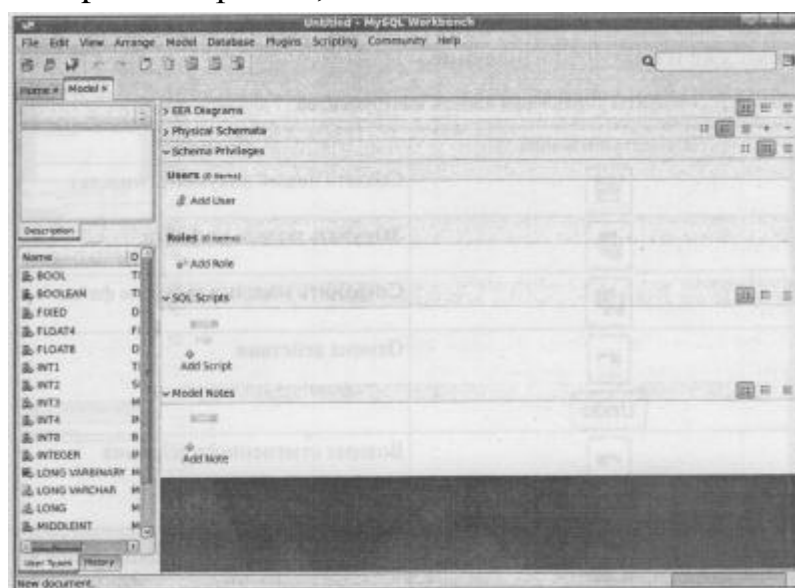
- Add Diagram (добавление диаграммы);
- Physical Schemata (физическая схема);
- Schema Privileges (привилегии);
- SQL Scripts (SQL скрипты);
- Model Notes (заметки по модели).

В каждой из этих секций можно выбирать объекты для добавления их в модель.

В секции Physical Schemata можно добавлять таблицы, вид, программу, группу:



В секции задания привилегий можно задавать пользователей и роли, в секции SQL scripts — скрипты, в секции Model Notes — замечания по модели:



Слева в подобласти Users Types/History показаны типы данных пользователей:

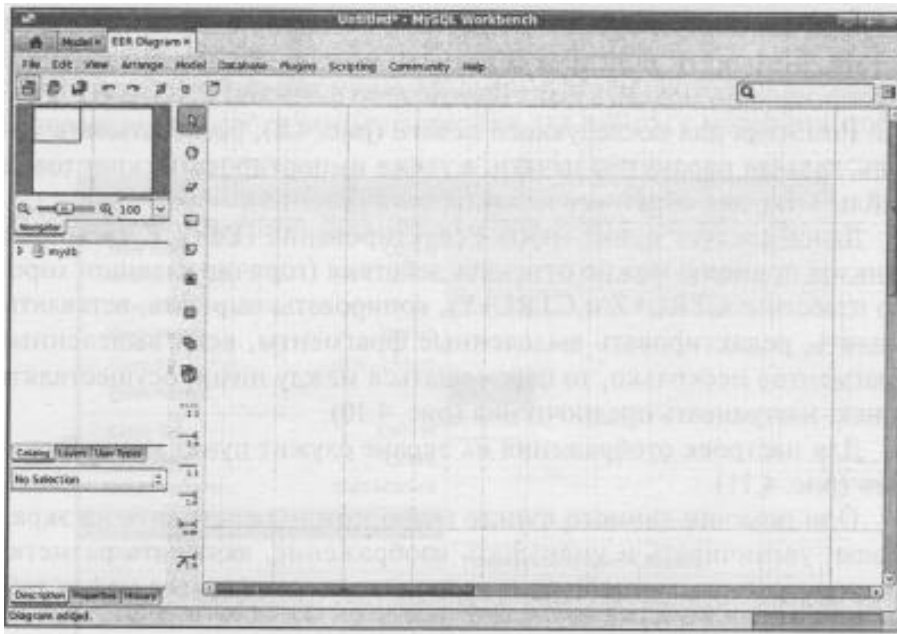
Name	Definition	Flags
BOOL	TINYINT(1)	
BOOLEAN	TINYINT(1)	
FIXED	DECIMAL(10,0)	
FLOAT4	FLOAT	
FLOAT8	DOUBLE	
INT1	TINYINT(4)	
INT2	SMALLINT(6)	
INT3	MEDIUMINT(9)	
INT4	INT(11)	
INT8	BIGINT(20)	
INTEGER	INT(11)	
LONG VARBINARY	MEDIUMBLOB	
LONG VARCHAR	MEDIUMTEXT	
LONG	MEDIUMTEXT	
MIDDLEINT	MEDIUMINT(9)	
NUMERIC	DECIMAL(10,0)	
DEC	DECIMAL(10,0)	
CHARACTER	CHAR(1)	

Опции основного меню являются аналогичными опциям меню для ER-модели и будут рассмотрены далее.

Ниже под пунктами основного меню расположены пиктограммы, позволяющие выполнять некоторые действия непосредственно по щелчку мыши, а не через пункты меню. Некоторые пункты меню имеют уже известное назначение.

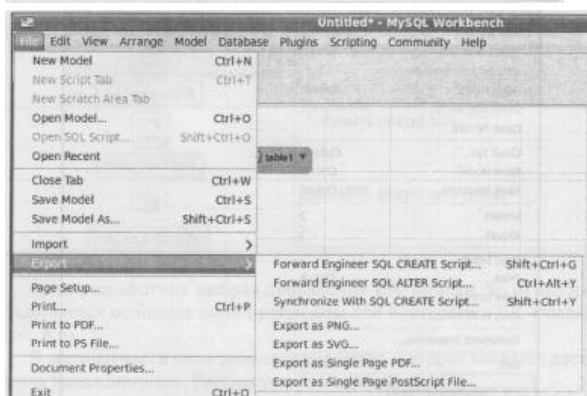
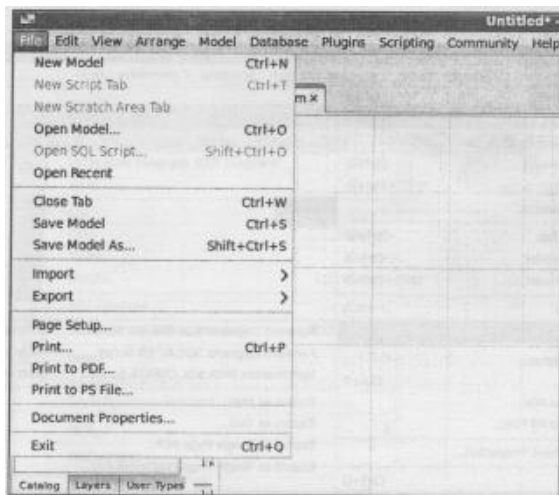
Элемент управления	Описание
	Создать новый документ (модель)
	Загрузить модель из файла
	Сохранить модель в текущем файле
	Отмена действия
Undo	
	Возврат отмененного действия
Redo	
	Добавить новую диаграмму
Add new Diagram	
	Добавить новую схему
Add new Schema	
	Добавить новую таблицу
Add new Table	
	Добавить новый вид
Add new View	
	Добавить новую программу
Add new Routine	

Если разработчик выбрал добавление ER-модели, то перед ним открывается основное окно программы для построения ER-моделей:

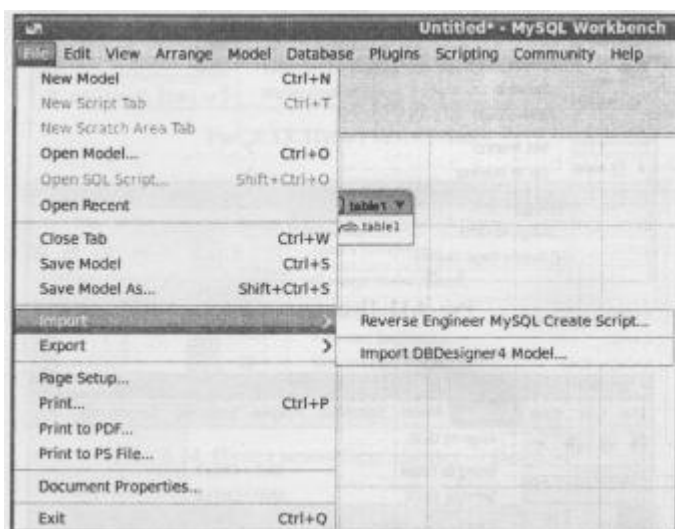


В верхней части окна расположено меню, которое является достаточно традиционным. Рассмотрим его подробнее.

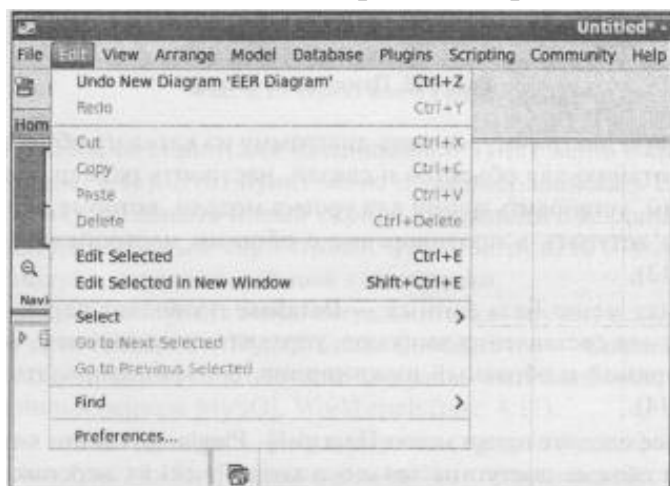
Пункт меню работы с файлом (File) позволяет создавать новую модель, создавать новую модель из скриптов, открывать существующую модель, открывать модель, с которой недавно проходила работа, сохранять модель и сохранять модель под другим именем, экспортировать модель в файл следующего формата: PNG, SVG, PDF или PostScript для последующей печати, распечатывать модель, задавая параметры печати, а также импортировать скриптовые файлы DDL для обратного инжиниринга:







Далее следует пункт меню Редактирование (Edit). С помощью пунктов подменю можно отменять действия (горячие клавиши хорошо известны: CTRL+Z и CTRL+Y), копировать, вырезать, вставлять, удалять, редактировать выделенные фрагменты, если выделенных фрагментов несколько, то перемещаться между ними, осуществлять поиск, настраивать предпочтения:



Для настроек отображения на экране служит пункт меню Вид — View:



При помощи данного пункта меню можно переходить на экран Home, увеличивать и уменьшать изображение, включать разметку сетки, выравнивать изображение по сетке, перемещаться между таблицами.

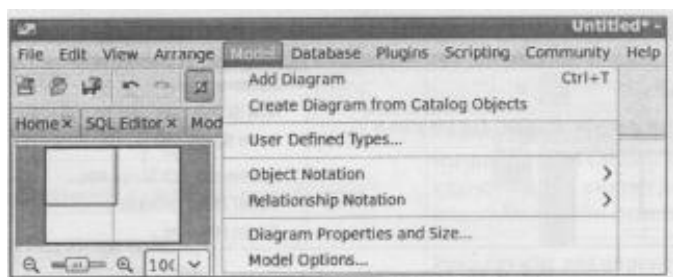
Для настройки расположения изображения, его растяжения и сжатия служит

пункт меню Размещение — Arrange:

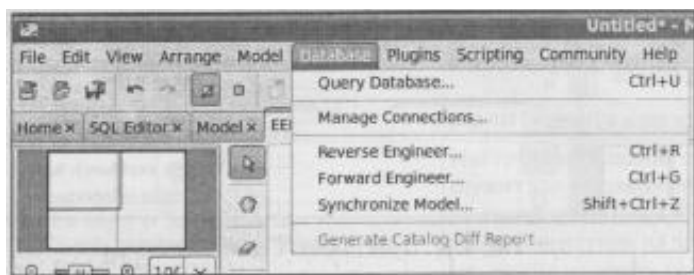


Данный пункт меню применим только к активным (выделенным) объектам на диаграмме.

