Н. Г. ЯРУШКИНА, Т.А. МЕРКУЛОВА

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ ДЛЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ В СРЕДЕ СЕРВЕРА ДАННЫХ ORACLE 7.3.

Рекомендовано Учебно-методическим объединением по образованию в области прикладной информатики в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальности 351400 «Прикладная информатика в экономике» и другим междисциплинарным специальностям

Ульяновск 2002

УДК 683.03 (075)
ББК 32.965 я 7
Я 78

Рецензенты: заведующий кафедрой «Проектирование экономических информационных систем» Московского государственного университета экономики, статистики и информатики (МЭСИ) кандидат экономических наук, профессор Тельнов Ю.Ф., заведующий кафедрой управления Ульяновского государственного университета (УлГУ), доктор экономических наук, профессор Кац И. Я.; кафедра математической кибернетики и информатики Ульяновского государственного университета (УлГУ).

Утверждено редакционно-издательским советом университета в качестве учебного пособия

Ярушкина Н. Г., Меркулова Т. А.
Я 78 Проектирование баз данных для экономических информационных систем в среде сервера данных ORACLE 7.3.: Учебное пособие. – Ульяновск: УлГТУ. – 2002. – с 160.

ISBN

Предназначено для преподавания дисциплины «Базы данных и знаний», частично может использоваться для преподавания дисциплин «Теория экономических информационных систем» «Проектирование экономических информационных систем» студентам специальности 351400 «Прикладная информатика (в экономике)». Может быть использовано для преподавания ряда дисциплин студентам специальности 071900 «Информационные системы и технологии», студентам специальности «Прикладная математика и информатика». Учебное пособие включает в себя описание состава, языка SQL сервера данных ORACLE. В настоящее время на предприятиях осуществляется переход от использования персональных баз данных технологии x base к «тяжелым» серверам данных типа ORACLE. Методическая литература или отсутствует или представлена фирменной технической документацией. Пособие включает в себя 18 занятий (упражнений), которые выполняются студентом в интерактивном режиме. Пособие включает часть, описывающую языки проектирования баз данных информационных систем, международные стандарты IDEF. Главы, посвященные языкам и стандартам проектирования, позволят студенту грамотно оформить курсовой и дипломный проект.
Работа подготовлена на кафедре «Информационные системы» Ульяновского государственного технического университета.

УДК 683.03 (075)
ББК 32.965 я 7

Оглавление

Введение

Учебное пособие разработано для студентов специальности 351400 «Прикладная информатика (в экономике)». Может быть использовано для преподавания дисциплины «Базы данных и знаний», частично использовано для преподавания дисциплин «Теория экономических информационных систем» и «Проектирование экономических информационных систем». Учебное пособие может быть использовано для организации различных видов занятий: практических занятий, лабораторных работ и курсового проектирования.

Главы 1, 2, 3 содержат материал, охватывающий основные виды диаграмм, используемых при проектировании баз данных; основные тенденции развития технологии баз данных в информационных системах; описание сервера данных ORACLE 7.3 с точки зрения администрирования; язык SQL. Глава 1 посвящена методам проектирования баз данных. Технологическая культура создания и эксплуатации баз данных упала на большинстве российских предприятий. Поэтому овладение методами структурного проектирования, международными стандартами функционального моделирования IDEF0, IDEF1, IDEF3, IDEF5 очень важно для сегодняшних студентов. Материал главы 1 используется студентами при выполнении курсового проектирования, позволяет им грамотно оформить пояснительную записку проекта.

Вторая глава посвящена перспективам развития технологии баз данных. Рассмотрено современное состояние распределенных, гетерогенных экономических ИС. Описаны перспективы баз данных персональных ЭВМ семейства x-base. Показаны возможности временных (темпоральных) и пространственных баз данных. Важнейшим вопросам безопасности баз данных посвящен отдельный пункт.

Глава 3 рассматривает функции серверов данных, основы теории транзакций, стандарты SQL-языка. Конкретные примеры приведены для сервера данных ORACLE 7.3. Содержание главы используется студентами при выполнении лабораторных работ, посвященных изучению SQL-языка. Упражнения и задачи приведены в главе 4. Инструментальным средством служит компонента ORACLE сервера SQL-Plus.

Глава 4 содержит задания, сгруппированные в 18 упражнений. На их основе можно организовать семестровый курс лабораторных работ, позволяющий усвоить как основы SQL-языка, так и сложные вложенные, коррелированные запросы. Изучаются системные средства

распределения полномочий, создания индексов, использования словаря данных.

Глава 5 представляет собой описание комплексной бухгалтерии для гипотетического предприятия. Описаны информационное обеспечение, функции автоматизированных рабочих мест, информационные связи между АРМ. Студентам в рамках курсового проектирования предлагается создать одно из таких рабочих мест. Курсовой проект приближен к практике создания и эксплуатации информационных систем на предприятиях. Ряд курсовых проектов рассчитан на работу в группе.

Список сокращений

ИС – информационная система
ПО – программное обеспечение
БД – база данных
АРМ – автоматизированное рабочее место
ЛВС – локальная вычислительная сеть
ГСМ – горюче-смазочные материалы
ТТН – товарно-транспортные накладные
З/П – заработная плата
СА – системный аналитик
ЭИС – экономическая ИС
CASE – средства автоматизированной поддержки программирования
САПР – системы автоматизированного проектирования
АСУ – автоматизированная система управления
АСПР – автоматизированная система программирования.
ОС – операционная система
DFD – диаграмма потоков данных
СД – сервер данных
МД – метаданные
РТ – рабочая таблица
АТД – абстрактный тип данных
ООБД – объектно-ориентированная база данных
GUI – графический интерфейс пользователя
ОС и НМА – основные средства и нематериальные активы
М/ч – машино-час
Т – тонна
АТП – автотранспортное предприятие

Глава 1. Основные виды диаграмм, используемых при проектировании баз данных информационных систем

1. Моделирование данных

1.1. Виды диаграмм, используемых в проектировании данных

В ходе анализа данные итерационно уточняются от логических или концептуальных моделей до физических. Физические модели отражаются на языке реализации, а концептуальные инфологические модели отображают реальность проблемной области. В условиях почти ежегодных технологических революций логические модели могут оказаться более долговечной компонентой ИС, чем фрагменты исходного кода. В настоящее время в основном используются логические модели, имеющие тип <сущность> – <связь>, (entity ↔ relationship ER-диаграммы).
Объектно-ориентированное программирование вызвало формирование новых объектно-ориентированных методов проектирования. Тем не менее, ER-диаграммы в модифицированном виде сохраняются и при объектно-ориентированном подходе.

1.2. Виды ER-диаграмм. Определение и особенности ER-диаграмм

ER-диаграмма математически представляет собой двудольный граф, в котором сущности отображаются прямоугольником, а отношение – ромбом (рис. 1.1.).

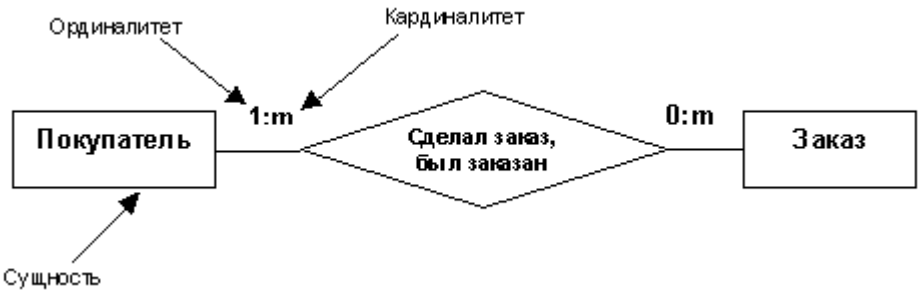


Рис. 1.1 . Элементы ER-диаграмм

Сущностями обычно называют объекты реального мира, роли пользователей (коллектива), события. Отношения характеризуют взаимодействие сущностей. Любое отношение двух сущностей характеризуется двумя значениями количества взаимодействующих сущностей, а именно минимальным и максимальным значимыми количествами взаимодействующих сущностей (ординалитетом и кардиналитетом). Ординалитет характеризует минимальное количество представителей одной сущности, вступающей в указанное отношение с представителем другой сущности. Кардиналитет (кардинальность) – это максимальное количество представителей сущностей, которые вступают в указанные отношения с представителями другой сущности.

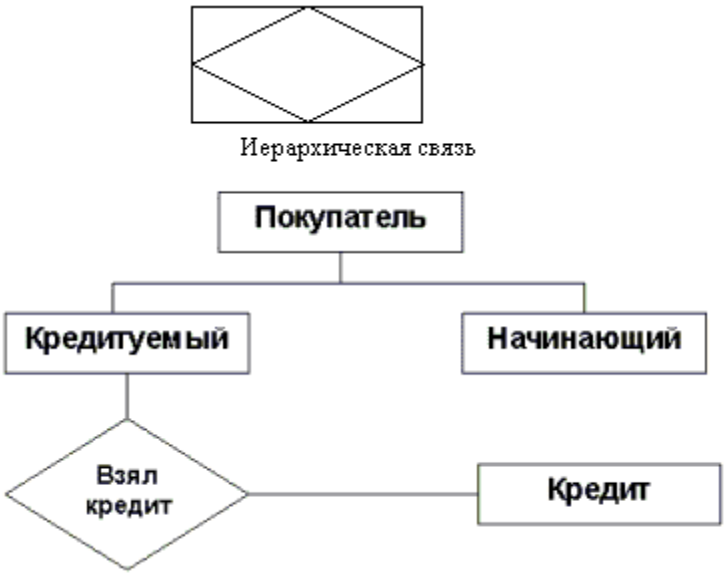


Рис. 1.2 Пример ER-диаграммы

Сущности образуют между собой естественные иерархии (рис. 1.2), то есть можно говорить о сущностях суперклассов и о сущностях подкласса. На ER-диаграммах иерархические связи обычно отражаются соединительными линиями. Причем, подклассы могут вступать в отношения, не характерные для суперкласса, а отношения суперклассов наследуются подклассами.

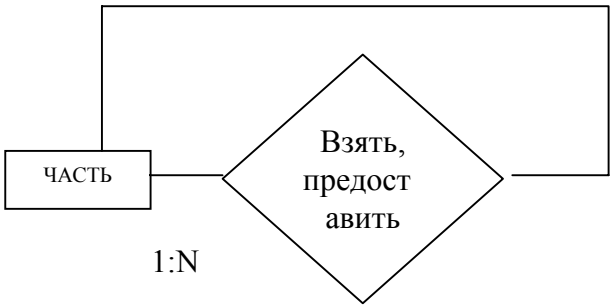


Рис. 1.3 Отношение часть-целое

Кроме естественных иерархий сущности реального мира могут быть связаны отношением часть – целое (рис. 1.3.), причем, часть и целое – представители одной и той же сущности. Такие отношения отображаются унарными ER-диаграммами.

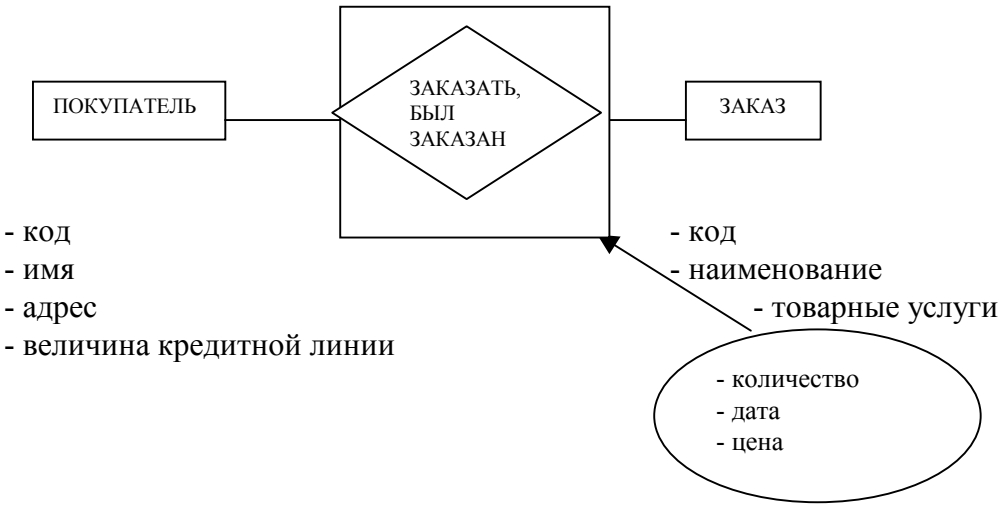


Рис. 1.4. Ассоциативное отношение

Для многих проблемных областей границы между сущностями и отношениями теряются, так как атрибуты могут принадлежать отношению. Такие отношения называют ассоциативными отношениями и изображают в виде ромба, вписанного в прямоугольник (рис. 1.4.).

На практике часто такие ассоциативные отношения возникают в случае отношения <многие> – <ко многим>. Ассоциативные отношения часто бывают полезны при моделировании отношений <многие> – <ко многим>. Такого рода отношения на ER-диаграмме отличаются тем, что левый и правый кардиналитет помечены как m. Описанная выше аннотация была предложена Питером Ченом [10].

ER-диаграмма МАРТИНА

В диаграмме Мартина введены новые обозначения ординалитетов и кардиналитетов (рис. 1.5.).

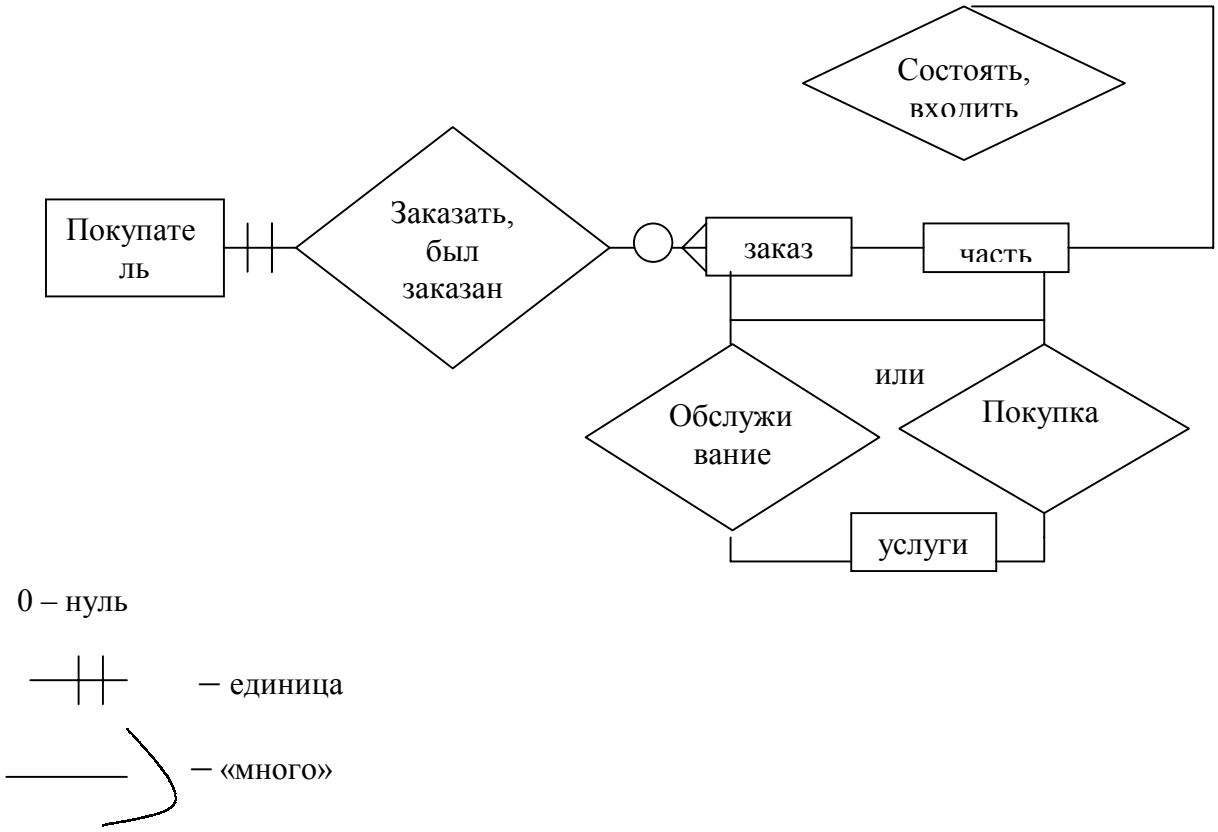


Рис. 1.5. ER-диаграмма Мартина

Помимо соединительных линий диаграмм Чена вводятся логические линии между отношениями, отношения «или», «и».

ER-диаграмма Бечмана

Рис.1.6. ER-диаграмма Бечмана

Диаграммы Бечмана (рис. 1.6.) подчеркивают важность читабельности ER-диаграмм, причем в результате чтения должны

складываться фразы на естественном языке. Поэтому Бечман ввел специальные знаки для отображения модальности: ● – модальность – должен, ❍ –модальность возможности. На схемах Бечмана отображается только кардиналитет, стрелка означает – многие, а отсутствие стрелки – один экземпляр.

1.3. Шаги формирования ER-диаграмм

ШАГ 1 – выделить все сущности, моделируемые в проблемной области (источником выделения сущностей является либо обсуждение с пользователем, либо бумажные или компьютерные формы).
ШАГ 2 – выделить значимые отношения. Если сущности выделены, то можно проанализировать с помощью матрицы все возможные отношения между сущностями. Некоторые отношения не существуют или в контексте данной задачи несущественны.
ШАГ 3 – составить ER-диаграммы и расставить кардиналитеты отношений. Кардиналитеты не только характеризуют отношения, но и позволяют исключить ложные сущности, то есть сущности за которыми не стоят многие экземпляры.
ШАГ 4 – выделить атрибуты и ключи для каждой сущности. Первичным ключом называют совокупность атрибутов, уникальную для каждого экземпляра.
ШАГ 5 – выполнить проверку предыдущих шагов, составить окончательную подробную ER – диаграмму.

2. Моделирование процессов

2.1. Определение потоковых диаграмм

При проектировании ИС с базами данных процессы выявления сущностей и уточнения их структуры итеративно повторяются. Сущности выявляются при анализе и моделировании деловых процессов, которые часто описывают с помощью потоковой диаграммы. Диаграммой потоков данных (Data flow diagramm DFD) называют схему, отображающую связи между процессами, БД и пользователем. Самой распространенной потоковой диаграммой является диаграмма, предложенная Де Марко и Джорданом [10]. Основными элементами DFD являются элементы, представленные на рис. 1.7

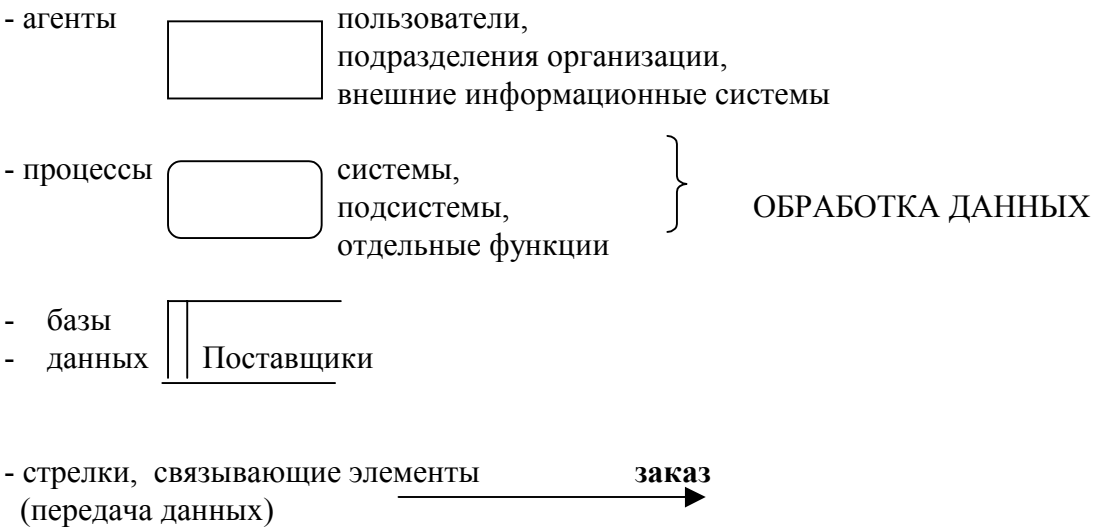


Рис. 1.7. Элементы DFD-диаграмм

DFD не описывают конкретный алгоритм преобразования информации, поэтому принципиально отличны от блок-схем алгоритмов. DFD изображают преобразование данных в укрупненной форме на стадии технического, а не детального проекта. При формировании DFD необходимо избегать следующих ошибок:

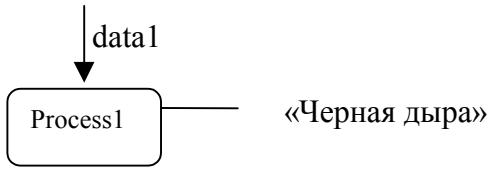


Рис. 1.8. Ошибка DFD «Черная дыра»

Каждый процесс кроме входов должен обязательно иметь выход, то есть выходные данные направляются в БД или другому процессу. Процессы, не имеющие выхода, называют «Черной дырой» (рис. 1.8.).

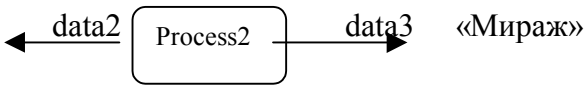


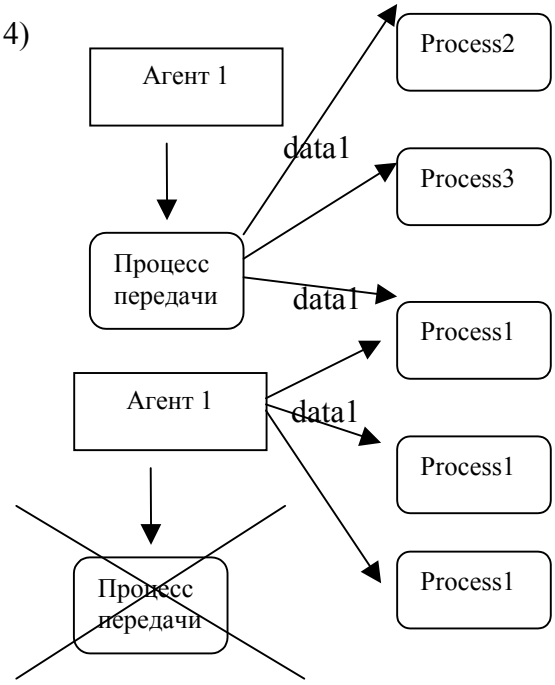
Рис. 1.9. Ошибка DFD «Мираж»

Процесс, не имеющий входа, а имеющий один или несколько выходов, является ошибочным. Обычно такой процесс называют «Мираж» (рис. 1.9.).

3) Наиболее типичной ошибкой для потоковых диаграмм является изображение процесса, имеющего и входы и выходы, но входы в

процесс недостаточны для формирования указанных выходов. Например, для процесса указали только вход от агента с экрана дисплея, но не указали вход от справочников, такие процессы на DFD называют «Серой дырой».

4)



Должно быть без посредников

Рис. 1.10 Ошибка использования посредников в DFD

При проектировании прикладной информационной системы DFD должны включать в себя только значимые процессы, преобразующие информацию. Процессы посредников, передающие, но не преобразующие данные должны быть исключены (рис. 1.10).

Среди соединительных линий в зависимости от степени детализации выделяют сложные соединительные линии и примитивные. Примитивные соединительные линии изображают передачу отдельных конкретных данных, например, паспортные данные. Сложные – отвечают за передачу интегральных данных, например, сведений о кадрах и встречаются на самых общих потоковых диаграммах. Автоматизация разработки DFD в современных CASE-системах сводится к специализированным графическим редакторам, которые называют диаграммерами.

2.2. Этапы разработки потоковых диаграмм

Разработка контекста (окружения) диаграммы верхнего уровня

Контекстная потоковая диаграмма содержит только один процесс, эквивалентный разрабатываемой системе. Такой процесс обычно изображают в центре листа, а по краям листа располагают основных агентов и известные БД.

Декомпозиция процесса

Декомпозицией процессов с точки зрения формирования потоковой диаграммы называют разбиение общего описания процесса на подпроцессы. Этап декомпозиции процесса может быть представлен набором иерархических диаграмм последовательно расположенных функций. Подход с точки зрения декомпозиции процессов характерен для крупных проектов, для небольших обозримых систем может использоваться подход расширения. Подход расширения диаграммы исключает использование функциональных иерархий, допустимы только потоковые диаграммы. Если произошла декомпозиция одного процесса на несколько, то следующая потоковая диаграмма содержит как порожденный процесс, так и все их связи с агентами и БД, то есть следующая диаграмма расширяет предыдущую.

Перечисление и уточнение всех БД и всех передаваемых данных

Перечисление и уточнение данных должно выполняться согласованно с ER-диаграммами, если их моделирование выполнялось правильно. Для бизнес-приложений обычно в начале моделируют данные, затем процессы. А для научно-технических расчетов обычно сначала моделируют процессы.

Разработка уточненной потоковой диаграммы
Итерационное повторение декомпозиции потоковой диаграммы

Процесс итерации заканчивается, если на всех DFD изображены только примитивные процессы и примитивные соединительные линии. Процесс называют примитивным, если он представлен в спецификации на программирование и не нуждается в детализации.

3. Моделирование сетей, концептуальные модели сетей

3.1. Понятие о концептуальном моделировании сетей. Виды сетевой обработки

При проектировании сетей можно выделить два уровня проекта: а) концептуальный; б) реализационный. Концептуальный уровень сети характеризует распределение бизнес-процессов в пространстве, а уровень реализации описывает конкретные технологии и протокол. Концептуальные модели сетей хоть и лишены деталей технической реализации, тем не менее, отражают основную массу сетевой обработки.

Основные типы сетевой обработки следующие.

1. Централизованная обработка – для центральной обработки характерна организация вычислений на одной мощной машине, то есть через терминальную сеть многие пользователи разделяют один процессор в режиме разделения времени. Следовательно, концептуальная модель сети разделяет пространство на машинный зал и терминалы по местам расположения.
2. Автономно стоящий персональный компьютер – концептуальная модель обработки сведется к расположению ПК и к порядку обмена носителями между этими компьютерами.
3. Децентрализованная обработка (распределенная) – эта обработка связана с распределением средств автоматизации бизнес-процессов в пределах локальной вычислительной сети (Local area network LAN).

   Концептуальная схема сети для распределенной обработки характеризуется появлением понятия глобальных и локальных ресурсов. Локальные – собственные ресурсы рабочих станций. Глобальные ресурсы хранятся на серверах и доступны по сети другим компьютерам. Концептуальная модель распределенной обработки включает в себя не просто схему расположения компьютера, но и схему распределения ресурсов по бизнес-процессам (Wide area network WAN). Развитие локальных сетей привело к формированию сетей, действующих в рамках предприятий. В таких сетях обычно выделяют уровни: предприятие, управление, подразделение, рабочее место. Поэтому концептуальная модель сети в масштабе корпорации должна быть многоуровневой. На уровне предприятия единицей сети могут быть отдельные сети подразделений и управления.
4. Кооперативные вычисления – при организации кооперативных вычислений несколько узлов сети могут работать над вычислением

одной задачи, то есть формируется кооператив рабочих узлов. Компьютеры, работающие в одном кооперативе, должны синхронизировать действия. Кооперативные вычисления редко используют в бизнес-приложении, но достаточно часто в науке или высоко-технологичной промышленности. Концептуальная модель кооперативной сетевой обработки должна включать в себя формирование вычислений.

5. Клиент-серверная обработка появилась на базе распределенной обработки и включает в себя все ее особенности, дополнительно клиент-серверная обработка характеризуется появлением ряда новых информационных технологий. Такие технологии в настоящее время представляют собой способ разработки программного обеспечения, при котором задачу решает как сервер, так и клиент за счет конкретных ресурсов. Примерами клиент-серверных технологий могут служить: сервера данных, отвечающих на sql-запросы; proxy-сервер, предоставляющий клиентам Интернет-сервис.

Концептуальные модели клиент-серверной обработки должны отражать расположение серверов конкретного вида и группы их клиентов. Группу клиентов важно указать, так как в будущем группы помогут администрировать права доступа.

3.2. Схемы расположения сети как концептуальная модель сети. Location connection diagram (LCD)

Язык схем расположения представляет собой граф, вершинами которого являются места расположения агентов (рис. 1.11), а соединительные линии отражают наличие передачи информации между сетевыми вершинами.

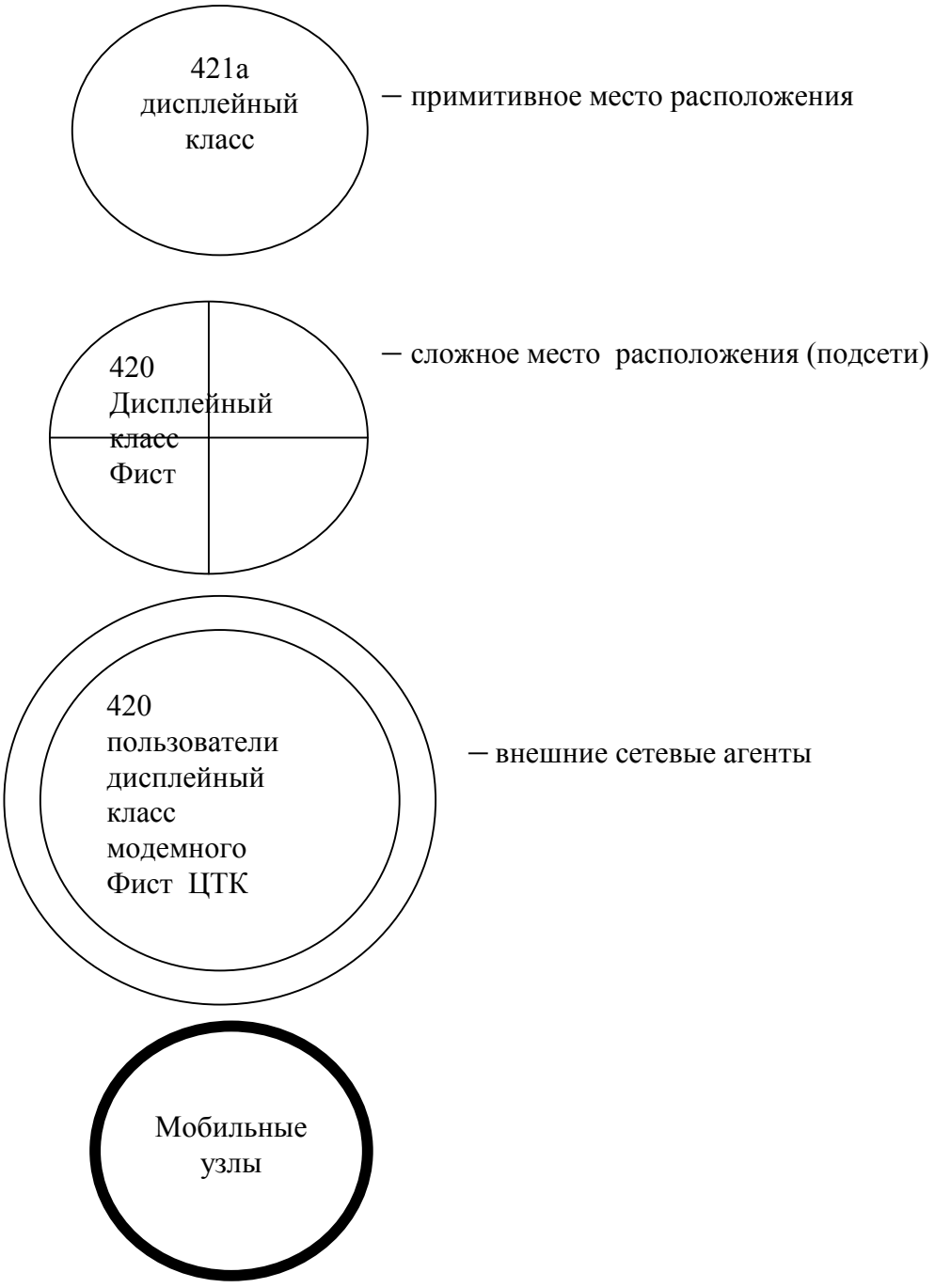


Рис. 1.11. Элементы схем расположения сети

Соединительные линии схем расположения обычно бывают помечены собственной длиной. При планировании распределенной обработки обычно места расположения связывают с локализацией бизнес-процессов, а для серверов указывают перечни глобальных ресурсов и схемы их видимости.

3.3. Пошаговое проектирование концептуальных моделей сетей

Концептуальная модель сети выполняется на стадии системного планирования и анализа. Пошаговая процедура формирования концептуальных моделей включает в себя следующие шаги:
Шаг 1 – идентификация мест расположения; в момент идентификации необходимо не просто выявить места расположения, но и распределить бизнес-процессы по данным местам, поэтому обычно сетевые диаграммы выполняют после моделирования процессов.
Шаг 2 – выполнение декомпозиции мест расположения, отдельные сетевые диаграммы должны составляться для одного уровня рассмотрения.
Шаг 3 – формирование схем расположения каждого уровня представления.
Шаг 4 – формирование комплекта схем расположения, которые детально отражают все фрагменты будущей сети.
Шаг 5 – отмечает сервера и группы однородных пользователей.
Формирование схем расположения сетей должно не просто опираться на ER-диаграммы и диаграммы процессов, но и должно быть согласовано с ними. Если планируется клиент-серверная обработка, то при формировании концептуальной модели сети появляется пятый шаг формирования.

