МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(национальный исследовательский университет)»

Базы данных и экспертные системы. Лабораторный практикум

Электронное методическое пособие

САМАРА
2011

004

Б 179

Составитель: **Логанова Лилия Владимировна**

Базы данных и экспертные системы. Лабораторный практикум
[Электронный ресурс] : электрон. метод. пособие / М-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); сост. Л. В. Логанова. - Электрон. текстовые и граф. дан. (1 Мбайт). - Самара, 2011. - 1 эл. опт. диск (CD-ROM).

Методическое пособие содержит файлы с методические указаниями по выполнению лабораторных работ средствами выбранной СУБД, пример оформления лабораторной работы, перечень предметных областей.

Методическое пособие предназначено для бакалавров направлений «*Прикладная математика и информатика*» (010400.62 – 5 семестр, 010500.62 – 7, 8 семестры), «*Прикладные математика и физика*» (010600.62 – *2, 3 семестры*, 010900.62 – *3, 4 семестры)*, студентов специальности «*Прикладная математика и информатика*» (010501.65 – 7 , 8 семестры).

Разработано на кафедре технической кибернетики.

Лабораторная работа №1.

Тема: Создание концептуальной модели предметной области с применением CASE – средств.

Разработка современных сложных информационных систем требует применения специальных методик и инструментов. Этим объясняется возросший среди разработчиков интерес к CASE – технологиям и инструментальным CASE – средствам. CASE – технология представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем программного обеспечения, поддержанных комплексом взаимосвязанных средств атоматизации. CASE – это инструментарий для системных аналитиков, разработчиков и программистов, который позволяет описывать бизнес – процессы на компьютере, используя полученные схемы при разработке или настройке системы.

CASE-средства ERwin и BPwin, разработанные фирмой Computer Associates International, Inc., являются, пожалуй, самыми популярными. Erwin применяется как средство для разработки IDEF1X – модели.

IDEF1X (IDEF1 Extended) — Data Modeling — методология построения реляционных структур (баз данных), относится к типу методологий «Сущность-взаимосвязь» (ER — Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе.

ERwin Data Modeler (ранее: ERwin) позволяет проектировать, документировать и сопровождать базы данных. Создав наглядную модель базы данных, предоставляет возможность оптимизации структур. Визуальное моделирование повышает качество создаваемой базы данных, продуктивность и скорость еѐ разработки. Кроме того, позволяет созданную модель данных перенести в выбранную базу данных в автоматическом режиме.

При создании новой модели пользователю предлагается выбрать тип создаваемой модели: логическая, физическая, логическая/физическая. После чего предоставляется возможность формирования структуры базы данных. Пользуясь возможностями ERwin Data Modeler созданную модель данных можно перенести в выбранную базу данных в автоматическом режиме. При этом отпадает необходимость вручную создавать структуру таблиц и связей между ними. Возможна и обратная операция – из существующей базы данных сформировать модель данных в программе.

Чтобы добавить в модель новую сущность, необходимо на панели «**Toolbox**» выбрать инструмент «**Entity**», после чего щелкнуть мышью на свободном месте рабочей области. Появится рамка новой сущности, и программа перейдет в режим ввода имени сущности. Завершается ввод имени нажатием на клавишу Enter.

После ввода имени сущности в зависимости от режима отображения, который может быть определен с помощью пункта меню «**Format**»_«**Display level**», возможны варианты:

в режиме «**Entity**» отобразится вновь добавленная сущность;

в режиме «**Attribute**» будет предложено ввести имена ключевых атрибутов (этот режим включен по умолчанию), при этом ввод каждого из них завершается нажатием «Enter», при нажатии клавиши «Tab» осуществляется переход для ввода неключевых атрибутов;

в режиме «**Definition**» будет предложено ввести определения для вновь добавленной сущности;

в режиме «Primary key» будет предложено ввести имена ключевых атрибутов;

в режиме «Icon» добавленная сущность будет представлена пиктограммой.

