Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Пермская государственная сельскохозяйственная академия
имени академика Д. Н. Прянишникова»

Кафедра Информационных технологий и
автоматизированного проектирования

Проектирование баз данных экономических информационных систем

Методическое пособие по контрольной работе
по дисциплине «Базы данных»

Пермь 2013

Прохоров А.А. Методическое пособие по контрольной работе «Проектирование баз данных экономических информационных систем». Пермь, Пермская государственная сельскохозяйственная академия имени академика Д.Н. Прянишникова.

Методическое пособие по контрольной работе предназначено для студентов дневного и заочного отделений направления 230700 «Прикладная информатика».

В настоящем пособии излагаются общие положения и методика разработки основных вопросов контрольной работы по дисциплине «Базы данных» с позиций использования семантических моделей данных. Даются указания по оформлению контрольной работы в соответствии с требованиями ГОСТов и ЕСКД.

Печатается по постановлению методической комиссии факультета прикладной информатики ФГБОУ ВПО «Пермская ГСХА имени академика Д.Н. Прянишникова» (протокол № __от ___201_ г.)

Содержание

Перечень обозначений и сокращений

ЭИС - экономическая информационная система

АРМ - автоматизированное рабочее место пользователя

БД - база данных

ПО - предметная область (сфера деятельности, для которой разрабатывается экономическая информационная система)

СУБД - система управления базой данных (системное программное обеспечение, используемое для хранения и обработки данных)

PK - primary key (первичный ключ)

FK - foreign key (внешний ключ)

Введение

Контрольная работа по дисциплине «Базы данных» выполняется студентами направления 230700 – прикладная информатика в четвертом семестре после изучения теоретической части и прохождения лабораторного практикума.

Целью контрольной работы является систематизация знаний и накопление первоначального опыта в области проектирования баз данных и автоматизированных рабочих мест в составе корпоративных информационных систем.

В процессе работы над контрольной работой студент осуществляет постановку задачи, выбор методики проектирования, обследование и анализ требований предметной области, проектирование структуры базы данных и разработку полной программы, взаимодействующей с БД, для автоматизации функций пользователей.

При выполнении контрольной работы студент учится применять знания по базам данных, языкам и системам программирования, работать со специальной технической и справочной литературой, самостоятельно принимать и оценивать качество принятых решений.

1. Цели и задачи контрольной работы

Контрольная работа должна способствовать:

- закреплению, углублению и обобщению знаний, полученных студентами в процессе изучения лекционного курса по дисциплине «Базы данных»;
- развитию умений и навыков, полученных при выполнении лабораторных работ;
- применению этих знаний, умений и навыков к решению конкретных проектных задач с позиций объектно-ориентированного подхода к проектированию баз данных информационных систем;
- развитию навыков работы со специальной литературой и навыков проектирования с использованием CASE-средств.

2. Тематика и содержание контрольной работы

Задание по контрольной работе посвящено проектированию базы данных информационной системы (ИС). При этом предусматривается закрепление знаний и навыков проектирования баз данных с применением методологии IDEF1X для разработки концептуальной и физической моделей данных.

При выполнении контрольной работы студент должен:

- разработать техническое задание (ТЗ) на разработку баз данных ИС по ГОСТ 34.602-89;
- выполнить системный анализ и анализ требований к создаваемой базе данных ИС, разработать концептуальную (логическую) модель предметной области, физическую модель данных, создать базу данных ИС предложенной предметной области в среде реляционной СУБД Visual FoxPro 9.0;
- оформить пояснительную записку (текстовую часть контрольной работы) и приложения (диаграммы, ИЛМ, таблицы, схемы БД, формы, запросы и отчеты) - электронную часть контрольной работы, записываемую на CD диск.

Кроме того, по решению кафедры в состав контрольной работы могут быть включены дополнительные разделы, связанные с научно-исследовательской работой.

3. Задания по контрольной работе

Индивидуальное задание по контрольной работе складывается из основного варианта баз данных ЭИС (таблица 1) и варианта предметной области, в которой должна функционировать экономическая информационная система (таблица 2).

Примечание: ***студенты заочной формы обучения индивидуально согласуют вариант предметной области с преподавателем***.

Таблица 1

Основной вариант	Разработка базы данных экономической информационной системы для автоматизации **предметной области**.

Таблица 2

N варианта	**Предметная область**
1	Расчет отчислений на медицинское страхование
2	Расчет финансовых показателей (на заказ)
3	Обработка информации по поступлению товарноматериальных ценностей на склад
4	Анализ движения денежных средств через кассу по целевому назначению
5	Расчет фактической себестоимости продукции
6	Обработка информации по отпуску материалов в производство
7	Обработка информации по расчетам с покупателями
8	Обработка информации по табелю учета отработанного времени
9	Учет рублевых операций в банке
10	Отдел кадров (учет перемещений работников)
11	Обработка информации по вводу в эксплуатацию основных средств
12	Обработка информации выписки из банка о движении денежных средств
13	Обработка информации по поступлению готовой продукции на склад
14	Анализ состояния рынка ценных бумаг
15	Обработка информации авансовых отчетов
16	Учет физических лиц-налогоплательщиков в налоговых органах
17	Обработка информации по выдаче денег из кассы
18	Расчет подоходного налога от фонда оплаты труда сотрудников предприятия
19	Учет движения готовой продукции на складе
20	Начисление оплаты за работу в выходные и праздничные дни
21	Обработка информации по оказанию услуг жилищного и коммунального хозяйства

	населения.
22	Обработка информации по списанию основных средств
23	Начисление доплаты за выслугу лет сотрудникам предприятия
24	Обработка информации по договорам с контрагентами
25	Формирование ведомостей на оплату за проживание студентов в общежитии
26	Учет выплаты стипендии студентам и аспирантам
27	Обработка информации по расчетам с поставщиками
28	Расчет отчислений в пенсионный фонд
29	Формирование цены продажи продукции
30	Фасовка продукции и формирование цены продажи
31	Начисление отпускных работникам предприятия
32	Учет вкладчиков в банке.
33	Начисление оплаты за дни временной нетрудоспособности
34	Расчет плановой себестоимости продукции
35	Обработка информации по поступлению основных средств
36	Обработка информации по отгрузке готовой продукции

4. Правила оформления пояснительной записки

Пояснительная записка выполняется на одной стороне листа бумаги формата А4. Общий объем не менее 20 страниц (без приложения). Все таблицы, рисунки, схемы, формулы, графики должны быть пронумерованы и снабжены подписями и ссылками в тексте. Оформление пояснительной записки должно соответствовать требованиям: ГОСТ 7.32-2001 СИБИД. Отчет о научно-исследовательской работе. Структура и правила оформления; ГОСТ 2.105—95 ЕСКД. Общие требования к текстовым документам. При оформлении пояснительной записки рекомендуется пользовать методическими указаниями, изложенными в этом методическом пособии.

Материалы в пояснительной записке следует располагать в следующем порядке:

- Титульный лист (приложение 1);
- Техническое задание на проектирование;
- Содержание;
- Введение;
- Раздел 1. Системный анализ предметной области (ПО) и анализ требований к базе данных;
- 1.1. Формулировка задания; 1.2. Конкретизация ПО; 1.3. Требования к хранению данных; 1.5. Сроки хранения информации; 1.6. Ситуации, изменяющие БД; 1.7. Основные запросы (на естественном языке).
- Раздел 2. Концептуальное моделирование предметной области;
- 2.1. ER-диаграмма модели ПО (на ERwin или Silveran); 2.2. Оценка мощностных характеристик сущностей и связей.
- Раздел 3. Концептуальное проектирование;
- 3.1. Концептуальная модель БД (ERwin или Silveran).
- Раздел 4. Логическое проектирование.
- 4.1. ER-диаграмма БД (ERwin Logical); 4.2. Схема отношений БД (ERwin Physical); 4.3. Схемы реляционной БД; 4.4. Схемы основных запросов.

- Раздел 5. Физическое проектирование (СУБД Visual FoxPro 9.0)
- 5.1. Создание БД; 5.2. Создание таблиц; 5.3. Заполнение таблиц данными контрольного примера; 5.4. Создание электронных форм (2-3), запросов (4-5) и отчетов (4-5) в среде СУБД Visual FoxPro 9.0); 5.5 Оценка размеров БД и каждого из файлов.
- Заключение;
- Список использованных источников;
- Приложение (приложения).

Законченная пояснительная записка подписывается студентом. Изложение должно быть ясным и четким, без повторений. Следует избегать необоснованного использования в тексте пояснительной записки большого количества теоретического материала.