4. Функциональное моделирование. Стандарты IDEF. IDEF 0

Для моделирования деловых процессов к настоящему времени предложены несколько различных стандартов и разнообразных методик, например, методология UML, IDEF, SADT и другие. Рассмотрим далее методологию функционального моделирования IDEF, учитывая ее пригодность для проектирования крупных экономических ИС с базами данных. В конце 70-х годов рядом крупнейших производителей аэрокосмической промышленности США была принята программа научных исследований ICAM (Integrated Computer Aided Management) и как результат в 1981 г. появился стандарт ICAM-definition (сокращенно IDEF). Он стал военным стандартом в США и назывался MIL-STD-1840 (1981 г.). В 1998 года при Государственном НИИ стандартов России этот международный стандарт был адаптирован. Сейчас стандарты группы IDEF являются

международными стандартами проектирования ИС. Подготавливается также национальный российский стандарт на базе международного стандврта. Стандарты IDEF включают в себя группы стандартов, разработанные к настоящему моменту:
IDEF 0 – функциональное моделирование информационных и бизнес-процессов;
IDEF 1 – диаграммы потоков данных (DFD-диаграммы);
IDEF 1x – ER-диаграммы;
IDEF 2 – динамические модели бизнес-процессов;
IDEF 3 – язык описания технологических процессов в производстве;
IDEF 4 – объектно-ориентированное описание бизнеса или системы;
IDEF 5 - онтологическое описание бизнес-системы и ИС.
IDEF 2 отражает временные характеристики бизнес-процессов, предполагает компьютерное моделирование и представляет собой сложный программный комплекс. На практике используется редко. Его развитие остановилось сразу же после создания.
Главную особенность стандарта IDEF 0 составляет функциональный блок IOCM (Input-Output-control-mechanism) (рис. 1.12), у которого кроме традиционного входа (I input) и выхода (O output) добавлены две входящие стрелки:
- управляющая стрелка (C control) описывает ограничения или правила выполнения функции;
- стрелка «механизм» (M mechanism) описывает средства выполнения функций.

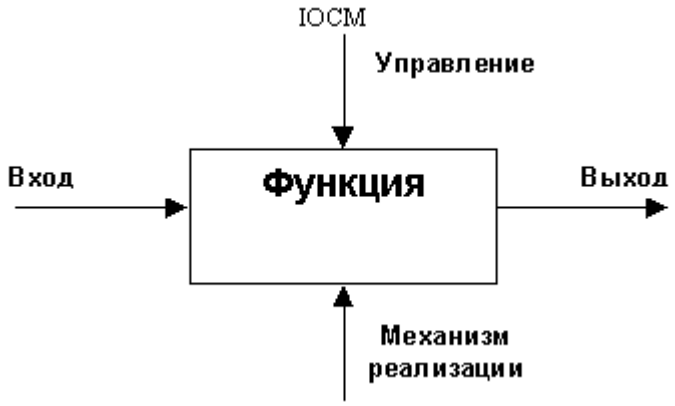


Рис. 1.12. Управляющий блок IDEF 0

Стрелки, связывающие функциональные блоки, выражают передачу объектов: материальных ресурсов, документов, правил и ограничений. Поэтому стрелки помечаются существительными.
Основными принципами IDEFO-моделирования является декомпозиция и формирование системы понятий проблемной области. Декомпозиция обеспечивается нумерацией диаграмм и блоков на диаграммах. Каждый блок в пределах диаграммы должен быть

пронумерован. Если какой-либо блок детализируется следующей диаграммой, то под ним записывается регистрационный номер диаграммы (рис. 1.13).

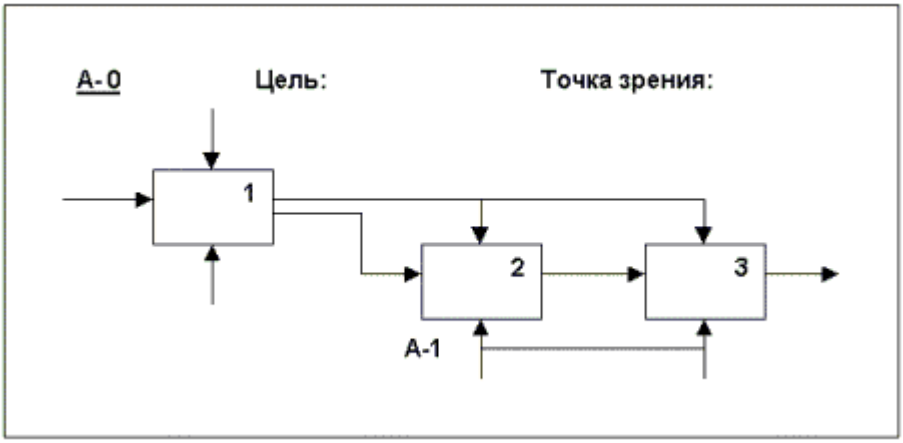


Рис. 1.13. Функциональная модель

Декомпозиция приводит к тому, что стрелки родительской диаграммы должны соответствовать стрелкам дочерней. На практике соблюдение такого соответствия может привести к излишней детализации или загромождению диаграмм. В стандарте IDEF 0 можно использовать специальные туннельные стрелки (рис. 1.14).

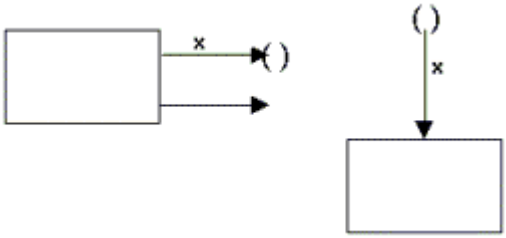


Рис. 1.14. Туннельные стрелки

Стандарт IDEF 0 связан с формированием глоссария проекта. Глоссарий представляет собой толковый словарь всех терминов проблемной области, которые встречаются в проекте в качестве названий функциональных блоков, целей и точек зрения.

Стандарт IDEF 0 разрабатывался с целью повысить взаимопонимание специалистов предприятий и специалистов по информационным технологиям. Для согласования общего проекта должны быть выполнены три этапа:

- интервьюирование специалистов;
- создание аналитики черновых IDEF0-диаграмм;
- согласование, изменение и утверждение диаграмм.

Для повышения читабельности IDEF0-диаграмм необходимо:

- на одной диаграмме располагать от 3 до 6 функциональных блоков;
- для каждого функционального блока расписывать не более 4 входных и выходных стрелок.

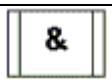
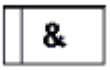
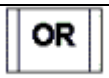
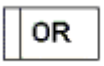
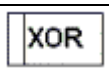
5. Стандарты проектирования IDEF 3 и IDEF 5

Стандарт IDEF 3 посвящен описанию технологических процессов предприятия. Любое производство сопровождается значительным документооборотом. Полная автоматизация предприятия возможна только на основе информатизации производства. Например, учет движения материалов для производственных предприятий не сводится к компьютеризации прихода и расхода на центральный склад, а учитывает материалы в цехах.
IDEF 3 включает в себя диаграммы двух видов:
1) диаграммы, отражающие последовательность этапов обработки (PFDD – Process Flow Description Diagram);
2) диаграммы изменения состояний (OSD - Object State Diagram).

5.1. PFDD-диаграммы

Элементы диаграммы связаны между собой стрелками. Слияние и разветвление стрелок осуществляется специальными переключателями.
Виды переключателей.

Таблица 1.1

Вид переключателя	Признак синхронности	Работа при слиянии	Работа при разветвлении
	Синхронный	Все входные работы должны быть завершены одновременно	Все выходные работы должны быть запущены одновременно
	Асинхронный	Все входные работы должны быть завершены	Все выходные работы должны быть запущены
	Синхронный	Одна или несколько входных работ должны быть завершены одновременно	Одна или несколько выходных работ должны быть запущены одновременно
	Асинхронный	Одна или несколько входных работ должны быть завершены	Одна или несколько выходных работ должны быть запущены
	Исключающее ИЛИ	Хотя бы одна работа должна быть завершена	Хотя бы одна работа должна быть запущена

Все переключатели нумеруются на каждой диаграмме (рис. 1.14), причем номеру предшествует буквенное обозначение, начинающееся с символа **J** (например, **J9**).

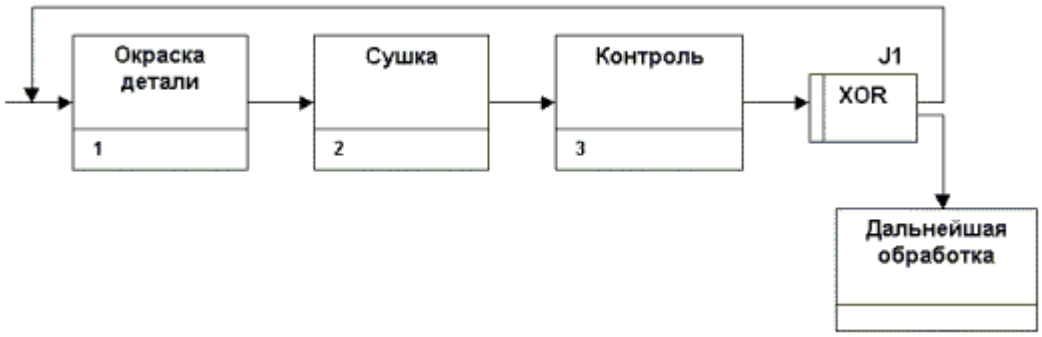


Рис. 1.14. IDEF3-диаграмма

5.2. OSD-диаграммы

Эти диаграммы отражают изменения свойств изделия, выполняемые в ходе обработки.

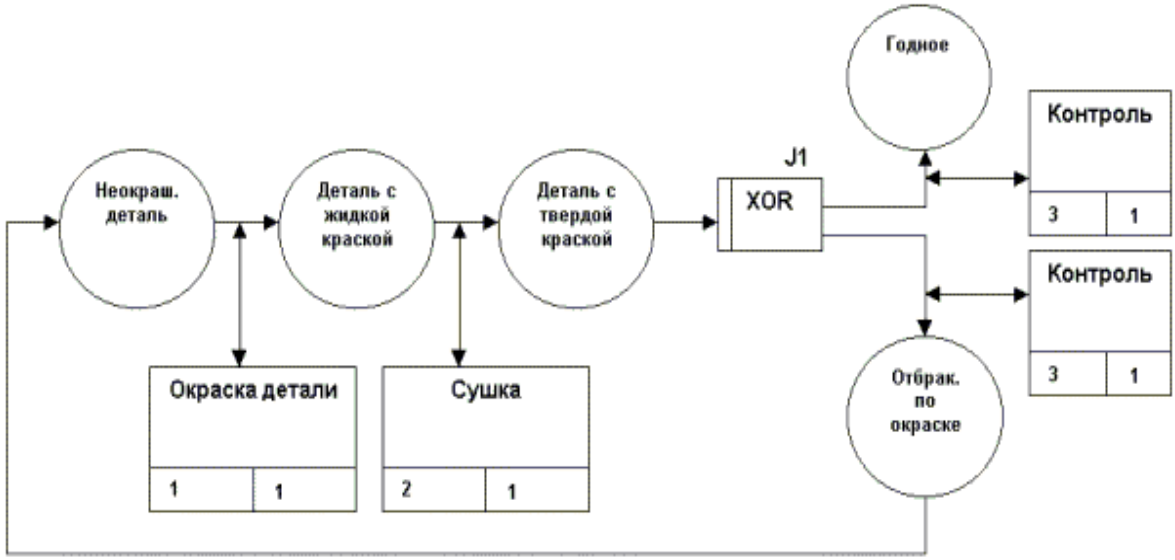


Рис.1.15. Диаграмма состояний объекта

Диаграммы состояния объектов (рис. 1.15) включают в себя как все элементы диаграмм последовательности этапов, так и дополнительный элемент, описывающий состояние объекта.

5.3. Стандарт IDEF 5

Стандарт IDEF 5 посвящен онтологическому анализу проблемной области. Основу онтологических описаний естественнонаучных дисциплин составляют классификации объектов с детальным описанием признаков, выявление законов, характеризующих поведение объектов. Примерами онтологии может являться:
- классификация растений;
- классификация многоугольников и других фигур.

Основу онтологического исследования составляют описания системы понятий заданной проблемной области. Для формирования онтологии необходимо перед разработкой графов классификации составить линейный список понятий или терминов проблемной области:
1) словарь терминов;
2) словарь дескрипторов понятий (описателей) – толковый словарь;
3) набор классификаций проблемной области.
При онтологическом исследовании выделяют два типа классификаций:
1) строгие, для которых любой вид обладает всеми свойствами класса и имеет хотя бы один дифференциальный признак; пример: классификация многоугольников (количество углов, признак правильности);
2) нестрогие (естественные) классификации – в них для каждого объекта характерен набор каких-либо отличительных признаков. При этом класс таких объектов не позволяет выделить один объединяющий признак.
Описание структур может выполняться с помощью композиционной схемы (см. рис. 1.17), связи в которой интерпретируются как «является частью». Для многих проблемных областей сложились свои языки описания структур: сборочный чертеж для механической единицы, структурная электротехническая схема и так далее

Рис. 1.17. Диаграмма структуры

Описание законов поведения выполняется процедурно. Например, для понятия равномерного движения закон выглядит так: s = V * t, а длина отрезка, заданного координатами границ равна
$d=SQRT[(x_2-x_1)^2+(y_2-y_1)^2]$.
Эти законы не всегда имеют аналитическую форму. Многие объекты могут описываться утверждениями, в том числе на естественном языке.

6. Автоматизация разработки баз данных информационных систем. Понятие CASE-средств. Классификация CASE-средств (Computer Aided Software Engineering)

CASE-средствами называют совокупность информационных средств и технологий, направленных на повышение эффективности труда разработчиков ИС, следовательно, CASE-средства могут быть предназначены для любой фазы жизненного цикла ИС. Для современных корпораций, работающих в области высоких технологий, характерно интегрированное использование систем САПР, АСУ и АСПР. Первыми видами CASE-средств явились компиляторы, линкеры, библиотеки подпрограмм. Значительным результатом в развитии CASE-средств явились Turbo-среды, представляющие собой автоматизированные рабочие места программиста, которые кроме компиляторов включали в себя отладчики, трассировщики, библиотеки подпрограмм, объединенные общим интерфейсом. Значительное количество CASE-средств было создано на базе различных методологий системного анализа. В настоящее время распространяются CASE-средства для полного цикла программного обеспечения.

В основу классификации CASE-средств современные иследователи кладут фазы жизненного цикла программного обеспечения и выделяют CASE-средства верхнего уровня и CASE-средства нижнего уровня. Первые автоматизируют системное планирование и системный анализ. Вторые автоматизируют проектирование, разработку и сопровождение.

CASE-средства для системного планирования представляет собой либо простые специализированные графические редакторы для создания диаграмм бизнеса и матриц соответствия бизнес-процессов конкретным подразделениям корпорации, либо сложные имитационные модели некоторых классов бизнеса.

CASE-средства для системного анализа работают в основном как специализированные диаграммеры и средства моделирования:

- ER-диаграмм;
- диаграмм потоков данных;
- диаграмм процессов.

Самые мощные из CASE-средств данного уровня позволяют генерировать третью нормальную форму реляционных таблиц по ER-диаграммам.

CASE-средства для проектировщика существенно зависят от выбранной методологии проектирования. Сущность таких CASE-средств представляется репозиторием проекта. Репозиторием проекта

называется специальная база данных, хранящая все материалы данного проекта.

К CASE-средствам разработчика можно отнести:

- автоматизированное рабочее место программиста;
- генераторы компонент (отчетов и экранов);
- генераторы кода.

Современный уровень CASE-средств характеризуется переходом от Turbo-среды к программным продуктам класса Workbench и Developers Kits. CASE-средства сопровождения обычно включают в себя генераторы новых отчетов и форм. Для некоторых развитых БД (Oracle, Informix, DB2) созданы измерители качества, обычно построенные по принципу регистрации успешности транзакции.

Некоторые работы характерны для всех этапов жизненного цикла, например, разработка документации, принятие технических или финансовых решений, такие CASE-средства называют средствами автоматизации управления проектами (cross-life-tool).

Основу архитектуры полного цикла CASE-средств составляет репозиторий проекта. Иногда в качестве синонимов употребляются термины: словарь, энциклопедия.

Архитектура CASE-средств прошла три стадии развития:

1) использование локальных репозиториев, т. е. предназначенных для одного разработчика;
2) использование файл-серверных репозиториев, обслуживающие группы разработчиков;
3) клиент-серверная архитектура репозиториев, обеспечивающая кроме локальных репозиториев центральный репозиторий, который обычно хранит несколько проектов.

Выделяют i-CASE-средства – интегрированные средства, m-CASE-средства – модульные средства, которые легко экспортируют или импортируют данные в CASE-средства. CASE-средства разных производителей могут быть ориентированы либо на концепцию открытых систем (m-CASE – модульные), способных работать с CASE-средствами разных производителей через импорт и экспорт. Многие предприятия предпочли ориентироваться на внутренний стандарт и распространяют интегрированные средства (i-CASE), которые полностью покрывают жизненный цикл системы и не связаны с другими CASE-средствами. Для крупных софтверных предприятий характерно наличие специального подразделения CASE-средств. В завершение пункта описания CASE-средств приведем некоторые распространенные в настоящее время CASE-средства: BPWin (PLATINUM Technology), Silverrun (Silverrun Technology), Oracle Designer (Oracle), Rational Rose (Rational Software), Paradigm Plus

(PLATINUM Technology), Power Designer (Sybase), System Architect (Popkin Software).

7. Репозиторий проекта базы данных информационной системы

Репозиторий — служебная база данных, предназначенная для хранения проектной и эксплуатационной информации. Репозитории были развиты из словарей баз данных. Изначально репозитории были специализированы и разработаны фирмами — поставщиками инструментальных средств. В результате в одной организации могут иметься репозитории от различных поставщиков, причем каждый из них был специализирован на какой-то вид проектной документации.

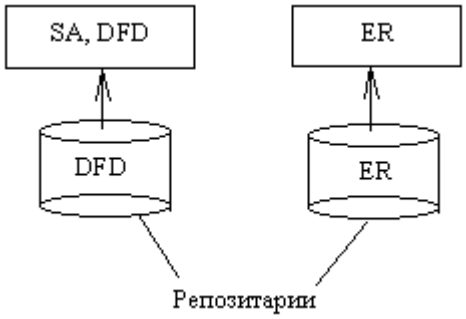


Рис. 1.18. Схема использования репозиториев разных поставщиков

В настоящее время завершается переход от периода, когда используются репозитории различных поставщиков, к периоду, когда будут использоваться репозитории одного поставщика (рис. 2.6, 2.7).

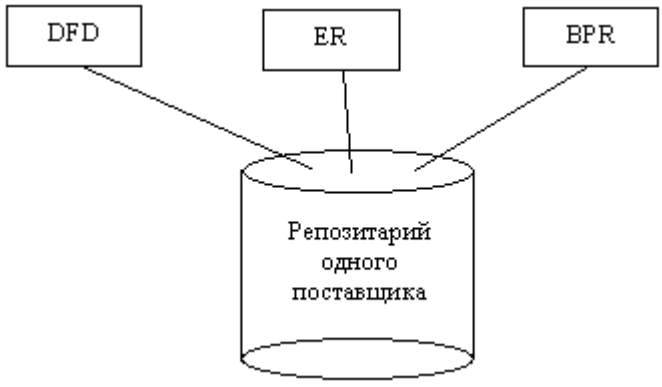


Рис. 1.19. Схема использования репозитария одного поставщика

В настоящее время в международной организации по стандартизации ISO начаты работы по стандартизации репозиториев. Созданный комитет рассмотрел пять проектов стандартов. Самый слабый из проектов стандарта перечисляет типы диаграмм и языков описаний, которые должны храниться в репозитории, причем для каждого вида информации определяется гранулярность, то есть

точность дискретного описания объектов. Например, может быть гранулярность на уровне наименований схем или на уровне сущностей диаграмм, или гранулярность на уровне атрибутов сущностей.

Самый строгий из предложенных проектов стандарта требует синхронного изменения всех описаний объекта в случае модификации самого объекта. Данный проект предусматривает хранение в репозитории фрагментов исходного кода, например иерархии классов на языке C++ или описание баз данных. Описанный в стандарте механизм уведомления об изменениях в настоящее время разрабатывается и не встроен ни в один из коммерческих продуктов.

Организация репозиториев проектной информации для информационных систем позволяет минимизировать издержки на модернизацию самих информационных систем и выделить следующие виды модернизации.

- Переход между версиями одного и того же проекта. Основные трудности вызывает перевод триггеров и хранимых процедур.
- Переход от одного программного продукта к другому при сохранении одной и той же модели данных. Например, организация переходит от информационной системы, написанной на FoxPro для файл-серверной технологии к серверу данных ORACLE с использованием клиент-серверной технологии. Сохраняется только тип модели - реляционная, хотя под FoxPro реализована псевдореляционная модель, а под ORACLE — реляционная модель в 3-ей нормальной форме. Репозитарий объектов сохраняет описания структур всех таблиц, чем позволяет воспроизвести логическую схему в новой среде.
- Переход от одной модели данных к другой. Например, от реляционной к объектно-ориентированной. Здесь репозиторий будет содержать описание проблемной области, по которой можно составить описание классов.

С точки зрения физической организации репозитории представляют собой гипермедийные базы данных. В настоящее время их роль часто играет документация.

Глава 2. Основные тенденции развития технологии баз данных

1. Перспективы технологии баз данных. Основные тенденции развития баз данных. Проблемы современных баз данных

Современные технические средства автоматизации экономической деятельности представлены локальными сетями — Local Area Network (LAN) и глобальными — Global Area Network (GAN). Такие технические средства предоставляют возможность распределения информации. Распределение может быть вызвано как необходимостью хранить большие объемы данных, так и необходимостью ускорить обработку удаленных запросов. В настоящее время не существует коммерческих систем управления распределенной информацией, решающих задачу в полном объеме.

В большинстве организаций сложившиеся информационные системы являются неоднородными, например в одной организации могут использоваться: персональные базы данных на основе dbase 4, на основе Visual Fox Pro, реляционные базы данных Ingress для Unix и так далее. В современных базах данных делаются попытки создать инструмент для корректной работы в неоднородной среде, такие технологии называют технологиями мультибаз данных, которые на основании глобальной схемы данных должны допускать выполнение запроса к данным независимо от их формата.

Большая часть деловой информации в организациях хранится не в базах данных, а в текстовых, так называемых плоских файлах. В настоящее время поставлена задача включить в мультибазы данных и менеджеры файловых систем.

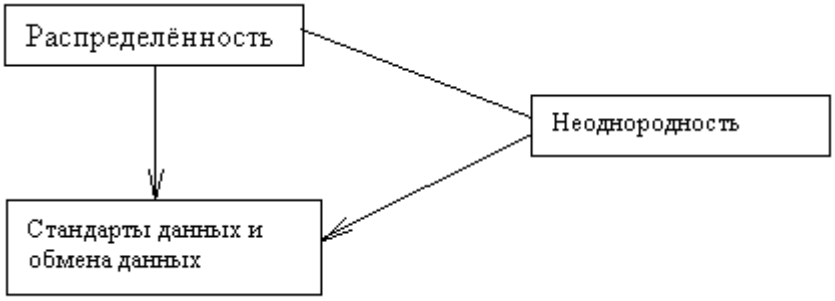


Рис. 2.1. Факторы распределенности

Одним из решений проблемы неоднородности и распределенности (рис. 2.1) данных является появление стандартов.

Первым стандартом в области баз данных был стандарт сетевой модели баз данных Conference On Data System Language (CODASYL).

Стандарт CODASYL впервые ввел понятие о языке определения данных, схеме данных и о языке манипулирования данными. Развитие реляционной модели данных привело к появлению серии стандартов SQL — 72, 76, 92. В настоящее время доминируют реляционные базы данных, но большинство реальных проблемных областей не может быть представлено в виде совокупности рабочих таблиц. Решение этой проблемы ищут на пути создания объектно-ориентированной модели данных.

Развитие объектно-ориентированных СУБД проводиться в двух направлениях.

— Дополнение к существующей реляционной модели данных объектно-ориентированной надстройки. Среда хранения по-прежнему является реляционными таблицами, а схема данных представляет собой иерархию наследуемых типов, то есть становится объектно-ориентированной. Сторонники этого направления сформулировали свои предложения в манифесте систем баз данных 3-го поколения.
— Разработка особой объектно-ориентированной модели данных. Такая модель данных должна быть создана на основе объектно-ориентированных языков программирования, в состав которых включена возможность определять и манипулировать с внешними данными. Сторонники этого направления выпустили манифест объектно-ориентированных баз данных.

2. Сравнение реляционных и объектно-ориентированных баз данных

2.1. Способы реализации объектно-ориентированных баз данных (ООБД)

Объектно-ориентированные базы данных должны сохранять все возможности объектно-ориентированного языка программирования и обеспечивать создание и долговременное хранение объектов во внешней памяти, то есть включающим языком ООБД могут служить C++, Smalltalk, Java, в которых обеспечено долговременное хранение этих объектов. Примерами объекто-ориентированных баз данных могут служить Jasmine (Computer Associates), JYD Object Database (JYD Software Engineering), TITANIUM (Micro Base Systems, Inc.) и

другие. Например, экземпляр класса «точка» Point может быть описан следующим образом:

Point (x,y);

Point *p = new [Point(5,5)];

Point *p3 = new Persistent [Point(7,5)].

Объектно-ориентированные СУБД должны обеспечивать долговременное хранение объектов, адекватное внутреннее представление объектов, относящихся к различным типам данных, параллелизм в обработке различных типов объектов, выполнение всех методов, наследуемых объектом. Такие базы данных называют объектно-ориентированными в полном смысле объектной парадигмы. Рынок ООБД достигает 25 — 30 млн. долларов. Основную долю рынка баз данных занимают классические реляционные базы данных и гибридные — объекто-реляционные базы данных (ORACLE 8). Существуют следующие способы реализации баз данных:

1) реляционные базы данных (РБД), 2) гибридные базы данных (ГБД), 3) расширенные базы данных, 4) ООБД.

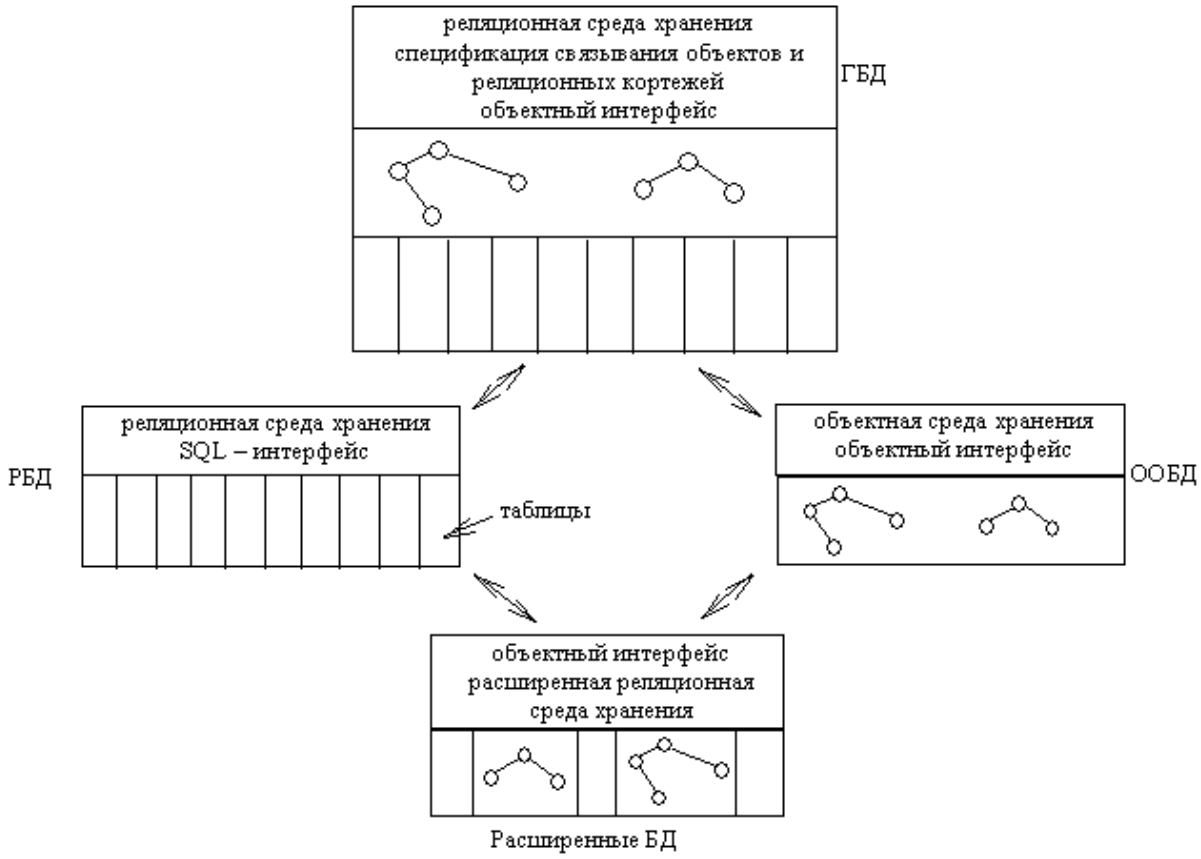


Рис. 2.2. Интеграция реляционных и объектно-ориентированных СУБД

Классификация баз данных (рис. 2.2.) построена по двум критериям:

1) организация среды хранения;
2) механизм обработки запросов.

По первому критерию БД делят на использующие реляционные таблицы или использующие расширенные реляционные таблицы, которые включают специальные средства поддержки объектов и абстрактных типов данных.

По второму критерию механизм обработки запросов разделяют на реляционный и объектный механизмы.

2.2. Манифест баз данных третьего поколения

В 1990 г. группой специалистов по базам данных был опубликован манифест баз данных третьего поколения. Он включал в себя три принципа и несколько предложений.

1) Базы данных третьего поколения должны включать в себя базы данных второго поколения, то есть реляционные базы данных. Тем самым были узаконены гибридные и расширенные базы данных.
2) Базы данных третьего поколения должны быть СУБД, а не языком программирования, то есть должны быть обеспечены три сервиса: 1) управление данными, 2) управление объектами, 3) управление знаниями.
3) Базы данных третьего поколения должны быть многоязычными, причем включенные языки должны относиться к C++ или JAVA.

Для реализации управления объектами было высказано 5 предложений.

1) Базы данных третьего поколения должны обеспечить иерархию классов. Пользователь свободно определяет собственные абстрактные типы данных.
2) По иерархии классов должно обеспечиваться корректное наследование.
3) Реализация и наследование методов должны выполнять требование инкапсуляции.
4) Каждый объект должен иметь индивидуальный идентификатор.
5) Управление знаниями должно включать в себя реализацию триггеров и хранимых процедур.

Некоторые предложения обеспечивают сохранение статуса баз данных 3-го поколения как СУБД. Навигация по базе данных должна осуществляться независимо от физической реализации среды хранения, то есть сохраняется возможность реляционной БД обеспечить поиск объектов не по адресам и ссылкам, а по содержанию данных.

Реализация индексов, кластеров, схем баз данных относится к физическому уровню организации и должна быть доступна пользователю.

Базы данных третьего поколения должны сохранять параллелизм обработки запросов.

Некоторые предложения обеспечивали открытость баз данных третьего поколения:

1) SQL должен сохранить свои позиции как язык запросов к базам данных;
2) SQL должен стать языком нижнего уровня для клиент-серверных систем.

2.3. Манифест объектно-ориентированных баз данных

Манифест (1990 г.) сформулировал свойства ООБД. Все свойства разбиты на 3 группы: 1) обязательные, 2) факультативные (необязательные), 3) обеспечивающие открытость. Манифест ООБД в качестве обязательных свойств назвал: 1) иерархию типов и наследование, 2) инкапсуляцию, 3) перегрузку методов. Наряду с требованием индивидуальных идентификаторов появилось требование поддержки сложных объектов. Отдельно подчеркивается необходимость:

— позднего связывания, то есть связывания имени метода и кода не на этапе трансляции, а на этапе исполнения;
— расширяемости типов, то есть предполагается равенство как встроенных, так и определённых пользователем типов данных;
— полной вычислимости, то есть предполагается реализация в языке манипулирования данными вычисления функций, реализуемых в обычных языках программирования.

К необязательным свойствам отнесены: 1) множественное наследование, 2) поддержка версий объектов.

Свойство открытости должно обеспечить связывание с C++, Smalltalk, JAVA.

2.4. Способы реализации объектно-ориентированных баз данных (ООБД). Объектно-ориентированные языки БД и SQL

В настоящее время язык SQL имеет несколько версий, связанных с появлением стандартов: SQL — 86, 89, 92, а также практически каждый поставщик серверов данных поставляет своё подмножество языка. Перспективой языка SQL является появление объектно-ориентированного стандарта SQL-3, который предназначен для расширенных или гибридных моделей данных. В стандарте SQL-92 имеется 149 элементов языка, которые должны быть определены и документированы разработчиком (implementation definition) и 75

элементов, зависимых от реализации (implementation dependent). Примером изменений сделанных поставщиком может быть длина разрядной сетки, диапазон представления чисел.

Разработчики серверов данных часто включают в состав языка SQL конструкции, которые не вошли в стандарт, но которые пользуются коммерческим спросом. Наиболее сильным примером являются триггеры (процедуры, хранимые на сервере и активизирующиеся по наступлению некоторого события). Например, синтаксис оператора создания триггера может быть следующим.

create trigger <имя_триггера>
<время срабатывания>
on <событие срабатывания>
<содержание триггера>

Триггер, например, дополняющий справочник товаров, может иметь нижеприведенную структуру.

create trigger Контроль
before
on insert Товары
insert into sys_j (…).

В настоящее время стандарт SQL-92 является доминирующим, но этот стандарт поделен на три уровня: базовый, промежуточный, полный.

Промежуточный уровень SQL-92 отличается от базового наличием доменов и многих вспомогательных функций и предикатов. Полный SQL дополнительно включает операторы drop, alter, grant. Большинство серверов данных, которые соответствуют стандарту SQL-92 реализуют на практике только его базовый или промежуточный уровень.