Редактор атрибутов вызывается двойным щелчком мышью на любом из них. Если сущность была создана без атрибутов, то добавить новый атрибут можно кнопкой «**New…**» в нижней левой части диалога. Если атрибуты уже есть, их можно переименовать кнопкой «**Rename…**» или удалить кнопкой «**Delete**».

В редакторе атрибутов можно указать тип атрибута и другие свойства.

При добавлении нового атрибута на экране появляется диалоговое окно, где предлагается указать его тип данных, логическое и физическое имя. Чтобы создать связь, необходимо в панели «**Toolbox**» выбрать тип связи и последовательно щелкнуть на двух связываемых сущностях. При этом для связей типа «один-ко-многим» сначала щелкнуть на родительской, потом на дочерней сущностях. Для связей типа «многие-ко-многим» в произвольном порядке. Открыв окно редактора связей (двойным щелчком), можно задать ее свойства. На вкладке «General» можно в разделе «Relationship Cardinality» можно указать конкретный тип связи, который обеспечивает ERwin:

• ноль, один или много;
• один или много;
• ноль или один;
• точное количество.

В поле «Verb Phrase» можно задать роли участия сущностей в связи.

Так же указать, является связь идентифицирующей или нет, допускает наличие пустых значений.

На вкладке «Rolename» можно указать, будет ли переименовываться атрибут, соответствующий внешнему ключу, и под каким именем он будет отображен в дочерней таблице.

Для обеспечения целостности данных при модификации информации в базе данных на вкладке «RI Actions» можно указать, какие действия выполняются с сущностью при выполнении операций (удаления, вставки, обновления) с другой сущностью.

На физическом уровне модель выглядит так, как она должна быть реализована средствами СУБД. В раскрывающемся списке в панели инструментов следует выбрать «Phisical». При модификации физической модели рекомендуется придерживаться английских наименований атрибутов и сущностей. Это определяется в соответствии с требованиями предъявляемыми СУБД и приложениями, работающие с разработанной БД. После переименования следует проверить значения типа данных для каждого атрибута, допустимость неопределенных значений. Работа с атрибутами выполняется в окне «Columns», которое можно открыть двойным щелчком по любому из столбцов таблицы. После завершения моделирования разработанная структура данных может быть транслирована в выбранную БД определенной СУБД.

Задание на лабораторную работу.

1. После согласования варианта выбранной предметной области (приложение 1) с преподавателем, пользуясь пакетом Erwin или Oracle SQL Developer Data Modeler, сформировать модель данных для реализации БД .
   1.1. Описать предметную область;
   1.2. Выделить сущности (модель должна включать не менее 6-8 сущностей);
   1.3. Определить связи между ними;
   1.4. Осуществить формализацию модели, устранить избыточность;
   1.5. Задать необходимые ограничения целостности.
2. Выполнив, все этапы 1 пункта, оформить отчет, который должен включать: описание предметной области, сущностей, ограничений целостности, а также описание работы, выполненной для приведения отношений к 3НФ или НФБК.
3. Выслать преподавателю на электронный адрес файлы с отчетом и моделью.

Лабораторная работа №2.

Тема: Реализация логической модели средствами СУБД.

Реляционная база данных – это совокупность взаимосвязанных таблиц, описывающих некоторую предметную область. Структура таблиц определяет эффективность программ, обрабатывающих эти таблицы, и все приложение в целом. Реляционная модель баз данных основывается на математических принципах теории реляционных наборов.

Создавать и модифицировать таблицы, добавлять, изменять и удалять записи в них возможно с помощью языка SQL (Structured Query Language – язык структурированных запросов). Особенность языка состоит в том, что его команды ориентированы на работу не с отдельными записями, а с их наборами, как свободных, так и реляционно-связанных таблиц и баз данных. В командах SQL формулируется критерий и способы преобразования информации. При этом SQL определяет наиболее эффективные последовательности операций для получения результата.

Для создания таблицы используется оператор ***CREATE TABLE:***

CREATE TABLE <имя таблицы>
({<имя поля> <тип данных> [(<размер>)]
[<ограничения целостности поля> …]},..
[,<ограничения целостности таблицы>,…]);
Для обязательных полей устанавливается ограничение not null.