5. Правила оформления графического материала

Графическая часть проекта является не иллюстративным материалом, а технической документацией на разработанный студентом проект БД ЭИС. Графический материал, помещенный в пояснительной записке - по формату, условным обозначениям, шрифтам и масштабам должен соответствовать требованиям единой системы конструкторской документации (ЕСКД). При выполнении графического материала с использованием CASE-средства ERWin Data Modeler в нотации *IDEF1X*, этому международному стандарту.

6. Методика выполнения контрольной работы

6.1. Техническое задание на проектирование

Выполняется по ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы. При этом студентом заполняются следующие разделы (и их подразделы): 1) общие сведения; 2) назначение и цели создания (развития) системы; 3) характеристика объектов автоматизации; 4) требования к системе; 5) состав и содержание работ по созданию системы.

6.2. Введение

Введение (общим объемом не более 2 стр.) должно содержать общие сведения о БД, его краткую характеристику, резюме. В нем необходимо отразить актуальность выбранной темы, цель и задачи, решаемые в проекте, используемые методики, практическую значимость полученных результатов. Во введении необходимо также перечислить вопросы, которые будут рассмотрены в проекте, выделив вопросы, которые предполагается решить практически.

6.3. Системный анализ и анализ требований

Первыми выполняемыми задачами являются системный анализ и анализ требований. Они закладывают фундамент для решения последующих задач.

Системный анализ проводится с целью:

1) выяснения потребностей заказчика;
2) оценки выполнимости системы;
3) выполнения экономического и технического анализа;
4) распределения функций по элементам компьютерной системы (аппаратуре, программам, людям, базам данных и т. д.);
5) определения стоимости и ограничений планирования;
6) создания системной спецификации.

Результаты системного анализа оформляются в *системной спецификации*, где описываются функции, характеристики системы, ограничения разработки, входная и выходная информация.

Анализ требований дает возможность:

1) определить функции и характеристики программного продукта;
2) обозначить интерфейс продукта с другими системными элементами;
3) определить проектные ограничения программного продукта;
4) построить модели: данных, режимов функционирования продукта;
5) создать такие формы представления информации и функций системы, которые можно использовать в ходе проектирования.

6.4. Использование методологии IDEF1X для разработки концептуальной модели данных

Важнейшая цель проектирования информационной модели - выработка непротиворечивой структурированной интерпретации реально существующей информации изучаемой предметной области и взаимодействия между ее структурными компонентами

Понятие ***концептуальной модели*** данных связано с методологией семантического моделирования данных, т.е. с представлением данных в контексте их взаимосвязей с другими данными.

Методология ***IDEF1X*** - один из подходов к семантическому моделированию данных, основанный на концепции "сущность-связь" (Entity-Relationship). Это инструмент для анализа информационной структуры систем различной природы. Информационная модель, построенная с помощью ***IDEF1X***-методологии, отображает логическую структуру информации об объектах системы [2, 4, 9].

Таким образом, ***концептуальная модель***, представленная в соответствии со стандартом ***IDEF1X***, является логической схемой базы данных для проектируемой системы.

Основными объектами концептуальной модели являются сущности и связи.

Сущность - некоторый обособленный объект или событие моделируемой системы, имеющий определенный набор свойств - атрибутов. Отдельный элемент этого множества называется "экземпляром сущности". Сущность может обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый образец сущности, и может обладать любым количеством связей с другими сущностями.

Правила для атрибутов сущности:

1. Каждый атрибут должен иметь уникальное имя.
2. Сущность может обладать любым количеством атрибутов.
3. Сущность может обладать любым количеством наследуемых атрибутов, но наследуемый атрибут должен быть частью первичного ключа сущности-родителя.
4. Для каждого экземпляра сущности должно существовать значение каждого его атрибута (правило необращения в нуль - Not Null).
5. Ни один из экземпляров сущности не может обладать более чем одним значением для ее атрибута.

Сущность изображается на ER-диаграмме в виде прямоугольника, в верхней части которого приводится ее название; далее следует список атрибутов. Ключевые атрибуты могут быть выделены подчеркиванием или иным способом.

Стандарт ***IDEF1X*** описывает способы изображения двух типов *сущностей* - *независимой* и *зависимой*, и *связей* - *идентифицирующих* и *неидентифицирующих* (см. рис. 6.1).

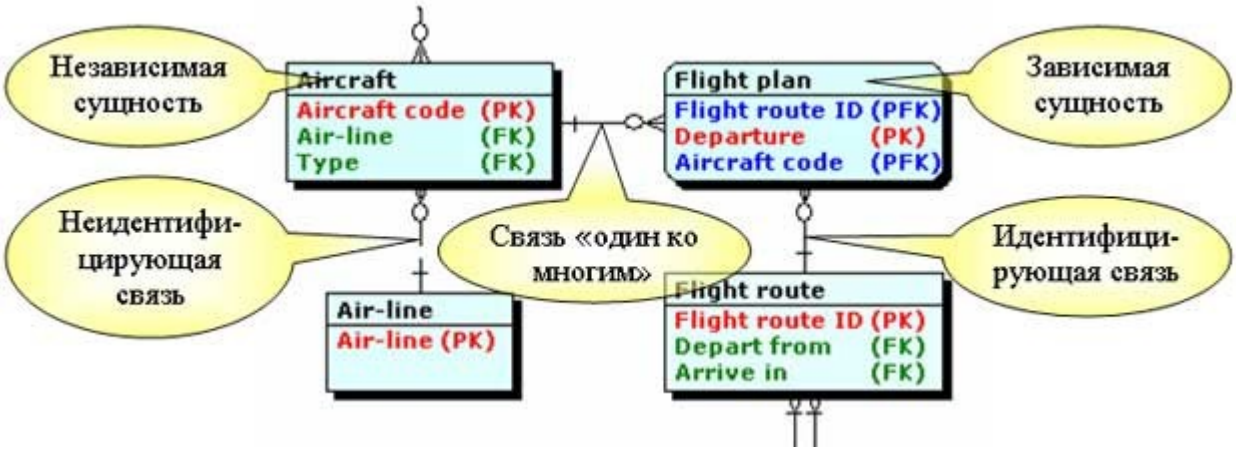

Рис. 6.1. Изображение сущностей и связей по стандарту *IDEF1X*

Каждая сущность может обладать любым количеством *связей* с другими сущностями.

Сущность является ***независимой***, если каждый ее экземпляр может быть однозначно идентифицирован без определения его связей с другими сущностями.

Сущность называется ***зависимой***, если однозначная идентификация ее экземпляра зависит от его связей с другими сущностями.

Сущность может обладать атрибутами, которые наследуются через связь с родительской сущностью. Последние обычно являются **внешними ключами** (FK на рис. 6.1) и служат для организации связей между сущностями. Если **внешний ключ** сущности используется в качестве ее первичного ключа (PK) или как часть составного первичного ключа, то сущность является *зависимой* от родительской сущности. Если внешний ключ не является первичным и не входит в составной первичный ключ, то сущность является *независимой* от родительской сущности.

Если сущность является *зависимой*, то связь ее с родительской сущностью называется ***идентифицирующей***, в противном случае - ***неидентифицирующей***.

Связь изображается на ER-диаграмме линией, проводимой между сущностью-родителем и сущностью-потомком с точкой на конце линии у сущности-потомка. ***Идентифицирующая*** связь изображается сплошной линией, ***неидентифицирующая*** - пунктирной.

Связи дается имя, выражаемое грамматической формой глагола. Для связи дополнительно может присутствовать указание мощности: какое количество экземпляров сущности-потомка может существовать для сущности-родителя. Имя связи всегда формируется с точки зрения родителя, так что может быть образовано предложение, если соединить имя сущности родителя, имя связи, выражение мощности и имя сущности-потомка (например "много СТУДЕНТов - сдают - ЭКЗАМЕН").

Принципы изображения концептуальных моделей баз данных стандарта *IDEF1* и *IDEF1X* используют *CASE Studio* и другие CASE-средства. Подобные системы позволяют на основе *концептуальной модели* генерировать *физическую модель* и программный код создания базы данных для большинства наиболее распространенных СУБД и серверов баз данных.

6.5. Пример описания модели данных информационной системы "Контингент студентов университета"

Первоначальный этап - создание текстового описания моделируемой системы.