Под SQL-3 понимают инициативу, получившую название MOOSE — Major Object Oriented SQL Extension (основное объектно-ориентированное расширение SQL (рис. 2.3.)). В рамках инициативы SQL-3 должны быть разработаны 3 новых языка:

— ODL (Object Definition language) — язык определения объектов (ранее – DDL, оператор create);
— OQL (Object Query language) — язык запросов к объектам, (ранее - DQL оператор select);
— OML (Object manipulation language) — язык манипулирования объектами (ранее — DML (data manipulation language).

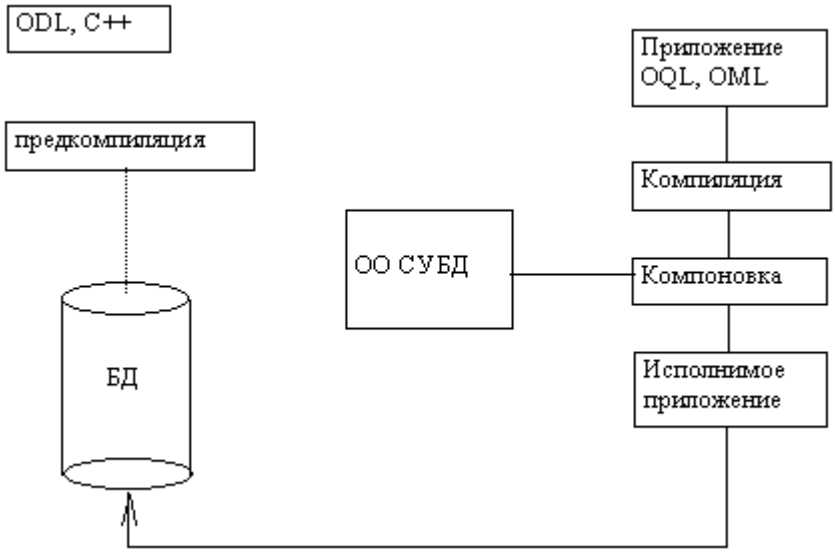


Рис. 2.3. Архитектура ООБД

2.5. Язык объектно-ориентированных баз данных

ODMG — Object Distribution Managing Group — группа управления распределенными объектами. Эта группа создаёт ODL, OQL, OML. Это ассоциация производителей программного обеспечения и специалистов, которая включает на настоящий момент около 800 участников. Группой предложены варианты языков определения объектов, запросов к объектам и манипулирования объектами.

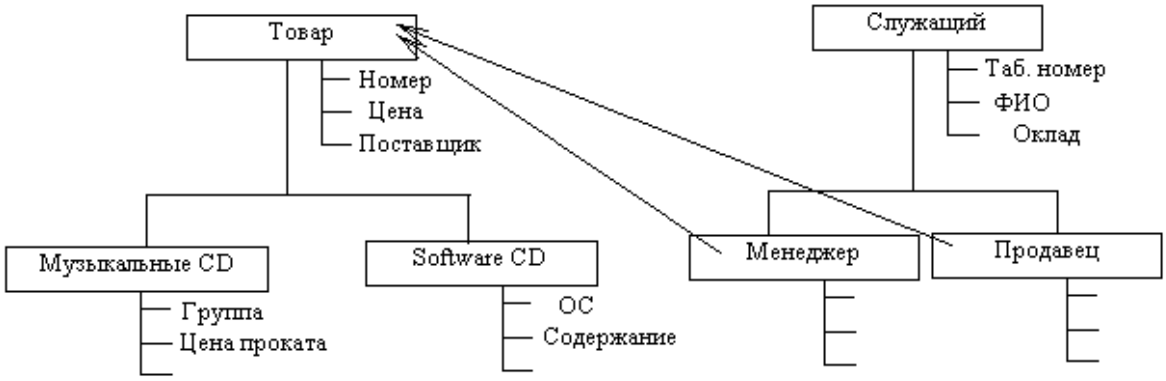


Рис. 2.4. Пример расширенного описания

На рис. 2.4. описаны классы и объекты системы «Учет товаров розничной торговли». Объектной базой данных называется совокупность экземпляров объектов, соответствующих определённому классу. Такую совокупность называют экстентом.

```
interface Товар {
attribute Номер short int;
attribute Цена float;
::::
```

```
relation set Менеджер заказываются inverse заказывают;
::::
};
interface Менеджер : Служащий
{ attribute
::::
relation set Продавец управляет inverse управляется;
::::
void Заказ (::);
}.
```

Основная конструкция языка определения объектов interface подобна оператору class в языке программирования C++. Особенностью является наличие предложения, описывающего отношения между объектами — предложение relation, с его помощью выражаются те связи между объектами, которые не наследуются по иерархии классов. В реляционной модели данных отношения реализуются через внешний и первичный ключи.

Кроме идеи самостоятельного языка определения объектов предложено использовать объектно-ориентированные языки: C++, JAVA, Ada, Smalltalk. Для всех перечисленных языков предложены спецификации связывания.

В качестве языка запросов к объектам сохраняется основной оператор select, но выполнение запросов требует наследования по иерархии классов и поиска в одном или нескольких экстентах. В настоящее время отработаны конструкции group by, having. Язык манипулирования объектами сводиться к выполнению методов, описанных в классах. Обычно заимствуется синтаксис вызовов методов из C++.

3. Содержание и перспективы языка БД семейства x-base

3.1. История языков семейства x-base

Языками семейства x-base называют различные языки, основанные на исходном языке dbase I, II, III, III plus, IV. К данному семейству относят языки Clipper, FoxPro и некоторые другие. Особенностью языка x-base является ориентация на персональные компьютеры. Языки называют псевдореляционными, так как они не поддерживают полную модель Кодда и нормальную форму реляционных таблиц. Второй особенностью языков является смешение 3-х составных частей: 1) операторы управления данными, 2) операторы

управления ходом вычислений (if, while и другие), 3) языки управления интерфейсом.

В настоящее время на рынке баз данных для персональных компьютеров работают следующие фирмы и продукты: Borland — Paradox, dBase (приобретён вместе с фирмой Ashton-Tate); Computers Associate (CA) — CA-Clipper, CA — db fast; Microsoft — FoxPro (приобретён с фирмой FoxSoftware), Access; IBM — DB2/2 для OS/2.

3.2. Стандартизация семейства x-base. Перспективы языков x-base

В 80-х гг. Международной организацией по стандартам (ISO) была предпринята попытка стандартизации языков семейства x-base. Был создан комитет X3J19, который объединил работы по созданию стандартного подмножества языка, но фирма Ashton-Tate заявила о правах собственности на язык. В результате стандарт не был создан. Языки xbase представляют собой родственные, но разнообразные языки. Перевод из среды в среду производят только вручную.

Хотя не удалось создать стандартный язык xbase и код приложений оказался не переносимым между системами, формат dbf-файлов обеспечивал в первое время перенос данных между системами. Преимуществами dbf-формата являются: 1) простота хранения данных, 2) открытость структуры управляющего блока. К недостаткам формата относят следующие особенности среды хранения:

1) dbf-файл не выделяет первичный ключ записи, поэтому хранение нормализованных таблиц невозможно;
2) неудачно организовано хранение текстовых данных (memo — поля);
3) не предусматривалось сжатие хранимых текстовых данных.

В настоящее время многие производители отказались от dbf-формата, либо используют его развитую форму. Поэтому для конвертирования данных требуются специальные средства импорта и экспорта.

Язык семейства x-base имеет потенциал для развития. Примером успешного преодоления проблемы является реализация версий продуктов, позволяющих реализовать Windows — подобные интерфейсы. Все производители языков x-base в настоящее время обладают средствами позволяющими организовать GUI. Хотя существуют прогнозы отмирания семейства x-base, большая накопленная масса приложений позволяет существовать данному семейству значительное время.

4. Активные базы данных

Идея активных баз данных родилась вследствие развития концепции серверов данных. Сервер данных для успешной работы должен выполнять работы по самоорганизации, например, отслеживать появление каких-либо событий (мониторинг), обнаруживать противоречивые данные, отслеживать ограничения целостности. Совокупность свойств сервера данных, обеспечивающих самоорганизацию, называется активной базой данных.

Для создания средств активизации баз данных были предложены специальные инструменты. Их можно сгруппировать в 3 группы: 1) мониторы, 2) триггеры, 3) хранимые процедуры. Мониторами называют системные программы управления данными, обнаруживающие появление фиксированных событий, например нарушение прав доступа, появление запроса к определённым таблицам, нарушение фиксированных ограничений. Триггерами называют реакции сервера данных на определенные события, то есть любой триггер связан с каким-то набором монитором. Процедура называется хранимой, если она храниться на сервере и запускается в приложении.

Логика триггеров и хранимых процедур отслеживает ограничения целостности. В общем случае ограничения целостности представляют собой описание семантики или смысла проблемной области, поэтому активные базы данных, столкнувшись со смысловыми описаниями, стали развиваться как интеллектуальные базы данных. Семантическое описание экономической проблемной области выполняется в виде бизнес-правил. Логически каждое бизнес-правило представляет собой систему продукций или конструкций: ЕСЛИ <условие> ТО <реакция>, следовательно, ограничения целостности в активных базах данных представляют собой базы знаний. Любой запрос к базе данных вызывает изменение текущего состояния данных. Перед выполнением такого изменения необходимо выполнить проверку на противоречие запроса и бизнес-правила. Проверка представляет собой логический вывод. Логический вывод занимает существенное время, поэтому база бизнес-правил обычно содержит небольшое количество критических правил.

Старшие версии серверов данных, такие как ORACLE 8, Informix, Interbase включают средства активных баз данных.

В настоящее время активные базы данных редко организуются даже в больших корпорациях из-за сложностей создания семантических моделей и неготовности пользователей и разработчиков.

5. Временные (темпоральные) базы данных

Традиционные информационные системы, построенные на реляционных базах данных, не могут выполнять запросы, связанные со временем, так как базы данных предназначены для оперативного хранения текущей информации. Данные о прошлых состояниях объекта в лучшем случае хранятся в архиве и недоступны непосредственно для запросов. В настоящее время для обслуживания запросов создают специальные таблицы. Например, для обслуживания запроса об изменении цен на какой-либо товар необходимо хранить базу данных:

Таблица 4.1.

Таблица обслуживания запроса об изменении цен на интервале времени

© товара	Цена	Дата

Такие таблицы можно создать только для заранее известных запросов, которые называют регламентированными. Современным решением проблемы временных запросов и являются темпоральные базы данных. Такие базы данных обычно предназначены для аналитических служб и систем OLAP (on-line Analytical Processing) и систем DMS (Decision Making Systems) — систем принятия решений.

Темпоральная модель данных подразумевает введение дополнительной координаты времени для каждой реляционной таблицы.

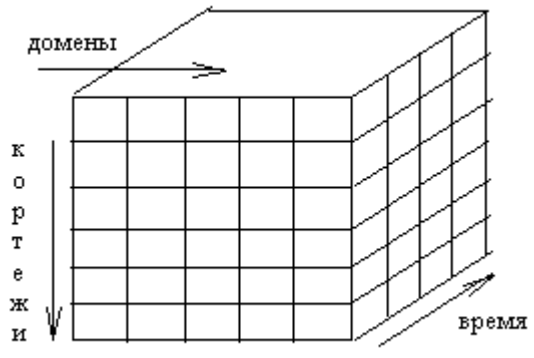


Рис. 2.5. Темпоральная модель данных

Временная модель данных (рис. 2.5.) снимает с программиста обязанности по изучению регламентированных запросов и организации специальных таблиц. Каждая база данных связана с

определенной периодичностью хранения, то есть гранулярностью (дискретность структуры объекта — минута, час, день и т.д.). Оператор создания временной таблицы должен содержать предложения о гранулярности таблицы.

create table <имя таблицы>
with granulation date;

Темпоральные модели баз данных связаны с двумя типами времени.

1) Эффективное время — время наступления события.
2) Текщее ввремя - время регистрации события в информационной системе.

К настоящему моменту создана расширенная реляционная модель, которая называется темпоральной реляционной моделью. Для нее определено понятие нормальной формы (TNF) и созданы корректные алгоритмы выполнения базовых алгебраических операций, т. е. TSQL, TempoSQL, HSQL (History SQL).

В настоящее время разрабатываются модели оператора select, учитывающие временную модель данных.

select
::::
where
when <условие с использованием времени>.

Модификация оператора запроса select для временных баз данных требует включения специального предложения when с условием, использующим время. Для формулировок условия необходимо ввести специальные временные предикаты — до, после, предшествует, следует за и другие.

6. Пространственные базы данных

Пространственные базы данных строятся как геоинформационные системы (ГИС). Примерами таких систем могут служить: карта города с возможным построением маршрутов движения и получения информации об учреждениях и населении; карта автодорог региона или страны, карты коммуникаций, теплосетей, сетей связи, электронных сетей. Особенностью пространственных баз данных является необходимость двух возможностей, не предусмотренных реляционной технологией: 1) средств хранения изображений, 2) средств навигации. Под навигацией понимается построение маршрутов в геоинформационной системе.

Современные базы данных позволяют хранить двоичные объекты (BLOB — Binary Large Object), но не включают специальных средств их обработки. Хотя реляционные базы данных не позволяют хранить сеть и автоматически получать маршрут на этой сети, но современные сервера данных имеют спецификации связывания с такими языками как C++, JAVA, на которых можно запрограммировать данные задачи.

С конца 90-х гг. кроме хранения статических изображений базы данных должны хранить движущиеся изображения (видеоданные) и звуковые данные. Появление стандартных форматов для хранения аудио и видеоданных, или появление стандартов де-факто, позволило включать в состав базовых операций серверов данных проигрыватели аудио и видеоданных. Если формат разрешает доступ к отдельным компонентам таких данных, то базовые операции могут быть более широкими, например, проиграть какую-либо часть, начиная с какой-либо части.

При включении таких средств естественно встраиваются в базы данных типы audio, movie. Базы данных, обеспечивающие хранение и обработку мультимедийной информации называют гипермедийными системами. Для гипермедийных систем традиционным интерфейсом является гипертекст, представляющий собой текст с ассоциированными ссылками.

Гипертекст в своей основе имеет «нелинейный учебник», то есть книгу, которую можно читать по произвольно построенному на основе ссылок пути. Идея была разработана Бушем в 1932 — 38 годах и нашла первое компьютерное воплощение гораздо позже в системе HyperCard фирмы Apple на компьютерах Macintosh.

Реализация гипертекстового интерфейса требует от баз данных хранения текстов, которые тоже являются неструктурированной информацией. Для организации сложного гипертекстового интерфейса система баз данных должна не только хранить полные тексты, но и включать в себя поисковую машину, осуществляющую индексирование каждого вновь появившегося текста в соответствии с заранее составленными рубриками.

7. Распределённые базы данных

Под распределённой базой данных понимают базу данных, объекты которой распределены между узлами вычислительной сети. Распределение возникает в том случае, если реляционная таблица распределяется по кортежам или по столбцам. Например, база данных «сеть магазинов» может быть распределена по географическому

принципу. Способ распределения данных по кортежам или столбцам называется фрагментацией. Тиражирование копий реляционных таблиц по нескольким узлам сети называют репликацией. Для управления транзакциями в условиях распределённости по фрагментам необходима глобальная схема базы данных. Глобальная схема позволяет ответить на вопрос о местоположении любого из фрагментов распределенной базы данных.

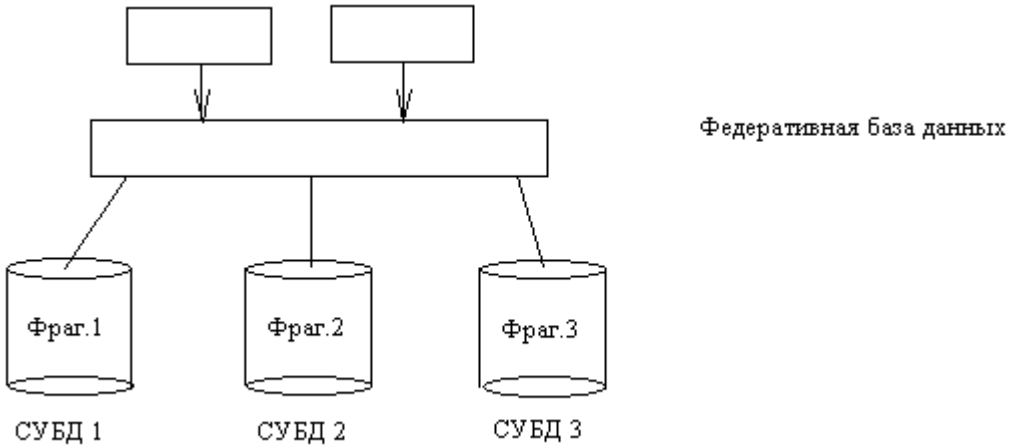


Рис. 2.6. Федеративная база данных

Распределённые СУБД называют федеративными. Федеративные базы данных (рис. 2.6.) существуют как коммерческие продукты для того случая, когда каждый фрагмент базы данных находится под управлением одной и той же СУБД, т. е. существуют однородные федеративные базы данных. Если распределенные объекты хранятся под различными СУБД, то создать единую систему на основе глобальной схемы затруднительно, поэтому для неоднородных сред используются объединяющие средства промежуточного слоя middleware (медиатор).

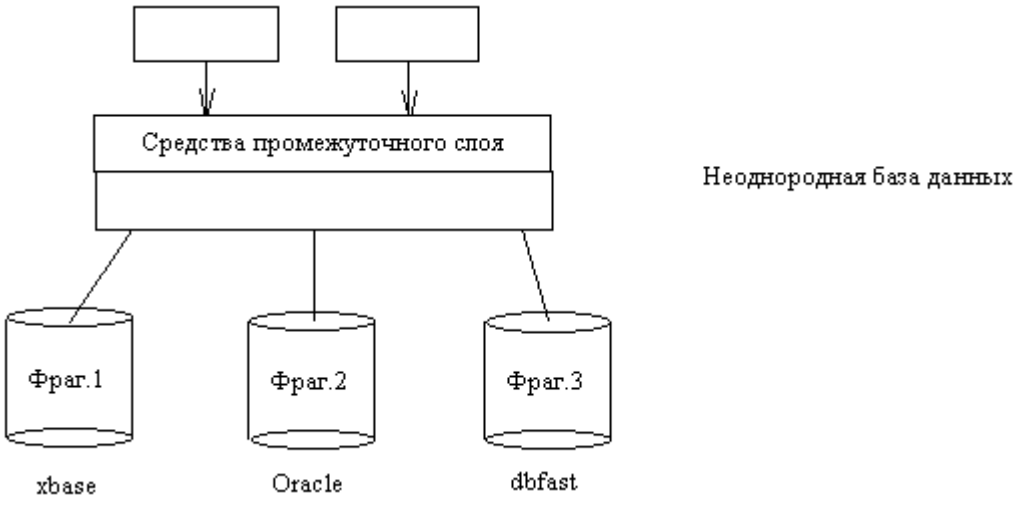


Рис. 2. 7. Средства промежуточного слоя

Средства промежуточного слоя (рис. 2.7.) благодаря конверторам и трансляторам способны выполнить операции, выраженные на языке одной системы в среде другой системы, то есть они обладают свойством интероперабельности. Примерами являются: ODBC (Open Data Base Connecting), работающее на множестве баз данных принадлежащих Microsoft; IDAPI (Integrated DataBase Application Programming Interface) фирмы Sun.

8. Некоторые проблемы современной технологии баз данных

Развитие технологии баз данных связано не только с использованием вычислительных сетей и нарастающим разнообразием данных, но и с внутренними проблемами развития технологии баз данных. В настоящее время можно выделить следующие проблемы.

1) Проблема эталонного тестирования серверов данных. Если для реляционных баз данных существует международный совет по обработке транзакций (TPC), который предложил 4 группы тестов. Известный тест TPC-D воспроизводит для сервера данных банковский операционный зал. Для объектно-ориентированных баз данных такие тесты только начали разрабатываться, причем в отличие от реляционных моделей примеры эксплуатационной среды берутся не из области экономики, а из области автоматизированного проектирования изделий, или управления техническими объектами.

2) Проблема третичной памяти. Под третичной памятью понимают организацию архивов на различных носителях. В настоящее время архивирование не подчиняется никаким стандартам, а осуществляется на каждом предприятии по-своему. В большинстве случаев архивы не всегда восстанавливаемы или требуют для восстановления значительного времени.

3) Проблема формирования оперативных баз данных. В связи с постоянным ростом объема оперативной памяти у компьютеров предложено располагать базы данных в оперативной памяти. Для таких баз данных необходимо разработать новое понимание среды хранения и выполнения транзакций.

4) С распространением мобильных компьютеров, включенных в общую корпоративную сеть средствами связи, появилась проблема хранения фрагментов распределенных баз данных на мобильных компьютерах.

5) Проблема сжатия данных большого объема. Несмотря на постоянный рост и удешевление дисковых накопителей, существует

потребность в сжатии данных. Основные проблемы сжатия данных носят системный характер. Например, возникают следующие вопросы: каким должен быть индекс, если база данных храниться в сжатом виде, можно ли сам индекс хранить в сжатом виде.

6) Проблема проектирования баз данных. В настоящее время созданы средства визуального проектирования баз данных, в том числе и средства для их объектно-ориентированного описания, например пакет Rational Rose и язык объектно-ориентированного описания UML. В дистрибутив современных серверов данных входят средства проектирования баз данных, например для сервера данных ORACLE в дистрибутив включены средства описания бизнес-процессов Designer 2000 и средства визуального программирования Developer 2000.

9. Безопасность баз данных

9.1. Состояние современных средств защиты данных

В настоящее время безопасные СУБД редко используются на предприятиях. Основной причиной является недооценка последствий разрушения данных и дороговизна систем безопасности. Системы безопасности в основном разрабатываются оборонными министерствами. Интересы таких ведомств и коммерческих структур различны: правительства опасаются утечки секретных данных, а коммерческие структуры — разрушения информационных систем и ущерба для бизнеса.

В 1985 г. были опубликованы основные документы, определяющие систему безопасности баз данных — Trusted Computer Systems Evaluation Criteries, — критерии оценки доверия к компьютерным системам. Документы содержали разделы по безопасности телекоммуникаций, компьютерных сетей и защиты баз данных в минимальном объеме. Полное определение защиты данных было дано в 1991 г. в документе, который назывался «Интерпретация защиты данных». Он определял 4 уровня защиты баз данных: 1) A — идеальный уровень, полная защита данных, 2) B (B1, B2), 3) C, 4) D — нет систем защиты.

Современные коммерческие системы баз данных работают на уровне безопасности C. Этот уровень включает в себя децентрализованную систему защиты данных. Децентрализация заключается в том, что не все полномочия выделяет администратор баз данных. Полномочия на пользовательские объекты выделяет пользователь их создавший. Привилегией или полномочиями

называется право совершать определенные операции над определенными объектами. Проверку полномочий называют проверкой санкционированного доступа. Кроме проверки полномочий любая система безопасности должна проверять подлинность пользователя, систему проверки подлинности называют аутентификацией.

Система безопасности присутствует в дистрибутиве любого сервера данных как предопределенная роль системного администратора, которому по умолчанию даны широкие полномочия по созданию и управлению ресурсами.

Обычно система безопасности любого сервера данных предусматривает типичные системные привилегии для стандартного пользователя. Такие привилегии включают в себя право создавать рабочие таблицы, индексы, представления, ограничения и выполнять операторы: select, update, insert, drop, delete, references, alter и т. д.

Привилегии на объекты, созданные пользователями называются объектными привилегиями, которые назначаются создателем объекта с помощью оператора grant.

grant <прив.1>, <прив.2> … on <объект1>, <объект2> … to <польз.1>, <польз.2> …;

grant select,update (fio, date_b), delete
on cadrs
to user sasha, masha.

Оператор снятия привилегии — revoke <прив.1>, <прив.2> … on <объект1>, <объект2> …from <польз.1>, <польз.2> …

Единицей выделения полномочий в системе безопасности является не только пользователь, но и группа пользователей, имеющих одинаковые полномочия или одинаковую роль. Роли создает системный администратор в соответствии с бизнес — процессами своего предприятия.

create role developer;
create role account;
grant
:::::.
to developer.

9.2. Перспективы развития систем безопасности

Современные системы безопасности строятся как системы мониторинга, контролирующие матрицу полномочий, в которой

колонки представляют объекты, а строки – пользователей. В ячейках указываются полномочия. Такая система безопасности не препятствует передаче полномочий между пользователями и не обеспечивает достаточной безопасности.

Свободная передача полномочий между пользователями позволяет пользователю с более высоким грифом секретности передать данные пользователю с более низким уровнем секретности, поэтому в настоящее время разрабатывается многоуровневая схема защиты.
Многоуровневая система защиты подчиняется трем принципам.

1) Все данные имеют гриф секретности (совершенно секретно, секретно, для служебного пользования, не секретно). Материалы могут быть снабжены специальными пометками, например «не для иностранных представителей». Каждый пользователь характеризуется классом допуска в соответствии с грифом, который он имеет.
2) Классы допуска предполагают обратное наследование, то есть класс «совершенно секретно» имеет доступ к документам «секретно».
3) Пользователи с высоким классом допуска могут читать объекты, имеющие более низкий гриф секретности, но они не имеют права создавать свои объекты в рабочих областях пользователей с более низким классом допуска. Данный принцип предотвращает появление «тайных каналов» передачи полномочий от лиц с высшим классом допуска к лицам с низшим классом допуска. Выполнение принципов многоуровневой модели расширяет представление о реляционной таблице. Реляционная модель должна стать многоэкземплярной.

Глава 3. Проектирование баз данных в среде сервера данных ORACLE 7.3

1. Понятие сервера данных. Основные функции

1.1. История технологии обработки данных. Определение клиент-серверной технологии. Состав технических средств организации сервера и клиента

Перечислим основные этапы технологии обработки данных.
1. Централизованная обработка на основе майнфреймов и терминалов удаленного доступа
2. «Распределенная» обработка на автономных ПК.

3. Файл-серверная технология на основе локальных вычислительных сетей.
4. Клиент-серверная технология на базе локальных вычислительных сетей.
Клиент-серверной технологией обработки называют технологию, связанную с организацией двух функционально различных частей: сервера и клиента. Сервер выполняет все функции, связанные с обслуживанием БД. Клиент выполняет пользовательское приложение. Единица общения клиента и сервера – запросы. Запрос – оператор доступа к БД.
Стандартным языком запросов является язык SQL (структурированный язык запросов). Особенностью клиент-серверных технологий является повышенное требование к серверу БД. Чаще всего требуется наличие нескольких жестких дисков. Иногда масштаб обработки требует многопроцессорной машины (SUN, Spare, Station, Netra3, 64 MS). Требования к клиентской части традиционны: ПК работающий под Windows 95 или Windows NT.

1.2. Функции сервера

Сервера данных (СД) выполняют следующие функции.
1. Разделение доступа к общим информационным ресурсам между многими пользователями, обеспечение корректного доступа как на чтение, так и на запись одного и того же ресурса многими пользователями в одно и то же время.
2. Обеспечение корректности БД, то есть соблюдение всех ограничений целостности и правил отката. Ограничение целостности – это смысловые ограничения на значения атрибутов. Откатом называется отмена всех изменений, которые произведены некорректно закончившейся операцией (возврат к предыдущему корректному состоянию).
3. СД позволяют администрировать права доступа пользователей. Администратор пользователей ведет реестр пользователей и фиксирует для каждого из них доступные информационные ресурсы и разрешенные операции.
4. СД не используют файловую систему операционных систем для хранения рабочих таблиц, они самостоятельно распределяют дисковое пространство на нижних уровнях.
В настоящее время СД, в основном, представлены на рынке четырьмя программными продуктами: ORACLE, Informix, Sybase, MS/SQL-server. СД часто называют SQL-сервером, так как языком запроса

является SQL-язык. Существует несколько стандартов SQL (92,95) ISO. У каждого СД, кроме SQL-стандарта, обычно существует уникальная часть языка. Основная функция сервера – многопользовательский доступ. СД обслуживает репозиторий ИС. Репозиторий – совокупность файлов различных форматов и установок, например, файлов БД, индексов, словарей данных, представлений, табличных областей, описание пользователей. Обслуживание подразумевает создание, модификацию и изменение структур. СД контролирует данные. Данные называют целостными, если они удовлетворяют заранее определенным правилам. Существует два вида целостности:

а) автоматические ограничения целостности (требование уникальности ключа; ограничения, наложенные типом атрибута; взаимосвязь между атрибутами нескольких таблиц);

б) ограничения целостности, связанные с деловыми правилами, для выражения которых у администратора сервера имеется язык описания.

Обычно деловое правило описывается как процедуры или триггеры, которые выражают деловые правила на языке SQL. Функция многопользовательского доступа выполняется сервером за счет блокировок. Выделяют блокировки: исключающие и разделяемые. Исключающая блокировка предоставляет данные в автономное использование. Процесс, требующий исключения блокировки называется транзакция. Разделяемая блокировка разрешает совместный доступ к данным. Влияние исключающих блокировок на производительность системы можно ослабить за счет механизма отката, то есть сервер для любых данных, в том числе захваченных в автономное использование, имеет целостный вариант данных, следовательно, при любом сбое выполняется откат к целостным данным. Администратор сервера ограничивает полномочия пользователя и устанавливает соответствия между объектами данных и категориями.

1.3. Основные функции и компоненты клиентской части

Клиентская часть – пользовательское приложение, решающее задачу по визуализации данных, то есть клиентское приложение – интерфейс, понимаемый в широком смысле. В такой интерфейс включают как экранные формы, так и отчеты; локальные вычисления и генерацию запросов в БД сервера. При внедрении клиент-серверных технологий приходится разрабатывать клиентскую часть, причем для разработки есть развитые средства автоматизированного проектирования (ORACLE / DESIGNER 2000 D2000). При разработке крупных

проектов информационных систем выделяют следующие функции любых участников разработки.
1) Руководство организаций определяет требования к системе.
2) Системные аналитики разрабатывают постановки задач.
3) Дизайнер БД разрабатывает схему репозитория.
4) Программисты разрабатывают клиентские приложения.
Каждая из перечисленных групп разработчиков описывает будущую систему с разной степенью абстракции. Требования к информационной системе обычно выражают на уровне бизнес-процессов. Процессом называют выполнение деловых функций, которые являются звеньями одной технологической цепочки, которые заканчиваются получением конкретного продукта или услуги. Процесс изменения реальных бизнес-процессов, достигаемый в ходе разработки ИС, называют бизнес-процесс-реинженерингом. Результатом формирования требований является полное описание желаемых процессов и их взаимодействие. Для реализации моделирования процессов в состав D 2000 включены средства моделирования процессов Process Modelar.
На основе требований к системе аналитики разрабатывают постановки задач, которые опираются на общие термины информационных технологий. В частности постановка задач должна содержать описание данных; описание функций, группы которых объединяются в иерархии; описание потоков данных, согласующих функции и данные. D2000 включает в себя средства моделирования системы (Systems Modelar), диаграммер ER-диаграмм, диаграммер иерархий функций, диаграммер потоков данных. Дизайнер БД выполняет формирование таблиц, индексов, словарей, табличных областей, полномочий доступа. Разработка схемы БД осуществляется в рамках двух инструментов: в рамках языка SQL; средствами генерации БД, которые генерируют таблицы на основе ER-диаграмм. Дополнительным преимуществом генерации является соблюдение нормальных форм. В распоряжение разработчика приложений предоставлен как язык SQL, так и ряд средств визуального программирования: генератор экранных форм, отчетов, генератор диалога. Совокупность перечисленных инструментов составляет набор современных CASE-средств: Computer Aided Software Engeeniring.

1.4. Интерфейс сервера и клиента

Так как физически клиент и сервер работают на разных машинах, то интерфейс между ними осуществляется на нескольких уровнях:
1) уровень связи;
2) уровень сетевой операционной системы;

3) уровень промежуточных средств ORACLE: SQL*NET обеспечивают многопротокольность и конверторы между БД различных производителей;
4) уровень параллельного доступа клиентов к серверу ORACLE 7.3.
Многопротокольность промежуточных средств необходима, так как встречаются многосегментные сети, разные сегменты которых используют разные протоколы: Unix – TCP / IP, Novell – SPX / IPX. Сейчас действуют стандарты на язык SQL-92 (ANSI / ISO). Система конвертов работает на основе стандартного подмножества языка.

1.5. Понятие масштабируемости ИС. Сравнение файл-серверной и клиент-серверной технологии

Сложная ИС представляет собой целую совокупность различных технологий. Это – технологии ЛВС, БД, прикладных программ. Но конкретные технологии группируют связи с идеей распределения информационных ресурсов. Выделяют файл-серверные технологии и клиент-серверные. Единицей передачи информации в файл-серверной технологии служит файл, если приложению необходима запись БД, а она хранится на сетевом диске, то по сети передают файл БД. Функции файл-сервера:
1) разделение запросов на файлы;
2) организация сетевых дисков;
3) связь таких дисков с настройками станций через логические имена устройств.
Таким образом, файл-сервер обеспечивает прозрачность дискового пространства, предоставляемого рабочим станциям. Свойства ИС развиваться пропорционально развитию бизнес-процессов называют масштабируемостью. Эту проблему решает клиент-серверная технология, где единицей передачи данных служит отклик на запрос пользователя. Если в приложении необходима запись в БД, то в сеть поступает ответ на вопрос, т. е. одна запись. Клиент-серверная технология усиливает централизацию в обработке данных, т. к. сервер должен выполнять всю обработку запросов. Клиент-серверная технология должна обладать большим дисковым пространством, мощность машины должна быть большой.

2. Транзакции

2.1. Организация многопользовательского доступа к информационным ресурсам Понятие транзакции:

Транзакцией называют активный процесс, порожденный каким-либо запросом в многопользовательской системе. Транзакция, которая использует общий информационный ресурс в общем случае при одновременном срабатывании работает некорректно. Например, транзакции T_1 и T_2 (рис. 3.1.) должны дважды изменить значение X, но при параллельном исполнении X меняется только один раз.