- <имя поля> - имя поля (столбца) таблицы;
- <тип данных> - один из выше перечисленных типов;
- <размер> - размер поля в символах;
- <ограничения целостности поля> - предполагает использование следующих ограничений:
  - ✓ primary key – первичный ключ (обязательный и уникальный);
  - ✓ unique – уникальное значение поля в пределах таблицы;
  - ✓ [not] null – [не] возможность не указывать значение поля;
  - ✓ check(<условие>) – проверка условия для поля (полей);
  - ✓ default<выражение> - задание значения поля по умолчанию;
  - ✓ references<имя таблицы>[<имя столбца>] внешний ключ;
  - ✓ <ограничения целостности таблицы> - те же ограничения, что и для поля и дополнительно::
  - ✓ foreign key [<список полей>, …] references <имя таблицы> [(<список полей>)] внешний ключ

Например,

Create table tab (id numeric(6) primary key, class numeric(3), fdata date, group char(6), foreign key (class, fdata) references exam(class, fdata));

Для вставки кортежа (записи) применяется оператор ***INSERT:***

INSERT INTO <имя таблицы> [(<имя поля>), …] VALUES (<список выражений>)| <запрос>;

Для создания триггеров используется оператор ***CREATE TRIGGER****:*

CREATE TRIGGER <имя_триггера>
ON <имя_таблицы>
FOR {[INSERT] [.UPDATE] [. DELETE] }
[WITH ENCRIPTING]
AS
SQL-операторы (Тело триггера)

- Имя триггера является идентификатором во встроенном языке программирования СУБД и должно удовлетворять соответствующим требованиям;
- В параметре FOR задается одна или несколько операций модификации, которые запускают данный триггер.
- Параметр WITH ENCRIPTING имеет тот же смысл, что и для хранимых процедур, он скрывает исходный текст тела триггера.

Существует несколько правил, ограничивающих набор операторов используемых в теле триггера.

В большинстве СУБД действуют следующие ограничения:

- Нельзя использовать в теле триггера операции создания объектов БД (новой БД, новой таблицы, нового индекса, новой хранимой процедуры, нового триггера, новых индексов, новых представлений);
- Нельзя использовать в триггере команду удаления объектов DROP для всех типов базовых объектов БД;
- Нельзя использовать в теле триггера команды изменения базовых объектов . ALTER TABLE, ALTER DATABASE;
- Нельзя изменять права доступа к объектам БД, то есть выполнять команду GRAND или REVOKE;
- Нельзя создать триггер для представления (VIEW);

- В отличие от хранимых процедур, триггер не может возвращать никаких значений, он запускается автоматически сервером и не может связаться самостоятельно ни с одним клиентом.

Задание на лабораторную работу.

1. В соответствии с моделью из л/р №1 создать необходимые таблицы, учитывая типы данных, характерные для выбранной СУБД, и необходимые внутренние ограничения целостности.
2. Создать необходимые триггеры и хранимые процедуры для обеспечения целостности данных.
3. Заполнить созданные структуры 8-10 записями.
4. Выслать преподавателю на электронный адрес файл со скриптами и их описанием.

Структура данных, разработанная в л/р №1 может быть транслирована в базу данных СУБД Oracle, MS SQL Server, MySQL и другие.

Лабораторная работа №3.

Тема: Реализация запросов к БД.

Для организации запросов к данным, хранящимся в базе данных, служит оператор выбора языка SQL.

Извлечение данных из отношений выполняется с помощью команды SELECT (селекция). Эта команда не изменяет данные в БД.