Постановка задачи. Главная задача системы - сохранение в базе данных всех необходимых сведений о студентах и их успеваемости, формирование необходимых печатных форм для проведения зачетной и экзаменационной работы преподавателей, генерация сводных итогов по результатам сессии для руководящих работников деканатов, институтов и университета. При разработке системы следует учитывать, что она взаимодействует с системами "Абитуриент", "Стипендия" и "Кадры университета". Информация о студентах первоначально поступает из системы "Абитуриент" и редактируется на уровне деканатов. Она

должна также удовлетворять требованиям бухгалтерского учета по начислению стипендий. Система должна использовать справочник специальностей, утвержденный в вышестоящем министерстве. Информация об успеваемости студентов накапливается постоянно и сохраняется за весь период обучения, после чего переносится в архивное хранилище данных. В системе должен использоваться единый справочник дисциплин (предметов) для всех подразделений университета.

Концептуальная модель базы данных

На концептуальном уровне данные информационной системы состоят из двух основных *сущностей*: **"Студент"** и **"Успеваемость"**.

Минимальный состав атрибутов и их описание для сущности "Студент" представлены в табл. 6.1.

Таблица 6.1. Атрибуты сущности "Студент"

Имя атрибута	Описание, особенности использования
Номер зачетки	Первичный ключ - уникальный номер, однозначно идентифицирующий студента университета
Фамилия, имя, отчество	Является простым с точки зрения экземпляра сущности, при необходимости из общего поля можно выделить составляющие его фамилию, имя и отчество или фамилию и инициалы, однако на практике часто этот атрибут разделяют на 3 отдельных; первый вариант является более экономичным по необходимой общей ширине поля таблицы
Дата поступления в университет	В нашей стране наиболее часто используется формат работы с датой в виде ДД.ММ.ГГ, что совпадает с немецким (German) форматом дат. Количество цифр года: либо две - для новых систем, поддерживающих заданный в Microsoft Windows годичный интервал (Панель управления - Язык и стандарты - Дата - "При вводе двух цифр года воспринимать их как год между:"), или для систем, в которых аналогичный интервал может быть задан в программе, - либо 4 цифры
Факультет (№ факультета)	Может быть сложным (кроме кода и названия, может содержать и другие сведения); даже в том случае, если для сущности "Студент" мы хотим сохранять название факультета, оно должно быть представлено в одинаковом виде для каждого факультета, поэтому, в соответствии с принципами нормализации баз данных, этот атрибут следует представить в виде номера, являющегося внешним ключом для новой сущности - "Факультет", в которой каждому номеру, являющемуся первичным ключом, будут соответствовать название и прочие атрибуты этой сущности
Специальность(код специальности)	Может быть сложным, кроме того, необходимо использовать справочник министерства с утвержденными кодами специальностей, поэтому данный атрибут должен хранить код специальности - внешний ключ для первичного ключа новой сущности "Специальность"
Курс	Число от 1 до 5
Номер группы	Трехзначное число
Номер паспорта	Состав и вид паспортных данных определяется требованиями бухгалтерской отчетности перед налоговыми органами, фондами социального страхования и пенсионным фондом
...	Прочие атрибуты, которых может быть достаточно много

В табл. 6.2-6.5 представлены атрибуты *сущностей* **"Успеваемость"**, **"Факультет"**, **"Специальность"**, **"Предмет"**.

Таблица 6.2. Атрибуты сущности "Успеваемость"

Имя атрибута	Описание, особенности использования
Номер зачетки	Внешний ключ (к сущности "Студент")
Номер семестра	Число от 1 до 10
Предмет (№ предмета)	Может быть сложным, его следует заменить на его номер (внешний ключ) и связать с новой сущностью "Предмет", состоящий, как минимум, из атрибутов "номер предмета" (первичный ключ) и "название предмета"
Оценка	Может быть представлена цифрами от 0 до 5 или 1 буквой: например "н" - неявка
Дата получения оценки	Формат даты обычно ДД.ММ.ГГ
Фамилия преподавателя	Это поле может быть связано с сущностью "Преподаватель". В данном учебном примере ограничимся простым атрибутом
...	Могут быть добавлены и другие атрибуты, например, номер экзаменационной ведомости

Таблица 6.3. Атрибуты сущности "Факультет"

Имя атрибута	Описание, особенности использования
Номер факультета	Первичный ключ
Название факультета	Может быть достаточно длинным, но не более 255 символов
...	Могут быть добавлены и другие атрибуты, например, декан, номер комнаты деканата и т.д.

Таблица 6.4. Атрибуты сущности "Специальность"

Имя атрибута	Описание, особенности использования
Код специальности	Первичный ключ - значение из справочника министерства
Название специальности	Значение из справочника министерства
...	Могут быть добавлены и другие атрибуты

Таблица 6.5. Атрибуты сущности "Предмет"

Имя атрибута	Описание, особенности использования
№ предмета	Первичный ключ
Название предмета	Общий справочник университета
...	Могут быть добавлены и другие атрибуты

В физической модели каждой сущности будет соответствовать таблица базы данных, а каждому атрибуту - поле таблицы. Имена таблиц и полей лучше задавать с использованием латинских букв и достаточно короткими для удобства использования при программировании и

для совместимости с системами, не использующими кириллицу. Состав данных и связи в концептуальной и физической моделях показаны в табл. 6.6 и табл. 6.7.

Таблица 6.6. Состав базы данных информационной системы

№ п/п	Сущности концептуальной модели	Таблицы физической модели	
		Название	Информация
1.	"Студент"	"SPISOK"	"Список студентов"
2.	"Успеваемость"	"OCENKI"	"Оценки студентов"
3.	"Факультет"	"FCLT"	Справочник факультетов
4.	"Специальность"	"SPECT"	Справочник специальностей
5.	"Предмет"	"PREDMET"	Справочник предметов

Таблица 6.7. Связи между объектами базы данных информационной системы

№ п/п	Концептуальная модель	Физическая модель
1.	"Студент" - "Успеваемость"	"SPISOK" - "OCENKI"
2.	"Студент" - "Факультет"	"SPISOK" - "FCLT"
3.	"Студент" - "Специальность"	"SPISOK" - "SPECT"
4.	"Успеваемость" - "Предмет"	"OCENKI" - "PREDMET"

Замечание: Все последующие диаграммы, входящие в состав моделей выполняются в CASE-средстве методологии *IDEF1X*.

6.6. Создание форм, запросов и отчетов в среде СУБД Visual FoxPro

Для баз данных, состоящих из большого количества таблиц, наглядная и удобная работа может быть организована при использовании ***экранных форм***. Данный режим позволяет использовать все необходимые данные из одной или нескольких таблиц. Можно разместить на *экранной форме* меню, панели инструментов, командные кнопки и другие сложные объекты для работы с данными.

На *экранных формах* можно организовать выбор информации из таблиц-справочников с использованием раскрывающихся списков или отдельных окон, использовать специальные режимы редактирования данных с сохранением или отменой изменений, режимы поиска и отбора информации, печати необходимых отчетов на принтере и пр.[5, 6].

Для работы с данными, отобранными в соответствии с каким-либо условием, может быть использована команда **SET FILTER TO** <условие> - установить фильтр для открытой таблицы базы данных.

Однако большими возможностями обладает так называемый ***SQL-запрос*** - команда **SELECT**, сформированная в соответствии с правилами языка *запросов SQL* (**Structured Query Language**).

Запрос позволяет отбирать данные по заданным сложным условиям из нескольких таблиц различных баз данных, с размещением результатов выполнения *запроса* на экране, во временной таблице (**cursor**), в новой таблице, в текстовом файле или в массиве переменных. При этом возможны сложные виды упорядочения информации и группировки данных с получением расчетных групповых результатов.

Отбор осуществляется непосредственно из файла на диске, таким образом, те же таблицы одновременно могут быть открыты с какими-либо установленными фильтрами (например, в программе, работающей с данными только за текущий месяц).

В **VFP** и других системах фирмы **Microsoft** (Word, Excel) можно использовать **Конструктор** *запросов*, что упрощает и ускоряет написание *запросов*. Кроме того, в VFP есть **Мастер** для разработки запросов разного вида. Однако использование этих средств не позволяет реализовать все возможности языка *запросов*. Максимальные возможности - при написании *запроса* в текстовом виде в любом программном модуле в соответствии с синтаксисом команды **SELECT** (полный синтаксис будет описан далее).

Принцип формирования *запросов* наиболее легко освоить при использовании **Мастера** *запросов* [5, 6].

Для разработки **отчетов** - печатных документов, отражающих информацию базы данных, в системе VFP существует **Конструктор отчетов** (*Report Designer*) и **Мастер отчетов** (**Report Wizard)**. Важным свойством отчетов является возможность группировки данных и получения итоговых данных для групп и всего отчета. При формировании отчетов можно задавать фильтр отбора необходимых данных либо формировать отчет на основе данных *SQL-запроса* или представления данных (View).