$T_1 ; T_2$ { Read X; X=X+1; WRITE X }

$T_1 ; T_2 = 3$

$t_1 : T_1$, read $X^1_{loc} = 1$

$t_2 : T_2$, read $X^2_{loc} = 1$

$t_3 : T_1$, X=X+1 $X^1_{loc} = 2$

$t_4 : T_2$, X=X=1 $X^2_{loc} = 2$

$t_5 ; T_1 \quad X^1_{loc} \rightarrow X=2$

$t_6 ; T_2 \quad X^2_{loc} \rightarrow X=2$

Рис. 3.1. Проблема общего ресурса для нескольких транзакций

Каждая транзакция считается успешно закончившейся, если она выполнялась до конца. Если транзакция не может закончится корректно, она должна вернуть общий информационный ресурс к исходному состоянию. СД для обслуживания транзакций выполняет блокировки ресурсов (рис. 3.2.) и откат общего ресурса к исходному состоянию.

БЛОКИРОВКА

Read A	Read B
Read B	Read A

t_1 : Read A - blok

t_2 : Read B - blok

⟨ …..	⟨……
write A	write B
write B	write A

Рис. 3.2. Коллизии кратных ресурсов

Механизм блокировок не избавляет ИС от коллизий. Клинч – это взаимная блокировка транзакций. СД обнаруживает взаимно заблокировавшиеся транзакции и откатывает все транзакции по истечении предельного времени транзакции (по тайм-ауту). Разработчики ИС, предназначенные для многопользовательского доступа, должны придерживаться определенной дисциплины разработки. Все информационные ресурсы, которые одновременно используются во многих транзакциях, должны вызываться в одной и той же последовательности. Соблюдение такого правила программирования называют двухфазным протоколом, то есть исходный текст программы любой транзакции должен состоять из двух фаз:

1) захвата и блокировки всех потребных ресурсов;
2) выполнения обработки и освобождения ресурсов, причем ресурсы во всех транзакциях захватываются в одном и том же порядке, противоположном порядку захвата.

Программирование многопользовательских систем порождает дополнительные требования к отладке систем. Транзакции называют сериализуемыми, если параллельное выполнение транзакций дает такой же результат, как и последовательное выполнение. Поэтому отладка должна включать в себя те транзакции, которые могут работать одновременно.

Старшие версии серверов ориентированы на SQL3. Общим подмножеством функции перечисленных серверов данных является следующее:

1) управление транзакциями, то есть сервер способен управлять запросами на чтение и запись, своевременно их блокируя;
2) обеспечение надежности, то есть выполнение отката к непротиворечивому состоянию данных при непредвиденных обстоятельствах;
3) оптимизация запросов.

2.2. Управление транзакциями

Транзакция – запрос к серверу данных или прогон программы. Транзакции выполняются параллельно. При параллельном выполнении транзакции большинство проблем сводится к использованию общих ресурсов.

Правило параллельного проектирования транзакций должно быть таким, чтобы результат параллельного и последовательного выполнения совпадал. Обычно при анализе противоречий

расписывают выполнение команд с учетом времени старта каждой команды. Такие последовательности команд называют расписанием. Если расписание подчиняется правилу, то говорят о легальном расписании.
Пусть в распоряжении программиста имеются следующие команды:
lock a – заблокировать ресурсы a;
unlock a – разблокировать ресурсы a.
Решение проблемы общего ресурса – блокировки будет следующим.

```
T1:  lock a
     read a
     a:=a+1
     write a
     unlock a

T2:  lock a
     read a
     a:=a+1
     write a
     unlock a
```

Все будет выполняется последовательно, а не параллельно. Наличие блокировок не исключает двух видов противоречий: бесконечного ожидания и тупиков.
T1: lock a, unlock a
T2 , T3 : lock a, unlock a
T4:
Таким образом T2 может ждать бесконечно. Проблему бесконечных ожиданий решает планировщик, встроенный в сервер. Например: с помощью дисциплины обслуживания «первым вошел – первым получил ресурс».
Появление тупиков связано с использованием кратных ресурсов.
T1: lock a; lock b, ..., unlock a, unlock b
T2: lock b; lock a, ..., unlock b, unlock a
Проблему тупиков можно решить двумя способами.
1. Сервер данных должен уметь отыскивать тупики, аннулировать результаты транзакции и активизировать только одну из них. Для реализации такого способа необходим математический аппарат, позволяющий выявить тупики.
2. Разработчик транзакций должен придерживаться некоторых правил проектирования: разработать линейный порядок для всех ресурсов и блокировать ресурсы во всех транзакциях только в этом порядке.

Для поиска тупиков надо представить расписание транзакций в виде графа предшествований. Если такой граф имеет циклы, то имеют место тупики. Если граф ациклический, то имеет место легальное расписание. Из теории графов известна процедура, преобразующая ациклический граф в последовательность:
1) найти вершину не имеющую входных дуг;
2) перенести вершину в последовательность и исключить из графа;
3) если граф не пуст, то повторить шаг первый, иначе конец.
Описанная процедура позволяет построить по расписанию последовательность выполнения, причем результаты последовательного и параллельного выполнения эквивалентны. Способность расписания быть эквивалентным последовательному выполнению транзакции называется сериализуемость. Свойством сериализуемости обладают транзакции, подчиняющиеся двухфазному протоколу. Это правило, требующее от каждой транзакции в первой фазе заблокировать все ресурсы, а во второй фазе симметрично их разблокировать. Откат аварийно завершившихся транзакций выполняется за счет полной регистрации модификации данных в журнале операций.

2.3. Развитие обработки транзакций

Традиционно обработка транзакций представляет собой ядро любого сервера данных. Обработка транзакций обеспечивает корректный многопользовательский и многозадачный режим. Современная теория обработки транзакций решает проблемы многопользовательского доступа в однородной среде. Типичным примером однородной среды является централизованная обработка данных, при которой сервер данных работает на мэйнфрейме, а пользователи работают за терминалами.

В локальных вычислительных сетях используются распределённые базы данных, причем иногда на различных рабочих узлах устанавливаются различные СУБД. С точки зрения общей организации сети такие СУБД называют локальными менеджерами ресурсов.

Транзакция, выполняемая в распределенной среде, характеризуется большей вероятностью неуспеха, так как из многих компьютеров, на которые распределены фрагменты базы данных, может оказаться хотя бы один неисправный, поэтому для неоднородной базы данных в настоящее время развивается новое

понимание транзакций. Классическая транзакция подчиняется четырем принципам.

- Атомарность. Транзакция – это последовательность операций, которые считаются неделимыми. Вся последовательность должна быть выполнена целиком или же все необходимо вернуть в исходное состояние.
- Целостность. Любая транзакция должна преобразовывать базу данных только в корректное состояние, то есть такое в котором выполняются ограничения целостности.
- Изолированность. Под ней понимают свойство транзакции скрывать произведенные модификации над данными от других транзакций.
- Долговременное хранение результатов. Результаты успешно выполнившихся транзакций фиксируются, то есть такие результаты хранятся долговременно до какой-либо следующей изменяющей транзакции.

Современные сервера данных обрабатывают транзакции, которые называются плоскими. Плоскими называют транзакции, имеющие один уровень управления. Один уровень управления для последовательных операций подразумевает линейный порядок их управления, который не разрешает какую-либо вложенность. Для решения проблем неоднородных баз данных было бы целесообразно разбить транзакцию на части — субтранзакции, некоторые из которых могут фиксировать свои результаты, могут быть вложены друг в друга, то есть вместо плоской транзакции должны работать сложные транзакции. Идея сложной транзакции берет свое начало от механизма контрольных точек, которые встроены в большинство коммерческих серверов данных. При создании приложения для такого сервера данных программист кроме традиционных операторов («начать транзакцию», «закончить транзакцию») имеет в своём распоряжении оператор «контрольная точка», где фиксируются все сделанные изменения, а оператор отката содержит в качестве параметра имя контрольной точки, до которой необходимо откатить транзакцию в случае неуспеха.

Дальнейшим развитием идеи сложных транзакций была идея вложенной транзакции, для которой необходимо сформулировать дополнительные правила выполнения, дополняющие четыре базовых принципа транзакции. Принцип изолированности для вложенных транзакций уточняется следующим образом: результаты изменений дочерней субтранзакции видимы для родительской транзакции, но по-прежнему невидимы для всех других транзакций.

Если какая-либо из вложенных транзакций завершилась аварийно, то откат осуществляется до родительской транзакции, то

есть принцип целостности выполняется только для всей транзакции целиком, выполнение же какой-либо субтранзакции может приводить и к некорректному состоянию базы данных.

В настоящее время обработка сложных транзакций выполняется в исследовательских университетских проектах, например: Interbase (Университет Индиана) предлагает специальный язык для программирования транзакций, а проект университета Питсбурга предлагает использовать специальные макросы и библиотеки, дополняющие C++ и позволяющие описывать транзакции.

В рамках международного института стандартизации в настоящее время работает группа над стандартом обработки транзакций; известен стандарт, сертифицирующий обработку плоских транзакций и рассматриваются проекты для обработки транзакций сложной структуры и транзакций в ООБД.

3. Администрирование баз данных. Ограничения целостности

3.1. Обеспечение непротиворечивости БД

Ограничения целостности хранятся в виде совокупности правил среди служебных данных файлов сервера данных. Для сервера данных ORACLE 7.3 ограничение целостности получили наименование бизнес-правил. Бизнес-правила записываются в форме условных операторов (например, в банке операционный день до 15.00, после этого времени не пройдет ни одна транзакция). Одним из средств эксплуатации больших ИС служат словари данных. Словари данных – это служебное описание объектов рабочих таблиц и их атрибутов. Крупные БД могут достигать нескольких десятков тысяч рабочих таблиц. В современных версиях серверов данных понятие словаря данных расширено до понятия репозитория проекта. Репозиторий кроме словаря данных включает в себя описание бизнес-правил, экранных форм, типичных запросов, встроенных процедур и триггеров. Триггером называют процедуру, которая хранится и выполняется на сервере, причем, срабатывает по определенному событию. Встроенными процедурами в контексте сервера ORACLE 7.3 являются процедуры, хранимые и выполняемые сервером, но активизирующиеся некоторыми приложениями.

СД ORACLE 7.3 кроме обеспечения ссылочной целостности обеспечивает ограничение проблемной области. Также ограничения называются assertions – утверждения – и являются одним из видов объектов БД. Ограничения принадлежат рабочей таблице, а не приложению. СД отслеживает выполнение ограничения для изменений рабочей таблицы, выполняемых любыми приложениями.

Для достижения смысловой целостности данных в некоторых СД расширяют понятие домена до понятия абстрактного типа данных (АТД). В младшей версии СД ORACLE 7.3. использовали только простые типы данных. Разработчик приложений не мог определить домен. Старшие версии СД ORACLE 7.3 позволяют определить домен как специальный тип данных. Например, домен – номер страхового полиса физически представляется целым числом, но такой домен позволяет присваивать значение только номеров страховых полисов, а не произвольных целых чисел.

3.2. Администрирование пользователей для сервера данных ORACLE 7.3

Задача администрирования пользователей представляет собой не только регистрацию пользователей и их паролей, но и формирование связей между объектами 3 типов:

1) пользователь;
2) информационный объект;
3) операция.

Примерами информационных объектов служат:

- рабочие таблицы;
- хранимые процедуры;
- фрагменты бизнес-правил;
- типичные запросы;
- операции, представляемые операторами языка SQL.

Для больших корпораций администратор пользователей сервера данных ORACLE 7.3 может создать описание, подходящее для групп пользователей, которое называют ролью. Примером роли может быть бухгалтер материального стола, операционист кассового зала, обычно администратор пользователей классифицирует конкретных пользователей под определенную роль. В настоящее время описание прав доступа пользователя часто называют бюджетом пользователя. Функции СД по контролю за бюджетами пользователей называют аудитом. Аудит представляет собой контроль всех запросов на соответствие запрошенных операций бюджетам пользователей.

Совокупность внутрикорпоративных правил по формированию прав доступа называют политикой безопасности. Диапазоны политики безопасности колеблются от полного отказа от ограничения прав доступа до организации тотального порядка информационных ресурсов. Тотальный порядок характерен для организации, в которой цена разрушения ИС очень высока (банковская система).

Пользователь с точки зрения СД представляет собой код авторизации доступа к ресурсам. Пользователь представлен всеми активными порожденными процессами: интерактивными и запущенными приложениями. Как интерактивный процесс, так и запущенные приложения называют SQL-агентами, т.е. с точки зрения СД пользователь – это совокупность SQL-агентов, подчиненных определенным правам доступа. Для SQL-агентов различают унаследованные права – **invoker's right** «права вызвавшего» и определенные права для данного приложения – **definer's right** «права определенные». В стандарт SQL-92 вошел оператор **grant**, позволяющий пользователю уступить часть своих прав другому пользователю. Наличие такого оператора позволяет строить как централизованные системы управления пользователями, в которой все права распределяет администратор системы, так и децентрализованные, в которых полномочия на управление некоторыми правами переведены пользователем. Клиент-серверная технология требует фиксации сеансов пользователя через процедуру подключения к серверу и отключения от сервера. В настоящее время процедура сеанса разрабатывается каждым СД по-своему. В требования стандарта данные операторы не включены. Планируется их включение в очередной стандарт SQL.

3.3. Конфигурирование сервера

Инсталляция сервера ORACLE требует выполнения предварительных условий подготовки:

1) нужно иметь локальную сеть с установленной сетевой ОС;

2) должны быть известны основные параметры будущей компьютерной системы.

Параметры выясняются в ходе ответов на вопросы.

1. Сколько планируется пользователей?
2. Сколько транзакций в среде порождает каждое приложение?
3. Каков первоначальный объем данных?
4. Какой прогнозируется рост объемов данных?

В ходе инсталляции сервера создается мощная БД, которая может создавать таблицы приложений, а может быть создана как фиктивная. Обычно администратор сервера создает два экземпляра ORACLE 7.3 для двух БД (рабочей и текстовой). Текстовая БД – для разработчиков, рабочая – эксплуатируемая.
Инсталляция сервера заканчивается созданием начальной БД и трех стандартных ролей администратора:
1) SYSTEM (Manager);
2) SYS (change_in_install);
3) INTERNAL (внутренняя).
Роль 1) позволят использовать утилиты администратора, т. е. создавать пользователей, выделять табличные области, следить за процессами; роль 2) позволяет делать все, что SYSTEM и дополнительно вести словари; роль 3) делает, что SYSTEM и пуск и остановку сервера.
Остановка является нормальной, если пользователи закончили работу с БД и ненормальной при обрыве работы.
В большинстве СД рабочая таблица представлена двумя видами таблиц. Базовая постоянная рабочая таблица (РТ) – это создаваемая таблица постоянного хранения. Определение таблицы входит в схему информационной БД. Представление (views) – это таблица, создаваемая из столбцов других рабочих таблиц БД постоянного хранения, необходимая для работы некоторых приложений. В новом стандарте SQL-языка введены временные рабочие таблицы. ВРТ (временные рабочие таблицы) бывают создаваемые оператором create и объявляемые оператором декларации declare. Объявляемые временные таблицы должны появляться в рамках одного приложения, объявляющего их, и уничтожаться после завершения сеанса приложения. Создаваемые временные таблицы являются таблицами постоянного хранения, причем их определение входят в информационную схему. Особенностью создаваемых временных таблиц является возможность задания зон видимости для одного или нескольких приложений.
Курсор – это таблица постоянного хранения, формируемая запросом пользователя. Курсоры могут быть определены со следующими атрибутами:
- read only – только для чтения – неизменяемый курсор, т. е. не реагирующий на изменение БД;
- scrolled – скроллируемый курсор – можно просматривать последовательно в некотором порядке, имитирующем порядок физического проекта.
В старших версиях SQL можно динамически формировать курсор.

3.4. Операторы управления транзакциями

Транзакция в БД – процесс, который должен быть выполнен целиком. Необходимо обозначить начало и конец транзакции. Нужны операторы для этих действий – операторы отката. Транзакция по своему определению требует выполнения двух системных действий:
1) фиксации начала и окончания транзакции;
2) отката транзакции.
Многие операторы управления транзакциями в реальных СД не подчинены требованиям стандарта. В стандарт входит два оператора:
- Commit – фиксация;
- Roll back – откат.
В настоящее время разработчик приложений может установить 4 уровня блокировок для выполнения транзакций:
- транзакции только читают данные, причем не фиксируются;
- транзакции только читают данные, но их действия фиксируются сервером;
- транзакции повторно читают данные;
- транзакции читают данные по частям.

4. Состав программного продукта ORACLE 7.3

Сервер данных ORACLE 7.3, как программный продукт, включает в себя, кроме сервера, большую совокупность программ автоматизации проектирования. В частности, данный программный продукт включает в себя следующие части.
1. Центральная компонента , которая представляет собой сервер данных, СУБД.
2. Программа бизнес-моделирования, которая предназначена для самого общего описания бизнеса корпорации, то есть создания моделей бизнеса. Обычно такая модель представляет собой описание бизнес-функций и схем информационных потоков. Для исходных моделей необходимо имитировать реализацию бизнес-функции, в частности, это может быть сделано на основе теории массового обслуживания. Частый вариант моделирования – это прогнозирование линий очереди заявок на обслуживание.
3. Программа логического проектирования БД (D 2 000 – Дизайнер 2 000).
Программа D 2 000 включает в себя диаграммер иерархий функций, ER-диаграммер и связана с генераторами структур рабочих таблиц для проектируемой БД.
4. Инструментальные средства разработки приложений (Developer 2 000). Предназначаются для создания клиентских рабочих мест. Данное

инструментальное средство состоит из генераторов 3 видов: генератор структуры рабочих таблиц, в том числе по ER-диаграммам; генератор экранных форм, представлений запросов; генератор отчетов или выходных документов.
5. Служба импорт-экспорта БД различных форматов. Служба необходима для обслуживания различных клиентов, в том числе работающих в другой среде (MS DOS и т. д.).
6. Web-server как средство представления внутрикорпоративных БД в Internet.

5. Стандарты SQL

5.1. Виды реализации SQL-языка Особенности реализации языка SQL в сервере данных ORACLE 7.3

Конкретные реализации **SQL** для различных серверов и версий серверов отличаются двумя свойствами:
1) полнотой реализации SQL-стандарта (SQL 92);
2) схемой реализации языка.
Современные SQL-реализации можно поделить на три типа.
1. Автономная реализация представляет собой многопользовательский интерпретатор.
2. Статический SQL-реализация SQL для разработки приложения (т. е. для программиста).
3. Динамическая реализация SQL.

Язык SQL существует в двух формах: встроенный SQL, модульный SQL (компановщик). Встроенным язык называется в том случае, если операторы SQL встроены в язык программирования (C, Pascal, PL/I, COBOL, Fortran).
Модульный язык позволяет компоновать модули на языках программирования на уровне объектного кода. Модульная реализация SQL была введена стандартом SQL-86 и поддерживается современными стандартами для достижения совместимости. Динамическая реализация SQL представляет собой возможность формирования запроса в ходе работы приложения, реализована в новейших версиях серверов данных (ORACLE 8.0).
Из-за непроцедурности SQL-языка разработчики SQL-языков формируют собственное расширение SQL до уровня процедурного языка. Такое расширение процедурного языка называют PL/ SQL. Расширения разработчиков являются несовместимыми между собой.

Например, ORACLE и Informix в лучшем случае совместимы с собственными версиями причем, снизу вверх.

SQL-операторы можно сгруппировать в три группы операторов: операторы определения данных (Data Definition Language DDL); операторы манипулирования данными (Data Manipulition Language DML); операторы управления данными (Data Control Statement. Особенностью СД является способ обработки атрибутов с неопределенным значением. ORACLE SQL всегда реализовывал трехзначную логику, то есть, кроме значения – истина и ложь, используется значение атрибута неопределенный. Причем, трехзначные логические выражения определяют результат выполнения запроса.

5.2. Основные понятия реляционных таблиц и их реализация в SQL-языках

СД строят рабочие таблицы, соблюдая следующие принципы реляционных баз данных (БД) (рис. 3.3.).

1. Используются только простые типы данных.
2. Используются только отношения, т. е. двумерные таблицы.
3. Операторы SQL используют только логику и не используют порядок физического следования строк.
4. Связывание таблиц организуется через запрос.

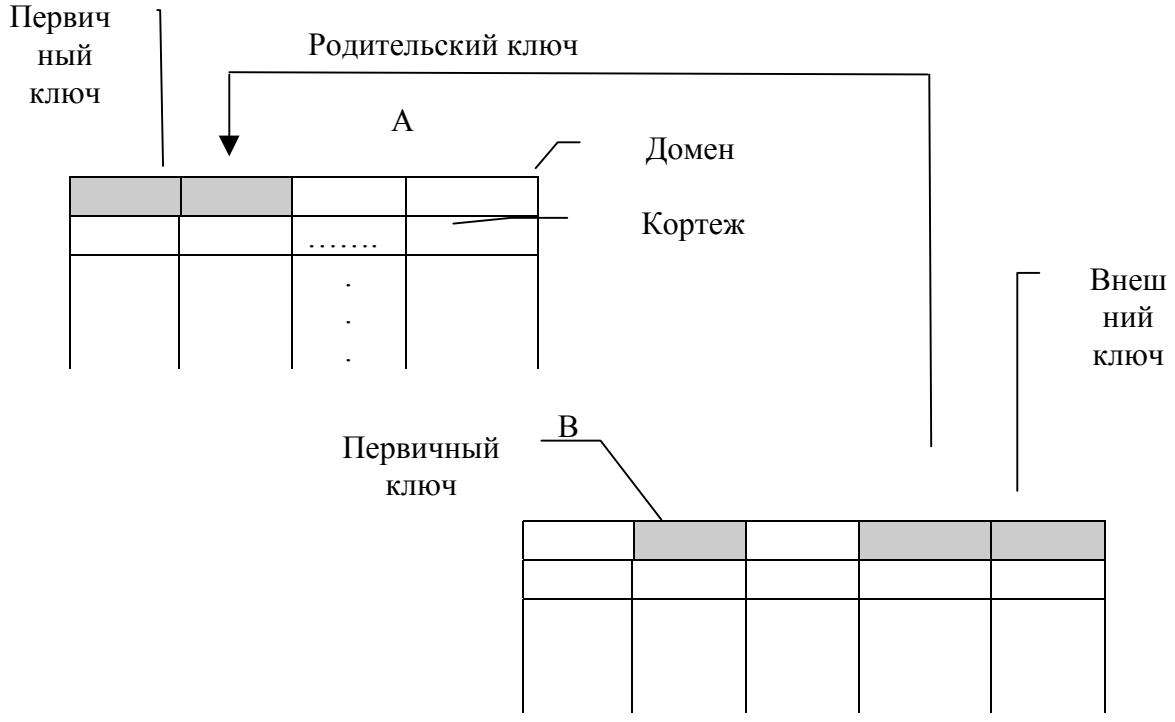


Рис. 3.3. Реляционные таблицы

Результатом запроса может быть третья таблица или соединение таблиц. Отклик на запрос называют курсором. Отклик на запрос к нескольким связанным таблицам так же называют курсором. Курсор – это временная рабочая таблица, создающая соединение (эквисоединение) строк нескольких взаимосвязанных таблиц. Многие параметры операторов запроса select можно объяснить, принимая во внимание алгебраические особенности выполнения операций на реляционной таблице.

5.3. Основные понятия метаданных

Слой метаданных (МД) исторически очень долго не подлежал стандартизации, например, понятие информационной схемы существует только начиная со стандарта 1992 г. Поэтому следующие термины для обозначения служебных понятий разными СД используются различно.

Под базой данных в серверах данных понимают не одну рабочую таблицу, а набор рабочих таблиц и соответствующих сопутствующих объектов. Рабочая таблица – это базовый элемент любой базы данных. Схема – это тоже рабочая таблица. Так как база данных является сложным понятием, то пользуются объединяющим термином – информационная схема. Информационная схема – это служебное описание всех именованных объектов БД (рабочих таблиц, доменов, представлений, запросов, курсоров и т. д.). Информационная схема логически представляет из себя служебную таблицу, с которой администратор БД работает с помощью языка SQL. Такое свойство метаданных называют замкнутостью. Схемой данных называют фрагмент БД, доступный определенному пользователю в соответствии с его привилегиями, следовательно, каждый пользователь имеет уникальный код авторизации, с которым и связаны его привилегии. Под привилегиями пользователя можно понимать набор прав доступа к БД.

Для сложных корпоративных систем часто определяют групповых пользователей (кассир, старший кассир, бухгалтер и т. д.). Объединение нескольких схем данных называют каталогами. Набор каталогов называют кластером.

6. Оператор выборки языка SQL

6.1. Структура языка SQL

Версия SQL ORACLE 7.3 включает в себя следующие составные части.

1. Команды манипулирования данными, которые позволяют осуществлять поиск и модификацию таблиц (select, insert, delete, update).
2. Команды управления транзакциями (commit, rollback, savepoint). Под транзакцией понимают запрос, который не должен быть прерван без нарушения целостности данных. Поэтому команды управления транзакциями устраняют и отменяют блокировки, организовывают точки сохранения и выполняют откат к точкам.
3. Команда определения данных (DDL) (create, table, role, function).
4. Расширение языка SQL до уровня процедурного языка PL / SQL.

Наличие расширения PL / SQL позволяет использовать ORACLE 7.0 не только в режиме интерпретации, но и компиляции. Если скомпилированная процедура выполняется сервером, то выигрыш в производительности оказывается значительным.

В состав языка PL / SQL включены:

1) конструкции определения типов данных и абстрактных типов данных, определение переменных;
2) правила формирования выражений (в том числе с функций);
3) операторы управления программами (if, to, while, for,...);
4) конструкции оформления программы, т. е. правила оформления функций, процедур, пакетов, триггеров.

6.2. Команды манипулирования данными

Команда выборки данных

Пусть задана БД Заказы (Orders) (рис. 3.4.)

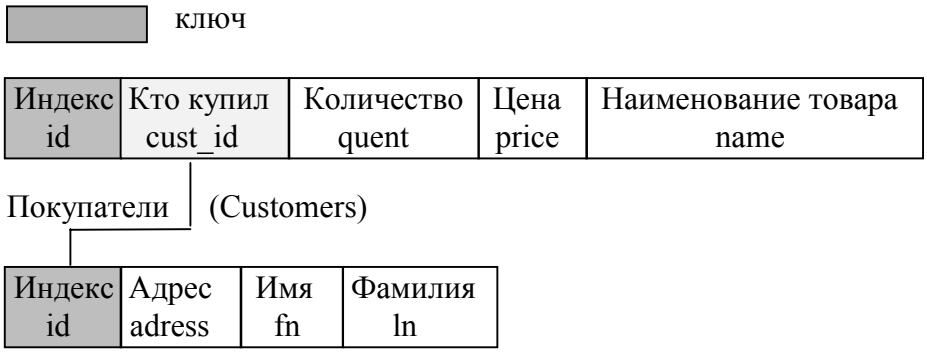


Рис. 3.4. Пример реляционных таблиц

Простейшая форма оператора выборки следующая.

Select <список столбцов> from <БД>
where <условие>.

Список столбцов – имена столбцов через**,** или ***** (все столбцы). Условие – выражение, операндами которого служат имена столбцов таблицы, константы, результаты вложенных операндов select.

Пример:
SQL> select * from Orders; – показать таблицу заказов
SQL> select id, adress from Customers; – показать номера и адреса покупателей
SQL> select id, cust_id from Orders
where price < = 1000; – показать номера заказов и покупателей, если цена купленного товара больше 1000
SQL> select id, cust_id
from Orders
where cust_id in (select id,
from Customers
where (adress like («Ульяновск»)));
- показать номера заказов и покупателей, если покупатель проживает в Ульяновске

Команда добавления строки

Insert into <БД>
values (<список значений>);

Insert into <БД>(<список столбцов>)
values (<список значений>);

Пример:
SQL> insert into Orders
values (33, 45, 10, 1.0, '...')
SQL> insert into Orders (id, cust_id)
values (33, 45);

Команда удаления

Delete from <БД>
where (<условие>);

Пример:
delete from Orders
where (id < = 50);

Модификация БД

Update <БД>
set (<выражение>)
where (<условие>);

Пример:
Update Orders
set (cust_id = 45)
where (cust_id < 33);

6.3. Полная синтаксическая и логическая схема выполнения запросов

Select <агрегатные функции> |
(<имя столбца...>) |
квалификатор.{<имя столбца...> * } |
from <имя таблицы> |
подзапрос as {<имя столбца>,...} |
соединенная таблица
where <предикат>
group by {<имя столбца>,...}
having <предикат>
union | intersect | except [all]
запрос В (оператор select)
order by {<имя столбца>,…).

Оператор select предназначен для извлечения строк из рабочих таблиц. Оператор может включать в себя другие операторы выбора, которые называют зависимыми от запроса. На практике оператор select не только извлекает строки, но и соединяет таблицы, проверяет предикаты, группирует строки, вычисляет агрегатные функции, объединяет результаты нескольких запросов и сортирует строки.

Схема выполнения оператора select сервером данных включает в себя следующие шаги.

1. Выполнение предложения from. Сервер данных либо проверяет права на использование указанной рабочей таблицы, либо соединяет перечисленные таблицы, либо начинает выполнять внутренний зависимый подзапрос. При выполнении соединений и вложенных подзапросов формируется виртуальная таблица запроса, которая существует только на время выполнения запросов, причем такая таблица может получить имя и переименованные столбцы. Такие виртуальные таблицы, существующие в рамках запроса, называют корреляциями (псевдонимами рабочих таблиц). После первого шага выполнения существует виртуальная таблица запроса, строки которой заполнены.
2. Выполнение предложения where. Все строки виртуальной таблицы запроса фильтруются в соответствии с условиями, определенными предикатом. Предикат может возвращать значение: истина-true, ложь-false, неизвестно-unknow. Предикаты в SQL-реализации относятся к трехзначной логике выполняют логические операции: or (или), and (и), not (не). Стандарт не определяет значение предиката для таких операций, поэтому каждый конкретный сервер данных может по-разному отфильтровать такие строки.
3. Выполнение группировки строк или предложения group by. Группировка понимается в том смысле, что в группу включаются строки с одинаковым значением указанного столбца.
4. Выполнение предложения having для перегруппированных столбцов. Появление предложения having связано с использованием агрегатных функций. Агрегатные функции выполняют операцию над столбцом рабочей таблицы или над частью столбца, входящей в одну группу. Предикат предложения having включает в себя агрегатные функции, и следовательно, тоже фильтрует строки, причем решение о включении в запрос принимается не об одной строке, а о группе строк.
5. Выполнение предложения select – это формирование ответа на запрос. Предложение select перечисляет имена столбцов, включая их в ответ на запрос. Для столбцов может использоваться агрегатная функция.
6. Выполнение объединения результатов двух или более запросов. Операция union – объединение – включает в запрос строки из результирующих таблиц, как запроса A, так и запроса B, причем дублирующиеся строки включаются. Операция Intersect наоборот исключает дублирующиеся строки. Except – исключает дубли

только из запроса B. Except all – определяет разность между количеством дублей запроса A и количеством дублей в запросе B и включает повторяющиеся строки в размере полученной разности.

7. Выполнение сортировки по предложению order by. После этого предложения пишутся имена столбцов. Список имен столбцов в предложении для сортировки формирует порядок расположения строк в отчете. Строки сортируются в соответствии с первым столбцом, если по первому столбцу есть повторы, то в соответствии со вторым столбцом и т. д.

6.4. Основные понятия обработки запроса

В процессе выполнения запроса формируется виртуальная картина запроса, которую называют корреляцией или псевдонимом запроса. Квалификатором или спецификатором при указании столбца называют контекст употребления столбца, то есть квалификатор определяет, в какую таблицу или корреляцию входит столбец. Такой элемент языка оказывается необходимым для того, чтобы точно указать повторяющиеся столбцы. На практике большинство запросов представляет собой не просто извлечение нужных строк, но и выполнение расчетов. Расчеты выполняются с помощью специальных функций, такие функции называют агрегатами, т. к. они действуют в пределах агрегата или столбца, таблицы или группы строк. Результаты запроса могут вычисляться с помощью традиционных выражений значений (*, + , -, /, ()). В процессе выполнения запроса можно осуществить переименование столбцов. В противном случае могут появиться несколько столбцов с одинаковыми именами. В процессе переименования можно осуществлять преобразование типов в пределах одного запроса.

Конструктор значения строки: (например, **Values** (0001, «.....»)).

Предложение **Values** позволяет сконструировать значение строки рабочей таблицы или виртуальной таблицы и использовать затем предикатом сравнения.

7. Подзапросы к нескольким рабочим таблицам. Вложенные подзапросы

7.1. Подзапросы, выполняющие соединения

Возможности SQL-языка по организации запросов к нескольким таблицам позволяют выполнять основные операции реляционной

алгебры (произведение, объединение, пересечение, разность). Произведение двух рабочих таблиц реализуется SQL – запросом, в котором отсутствуют условия соединения, например, выполним произведение БД кадров предприятия.

```
select c.emp_no,  e.dept_no
from emp c, dept e;
```

Косвенным признаком запроса, направленного к нескольким таблицам служит использование псевдонимов, они решают проблему одноименных столбцов в нескольких рабочих таблицах.

Операция соединения всегда связана с некоторым условием. Операция соединения позволяет получить отчет, содержащий столбцы из 2-х и более таблиц, причем строки соединяются на основании равенства значений некоторых однотипных столбцов.

```
select emp_no, job, sal, d.d_name
from emp, dept d
where dept_no = d.dept_no;
```

На практике кроме эквивалентности могут быть использованы любые условия (>, <, ≥ , ≤).

```
select emp_no, job, sal, d.d_name
from emp, dept d, salgrade s
where dept_no = d.dept_no;
and sal > s. low_sal;
```

Внешнее соединение (+) решает проблему формирования пустой строки, соответствующей по внешнему ключу другой рабочей таблице.

```
select c.emp_no, c.job, d.d_name
from emp c, dept d
where c.dept_no (+) = d.dept_no;
```

Операция внешнего соединения может употребляется после столбца той рабочей таблицы, в которой недостает данных.

Некоторые пользовательские запросы позволяют выполнить соединение строк одной и той же таблицы. Такие запросы реализуются за счет самосоединения, если какая-либо иерархия записана в реляционной таблице, то для получения отчетов по иерархии приходится использовать самосоединение. Напечатать табличные номера и должности сотрудников, с указанием табличного номера их руководителей и окладом руководителей.

```
select e.emp_no, e.job, a.mgr, b.sal
from emp e, cadrs b,
where a.mgr = b.emp_no;
```

Признаком самосоединения является присвоение одной и той же рабочей таблице разных псевдонимов.

Особенностью выполнения операций объединения, пересечения и разности служит тот факт, что аргументами операций являются результаты выполнения запросов.

```
select <столбцы>
from <таблицы>
:
where
union [all] / intersect / minus
select <столбцы>
from <таблицы>
where <условие>;
```

При выполнении теоретико-множественных операций над запросами необходимо согласование типов столбцов в каждом аргументе.

7.2. Вложенные запросы

SQL допускает вложение запросов до 255 раз, обработка вложенных запросов осуществляется от самого глубокого запроса к внешнему, причем результат предыдущего по вложенности запроса подставляется в условие верхнего запроса.

Пример: выбрать сотрудника, который занимает такую же должность, как и известный нам человек, табельный номер которого задан.

```
select emp_no, job, sal
from emp
where job = (select job
from emp
where emp_no = '0002' );
```

При организации вложенного запроса необходимо учитывать, какой результат вернет вложенный запрос (однострокового или многострокового и какой тип будут иметь столбцы результата).

Пример: найти всех служащих, с такими же должностями, что и в 30-м отделе.

```
select emp_no, job, sal
from emp
where job in (select job
from emp
where dept_no = '30' );
```

Если условие, требующее выполненного запроса содержит групповую функцию, то такой запрос вкладывается в предложение having. Например, запрос: найти всех сотрудников, получающих оклад выше среднего в своем отделе.

7.3. Соединение таблиц при манипулировании данными

Стандарт SQL-92 предусматривает отличный от вышеописанного сисинтаксис соединений. Современные сервера данных позволяют выполнять различные типы соединений таблиц. Соединение, выполняющее декартово произведение строк, называют перекрестным соединением (cross join), а соединение по совпадению значений заданного столбца называют естественным соединением (nature join).

<перекрестное_соединение>::= таблица A **cross join** таблица B
<естественное_соединение>::= таблица A **nature** [тип_соединения] **join** таблица B [атрибут]
таблица A **union** [тип_соединения] **join** таблица B [атрибут]
<атрибуты>::= **ON** <предикат>| **using** (<имя столбца>, ...)
Использование предиката **ON** позволяет отфильтровать строки, а предложение **using** позволяет задать имена столбцов, которые объединяют таблицы.
[тип_соединения] ::= {left/right/full}

A

Таб_номер	Фамилия
001	Иванов
002	Петров
003	Сидоров

B

Фамилия	Аудитория
Иванов	413
Петров	411
Кузнецов	420

A cross join B

Таб_номер	Фамилия	Фамилия	Аудитория
001	Иванов	Иванов	413
001	Иванов	Петров	411
…….	…….	…….	……

A nature join B

Таб_номер	Фамилия	Аудитория
001	Иванов	413
002	Петров	411
…….	…….	……

Рис. 3.5. Пример соединения таблиц

Объединение, соединение таблиц (рис. 3.5.) в рамках одного запроса может позволить сократить объем исходного кода приложения за счет использования дополнительных возможностей оператора select. Сложные запросы обладают недотерминированностью с точки зрения реализации.
Предикаты в стандарте SQL являются предикатами трехзначной логики, поэтому необходимо про каждую реализацию знать, каким

образом обрабатывается значение «неизвестно». Целесообразно использовать в запросе встроенные предикаты.

8. Коррелированные запросы

Для обработки рабочих таблиц бывает необходимо организовать выполнение вложенного запроса для каждой строки основной таблицы, причем результат выполнения вложенного запроса либо включает строку кандидат в запрос, либо отвергает. Примером коррелированного запроса может служить следующее: ввести всех сотрудников, получающих оклад ниже, чем среднее значение оклада по тому отделу, в котором служит сотрудник.

```
SELECT          ENAME, SAL SALARY, DEPTNO
FROM      EMP E
WHERE           SAL < (SELECT         AVG (SAL)
                              FROM      EMP
                              WHERE     DEPTNO = E .DEPTNO)
ORDER BY        DEPTNO;
```

Реляционные таблицы используются для хранения иерархических справочников:

- консолидация отчетов;
- часть – целое для узлов и деталей;
- подчиненность сотрудников.

Вводятся атрибуты для каждого объекта, характеризующие родительский узел. Существует особая форма оператора select, реализующая обход дерева. Для организации обхода дерева используется псевдостолбец с именем level, который содержит уровни иерархии для объектов, автоматически формируется сервером данных.
Пример: сформировать отчет содержащий следующие сведения: уровень подчиненности, табельный номер сотрудника, табельный номер его начальника и их оклады.

```
SELECT          LPAD (EMPNO, LEVEL*4) EMPLOYEE_NUMBER,
                        DEPTNO, EMPNO, ENAME, JOB, SAL
        FROM            EMP
CONNECT BY PRIOR EMPNO = MGR
START WITH MGR IS NULL;
```

Предложение connect by включает условие связывания родительской и дочерней вершин. Параметр prior управляет направлением обхода дерева. Если prior предшествует emp_no, то строка с таким табельным номером является предыдущей, а следующей будет строка с табличным номером начальника, т. е. дерево обходится снизу вверх.

Традиционное условное предложение where может использоваться для исключения из отчета отдельных строк. Если необходимо исключить из дерева обхода целую ветвь, то условие исключения добавляется в предложение connect by, такую операцию называют «стрижкой дерева».

9. Языки определения и манипулирования данными в SQL. Создание рабочих таблиц

9.1. Определение информационной схемы в стандарте SQL 92

Информационная схема представляет собой группу взаимосвязанных объектов, рабочих таблиц, представлений, ограничений, доменов. Схема принадлежит одному из пользователей, который идентифицируется при ее создании. Допустим, что необходимо создать информационную схему. Создатель – пользователь с идентификатором My_id. Схема будет названа My_schema. Схема должна содержать следующий объект: таблицу со сведениями о клиентах, причем, ключом таблицы является номер паспорта Pass_num, имеющий специальный тип Pass. Строки в БД должны храниться в виде символов русского алфавита Russian_ascii. Допустим, что правами выполнения запросов к рабочей таблице Clients обладает еще один пользователь с идентификатором other_id.

```
Create schema    My_schema
Authouration     My_id
Default cheracher set Russian_ascii
Create table Clients
(Pass_num number(6) primery key)
age      integer
.
.
.
.)
Createdomain pass as integer
Check (value>0)
Grant
Select*
To other_id;
```

Информационная схема – это группа взаимосвязанных объектов, это конкретная БД может содержать объекты-листья, в том смысле, что объекты созданы по определениям, описанным в информационной

схеме. Например, определение домена pass может быть использовано для определения других рабочих таблиц. Поэтому для уничтожения схемы можно предусмотреть 2 механизма ликвидации объектов.

1. Перед удалением схемы предполагается, что не существует объектов-листьев.
2. При удалении схемы объекты-листья преобразуются по определенным правилам.

Например, если рабочая таблица-лист использует домен pass, а схема My_schema удаляется, то ее столбцы получают определенные целые – integer. А условие check – теряет силу.

Drop schema My_schema – уничтожение схемы.

Cascaded / resctrict – механизм уничтожения.

9.2. Создание и уничтожение рабочих таблиц

Команда определения данных – это команда создать. В качестве создаваемых объектов могут выступать таблицы, роли, представления и другие объекты.

Create table <имя таблицы>
(<имя столбца> <тип столбца> <атрибуты>).

Для создания таблицы используется оператор Create table. Рассмотрим употребление оператора на примере создания следующей таблицы. Создайте таблицу COMPANY_CARS, используя приведенную ниже спецификацию. Определяя имена для правил целостности, вы можете выбирать их по своему усмотрению.

Таблица 3.1

Пример рабочей таблицы

Имя столбца	Тип данных и размер	Правила
CHASSIS_NUMBER	VARCHAR2 20	не может быть пустым, первичный ключ
MAKE	VARCHAR2 15	только большие буквы
MODEL	VARCHAR2 15	только большие буквы
CUB_CAP	NUMBER 4	
COLOR	VARCHAR2 10	
DOR	DATE	
REG_NO	VARCHAR2 10	не может быть пустым, только большие буквы
EMPNO	NUMBER 4	не может быть пустым, внешний ключ к EMP. EMPNO
DEPTNO	NUMBER 2	не может быть пустым, внешний ключ к DEPT.DEPTNO

```
CREATE TABLE COMPANY_CARS (
    CHASSIS_NUMBER VARCHAR2(20 )
        CONSTRAINT CC_CH_NO_NNULL NOT NULL
        CONSTRAINT CC_CH_NO_PK PRIMARY KEY,
 MAKE   VARCHAR2 ( 15 )
    CONSTRAINT CC_MAKE_CHKUPP
        CHECK (MAKE=UPPER (MAKE) ) ,
 MODEL  VARCHAR2(15)
    CONSTRAINT CC_MODEL_CHKUPP
        CHECK (MODEL=UPPER (MODEL) ) ,
 CUB_CAP NUMBER (4),
 COLOUR VARCHAR2 ( 10 ) ,
 DOR    DATE,
 REG_NO VARCHAR2 (10)
   CONSTRAINT CC_REG_NO_CHKUPP
        CHECK (REG_NO = UPPER (REG_NO))
   CONSTRAINT CC_REG_NO_NNULL NOT NULL,
 EMPNO  NUMBER ( 4 )
  CONSTRAINT CC_EMPNO_NNULL NOT NULL
  CONSTRAINT CC_EMPNO_FK
        REFERENCES EMP (EMPNO),
 DEPTNO NUMBER (2)
  CONSTRAINT CC_DEPTNO_NNULL NOT NULL
  CONSTRAINT CC_DEPTNO_FK
        REFERENCES DEPT (DEPTNO)
  );
```

Таблицы бывают глобальными или локальными global/local, временными temporary. Сервера данных различают рабочие таблицы по атрибутам их доступности и времени существования. Если не используется атрибут temporary (временная), то создается БД постоянного хранения, то есть она хранится после окончания сеанса. Временные БД существуют на время сеанса работы пользователя. Сеанс ограничен оператором:

Connect – disconnect

Если БД имеет атрибут local (локальная), то БД доступна лишь одному приложению. Если БД имеет атрибут global (глобальная), то она доступна многим приложениям во время своего существования. Оператор создания таблицы create table кроме определения свойств самих рабичих таблиц, позволяет определить обработку после завершения транзакции. Если в операторе on указан атрибут delete, то после прохождения транзакции строки таблицы удаляются. Если написан атрибут preserve rows, то строки сохраняются между транзакциями. Определение структуры рабочих таблиц требует определения первичного ключа, внешних ключей, ссылок на

родительский ключ, столбцов с уникальными и ненулевыми значениями. Определение любого оператора SQL предполагает использование усредняющих функций, служебных функций, ограничений, предикатов, использование которых повышает выразительные возможности оператора и делают даже синтаксически простой оператор сложным.

9.3. Служебные объекты информационных систем

Определение и уничтожение домена

Домен в БД – абстрактный тип данных, определенный на базе стандартного типа, использующий предикаты для контроля значений и предназначенный для определения столбцов рабочих таблиц.
Пример:
create domain <имя домена>
as <стандартный_тип_данных>
check <предикат_значений>;
для записи предиката значений необходимо использовать предикаты сравнения, существования, принадлежности и т. д.
drop domain <имя домена> – уничтожение домена
cascaded/resctrict;

Определение набора символов

Стандарт SQL учитывает особенности международных приложений СД. В частности, строковые данные типа содержат не только английские символы, но и символы национальных алфавитов. Поэтому возможно создавать свой набор символов. Строки в языке SQL являются сравнимыми объектами, т. е. предикат «а» больше «в» имеет значение истины, если «а» в наборе символов предшествует «в», следовательно, любой оператор создания набора символов должен задавать порядок сортировки или сравнения символов. В одном приложении может действовать несколько наборов символов, поэтому необходимо задавать перевод строки из одного набора в другой. Такой перевод называется трансляцией.
Create character set Russian_ascii
Get ("Russian.bin ");
Оператор уничтожения набора символов предполагает, что к моменту уничтожения ни один из объектов-листьев не использует данный набор символов.
Create translation <имя>
From («…….»);

Создание трансляции эквивалентно заданию или определению оператора перекодировки символов. Трансляция уничтожается drop translation.

10. Индексы. Словарь данных. Модификация структуры рабочих таблиц

10.1. Словарь данных

Словарь данных – это специализированная служебная БД, которая содержит сведения о пользователях, рабочих таблицах, представлениях и ограничениях целостности. Словарь данных содержит служебные рабочие таблицы фиксированной структуры. Для реальных приложений объем словаря данных может достигать десятков тысяч объектов. Словарь данных формируется автоматически по фиксированной структуре, когда пользователь выполняет команду создания БД. Доступ к словарю данных имеет пользователь с именем SYS. Словари данных полезны как администраторам БД так и продвинутым пользователям. Доступ к словарю данных открыт с помощью оператора select языка SQL. Структура словаря данных является уникальной для каждого сервера данных, т. е. не существует стандартов на словари данных.

10.2. Модификация структуры рабочих таблиц

В ходе эксплуатации реальных приложений модифицируется не только содержание рабочих таблиц, но их структура. Модификация может заключаться в следующем: добавить, удалить столбцы; добавить, удалить ограничение целостности; активировать или заморозить действие какого-либо ограничения целостности.

Рассмотрим пример оператора – добавить к рабочей таблице emp новый столбец типа data с именем hiredata:

Alter table add ..... emp (hiredata);

Из рабочей таблицы cadrs удалить ограничения целостности sal_5000:

Alter table drop construction sal_5000.

Любая модификация рабочих таблиц отражается в словаре данных. Ведение словаря данных относится к служебным операциям и, следовательно, замедляет работу сервера данных, т. е. относится к накладным расходам.

10.3. Индексы

Это специальные служебные файлы, которые обеспечивают прямой доступ к записи по ключу. ORACLE сервер создает индексы автоматически при создании рабочих таблиц, если явно указан первичный ключ (primary key). При поступлении запроса к индексированной рабочей таблице оптимизатор запросов использует индексы там, где это возможно. Оптимизатор запросов – это внутренняя программа ORACLE, транслирующая пользовательский запрос во внутренние операции и оптимизирующая последовательность внутренних команд. Оптимизаторы запросов, встраиваемые в сервера данных работают по двум методам.

1. Оптимизация на основе правил, т. е. оптимизатор настраивается на явно указанный критерий оптимизации. Например, если в запросе указан первичный ключ рабочей таблицы, то необходимо использовать индекс.
2. Кроме автоматических индексов можно построить вторичные индексы, т. е. те, которые построены не по ключу и, возможно, не по уникальному атрибуту.

create index <имя индекса> , <имя таблицы> (<имя столбца>, :);
create index dep_ind emp(dept_no);
select *
from emp
where (upper(dept_no)=10).

Индексы обычно используются в операторах запросов, если запрос включает предложение where. Если в условии используются выражения, тогда индексы оказываются неприменимы.

11. Представления в SQL

11.1. Создание ограничений

Ограничение assertion обрабатывается сервером БД и отрабатывает каждую транзакцию, изменяющую или добавляющую новые данные. Определение оператора

Create assertion <имя_ограничения>
Check (<предикат>).

Рассмотрим пример ограничения – поле номер паспорта не нулевое.

Create assertion My_ass
Check (pasport_num is no null).

11.2. Создание представлений

Современные СД позволяют сформировать виртуальные таблицы, называемые представлениями (view) для отображения рабочих таблиц на информационные потребности отдельных групп пользователей. Алгебраически представления — это проекция рабочей таблицы на заданный перечень столбцов. Чтобы заполнить представления конкретными данными, каждое представление связывается с оператором запроса (select). Представление доступно пользователю не только для просмотра, но и для модификации, добавления и удаления строк. Все изменения осуществляются с основной рабочей таблицы. Создаваемое представление доступно тем пользователям, которым разрешена работа с основной таблицей, т. е. представление наследует своих пользователей.

Создайте простое представление с именем CC_VIEW на основе таблицы COMPANY_CARS, которое выбирает столбцы chassis_number, reg_no, empno и deptno.

```
CREATE VIEW CC_VIEW
AS
SELECT      CHASSIS_NUMBER,
              REG_NO,
              EMPNO,
              DEPTNO
FROM        COMPANY_CARS ;
```

11.3. Объявление временных рабочих таблиц

Синтаксически создаваемый объект отличается от объявляемых объектов теми операторами, которыми они вводятся. Создаваемые объекты вводятся оператором Create. Объявляемые — Declare. Создаваемые объекты принадлежат информационной схеме БД, а объявляемый объект принадлежит модулю, программе, аналогично всем внутренним переменным программы.

Declare temporary local
Table My_ table
(id_name integer primary key,

price integer is no null)
on commit delete/preservs rows.

Объявляемая временная таблица всегда является локальной, т. е. недоступна для других выполняемых приложений. Оператор объявления задает состояние таблицы после завершения транзакции. Может быть установлено 2 режима:
1) delete-удаление строк;
2) preservs rows — сохранение строк между транзакциями в рамках одного приложения.

11.4. Объявление курсора

Курсором называют переменную, объявляемую в приложении и содержащую отклик на запрос.

Declare My_cursor [insensitive/scroll] cursor
For select new_price
From Sales
Where (new_price >=old_ price).

Объявляемый курсор создает столбик новых цен new_price**,** причем в столбик включены те цены, которые больше старой цены. Курсор, являясь внутренней переменной модуля, должен быть открыт с помощью оператора open cursor My_cursor. Оператор Open означивает содержимое курсора с помощью оператора select, записанного в декларации курсора. Если курсор снабжен атрибутом insensitive, то курсор предназначен только для чтения. Если атрибут scroll указан, то курсор можно последовательно просматривать, скроллировать, то есть курсор считается упорядоченным (ordered), и для него действуют предикаты: first, last, next, prior…

Представления могут использоваться несколькими приложениями, а курсор нет.

12. Управление транзакциями

12.1. Команды управления транзакциями

Команда завершения транзакции

Commit on-завершает транзакцию и снимает установленные блокировки.

Команда создания точки сохранения

Savepoint <имя точки>
Пример:
insert...
delete...
savepoint first
update
delete
savepoint two.

Команда отката к точке сохранения

Rollback <имя точки, к которой желаем вернуться>.

Пример:
rollback two.

Как команды манипулирования данными, так команды сохранения контекста, подразумевают работу пользователя в рамках установленных полномочий, которые в ORACLE 7.3 называют ролью. Каждый пользователь работает в рамках определенной роли. Роль описывает администратор сервера. Описание роли включает в себя перечень таблиц и функций, доступных пользователю.
Пример:
set role Order_counter
grant select on Orders
to order_counter
grant order_counter to user1, user2 ...

12.2. Управление транзакциями с помощью оператора SQL

Под транзакцией понимают единицу работы СД по обработке запроса. Запрос предполагает обязательное успешное выполнение, причем, всех операторов, входящих в запрос. Если транзакция не может завершиться успешно, то СД выполняет откат транзакции. Все промежуточные результаты обработки транзакции хранятся в системных журналах так, что СД всегда способен восстановить корректное сосотояние БД. В SQL-92 введены операторы, которые

позволяют программисту оформить последовательность операторов как транзакцию, а также выполнить откат.

Commit [work]; (зафиксировать [работу])
Оператор, не имеющий параметров
Оператор начала транзакции
rollback [work]; выполнить откат

create table……ON delete rows.

Некоторые операторы создания таблиц, например, временных таблиц, содержат специальные конструкции предложения ON, которые влияют на регистрацию транзакции. Например, предложение ON delete rows дополнительно требует удалить содержимое из временной таблицы по окончании транзакции. Следовательно, выполнение транзакций управляется с помощью контекста, т. е. зависит от других операторов. Влияние на выполнение транзакций оказывает не только другие операторы, но и оператор настройки SET. Если в информационной схеме пользователя созданы не только постоянные таблицы, но и ограничения, то пользователь может явно указывать, какие ограничения должна проверять транзакция. Причем ограничения могут проверяться в немедленном или отложенном режимах.

В реальных системах разработчик должен найти компромисс между плотностью контроля состояния БД и между быстродействием обработки запросов. Большинство запросов носит информационный характер, т. е. они работают только на чтение. Количество действий к таким запросам со стороны СД меньше, чем с запросом на запись. Явное указание транзакций «только чтение» ускоряет обработку запроса, но может привести к трем видам аномалий:

- «Грязное чтение» – транзакция А («read only») читает сразу, В — модифицирует. Транзакция А видит не окончательное состояние записи.
- «Неповторяющееся чтение» — А читает, В — модифицирует, но не успешно. В результате происходит откат. Невозможно повторно получить ответ на запрос.
- «Фантомное чтение» - возникает при обработке группы строк, запрос А — читает группу строк, удовлетворяющих предикату, В их модифицирует.

Под уровнем изоляции понимают степень независимости транзакции от выполнения других транзакций. Уровень Read uncommited позволяет транзакции читать незарегистрированные изменения, сделанные другими транзакциями. Read commited — позволяет читать только зарегистрированные транзакции. Repeatable read — подразумевает повторное контрольное чтение данных. Serializable read — подразумевает чтение данных из группы записей по одной записи последовательно. Diangnosies — установка диагностики, управляет количеством системных кодов, которые генерятся в случае неуспешной транзакции.

Уровни изоляции транзакций

Таблица 3.2

Уровень изоляции	Грязное чтение	Неповторяющееся чтение	Фантомное чтение
Read uncommited	Да	Да	Да
Read commited	Нет	Да	Да
Repeatable read	нет	Нет	Да
Serializable read	нет	Нет	Нет

Да — аномалия сохраняется.
Нет — аномалия ликвидируется.
При разработке клиент-серверных систем необходимо рассматривать взаимное влияние транзакций.

12.3. Средства подключения к СД

Подключение к СД — это связь пользователя или приложения с СД. Связь предполагает регистрацию активного идентификатора (ID) пользователя. С точки зрения обработки авторизации СД различает системную регистрацию пользователя, т. е. регистрацию в операционной системе, регистрацию сеанса работы пользователя и подключение приложения к СД.

Connect <среда_реализации_сервера>
User <имя>
As <имя_подключения>;

Connect IS_Department
User ised4
As link_1;
Disconnect <имя_подключения**>;**

Оператор **Connect** не только регистрирует подключение, но и делает его активным, а все ранее выполненные подключения являются пассивными. Оператор SET Connection <имя_подключения> делает указанную связь активной.

13. Управление пользователями

13.1. Определение привилегий для СД

Понятие «привилегия» в СД указывает для каждого пользователя перечень объектов, к которым пользователь имеет доступ и перечень операций, разрешенных ему для каждого объекта. Привилегия с точки зрения СД - это системная таблица, дескриптор привилегий (описатель), который содержит следующие поля:
— ID идентификатор пользователя;
— перечень объектов;
— перечень разрешенных операций;
— право передавать привилегии.

Под привилегией понимается возможность для пользователя выполнить конкретный оператор над столбцами конкретной таблицы.

```
Grant  select (id_num, price, Sales_date)
On Sales                    из таблицы Sales
To Shasha                   кому передаем
[with grant option];        даем право самому раздавать права
```

С помощью оператора grant для каждого пользователя формируется список привилегий, привилегии управляют работой сервера данных с точки зрения защиты данных. Выполнению каждой транзакции предшествует проверка привилегий пользователя, сеанс которого породил транзакцию.

Дескриптор привилегий — это единица работы СД с пользователем, т. е. сервер должен изменять дескриптор в соответствии с текущей ситуацией. Объектом привилегии может быть любой из объектов информационной схемы, в частности, рабочие таблицы, представления (view), утверждения (assertions), ограничения (constrains), наборы символов (character set), трансляции (translation), сортировки (collation). Перечень разрешенных операций задается в форме ключевых слов тех операторов, которые может использовать пользователь, причем для операций update (модифицировать), insert (добавить) можно указывать конкретные столбцы в рабочей таблице.

13.2. Оператор представлений привилегий

grant <привилегия>,…..
ON < объект >,….
TO <имя>
[with grant option];

grant select, update (Sales, num)
ON Sales_data
TO user1
with grant option;

Пользователь, предоставивший привилегию другому называется грантор (grantor — предоставитель). Привилегия является предоставляемой, если право на нее можно предоставить другим пользователям.
<имя_пользователя>:=PUBLIC
PUBLIC — имя пользователя распространяет привилегии на все множество зарегистрированных в системе пользователей.

13.3. Оператор отмены привилегий

Оператор отмены привилегий входит в стандарт SQL, начиная с 1992 года. Поэтому в ранних версиях различных СД встречаются уникальные операторы отмены. Основная сложность конструирования оператора отмены состоит в том, что определенная привилегия порождает множество зависимых привилегий. Следовательно, оператор отмены должен обрабатывать всю цепочку зависимых привилегий.

Revoke [with grant option]
< привилегии >,…
ON < объект >,…
FROM <имя_пользователя>
CASCADE/RESTRICT.

Предложение with grant option сохраняет за пользователем перечисленные привилегии, но отменяет его право передавать их кому-либо другому. Если оператор отменяет привилегию только одного дескриптора пользователя, а все зависимые привилегии сохраняются, то указывается параметр RESTRICT. Параметр

CASCADE подразумевает обработку цепочки зависимых привилегий. Стандарт SQL-92 вводит параметр CASCADE, но не определяет алгоритм его выполнения, поэтому каждый СД выполняет обработку уникальным образом. К зависимым привилегиям относят привилегии 2 типов.

- Привилегии, определенные грантором, который в настоящий момент лишается привилегий.
- Унаследованные привилегии, которые по рабочей таблице можно определить как представления, по набору символов как транзакции и сортировки.

Такие объекты наследуют права определителя. Наследование привилегий организуется как передача привилегий от системного SQL-агента с именем System_ к определенным объектам. На практике параметр CASCADE отменяет привилегии второго рода, т. е. для определенных объектов, а привилегии первого рода, т. е. переданные пользователем, утратившим привилегии, не отменяются. Поэтому администратор пользователей должен очень осторожно выдавать право предоставления привилегий. Администратор БД распределяет функции создания и манипулирования между пользователями в рамках отдельной роли.

Пусть в некоторой информационной системе должны работать одновременно 100 бухгалтеров. Каждый из них может модифицировать БД «Orders».

Create role accouter – создать роль «бухгалтер».

Grant update, insert, delete, select on Orders to accounter – выделить роли «бухгалтер» полномочия на модификацию БД «Orders».

Grant accounter to user1, ... user100.

Общий вид операторов создания роли и выделения полномочий:

create role <имя роли>

grant <список функций> on <имя БД> to <имя роли, имя пользователя> with <параметры>

grant <список ролей> to <список ролей, пользователей>

Список функций может содержать как функции манипулирования данными, так и функции создания данных. Специальное значение списка ALL выделяет полномочия на все допустимые действия. В качестве параметров операнда GRANT могут использоваться следующие: Admin option — опции адмистратора.

Пример:

grant creat table to user2 with admin option.

Сеанс работы любого клиента определяется набором установленных ролей. Формат команды установления роли: set role <список ролей>
Пример:
set role accounter.

Значение списка ролей ALL выделяет все определенные на данный момент роли. Set role all except <список ролей> — установить все роли, исключая перечисленные. Оператор уничтожения роли:
Revoke <список ролей> from <список пользователей>.
Пример:
Revoke accounter from user1, user3, user57.

14. Динамический SQL

14.1. Области дескриптора

Динамическим **SQL** называют совокупность операторов и структур данных, которые позволяют сформировать запрос в ходе выполнения приложения, т. е. динамический **SQL** позволяет приложению модифицировать свой собственный исходный код. Для подготовки динамического оператора можно использовать средство обработки строк и необходимые специальные операторы, превращающие строку в оператор приложения. Самым сложным моментом выполнения динамического оператора является направление потоков входных и выходных параметров. Для корректного обслуживания используется специальный описатель (дескриптор) каждого оператора, т. е. дескриптор — это внутренняя структура данных.

Дескриптор представляет собой список элементов, длина списка указывается в заголовке, поле с именем count, каждый элемент создает набор полей, определяющих имя, тип и значение параметра.

Type— задает тип параметра
Length — длина
Octet_length — длина в байтах
Jcale— масштаб
Precision — точность
Nullable — допустимость Null
Indicator — 1 - Null
Name — имя параметра
Data — значение параметра

Поля, задающие контекст действия:
Character_set_catalog
Character_set_schema
В составе полей дескриптора можно выделить 4 группы полей:
1) определители типа;
2) определители значения;
3) определители имени;
4) определители контекста действия.

14.2. Операторы подготовки и выполнения динамических операторов

Оператор подготовки динамического оператора

Подготовка содержимого оператора выполняется средствами обработки строк, а оператор подготовки присваивает динамическому оператору имя и связывает со строкой содержащей его описание.

```
St_t1 = 'select *  from peoples
      Where city = Uljanovsk;
Prepare <имя>
From <имя переменной>
Prepare st_d
From  St_t1.
```

Подготовленный оператор рассматривается сервером данных как объект схемы данного приложения. Такой объект активизируется с помощью оператора – выполнить.

Оператор выполнения динамического SQL

Динамический **SQL** оператор может либо использовать параметры, либо быть независимым по входу и выходу. Для выполнения динамического оператора без парамеров используется следующий код:

```
Execute immediate <имя>.
```

Для обработки потоков входных и выходных параметров используюся операторы выполнения в следующей форме:

```
Execute <имя>
Into (<параметр>,…)/<имя_дескриптора>
Using (<параметр>,…)/<имя_дескриптора>.
```

St_t1= 'select* from peaples
Where (city = Ulyanovsk).

При подготовке оператора можно использовать вопросительный знак как выражение неопределенного неозначенного параметра.

Оператор удаления подготовленного динамического оператора

deallocate prepare <имя>;

Операторы обработки дексрипторов

Дескриптор хоть и является внутренней структурой данных, тем не менее формируется программистом, использующий динамический SQL. В частности, программист определяет длину списка дескриптора и означивает поля элементов.

Оператор размещения дескриптора

allocate descriptor <имя>
with max <число элеменов>.

Оператор уничтожения дескриптора

deallocate descriptor <имя>.

Оператор означивания полей элементов дескриптора

set descriptor <имя>
value = <номер элемента>
(<имя пол>=<значение>);
set descriptor D1
value = 3
(name = : My_name,
data = : My_ data,
type = string).

Оператор чтения полей дескриптора

get descriptor <имя>
value = <номер элемента>
(<переменная>=<имя поля>);

get descriptor D1
value <номер>(My_string = data).

Оператор заполнения дескриптора по подготовленному динамическому оператору

Если с помощью оператора prepare подготовлен динамический оператор, то дескриптор для него СД может заполнить автоматически, выполняя оператор.
Describe [inpat/outpat] <имя оператора>
Using SQL descriptor <имя>.
Оператор Describe формирует в дескрипторе список входных и выходных параметров динамического оператора.

14.4. Оператор размещения динамического курсора

allocate [insensitive] scroll <имя>
for <имя оператора>;

На месте имени курсора в операторе allocate может стоять переменная. Это позволяет разработать приложение, в котором используется произвольное количество курсоров, причем, содержимое каждого из них определяется динамически.

14.5. Оператор закрытия

close <имя>

14.6. Оператор объявления курсора

Курсор представляет собой переменную, поэтому статический курсор и динамический курсор может быть объявлен.

Declare [insensitive/ scroll] cursor <имя>
For [select...../<динамический оператор>]
[for read only/update].

Особенностью оператора Declare по сравнению с оператором allocate служит запрещение использовать переменную в качестве курсора, т. е. объявить можно только один курсор с определенным именем.

14.7 Оператор — открыть курсор

open cursor <имя>
[using SQL descriptor <имя>].

Оператор open выполняет запрос, связанный с данным курсором. Если курсор связан с динамическим оператором, то для его корректного выполнения может потребоваться дескриптор, который указывается в предложении using.

Глава 4. Упражнения и задачи по использованию языка SQL

1. Виды демонстрационных реляционных таблиц. Описание демонстрационных таблиц и данные

Задачи сформулированы для системы «Управление персоналом». Созданы три рабочие таблицы. Первая из них – рабочая таблица «Служащие». Таблица включает в себя следующие поля: табельный номер (empno), ФИО (ename), должность (job), табельный номер начальника (mgr), дата приема на работу (hiredate), оклад (sal), комиссионные (sal), номер отдела (deptno).

Таблица EMP

Name	Type
EMPNO	NOT NULL NUMBER (4)
ENAME	VARCHAR2(10)
JOB	VARCHAR2(9)
MGR	NUMBER (4)
HIREDATE	DATE
SAL	NUMBER (7,2)
COMM	NUMBER (7,2)
DEPTNO	NUMBER (2)

Приведем пример наполнения таблицы.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	13-JUN-83	800		20
7499	ALLEN	SALESMAN	7698	15-AUG-83	1600	300	30
7521	WARD	SALESMAN	7698	26-MAR-84	1250	500	30
7566	JONES	MANAGER	7839	31-OCT-83	2975		20

Таблица «Отделы» (dept) состоит из столбцов: номер отдела (deptno), наименование отдела (dname), город расположения (loc). Таблица может быть наполнена следующим образом.