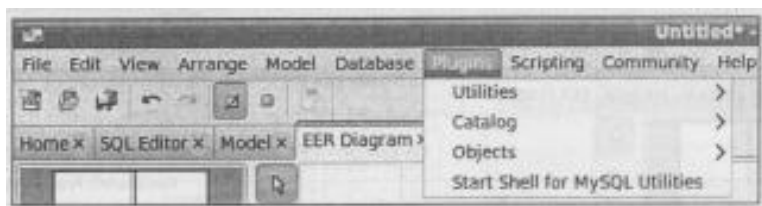
Далее следует пункт меню Модель — Model, который позволяет произвести все необходимые настройки для работы с моделями: добавить новую диаграмму, создать диаграмму из каталога объектов, выбрать нотацию для объектов и связей, настроить размер и свойства диаграмм, установить опции для уровня модели, которые, однако, не должны вступать в противоречие с общими настройками модели:



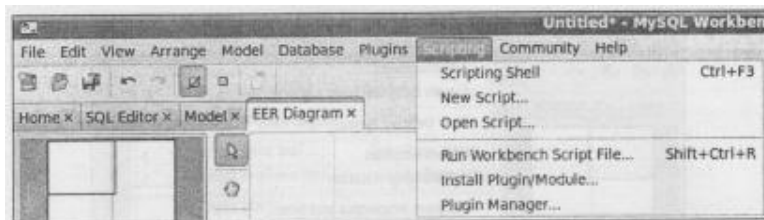
Пункт меню База данных — Database позволяет переходить на вкладку для составления запросов, управлять соединением, осуществлять прямой и обратный инжиниринг, синхронизировать модели:



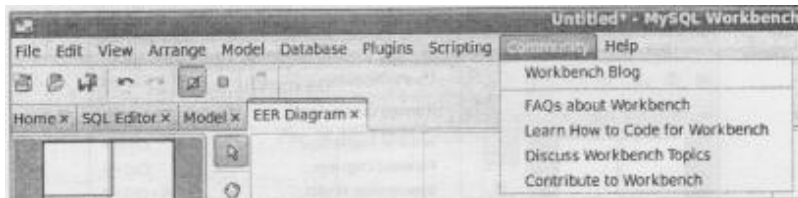
Далее следует пункт меню Плагин — Plugin, функции которого в полном объеме доступны только в коммерческих версиях MySQL:



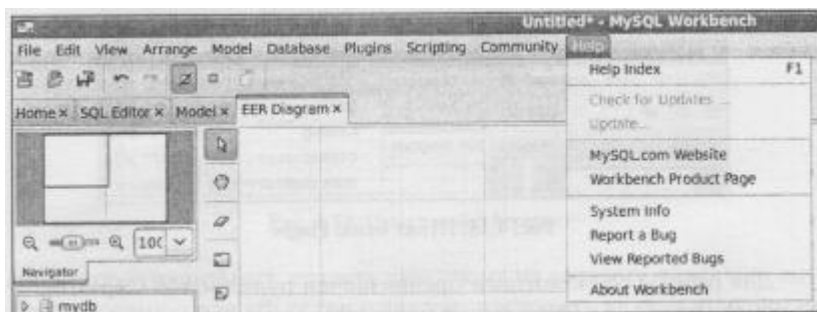
Для работы со скриптами предназначен пункт меню Скрипты — Scripting. Этот пункт меню позволяет запускать скриптовую оболочку, создавать новый скрипт, открывать имеющийся, выполнять определенный скриптовый файл, загружать и выполнять плагин/модуль, управлять работой с плагинами:












Два последних пункта меню актуальны в том случае, если имеется доступ к сети Интернет. Пункт меню Сообщество — Community позволяет получить доступ к обсуждению различных вопросов, связанных с использованием MySQL Workbench:









Пункт меню Помощь — Help содержит справочную информацию по продукту. Традиционно вызов справки осуществляется при помощи клавиши F1:






Далее рассмотрим вертикальную панель управления, служащую для создания и редактирования ER-диаграмм. Элементы управления приведены в таблице:

Элемент управления и горячая клавиша	Описание
 Select Object(s) (Quick key - press Escape)	Стандартный указатель мыши, на него следует переключаться для снятия любого выделения иного пункта меню
 Move Model (Quick key - press H)	Рука служит для перемещения ER-модели по диаграмме
 Delete Object (Quick key - press D)	Ластик служит для удаления объектов с диаграммы
 Place a New Layer at Selected Area. Use for visually grouping related objects in the diagram. (Quick key - press L)	Инструмент создания слоев служит для создания объектов на диаграмме. После того, как объект поместили на диаграмме, можно изменять размеры, растягивая его. Также можно произвести настройки изображения
 Place a New Text Object (Quick key - press N)	Данная пиктограмма служит для создания текста
 Place a New Image (Quick key - press I)	Данная пиктограмма служит для помещения рисунков на диаграмму
 Place a New Table (Quick key - press T)	Для создания новой таблицы на диаграмме используется следующая пиктограмма
 Place a New View (Quick key - press V)	Для настройки изображения следует использовать пиктограмму View
 Place a New Routine Group (Quick key - press G)	Помещение новой области программ






В таблице ниже приведены связи, пиктограммы которых (и горячие клавиши) расположены на вертикальной панели инструментов под элементами управления:

Элемент управления	Описание
 Place a New 1:1 Non-Identifying Relationship (Quick key - press 1)	Создание неидентифицирующей связи один-к-одному
 Place a New 1:n Non-Identifying Relationship (Quick key - press 2)	Создание неидентифицирующей связи один-ко-многим
 Place a New 1:1 Identifying Relationship (Quick key - press 3)	Создание идентифицирующей связи один-к-одному
 Place a New 1:n Identifying Relationship (Quick key - press 4)	Создание идентифицирующей связи один-ко-многим
 Place a New n:m Identifying Relationship (Quick key - press 5)	Создание идентифицирующей связи многие-ко-многим
 Place a Relationship Using Existing Columns (Quick key - press 6)	Создать связь, используя имеющиеся столбцы

Последнее, что следует рассмотреть — элементы управления, которые расположены под главным меню. Они приведены в таблице:

Элемент управления	Описание
	Создать новый документ (модель)
	Загрузить модель из файла
	Сохранить модель в текущем файле

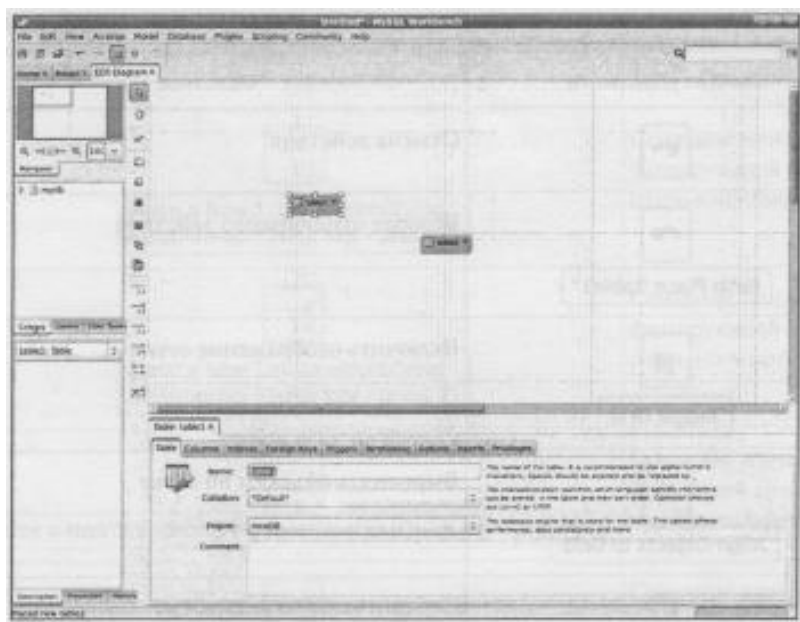
Элемент управления	Описание
	Отмена действия
 Redo Place 'table1'	Возврат отмененного действия
 Toggle Grid	Включить отображение сетки
 Align Objects to Grid	Выровнять объекты по сетке
 Add new Diagram	Добавить новую диаграмму

Некоторые пиктограммы имеют уже известное назначение.

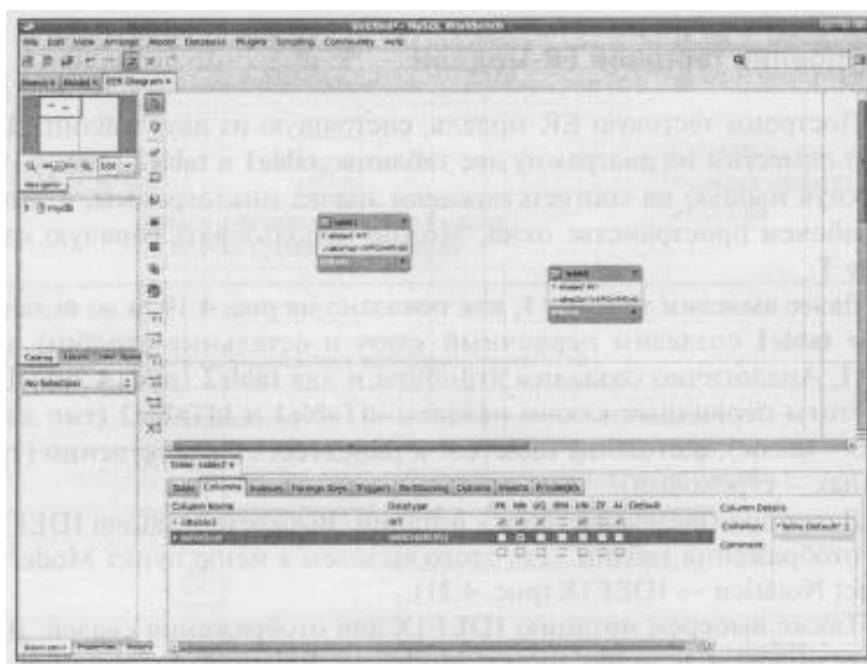
## Построение тестовой ER-модели

Построим тестовую ER-модель, состоящую из двух таблиц. Для этого поместим на диаграмму две таблицы: table1 и table2, щелкнув мышью на соответствующем

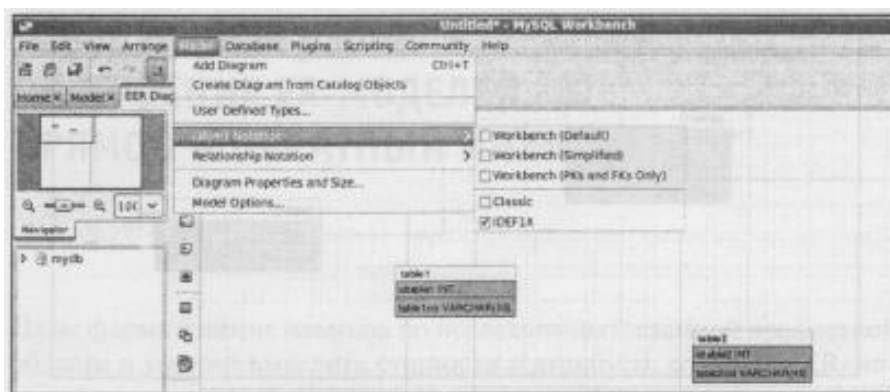
значке пиктограммы, а затем на рабочем пространстве окна. Можно использовать горячую клавишу T:



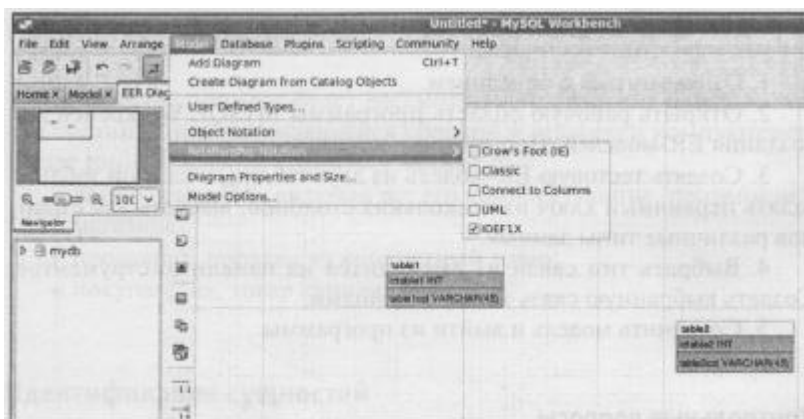
Далее выделим таблицу 1 и во вкладке Table table1 создадим первичный ключ и остальные столбцы для table1. Аналогично создадим атрибуты и для table2. Для простоты первичные ключи назовем idTable 1 и idTable2 (тип данных — целое), а столбцы table1 col и table2 col соответственно (тип данных — строковый):



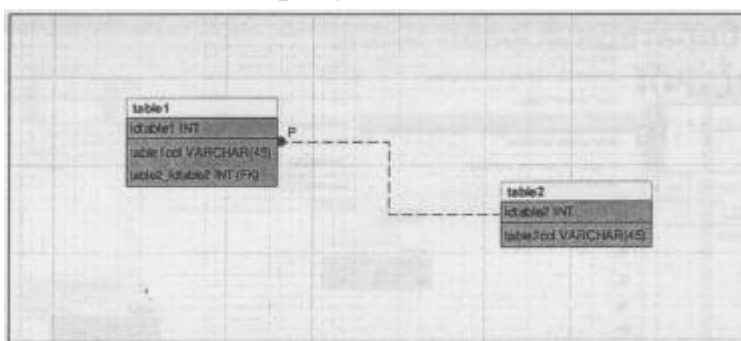
Далее произведем настройку нотаций. Выберем нотацию IDEF1X для отображения таблиц. Для этого вызовем в меню пункт Model —> Object Notation—> IDEF1X:



Также выберем нотацию IDEF1X для отображения связей. Для этого вызовем в меню пункт Model —> Relationship Notation —> IDEF1X:



После этого можно соединить таблицы. Очевидно, что между таблицами (сущностями) устанавливаются неидентифицирующие связи, поскольку каждая сущность имеет свой уникальный идентификатор — первичный ключ. Предположим, что связь между данными таблицами у нас один-ко-многим и первичный ключ таблицы 2 является внешним ключом таблицы 1. Тогда связь имеет вид, как показано на рисунке:



Построение тестовой ER-модели завершено.

## Лабораторная работа № 2

**Тема:** изучение базовых возможностей создания реляционной базы данных, построение SQL-запросов к ней.

**Цель:** формирование навыков по созданию базы данных с использованием операторов DDL языка запросов SQL в СУБД MySQL. Формирование навыков применения базовых операторов языка SQL (Select) для выбора данных из базы данных.

### Порядок выполнения работы №2:

1. Создать в MySQL Workbench отношения для БД «Сведения о результатах сдачи экзаменационной сессии», соответствующие следующим типам сущностей:
  - 1) Студент: номер зачетной книжки, ФИО, дата рождения.
  - 2) Преподаватель: идентификатор, ФИО.
  - 3) Предмет: идентификатор, название, семестр, преподаватель
  - 4) Оценки: связь студент-предмет-оценкаВыбрать ключи отношений, установить соответствующие связи между отношениями.
2. В таблицы внести несколько записей (5-10) с помощью редактора запросов и операции INSERT. Изменить записи с помощью операции UPDATE, удалить запись с помощью операции DELETE.
3. Написать запросы для вывода данных:
  - 1) Вывод всех предметов, упорядоченных по алфавиту.
  - 2) Вывод количества студентов.
  - 3) Вывод студентов, чья фамилия начинается на «Ива».
  - 4) Вывод студентов (фамилий), получивших оценки 5 по указанному предмету, упорядоченных по алфавиту в обратном порядке.
  - 5) Вывод студентов, набравших более 12 баллов по трем предметам (три выбранных предмета - у всех студентов должны быть одинаковыми).
4. С помощью процедуры обратного инжиниринга (обратного проектирования), создать информационную модель по созданной базе данных «Сведения о результатах сдачи экзаменационной сессии».

### Теоретические сведения

#### Оператор SELECT

Упрощенный синтаксис оператора SELECT:

SELECT

[ALL | DISTINCT]

*select\_expr* [, *select\_expr* ...]



[FROM *table\_references*  
[WHERE *where\_condition*]  
[GROUP BY {*col\_name* | *expr* | *position*}, ... [WITH ROLLUP]]  
[HAVING *where\_condition*]  
[ORDER BY {*col\_name* | *expr* | *position*}]  
[ASC | DESC]]  
[LIMIT {[*offset*,] *row\_count* | *row\_count* OFFSET *offset*}]

SELECT применяется для извлечения строк, выбранных из одной или нескольких таблиц. Выражение *select\_expr* задает столбцы, в которых необходимо проводить выборку. Если требуется получить только уникальные строки, то можно использовать ключевое слово DISTINCT. Помимо DISTINCT может применяться также ключевое слово ALL (все строки), которое принимается по умолчанию. Кроме того, оператор SELECT можно использовать для извлечения строк, вычисленных без ссылки на какую-либо таблицу.

Например:

```
mysql> SELECT 1 + 1;  
-> 2
```

Выражение FROM *table\_references* задает таблицы, из которых надлежит извлекать строки. Если указано имя более чем одной таблицы, следует выполнить объединение (JOIN).

Выражение WHERE, если оно задано, указывает условие или условия, которым должны удовлетворять строки. *where\_condition* - это выражение, которое оценивается как истинное для каждой выбранной строки. Оператор SELECT выбирает все строки, если нет предложения WHERE.

Выражение GROUP BY используется для определения групп выходных строк, к которым могут применяться агрегатные функции (COUNT, MIN, MAX, AVG и SUM). Если это предложение отсутствует, и используются агрегатные функции, то все столбцы с именами, упомянутыми в SELECT, должны быть включены в агрегатные функции, и эти функции будут применяться ко всему набору строк, которые удовлетворяют предикату запроса. Если при наличии предложения GROUP BY, в предложении SELECT отсутствуют агрегатные функции, то запрос просто вернет по одной строке из каждой группы. Эту возможность, наряду с ключевым словом DISTINCT, можно использовать для исключения дубликатов строк в результирующем наборе. Для получения значения функции агрегации, примененной к множеству полученных групп, в новой строке используется выражение WITH ROLLUP.

Выражение HAVING применяется после группировки для фильтрации групп по значениям агрегатных функций.



Выражение ORDER BY выполняет сортировку по любому количеству полей, указанных в предложении SELECT. При этом в списке полей могут указываться как имена полей, так и их порядковые позиции в списке предложения SELECT. Сортировку можно проводить по возрастанию (параметр ASC принимается по умолчанию) или по убыванию (параметр DESC).

Выражение LIMIT может использоваться для ограничения количества строк, возвращенных командой SELECT. LIMIT принимает один или два числовых аргумента. Эти аргументы должны быть целочисленными константами. Если заданы два аргумента, то первый указывает на смещение относительно первой возвращаемой строки, а второй задает максимальное количество возвращаемых строк.

## **Оператор INSERT**

Упрощенный синтаксис оператора INSERT:

```
INSERT [IGNORE]
[INTO] tbl_name
[(col_name [, col_name] ...)]
{VALUES | VALUE} (value_list) [, (value_list)] ...
[ON DUPLICATE KEY UPDATE assignment_list]
ИЛИ
INSERT [IGNORE]
[INTO] tbl_name
SET assignment_list
[ON DUPLICATE KEY UPDATE assignment_list]
ИЛИ
INSERT [IGNORE]
[INTO] tbl_name
[(col_name [, col_name] ...)]
SELECT ...
[ON DUPLICATE KEY UPDATE assignment_list]
```

Оператор INSERT вставляет новые строки в существующую таблицу. Форма данной команды INSERT... {VALUES | VALUE} вставляет строки в соответствии с точно указанными в команде значениями. При использовании формы данной команды INSERT... SET каждому полю, присутствующему в таблице, присваивается значение в виде имя поля = 'значение'. Форма INSERT... SELECT вставляет строки, выбранные из другой таблицы или таблиц.

Если некоторые поля таблицы имеют ключи PRIMARY или UNIQUE, и производится вставка новой строки, в которой эти поля имеют дублирующее

значение, то действие команды аварийно завершается и выдается ошибка №1062 ("Duplicate entry 'val' for key N"). Если в команде INSERT указано ключевое слово IGNORE, то вставка записей не прерывается, а строки с дублирующими значениями просто не вставляются.

Если некоторые поля таблицы имеют ключи PRIMARY или UNIQUE, и производится вставка новой строки, в которой эти поля имеют дублирующее значение, то при использовании выражения ON DUPLICATE KEY UPDATE будет выполнена операция UPDATE для такой строки.

## Оператор UPDATE

Упрощенный синтаксис оператора UPDATE:

UPDATE [IGNORE] *table\_reference*

SET *assignment\_list*

[WHERE *where\_condition*]

[ORDER BY ...]

[LIMIT *row\_count*]

Оператор UPDATE обновляет столбцы в соответствии с их новыми значениями в строках существующей таблицы. В выражении SET указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении WHERE, если оно присутствует, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Если задано выражение ORDER BY, то строки будут обновляться в указанном в нем порядке.

Если указывается ключевое слово IGNORE, то команда обновления не будет прервана, даже если при обновлении возникнет ошибка дублирования ключей. Строки, из-за которых возникают конфликтные ситуации, обновлены не будут.

Выражение LIMIT ограничивает число обновляемых строк.

## Оператор DELETE

Упрощенный синтаксис оператора DELETE:

DELETE [IGNORE] FROM *tbl\_name*

[WHERE *where\_condition*]

[ORDER BY ...]

[LIMIT *row\_count*]

ИЛИ

DELETE [IGNORE]

*tbl\_name*[.\*] [, *tbl\_name*[.\*]] ...

FROM *table\_references*

[WHERE *where\_condition*]

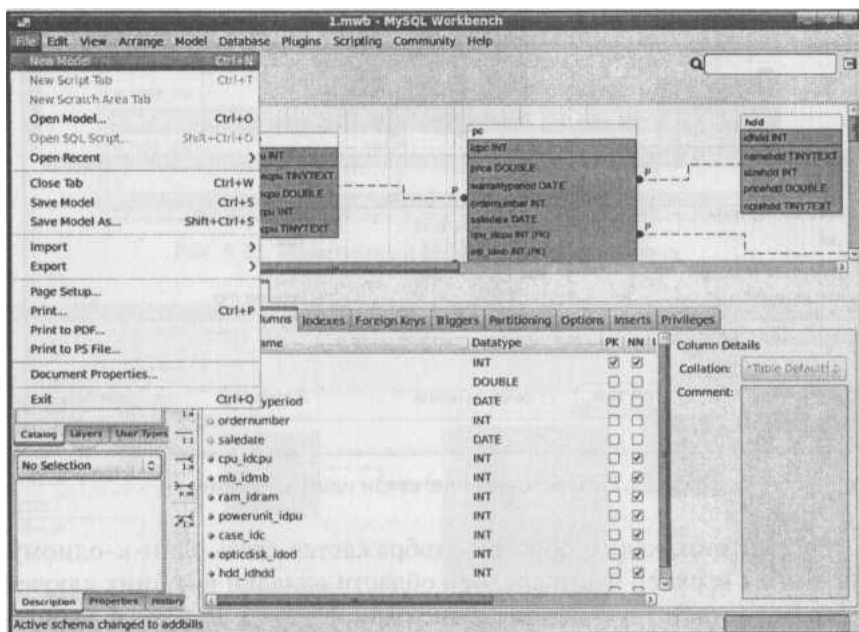
Оператор DELETE удаляет из таблицы *tbl\_name* строки, удовлетворяющие заданным в *where\_condition* условиям, и возвращает число удаленных записей. Если оператор DELETE запускается без определения WHERE, то удаляются все строки. Если применяется выражение ORDER BY, то строки будут удалены в указанном порядке. Выражение LIMIT ограничивает число удаляемых строк.

Можно указать несколько таблиц в операторе DELETE для удаления строк из одной или нескольких таблиц в зависимости от условия в предложении WHERE. В данном случае нельзя использовать выражения ORDER BY или LIMIT.

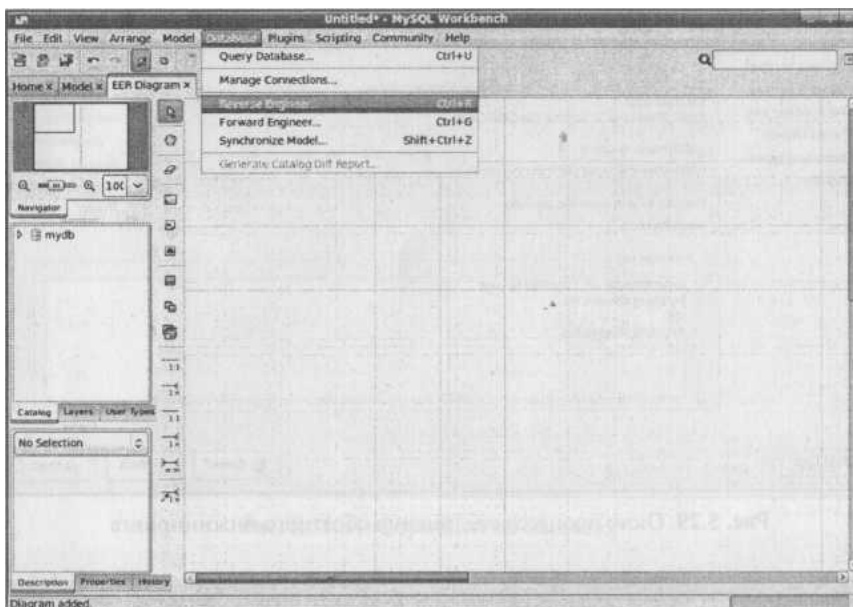
## Обратное проектирование (Reverse engineering)

Обратное проектирование, т. е. восстановление информационной модели по существующей базе данных. Это бывает необходимо при расширении (или модификации) существующей структуры, особенно в тех случаях, когда база данных была создана без необходимой сопроводительной документации. После завершения процесса восстановления ER-модели таблицы автоматически создаются на диаграмме. Теперь можно выполнять модификации уже с использованием логической схемы — добавлять сущности, атрибуты, связи и т. д. По завершении изменений можно синхронизировать модель с базой данных, что актуализирует все проведенные изменения.

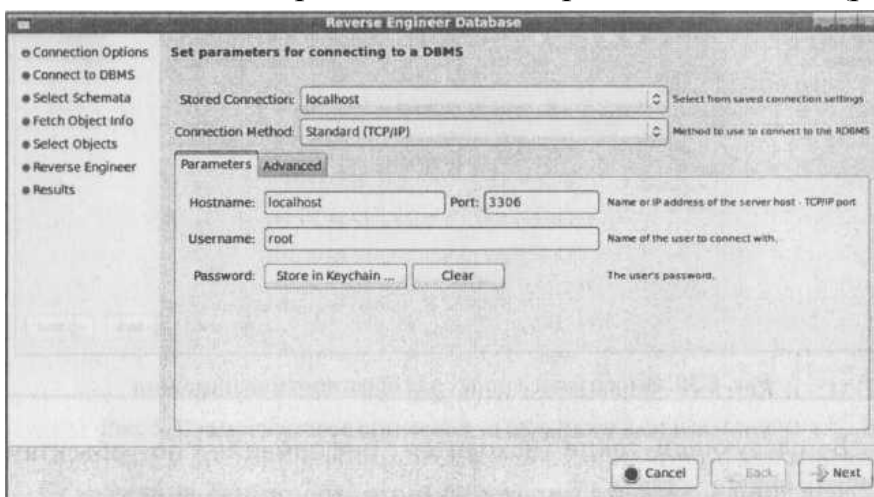
Рассмотрим более подробно процесс обратного инжиниринга. Создадим новую модель через меню File:



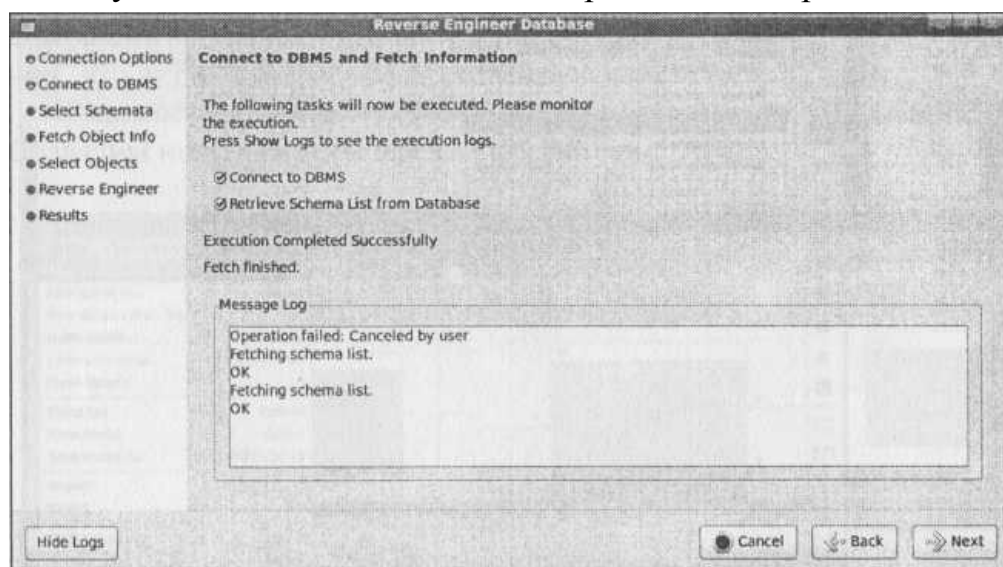
Далее следует вызвать пункт меню Database —> Reverse Engineer:



После чего откроется окно настроек соединения (рис. 5.28):



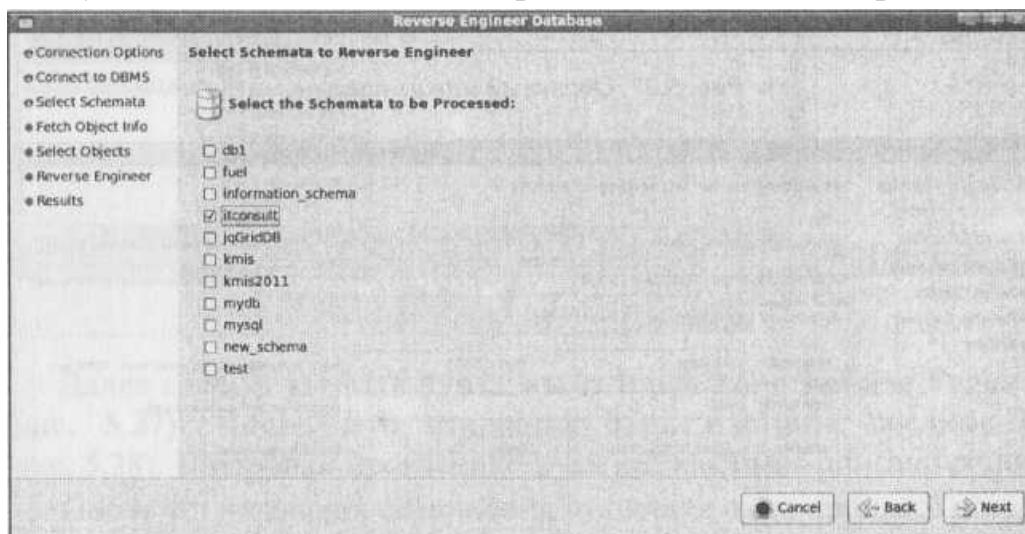
После установления соединения откроется окно процесса соединения:



Когда процесс будет завершен, процессы Connect to DBMS и Retrieve Schema List from Database будут помечены галочками. В поле Message Log появятся сообщения об успешном завершении или возникших проблемах. Если выполнение

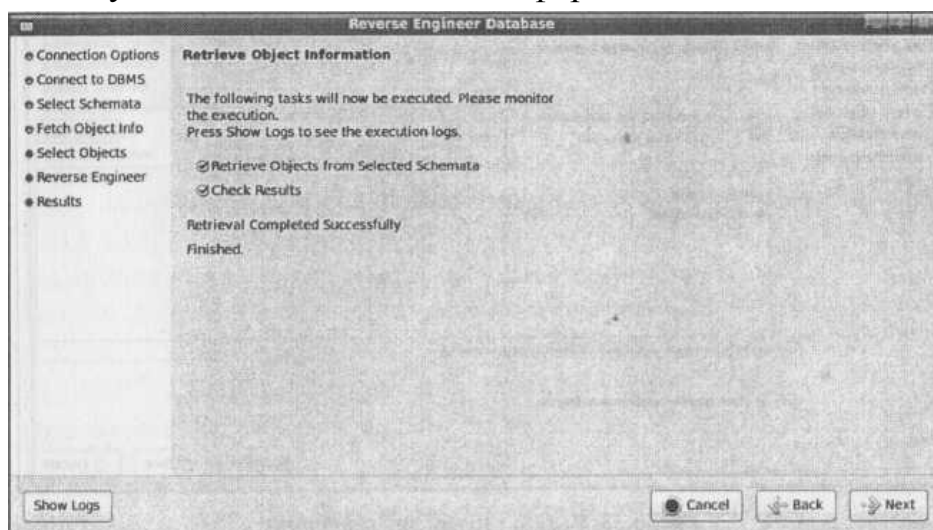
успешно завершено (Execution Completed Successfully), необходимо перейти к следующему шагу, нажав кнопку Next.

Следующий шаг состоит в выборе базы данных для обратного инжиниринга:

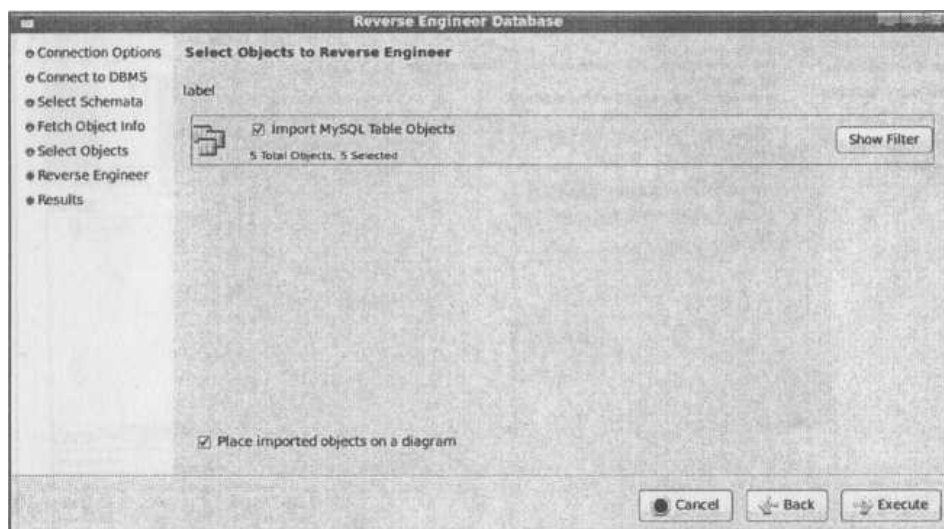


Из имеющегося списка баз данных, расположенных на выбранном сервере (в данном случае настройки проведены для сервера localhost, поэтому доступны все базы, расположенные на локальном сервере), следует выбрать ту базу данных, обратный инжиниринг которой необходимо сделать. Нажать кнопку Next и перейти на следующий шаг.

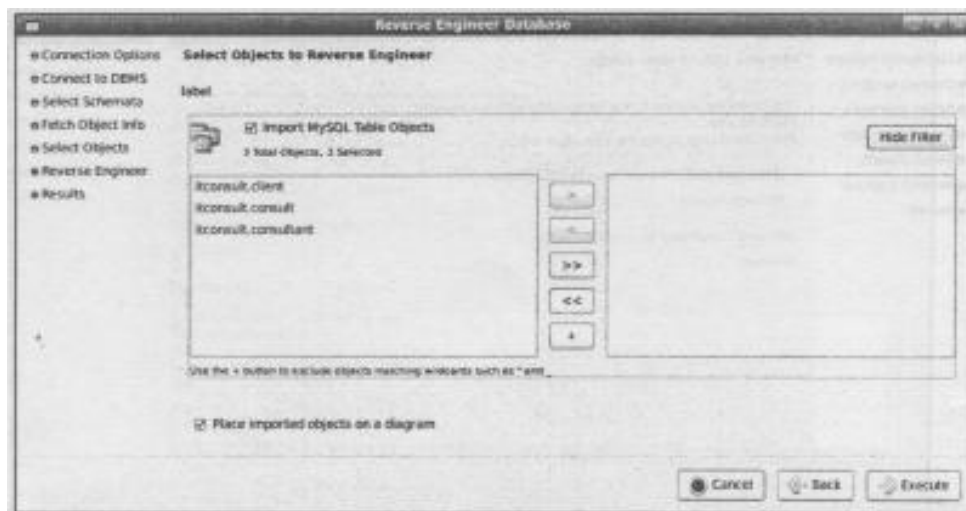
В следующем окне находится информация по объектам:



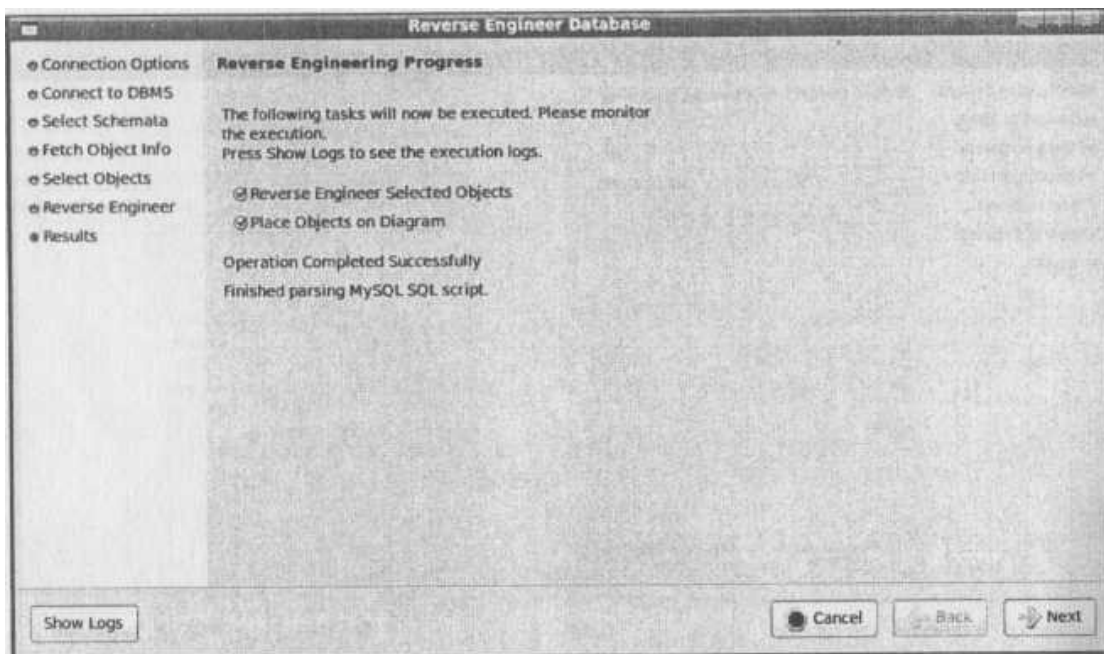
После нажатия кнопки Next открывается окно выбора объектов для обратного инжиниринга:



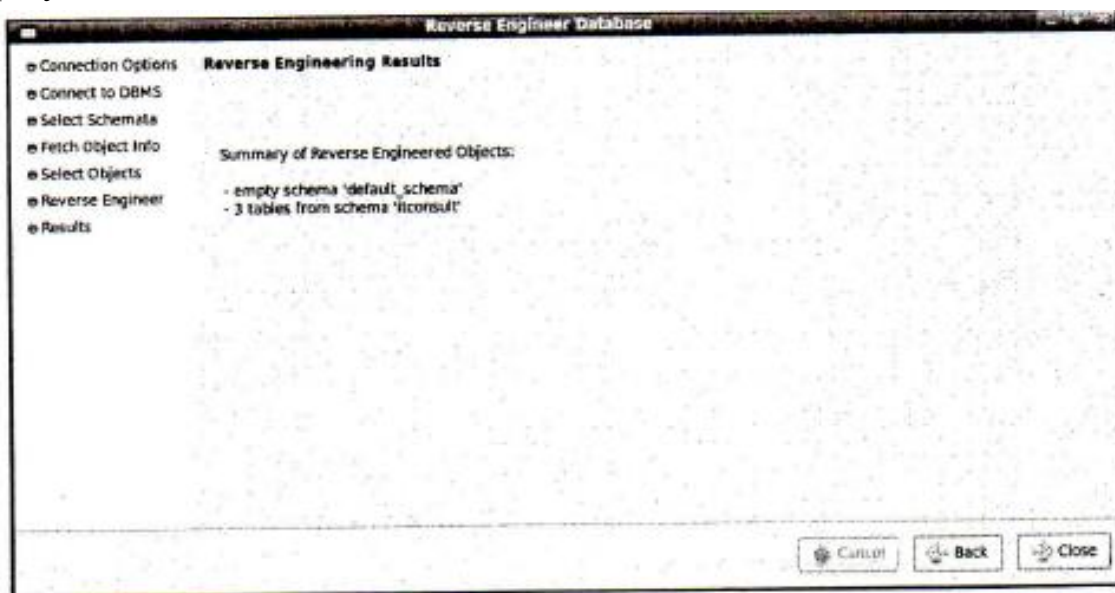
Если есть необходимость генерировать не все таблицы, а только некоторые, то, нажав на кнопку **Show Filter**, можно выбрать, какие именно таблицы будут сгенерированы:



Осуществив выбор объектов, переходим на следующий шаг — собственно генерацию процесса обратного инжиниринга. После того как на экране появится сообщение, что процесс успешно завершен, можно нажать кнопку **Next**. Если в процессе обратного инжиниринга возникли проблемы, можно нажать кнопку **Show Logs** и посмотреть сообщения:

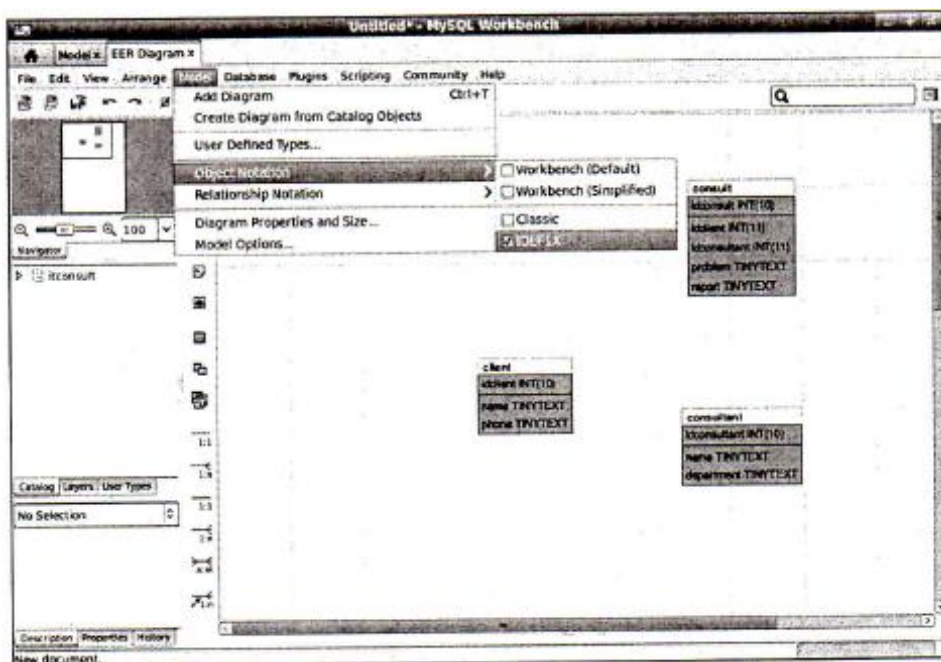


После завершения процесса обратного инжиниринга появится окно с результатами выполнения:



В данном случае в окне приведен отчет, в котором указано, что 3 таблицы из базы данных itconsult сгенерированы. Процесс создания ER-диаграммы базы данных прошел успешно. Нажав на кнопку Close, закрываем окно. Результат выполнения обратного инжиниринга:

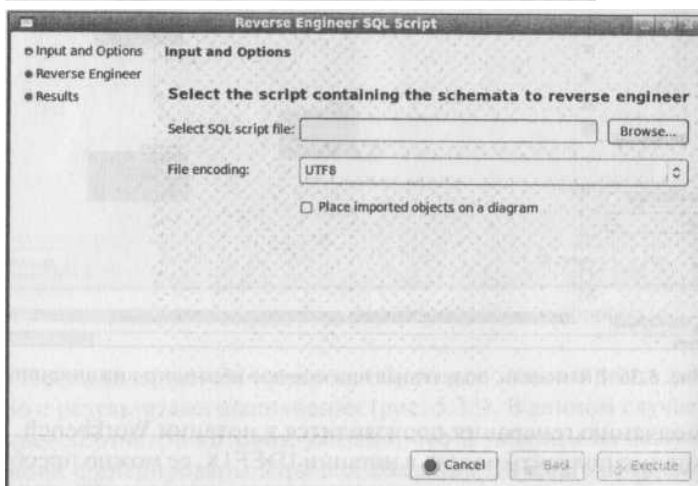
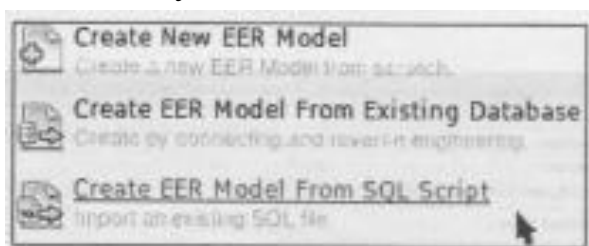




По умолчанию генерация производится в нотации Workbench. Если необходимо получить модель в нотации IDEFIX, ее можно преобразовать при помощи вызова соответствующего пункта меню.

При необходимости полученную ER-модель можно дорабатывать, изменять, удалять, добавлять таблицы, менять поля и т. п. Однако следует заметить, что валидация моделей доступна только в коммерческой версии MySQL.

Если существует необходимость сгенерировать ER-модель из SQL скрипта, то при выборе соответствующего пункта меню на главной странице Create EER Model From SQL Script открывается окно. В этом окне через браузер можно выбрать файл, в котором хранится SQL скрипт. Далее процесс создания ER-модели аналогичен описанному выше:





Таким образом, рассмотренные принципы построения ER-моделей позволяют в значительной мере упростить процесс проектирования баз данных и сделать его более наглядным.

### Лабораторная работа № 3

**Тема:** создание физической модели базы данных из ER-модели (прямой инжиниринг) и построение SQL-запросов к ней.

**Цель:** формирование навыков по созданию базы данных в СУБД MySQL, использованию инструментов языка запросов SQL.

#### Порядок выполнения работы №3:

1. С использованием прямого инжиниринга экспортировать ER-модель, созданную в лабораторной работе №1, на MySQL сервер.
2. Написать запросы добавления данных в таблицы (минимум 10 записей для каждой таблицы).
3. Создать SQL-запросы согласно вариантам задания, выполняющие основные требования к функциям системы.
4. Придумайте и напишите SQL запросы, которые будут необходимы для предметной области (в соответствии с вариантом задания):
  - 1) Запрос на выборку избранных полей таблицы, с использованием синонима (алиаса) и сортировкой записей (ORDER BY).
  - 2) Запрос с использованием сортировки (ORDER BY) и группировки (GROUP BY).
  - 3) Запрос с использованием предложения DISTINCT.
  - 4) Запрос с использованием операций сравнения.
  - 5) Запросы для предикатов: IN, BETWEEN, LIKE, IS NULL.
  - 6) Запросы с использованием агрегатных функций (COUNT, SUM, AVG, MAX, MIN ), производящие обобщенную групповую обработку значений полей (используя ключевые фразы GROUP BY и HAVING).
  - 7) Запрос на выборку данных из двух связанных таблиц. Выбрать несколько полей, по которым сортируется вывод.
  - 8) Многотабличный запрос с использованием внутреннего и внешнего соединения.
  - 9) Многотабличный запрос с использованием оператора UNION.
  - 10) Запрос с применением подзапроса в части WHERE команды SQL SELECT и в внутри предложения HAVING.
  - 11) Запрос с применением подзапроса с применением следующих операторов: ALL, EXIST, ANY.

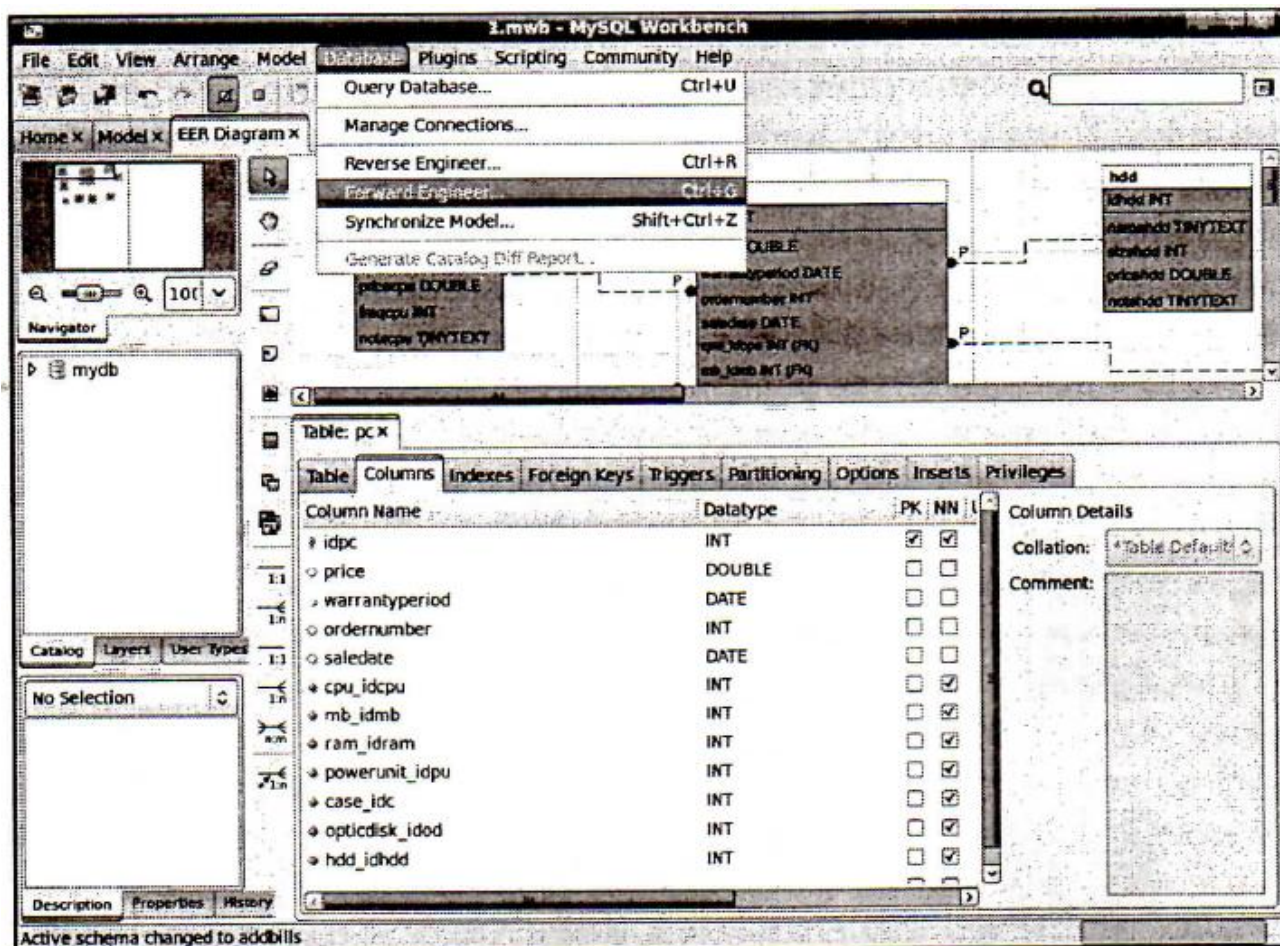
## Теоретические сведения

### Прямой инжиниринг

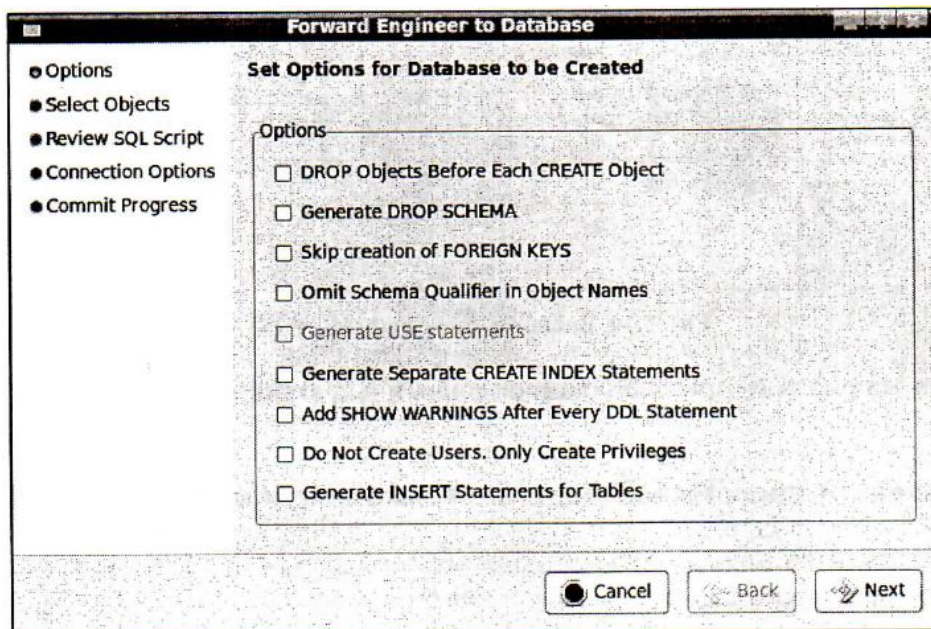
Для того чтобы построить функционирующую модель, необходимо воспользоваться функцией прямого инжиниринга.

Прямой инжиниринг предназначен для экспорта ER-модели на MySQL сервер.

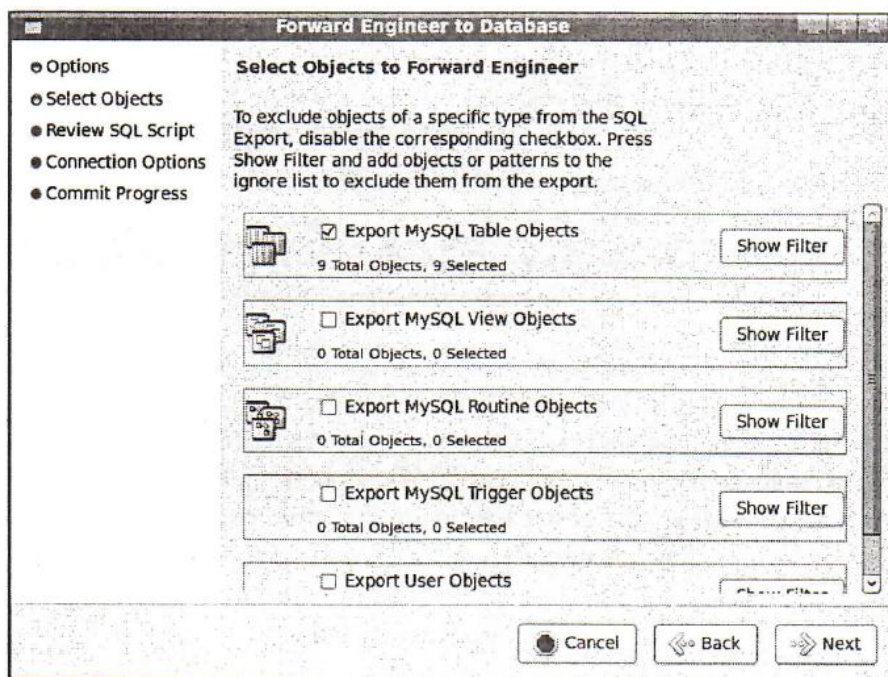
Выберем, как показано на рисунке, меню **Database - Forward Engineering**:



Далее будет открыто меню опций, в котором можно выбрать требуемые опции. При наведении на опции курсора возникает всплывающая подсказка, которая поясняет смысл выбираемой опции. Если нет необходимости, можно не выбирать опции, а сразу нажать на кнопку **Next**.

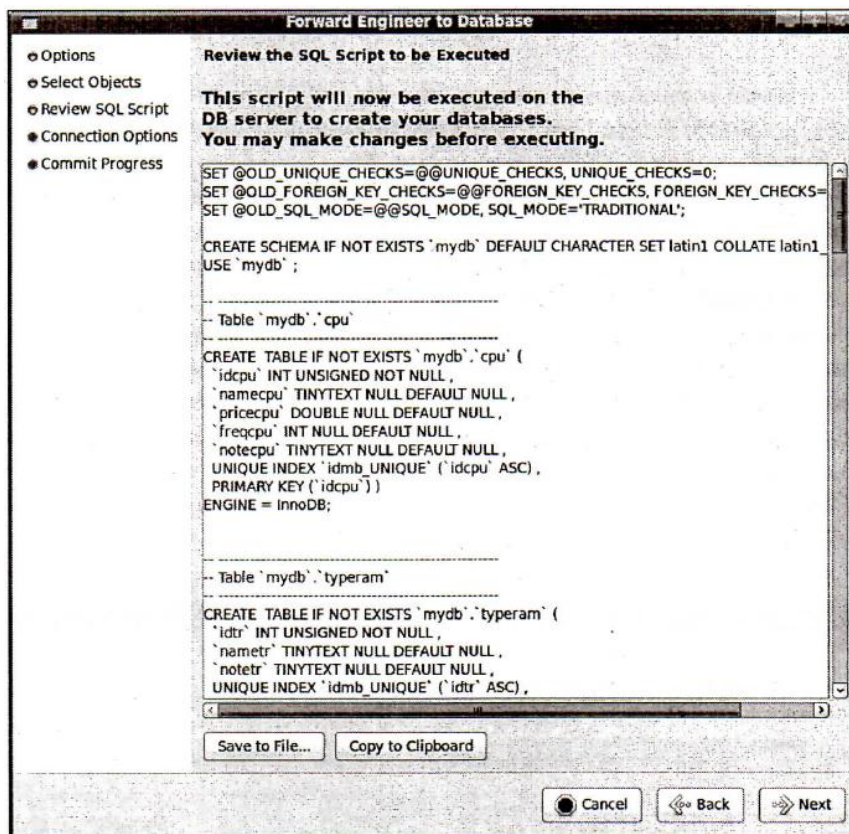


Следующим шагом является выбор объектов. Здесь можно выбрать опции, связанные с объектами, и посмотреть настройки фильтров. В данном случае выбирается только создание объектов-таблиц (9 штук), поскольку никаких других объектов в рамках ER-модели создано не было. В том случае, если были бы созданы иные объекты, например триггеры, следовало бы поставить галочку в соответствующем поле для их генерации при создании базы данных на сервере



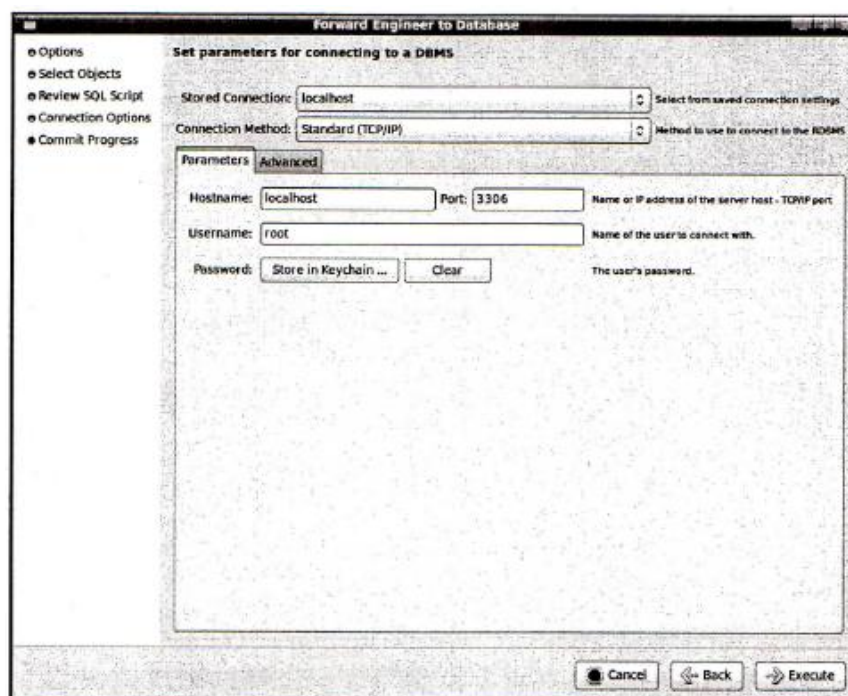
Далее можно осуществить предварительный просмотр SQL скрипта:



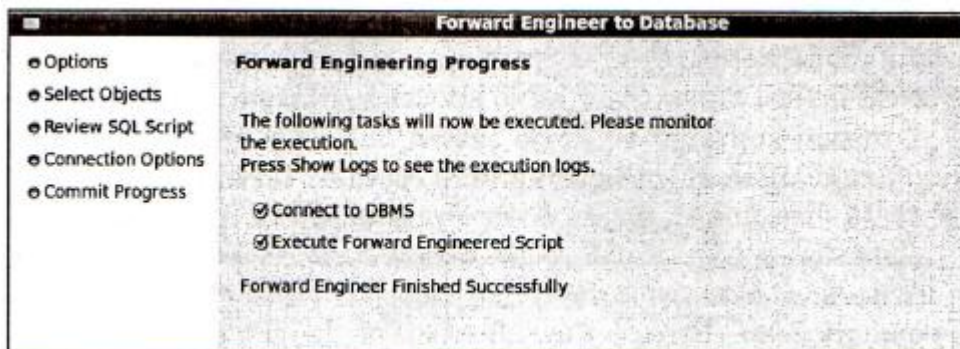


Если нет необходимости вносить изменения в сгенерированный код, тогда можно переходить на следующий этап, на котором по данному коду будет создана база данных. Не забудьте по мере построения модели периодически сохранять результаты работы. Первоначально, чтобы сохранить модель, необходимо выбрать пункт меню **Save as**, затем следует выбирать просто **Save**.

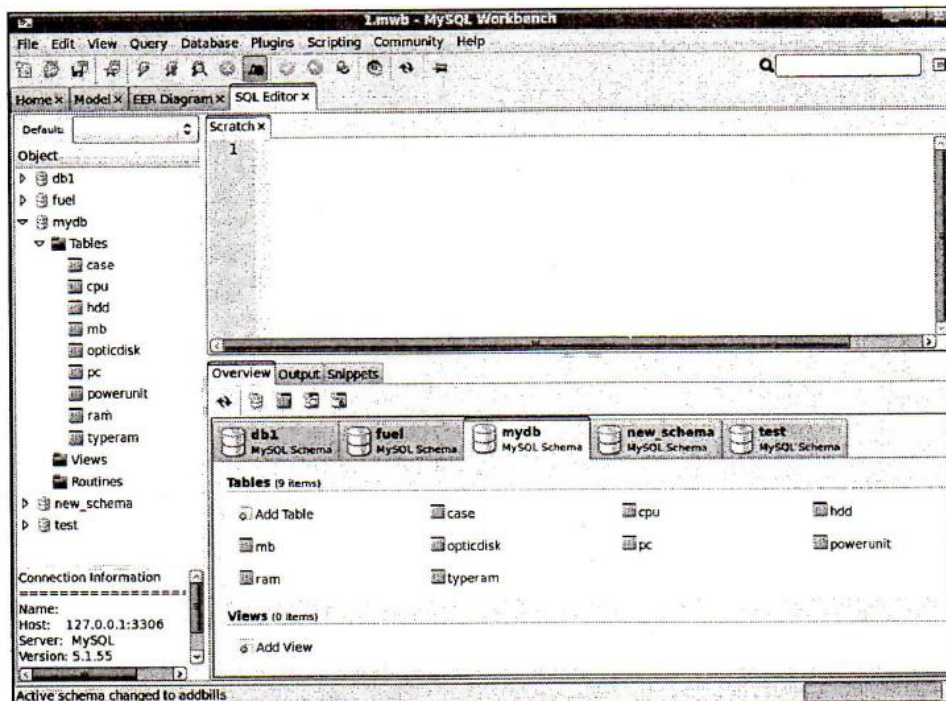
Следующий шаг заключается в настройке параметров соединения с сервером базы данных:



В данном случае следует выбрать соединение с локальным сервером — localhost. Также необходимо проверить, что настроен протокол TCP/IP. Остальные настройки, такие как имя пользователя и пароль, проводятся по необходимости. Если все настройки завершены, для окончания генерации базы данных необходимо нажать кнопку **Execute**. После ее нажатия появится окно процесса генерации:



Нажимаем кнопку **Close** внизу страницы. По завершении генерации новая база данных будет создана из ER-модели. Ее можно будет увидеть во вкладке объектов слева и во вкладке **Overview** внизу.



## Оператор JOIN

Синтаксис:

table\_reference, table\_reference

table\_reference [CROSS] JOIN table\_reference

table\_reference INNER JOIN table\_reference join\_condition

table\_reference STRAIGHT\_JOIN table\_reference

table\_reference LEFT [OUTER] JOIN table\_reference join\_condition

table\_reference LEFT [OUTER] JOIN table\_reference

table\_reference NATURAL [LEFT [OUTER]] JOIN table\_reference

table\_reference RIGHT [OUTER] JOIN table\_reference join\_condition

table\_reference RIGHT [OUTER] JOIN table\_reference

table\_reference NATURAL [RIGHT [OUTER]] JOIN table\_reference

где table\_reference определено, как:

table\_name [[AS] alias]

и join\_condition определено, как:

ON conditional\_expr |

USING (column\_list)

Оператор JOIN позволяет извлекать данные из нескольких таблиц без создания временных таблиц и за один запрос.

Условный оператор ON представляет собой условие в любой форме из числа тех, которые можно использовать в выражении WHERE.

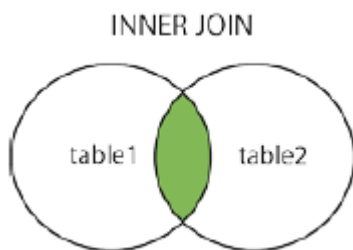
USING (column\_list) служит для указания списка столбцов, которые должны существовать в обеих таблицах. Такое выражение USING, как:

A LEFT JOIN B USING (C1, C2, C3, ...)

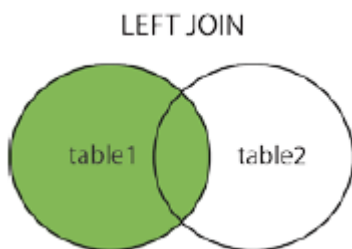
семантически идентично выражению ON, например:

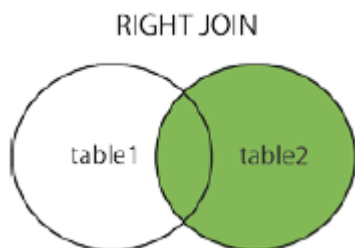
A LEFT JOIN B ON A.C1=B.C1 AND A.C2=B.C2 AND A.C3=B.C3,...

При внутреннем объединении (INNER JOIN) выбираются только совпадающие данные из объединяемых таблиц.



Чтобы получить данные, которые подходят по условию частично, необходимо использовать внешнее объединение - OUTER JOIN. Такое объединение вернет данные из обеих таблиц (совпадающие по условию объединения) ПЛЮС дополнит выборку оставшимися данными из внешней таблицы, которые по условию не подходят, заполнив недостающие данные значением NULL. Существует два типа внешнего объединения OUTER JOIN - LEFT OUTER JOIN и RIGHT OUTER JOIN.





Выражение NATURAL [LEFT] JOIN для двух таблиц определяется так, чтобы оно являлось семантическим эквивалентом INNER JOIN или LEFT JOIN с выражением USING, в котором указаны все столбцы, имеющиеся в обеих таблицах. INNER JOIN и , (запятая) являются семантическими эквивалентами. Оба осуществляют полное объединение используемых таблиц. Способ связывания таблиц обычно задается в условии WHERE.

STRAIGHT\_JOIN идентично JOIN, за исключением того, что левая таблица всегда читается раньше правой. Это выражение может использоваться для тех (немногих) случаев, когда оптимизатор объединения располагает таблицы в неправильном порядке.

## Оператор UNION

Синтаксис:

```
SELECT ... UNION [ALL | DISTINCT] SELECT ... [UNION [ALL | DISTINCT]
SELECT ...]
```

UNION используется для объединения результатов работы нескольких команд SELECT в один набор результатов. Каждое предложение SELECT в операторе UNION должно иметь одинаковое количество полей в наборах результатов с одинаковыми типами данных.

Пример:

```
SELECT film.title,film.length from film WHERE film.title LIKE 'cat%'
```

```
UNION
```

```
SELECT film.title,film.length from film WHERE film.length < 47
```

Результат:

	title	length
▶	CAT CONEHEADS	112
	CATCH AMISTAD	183
	ALIEN CENTER	46
	IRON MOON	46
	KWAI HOMEWARD	46
	LABYRINTH LEAGUE	46
	RIDGEMONT SLEMARINE	46



## Оператор WITH

Пример:

WITH

```
cte1 AS (SELECT a, b FROM table1),
```

```
cte2 AS (SELECT c, d FROM table2)
```

```
SELECT b, d FROM cte1 JOIN cte2
```

```
WHERE cte1.a = cte2.c;
```

В данном примере cte1 и cte2 являются общими табличными выражениями (common table expression). Общее табличное выражение (СТЕ) - это именованный временный набор результатов, который существует в области действия одного оператора и на который можно ссылаться позже в этом операторе, возможно, несколько раз.

Для определения СТЕ используется выражение WITH содержащее один или несколько разделенных запятыми выражений. Общие табличные выражения позволяют существенно уменьшить объем кода, если многократно приходится обращаться к одним и тем же производным таблицам.

## Лабораторная работа № 4

**Тема:** обработка данных в реляционных БД с помощью структурированного языка запросов SQL.

**Цель:** оттачивание навыков использования инструментов языка запросов SQL (выборка данных из таблиц и предоставление ее в надлежащем виде, добавление, удаление, редактирование информации).

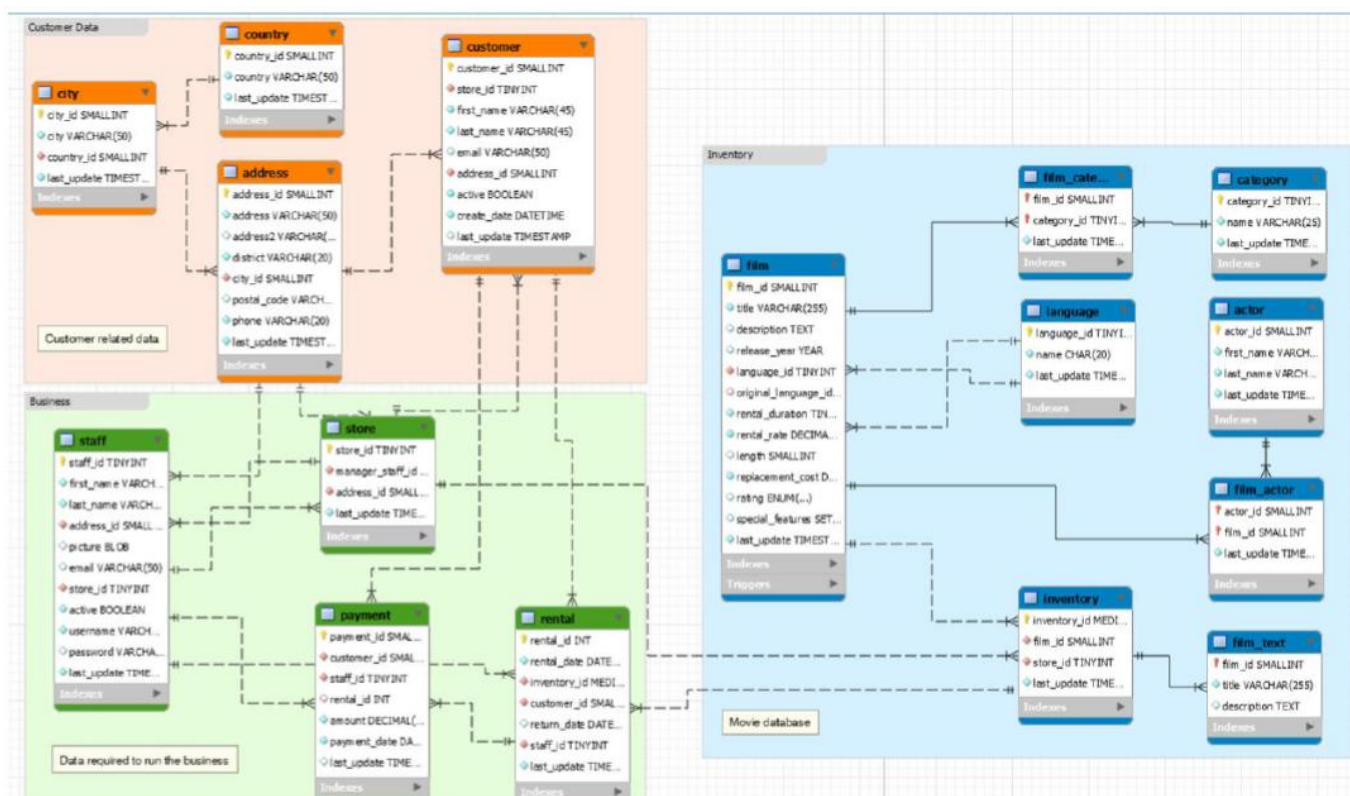
### Порядок выполнения работы №4:

Выполнить следующие запросы для получения данных из БД Sakila:

1. Вывести всех покупателей из указанного списка стран: отобразить имя, фамилию, страну.
2. Вывести все фильмы, в которых снимался указанный актер: отобразить название фильма, жанр.
3. Вывести топ 10 жанров фильмов по величине дохода в указанном месяце: отобразить жанр, доход.
4. Вывести список из 5 клиентов, упорядоченный по количеству купленных фильмов с указанным актером, начиная с 10-й позиции: отобразить имя, фамилию, количество купленных фильмов.
5. Вывести для каждого магазина его город, страну расположения и суммарный доход за первую неделю продаж.
6. Вывести всех актеров для фильма, принесшего наибольший доход: отобразить фильм, имя актера, фамилия актера.
7. Для всех покупателей вывести информацию о покупателях и актерах-однофамильцах (используя *LEFT JOIN*, если однофамильцев нет – вывести *NULL*).
8. Для всех актеров вывести информацию о покупателях и актерах-однофамильцах (используя *RIGHT JOIN*, если однофамильцев нет – вывести *NULL*).
9. В одном запросе вывести статистические данные о фильмах:
  - 10.11.1. Длина самого продолжительного фильма – отобразить значение длины; количество фильмов, имеющих такую продолжительность.
  - 11.11.2. Длина самого короткого фильма – отобразить значение длины; количество фильмов, имеющих такую продолжительность.
  - 12.11.3. Максимальное количество задействованных актеров в фильме – отобразить максимальное количество актеров; количество фильмов, имеющих такое число актеров.
  - 13.11.4. Минимальное количество задействованных актеров в фильме - отобразить минимальное количество актеров; количество фильмов, имеющих такое число актеров.

## СХЕМА БД SAKILA

База данных sakila представляет собой базу данных сети магазинов проката DVD фильмов. На рисунке представлена схема БД sakila:



БД sakila включает в себя 16 таблиц.

Информация о клиентах магазинов хранится в таблицах `customer`, `address`, `city`, `country`.

Данные о магазинах, в которых были совершены покупки, продавцах, платежах и арендных сроках содержатся в таблицах `store`, `staff`, `payment`, `rental`.

Остальные таблицы содержат информации о базе DVD фильмов сети магазинов: описание и характеристики фильмов (`film`), жанры фильмов (`film_category`, `category`), данные об актерах (`actor`, `film_actor`), данные о наличии фильмов в магазинах (`inventory`, `film_text`).

Подробное описание БД sakila представлено на сайте <https://dev.mysql.com/doc/sakila/en/>.

## Лабораторная работа № 5

**Тема:** работа с представлениями и транзакциями.

**Цель:** приобретение навыков создания представлений, изучение механизма транзакций.

**Порядок выполнения работы №5:**

### ПРЕДСТАВЛЕНИЯ

1. Создать два не обновляемых представления, возвращающих пользователю результат из нескольких таблиц, с разными алгоритмами обработки представления.
2. Создать обновляемое представление, не позволяющее выполнить команду INSERT.
3. Создать обновляемое представление, позволяющее выполнить команду INSERT.
4. Создать вложенное обновляемое представление с проверкой ограничений (WITH CHECK OPTION).

### ТРАНЗАКЦИИ

5. Выберите любую таблицу, созданную в предыдущих лабораторных работах.  
Создайте транзакцию, произведите ее откат и фиксацию:
  - a. Отключите режим автоматического завершения;
  - b. Добавьте в выбранную таблицу новые записи, проверьте добавились ли они;
  - c. Произведите откат транзакции, т. е. отмену произведенных действий;
  - d. Откатите транзакцию оператором ROLLBACK(изменения не сохранились);
  - e. Воспроизведите транзакцию и сохраните действия оператором COMMIT.
6. Продемонстрировать возможность чтения незафиксированных изменений при использовании уровня изоляции READ UNCOMMITTED и отсутствие такой возможности при уровне изоляции READ COMMITTED.
7. Продемонстрировать возможность записи в уже прочитанные данные при использовании уровня изоляции READ COMMITTED и отсутствие такой возможности при уровне изоляции REPEATABLE READ.

### Теоретические сведения

#### Представления

<https://dev.mysql.com/doc/refman/8.0/en/views.html>

Представление (VIEW) - объект базы данных, представляющий собой виртуальную таблицу. В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Преимущества использования представлений:

1. Дает возможность гибкой настройки прав доступа к данным за счет того, что права даются не на таблицу, а на представление. Это позволяет назначить права на отдельные строки таблицы или возможность получения не самих данных, а результата каких-то действий над ними.
2. Позволяет разделить логику хранения данных и программного обеспечения. Можно менять структуру данных, не затрагивая программный код. При изменении схемы БД достаточно создать представления, аналогичные таблицам, к которым раньше обращались приложения. Это очень удобно, когда нет возможности изменить программный код или к одной базе данных обращаются несколько приложений с различными требованиями к структуре данных.
3. Удобство в использовании за счет автоматического выполнения таких действий как доступ к определенной части строк и/или столбцов, получение данных из нескольких таблиц и их преобразование с помощью различных функций.
4. Поскольку SQL-запрос, выбирающий данные представления, зафиксирован на момент его создания, СУБД получает возможность применить к этому запросу оптимизацию или предварительную компиляцию, что может повысить производительность выполнения запросов.

Создание представления (упрощенный синтаксис):

```
CREATE [OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Данное выражение создает новое представление или заменяет существующее, если используется выражение OR REPLACE. Выражение *select\_statement* обеспечивает определение представления. SELECT может относиться как к таблицам, так и к другим представлениям, т.е. могут быть вложенными.

Представления должны иметь уникальные имена столбцов. По умолчанию имена столбцов, полученные выражением SELECT, используются для имени столбца представления. Явные имена для столбцов представления могут быть заданы с помощью выражения *column\_list* как список разделяемых запятой идентификаторов.

Опция ALGORITHM определяет, как MySQL обрабатывает представления:

- MERGE: текст запроса к представлению объединяется с текстом запроса самого представления к таблицам БД, результат выполнения объединенного запроса возвращается пользователю.
- TEMPTABLE: содержимое представлению сохраняется во временную таблицу, которая затем используется для выполнения запроса.
- UNDEFINED (значение по умолчанию): MySQL самостоятельно выбирает какой алгоритм использовать при обращении к представлению.

Представление называется *обновляемым*, если к нему могут быть применимы операторы UPDATE и DELETE для изменения данных в таблицах, на которых основано представление. Для того чтобы представление было обновляемым, должны быть выполнены некоторые условия, в т.ч.:

1. Отсутствие функций агрегации в представлении.
2. Отсутствие следующих выражений в представлении: DISTINCT, GROUP BY, HAVING, UNION.
3. Отсутствие подзапросов в списке выражения SELECT
4. Колонки представления быть простыми ссылками на поля таблиц (а не, например, арифметическими выражениями) и т.д.

Обновляемое представление может допускать добавление данных (INSERT), если все поля таблицы-источника, не присутствующие в представлении, имеют значения по умолчанию. Для представлений, основанных на нескольких таблицах, операция добавления данных (INSERT) работает только в случае если происходит добавление в единственную реальную таблицу. Удаление данных (DELETE) для таких представлений не поддерживается.

При использовании в определении представления конструкции WITH [CASCADED | LOCAL] CHECK OPTION все добавляемые или изменяемые строки будут проверяться на соответствие определению представления.

- Изменение данных (UPDATE) разрешено только для строк, удовлетворяющих условию WHERE в определении представления. Кроме того, новые значения строки также должны удовлетворять значениями удовлетворяет условию WHERE.
- Добавление данных (INSERT) будет происходить, только если новая строка удовлетворяет условию WHERE в определении представления.

Ключевые слова **CASCADED** и **LOCAL** определяют глубину проверки для представлений, основанных на других представлениях:

- Для **LOCAL** происходит проверка условия **WHERE** только в собственном определении представления.
- Для **CASCADED** происходит проверка для всех представлений на которых основано данное представление. Значением по умолчанию является **CASCADED**.

В начале выполнения работы следует выполнить запрос

```
SET SQL_SAFE_UPDATES = 0
```

При значении параметра **SQL\_SAFE\_UPDATES = 1** выполнение команды **UPDATE** возможно только в том случае, если в запросе указан первичный ключ.

## Транзакции

<https://dev.mysql.com/doc/refman/8.0/en/sql-syntax-transactions.html>

Транзакция — это операция, состоящая из одного или нескольких запросов к базе данных. Суть транзакций — обеспечить корректное выполнение всех запросов в рамках одной транзакции, а также обеспечить механизм изоляции транзакций друг от друга для решения проблемы совместного доступа к данным. Любая транзакция либо выполняется полностью, либо не выполняется вообще.

В транзакционной модели есть две базовых операции: **COMMIT** и **ROLLBACK**. **COMMIT** выполняет фиксацию всех изменений в транзакции. **ROLLBACK** отменяет (откатывает) изменения, произошедшие в транзакции.

При старте транзакции все последующие изменения сохраняются во временном хранилище. В случае выполнения **COMMIT**, все изменения, выполненные в рамках одной транзакции, сохраняются в физическую БД. В случае выполнения **ROLLBACK** произойдет откат и все изменения, выполненные в рамках этой транзакции, не сохраняются.

По умолчанию MySQL работает в режиме автоматического завершения транзакций, т. е. как только выполняется оператор обновления данных, который модифицирует таблицу, изменения тут же сохраняются на диске. Чтобы объединить операторы в транзакцию, следует отключить этот режим: `set AUTOCOMMIT=0`. Также отключить режим автоматического завершения транзакций для отдельной последовательности операторов можно оператором **START TRANSACTION**.

Для некоторых операторов нельзя выполнить откат с помощью **ROLLBACK**. Это операторы языка определения данных (Data Definition Language). Сюда входят запросы **CREATE**, **ALTER**, **DROP**, **TRUNCATE**, **COMMENT**, **RENAME**.

Следующие операторы неявно завершают транзакцию (как если бы перед их выполнением был выдан **COMMIT**): **ALTER FUNCTION**, **ALTER PROCEDURE**, **ALTER TABLE**, **ALTER VIEW**, **CREATE DATABASE**, **CREATE FUNCTION**,

CREATE INDEX, CREATE PROCEDURE, CREATE TABLE, CREATE TRIGGER, CREATE VIEW, DROP DATABASE, DROP FUNCTION, DROP INDEX, DROP PROCEDURE, DROP TABLE, DROP TRIGGER, DROP VIEW, RENAME TABLE, TRUNCATE TABLE, BEGIN, SET autocommit = 1, START TRANSACTION.

Транзакция может быть разделена на точки сохранения. Оператор SAVEPOINT identifier\_name устанавливает именованную точку сохранения транзакции с именем идентификатора. Если текущая транзакция имеет точку сохранения с тем же именем, старая точка сохранения удаляется, а новая устанавливается. Оператор ROLLBACK TO SAVEPOINT identifier\_name откатывает транзакцию до указанной точки сохранения без прерывания транзакции. Изменения, которые выполняются в текущей транзакции для строк после установки точки сохранения, отменяются при откате. Для удаления одной или нескольких точек сохранения используется команда RELEASE SAVEPOINT identifier\_name.

Уровни изоляций транзакций с разной степенью обеспечивают целостность данных при их одновременной обработке множеством процессов (пользователей). В MySQL существует 4 уровня изоляции транзакций:

- READ UNCOMMITTED — самый низкий уровень изоляции. При этом уровне возможно чтение незафиксированных изменений параллельных транзакций.
- READ COMMITTED — на этом уровне возможно чтение данных только зафиксированных транзакций. Однако, возможна запись в уже прочитанные внутри транзакции данные.
- REPEATABLE READ — на этом уровне изоляции так же возможно чтение данных только зафиксированных транзакций. Так же на этом уровне отсутствует проблема «Неповторяемого чтения», то есть строки, которые участвуют в выборке в рамках транзакции, блокируются и не могут быть *изменены* другими параллельными транзакциями. Но таблицы целиком не блокируются и возможно фантомное чтение — одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. Другая транзакция в интервалах между этими выборками *добавляет* или *удаляет* строки, используемые в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк. Однако в MySQL проблема фантомного чтения решена на данном уровне изоляции: все чтения данных в пределах одной транзакции используют «снимок» данных, полученный при первом чтении внутри этой транзакции.
- SERIALIZABLE — максимальный уровень изоляции, гарантирует неизменяемость данных другими процессами до завершения транзакции. Но в то



же время является самым медленным. В MySQL этот уровень изоляции схож с REPEATABLE READ, однако все простые операторы SELECT неявно преобразуются к виду SELECT ... FOR SHARE, если режим автоматического завершения транзакций выключен.

По умолчанию в MySQL установлен уровень изоляции REPEATABLE READ. Для смены уровня изоляции используется оператор SET TRANSACTION. Этот оператор устанавливает уровень изоляции следующей транзакции, глобально либо только для текущего сеанса.

SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL

{READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |  
SERIALIZABLE }

## **Литература**

1. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: учебное пособие/С.А. Мартишин, В.Л. Симонов, М.В. Храпченко.- Москва: ИД «Форум»:ИНФРА-М, 2021.-160с.
2. Базы данных: практикум/ А.С. Копырин. - Москва: ФЛИНТА. 2021. -106с.
3. MySQL 5.0. Библиотека программиста / В. Гольцман — «Питер», 2010.-268с.

## **Рекомендуемая литература**

1. Кузнецов С.Д. Основы баз данных : учебное пособие.
2. Дейт К. Дж. Введение в системы баз данных, 8-е издание: Пер. с англ. –М.: Издательский дом «Вильямс», 2005. -1328 с.
3. Крёнке Д. Теория и практика построения баз данных. 8-е изд. –СПб.: Питер, 2003. -800 с.
4. Карпова Т.С. Базы данных: модели, разработка, реализация. –СПб.: Питер, 2001. -304 с.
5. Грубер М. Понимание SQL /Перевод с английского. –М.: Финансы и статистика, 1993. -289 с.
6. Малыхина М. П. Базы данных: основы, проектирование, использование. –СПб.: БХВ-Петербург, 2004. –512 с.
7. Голицина О. Л, Максимов Н. В., Попов И. И. Базы данных: Учебное пособие. – М.: ФОРУМ: ИНФА-М, 2003. –352 с.
8. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных. –М.: КОРОНА принт, 2002. –672 с.
9. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. -СПб.: ИТМО, 1994. –90 с.