Наиболее просто для разработки основы отчета воспользоваться **Мастером отчетов** с последующей модификацией и дополнением отчета в **Конструкторе**.

Существует 2 типа **Мастера отчетов**:

One-to-Many Report Wizard - Мастер отчета, в котором для одной записи главной таблицы существует множество записей связанной с ней дочерней таблицы.

Report Wizard - Мастер простого отчета, но с возможностью задания группировки данных [5, 6].

6.7. Заключение

Рекомендуется сделать выводы по проекту, определить пути его внедрения и направления дальнейшего совершенствования БД ЭИС.

6.8. Приложение

Обязательно должна быть распечатка созданных: форм (2-3), запросов (3-4) и отчетов (3-4) в среде СУБД Visual FoxPro 9.0 [5, 6].

7. Для защиты курсового проекта предоставляются

7.1. Пояснительная записка в распечатанном и сброшюрованном виде, оформленная в соответствии с методическими указаниями к курсовому проекту;

7.2. Электронный носитель, содержащий:

- файл с пояснительной запиской,
- файл со скриптами создания базы данных, всех ее объектов и заполнения данными контрольного примера,
- исходные и исполняемые модули приложений для пользователя,

– файлы, используемые и создаваемые приложением (файлы инициализации, файлы отчетов и справок контрольного примера, сохраняемые в MS Office);

7.3. Дистрибутив приложения пользователя (если он предусмотрен проектом)

8. Порядок защиты

Защита производится перед комиссией, утверждаемой кафедрой.

Студент допускается к защите при условии наличия подписанной руководителем и студентом пояснительной записки и расчетно-графической части проекта.

Для защиты студенту отводится 8 - 10 минут на изложение содержания работы; в процессе защиты комиссия высказывает свои замечания; выявленные ошибки проекта должны быть отмечены красным карандашом.

По результатам защиты (доклад, ответы на вопросы, качество проекта) выставляется оценка в ведомости и на титульном листе пояснительной записки. В случае выявления принципиальных ошибок проект возвращается на доработку.

После защиты студент должен сдать пояснительную записку руководителю проекта. В случае неудовлетворительной оценки назначается повторная защита с устранением всех ошибок проекта или с выдачей нового задания. При отсутствии достаточного материала по проекту в контрольные сроки, студенту, как правило, выдается новое задание.

9. Список рекомендуемой литературы

1. Хомоненко, А.Д., Цыганков, В.М., Мальцев, М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. – 5-е изд., доп. и перераб. – СПб.: КОРОНА принт, 2006. – 736 с.
2. Диго, С.М. Базы данных: проектирование и использование: Учебник. – М.: Финансы и статистика, 2005. – 592 с.
3. Кузнецов, С.Д. Основы баз данных: Курс лекций. Учебное пособие / С.Д. Кузнецов. - М.: ИНТУИТ.РУ, Интернет-Университет Информационных Технологий, 2005. — 488 с.
4. Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite. – М.: ДИАЛОГ-МИФИ, 2003. – 432 с.
5. Мусина, Т.В. Visual FoxPro 8.0. Учебный курс – К.: ВЕК +, СПб.: КОРОНА принт, К.: НТИ, 2004. – 464 с.
6. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – М.: Финансы и статистика, 2002. – 352 с.: ил.
7. Технологии разработки программного обеспечения/ С.Орлов. СПб.: Питер, 2002. – 464 с.
8. Калянов, Г.Н. CASE – технологии. Консалтинг в автоматизации бизнес-процессов. – 3-е изд. – М.: Горячая линия – Телеком, 2002. – 320 с.

Список использованных источников

1. Хомоненко, А.Д., Цыганков, В.М., Мальцев, М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. – 5-е изд., доп. и перераб. – СПб.: КОРОНА принт, 2006. – 736 с.
2. Диго, С.М. Базы данных: проектирование и использование: Учебник. – М.: Финансы и статистика, 2005. – 592 с.
3. Кузнецов, С.Д. Основы баз данных: Курс лекций. Учебное пособие / С.Д. Кузнецов. - М.: ИНТУИТ.РУ, Интернет-Университет Информационных Технологий, 2005. — 488 с.
4. Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite. – М.: ДИАЛОГ-МИФИ, 2003. – 432 с.
5. Мусина, Т.В. Visual FoxPro 8.0. Учебный курс – К.: ВЕК +, СПб.: КОРОНА принт, К.: НТИ, 2004. – 464 с.
6. Омельченко, Л.Н. Самоучитель Visual Foxpro 8. – СПб: БХВ-Петербург, 2003. – 688 с.
7. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – М.: Финансы и статистика, 2002. – 352 с.
8. Белых А.А. Проектирование экономических информационных систем. Методическое пособие по курсовому проектированию. Пермь: Пермская ГСХА, 2005 - 34 с.
9. Белых А.А. Проектирование ЭИС. Методическое пособие по дипломному проектированию. Пермь: Пермская ГСХА, 2006 - 60 с.

Приложение 1

Министерство сельского хозяйства РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Пермская государственная сельскохозяйственная академия
имени академика Д. Н. Прянишникова»

Кафедра: ***Информационных технологий и автоматизированного проектирования***

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «Базы данных»

на тему: «Разработка базы данных экономической информационной системы для автоматизации учета сведений об оргтехнике на предприятии»

Проект выполнил:
студент факультета Прикладной информатики
Направления подготовки 230700
группы ПИ – 31а
Пепеляева Анастасия Ивановна

Руководитель:
доцент кафедры ИТАП,
к.т.н., доцент Прохоров А.А.

Оценка ……

……
(дата защиты)

……
(подпись преподавателя)

Пермь 201_

Приложение 2

Основы методологии IDEF1X

Предназначение IDEF1X

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. **Концептуальной схемой** мы называем универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято. Однако не стоит забывать, что средства моделирования IDEF1X специально разработаны для построения реляционных информационных систем, и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы моделирования.

Существует несколько очевидных причин, по которым IDEF1X не следует применять в случае построения нереляционных систем. Во-первых, IDEF1X требует от проектировщика определить ключевые атрибуты, для того чтобы отличить одну сущность от другой, в то время как объектно-ориентированные системы не требуют задания ключевых ключей, в целях идентифицирования объектов. Во-вторых, в тех случаях, когда более чем один атрибут является однозначно идентифицирующим сущность, проектировщик должен определить один из этих атрибутов первичным ключом, а все остальные вторичными. И, таким образом, построенная проектировщиком IDEF1X-модель и переданная для окончательной реализации программисту является некорректной для применения методов объектно-ориентированной реализации, и предназначена для построения реляционной системы.

Концепция и семантика IDEF1X

Сущности в IDEF1X и их атрибуты

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но

однозначно отличаемых друх от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира. Примером сущности IDEF1X может быть сущность "СОТРУДНИК", которая представляет собой всех сотрудников предприятия, а один из них, скажем, Иванов Петр Сергеевич, является конкретной реализацией этой сущности. В примере, приведенном на рис. 1, каждый экземпляр сущности СОТРУДНИК содержит следующую информацию: ID сотрудника, имя сотрудника, адрес сотрудника и т.п. В IDEF1X модели эти свойства называются атрибутами сущности. Каждый атрибут содержит только часть информации о сущности.

Связи между сущностями

Связи в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями. Связи это суть глаголы, которые показывают, как соотносятся сущности между собой. Ниже приведен ряд примеров связи между сущностями:

Отдел <состоит из> нескольких **Сотрудников**.

Самолет <перевозит> нескольких **Пассажиров**.

Сотрудник <пишет> разные **Отчеты**.

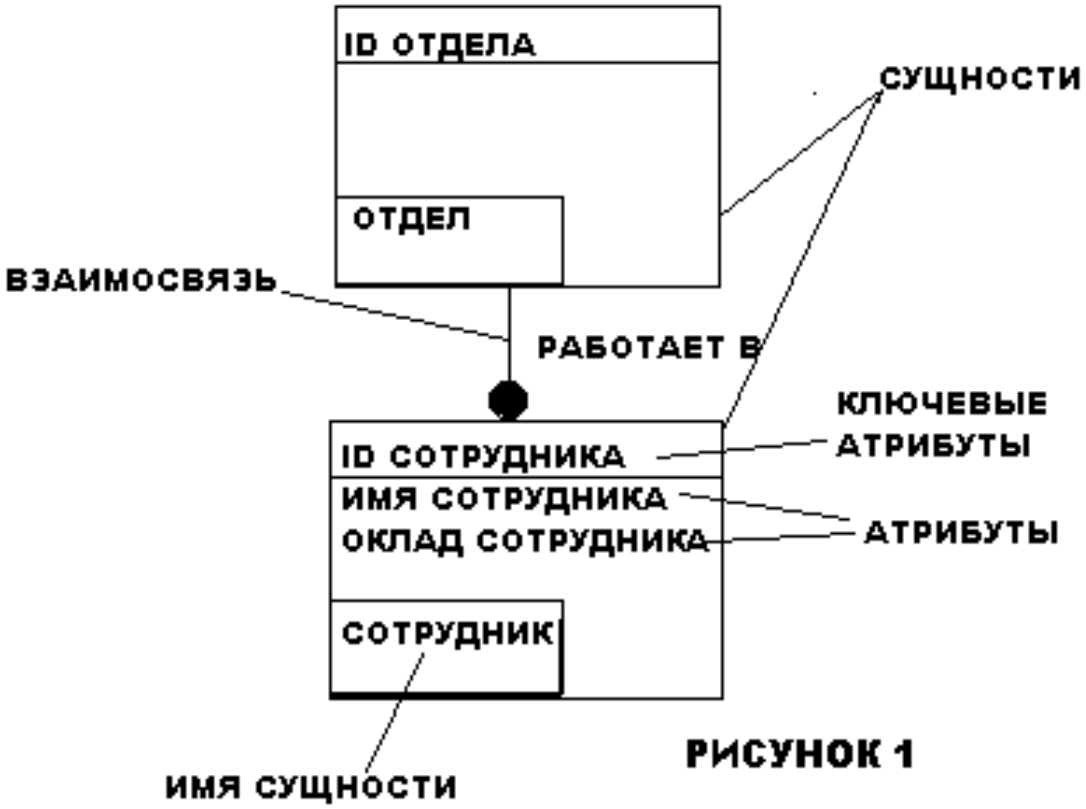

РИСУНОК 1

Во всех перечисленных примерах взаимосвязи между сущностями соответствуют схеме **один ко многим**. Это означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Причем первая сущность называется **родительской**, а вторая — **дочерней**. В приведенных примерах глаголы заключены в угловые скобки. Связи отображаются в виде линии между двумя сущностями с точкой на одном конце и глагольной фразой, отображаемой над линией. На рис. 1 приводится диаграмма связи между Сотрудником и Отделом.

Отношения многие ко многим обычно используются на начальной стадии разработки диаграммы, например, в диаграмме зависимости сущностей и отображаются в IDEF1X в виде сплошной линии с точками на обоих концах. Так как отношения многие ко многим могут скрыть другие бизнес правила или ограничения, они должны быть полностью исследованы на одном из этапов моделирования. Например, иногда отношение многие ко многим на ранних стадиях моделирования идентифицируется неправильно, на самом деле представляя два или несколько случаев отношений один-ко-многим между связанными сущностями. Или, в случае необходимости хранения дополнительных сведений о связи многие-ко-многим, например, даты или комментария, такая связь должна быть заменена дополнительной сущностью, содержащей эти сведения. При моделировании необходимо быть уверенным в том, что все отношения многие ко многим будут подробно обсуждены на более поздних стадиях моделирования для обеспечения правильного моделирования отношений.

Идентификация сущностей. Представление о ключах.

Сущность описывается в диаграмме IDEF1X графическим объектом в виде прямоугольника. На рис.2 приведен пример IDEF1X диаграммы. Каждый прямоугольник, отображающий собой сущность, разделяется горизонтальной линией на часть, в которой расположены **ключевые поля** и часть, где расположены неключевые поля. Верхняя часть называется ключевой областью, а нижняя часть областью данных. Ключевая область объекта СОТРУДНИК содержит поле «Уникальный идентификатор сотрудника», в области данных находятся поля «Имя сотрудника», «Адрес сотрудника», «Телефон сотрудника» и т. д.

Ключевая область содержит **первичный ключ** для сущности. Первичный ключ — это набор атрибутов, выбранных для идентификации уникальных экземпляров сущности. Атрибуты первичного ключа располагаются над линией в ключевой области. Как следует из названия, неключевой атрибут — это атрибут, который не был выбран ключевым. Неключевые атрибуты располагаются под чертой, в области данных.

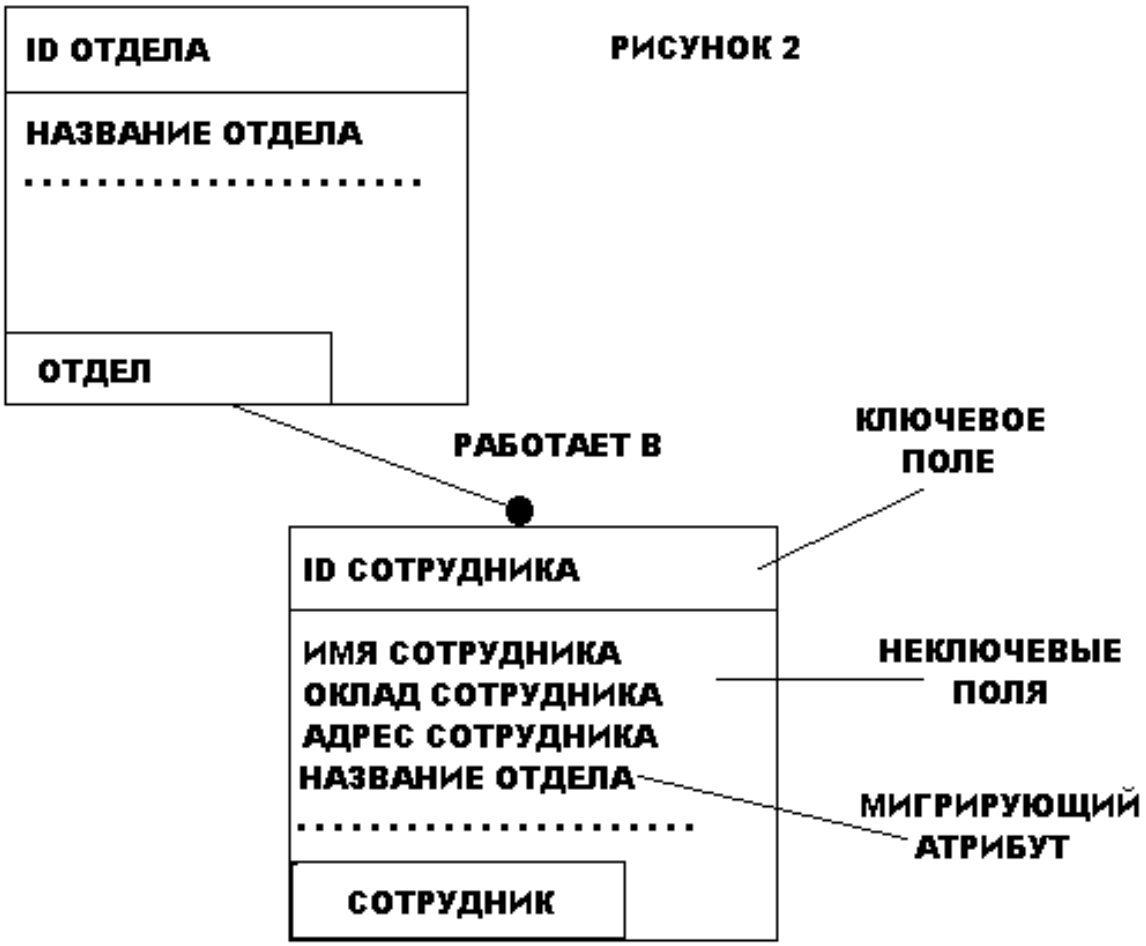

РИСУНОК 2

При создании сущности в IDEF1X модели, одним из главных вопросов, на который нужно ответить, является: «Как можно идентифицировать уникальную запись?». Для этого требуется уникальная идентификация каждой записи в сущности для того, чтобы правильно создать логическую модель данных. Напомним, что сущности в IDEF1X всегда имеют ключевую область и, поэтому в каждой сущности должны быть определены ключевые атрибуты.

Выбор первичного ключа для сущности является очень важным шагом, и требует большого внимания. В качестве первичных ключей могут быть использованы несколько атрибутов или групп атрибутов. Атрибуты, которые могут быть выбраны первичными ключами, называются кандидатами в ключевые атрибуты (потенциальные атрибуты). Кандидаты в ключи должны уникально идентифицировать каждую запись сущности. В соответствии с этим, ни одна из частей ключа не может быть NULL, не заполненной или отсутствующей.

Например, для того, чтобы корректно использовать сущность СОТРУДНИК в IDEF1X модели данных (а позже в базе данных), необходимо иметь возможность уникально идентифицировать записи. Правила, по которым вы выбираете первичный ключ из списка

предполагаемых ключей, очень строги, однако могут быть применены ко всем типам баз данных и информации. Правила устанавливают, что атрибуты и группы атрибутов должны:

- Уникальным образом идентифицировать экземпляр сущности.
- Не использовать NULL значений.
- Не изменяться со временем. Экземпляр идентифицируется при помощи ключа. При изменении ключа, соответственно меняется экземпляр.
- Быть как можно более короткими для использования индексирования и получения данных. Если вам нужно использовать ключ, являющийся комбинацией ключей из других сущностей, убедитесь в том, что каждая из частей ключа соответствует правилам.

Для наглядного представления о том, как целесообразно выбирать первичные ключи, приведем следующий пример — выберем первичный ключ для знакомой нам сущности «СОТРУДНИК»:

- атрибут «ID сотрудника» является потенциальным ключом, так как он уникален для всех экземпляров сущности СОТРУДНИК;
- атрибут «Имя сотрудника» не очень хорош для потенциального ключа, так как среди служащих на предприятии может быть, к примеру, двое Иванов Петровых;
- атрибут «Номер страхового полиса сотрудника» является уникальным, но проблема в том, что СОТРУДНИКА может не иметь такового;
- комбинация атрибутов «имя сотрудника» и «дата рождения сотрудника» может оказаться удачной для наших целей и стать искомым потенциальным ключом.

После проведенного анализа можно назвать два потенциальных ключа — первый «Номер сотрудника» и комбинация, включающая поля «имя сотрудника» и «Дата рождения сотрудника». Так как атрибут «Номер сотрудника» имеет самые короткие и уникальные значения, то он лучше других подходит для первичного ключа.

При выборе первичного ключа для сущности, разработчики модели часто используют дополнительный (суррогатный) ключ, т. е. произвольный номер, который уникальным образом определяет запись в сущности. Атрибут «Номер сотрудника» является примером суррогатного ключа. **Суррогатный ключ** лучше всего подходит на роль первичного ключа потому, что является коротким и быстрее всего идентифицирует экземпляры в объекте. К тому же суррогатные ключи могут автоматически генерироваться системой так, чтобы нумерация была сплошной, т. е. без пропусков.

Потенциальные ключи, которые не выбраны первичными, могут быть использованы в качестве вторичных или альтернативных ключей. С помощью альтернативных ключей часто отображают различные индексы доступа к данным в конечной реализации реляционной базы.

Если сущности в IDEF1X диаграмме связаны, связь передает ключ (или набор ключевых атрибутов) дочерней сущности. Эти атрибуты называются **внешними ключами**. **Внешние ключи** определяются как атрибуты первичных ключей родительского объекта, переданные дочернему объекту через их связь. Передаваемые атрибуты называются **мигрирующими**.

Классификация сущностей в IDEF1X. Зависимые и независимые сущности

При разработке модели, зачастую, приходится сталкиваться с сущностями, уникальность которых зависит от значений атрибута внешнего ключа. Для этих сущностей (для уникального определения каждой сущности) внешний ключ должен быть частью первичного ключа дочернего объекта.

Дочерняя сущность, уникальность которой зависит от атрибута внешнего ключа, называется **зависимой сущностью**. В примере на рис. 1 сущность СОТРУДНИК является зависимой сущностью потому, что его идентификация зависит от сущности ОТДЕЛ. В обозначениях IDEF1X зависимые сущности представлены в виде закругленных прямоугольников.

Зависимые сущности далее классифицируются на сущности, которые не могут существовать без родительской сущности и сущности, которые не могут быть идентифицированы без использования ключа родителя (сущности, зависящие от идентификации). Сущность СОТРУДНИК принадлежит ко второму типу зависимых сущностей, так как сотрудники могут существовать и без отдела.

Напротив, существуют ситуации в которых сущность зависит от существования другой сущности. Рассмотрим две сущности: ЗАПРОС, используемый для отслеживания запросов покупателей, и ПОЗИЦИЯ ЗАПРОСА, который отслеживает отдельные элементы в ЗАПРОСе. Связь между этими двумя сущностями может быть выражена в виде ЗАПРОС <содержит> один или несколько ПОЗИЦИЙ ЗАПРОСА. В этом случае, ПОЗИЦИЯ ЗАПРОСА зависит от существования ЗАКАЗА.

Сущности, независящие при идентификации от других объектов в модели, называются независимыми **сущностями**. В вышеописанном примере сущность ОТДЕЛ можно считать независимой. В IDEF1X независимые сущности представлены в виде прямоугольников.

Типы связей между сущностями.

Идентифицирующие и неидентифицирующие связи

В IDEF1X концепция зависимых и независимых сущностей усиливается типом взаимосвязей между двумя сущностями. Если вы хотите, чтобы внешний ключ передавался в

дочернюю сущность (и, в результате, создавал зависимую сущность), то можете создать **идентифицирующую связь** между родительской и дочерней сущность.

Идентифицирующие взаимосвязи обозначаются сплошной линией между сущностями

Неидентифицирующие связи, являющиеся уникальными для IDEF1X, также связывают родительскую сущность с дочерней. Неидентифицирующие связи используются для отображения другого типа передачи атрибутов внешних ключей — передача в область данных дочерней сущности (под линией).

Неидентифицирующие связи отображаются пунктирной линией между объектами. Так как переданные ключи в неидентифицирующей связи не являются составной частью первичного ключа дочерней сущности, то этот вид связи не проявляется ни в одной идентифицирующей зависимости. В этом случае и ОТДЕЛ, и СОТРУДНИК рассматриваются как независимые сущности.

Тем не менее, взаимосвязь может отражать зависимость существования, если бизнес правило для взаимосвязи определяет то, что внешний ключ не может принимать значение NULL. Если внешний ключ должен существовать, то это означает, что запись в дочерней сущности может существовать только при наличии ассоциированной с ним родительской записи.

Преимущества IDEF1X

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER и ENALIM является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая, несомненно, является значительным недостатком ER.

Приложение 3

Некоторые команды и функции для работы с базой данных

Далее при описании команд приводится их полный синтаксис. Следует помнить, что элементы команд, заключенные в квадратные скобки, являются необязательными, могут присутствовать или отсутствовать. Если элементы разделены вертикальной чертой - должен присутствовать один из них.

Полный список команд и функций с разъяснением их параметров и примерами использования можно найти в справочной системе VFP в разделе Language Reference.

Команды для работы с базами данных, таблицами, индексами, связями

- Создать базу

```
CREATE DATABASE [DatabaseName | ?]
```

- Создать таблицу базы данных (SQL-команда)

```
CREATE TABLE | DBF TableName1 [NAME LongTableName] [FREE]
    [CODEPAGE = nCodePage]
    ( FieldName1 FieldType [( nFieldWidth [, nPrecision] )] [NULL | NOT
NULL]
    [CHECK lExpression1 [ERROR cMessageText1]]
    [AUTOINC [NEXTVALUE NextValue [STEP StepValue]]] [DEFAULT eExpression1]
    [PRIMARY KEY | UNIQUE [COLLATE cCollateSequence]]
    [REFERENCES TableName2 [TAG TagName1]] [NOCPTRANS]
    [, FieldName2... ]
    [, PRIMARY KEY eExpression2 TAG TagName2 |, UNIQUE eExpression3 TAG
TagName3
    [COLLATE cCollateSequence]]
    [, FOREIGN KEY eExpression4 TAG TagName4 [NODUP]
    [COLLATE cCollateSequence]
    REFERENCES TableName3 [TAG TagName5]] [, CHECK lExpression2 [ERROR
cMessageText2]] )
    | FROM ARRAY ArrayName
```

- Открыть базу данных, представление (View) или таблицу базы

```
USE [[DatabaseName!]Table | SQLViewName | ?]
   [IN nWorkArea | cTableAlias] [ONLINE] [ADMIN] [AGAIN]
   [NOREQUERY [nDataSessionNumber]] [NODATA] [INDEX IndexFileList | ?
   [ORDER [nIndexNumber | IDXFileName   | [TAG] TagName [OF CDXFileName]
      [ASCENDING | DESCENDING]]]]   [ALIAS cTableAlias]   [EXCLUSIVE]
   [SHARED]   [NOUPDATE]   [CONNSTRING cConnectionString |
(m.nStatementHandle) ]
Функция ALIAS([nWorkArea]) возвращает алиас для текущей или заданной рабочей
зоны.
```

- Выбрать свободную рабочую зону (0), заданную рабочую зону или выбрать таблицу

```
SELECT([ 0 | 1 | cTableAlias ])
   SELECT(0) - возвращает номер выбранной рабочей зоны
   SELECT(1) - возвращает наибольший номер свободной зоны
   SELECT 0 - выбор свободной зоны с наименьшим номером
```

- Создать индексный файл

```
INDEX ON eExpression TO IDXFileName | TAG TagName
   [COLLATE cCollateSequence] [OF CDXFileName] [FOR lExpression]
   [COMPACT] [ASCENDING | DESCENDING] [UNIQUE | CANDIDATE] [ADDITIVE]
   [BINARY]
```

- Открыть индекс

```
SET INDEX TO [IndexFileList | ? ]
 ORDER nIndexNumber | IDXIndexFileName |
 [TAG] TagName [OF CDXFileName] [ASCENDING |
 DESCENDING]] [ADDITIVE]
```

- Установить порядок по индексу

```
SET ORDER TO [nIndexNumber | IDXIndexFileName |
```

```
 [TAG] TagName [OF CDXFileName] [IN nWorkArea |
 cTableAlias][ASCENDING |  DESCENDING]]
```

- Установить связь между таблицами

```
SET RELATION TO [eExpression1 INTO nWorkArea1 |
 cTableAlias1 [, eExpression2 INTO nWorkArea2 |
 cTableAlias2 ] [IN nWorkArea | cTableAlias]
 [ADDITIVE]]
```

- Установить множественную связь между таблицами (используется при формировании отчетов)

```
SET SKIP TO [TableAlias1 [, TableAlias2] ...]
```

Команды перемещения по таблице, поиска и отбора данных

- Перейти к записи...

```
GO [RECORD] nRecordNumber [IN nWorkArea |
    IN cTableAlias]
GO TOP | BOTTOM [IN nWorkArea | IN cTableAlias]
   (вместо GO можно использовать GOTO)
```

- Переместиться по таблице (вперед или назад)

```
SKIP [nRecords] [IN nWorkArea | cTableAlias]
```

Для nRecords>0 - перемещение далее по таблице, для nRecords<0 - назад к предыдущим записям.

Функция BOF() возвращает .T., если текущая запись - первая и Вы пытаетесь выполнить команду SKIP -1, аналогично для последней записи - EOF()=.T.

- Поиск для заданного логического условия

```
LOCATE [FOR lExpression1] [Scope] [WHILE]
 [NOOPTIMIZE]
```

(Найти следующую запись, соответствующую условию - команда CONTINUE)

- Поиск по значению индекса

```
SEEK eExpression ORDER nIndexNumber |
    IDXIndexFileName | [TAG] TagName
    [OF CDXFileName] [ASCENDING | DESCENDING]]
    [IN nWorkArea | cTableAlias]
```

- Установить фильтр

```
SET FILTER TO [lExpression]
     [IN nWorkArea | cTableAlias]
```

- Выполнить запрос (SQL-команда)

```
SELECT [ALL | DISTINCT] [TOP nExpr [PERCENT]] Select_List_Item [, ...]
   FROM [FORCE] Table_List_Item [, ...]
      [[JoinType] JOIN DatabaseName!]Table[[AS] Local_Alias]
      [ON JoinCondition [AND | OR [JoinCondition | FilterCondition] ...]
   [WITH (BUFFERING = lExpr)]
   [WHERE JoinCondition | FilterCondition [AND | OR JoinCondition |
FilterCondition] ...]
   [GROUP BY Column_List_Item [, ...]] [HAVING FilterCondition [AND |
OR ...]]
   [UNION [ALL] SELECTCommand]
   [ORDER BY Order_Item [ASC | DESC] [, ...]]
   [INTO StorageDestination | TO DisplayDestination]
   [PREFERENCE PreferenceName] [NOCONSOLE] [PLAIN] [NOWAIT]
```

В качестве StorageDestination можно использовать одно из следующих предложений:

- ARRAY ArrayName - в массив переменных памяти;
- CURSOR CursorName - в курсор;
- DBF TableName | TABLE TableName - в таблицу.

В качестве DisplayDestination можно использовать одно из следующих предложений:

- FILE FileName [ADDITIVE] - ASCII текстовый файл;
- PRINTER [PROMPT] - вывод на принтер;
- SCREEN - в главное окно системы VFP.

Команды для добавления, модификации и удаления данных

- Открыть окно для работы в табличном формате с таблицей базы данных:

```
BROWSE [FIELDS FieldList] [FONT cFontName [, nFontSize]]
   [STYLE cFontStyle] [FOR lExpression1 [REST]] [FORMAT]
   [FREEZE FieldName] [KEY eExpression1 [, eExpression2]] [LAST | NOINIT]
   [LOCK nNumberOfFields] [LPARTITION] [NAME ObjectName] [NOAPPEND]
   [NOCAPTIONS] [NODELETE] [NOEDIT | NOMODIFY] [NOLGRID] [NORGRID]
   [NOLINK] [NOMENU] [NOOPTIMIZE] [NOREFRESH] [NORMAL] [NOWAIT]
   [PARTITION nColumnNumber [LEDIT] [REDIT]]
   [PREFERENCE PreferenceName] [SAVE] [TIMEOUT nSeconds]
   [TITLE cTitleText] [VALID [:F] lExpression2 [ERROR cMessageText]]
   [WHEN lExpression3] [WIDTH nFieldWidth] [WINDOW WindowName1]
   [IN [WINDOW] WindowName2 | IN SCREEN] [COLOR SCHEME nSchemeNumber]
```

При описании полей (в параметре FIELDS) список может содержать следующие параметры:

```
FieldName                                  (имя поля)
   [:R]                                                (только чтение)
   [:nColWidth]                                  (ширина поля)
[:V = lExpr1 [:F] [:E = cTxt]]            (функция, выполняемая при выходе из
поля)
   [:P = cFormatCodes]                     (формат)
   [:B = eMin, eMax [:F]]                     (диапазон данных)
   [:H = cHeadingText]                        (заголовок)
   [:W = lExpr2]                                      (функция,выполняемая
                                               перед входом в поле)
```

Близкий синтаксис имеют команды EDIT и CHANGE для работы с таблицей при построчном расположении полей.

- Добавление записей

```
APPEND [BLANK] [IN nWorkArea | cTableAlias] [NOMENU]
APPEND FROM FileName | ? [FIELDS FieldList]
   [FOR lExpression][[TYPE] [DELIMITED
 [WITH Delimiter | WITH BLANK | WITH TAB |
 WITH CHARACTER Delimiter] | DIF | FW2 | MOD |
 PDOX | RPD | SDF | SYLK | WK1 |WK3 | WKS | WR1 |
 WRK | CSV | XLS | XL5 [SHEET cSheetName]| XL8
 [SHEET cSheetName]]] [AS nCodePage]

APPEND FROM ARRAY ArrayName [FOR lExpression] [
   FIELDS FieldList | FIELDS LIKE Skeleton |
   FIELDS EXCEPT Skeleton]
```

SQL-команда INSERT INTO - добавить запись с заданными значениями полей:

```
INSERT INTO dbf_name [(FieldName1 [, FieldName2, ...])]
```

```
   VALUES (eExpression1 [, eExpression2, ...])
   INSERT INTO dbf_name FROM ARRAY ArrayName | FROM MEMVAR | FROM NAME
ObjectName
INSERT INTO dbf_name [(FieldName1 [, FieldName2, ...])]
   SELECT SELECTClauses [UNION UnionClause SELECT SELECTClauses ...]
```

- Занести данные в поля таблицы

```
REPLACE FieldName1 WITH eExpression1 [ADDITIVE]
   [, FieldName2 WITH eExpression2 [ADDITIVE]]
   ...[Scope][FOR lExpression1][WHILE lExpression2]
  [IN nWorkArea | cTableAlias][NOOPTIMIZE]
```

- Копировать данные текущей записи в массив переменных

```
SCATTER [FIELDS FieldNameList | FIELDS LIKE
 Skeleton | FIELDS EXCEPT Skeleton] [MEMO]
 TO ArrayName | TO ArrayName BLANK | MEMVAR
 | MEMVAR BLANK | NAME ObjectName [BLANK]
```

параметр MEMVAR означает - используются переменные с теми же именами, что и имена полей записи; имя переменной в программе следует писать как m.<имя поля>).

- Копировать данные из массива переменных в текущую запись

```
GATHER FROM ArrayName | MEMVAR | NAME ObjectName
  [FIELDS FieldList | FIELDS LIKE Skeleton |
   FIELDS EXCEPT Skeleton] [MEMO]
```

- Копирование данных таблицы в массив

```
COPY TO ARRAY ArrayName [FIELDS FieldList |
   FIELDS LIKE Skeleton | FIELDS EXCEPT Skeleton]
 [Scope] [FOR lExpr1a] [WHILE lExpression2]
 [NOOPTIMIZE]
```

- Выполнить обновление данных (SQL-команда)

```
UPDATE Target
   SET Column_Name1 = eExpression1 [, Column_Name2 = eExpression2 ...]
   [FROM [FORCE] Table_List_Item [[, ...] | [JOIN [ Table_List_Item]]]
   WHERE FilterCondition1 [AND | OR FilterCondition2 ...]
```

Здесь Target - таблица (table), курсор (cursor) или их алиас или файл для обновления.

- Пометить записи таблицы для удаления записи

```
DELETE [Scope] [FOR lExpression1] [WHILE lExpression2]
   [IN nWorkArea | cTableAlias] [NOOPTIMIZE]
```

- Удалить помеченные для удаления записи из файла

```
PACK [MEMO | DBF] [Tablename ] [IN nWorkarea | cTableAlias]
```

- Выполнить удаление записей (SQL-команда)

```
DELETE [Target] FROM [FORCE] Table_List [[, Table_List ...] | [JOIN
[ Table_List]]]
   [WHERE FilterCondition1 [AND | OR FilterCondition2 ...]]
```

Команды вычислений по данным таблиц

- CALCULATE eExpressionList [Scope] [FOR lExpression1] [WHILE lExpression2][TO VarList | TO ARRAY ArrayName] [NOOPTIMIZE] [IN nWorkArea | cTableAlias]

где для eExpressionList - можно использовать следующие функции:

```
AVG(nExpression) - среднее значение
CNT( )             - количество
MAX(eExpression) - максимальное значение
MIN(eExpression) - минимальное значение
NPV(nExpression1, nExpression2 [, nExpression3])
  - банковская функция
```

```
STD(nExpression) - стандартное отклонение
SUM(nExpression) - сумма
VAR(nExpression) - статистическая функция

SUM [eExpressionList]   [Scope] [FOR lExpression1]
   [WHILE lExpression2][TO MemVarNameList |
   TO ARRAY ArrayName]   [NOOPTIMIZE]

AVERAGE [ExpressionList]   [Scope] [FOR lExpression1]
   [WHILElExpression2][TO VarList |
   TO ARRAY ArrayName]   [NOOPTIMIZE]

COUNT [Scope] [FOR lExpression1] [WHILE lExpression2]
   [TO VarName][NOOPTIMIZE]
```

Математические функции

```
^  *  /  +  -  ABS() ACOS() ASIN() ATAN() ATN2() AVG()
BINTOC() BITAND() BITCLEAR() BITLSHIFT() BITRSHIFT() BITSET() BITTEST() BITXOR()
CEILING() COS() COUNT() DTOR() EXP() FLOOR() FV() INT() LOG() LOG10() MAX() MIN()
MOD() MTON() NTOM() PAYMENT() PI() PV() RAND() ROUND(,) RECCOUNT() RECNO() RTOD()
SIGN() SIN() SQRT() SUM() TAN() VAL()
```

Функции для операций с текстовыми данными

```
+  -  ASC()  ALLTRIM()  AT(,,)  ATC(,,)  CHR()  CHRTRAN(,,)  CTOBIN()
CURSORTOXML(,,,,)  CURVAL(,)  FILETOSTR()  GETPEM(,)  GETWORDCOUNT(,)
GETWORDNUM(,)
LEFT(,)  LEN()  LOWER()  LTRIM()  MAX(,)  MIN(,)  OCCURS(,)  OEMTOANSI()
OLDVAL(,)  PADC(,)
PADL(,)  PADR(,) PEMSTATUS(,,)  PROPER()  RAT(,,)  REPLICATE(,)  RIGHT(,) RTRIM()
SOUNDEX()
SPACE()  STR(,,)  STREXTRACT(,,,,)  STRTRAN(,,)  STUFF(,,,)  SUBSTR(,,)
TRANSFORM(,)  TRIM()
TYPE()  UPPER()  $  AT_C(,,)  ATCC(,,)  CHRTRANC(,,)  LEFTC(,)  LENC( )  RATC(,,)
RIGHTC(expC,)
TEXTMERGE(,,,)  STRCONV(,)  STUFFC(,,,)  SUBSTRC(,,)
```

Функции для операций с данными типа "дата" и "время"

```
{date} CDOW() CMONTH() CTOD() CTOT() DATE() DATETIME() DAY() DMY() DOW()
DTOC() DTOS() DTOT() GOMONTH(,) HOUR() MAX(,) MDY() MIN(,) MINUTE() MONTH()
QUARTER()
SEC() SECONDS() TIME() TTOC() TTOD() WEEK() YEAR()
```

Логические функции

```
<  >  =  <  >  <=  >=  ==  .T.  .F.  .NULL.  NOT  AND  OR  BETWEEN(,,)
DELETED()  EMPTY()  IIF(,,) INLIST(,,)  NVL(,)  SEEK(,,)
```

Приложение 4

Сводный перечень государственных и международных стандартов, используемых при курсовом и дипломном проектировании

Наименование документа	Стандарт
Пояснительная записка	ГОСТ 7.32-2001 СИБИД. Отчет о научно-исследовательской работе. Структура и правила оформления. ГОСТ 2.105—95 ЕСКД. Общие требования к текстовым документам. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание**.** ГОСТ 7.12—93 СИБИД. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила. ГОСТ 8.417-2002 Государственная система обеспечения единства измерений. Единицы величин. ГОСТ 9327—60 Бумага и изделия из бумаги. Потребительские форматы. ГОСТ 24.104-85 АСУ. Общие требования. ГОСТ 34.003-90 "Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения". ГОСТ 34.201-89 "Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем". ГОСТ 34.601-90 "Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания". ГОСТ 34.602-89 "Информационная технология. Комплекс

	стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы".
Чертежи	ГОСТ 2.109-73 ЕСКД. Основные требования к чертежам. ГОСТ 2.119-73. ЕСКД. Эскизный проект. ГОСТ 2.120-73 ЕСКД. Технический проект. ГОСТ 24.304-82 АСУ. Требования к выполнению чертежей.
Схемы	ГОСТ 2.701-84 Правила выполнения схем. ГОСТ 2.711-82 ЕСКД. Схема деления изделия на составные части.
Программы и программные документы	ГОСТ 19.001-77 ЕСПД. Общие положения. ГОСТ 19.005-85 ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов. ГОСТ 19.102-77 ЕСПД. Стадии разработки ГОСТ 19.103-77 ЕСПД. Обозначения программ и программных документов. ГОСТ 19.104-78 ЕСПД. Основные надписи. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению. ГОСТ 19.402-78 ЕСПД. Описание программы. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению. ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению. ГОСТ 19.603-78 ЕСПД. Общие правила внесения изменений. ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом. ГОСТ 19.701-90 (ИСО 5807-85) ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
Управление производственными процессами	**MRPII** - методология планирования потребности в материалах в производственных процессах. Эта система была создана для эффективного планирования всех ресурсов производственного предприятия, в том числе финансовых и кадровых. Кроме того, система класса MRRPII способна адаптироваться к изменениям внешней ситуации и эмулировать ответ на вопрос "Что если". **ERP** - технология оптимизации производственного процесса с точки зрения производственных, коммерческих и финансовых целей. Основная цель оптимизации организации производства и управления предприятием - максимальный уровень сервиса для потребителей, минимальные вложения в основные фонды и эффективная, с точки зрения низкого уровня издержек, работа предприятия.
Описание, анализ и реорганизация процессов	**IDEF0 - Function Modeling** - используется для создания функциональной модели, которая является структурированным отображением функций производственной системы или среды, а также информации и объектов, связывающих эти функции.; **IDEF1 - Information Modeling** - применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы или среды; **IDEF1X – Data Modeling -** является методом для разработки реляционных баз данных;

	DFD – (методология Gane / Sarson) построение модели анализируемой ИС - проектируемой или реально существующей. В соответствии с методологией **модель системы** определяется как иерархия диаграмм потоков данных (ДПД или DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. **IDEF3 – (Process Description Capture)** - методология документирования процессов, происходящих в системе. С помощью IDEF3 описываются сценарий и последовательность операций для каждого процесса. IDEF3 напрямую связана с методологией IDEF0: каждая функция (функциональный блок) может быть представлена средствами IDEF3 в виде отдельного процесса.