Таблица DEPT

Name	Type
DEPTNO NOT NULL	NUMBER (2)
DNAME	VARCHAR2 (14)
LOC	VARCHAR2 (13)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Таблица «Разряды и тарифы оплаты» содержит три столбца: номер разряда (grade), нижняя граница оплаты по заданному разряду (LOSAL), верхняя граница оплаты по заданному разряду (HISAL). Ниже приведено наполнение таблицы «Вилки окладов».

Таблица SALGRADE

Name	Type
GRADE	NOT NULL NUMBER
LOSAL	NOT NULL NUMBER
HISAL	NOT NULL NUMBER

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

Задачи построены на основе обучающих курсов Oracle Corp.

2. Упражнения на выборку данных из различных таблиц

Ответьте на следующие вопросы:
1. Команды SQL всегда хранятся в буфере? (Да/Нет).
2. Команды SQL*Plus используются для выборки данных? (Да/Нет).
3. Можно задавать сортировку по столбцу, который не указан в списке выборки? (Да/Нет).
4. Следующая команда будет выполнена успешно? (Да/Нет).

```
Select      Ename, Job, Sal Salary
From        Emp
Where       Ename = 'FORD'.
```

5. Следующая команда будет выполнена успешно? (Да/Нет).

```
Select      *
From        Emp
Where Sal * 12 = 9600.
```

6. Следующая команда содержит ряд ошибок. Попробуйте найти их.

```
SELECT  EMNO, ENAME, SAL* 12 ANNUAL SALARY
FROM    EMP
WHERE   SALARY > 3000
AND     HIREDATE LIKE % 84.
```

После того, как вы проверили вашу память, войдите в SQL*Plus. Лектор должен был объяснить вам, как это сделать.

7. Извлеките всю информацию из таблицы SALGRADE (вилки окладов):

Комментарий: GRADE – категория, LOSAL (Lower Salary) и HISAL (High Salary) – минимальный и максимальный оклады для данной категории.

8. Извлеките всю информацию из таблицы EMP (сотрудники).

Комментарий: EMPNO, ENAME и JOB — табельный номер, фамилия и должность сотрудника, MGR — табельный номер его руководителя, HIREDATE — дата приема на работу, SAL и COMM — оклад и комиссионное вознаграждение, DEPTNO — номер отдела.

9. Найдите всех сотрудников с окладом в интервале от 1600 до 3000 включительно.

ENAME	DEPTNO	SAL
ALLEN	30	1600.00
JONES	20	2975.00
BLAKE	30	2850.00
CLARK	10	2450.00
SCOTT	20	3000.00
FORD	20	3000.00

10. Выдайте номера (DEPTNO) и названия (DNAME) всех отделов в алфавитном порядке.

DEPTNO DNAME

10 ACCOUNTING
40 OPERATIONS
20 RESEARCH
30 SALES

11. Выдайте названия всех должностей (JOB) без повторов. Измените на обратный порядок вывода строк.

JOB

SALESMAN
PRESIDENT
MANAGER
CLERK
ANALYST

12. Извлеките полную информацию о всех сотрудниках в отделах 10 и 20 так, чтобы их фамилии шли в алфавитном порядке.

13. Выведите фамилии (ENAME) и должности (JOB) всех клерков в отделе 20.

ENAME JOB

SMITH CLERK
ADAMS CLERK

14. Найдите всех сотрудников, чьи фамилии (ENAME) содержат сочетания «TH»
или «LL».

ENAME

SMITH
ALLEN

MILLER

15. Извлеките следующие данные о сотрудниках, у которых есть руководитель:

ENAME JOB MGR SAL

SMITH	CLERK	7902	800.00
ALLEN	SALESMAN	7698	1600.00
WARD	SALESMAN	7698	1250.00
JONES	MANAGER	7839	2975.00
MARTIN	SALESMAN	7698	1250.00
BLAKE	MANAGER	7839	2850.00
CLARK	MANAGER	7839	2450.00
SCOTT	ANALYST	7566	3000.00
TURNER	SALESMAN	7698	1500.00
ADAMS	CLERK	7788	1100.00
JAMES	CLERK	7698	950.00
FORD	ANALYST	7566	3000.00
MILLER	CLERK	7782	1300.00

16. Выведите фамилии (ENAME) и полный годовой доход (REMUNERATION, т. е. зарплата плюс комиссионные) всех сотрудников.

ENAME REMUNERATION

SMITH	9600.00
ALLEN	19500.00
WARD	15500.00
JONES	35700.00
MARTIN	16400.00
BLAKE	34200.00
CLARK	29400.00
SCOTT	36000.00
KING	60000.00
TURNER	18000.00
ADAMS	13200.00
JAMES	11400.00
FORD	36000.00
MILLER	15600.00

17. Найдите всех сотрудников, которые были приняты на работу в 1983 году.

EMAME	DEPTNO	HIREDATE
SMITH	20	13-JUN-83
ALLEN	30	15-AUG-83
JONES	20	31-OCT-83
MARTIN	30	05-DEC-83
FORD	20	05-DEC-83
MILLER	10	21-NOV-83

18. Выдайте фамилии (ENAME), годовую зарплату (ANNUAL_SAL) и комиссионное вознаграждение (COMM) всех продавцов (SALESMAN), у которых оклад превышает их комиссионные за месяц. Расположите их в порядке убывания окладов.

ENAME	ANNUAL_SAL	COMM
ALLEN	19200	300.00
TURNER	18000	.00
WARD	15000	500.00

Дополнительное задание: выдайте данные, как показано ниже.

Who, what and when

SMITH HAS HELD THE POSITION OF CLERK IN DEPT 20 SINCE 13-JUN-83
ALLEN HAS HELD THE POSITION OF SALESMAN IN DEPT 30 SINCE 15-AUG-83
WARD HAS HELD THE POSITION OF SALESMAN IN DEPT 30 SINCE 26-MAR-84
JONES HAS HELD THE POSITION OF MANAGER IN DEPT 20 SINCE 31-OCT-83
MARTIN HAS HELD THE POSITION OF SALESMAN IN DEPT 30 SINCE 05-DEC-83
BLAKE HAS HELD THE POSITION OF MANAGER IN DEPT 30 SINCE 11-JUN-84
CLARK HAS HELD THE POSITION OF MANAGER IN DEPT 10 SINCE 14-MAY-84

SCOTT HAS HELD THE POSITION OF ANALYST IN DEPT 20 SINCE 05-MAR-84
KING HAS HELD THE POSITION OF PRESIDENT IN DEPT 10 SINCE 09-JUL-84
TURNER HAS HELD THE POSITION OF SALESMAN IN DEPT 30 SINCE 04-JUN-84
ADAMS HAS HELD THE POSITION OF CLERK IN DEPT 20 SINCE 04-JUN-84
JAMES HAS HELD THE POSITION OF CLERK IN DEPT 30 SINCE 23-JUL-84
FORD HAS HELD THE POSITION OF ANALYST IN DEPT 20 SINCE 05-DEC-83
MILLER HAS HELD THE POSITION OF CLERK IN DEPT 10 SINCE 21-NOV-83

3. Упражнения по оперативному вводу SQL-оператора

1. Ответьте на следующие вопросы:
Приглашение для ввода значения переменной подстановки с одним амперсандом выдается только один раз?
Переменная подстановки с двумя амперсандами сохраняет свое значение на время текущего сеанса?
Используется ли команда DEFINE для просмотра существующих переменных?
Команда ACCEPT – это команда языка SQL?
2. Напишите команду, которая приглашает пользователя ввести две произвольные даты и выбирает всех сотрудников, принятых на работу в период между этими датами. Выполните запрос дважды.
Модифицируйте запрос, используя переменные с двойным амперсандом. Выполните запрос несколько раз и объясните разницу.
3. Составьте запрос, в котором название должности может быть любым. Протестируйте запрос, выполнив его несколько раз.

ENAME	JOB	SAL	MGR	DEPTNO
SCOTT	ANALYST	3000.00	756620	
FORD	ANALYST	3000.00	756620	

Дополнительное упражнение:
Определите переменную, представляющую выражение для вычисления полных годовых начислений сотрудникам. Используйте эту

переменную в команде, которая находит всех сотрудников, чьи годовые начисления не меньше $30000.

ENAME	SAL* 12+NVL(COMM,0)
JONES	35700
BLAKE	34200
SCOTT	36000
KING	60000
FORD	36000

4. Упражнения на запросы, в которых требуется использование функций, работающих с числами, символами и календарными датами; на использование конкатенации вместе с функциями

1. Ответьте на следующие вопросы:

Можно ли использовать все арифметические операции для календарных дат?

Как называется псевдостолбец, в котором хранится текущая дата?

2. Выведите список, содержащий фамилию каждого сотрудника и его оклад, увеличенный на 15% и округленный до целого числа долларов.

DEPTNO	ENAME	PCTSAL
20	SMITH	920
30	ALLEN	1840
30	WARD	1438
20	JONES	3421
30	MARTIN	1438
30	BLAKE	3278
10	CLARK	2818
20	SCOTT	3450
10	KING	5750
30	TURNER	1725
20	ADAMS	1265
30	JAMES	1093
20	FORD	3450
10	MILLER	1495

3. Получите следующий результат:

EMPLOYE_AND_JOB

SMITH CLERK
ALLEN SALESMAN
WARD SALESMAN
JONES MANAGER
MARTIN SALESMAN
BLAKE MANAGER
CLARK MANAGER
SCOTT ANALYST
KING PRESIDENT
TURNER SALESMAN
ADAMS CLERK
JAMES CLERK
FORD ANALYST
MILLER CLERK

4. Получите следующий результат:

EMPLOYEE

SMITH (Clerk)
ALLEN (Salesman)
WARD (Salesman)
JONES (Manager)
MARTIN (Salesman)
BLAKE (Manager)
CLARK (Manager)
SCOTT (Analyst)
KING (President)
TURNER (Salesman)
ADAMS (Clerk)
JAMES (Clerk)
FORD (Analyst)
MILLER (Clerk)

5. Напишите команду SELECT для отбора сотрудников по должности, которая будет задаваться вами во время выполнения. Сделайте это так,

чтобы должность можно было вводить как строчными, так и прописными буквами.

```
Enter value for job:  clerk
old 3: WHERE UPPER (JOB) = UPPER ('&JOB')
new 3: WHERE UPPER(JOB) .= UPPER( 'clerk' )
```

6. Оказывается, в отделе с номером 30 не все продавцы — мужчины, и поэтому название SALESMAN для их должности не всегда подходит. Получите следующий результат:

ENAME	DEPTNO	JOB
SMITH	20	Clerk
ALLEN	30	Salesperson
WARD	30	Salesperson
JONES	20	Manager
MARTIN	30	Salesperson
BLAKE	30	Manager
CLARK	10	Manager
SCOTT	20	Analyst
KING	10	President
TURNER	30	Salesperson
ADAMS	20	Clerk
JAMES	30	Clerk
FORD	20	Analyst
MILLER	10	Clerk

7. Выведите фамилию каждого сотрудника вместе с датой приема на работу и датой назначения нового оклада. Считайте, что новый оклад назначается через год после приема на работу. Расположите данные в порядке возрастания даты назначения нового оклада.

ENAME	HIREDATE	REVIEW
SMITH	13-JUN-83	13-JUN-84
ALLEN	15-AUG-83	15-AUG-84
JONES	31-OCT-83	31-OCT-84
MILLER	21-NOV-83	21-NOV--84
MARTIN	05-DEC-83	05-DEC-84
FORD	05-DEC-83	05-DEC-84

SCOTT 05-MAR-84 05-MAR-85
WARD 26-MAR-84 26-MAR-85
CLARK 14-MAY-84 14-MAY-85
TURNER 04-JUN-84 04-JUN-85
ADAMS 04-JUN-84 04-JUN-85
BLAKE 11-JUN.-84 11-JUN-85
KING 09-JUL-84 09-JUL-85
JAMES 23-JUL-84 23-JUL-85

Дополнительные упражнения
8. Создайте запрос, с помощью которого можно определить стаж работы любого сотрудника. Можете использовать команду DEFINE, чтобы избежать повторного ввода функции.

ENAME LENGTH OF SERVICE

KING 10 YEARS 10 MONTHS

9. Сотрудники, принятые на работу до 15-го числа (включительно) каждого месяца, получают первую зарплату в последнюю пятницу того же месяца. Сотрудники, принятые после 15-го числа, получают первую зарплату в последнюю пятницу следующего месяца. Выведите список, содержащий фамилии сотрудников, даты их приема на работу и даты первой зарплаты, упорядочив список по дате приема.

ENAME HIREDATE PAYDAY

SMITH 13-JUN-83 24-JUN-83
ALLEN 15-AUG-83 26-AUG-83
JONES 31-OCT-83 25-NOV-83
MILLER 21-NOV-83 30-DEC-83
MARTIN 05-DEC-83 30-DEC-83
FORD 05-DEC-83 30-DEC-83
SCOTT 05-MAR-84 30-MAR-84
WARD 26-MAR-84 27-APR-84
CLARK 14-MAY-84 25-MAY-84
TURNER 04-JUN-84 29-JUN-84
ADAMS 04-JUN-84 29-JUN-84
BLAKE 11-JUN-84 29-JUN-84
KING 09-JUL-84 27-JUL-84
JAMES 23-JUL-84 31-AUG-84

5. Упражнения на использование функции TO_CHAR для изменения формата вывода календарной даты; на использование функций работы с датами

1. Выведите фамилии всех сотрудников отдела 20 и даты их приема на работу. Не забудьте указать псевдоним «DATE_HIRED» после соответствующего выражения, иначе при выводе столбца произойдет самопроизвольный перенос (по умолчанию, ширина символьного столбца составляет 80 знаков).

ENAME	DATE_HIRED
SMITH	June, Thirteenth 1983
JONES	October, Thirty First 1983
SCOTT	March, Fifth 1984
ADAMS	June, Fourth 1984
FORD	December, Fifth 1983

2. Выведите список, содержащий фамилию каждого сотрудника и его оклад, если он превышает 1500. Если оклад в точности равен 1500, вместо оклада укажите «On Target», если он меньше 1500, укажите «Below 1500».

ENAME	SALARY
ADAMS	Below 1500
ALLEN	1600
BLAKE	2850
CLARK	2450
FORD	3000
JAMES	Below 1500
JONES	2975
KING	5000
MARTIN	Below 1500
MILLER	Below 1500
SCOTT	3000
SMITH	Below 1500
TURNER	On Target

WARD Below 1500

Дополнительные упражнения

3. Составьте запрос, который возвращает день недели (DAY) для любой даты, введенной в формате DD.MM.YY.

Enter value for anydate: 15.04.95

DAY

Saturday

4. Для заданной символьной строки формата «nn/nn» проверьте, что первая и последняя пара символов являются числами и они разделены наклонной чертой «/». Напечатайте «YES», если это действительно так, и «NO» - в противном случае. Для тестирования вашей команды предложите ей символьные строки «12/34», «01/1a», «99\88».

SQL> ACCEPT STRING CHAR PROMPT 'Enter string in nn/nn format =>

VALUE VALID?

12/34 YES

6. Упражнения на использование групповых функций и выборки данных с разбиением на группы

1. Ответьте на следующие вопросы:
Групповые функции обрабатывают множество строк и возвращают один результат для этого множества?
Групповые функции обрабатывают все значения, включая NULL— значения?
Чтобы исключить строки из групповых вычислений, используется фраза HAVING?
Чтобы включить группы строк в результат запроса, используется фраза HAVING?
Уровень вложенности групповых функций может быть произвольным?
2 . Найдите оклад, минимальный среди всех сотрудников.

MINIMUM

800.00

3. Определите максимальный, минимальный и средний оклад по всем сотрудникам.

MAX (SAL) MIN (SAL) AVG (SAL)

5000.00 800.00 2073.21

4. Выведите максимальный и минимальный оклад по каждой должности.

JOB	MAXIMUM	MINIMUM
ANALYST	3000	3000
CLERK	1300	800
MANAGER	2975	2450
PRESIDENT	5000	5000
SALESMAN	1600	1250

5. Выведите число руководителей, не перечисляя их поименно.

MANAGERS 3

6. Найдите средний оклад и средние начисления за год по каждой должности, не забывая, что продавцы получают комиссионные.

JOB	AVSAL	AVCOMP
ANALYST	3000	36000
CLERK	1037.5	12450
MANAGER	2758.33333	33100
PRESIDENT	5000	60000
SALESMAN	1400	17350

7. Определите разницу между максимальным и минимальным окладами.

DIFFERENCE
4200

8. Найдите все отделы, где работает больше трех сотрудников.

DEPTNO	COUNT (*)
20	5
6	

Дополнительные задания

9. Проверьте, действительно ли среди табельных номеров сотрудников (EMPNO) нет повторяющихся.

10. Для каждого руководителя найдите оклад, минимальный среди всех сотрудников, которыми он руководит. Исключите все группы, где минимальный оклад меньше 1000. Выходные данные отсортируйте по окладу.

MGR	MIN(SAL)
1100	
1300	
2450	
3000	
5000	

7. Упражнения в выборке данных из нескольких таблиц

1. Ответьте на следующие вопросы:

Как называется тип соединения, в котором задается совпадение значений столбцов из разных таблиц?

Если команда SELECT обращается к шести таблицам, то сколько надо задать условий соединения? Может ли их быть больше?

Как обозначается операция внешнего соединения?

Операция внешнего соединения указывается для той таблицы, в которой недостаточно строк? (Да / Нет).

Операция внешнего соединения может быть указана с обеих сторон условия соединения? (Да / Нет).

2. Выведите список фамилий сотрудников вместе с названиями отделов, где они работают, упорядочив список по названиям отделов.

ENAME	DNAME
CLARK	ACCOUNTING
MILLER	ACCOUNTING
KING	ACCOUNTING
SMITH	RESEARCH
SCOTT	RESEARCH
JONES	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
ALLEN	SALES
BLAKE	SALES
TURNER	SALES
JAMES	SALES
MARTIN	SALES
WARD	SALES

3. Выведите список фамилий сотрудников вместе с номерами и названиями отделов, где они работают.

ENAME	DEPTNO	DNAME
CLARK	10	ACCOUNTING
MILLER	10	ACCOUNTING
KING	10	ACCOUNTING
SMITH	20	RESEARCH
SCOTT	20	RESEARCH

JONES	20	RESEARCH
ADAMS	20	RESEARCH
FORD	20	RESEARCH
ALLEN	30	SALES
BLAKE	30	SALES
TURNER	30	SALES
JAMES	30	SALES
MARTIN	30	SALES
WARD	30	SALES

4. Для каждого сотрудника с окладом выше 1500 выведите фамилию, город и название отдела, где он работает.

ENAME	LOCATIC	DNAME
CLARK	NEW YORK	ACCOUNTING
KING	NEW YORK	ACCOUNTING
SCOTT	DALLAS	RESEARCH
JONES	DALLAS	RESEARCH
FORD	DALLAS	RESEARCH
ALLEN	CHICAGO	SALES
BLAKE	CHICAGO	SALES

5. Получите следующий список с категориями оплаты труда сотрудников.

ENAME	JOB	SAL	GRADE
SMITH	CLERK	800.00	1
ADAMS	CLERK	1100.00	1
JAMES	CLERK	950.00	1
WARD	SALESMAN	1,250.00	2
MARTIN	SALESMAN	1,250.00	2
MILLER	CLERK	1,300.00	2
ALLEN	SALESMAN	1,600.00	3
TURNER	SALESMAN	1,500.00	3
JONES	MANAGER	2,975.00	4
BLAKE	MANAGER	2,850.00	4
CLARK	MANAGER	2,450.00	4
SCOTT	ANALYST	3,000.00	4
FORD	ANALYST	3,000.00	4
KING	PRESIDENT	5,000.00	5

6. Используя запрос, созданный в упражнении 5, выведите список только тех сотрудников, которые оплачиваются по категории 3.

ENAME	JOB	SAL	GRADE
ALLEN	SALESMAN	1600.00	3
TURNER	SALESMAN	1500.00	3

7. Получите следующую информацию для всех сотрудников, работающих в Далласе.

ENAME	SAL	LOCATION
SMITH	800.00	DALLAS
SCOTT	3,000.00	DALLAS
JONES	2,975.00	DALLAS
ADAMS	1,100.00	DALLAS
FORD	3,000.00	DALLAS

8. Найдите отдел, в котором нет сотрудников.

DEPTNO	DNAME
OPERATIONS	

9. Составьте список, содержащий для каждого сотрудника фамилию, табельный номер, а также табельный номер и фамилию его руководителя.

EMPNO	ENAME	MGRNO	MGR_NAME
7369	SMITH	7902	FORD
7499	ALLEN	7698	BLAKE
7521	WARD	7698	BLAKE
7566	JONES	7839	KING
7654	MARTIN	7698	BLAKE
7698	BLAKE	7839	KING
7782	CLARK	7839	KING
7788	SCOTT	7566	JONES
7844	TURNER	7698	BLAKE
7876	ADAMS	7788	SCOTT
7900	JAMES	7698	BLAKE

7902 FORD 7566 JONES
7934 MILLER 7782 CLARK

10. Измените запрос, созданный в упражнении 9 так, чтобы в список был включен KING, у которого нет руководителя.
11. Найдите те должности, на которые были приняты сотрудники как в первом полугодии 1983 года, так и в первом полугодии 1984 года.

JOB

CLERK

Дополнительные упражнения
12. Для всех сотрудников, кроме клерков, выведите фамилию, должность, оклад, категорию оплаты и название отдела. Расположите информацию в порядке убывания оклада.

ENAME JOB SAL GRADE DNAME

KING PRESIDENT 5000.00 5 ACCOUNTING
FORD ANALYST 3000.00 4 RESEARCH
SCOTT ANALYST 3000.00 4 RESEARCH
JONES MANAGER 2975.00 4 RESEARCH
BLAKE MANAGER 2850.00 4 SALES
CLARK MANAGER 2450.00 4 ACCOUNTING
ALLEN SALESMAN 1600.00 3 SALES
TURNER SALESMAN 1500.00 3 SALES
MARTIN SALESMAN 1250.00 2 SALES
WARD SALESMAN 1250.00 2 SALES

13. Выберите следующую информацию о сотрудниках, которые получают $36000 в год или являются клерками.

ENAME JOB ANNUAL_SAL DEPTNO DNAME GRADE

FORD ANALYST 36000 20 RESEARCH 4
SCOTT ANALYST 36000 20 RESEARCH 4
MILLER CLERK 15600 10 ACCOUNTING 2
JAMES CLERK 11400 30 SALES 1
ADAMS CLERK 13200 20 RESEARCH 1
SMITH CLERK 9600 20 RESEARCH 1

14. Составьте список всех сотрудников, которые приняты на работу раньше их руководителей.

ENAME	HIREDATE	MANAGER	HIREDATE
ALLEN	15-AUG-83	BLAKE	11-JUN-84
BLAKE	11-JUN-84	KING	09-JUL-84
CLARK	14-MAY-84	KING	09-JUL-84
JONES	31-OCT-83	KING	09-JUL-84
MARTIN	05-DEC-83	BLAKE	11-JUN-84
MILLER	21-NOV-83	CLARK	14-MAY-84
SMITH	13-JUN-83	FORD	05-DEC-83
TURNER	04-JUN-84	BLAKE	11-JUN-84
WARD	26-MAR-84	BLAKE	11-JUN-84

15. Найдите другое решение упражнения 8

DEPTNO	DNAME
40	OPERATIONS

8. Упражнения в составлении сложных запросов, включающих вложенные и коррелированные подзапросы

1. Ответьте на следующие вопросы:
Какой запрос выполняется первым в командах, содержащих подзапросы? Сколько раз он выполняется?
Можно ли использовать операцию «равно», если внутренний запрос возвращает несколько строк? (Да / Нет).
Если «НЕТ», то почему и какие операции можно использовать?
Если «ДА», то почему?
Какую операцию можно использовать для более эффективного выполнения коррелированного подзапроса?
2. Найдите сотрудников, чей оклад является максимальным для той должности, которую они занимают. Отсортируйте результат в порядке убывания оклада.

JOB	ENAME	SAL
PRESIDENT	KING	5000.00
ANALYST	SCOTT	3000.00

ANALYST	FORD	3000.00
MANAGER	JONES	2975.00
SALESMAN	ALLEN	1600.00
CLERK	MILLER	1300.00

3. Найдите сотрудников, чей оклад является минимальным для той должности, которую они занимают. Отсортируйте результат в порядке возрастания оклада.

ENAME	JOB	SAL
SMITH	CLERK	800.00
WARD	SALESMAN	1250.00
MARTIN	SALESMAN	1250.00
CLARK	MANAGER	2450.00
SCOTT	ANALYST	3000.00
FORD	ANALYST	3000.00
KING	PRESIDENT	5000.00

4. Найдите в каждом отделе сотрудников, принятых на работу последними. Отсортируйте результат по дате приема.

DEPTNO	ENAME	HIREDATE
20	ADAMS	04-JUN-84
10	KING	09-JUL-84
30	JAMES	23-JUL-84

5. Выведите указанную информацию о сотрудниках, у которых оклад меньше среднего по их отделу. Отсортируйте результат по номерам отделов.

ENAME	SALARY	DEPTNO
CLARK	2450	10
MILLER	1300	10
SMITH	800	20
ADAMS	1100	20
WARD	1250	30
JAMES	950	30
TURNER	1500	30
MARTIN	1250	30

6. Перечислите отделы, в которых нет сотрудников (на этот раз используйте подзапрос).

DEPTNO DNAME

40 OPERATIONS

7. Получите следующую информацию об отделе с самым высоким суммарным объемом годовых начислений.

DEPTNO COMPENSATION

130500

Дополнительные упражнения

8. Выведите фамилии и оклады трех наиболее высоко оплачиваемых сотрудников.

ENAME SAL

SCOTT 3000.00
KING 5000.00
FORD 3000.00

9. В каком году в фирму было принято больше всего людей? Выведите этот год и число принятых сотрудников.

YEAR NUMBER_OF_EMPS

8

10. Модифицируйте запрос упражнения 5 так, чтобы вывести также средний оклад по каждому отделу.

ENAME SALARY DEPTNO DEPT AVG
MILLER 1300 10 2916.6667
CLARK 2450 10 2916.6667
SMITH 800 20 2157
ADAMS 1100 20 2157
JAMES 950 30 1566.6667
MARTIN 1250 30 1566.6667
WARD 1250 30 1566.6667
TURNER 1500 30 1566.6667

11. Составьте запрос, который помечает звездочкой строку, соответствующую сотруднику, принятому в фирму позже всех. Выведите столбцы ENAME,
HIREDATE, а также столбец MAXDATE для звездочки.

ENAME HIREDATE M

ADAMS 04-JUN-84
ALLEN 15-AUG-83
BLAKE 11-JUN-84
CLARK 14-MAY-84
FORD 05-DEC-83
JAMES 23-JUL-84
JONES 31-OCT-83
KING 09-JUL-84
MARTIN 05-DEC-83
MILLER 21-NOV-83
SCOTT 05-MAR-84
SMITH 13-JUN-83
TURNER 04-JUN-84
WARD 26-MAR-84

9. Упражнения в обходе дерева

1. Составьте следующий список, включив в него всех сотрудников. Список должен начинаться с сотрудника, который находится на вершине иерархического дерева компании. Фраза START WITH не должна определять какого-то конкретного сотрудника.

EMPLOYEE_NUMBER DEPTNO EMPNO ENAME JOB SAL

7839 10 7839 KING PRESIDENT 5,000.00
7566 20 7566 JONES MANAGER 2,975.00
7788 20 7788 SCOTT ANALYST 3,000.00

10. Упражнения на создание простого отчета в форме таблицы с разбивкой на секции и итогами; все команды должны находиться в командном файле

1. Для каждой из следующих команд определите, является ли она командой SQL или командой SQL*Plus, отметив свой выбор в соответствующей клетке.

Команда	SQL	SQL*Plus
DESCRIBE		
SPOOL		
TTITLE		
SELECT		
CONNECT		
ACCEPT		
SET		
INSERT		
DEFINE		

2. В приведенной ниже таблице слева указаны задачи, а справа — команды, при помощи которых их можно решить, однако команды расставлены неправильно. Переставьте команды так, чтобы они соответствовали задачам.

Задача	Команда
Получить определение таблицы	SET LINES
Отменить вывод сообщения о числе выбранных строк по окончании запроса	SPOOL
Задать размер строки в символах	DESCRIBE
Задать верхний колонтитул каждой страницы отчета	SET FEED OFF
Изменить формат вывода столбца	TTITLE
Выйти в операционную систему, не прерывая сеанса SQL*Plus	PROMPT
Вывести сообщение для пользователя при выполнении командного файла	COLUMN
Вставить комментарии в командный файл	HOST
Направить выходной поток в файл	REM

2. Используя командный файл, создайте следующий отчет.
Вторая страница отчета должна содержать информацию об отделе 20, а третья — об отделе 30. К сожалению, на этой странице не так много места, чтобы показать весь отчет полностью !

Номер страницы: 1 Дата выдачи: 23/01/96

Список сотрудников
по отделам

Отдел	Фамилия сотрудника	Дата приема	Месячный оклад в ($)	Годовой оклад в ($)	Общий доход в ($)

Номер страницы: 2 . . .
Номер страницы: 3 . . .

Номер страницы: 4 Дата выдачи: 23/01/96

Список сотрудников
по отделам

Отдел	Фамилия сотрудника	Дата приема	Месячный оклад в ($)	Годовой оклад в ($)	Общий доход в ($)
Итого			29,025	3,48,300	3,50,500

11. Упражнение на создание таблиц

1. Правильно ли сформированы следующие имена таблиц?

Имя таблицы	Правильно	Неправильно
T_3000		
EMP DATA		
EMPLOYEE_INFORMATION		
1995-EXPENSES		

2. Создайте таблицу COMPANY_CARS, используя приведенную ниже спецификацию. Определяя имена для правил целостности, вы можете выбирать их по своему усмотрению.

Имя столбца	Тип данных и размер	Правила
CHASSIS_NUMBER	VARCHAR2 20	Не может быть пустым, первичный ключ
MAKE	VARCHAR2 15	Только большие буквы
MODEL	VARCHAR2 15	Только большие буквы
CUB-CAP	NUMBER 4	
COLOR	VARCHAR2 10	
DOR	DATE	
REG-NO	VARRCHAR2 10	Не может быть пустым, только Большие буквы
EMPNO	NUMBER 4	Не может быть пустым, внешний ключ к EMP.EMPNO
DEPTNO	NUMBER 2	Не может быть пустым, внешний ключ к DEPT.DEPTNO

Если после создания таблицы вы решили изменить ее определение, вам будет нужно сначала удалить таблицу, а затем повторить ее создание. Для удаления таблицы используется команда DROP TABLE.

```
DROP TABLE COMPANY_CARS;
```

12. Упражнения на просмотр структуры таблицы и правила целостности

1. Выберите все представления из словаря данных. Если вас не удовлетворяет внешний вид результата запроса, задайте форматы вывода с помощью команды COLUMN.

Если результат запроса содержит много строк, вы можете установить режим вывода с приостановками между страницами. Как это сделать?
Если возникнет необходимость отменить уже выполняющийся запрос, нажмите [CTRL} C.
2. Выберите имя таблицы, которую вы создали на предыдущем уроке. Существует, несколько таблиц словаря данных, в которых хранится нужная вам информация.

```
TABLE
NAME

COMPANY_CARS
```

Выберите все правила целостности, которые вы создали.

```
CONSTRAINT_NAME  C     TABLE_NAME

CC_CH_NO_NNULL           C     COMPANY_CARS
```

CC_REG_NO_NNULL	C	COMPANY_CARS
CC_EMPNO_NNULL	C	COMPANY_CARS
CC_DEPTNO_NNULL	C	COMPANY_CARS
CC_CH_NO_PK	P	COMPANY_CARS
CC_MAKE_CHKUPP	C	COMPANY_CARS
CC_MODEL_CHKUPP	C	COMPANY_CARS
CC_REG_NO_CHKUPP	C	COMPANY_CARS
CC_EMPNO_FK	R	COMPANY_CARS
CC_DEPTNO_FK	R	COMPANY_CARS

Сохраните запрос для дальнейшего использования.

13. Упражнения на создание новой таблицы на основе другой; добавление правила целостности в созданную таблицу; анализ информации из словаря базы данных

1. Ответьте на следующие вопросы.
Какая команда используется для добавления нового столбца в таблицу?
Какую позицию займет новый столбец среди существующих?
Как можно добиться, чтобы новый столбец занял выбранную вами позицию?
Можно ли модифицировать определение правила целостности?
Какую команду вы бы использовали для удаления правила целостности, определенного для таблицы?
2. Используя команду CREATE TABLE AS, создайте другую версию вашей таблицы COMPANY — CARS с именем CC2.
3. Выполните запрос к таблице USER-CONSTRAINTS словаря данных.
Сколько правил целостности связано с новой таблицей?

CONSTRAINT NAME	C	SEARCH_CONDITION
SYS_C0023089	C	CHASSIS_NUMBER IS NOT NULL
SYS_C0023090	C	REG_NO IS NOT NULL
SYS_C0023091	C	EMPNO IS NOT NULL
SYS_C0023092	C	DEPTNO IS NOT NULL

Какого типа эти правила целостности?
Замечание: формируемое по умолчанию имя правила целостности имеет вид: SYS — Cnnnnn.

4. При помощи правила на уровне таблицы определите для новой таблицы первичный ключ и сразу же сделайте его активным.
Выполните запрос к таблице USER_OBJECTS. 0братитевнимание, что среди ваших объектов, кроме созданной вами новой таблицы, появился также новый индекс.

Дополнительные упражнения
5. В таблицу COMPANY-CARS добавьте столбец типа LONG для хранения комментариев.
6. Модифицируйте столбец MAKE в таблице COMPANY-CARS, увеличив его размер на 10 символов.

14. Использование индексов

1. Будет ли использован индекс в следующих запросах?

Команда SELECT	*Индекс*	*Да/нет*
SELECT * FROM EMP WHERE ENAME = 'SMITH'	Неуникальный по ename	
SELECT REG-NO, CHASSIS-NUMBER FROM COMPANY-CARS WHERE DOR = '01-JAN-94'	Неуникальный по reg-no, уникальный по chassis-number	
SELECT * FROM EMP WHERE EMPNO = 7902	уникальный по первичному ключу empno	
SELECT ENAME, DNAME FROM EMP, DEPT WHERE EMP.DEPTNO = DEPT.DEPTNO	уникальный по dept.deptno, неуникальный по emp.deptno	

2. Создайте неуникальные индексы по столбцам внешних ключей для вашей таблицы company_cars. При создании таблицы вы должны были обеспечить проверку правильности значений для табельных номеров и номеров отделов, используя при этом правила целостности FOREIGN KEY со ссылкой на таблицы EMP и DEPT, соответственно.
3. Создайте неуникальный индекс по столбцу MODEL. Предположим, что пользователи постоянно интересуются моделями автомобилей, выполняя запросы типа «покажи всех, кто имеет FORD, VOLVO, CITROEN и т.д.» .
4. Выберите из соответствующего представления словаря данных информацию об индексах для таблицы company_cars. Имена ваших

объектов необязательно должны совпадать с теми, что приведены ниже.

INDEX_NAME	TABLE_NAME	UNIQUENES
CC_CH_NO_PK	COMPANY_CARS	UNIQUE
CC_DEPTNO_FK	COMPANY_CARS	NONUNIQUE
CC_EMPNO_FK	COMPANY_CARS	NONUNIQUE
CC_MODEL_INDEX	COMPANY_CARS	NONUNIQUE

Сколько всего индексов у вас получилось?

15. Упражнение на создание последовательности

1. Если бы вы создавали последовательность, которая должна использоваться для генерации значений первичного ключа таблицы, что вы выберете: CYCLE или NOCYCLE? И почему?
2. Создайте последовательность со следующими свойствами:
   Имя : CC_SEQUENCE.
   Начальное значение = 1.
   Шаг = 1.
   Максимальное значение = 5.

Проверьте существование этого объекта в словаре данных.

16. Упражнение на создание простого и сложного представления

1. Создайте простое представление с именем CC_VIEW на основе таблицы COMPANY_CARS, которое выбирает столбцы chassis_number, reg_no, empno и deptno.
Выполните SELECT из представления.
Сколько строк вы видите и почему?
2. Определите представление, из которого можно будет получить следующие ниже данные. Выполните запрос для проверки представления.

DEPTNO	AVERAGE	MAXIMUM	MINIMUM	SUM	NO_SALS	NO_COMMS
10	2916.66667	5000	1300	8750	3	0
20	2175	3000	800	10875	5	0
30	1566.66667	2850	950	9400	6	4

Подавите вывод дробной части в столбце AVERAGE.

DEPTNO	AVERAGE	MAXIMUM	MINIMUM	SUM	NO_SALS	NO_COMMS
10	2917	5000	1300	8750	3	0
20	2175	3000	800	10875	5	0
30	1567	2850	950	9400	6	4

3. Используя представление из задания 2, выведите следующую информацию. Номер сотрудника должен вводиться во время выполнения.

EMPNO	EMAME	JOB	SAL	HIREDATE	MINIMUM	MAXIMUM	AVERAGE
7902	FORD	ANALYST	3000.00	05-DEC-33	800	3000	2175

Выберите из словаря данных текст команды SELECT для вашего представления.

4. Создайте представление, позволяющее руководителю отдела SALES вывести список его подчиненных (не включая его самого).

EMPNO	NAME	JOB	HIREDATE	SALARY	COMMISSION
7499	ALLEN	SALESMAN	15-AUG-83	1600	300
7521	WARD	SALESMAN	26-MAR-84	1250	500
7654	MARTIN	SALESMAN	05-DEC-83	1250	1400
7844	TURNER	SALESMAN	04-JUN-84	1500	0
7900	JAMES	CLERK	23-JUL-84	950	

Если у вас осталось время, выполните еще несколько заданий.

5. Выведите список сотрудников, где для каждого указаны фамилия, должность, а также общее число сотрудников компании, занимающих эту должность. Результат отсортируйте по должности.

Name	Job	Total Employees With This Job
SCOTT	ANALYST	2
FORD	ANALYST	2
SMITH	CLERK	4
ADAMS	CLERK	4
MILLER	CLERK	4
JAMES	CLERK	4
JONES	MANAGER	3

CLARK MANAGER 3
BLAKE MANAGER 3
KING PRESIDENT 1
ALLEN SALESMAN 4
MARTIN SALESMAN 4
TURNER SALESMAN 4
WARD SALESMAN 4

Подавите вывод повторяющихся чисел

		Total Employees With This Job
Name	Job	
SCOTT	ANALYST	2
FORD	ANALYST	
SMITH	CLERK	4
ADAMS	CLERK	
MILLER	CLERK	
JAMES	CLERK	
JONES	MANAGER	3
CLARK	MANAGER	
BLAKE	MANAGER	
KING	PRESIDENT	1
ALLEN	SALESMAN	4
MARTIN	SALESMAN	
TURNER	SALESMAN	
WARD	SALESMAN	

6. Создайте представление, выбирающее по каждой должности в каждом из отделов число сотрудников и средний оклад.

DEPARTMENT	Job	NO_OF_EMPS	AVG_SAL
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT		5000
OPERATIONS		0	
RESEARCH	ANALYST	2	3000

17. Упражнение на управление транзакциями

1. Ответьте на следующие вопросы:
Как называется последовательность команд DML, пока их результаты не зафиксированы в базе данных командой COMMIT?
Можете ли вы отменить свои изменения после ввода команды COMMIT?
Можете ли вы отменить откат после ввода команды ROLLBACK WORK?
Как обеспечивает Oracle одновременное выполнение пользователями операций чтения и записи данных?
Где хранится информация, обеспечивающая откат?
2. Получите описание структуры таблицы COMPANY_CARS, которую вы создали ранее (на уроке 12). Используя переменные подстановки, введите в таблицу три строки.
Сохраните команду INSERT: она вам понадобится в следующем упражнении.
3. Вставьте в таблицу COMPANY_CARS еще несколько строк, на этот раз по своему выбору. Для формирования значения chassis_number используйте последовательность, созданную на уроке 15.

Измените команду INSERT, созданную в предыдущем упражнении, чтобы она работала быстрее. Запустив команду, вы заметите, что приглашение для ввода номера шасси больше не выдается, благодаря используемой последовательности.

Другой эффект вы можете обнаружить, если доберетесь до вставки шестой строки, когда получите сообщение о достижении верхнего предела значений последовательности. Как это можно исправить? Найдите решение, но не применяйте его!
4. Получите описание структуры представления CC_VIEW, созданного на уроке 16.

Вставьте строку в представление.

Выберите все данные из представления. Содержит ли полученный результат новую строку?

Выберите все данные из таблицы COMPANY_CARS. Видите ли вы здесь свою новую строку?
Если да, можете объяснить, почему?
Если нет, вы где-то ошиблись! - Обратитесь к преподавателю.
5. Создайте новую версию таблицы EMP при помощи следующей команды:

```
CREATE TABLE    EMP2
AS
SELECT          *
FROM            EMP;
```

Мы выкупили нашу компанию, и нам нужно внести решительные изменения в данные о сотрудниках.

Обновите таблицу EMP2, установив для всех сотрудников 1 июня 1995 г. в качестве даты приема на работу.

Обновите таблицу EMP2, установив каждому сотруднику оклад, равный среднему окладу в отделе, где он работает.

Если ваш конек - коррелированные подзапросы, можете выполнить только второй пункт этого упражнения.

Если у вас осталось время, выполните еще несколько заданий.

6. Под именем EMP3 создайте еще одну версию таблицы EMP.

Удалите все строки из EMP2 командой DELETE.

Выполните запрос всех строк EMP2. Сколько строк вы получили?

Выполните откат вашей транзакции и повторите запрос к таблице EMP2. Сколько строк вы видите теперь?

Удалите все строки из EMP3 командой TRUNCATE.

Выполните запрос всех строк EMP3. Сколько строк вы получили?

Выполните откат вашей транзакции и повторите запрос к таблице EMP3. Сколько строк вы видите теперь?

В чем различие между этими двумя командами?

7. Уничтожьте таблицу EMP3. Для проверки выполненной операции попробуйте получить описание ее структуры.

8. Удалите из вашей схемы все таблицы, кроме EMP, DEPT и SALGRADE и созданной вами таблицы COMPANY_CARS.

Для выполнения такой работы неплохо бы попробовать вариант с генерацией SQL при помощи SQL!

18. Упражнения на создание синонимов для таблиц; выборку информации о синонимах из словаря данных; выборку информации о привилегиях из словаря данных

1. Ответьте как можно точнее на следующие вопросы:

Как называется пользователь, в чьи обязанности входит создание других пользователей?

Какая нужна привилегия, чтобы соединиться с базой данных Oracle?

i. Это системная или объектная привилегия?

Какая нужна привилегия, чтобы пользователь мог создавать таблицы?

Если вы создали таблицу, кто может предоставить другим пользователям необходимые для работы с ней привилегии?

Может ли кто-нибудь еще, кроме вас, удалить вашу таблицу?

Предположим, что вы администратор базы данных. Вы создали достаточно много пользователей, и всем им необходимы одни и те же системные привилегии. Каким образом вы можете упростить себе работу?

Какой командой вы можете изменить свой пароль?

Может ли роль содержать объектные и системные привилегии?

Может ли пользователь получить несколько ролей?

2. Создайте возможно более короткий синоним для ранее созданной таблицы COMPANY_CARS. Обратитесь к таблице, используя синоним.

Выберите информацию о вашем синониме из словаря базы данных.

3. Удалите только что созданный синоним.

4. Найдите в базе данных ответы на следующие вопросы:

Какая роль вам предоставлена?

Какие системные привилегии выданы этой роли?

Глава 5. Содержание курсового проектирования по дисциплине «Базы данных и знаний», раздела по базам данных в курсовом проекте по дисциплине «Проектирование экономических информационных систем».

1. Содержание пояснительной записки курсового проекта

Задание на курсовой проект представляет собой проектирование и реализацию автоматизированного рабочего места бухгалтерии. Студент должен понимать место своего АРМ в составе комплексной бухгалтерии. Поэтому в последующих пунктах описывается проект комплексной автоматизации гипотетического предприятия. Предприятие осуществляет производственную деятельность, например

дорожное строительство. Задание представляет собой описание функций, баз данных АРМ. Пояснительная записка включает проектные документы, подготовленные на основе главы 1. Ниже приводится пример оглавления пояснительной записки и рекомендуемый объем.
Допустим, тема курсового проекта «Разработка АРМ учета основных средств».

Введение (1 - 2 страницы)
Основные термины — (АРМ, основные средства, нематериальные активы, документный, регистрационный учет).
Глава 1. Описание основных средств и нематериальных активов в хозяйственной деятельности предприятия. Организация учета.
1.1. Основные средства. Классификация (естественная или учетная, 5 - 7 стр.)
1.2. Нематериальные активы — определения (право на интеллектуальную собственность). Особенности учета. (5 –7 стр.)
1.3. Организация учета ОС и НМА на российских предприятиях. Перечень обязанностей бухгалтера — функциональная диаграмма, набор входных и выходных первичных документов, набор справочников (7—9 стр.).
1.4. Постановка задачи — что необходимо разработать, что должно облегчить работу бухгалтера (1стр.). Необходимо четко сформулировать цель автоматизации рабочего места, определить вход и выход программы.
Всего в главе около 20 – 25 стр.

Глава 2. Проектирование (разработка) АРМ ОС и НМА.
2.1. Структура информации АРМ ОС (перечень основных объектов проблемной области, первичных документов с атрибутами 1 стр.).
2.2. ER-диаграммы, сущность — связь БД (3-4 стр.).

2.3. Структуры данных (до 10 стр.).
2.4. Выбор инструмента (до 2 стр.).
2.5. Описание проекта (до 10 стр.). Необходимо использовать потоковые диаграммы (DFD) диаграммы и/или набор связанных функциональных IDEF-диаграмм для описания автоматизированных процессов.
Всего в главе около 20 — 25 стр.

Глава 3. Реализация, эксплуатация АРМ ОС и НМА.
3.1. Инсталляция АРМ (1 стр.).

3.2. Информация для программиста (инструкция программисту (2 стр.).
3.3. Пользовательская инструкция (5 — 7 стр.).
3.4. Технологическая карта эксплуатации (процедура эксплуатации АРМ, 1 стр.).
Всего в главе около 12-15 стр.

Заключение. Анализ эффективности и целесообразности вашей разработки (1-3 стр.).

2. Информационное обеспечение комплекса

2.1. Общее описание информации

В настоящем пункте приведена *логическая структура баз данных* комплекса автоматизированной бухгалтерии, приведены логические имена полей. Базы данных описаны в соответствии с задачами.

2.2. Сведения о кадрах предприятия

База кадров содержит поля: организация, табельный номер, категория, отдел, бригада, должность, фамилия и. о., дата рождения, пол, количество детей, количество иждивенцев, дата приема на работу, оклад, тариф, разряд, тариф ремонта, разряд ремонта, признаки льгот по подоходному налогу, сумма аванса, дата увольнения, причина увольнения, признак совместителя, признак члена профсоюза.

2.3. Сведения об основных средствах

1. База данных Карта учета основных средств, износа ОС по кодам затрат (счета с субсчетами) включает поля: дата выпуска, код материально ответственного лица, наименование объекта, цех, отдел, участок, дебет (счет), код аналитического учета, первоначальная стоимость, инвентарный номер, код затрат, код нормы износа, норма амортизационных отчислений на полное восстановление, на капитальный ремонт, вид, код оборудования, год выпуска, дата приобретения, дата списания, дата платы за фонды, суммарный износ, сумма амортизационных отчислений (всего), сумма за кап. Ремонт (всего), код источника поступления, сумма износа за каждый мес. фактический пробег.

2. Карта движения основных средств: вид операции, дата поступления (приобретения) ОС, код материально ответственного лица, месяц, год, цех, отдел, участок, дебет (счет), код аналитического учета кредит, сумма, инвентарный номер ОС, код затрат, код нормы износа, норма амортизационных отчислений на полное восстановление, на кап. ремонт, наименование ОС, сумма износа, код вида оборудования, год выпуска, дата выбытия, дата платы за фонды, код источникам поступления, фактический пробег.
3. База данных — Журнал операций для журнала - ордера 13: номер кредитового счета, номер дебетового счета, сумма, месяц, год, номер документа, дата название, признак поступления записи, комплексный ключ.

2.4. Сведения о материалах и движении материалов

Описание баз данных используемых задачей «ГСМ»

База данных путевых листов: номер документа, дата заполнения, инвентарный номер, код объекта, табельный номер, калькуляция, наименование машины, часы (всего), часы (ночные), показания спидометра в начале, показания спидометра в конце, общий пробег, общая масса в тоннах (т)., общий пробег под загрузкой, дополнительная норма ГСМ, код материала, остаток ГСМ на начало дня, выдано ГСМ, остаток ГСМ на конец дня, расход ГСМ факт., норма ГСМ на Машино-час (м/ч), норма ГСМ на 100 км., норма ГСМ на 100 т. на км., расход ГСМ по нормам, зарплата, склад, тариф, вид работ (до 8 видов), часы на работу (до 8 работ), расстояние в км (до 8 км.), количество рейсов(до 8 рейсов), масса в т (до 8), норма расхода ГСМ(до 8), норма расхода ГСМ(до 8), количество заправок (до 8), количество погрузок (до 8), код материала (до 8), единица измерения (до 8).
База данных расхода ГСМ: табельный номер, ФИО, код ГСМ, название ГСМ, ГСМ за месяц (для 12 месяцев), расход ГСМ факт., расход ГСМ нормальный, отклонение.
База данных — Пробег машин: инвентарный номер, название машины, пробег за месяц (для 12 месяцев), пробег (всего)
Справочник парка машин: инвентарный номер, марка машины, номер, группа, стоимость.
Справочник видов работ: код, название работы, единица измерения, норма, расценка.
Справочник ГСМ: код материала, наименование, категория, единица измерения, цена.

Справочник единиц измерения: код, название.
Справочник норм расхода ГСМ и З/П по машинам: код машины, вид работ, код материала, норма на 100 км, норма на м/ч, норма т/км, коэффициент (коэф.) ГСМ летний, коэф. ГСМ осенний, коэф. ГСМ зимний, коэф. ГСМ весенний, дополнительный расход.
Справочник объектов: код объекта, название объекта, шифр затрат, тариф командировок
Справочник категорий: категория, название категории, количество человек.

Описание баз данных используемых задачей «ТТН»

База данных поставок материалов: код объекта, код материала, сумма.
БД прейскуранта календарь поправочных коэф.: код автотранспортного предприятия (АТП), название прейскуранта, код автомобиля, месяц, данные по 1 декаде (для 10 дней), данные по 2 декаде (для 10 дней), данные по 3 декаде (для 10 дней).
БД парка машин: код автомобиля, марка, группа.
БД организаций: код организации, краткое название организации, полное название организации, расчетный счет, код банка, название банка, корреспондентский счет.
БД участков: код участка, название.
БД объектов: код объекта, название объекта, шифр затрат, тариф командировок.
БД материалов: код материала, общий код предприятия (ОКП), название, ГОСТ, марка, единица измерения, цена, коэф. перевода.
БД ТТН: номер ТТН, дата, код АТП, код склада, заказчик, получатель, код объекта, отправитель, код материала, единица измерения, количество материала, цена материала, сумма, класс груза, расстояние, время, тарифный коэф., месяц ввода, год ввода.
БД материально-ответственных лиц: код, фамилия.
БД отказов: месяц, код АТП, номер реестра, сумма отказа, признак отказа.
БД итогов по АПТ накопительные: код организации, название организации, сумма за год.
База данных в формате журнала-ордера: год, месяц, код организации, номер документа, шифр затрат, счет (дебет, кредит), сумма.
БД кодов затрат: код, название.
БД прейскуранта 13-01-01 раздел 1. п. 2: расстояние перевозки, цена (руб. коп.) < 0.5 т, цена (руб. коп.) < 1 т, цена (руб. коп.) < 1.5 т, цена (руб. коп.) < 2 т, цена (руб. коп.) < 3 т, цена (руб. коп.) < 4 т, цена (руб.

коп.) < 5 т, цена (руб. коп.) < 10 т, цена (руб. коп.) < 20 т, цена (руб. коп.) => 20 т.
БД прейскуранта раздел 1. п. 2: расстояние перевозки, класс груза 1, цена за 1т, класс груза 2, цена за 1т, класс груза 3, цена за 1т, класс груза 4, цена за 1т.
БД прейскуранта 13-01-01 раздел 3. п. 1: нижний предел грузоподъемности, верхний предел грузоподъемности, стоимость авточаса (руб. коп.), стоимость 1 км пробега, мин. плата, код грузоподъемности.
БД видов работ: код, название.

2.5. Сведения об объектах строительства и затратах

Структуры БД, используемые для команды «Обработка журнала ордера N 10»:
1. Базы данных Сводные файлы по отчетным периодам (месяцам): год, месяц, балансовый счет, кредитный счет, шифр затрат, сумма.
2. База данных операций из подсистемы «Зарплата»: год, месяц, шифр затрат, номер дебетового счета, номер кредитового счета, сумма, номер журнала-ордера, комплексный ключ.
3. База данных ТТН из задачи ТТН.
4. База данных Файл объектов ремонта из ТТН.
5. База данных список балансовых счетов: номер счета, название счета.
6. База данных нормы расходов ресурсов на объект: год, код объекта, номер счета, шифр затрат, сумма комплексный ключ.

2.6. Сведения о дебиторах и кредиторах

1. База данных операций (расчеты с дебиторами и кредиторами): номер кредитового счета, номер дебетового счета, сумма, месяц, год (две последние цифры), номер документа, дата, код статьи аналитического учета, код организации-дебитора, код организации-кредитора, номер журнала-ордера.
2 . База данных Список корреспонденций кредитовых счетов: номер кредитового счета, номер дебетового счета.
3. База данных Справочник дебиторов и кредиторов: код организации, название организации (краткое), название организации (полное), расчетный счет организации, код банка, название банка организации, корреспондентский счет.
4. База данных План статей аналитического учета: код строки в форме аналитического учета, название статьи.
5. База данных движения материалов: номер документа, дата, номер месяца, номер года, код организации, корреспондентский счет,

расчетный счет, номер дебетового счета, номер кредитового счета, примечание(содержание операции), сумма.
6. База данных в формате журнала-ордера из задачи ТТН.
7. База данных Список статей аналитического учета: номер счета (со всеми субсчетами), содержание счета.
8. База данных журнал-ордер 8: номер кредитового счета, номер дебетового счета, сумма, месяц, год, номер документа, дата, наименование статьи аналитического учета, код строки в форме аналитического учета, код организации.

2.7. Сведения о кассовых и банковских операциях

Описание баз данных используемых задачей «Касса»

База данных движения средств по кассе (прихода-расхода): номер документа, дата заполнения документа, месяц заполнения документа, год заполнения документа, табельный номер, сумма, корреспондентский счет и субсчет (по расходу), корреспондентский счет + субсчет (по приходу), поле примечаний, тип движения — приход/расход.
База данных остатков по кассе: остатки на начало месяца(для 12 месяцев) и года.
База корреспонденций: основной счет, корреспондирующий счет.
База кадров: организация, табельный номер, категория, отдел, бригада, должность, фамилия и.о., дата рождения, пол, количество детей, количество иждивенцев, дата приема на работу, оклад, тариф, разряд, тариф ремонта, разряд ремонта, признаки льгот по подоходному налогу, сумма аванса, дата увольнения, причина увольнения, признак совместителя, признак члена профсоюза.
База данных журнал-ордер1: счет (кредит), счет (дебет), сумма, дата, поле примечаний, номер документа.

Описание баз данных используемых задачей «Банк»

База налогов: код налога, название налога.
База остатков: остаток на расчетном счете (р/с) на начало месяца (для 12 мес.) остаток на р/с на начало года.
База данных движения средств по счету: номер документа, дата заполнения, месяц заполнения, год заполнения, сумма, счет (дебет), счет (кредит), код банка, р/с отправителя, код организации, поле примечаний, код статьи аналитического учета.

База банков: код банка, название банка.
База организаций: код организации, название организации, краткое название организации, полное название организации, расчетный счет, название банка, код банка.
База данных журнал-ордер 2: счет (кредит), счет (дебет), сумма, месяц, год, номер документа, дата, поле примечаний.

Структуры БД, используемые для команды «Обработка журнала ордера N 7»- расчеты с подотчетными лицами

БД операций для счетов (расчеты с подотчетными лицами): год (две последние цифры), месяц, дата, вид операции, номер документа, табельный номер, номер дебетового счета, номер кредитового счета, шифр затрат, сумма, основание для операции, номер журнала-ордера.
База данных справочная: список корреспонденций кредитовых счетов: номер кредитового счета, номер дебетового счета.

База данных Журнал-ордер № 7 по кредиту счета № 71 — расчеты с подотчетными лицами за период: табельный номер, фамилия, кредитовое сальдо на начало периода, дебетовое сальдо на начало периода, выдано в подотчет, возмещение перерасхода, удержание неиспользованной суммы аванса, с кредита счета 71 в дебет 50, удержание неиспользованной суммы аванса с кредита счета 71 в дебет 51, удержание неиспользованной суммы аванса с кредита счета 71 в дебет 70, удержание неиспользованной суммы аванса с кредита счета 71 в дебет 26, израсходовано из подотчетных сумм с кредитового счета 71 в дебеты различных счетов (не более 13 различных счетов) дебетовое сальдо на конец периода, кредитовое сальдо на конец периода.

Структуры БД, используемые для команды «Обработка журнала ордера N 11»

1. Базы данных: Входящее сальдо на начало года;
Входящее сальдо на начало месяца;
Оплата с кредита 46 счета;
Поступление в дебет 46 счета:
год, месяц, тип записи (1), кому отпущено (код физ. лица), номер документа, дата документа, номер дебетового счета, номер кредитового счета, сумма, комментарий, комплексный ключ комментарий (ОСН—основной, НДС — налог на добавленную стоимость, СНЛ — специальный налог).
2. База данных Список сотрудников предприятия.

Описание баз данных используемых задачей «Квартплата»

Тарифы на коммунальные услуги: название населенного пункта, код населенного пункта, год введения тарифных коэффициентов, месяц введения тарифных коэффициентов, категория жилья, плата за площадь, плата за отопление, плата за горячую воду, плата за холодную воду, плата за колонку, плата за канализацию, плата за мусор, плата за газ, плата за радио.

БД начисления квартплаты: год начисления, месяц начисления, табельный номер, фамилия, инициалы, количество жильцов, количество комнат, общая площадь, жилая площадь, категория жилья, код населенного пункта, 2 % оплаты коммунальных услуг, квартирная плата, плата за отопление, плата за холодную воду, плата за горячую воду, плата за канализацию, плата за вывоз мусора, плата за колонку, плата за газ, сальдо за предыдущий месяц, суммарная плата, сумма по 50 счету, сумма по 51 счету, сумма по 70 счету, сальдо за текущий месяц.

БД льгот по коммунальным платежам: номер лицевого счета, табельный номер, 2 % оплаты по коммунальным платежам, признаки льгот (010-услуги, 001-квартплата), начало действия льгот, конец действия льгот, комментарий.

3. Основные функции задач комплекса

3.1. Задача «Кадры»

Программа «Кадры» предназначена для автоматизированного ведения картотеки кадрового состава; учета поступлений и увольнений, внутреннего передвижения, отпусков; учета нарушений трудовой дисциплины; учета работников, занятых на рабочих местах с вредными условиями труда, учета выданной спецодежды. Программа позволяет наполнять, корректировать базу данных кадрового состава в диалоговом режиме. По состоянию картотеки кадров формируются сводные ведомости:

1. ПЕРЕЧЕНЬ КАДРОВОГО СОСТАВА ОРГАНИЗАЦИИ
2. ОТЧЕТ О НАЛИЧИИ И ДВИЖЕНИИ РАБОЧИХ ПО ПРОФЕССИЯМ
3. ПОИМЕННЫЙ СПИСОК РАБОЧИХ ВРЕДНЫХ ДЛЯ ЗДОРОВЬЯ ПРОФЕССИЙ, ПОДЛЕЖАЩИХ МЕДОСМОТРУ
4. СВЕДЕНИЯ О ДВИЖЕНИИ ИТР И СЛУЖАЩИХ ПО ОРГАНИЗАЦИИ

5.ОТЧЕТ О НАЛИЧИИ СОТРУДНИКОВ ПО КАТЕГОРИЯМ
6.ПЕРЕЧЕНЬ СПЕЦОДЕЖДЫ ПО ТАБЕЛЬНЫМ НОМЕРАМ
7. ПЕРЕЧЕНЬ СПЕЦОДЕЖДЫ ПО РАЗМЕРАМ
8. СПИСОК СОТРУДНИКОВ НАХОДЯЩИХСЯ В ОТПУСКЕ
9. НЕИСПОЛЬЗОВАННЫЕ ДНИ ОТПУСКОВ СОТРУДНИКОВ
10. СВЕДЕНИЯ О СОСТОЯНИИ ТРУДОВОЙ ДИСЦИПЛИНЫ

База данных кадрового состава эксплуатируется ежедневно в соответствии с движением контингента ИТР и рабочих. Ведомости частично формируются в конце отчетного периода, частично по запросу.

3.2. Задача «Учет заработной платы»

Программа «Учет заработной платы (з/п)» выполняет расчет з/п (в том числе аванса). С целью расчета з/п организована подзадача ведения нормативно-справочной информации (НСИ), для всех баз данных выполняется ввод, корректировка и распечатка содержимого. В состав НСИ входят:

1. Справочник КАДРЫ
2. Справочник ПАРК МАШИН
3. Справочник КЛАССИФИКАТОР МАШИН
4. Справочник ОРГАНИЗАЦИЙ
5. Справочник ОБЪЕКТОВ
6. Справочник ВИДОВ РАБОТ
7. Справочник МАТЕРИАЛОВ
8. Справочник ВИДОВ ОПЛАТ И УДЕРЖАНИЙ
9. Справочник ПЕРСОНАЛЬНЫЕ ДОПЛАТЫ И УДЕРЖАНИЯ
10.Справочник КАЛЕНДАРЬ РАБОЧЕГО ВРЕМЕНИ
11.Справочник ПРОФЕССИЙ
12.Справочник КАТЕГОРИЙ

Программа формирует следующие выходные ведомости.

1. Ведомость аванса
2. Балансовая ведомость
3. Расчетные листки
4. Расчетная ведомость
5. Ведомость полевых
6. Ведомость ПРЕМИИ ПО Коэффициенту трудового участия КТУ
7. Свод начислений и удержаний по категриям и ВО
8. ДОХОДЫ И НАЛОГИ
9. Ведомость удержаний алиментов

10. Ведомость распределения по шифру производственных затрат ШПЗ и отчислений
11. Ведомость распред по балансовым счетам и видам оплат
12. Ведомость ОБОРОТОВ ПО БАЛ. СЧЕТУ 70
13. Запросные ведомости по видам оплат - удержаний
14. Справка на удержания в сбербанк по заявлениям
15. П Л А Т Е Ж Н Ы Е ПОРУЧЕНИЯ В БАНК НА УДЕРЖАНИЯ
16. Своды для планово-экономического отдела и отдела труда и з.п. (П Э О и О Т И З)

Программа используется для расчета з/п ежемесячно, накапливает результаты расчетов. К функциям программы наряду с основными относятся следующие: ввод табелей и нарядов, ввод начислений и удержаний, удержаний промежуточных выплат, расчет по КТУ, ввод распределения коллективного фонда з/п сдельщиков, подсчет фонда заработной платы по организации. Выполняется подготовка и распечатка платежных поручений в банк по результатам начисления з/п.
Программа «Учет заработной платы» связана с задачами «Учет горюче-смазочных материалов» и «Касса», «Справочная система руководителя».

3.3. Задача «Расчет квартплаты»

Программа «Расчет квартплаты» выполняет расчет квартплаты на основе списка жильцов и действующих тарифов. Программа должна эксплуатироваться ежемесячно. Программа не связана технологически и технически с другими программами. Организационно входит в АРМ «Расчет заработной платы».

3.4. Задача «Учет основных средств»

Задача «Учет основных средств» выполняет ведение инвентарной картотеки основных средств, учет движения основных средств и пробега автомобилей, выполняет расчет износа и остаточной стоимости основных средств (ОС), переоценку стоимости (в том числе жилого фонда), переносит результат переоценки в инвентарную картотеку, формирует журнал-ордер №13 рассчитывает сальдо и обороты по счетам 01, 02, 07, 47, 88), переносит износ по балансовым счетам 01, 02,07, 47, 88. Задача формирует выходные ведомости, в том числе:

ВЕДОМОСТЬ ПЕРЕОЦЕНКИ
СВОДНАЯ ВЕДОМОСТЬ ПЕРЕОЦЕНКИ
ФОРМА 1 ПЕРЕОЦЕНКА ПО ИНВЕНТАРНЫМ НОМЕРАМ
ФОРМА 2 СВОДНАЯ ПО ГРУППАМ

Разные функции задачи используются с разной периодичностью. Ведение инвентарной картотеки и учет движения ОС выполняется по мере поступления ОС. Переоценка согласно сложившейся практике выполнялась ежегодно, журнал-ордер №13 формируется в соответствии с темпом, задаваемым ежеквартальным формирование отчетности (в том числе бухгалтерского баланса).

Программа «Учет основных средств» связана с задачей «Журналы-ордера», «Справочная система руководителя».

3.5. Задача «Учет горюче-смазочных материалов»

Задача «Горюче-смазочные материалы» предназначена для учета расхода ГСМ, учета пробега автомобилей, формирования сведений о расходе ГСМ для задач «Учет движения материалов» и «Учет заработной платы». Учет расхода ГСМ ведется на основе ввода и последующего ведения баз данных нарядов, сменных рапортов, путевых листов. В задаче ведется картотека учета работы автомобиля.

Для выполнения функций ГСМ в задаче используются следующие справочники:

1. СПРАВОЧНИК ПАРКА МАШИН
2. СПРАВОЧНИК НОРМ РАСХОДА ГСМ И НОРМ З/П
3. СПРАВОЧНИК ВИДОВ РАБОТ
4. СПРАВОЧНИК ОБЪЕКТОВ
5. СПРАВОЧНИК КАДРЫ
6. СПРАВОЧНИК ГСМ
7. СПРАВОЧНИК ЕДИНИЦ ИЗМЕРЕНИЯ ГСМ
8. СПРАВОЧНИК ПОПРАВОЧНЫХ КОЭФФИЦИЕНТОВ стоимости машинных смен
9. ОСТАТКИ ГСМ

По результатам учета формируются следующие выходные ведомости:

1. СПРАВКА РАСХОДА ГСМ ПО ТАБЕЛЬНЫМ НОМЕРАМ
2. СПРАВКА РАСХОДА ГСМ ПО ОБЪЕКТАМ
3. СПРАВКА РАСХОДА ГСМ ПО ИНВЕНТАРНЫМ НОМЕРАМ

4. СПРАВКА СВОДНАЯ С НАЧАЛА ГОДА ПО ТАБ. НОМЕРАМ
5. СВОДНЫЕ ВЕДОМОСТИ ВЫДАЧИ ГСМ ПО СКЛАДАМ
6. МАТЕРИАЛЬНЫЕ ОТЧЕТЫ ПО МАСТЕРАМ
7. СПРАВКА О ПРОБЕГЕ
8. СПРАВКА О ПРОБЕГЕ С НАЧАЛА ГОДА

Задача используется ежедневно по мере поступления и обработки путевых листов, нарядов и сменных рапортов. Ритм использования задачи «Учет ГСМ» диктует связь с задачей «Учет заработной платы» и «Учет движения материалов». До начисления очередной заработной платы и формирования сводных данных по учету движения материалов должна быть закончена обработка расхода ГСМ. Данные о ГСМ используются справочной системой руководителя.

3.6. Задача «Учет движения материалов»

Задача «Учет движения материалов» предназначена для учета прихода, расхода и остатков материалов на складах (движения материалов), учета доверенностей на получение материалов, комплектации и списания затрат по форме М-29, формирования журнала-ордера №10 С, учета товарно-транспортных накладных.
Программа «Учет движения материалов» может быть основой нескольких АРМ сотрудников бухгалтерии, в зависимости от разделения обязанностей между сотрудниками, принятыми в конкретной бухгалтерии.
Программа для выполнения функций позволяет вести следующие справочники:

1.Справочник ОБЪЕКТОВ
2.Справочник МАТЕРИАЛОВ
3.Справочник ОРГАНИЗАЦИЙ
4.Справочник НАИМЕНОВАНИЙ ЗАТРАТ
5.Справочник МАРОК АВТОМОБИЛЕЙ
6.Справочник МАТЕРИАЛЬНО-ОТВЕТСТВЕННЫХ ЛИЦ
7.Справочник ПРЕЙСКУРАНТ АВТОПЕРЕВОЗОК (р. 1. п. 1.)
8.Справочник ПРЕЙСКУРАНТ АВТОПЕРЕВОЗОК (р. 1. п. 2.)
9.Справочник ПРЕЙСКУРАНТ АВТОПЕРЕВОЗОК (р. 3. п. 1.)
10.Справочник КОЭФФИЦИЕНТОВ К ПРЕЙСКУРАНТУ (13-01-01)
11.Справочник УЧАСТКОВ
12.Справочник ВИДОВ РАБОТ

Функционально задачу «Учет движения материалов» можно разбить на четыре подзадачи: «Обработка ТТН», «Учет движения материалов» и «Комплектация и списания», «Обработка доверенностей». Подзадача «Обработка ТТН» — это ведение базы данных ТТН, таксировка и подготовка счетов по организациям и сводных документов по поставкам и транспортным расходам:

1.Реестр ТТН о поставках на объекты
2.Счета по коду организации
3.Справка о комплектации объектов по запросу
4.Сводная ведомость транспорных расходов
5.Ведомость транспортных расходов по ОБЪЕКТАМ
6.Поставка материалов на объекты с начала года

Для подготовки сводных документов вводятся нормы комплектации материалами объектов. Обрабатываются отказы от материалов, проверяется реестр ТТН. Подзадача «Учет движения материалов» включает в себя ведение базы данных приходно-расходных документов, корректировку движения материалов, подготовку выходных форм и сведений для задачи «Справочная система руководителя» Подзадача «Комплектация и списания» включает следующие функции: ввод и распечатку нормативов по М-29, ввод фактически выполненных объемов работ, ввод нормативной потребности материалов на объект. Формируются ведомости:

1.Форма М-29 объем выполненных работ по месяцам
2.Форма М-29 сравнение расхода материалов с нормами

Подзадача «Обработка доверенностей» предназначена для учета доверенностей, отчетов, возвратов. Сведения о доверенностях используются в «Справочной системе руководителя». Задача «Учет движения материалов» используется ежедневно, требуется подготовка к расчетам следующего месяца.

3.7. Задача «Касса»

Задача «Касса» представляет собой АРМ кассира. Программа позволяет выполнять следующие функции: ввод остатков по кассе, ввод документов по приходу (дебет счета 50), ввод документов по расходу (кредит счета 50), просмотр движения средств по кассе, формирование выходных ведомостей (в том числе журнала-ордера №1), запись удержаний в задачу «Учет заработной платы».

Задача формирует следующие выходные формы:

1.Отчет кассира (полный)
2.Отчет кассира (сокращенный)

3. Журнал-ордер №1
4. Ведомость №1 (по дебету счета 50).

Выходные формы формируются ежедневно и по результатам месяца. Задача связана с программой «Учет заработной платы».

3.8. Задача «Банк»

Задача «Банк» выполняет следующие функции: ввод остатков на р/с; ввод документов по приходу (дебет 51 счета); ввод документов по расходу (кредит 51 счета); формирование выходных ведомостей(в том числе журнала-ордера №1; предоставляет формируемую базу данных прихода-расхода по счету 51 задаче «Журнал-ордер 8».
Задача формирует следующие выходные формы:
- вспомогательная ведомость;
- разработочная ведомость;
- ведомость номер два;
- журнал - ордер номер два.

Ведомость может включать расчетные данные в двух вариантах:
- по конкретной дате;
- по итогам месяца;

При работе задачи используются следующие справочники: справочник организаций, справочник банков, справочник аналитического учета.

3.9. Задача «Справочная система для руководителя предприятия»

Справочная система для руководителя предприятия предназначена для получения оперативной информации из подсистем автоматизированной бухгалтерии. Справочная система включает в себя:

1. УЧЕТ З/П
2. МАТЕРИАЛЫ (СКЛАД)
3. ОБЪЕМЫ ВЫПОЛНЕННЫХ РАБОТ ПО ОБЪЕКТАМ (М-29)
4. ПОСТАВКИ МАТЕРИАЛОВ НА ОБЪЕКТЫ
5. УЧЕТ РАБОТЫ МАШИН И МЕХАНИЗМОВ
6. ТРАНСПОРТНЫЕ РАСХОДЫ
7. РАСЧЕТЫ С ПОСТАВЩИКАМИ И ЗАКАЗЧИКАМИ
8. ОСНОВНЫЕ СРЕДСТВА
9. НАЛОГИ
10. ФИНАНСОВЫЕ РЕЗУЛЬТАТЫ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ
11. КРЕДИТЫ (ССУДЫ БАНКА)

12. ДОВЕРЕННОСТИ

Эффективность справочной системы определяется заполненностью баз данных задач автоматизированной бухгалтерии. Если учесть, что большинство задач эксплуатируются с месячным расчетным периодом, то оперативность информации в лучшем случае один месяц.

3.10. Задача «Журналы-ордера»

Задача «Журналы-Ордера» представляет собой задачу получения журналов-ордеров по вводимым операциям. По замыслу (оставшемуся не реализованным) сальдо из журналов-ордеров должны сбрасываться в главную книгу, на основе которой строится бухгалтерский баланс. Все журналы-ордера (кроме 10) формируются на основе вводимых операций. Десятый журнал-ордер формируется по результатам функционирования задач «Расчет заработной платы», «Учет движения материалов» (списания по форме М-29), «Товарно-транспортные накладные». При формировании журналов-ордеров предусмотрен некоторый аналитический учет. При формировании журнала-ордера 7 формируются сальдо и обороты по подотчетным лицам. При формировании журнала-ордера 8 ведется аналитика по счетам 68, 77, 78. При формировании журнала-ордера 11 — по источникам поступления оплаты — физическим лицам, при формировании журнала-ордера 13 — по материально-ответственным лицам.

Только для журнала-ордера 8 ведется план статей аналитического учета, справочник статей аналитического учета, справочник дебиторов и кредиторов.

3.11. Задача «Главная книга, баланс».

Формирует Главную книгу и сводную бухгалтерскую отчетность.

4. Схемы информационных связей задач комплекса

4.1. Информационные связи программы «Кадры»

Рассмотрим в настоящем пункте автономную программу «Кадры».

Вход — экранные формы:

- картотека кадрового состава;

- учет поступлений и увольнений, внутреннего передвижения, отпусков;
- учет нарушений трудовой дисциплины;
- учет работников, занятых на рабочих местах с вредными условиями труда,
- учет выданной спецодежды.
Справочники не используются.
Выход — базы данных:
- база данных картотека кадрового состава;
- выходные формы:
1.ПЕРЕЧЕНЬ КАДРОВОГО СОСТАВА ОРГАНИЗАЦИИ
2.ОТЧЕТ О НАЛИЧИИ И ДВИЖЕНИИ РАБОЧИХ ПО ПРОФЕССИЯМ
3.ПОИМЕННЫЙ СПИСОК РАБОЧИХ ВРЕДНЫХ ПРОФЕССИЙ, ПОДЛЕЖАЩИХ МЕДОСМОТРУ
4.СВЕДЕНИЯ О ДВИЖЕНИИ ИТР И СЛУЖАЩИХ ПО ОРГАНИЗАЦИИ
5.ОТЧЕТ О НАЛИЧИИ СОТРУДНИКОВ ПО КАТЕГОРИЯМ
6.ПЕРЕЧЕНЬ СПЕЦОДЕЖДЫ ПО ТАБЕЛЬНЫМ НОМЕРАМ
7. ПЕРЕЧЕНЬ СПЕЦОДЕЖДЫ ПО РАЗМЕРАМ
8. СПИСОК СОТРУДНИКОВ В ОТПУСКАХ
9. НЕИСПОЛЬЗОВАННЫЕ ДНИ ОТПУСКОВ СОТРУДНИКОВ
10. СВЕДЕНИЯ О СОСТОЯНИИ ТРУДОВОЙ ДИСЦИПЛИНЫ

Связь с АРМ «Учет заработной платы», «Учет подотчетных лиц», АРМ «Учет движения материалов».

4.2. Информационные связи программы «Учет заработной платы»

Вход — экранные формы:
- табели;
- начисления и удержания;
- распределение коллективного фонда з/п сдельщиков.
Сервисные базы данных:
- база данных настройки на предприятие.
Справочники:
Справочник КАДРЫ
Справочник ВИДОВ ОПЛАТ И УДЕРЖАНИЙ
Справочник ПЕРСОНАЛЬНЫЕ ДОПЛАТЫ И УДЕРЖАНИЯ
Справочник КАЛЕНДАРЬ РАБОЧЕГО ВРЕМЕНИ

Справочник ПРОФЕССИЙ
Справочник КАТЕГОРИЙ
Программа накапливает результаты расчетов.
Выходные формы:

1. Ведомость аванса
2. Балансовая ведомость
3. Расчетные листки
4. Расчетная ведомость
5. Ведомость полевых работников
6. Ведомость ПРЕМИИ ПО КТУ
7. Свод начислений и удержаний по категориям и ВО
8. ДОХОДЫ И НАЛОГИ
9. Ведомость удержаний алиментов
10. Ведомость распределения по ШПЗ и отчислений
11. Ведомость распределения по балансовым счетам и видам оплат
12. Ведомость ОБОРОТОВ ПО БАЛ. СЧЕТУ 70
13. Запросные ведомости по видам оплат-удержаний
14. Справка на удержания в сбербанк по заявлениям
15. ПЛАТЕЖНЫЕ ПОРУЧЕНИЯ В БАНК НА УДЕРЖАНИЯ
16. Своды для ПЭО и ОТИЗ

Связь с другими программами:
- по входу:
- программа «Учет горюче- смазочных материалов»
- программа «Касса»,
- по выходу:
-программа «Справочная система руководителя»
-программа «Учет движения материалов» (списания по форме М-29)
-программа «Платежные поручения»

4.3. Информационные связи программы «Расчет квартплаты»

Программа «Расчет квартплаты» выполняет расчет квартплаты на основе списка жильцов и действующих тарифов. Программа не связана технологически и технически с другими программами. Организационно входит в АРМ «Расчет заработной платы».
Вход — экранные формы:
Справочники:
- база данных тарифов;
- база данных жильцов;
Связь с другими программами:

- по входу:
-программа «Учет заработной платы»;
- по выходу отсутствует.

4.4. Информационные связи программы «Учет основных средств»

Программа «Учет основных средств» связана с задачей «Журналы-ордера», «Справочная система руководителя».
Вход — экранные формы:
- инвентарная картотека основных средств;
- движение основных средств;
- карточка учета пробега автомобилей;
Выход — базы данных:
- база данных инвентарная картотека основных средств;
- база данных карточек учета пробега автомобилей;
Выходные формы:
ВЕДОМОСТЬ ПЕРЕОЦЕНКИ
СВОДНАЯ ВЕДОМОСТЬ ПЕРЕОЦЕНКИ
ФОРМА 1 ПЕРЕОЦЕНКА ПО ИНВ. НОМЕРАМ
ФОРМА 2 СВОДНАЯ ПО ГРУППАМ
Связь с другими программами:
- по входу отсутствует;
- по выходу:
- программа «Журналы-ордера»;
- программа «Справочная система руководителя».

4.5. Информационные связи программы «Учет горюче-смазочных материалов»

Вход - экранные формы:
- картотека учета работы автомобиля;
- наряды, сменные рапорты, путевые листы;
Справочники:
1. СПРАВОЧНИК ПАРКА МАШИН
2. СПРАВОЧНИК НОРМ РАСХОДА ГСМ И НОРМ З/П
3. СПРАВОЧНИК ВИДОВ РАБОТ
4. СПРАВОЧНИК ОБЪЕКТОВ
5. СПРАВОЧНИК КАДРЫ
6. СПРАВОЧНИК ГСМ
7. СПРАВОЧНИК ЕДИНИЦ ИЗМЕРЕНИЯ ГСМ
8. СПРАВОЧНИК ПОПРАВ. КОЭФ. стоимости машино-смен

9. ОСТАТКИ ГСМ

Выход — базы данных:

- база данных учета работы автомобиля;
- выходные формы:

1. СПРАВКА РАСХОДА ГСМ ПО ТАБЕЛЬНЫМ НОМЕРАМ
2. СПРАВКА РАСХОДА ГСМ ПО ОБЪЕКТАМ
3. СПРАВКА РАСХОДА ГСМ ПО ИНВЕНТАРНЫМ НОМЕРАМ
4. СПРАВКА СВОДНАЯ С НАЧАЛА ГОДА ПО ТАБЕЛЬНЫМ. НОМЕРАМ
5. СВОДНЫЕ ВЕДОМОСТИ ВЫДАЧИ ГСМ ПО СКЛАДАМ
6. МАТЕРИАЛЬНЫЕ ОТЧЕТЫ ПО МАСТЕРАМ
7. СПРАВКА О ПРОБЕГЕ
8. СПРАВКА О ПРОБЕГЕ С НАЧАЛА ГОДА

Связь с другими программами:

- по входу — отсутствует;
- по выходу :

-программа «Учет заработной платы»;

-программа «Учет движения материалов»;

-программа «Справочная система руководителя».

4.6. Информационные связи программы «Учет движения материалов»

Вход - экранные формы:

- доверенности на получение материалов;
- приход, расход, отказы и остатки материалов на складах;
- фактически выполненные объемов работ;
- товарно-транспортные накладные;
- нормы комплектации материалами объектов;

Справочники:

1. Справочник ОБЪЕКТОВ
2. Справочник МАТЕРИАЛОВ
3. Справочник ОРГАНИЗАЦИЙ
4. Справочник НАИМЕНОВАНИЙ ЗАТРАТ
5. Справочник МАРОК АВТОМОБИЛЕЙ
6. Справочник МАТЕРИАЛЬНО-ОТВЕТСТВЕННЫХ ЛИЦ
7. Справочник ПРЕЙСКУРАНТ АВТОПЕРЕВОЗОК (р. 1. п. 1.)
8. Справочник ПРЕЙСКУРАНТ АВТОПЕРЕВОЗОК (р. 1. п. 2.)
9. Справочник ПРЕЙСКУРАНТ АВТОПЕРЕВОЗОК (р. 3. п. 1.)
10. Справочник КОЭФФИЦИЕНТОВ К ПРЕЙСКУРАНТУ (13-01-01)
11. Справочник УЧАСТКОВ
12. Справочник ВИДОВ РАБОТ

Выход — базы данных:
- база данных прихода расхода материалов;
- база данных доверенностей;
- база данных ТТН;
- база данных фактически выполненных объемов работ;
- база данных нормативной потребности объектов в материалах.
- выходные формы:
- «Обработка ТТН»;
1. Реестр ТТН о поставках на обьекты
2. Счета по коду организации
3. Справка о комплектации объектов по запросу
4. Сводная ведомость транспорных расходов
5. Ведомость транспортных расходов по ОБЪЕКТАМ
6. Ведомость 2 транспортных расходов по ОБЪЕКТАМ
7. Поставка материалов на объекты с начала года
8. Счета по организациям
- «Комплектация и списания»
1. Форма М-29 объем выполненных работ по месяцам
2. Форма М-29 сравнение расхода материалов с нормами
Связь с другими программами:
- по входу отсутствует.
- по выходу :
- программа «Справочная система руководителя»;
- программа «Журналы-ордера».

4.7. Информационные связи программы «Касса»

Вход — экранные формы:
- документы по приходу (дебет счета 50);
- остатки по кассе;
- документ по расходу (кредит счета 50);
Справочники:
- база данных кадрового состава;
- справочник корреспондирующих счетов.
Выход — базы данных:
- база данных движения средств по кассе;
- база данных журнал-ордер 1.
Выходные формы:
1. Отчет кассира (полный)
2. Отчет кассира (сокращенный)
3. Журнал-ордер №1
4. Ведомость №1(по дебету счета 50).

Связь с другими программами:
- по входу отсутствует:
- по выходу:
- программа «Учет заработной платы».

4.8. Информационные связи программы «Банк»

Вход - экранные формы:
1. Ввод остатков на р/с
2. Ввод документов по приходу (дебет 51 счета)
3. Ввод документов по расходу (кредит 51 счета)

Справочники:
- справочник счетов;
- справочник банков;
- справочник организаций;
- справочник аналитического учета.

Выход — базы данных:
- база данных прихода-расхода по 51 счету.

Выходные формы:
1. Получение вспомогательной ведомости
2. Разработочная ведомость
3. Ведомость номер два
4. Журнал – ордер №2

Ведомость может включать расчетные данные в двух вариантах:
- по конкретной дате;
- по итогам месяца.

Связь с другими программами:
- по входу отсутствуют.
- по выходу:
- программа «1С-бухгалтерия»;
- программа «Журнал-ордер 8»

4.9. Информационные связи программы «Справочная система для руководителя предприятия»

Вход — экранные формы отсутствуют.

Справочники — базы данных задач:
1. УЧЕТ З/П
2. МАТЕРИАЛЫ (СКЛАД)
3. ОБЪЕМЫ ВЫПОЛНЕННЫХ РАБОТ ПО ОБЪЕКТАМ (М-29)
4. ПОСТАВКИ МАТЕРИАЛОВ НА ОБЪЕКТЫ

5. УЧЕТ РАБОТЫ МАШИН И МЕХАНИЗМОВ
6. ТРАНСПОРТНЫЕ РАСХОДЫ
7. РАСЧЕТЫ С ПОСТАВЩИКАМИ И ЗАКАЗЧИКАМИ
8. ОСНОВНЫЕ СРЕДСТВА
9. НАЛОГИ
10. ФИНАНСОВЫЕ РЕЗУЛЬТАТЫ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ
11. КРЕДИТЫ (ССУДЫ БАНКА)

Выход — базы данных отсутствуют.
- выходные формы — справки по перечисленным справочникам.
Связь с другими программами:
- по входу:
Задача «Учет заработной платы».
Задача «Учет основных средств».
Задача «Учет горюче-смазочных материалов».
Задача «Учет движения материалов».
Задача «Банк».
Задача «Журналы-ордера».
- по выходу отсутствует.

4.10. Информационные связи программы «Журналы-ордера»

ЖУРНАЛ ОРДЕР 1 — КАССА
ЖУРНАЛ ОРДЕР 2 — РАСЧЕТНЫЙ СЧЕТ
ЖУРНАЛ ОРДЕР 7 — РАСЧЕТЫ С ПОДОТЧЕТНЫМИ ЛИЦАМИ
ЖУРНАЛ ОРДЕР 8 — РАСЧЕТЫ С ДЕБИТОРАМИ КРЕД.
ЖУРНАЛ ОРДЕР 10 — 10/1 ЗАТРАТЫ НА ПРОИЗВОДСТВО
ЖУРНАЛ ОРДЕР 11 — МАТЕРИАЛЫ ЗА НАЛИЧНЫЙ РАСЧЕТ
ЖУРНАЛ ОРДЕР 13 — ОСНОВНЫЕ СРЕДСТВА

Информационные связи программы «Журнал-ордер 7»

Вход — экранные формы:
- суммы по подотчетным лицам;
Справочники:
- база данных кадрового состава;
Выход — базы данных:
- база данных по подотчетным суммам;

- выходные формы:
- журнал-ордер 7;
- сальдо и обороты по подотчетным лицам.
- сальдо и обороты по 71 счету.
Связь с другими программами:
- по входу отсутствует.
- по выходу :
- сводная отчетность.

Информационные связи программы «Журнал-ордер 8»

Вход — экранные формы:
- операции по журналу-ордеру 8.
Справочники:
СПРАВОЧНИК СТАТЕЙ АНАЛИТИЧЕСКОГО УЧЕТА
СПРАВОЧНИК ДЕБИТОРОВ КРЕДИТОРОВ
Выход — базы данных:
- база данных журнала-ордера 8.
Выходные формы:
ОБОРОТЫ ПО СЧЕТАМ ЗА ТЕКУЩИЙ МЕСЯЦ
ЖУРНАЛ ОРДЕР № 8
ВЕДОМОСТЬ № 7
АНАЛИТИКА ПО СЧЕТАМ № 68, 77, 78
САЛЬДО ДЕБИТОРОВ КРЕДИТОРОВ ПО СЧЕТАМ
САЛЬДО И ОБОРОТЫ ПО СЧЕТАМ №60, 68, 76
1. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 60
2. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 68
3. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 681
4. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 682
5. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 684
6. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 76
7. САЛЬДО И ОБОРОТЫ ПО СЧЕТУ 765
Связь с другими программами:
- по входу:
- программа «Банк»;
- программа «Учет движения материалов»(подзадача «ТТН»).
- по выходу:
- сводная отчетность.

Информационные связи программы «Журнал-ордер 10»

Вход — экранные формы:

- нормы затрат по статьям по объектам;
Выход — базы данных:
- база данных журнала-ордера 10;
- база данных нормативных затрат по объектам.
- выходные формы:
- журнал-ордер 10.
Связь с другими программами:
- по входу:
- программа «Учет З/П»
- программа «Учет движения материалов»
- подзадача «ТТН».
- по выходу:
- сводная отчетность.

Информационные связи программы «Журнал-ордер 11»

Вход — экранные формы:
- входные сальдо;
- реализация за наличный расчет (46 счет);
- оплата по 46 счету.
Справочники:
- база данных кадрового состава.
Выход — базы данных:
- база данных по оплате (кредит счета 46);
- база данных по реализации (дебит счета 46).
- выходные формы:
- журнал-ордер 11;
- обороты по счету 46.
Связь с другими программами:
- по входу отсутствуют.
- по выходу:
- сводная отчетность.

4.11. Информационные связи программы Главная книга, баланс

Вход — экранные формы:
- журнал операций.
Сервисные базы данных:
1) база данных список констант;
2) база данных список файлов задачи;
Справочники:

1) база данных список счетов;
2) база данных список субконто;
3) база данных список типовых проводок.
Выход — базы данных:
1) база данных главная книга;
2) база данных журнал операций:
- выходные формы определяются языком выходных форм.
Связь с другими программами:
- по входу:
- задача «Журналы-ордера»;
- по выходу отсутствуют.

5. Технологические связи задач комплекса

Для каждого предприятия, бухгалтерия которого автоматизирована вышеописанным комплексом программ, для успешной эксплуатации необходимо строгое соблюдение технологии: сроков заполнения баз данных и проведения расчетов. В настоящем пункте приведем основы технологической схемы эксплуатации. Опишем технологические связи задач комплекса автоматизированной бухгалтерии сверху вниз.

Чтобы сформировать сводную бухгалтерскую отчетность (баланс), главную книгу, необходимо зарегистрировать все операции по всем журналам-ордерам в 1С-бухгалтерии.
1. Чтобы сформировать 1-й журнал-ордер «Касса», необходимо обработать все первичные документы по приходу-расходу в кассе за отчетный период, сформировать обороты, сальдо и остатки на конец периода (Задача «Касса»).
2. Чтобы сформировать 2-й журнал-ордер «Расчетный счет» необходимо обработать платежные документы, отражающие движение на расчетном счете, сформировать обороты, сальдо, остатки на конец периода (Задача «Банк»).
3. Чтобы сформировать 3-й журнал-ордер «Чековая книжка» необходимо ввести операции в общий журнал 1С-бухгалтерии.
4. Чтобы сформировать 4-й журнал-ордер «Ссуды в банке» необходимо ввести операции в общий журнал.
5. Чтобы сформировать 5-й журнал-ордер необходимо ввести операции в общий журнал-ордер 1С-бухгалтерии.
6. Чтобы сформировать 6-й журнал-ордер необходимо ввести операции в общий журнал-ордер 1С-бухгалтерии.

7. Чтобы сформировать 7-й журнал-ордер «Расчеты с подотчетными лицами» необходимо ввести операции по 7-му журналу-ордеру, сформировать обороты сальдо и остатки на конец периода, получить аналитический учет по подотчетным лицам (Задача «Журналы-ордера»).
8. Чтобы сформировать 8-й журнал-ордер «Расчеты с дебиторами кредиторами» необходимо ввести операции по 8-му журналу-ордеру. Получить часть операций и информации из задач «Банк» и «Товарно-транспортные накладные», сформировать обороты сальдо и остатки на конец периода. Получить аналитику по дебиторам и кредиторам. До расчета 8-го журнала-ордера должны быть выполнены задачи «Банк» и «Товарно-транспортные накладные». В задаче «ТТН» должны быть обработаны все ТТН за отчетный период, а в задаче «Банк» — все платежные документы за отчетный период.
9. Чтобы сформировать 9-й журнал-ордер «Расчеты внутрихозяйственные» необходимо ввести операции в общий журнал-ордер 1С-бухгалтерии (Задача 1С-бухгалтерия).
10. Чтобы сформировать 10-й журнал-ордер 10/1 «Затраты на производство» необходимо ввести фактические объемы работ по объектам, нормативы по объектам, провести списание материалов на объекты по форме М-29, передать информацию о заработной плате из задачи «Учет заработной платы». Списание материалов, ввод фактических объемов и норм выполняется в задаче «Учет движения материалов». Перед списанием все материалы должны быть оприходованы по приходно-расходным документам в задаче «Учет движения материалов». То есть за отчетный период должны быть обработаны приходно-расходные документы на материалы, обработаны доверенности на получение материалов, проведены списания. Для учета заработной платы в затратах на производство за отчетный период должен быть выполнен расчет заработной платы. Для расчета заработной платы должны быть введены все табели учета рабочего времени, удержания из задачи «Касса», расход ГСМ из задачи «Горюче-смазочные материалы». Для формирования фактического расхода ГСМ должны быть обработаны все путевые листы и сменные рапорта за отчетный период в задаче «Горюче-смазочные материалы».
При выполнении перечисленных работ комплекс автоматизации бухгалтерии позволит сформировать 10-й журнал-ордер.
11. Чтобы сформировать 11-й журнал-ордер «Материалы за наличный расчет» необходимо ввести операции в 11-й журнал-ордер, сформировать обороты, сальдо и остатки на конец отчетного периода (Задача «Журналы-ордера»).

12. Чтобы сформировать 12-й «Движение фондов» журнал-ордер необходимо ввести операции в общий журнал-ордер.
13. Чтобы сформировать 13-ый журнал-ордер «Основные средства» необходимо обработать все документы по движению основных средств (инвентарная картотека), учесть пробег транспортных средств с помощью карточек автомобиля в системе «Учет ГСМ».
Для успешного использования каждой задачи необходимо корректное содержание справочников и собственных баз данных.

6. Ввод комплекса автоматизированной бухгалтерии в эксплуатацию

Ввод комплекса в эксплуатацию представляет собой переход от ручной обработки данных к компьютерной и должен включать в себя обязательное обучение сотрудников бухгалтерии основам работы за персональным компьютером. Такое обучение можно считать нулевым этапом ввода комплекса в эксплуатацию. (Этап 0).В нулевой этап автоматизатиции следует включить исследование масштаба документооборота конкретного предприятия, выбор и поставку технических и системных средств. Предполагается, что технические средства предприятий могут быть сформированы по трем вариантам: автономно стоящий компьютер, несколько автономных компьютеров и локальная вычислительная сеть.

Ввод комплекса существенно различается в зависимости от момента внедрения: внедрение в начале отчетного периода (календарный год, квартал) или в середине отчетного периода.
Первый этап ввода в эксплуатацию включает в себя инсталляцию комплекса и заполнение справочников. Инсталляцию и настройку на предприятие должен выполнять персонал отдела автоматизации, заполнение справочников может быть частично выполнено сотрудниками бухгалтерии. Часть справочников поставляются заполненными, например, справочник праздничных и выходных дней текущего года.

Содержимое всех заполненных справочников возможно скорректировать при вводе в эксплуатацию. Часть справочников заполняется на предприятии.

Второй этап ввода в эксплуатацию включает в себя ввод оперативной информации и первые расчеты с целью получения выходных форм.

Третий этап состоит в рабочей эксплуатации комплекса. Ввод в рабочую эксплуатацию комплекса предлагается проводить по следующим группам задач, имеющих тесные связи:

- «Главная книга, баланс»;
- «Учет заработной платы», «Касса», «Учет горюче-смазочных материалов»;
- «Учет движения материалов», «Учет товарно-транспортных накладных»;
- «Журнал-ордер №10»;
- «Учет основных средств», «Журнал-ордер №13»;
- «Банк», «Журналы-ордера №7, 8, 11»;
- справочная система для руководителя предприятия.

Ввод в эксплуатацию очередной группы задач подразумевает формирование и отладку связей с ранее введенными в эксплуатацию задачами. Таким образом, предлагается вести автоматизацию по принципу сверху-вниз, то есть первой вводится система, формирующая главную книгу и баланс на основе ручного ввода операций по всем журналам ордерам. Затем последовательно или параллельно по мере готовности программного обеспечения вводятся вышеперечисленные группы задач. В каждую группу задач входит соответствующий журнал-ордер. После ввода группы задач, автоматически формирующих журнал-ордер, такой журнал-ордер может вводиться с помощью конвертера в 1С-бухгалтерию. Следовательно, постепенно сокращается ручной ввод при формировании главной книги и снижается трудоемкость формирования баланса. Такая схема автоматизации позволит превратить бухгалтерский баланс из формы отчетности в инструмент анализа экономической ситуации. Снижение трудоемкости формирования баланса позволит сводить его ежемесячно и сделает полезной для руководителя предприятия "Справочную систему".

7. Задания на курсовое проектирование

Разработать автоматизированное рабочее место бухгалтера, решающего следующие задачи

1. Учет кадров предприятия (выполняется двумя студентами).
2. Учет основных средств (выполняется двумя студентами).
3. Учет движения материалов (выполняется двумя студентами).
4. Учет доверенностей.
5. Учет комплектования объектов и списание материалов.
6. Учет горюче-смазочных средств.

7. Учет товарно-транспортных накладных.
8. Журнал ордер 1 касса.
9. Журнал ордер 2 расчетный счет.
10. Журнал ордер 3 чековая книжка.
11. Журнал ордер 4 ссуды в банке.
12. Журнал ордер 7 расчеты с подотчетными лицами.
13. Журнал ордер 8 расчеты с дебиторами кредиторами.
14. Журнал ордер 9 расчеты внутрихозяйственные.
15. Журнал ордер 10, 10/1 затраты на производство.
16. Журнал ордер 11 материалы за нал. расчет.
17. Журнал ордер 12 движение фондов.
18. Журнал ордер 13 основные средства

Первые три варианта заданий выполняются двумя студентами, так как предполагают больший объем работы.

Заключение

Основной материал учебного пособия сложился у авторов в ходе преподавания дисциплин «Базы данных и знаний», «Теория ЭИС», «Проектирование ЭИС» студентам специальности «Прикладная информатика (в экономике)» факультета информационных систем и технологий Ульяновского государственного технического университета в 1997 — 2002 г. Основная идея, положенная в основу разработанного учебного пособия - это обеспечение глубокой интеграции компьютерных и экономических знаний. Авторы понимают русский термин информатика, как аналог западного computer science, поэтому полагают необходимым совершенное овладение студентами базовыми информационными технологиями. Все конкретные описания информационных систем, приведенные в учебном пособии, взяты из области организации бухгалтерского учета и экономического анализа.

Основная литература

1. Вендров А.М. Практикум по проектированию программного обеспечения экономических информационных систем: Учебное пособие. - М.: Финансы и статистика, 2002.-192 с.: ил.
2. Грабер М. SQL. Справочное руководство. Изд-во ЛОРИ, 1997.
3. Дейт К.Д. Введение в системы баз данных, 7-ое издание: Пер. с англ.- М.: Издательский дом «Вильямс», 2001.- 1072 с.: ил
4. Диго С.М. Проектирование и эксплуатация баз данных. - М.: Финансы и статистика, 1995.
5. Когаловский М.Р. Энциклопедия технологий баз данных. - М.: Финансы и статистика, 2002.- 800с.: ил.
6. Мишенин А. И. Теория экономических информационных систем. — М.: Финансы и статистика, 2001.
7. Г.Н. Смирнова, А.А.Сорокин, Ю.Ф.Тельнов Проектирование экономических информационных систем: Учебник/Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф.; под ред. Ю.Ф. Тельнова. - М.: Финансы и статистика, 2002.-512 с.: ил.
8. Ричардс М. и др. Oracle 7.3. Энциклопедия пользователя: Пер. с англ. К.: Изд-во «ДиаСофт», 1997. — 832 с.
9. Черемных С. В. и др. Структурный анализ систем: IDEF – технологии — М.: Финансы и статистика, 1998.

Дополнительная литература

10. Whitten J.L., Bentley L. D., Barlow V.M. Systems analysis and design methods. IRWIN, Burr Ridge, Illinois, Boston, Massachusetts, Sydney, Australia, 1991

11. Саймон А.Р. Стратегические технологии баз данных: менеджмент на 2000 год: Пер. с англ./Под ред. и с предисловием 12. М.Р. Когаловского.-М.:Финансы и статистика, 1999.- 479 с. ил.:

12. http://www.odmg.org
13. http://www.omg.org
14. http://www.sqlj.org
15. http://www.osp.ru/dbms
16. http://www.citforum.ru/database