Результатом выполнения команды ***SELECT*** является временное отношение, которое помещается в курсор (специальную область памяти СУБД) и обычно сразу выводится на экран. Синтаксис этой команды:

SELECT * | { [ALL | DISTINCT] <список вывода>.,..}
FROM {<имя таблицы> [<алиас>] }.,..
[WHERE <условие>]
[GROUP BY {<имя поля> | <целое>}.,.. [HAVING <условие>]]
[ORDER BY {<имя поля> | <целое> [ASC | DESC] }.,..]
[UNION [ALL] SELECT …];

- список вывода
  Элементом списка вывода может быть выражение и необязательный алиас. Выражение может включать имена полей, знаки операций, вызовы функций и константы. Список элементов, разделяется запятыми.
- алиас - временный синоним имени таблицы, определённый внутри запроса.
- условие – содержит условие-предикат выборки или| и условие соединения таблиц

DISTINCT – предикат удаления из результирующего отношения повторяющихся кортежей.

ALL – предикат, обратный к ***DISTINCT*** (используется по умолчанию).

Рассмотрим основные предложения команды ***SELECT***:

SELECT – после этого ключевого слова указывается список вывода

Например:

Select поле1*поле2 as синоним, поле3

Если надо вывести все поля из отношений, к которым обращается данный запрос, можно указать символ * (если в отношениях нет полей с одинаковыми именами). В этом случае сначала будут выведены поля таблицы, стоящей первой в предложении ***FROM***, затем – второй и т.д. Поля, относящиеся к одной таблице, будут выводиться в том порядке, в каком они были записаны при создании таблицы.

FROM – в этом предложении указывается имя таблицы (имена таблиц), из которых осуществляется выборка.

WHERE – содержит условия выбора отдельных записей.

GROUP BY – группирует записи по значению одного или нескольких полей. Каждой группе в результирующем отношении соответствует одна запись.

HAVING – позволяет указать условия выбора для групп записей. Может использоваться только после **GROUP BY**.

ORDER BY – упорядочивает результирующие записи по значению одного или нескольких полей: *ASC* – по возрастанию, *DESC* – по убыванию.

Если во фразе **FROM** указаны две и более таблицы, то эта последовательность действий выполняется для декартова произведения указанных таблиц.

Представление (view, обзор) – это хранимый запрос, создаваемый на основе команды *SELECT*.

Представление реально не содержит данных. Запрос, определяющий представление, выполняется тогда, когда к представлению происходит обращение с другим запросом, например, *SELECT*, *UPDATE* и т.д.

Создание представления выполняется командой **CREATE VIEW**:

CREATE VIEW <имя представления> [(<имя столбца>.,..)]

AS <запрос>;

Запрос, на основании которого создаётся представление, называется **определяющим запросом**, а таблицы, к которым происходит обращение в определяющем запросе – **базовыми таблицами**. Определяющий запрос не может включать предложение *ORDER BY*.

Если не указывать имена столбцов, то они получат названия по именам, перечисленным в списке выбора определяющего запроса. Указывать имена столбцов представления обязательно, если список выбора содержит агрегирующие функции или столбцы с одинаковыми именами из разных таблиц.

Например: Создать представление, содержащее информацию о студентах:

create view Wstud

as Num, Name, Group,Year, Bd, Money, Adress

from Stud;

select * from Wstud;

Представление может быть обновляемым и не обновляемым. Обновляемым является представление, при обращении к которому можно обновить базовую таблицу. Изменения будут произведены в базовой таблице и отразятся в представлении. По стандарту

SQL-2 представление не является обновляемым, если определяющий запрос:

- содержит ключевое слово *DISTINCT*;
- ссылается на другое необновляемое представление;
- содержит вычисляемые выражения в списке выбора;
- выбирает данные более чем из одной таблицы;
- содержит предложение *GROUP BY*.

Если вносимые изменения выходят за рамки определяющего запроса и поэтому не могут быть отражены в представлении, они могут быть отвергнуты системой (это зависит от реализации).

Задание на лабораторную работу.

После выполнения работы, связанной с созданием модели предметной области, преподавателем на Ваш электронный адрес будет выслан файл с формулировками необходимых к реализации запросов, представлений.

1. Выполнить запросы, указанные в задании.
2. Создать представления.
3. Оформить отчет. Выслать его преподавателю на электронный адрес.

Для выполнения данной работы можно также обратиться к материалам предыдущего семестра:

1. лекционный материал, посвященный оператору выбора языка SQL;

Лабораторная работа №4.

Тема: Создание приложения.

Опыт создания компьютерных приложений показывает, что для получения хорошего результата необходим систематический подход при их разработке. Модель, известная как цикл разработки приложения, представляет собой последовательность этапов, позволяющих создавать высокоэффективные приложения. Такая модель включает следующие этапы:

Анализ (сбор требований пользователей);

Проектировании (учет требований к проектированию);

Реализация проекта;

Тестирование (проверка корректной работы);

Поддержка работы приложения (реализация новых функций приложения);

Размещение приложения (обеспечение доступа к приложению пользователям).

Первоначальный перечень требований к приложению может быть следующим: иметь информацию о каждом компьютере; поддерживать информацию о базе данных; обеспечить для приложения легкий, интуитивно понятный интерфейс; рационально использовать аппаратные ресурсы.

Целью этапа проектирования является создание приложения, которое будет удовлетворять требованиям, заявленным на предыдущем этапе. На практике рекомендуется обеспечить доступ к проекту всех его потенциальных пользователей для возможного его дополнения и модификации.

Этап реализации заключается в создании первой версии приложения. Прежде, чем приступить к нему, следует с особым вниманием отнестись к выбору средств и среды разработки, учитывая требования. Создание приложения базы данных обычно заключается в создании схемы базы данных и последующей разработке интерфейса приложения. Определения классов для языков программирования, таких C++, Java могут быть сгенерированы средствами UML.

Следующая ссылка указывает на ознакомительное руководство по созданию приложений в среде **Oracle® Database Express Edition 10*g:***
http://www.oranet.ru/OraDoc10gXE/admin.102/b25610/toc.htm

Задание к лабораторной работе.

1. Реализовать приложение для доступа к данным базы данных из л.р. №2. *(Инструментальные средства выбираются по желанию);*
2. Обеспечить возможность добавления, удаления и изменения данных;
3. Включить в приложение запросы и представления из л.р. №3;
4. Создать отчеты для вывода данных (по согласованию с преподавателем) из базы данных;
5. Оформить отчет с описанием приложения.

Требования к приложению: удобный, интуитивно понятный интерфейс;

Список использованных источников:

1. Visual Foxpro - система управления базами данных [Текст] = Висуал фокспро : метод. указания к лаб. практикуму / Самар. гос. аэрокосм. ун-т им. С. П. Королева ; [сост. Логанова Л. В.]. - Самара : [б. и.], 2004. - 41 с. - 53.75 р.
2. ORACLE - система управления базами данных [Текст] : [метод. указания] / Федер. агентство по образованию, Самар. гос. аэрокосм. ун-т им. С. П. Королева ; [сост. Л. В. Логанова, сост. Ю. В. Колчин]. - Самара : Изд-во СГАУ, 2006. - 79 с. - (Приоритетные национальные проекты «Образование»).
3. Базы данных: модели, разработка, реализация / Т.С. Карпова. – СПб.: Питер, 2002. – 304с.: ил.
4. Кайт Т. ORACLE для профессионалов. –СПб.: - «Диа Софт Ю П», 2003. – 672 с.
5. Ален К. 101 Oracle PL\SQL. - М.: «Лори», 2001. – 350 с.
6. Дейт, К. Введение в системы баз данных. М. :Издательский дом «Вильямс», 2002. – 1072с. Приложение

Приложение **1**

ПРЕДМЕТНЫЕ ОБЛАСТИ:

1.Регистратура лечебного учреждения

2.Склад-магазин компьютерной техники

3.Склад-магазин книжной продукции

4.Издательство

5.Регистрация транспортных средств

6.Риэлторское агентство

7.Юридическая консультация

8.Расписание занятий

9.Научная конференция

10.Вуз

11.Спортивные соревнования

12.Сессия

13.Страхование

14.Продажа билетов

15.Объекты недвижимости

16.Каталоги (фильмов)

17.Каталоги (музыкальные)

18.Гостинница

19.Частный медицинский кабинет

20.Больница

21.Аптеки и лекарственные препараты

22.Справочник

23.Биржа труда

24.Система питания

25.Турагенство

Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ им. академика С.П. КОРОЛЕВА
(национальный исследовательский университет)

Факультет информатики

Кафедра технической кибернетики

Лабораторная работа № 1
по дисциплине «Базы данных и экспертные системы»

Тема:
«Создание концептуальной модели
предметной области с применением CASE – средств»

Выполнил студент: ФИО

Группа: XXX

Преподаватель: Логанова Л. В.

Самара 2011

Выполнение:

С помощью Erwin или Oracle SQL Developer Data Modeler сформировать модель данных для реализации БД «Успеваемость студентов».

- *Описание предметной области*

Каждому студенту, обучающемуся в университете необходимо 2 раза в год по окончании семестра сдавать сессию. О студенте в базе данных должна храниться следующая информация: ФИО, Год поступления, Группа, Факультет, Адрес. Любой студент имеет уникальный номер зачетной книжки.

- *Выделение сущностей*

Выделим следующие сущности:

Студент с атрибутами: ФИО, Год поступления, Группа, Факультет, Адрес .

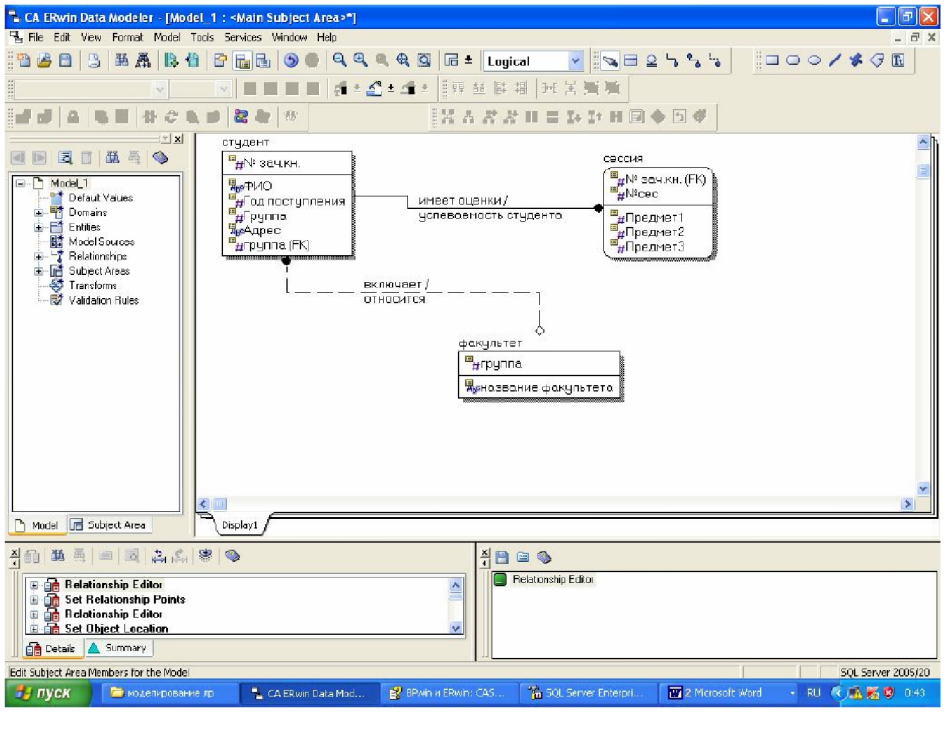
Сессия с атрибутами: № сессии, Предмет1, Предмет2, Предмет3

- *Определить связи между ними*

Между сущностями Студент и Сессия устанавливается связь один ко многим.

- *Осуществить формализацию модели, устранить избыточность*

Для исключения явлений избыточности и аномалий проведем нормализацию до 3 НФ. Получим следующие сущности:



На физическом уровне данная модель будет выглядеть следующим образом: