Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Факультет непрерывного и дистанционного обучения

Кафедра информатики

Электронный учебно-методический комплекс

по дисциплине

**Модели данных**

**и**

**системы управления базами данных**

**(часть 2)**

Для студентов специальности

**1-31 03 04 “Информатика“**

Минск 2011

# Общие сведения

## Сведения об ЭУМК

Данный комплекс специально разработан для студентов, обучающихся дистанционно. Целью его создания было сделать процесс изучения данной дисциплины максимально удобным и комфортным для студентов. Кроме того, обладание комплексом должно позволить студенту минимизировать свои непосредственные контакты с университетом и компьютером, что особенно удобно для иногородних студентов, а также тех, кто по различным причинам вынужден на долгое время уезжать из города. В принципе данный комплекс позволяет студенту изучить дисциплину и подготовиться к сдаче экзамена по ней “автономно”

Выше приведён перечень и описание составных частей данного комплекса. Они выполнены в виде гиперссылок и для перехода к нужной части требуется, удерживая клавишу “Ctrl”, щёлкнуть левой кнопкой мыши выбранную гиперссылку. Впрочем, поскольку весь комплекс представляет собой файл Microsoft Word, его можно просматривать в обычном режиме, а также полностью или частично печатать

**Составитель: Калоша Е.П., Тараканов А.Н.**, доцент и ассистент**,** кафедры информатики Учреждения образования «Белорусский государственный университет информатики и радиоэлектроники».

Рассмотрен и рекомендован к изданию на заседании кафедры информатики, протокол № \_\_ от \_\_.\_\_.2011.

## Методические рекомендации по изучению дисциплины

В соответствии с учебным планом студенты дистанционной формы обучения инженерных специальностей изучают курс «Модели данных и системы управления базами данных (часть 2)».

Учебным планом по данному курсу предусмотрено изучение теоретических вопросов, решение задач, выполнение контрольной работы, 2 ИПР с ИКТ и курсовой работы. Изучение курса заканчивается сдачей экзамена. К сдаче экзамена студенты допускаются только при условии выполненных и защищенных контрольных работ, ИПР с ИКТ и курсовой работы.

Рекомендуется изучать курс «Модели данных и системы управления базами данных (часть 2)» в соответствии с рабочей программой. Сначала необходимо ознакомиться с содержанием курса, затем изучить рекомендуемую литературу, обращая внимание на вопросы, выделенные в рабочей программе, после чего изучить теоретическое изложение курса по приведенным разделам, темам и вопросам, ответить на контрольные вопросы, выполнить задачи для решения (выполнения контрольных работ) в соответствии с заданием.

Так как теоретический материал излагается в строгой логической последовательности, рекомендуется изучать данную дисциплину, придерживаясь данной логики.

## Рабочая учебная программа

**Учреждение образования**

**«Белорусский государственный университет**

**информатики и радиоэлектроники»**

УТВЕРЖДАЮ

Декан факультета непрерывного и дистанционного обучения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. М. Бондарик

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2010 г.

Регистрационный № -\_\_\_/.

**Модели данных**

**и**

**системы управления базами данных (часть 2)**

Рабочая учебная программа

**для направления специальности 1-31 03 04**

**“Информатика“**

Факультет **непрерывного и дистанционного обучения**

Кафедра **информатики**

Курс **пятый (семестр 10)**

**Контрольные работы** **1 работа**

**ИПР с ИКТ 2 работы**

**Курсовые работы 1 работа**

Всего часов **126часов**

Экзамен **5 курс**

Форма получения

высшего образования **дистанционная**

Минск 2011

Составители Калоша Е.П., Тараканов А.Н.

Учебная программа составлена на основе типовой учебной  *«Модели данных и системы управления базами данных (часть 2)*» для специальности 1-31 03 04 Информатика, утвержденной Министерством образования Республики Беларусь 14. 04. 2010 регистрационный № ТД-G.266/тип.

Рассмотрена и рекомендована к утверждению на заседании кафедры информатики

протокол № \_\_ от \_\_\_\_\_ \_\_\_\_

Заведующий кафедрой Минченко Л.И.

Одобрена и рекомендована к утверждению Научно-методической комиссией факультета компьютерных систем и сетей Учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

протокол № \_\_ от \_\_\_\_\_ \_\_\_

Председатель Лукашевич М.М.

СОГЛАСОВАНО

Начальник отдела методического обеспечения

учебного процесса \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ц. С. Шикова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Цель преподавания дисциплины. Дисциплина «СУБД» знакомит студентов с теоретическими основами БД и методами решения практических задач в этой области. Она связана с дисциплинами «Объектно-ориентированное программирование», «Теория алгоритмов и методы трансляции», «Методы и технологии программирования».

**Задачи изучения дисциплины.** В процессе обучения, студенты должны изучить методы проектирования баз данных; познакомиться с реляционной моделью, изучить языки запросов к базам данных; усвоить связь между теоретическими положениями, лежащими в основе баз данных, и практическими задачами проектирования, разработки СУБД и прикладных программ управления БД; приобрести навыки к решению практических задач проектирования баз данных и разработки программных приложений в этой области.

2. ПЕРЕЧЕНЬ ТЕМ ЛАБОРАТОРНЫХ ЗАНЯТИЙ, ИХ СОДЕРЖАНИЕ И ОБЪЕМ В ЧАСАХ

Лабораторные работы учебным планом не предусмотрены

Содержание дисциплины.

1. Название тем теоретического материала, их содержание, рекомендуемый объем в часах

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ку  р  с | | Название и содержание тем (*по типовой или учебной программе*) | Контрольная работа (номер и тема по п.2) | Лабораторная работа с указанием вида 1  (по п.1) | Оснащение контрольных и лабораторных работ  (по п.5) | Литература (по п.4) | Рекомендуемый объем для изучения (в часах)2 | Форма контроля знаний (зачет по контрольной работе, тесты, защита лабораторной работы, защита курсового проекта, экзамен, зачет) |
| 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 |
| **Часть 2** | | | | | | | | |
|  | | ***Тема 1. Среда SQL\*Plus. Взаимодействие с Oracle Server – команды SQL\*Plus – написание и выполнение SQL-предложений средствами SQL\*Plus.*** |  |  |  |  | 2 |  |
|  | | ***Тема 2. PL/SQL. Язык PL/SQL – посл-ти, блоки, переменные.*** |  |  |  |  | 2 |  |
|  | | ***Тема 3. Условия, циклы, курсоры.*** |  |  |  |  | 4 |  |
|  | | ***Тема 4. Хранимые процедуры , Триггеры.*** | КР |  |  |  | 2 | Зачет по контрольной работе |
|  | | ***Тема 5. Объекты, Транзакции. Реализация в языке PL/SQL.*** |  |  |  |  | 2 |  |
|  | | ***Тема 6. Функциональные зависимости***  ***избыточность – функциональные зависимости – аксиомы Армстронга – замыкание множества атрибутов – неприводимое множество функциональных зависимостей.*** |  |  |  |  | 4 |  |
|  | | ***Тема 7. Нормализация. Декомпозиция реляционных схем – восстановимая декомпозиция – 1НФ. 2НФ, 3НФ*** |  |  |  |  | 2 |  |
|  | | ***Тема 8.  Нормальная форма Бойса-Кода. Декомпозиция, сохраняющая зависимости – НФБК – пример декомпозиции в 3НФ*** |  | ИПР с ИКТ№1 |  |  | 2 | Защита ИПР с ИКТ №1 |
|  | | ***Тема 9. Нормальные формы более высокого порядка многозначные зависимости – 4НФ – зависимости соединения – 5НФ.*** |  |  |  |  | 2 |  |
|  | | ***Тема 10. Обработка запросов. Оптимизация запросов – процесс оптимизации, декомпозиция запроса – преобразование в каноническую форму – выбор низкоуровневых процедур, генерация планов запроса.*** |  |  |  |  | 3 |  |
|  | | ***Тема 11.  Алгоритмическая реализация операций реляционной алгебры.*** |  |  |  |  | 3 |  |
|  | | ***Тема 12. Физический уровень БД. Представление элементов данных.*** |  | ИПР с ИКТ№2 |  |  | 2 | Защита ИПР с ИКТ №2 |
|  | | ***Тема 13. Индексы. Индексы для последовательных файлов – В-деревья – Хеш-таблицы.*** |  |  |  |  | 2 |  |

2.КОНТРОЛЬНЫЕ РАБОТЫ, ИХ ХАРАКТЕРИСТИКИ

|  |  |  |  |
| --- | --- | --- | --- |
| № пп | Тема | Характеристика | Рекомендуемый объем  в часах |
| 1. | SQL: подзапросы, сложные запросы. | Применить подзапросы в операторе SELECT для построения сложных выборок. | 8 |

3.ИНДИВИДУАЛЬНЫЕ ПРАКТИЧЕСКИЕ РАБОТЫ, ИХ ХАРАКТЕРИСТИКИ

|  |  |  |  |
| --- | --- | --- | --- |
| № пп | Название темы | Содержание | Объем в часах |
| 1 | 2 | 3 | 4 |
| Десятый семестр | | | |
| 1. | PL/SQL: переменные, анонимный блок. | Применить основные элементы программы на PL/SQL, написать простейшие программы, использующие выборку данных из таблиц БД. | 4 |
| 2. | PL/SQL: курсоры, циклы, условия. | Применить основные управляющие конструкций программы на PL/SQL, написать программы, использующие ветвление и циклическую обработку и последовательное извлечение данных многострочных выборок с использованием явного курсора. | 4 |
| Итого: | | | 8 |

4.КУРСОВЫЕ РАБОТЫ (ПРОЕКТЫ), ИХ ХАРАКТЕРИСТИКИ

4.1. Тематика курсовых работ

Проектирование и реализация структуры БД и приложения для работы с БД с использованием современных программных платформ и средств программирования.

Работа с различными форматами БД. Преобразование форматов.

5. ЛИТЕРАТУРА

5.1.ОСНОВНАЯ

* + - 1. Дейт К.Дж. Введение в системы баз данных. К.; Диалектика, 1998
      2. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. М.: Издательский дом «Вильямс», 2003
      3. Мейер Д. Теория реляционных баз данных. М. Мир. 1987

# Теоретический раздел

## Лекции

### 1.модели данных реляционная модель.

**1.1** СБД предоставляет пользователю некие стандартные способы организации данных и методы доступа к ним. Эти стандартные способы задаются понятием модели данных.

Модель данных связана с тремя аспектами данных:

1.структурой;

2.целостностью;

3.обработкой данных.

**1.2** Структуры данных описывают объекты, в которых хранятся сущности и отношения.

Правила целостности описывают определённые законы, которые должны соблюдаться в БД, если она соответствует модели.

Операции над данными позволяют представить запросы в БД и манипуляции над данными.

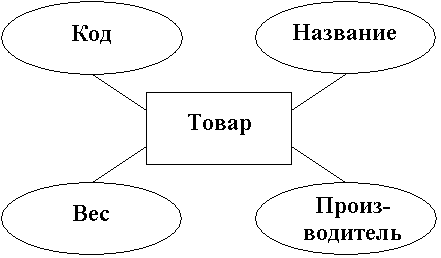
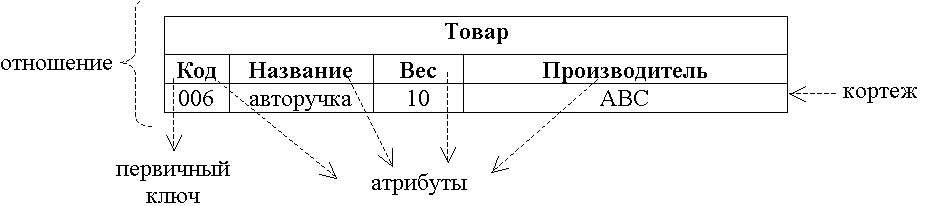
Существует три основных модели данных:

• иерархическая;

• сетевая;

• реляционная.

Между структурной диаграммой и реляционной моделью имеется следующее соответствие:



Домен-множество значений, которые принимает атрибут.

Количество атрибутов - степень (арность) отношения.

Первичный ключ - атрибут или совокупность атрибутов таких, что в любой момент времени не существует двух или более кортежей, содержащих одинаковые значения в этих атрибутах.

Переменная "отношение именованный объект, значение которого является отношением.

Реляционная БД - БД, воспринимаемая пользователем как набор отношений.

**1.4 Целостность реляционной БД**

в любой БД существуют правила целостности.

1. Потенциальные ключи.

Пусть R - переменная отношения, тогда потенциальный ключ К для R-подмножеств множества атрибутов R, всегда обладает следующими свойствами:

• уникальности (нет двух различных кортежей в некотором значении R с одинаковыми значениями К);

• не избыточности (никакое из подмножеств К не обладает свойством уникальности).

В реляционной модели один из потенциальных ключей выбирается в качестве первичного, а остальные называются альтернативными.

1. Внешние ключи.



Пусть R2 - отношение, тогда внешний ключ FK (Foreign кеу) в отношении R2 - это подмножество множества атрибутов в R2 такое, что:

• существует базовое отношение R1 с потенциальным ключом СК;

• каждое значение FK в текущем значении R2 всегда совпадает со значением СК некоторого кортежа в текущем значении R1.

Каскадирование.

### 2.Целостность реляционной модели.

СБД предоставляет пользователю некие стандартные способы организации данных и методы доступа к ним. Эти стандартные способы задаются понятием модели данных.

Модель данных связана с тремя аспектами данных:

1.структурой;

2.целостностью;

3.обработкой данных.

в любой БД существуют правила целостности. Они либо зависят от приложения (на- пример номера месяцев от 1 до 12), либо должны выполняться всегда для каждой БД.

**1. Потенциальные ключи.**

Пусть R - переменная отношения, тогда потенциальный ключ К для R-подмножеств множества атрибутов R, всегда обладает следующими свойствами:

• уникальности (нет двух различных кортежей в некотором значении R с одинаковыми значениями К);

• не избыточности (никакое из подмножеств К не обладает свойством уникальности).

В реляционной модели один из потенциальных ключей выбирается в качестве первичного, а остальные называются альтернативными.

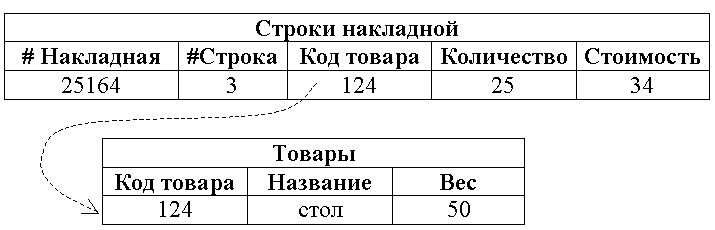
**2.Внешние ключи.**

Пусть R2 - отношение, тогда внешний ключ FK (Foreign кеу) в отношении R2 - это подмножество множества атрибутов в R2 такое, что:

• существует базовое отношение R1 с потенциальным ключом СК;

• каждое значение FK в текущем значении R2 всегда совпадает со значением СК некоторого кортежа в текущем значении R1.

БД должна обладать ссылочной целостностью, т.е. внешние ключи не должны содержать неверные значения, т.е. внешние ключи всегда должны быть согласованы, т.е. значению внешнего ключа в кортеже ссылающегося отношения должен всегда соответствовать кортеж; в целевом отношении.

Пример: как обеспечить ссылочную целостность?

Если удалить товар с кодом 124, либо изменить его в отношении Товары, то ссылочная целостность будет нарушена. Т.о. СУБД должна решать две следующие проблемы:

1. Что случится при попытке удалить объект ссылки внешнего ключа?

• ограничить (удалить нельзя) - запрет удалять целевой кортеж;пока есть хотя бы один ссылающийся на него кортеж;;

• каскадировать - удалить целевой кортеж; и все ссылающиеся.

2. Что должно случиться при попытке модифицировать потенциальный ключ, на который ссылается внешний ключ?

• ограничить;

• каскадировать.

При задании схемы БД разработчик должен указать не только множество атрибутов, составляющих потенциальный и внешний ключи, но и определить поведение СУБД при вышеперечисленных ситуациях.

1. Целостность атрибута. Значение атрибута должно быть из соответствующего ему домена.

### 3.Ссылочная целостность внешних ключей в реляционной модели.

СБД предоставляет пользователю некие стандартные способы организации данных и методы доступа к ним. Эти стандартные способы задаются понятием модели данных.

Модель данных связана с тремя аспектами данных:

1.структурой;

2.целостностью;

3.обработкой данных.

в любой БД существуют правила целостности. Они либо зависят от приложения (на- пример номера месяцев от 1 до 12), либо должны выполняться всегда для каждой БД.

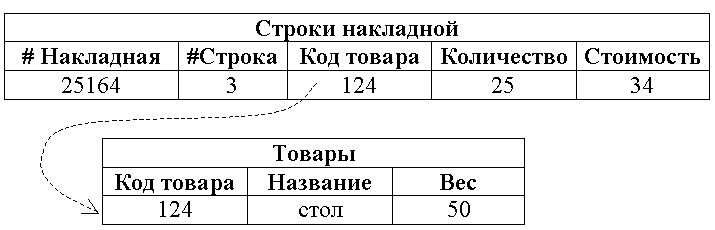
**2.Внешние ключи.**

Пусть R2 - отношение, тогда внешний ключ FK (Foreign кеу) в отношении R2 - это подмножество множества атрибутов в R2 такое, что:

• существует базовое отношение R1 с потенциальным ключом СК;

• каждое значение FK в текущем значении R2 всегда совпадает со значением СК некоторого кортежа в текущем значении R1.

БД должна обладать ссылочной целостностью, т.е. внешние ключи не должны содержать неверные значения, т.е. внешние ключи всегда должны быть согласованы, т.е. значению внешнего ключа в кортеже ссылающегося отношения должен всегда соответствовать кортеж; в целевом отношении.

Пример: как обеспечить ссылочную целостность?

Если удалить товар с кодом 124, либо изменить его в отношении Товары, то ссылочная целостность будет нарушена. Т.о. СУБД должна решать две следующие проблемы:

1. Что случится при попытке удалить объект ссылки внешнего ключа?

• ограничить (удалить нельзя) - запрет удалять целевой кортеж;пока есть хотя бы один ссылающийся на него кортеж;;

• каскадировать - удалить целевой кортеж; и все ссылающиеся.

2. Что должно случиться при попытке модифицировать потенциальный ключ, на который ссылается внешний ключ?

• ограничить;

• каскадировать.

При задании схемы БД разработчик должен указать не только множество атрибутов, составляющих потенциальный и внешний ключи, но и определить поведение СУБД при вышеперечисленных ситуациях.

1. Целостность атрибута. Значение атрибута должно быть из соответствующего ему домена.

### 4.Переход от структурной диаграммы к реляционной модели.

Структурная диаграмма - неформальная модель данных, которая служит для описания предметной области, используется как первое приближение схемы **БД.**

Сущность - вещь, которая существует и может быть выделена из других вещей того же рода.

Группа сходных сущностей - множество сущностей.

Атрибут: каждая сущность имеет свойство, называемое атрибутом, с каждым атрибутом ассоциируется значение, которое берётся из домена атрибута (множества его значений).

Ключ - атрибут или множество атрибутов, которые однозначно определяют сущность на множестве значений сущностей.

Отношение связывает один или более множеств сущностей.

Пример:

|  |  |  |
| --- | --- | --- |
| Сущности | Отношение | Тип отношения |
| детали | состоят из | замыкающее множество сущностей само на себя |
| склады детали | хранить | бинарное |
| поставщики детали проекты | поставлять | триадное |

Бинарные отношения классифицируются

1. Один-к-одному - каждой сущности из первого множества сущностей соответствует ровно одна в другом. Пример: НАЧАЛЬНИКИ, ОТДЕЛЫ.

2. Один-ко-многим - каждой сущности из первого множества сущностей соответствует несколько в другом. Пример: ГРУППЫ, СТУДЕНТЫ -ПРИНАДЛЕЖАТЬ.

3. Многие-ко-многим - каждой сущности из первого множества сущностей соответствует несколько в другом, каждой сущности из второго множества сущностей соответствует несколько в первом. Пример: СТУДЕНТЫ, ПРЕДМЕТЫ - УЧИТЬ.

4. Отношение "является"("is а"), если множество сущностей является подмножеством другого множества сущностей. Пример: НАЧАЛЬНИКИ, СОТРУДНИКИ.

Структурные диаграммы используют для наглядного преставления всей информации, которую необходимо ввести в БД.

**Преобразование структурных диаграмм в реляционную модель**

Преобразование множеств сущностей: результат - отношение, атрибутами которого являются атрибуты множества сущностей.

Преобразование отношения "является"("18 а"): в отношение, соответствующее бо

лее конкретному множеству сущностей, включающему внешний ключ на более общее.

# Сотрудники

“is a”

# Программисты

№ паспорта язык

Основной язык

|  |  |  |
| --- | --- | --- |
| **Программисты** | | |
| **ID\_Программиста** | **№ паспорта** | **Основной язык программирования** |
|  |  |  |

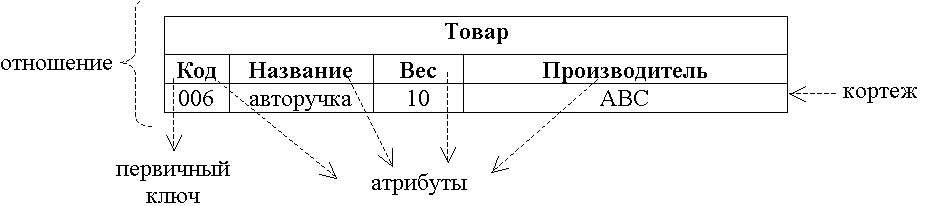
внешний ключ (FK)

Преобразование отношения один-к-одному: в оба отношения включают в качестве внешних ключей ссылки на другое отношение, один из ключей при этом можно убрать.

Преобразование отношения один-ко-многим: в отношении со стороны "многие" включают внешние ключи на отношение со стороны "один".

Преобразование отношения многие-ко-многим и парных отношений: создают вспомогательное отношение, которое содержит в себе ключи других отношений как внешние.

### 5.Реляционная алгебра. Операции реляционной алгебры.

Между структурной диаграммой и реляционной моделью имеется следующее соответствие:

Домен-множество значений, которые принимает атрибут.

Количество атрибутов - степень (арность) отношения.

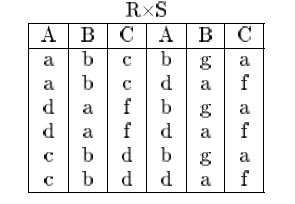
Первичный ключ - атрибут или совокупность атрибутов таких, что в любой момент времени не существует двух или более кортежей, содержащих одинаковые значения в этих атрибутах.

Переменная "отношение именованный объект, значение которого является отношением.

**Реляционная БД** - БД, воспринимаемая пользователем как набор отношений.

**Реляционная алгебра** состоит из операторов, использующих отношения в качестве операндов и возвращающих отношение в качестве результата. Операции делятся на группы:

традиционные операции над множествами **∪ ,∩, ⁄:**

пример:

R S

|  |  |  |
| --- | --- | --- |
| A | B | C |
| a  d  c | b  a  b | c  a  b |

|  |  |  |
| --- | --- | --- |
| A | B | C |
| b  d | g  a | a  f |

специальaные реляционные операции:

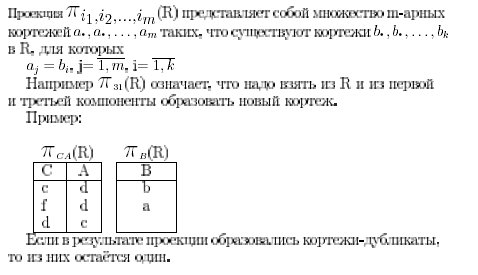
*выборка;*

*проекция;*

*соединение;*

*деление.*

**Проекция**

****

**Операция выборки**

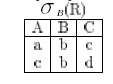
Пусть F - некоторая формула, которая включает в себя операнды, представляющих

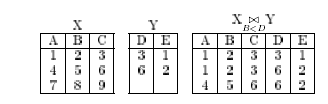
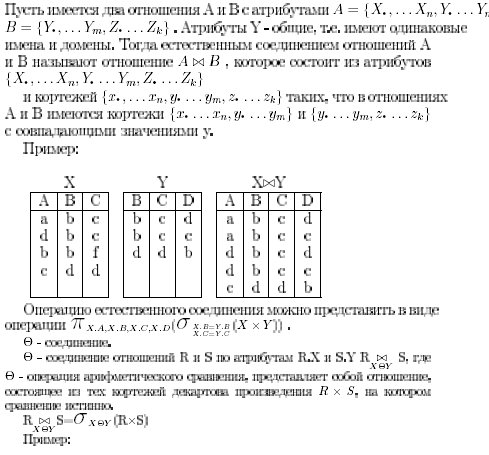
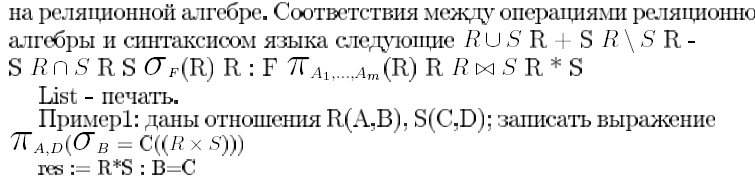
собой константы и имена атрибутов, а также операции сравнения и логические

операции

Выборка  из отношения R представляет собой отношение с теми же атрибутами, что и R, и со всеми такими кортежами, на которых формула F истинна.

Пример:

**Операция естественного соединения**

****

Покупатель (Имя, Адрес, Счёт)

Заказ (Номер, Имя, Товар, Количество)

Поставщик (Фирма, Адрес фирмы, Товар, Цена)

Эта **БД** поддерживается фирмой, которая поставляет товары покупателям на основе их заказов. Это фирма-посредник, которая покупает товары у поставщиков. Покупатели предоставляют фирме некоторое количество денег, которые отмечаютя в атрибуте Счёт.

Запрос!: напечатать всех должников

List Покупатель : Счёт < О

Запрос2: напечатать имена фирм, набор товаров и их цены для заказа покупателя с именем Петров.

List Заказ\*Поставщик : Имя = 'Петров'

ЗапросЗ: напечатать имена поставщиков, которые могут поставить все предметы, затребованные Петровым.

S := Поставщик I := Поставщик В:= Заказ : Имя = 'Петров' NS := 8\*1 - (Поставщик //фирма может поставить NSB := NS.(S\*B) //все пары Поставщик-Товар, где поставщик не может поставить //товары, заказанные Петровым. List S - (NSB

### 6.Аномалии избыточности. Функциональные зависимости.

При проектировании реляционных БД следует выбрать такую ее схему , чтобы она была в некотором смысле лучшей. Рассмотрим вначале ,какими свойствами не должна обладать БД.

***Пример***: Задано отношение : Поставщики(Название , Телефон , Предмет ,Цена) отношение создает информацию о поставщиках товара. Для каждого поставщика имя и телефон ,для каждого товара название и цена.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | телефон | предмет | цена |
| КИРС КИРС  РОСУАМ РОСУАМ | 235523 235523 330793 330793 | Compa Q 8086  Epson 8088  RV 286 RO  Intel 386 | 170 260 596 591 |

Ключ отношение название предмет. В этой схеме есть недостатки

1. В ней есть избыточная информация - дублируется номер телефона, последствия : увеличение файла БД , уменьшение эффективности запроса.

2. Аномалии изменения : бд потенциально может содержать противоречивую информацию - для одной и той же фирмы разные номера телефонов.

3. Аномалии вставки : если фирма ничего не продает , то нельзя сохранить информацию о фирме.

4. аномалия удаления: Если все предметы, то удалится информация о поставщике.

Аномалии - следствие избыточности.

Избыточность и все сопутствующие проблемы можно исключить , если отношение поставщики (Название , телефон , предмет , цена) разбить : на Фирмы(название, телефон ) и Товары (Название , Предмет , Цена)

Недостатки новой схемы : Если необходимо получить информацию и о товаре и о телефоне фирмы-продавца ,то необходимо сделать соединение 2-х таблиц. Операция соединения крайне трудоемка. В предметном варианте схемы эту операцию не нужно было проводить.

Основная проблема проектирования БД : Как избежать избыточности , как получить из плохой избыточности хорошую без избыточную.

Центральным понятием в нормализации БД (т. е. приведение к избыточному состоянию) является функциональные зависимости . Если 1 атрибут однозначно определяет значение другого , то говорят о функциональной зависимости. Простейший случай - ключ. Пример: Название фирмы функционально определяет телефон фирмы.

Определение: Пусть R - отношение . X,Y- произвольные подмножества множества атрибутов R. X функционально определяет Y. Y функционально зависит от X. Если для 2 кортежей с одинаковыми значениями атрибутов X , атрибуты Y будут также равны. *X* —> *Y*

Пример :Название —> Телефон

Пример :Название ,Предмет —> Цена

Единственный способ определить функциональную зависимость - анализ предметной области, функциональная зависимость - смысловое свойство атрибутов отношения, функциональные зависимости являются ограничениями целостности БД. При каждом обновлении БД все они должны быть проверены. В БД очень много функциональных зависимостей. Хотелось бы найти минимальное множество такое, что если все выполняется в БД , то выполняется и все оставшиеся функциональные зависимости.

### 7.Аксиомы Армстронга.

Центральным понятием в нормализации БД (т. е. приведение к избыточному состоянию) является функциональные зависимости. Если 1 атрибут однозначно определяет значение другого , то говорят о функциональной зависимости. Простейший случай - ключ. Пример: Название фирмы функционально определяет телефон фирмы.

Определение: Пусть R - отношение . X,Y- произвольные подмножества множества атрибутов R. X функционально опредеяет Y. Y функционально зависит от X. Если для 2 кортежей с одинаковыми значениями атрибутов X , атрибуты Y будут также равны. *X* —> *Y*

Пример :Название —> Телефон

Пример :Название,Предмет —> Цена

#### Аксиомы Армстронга

Если в отношении имеются некоторые функциональные зависимости , то из них можно вывести другие функциональные зависимости также выполняющиеся в этой БД.

Пример: Если в отношении имеется R некоторые функциональные зависимости , то из них можно вывести другие функциональные зависимости *А* —> *В* и *В* —> *С* , то в нем имеются функциональные зависимость *А* —> *С*

О**пределение:** пусть в R имеется 2 кортежа такие что А- равны , а С - нет. В соответствии с определением функциональной зависимости для этих 2 кортежей значение атрибутов В будут равны, т. к. имеет место функциональная зависимость *А* —> *В .* Из наличия функциональной зависимости *В* —> *С* следует , что значения атрибутов С равны. Противоречие доказывает утверждение .

**Определение:** Множество всех функциональных зависимостей ,которые выводятся из множества функциональных зависимостей F называется замыканием F и обозначается *F+*

Пример : Пусть задано отношение *R = ABC* и есть функциональная зависимость : *F = {А* —> *В, В* —> *С}*

*F+* состоит из следующих функциональных зависимостей вида *X* —> *Y:*

1. *А ∈ X : ABC -> АС, АВ -> ВС, А* -> *А,...*

1. *А ∉ X B ∈X*

*A∉Y :BC ->B,B->C,B->0*

3. *С -> С,С -> О*

Задание правила вывода одних функциональных зависимостей из других.

Аксиомы:

1. рефлексивность : если , то *А* —> *В*

*2.* дополнение : если *А* —> *В* ,то *АС* —> *ВС*

3. транзитивность : если *А* —> B,В —> С, то A -> *С*

Аксиомы Армстронга непротиворечивы следовательно функциональные зависимости , которые получают с их помощью действительно содержатся в *F+.* Аксиомы Армстронга полны. Это значит что по этим правилам можно получить все функциональные зависимости которые есть в *F+.* Эти 2 утверждения позволяют утверждать , что вывод новых функциональных зависимостей на основе определения функциональной зависимости и при помощи Аксиом Армстронга эквивалентны.

Из Аксиом Армстронга можно получить произвольные правила:

Если *X -> У* и *X -> Z,* то *X -> YZ*



### 8. Замыкание множества атрибутов.

Цель :Найти множество неприводимых зависимостей.

Рассмотрим следующую задачу: Дано множество функциональных зависимостей F и функциональная зависимость *X —>Y.*

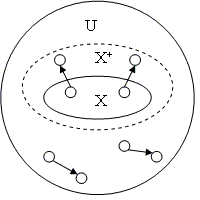
Необходимо определить : принадлежит ли *X* —> *Y* к замыканию *F+.* Иначе говоря выводима ли *X* —> У из множества F.

*F -> F+*

*X ->Y*

Для решения этой задачи используется замыкание атрибутов.

**Определение :** Пусть F - множество функциональных зависимостей , определенных на множестве атрибутов U (все атрибуты , которые могут встретится на F) и X подмножество U . Тогда оболочки X по отношению к F (обозначение *X*+) есть множество атрибутов А , таких что функциональные зависимости *X* —> *А* могут быть выведены из F при помощи Аксиом Армстронга.



**Алгоритм построения замыкания атрибутов**

Алгоритм приводится на псевдо коде.

Замыкание= *X*

hile (1)

для каждого *А* —> *В ∈F*

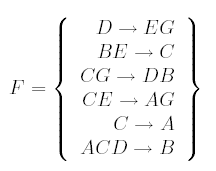
if *(А* ⊆Замыкание)

Замыкание = Замыкание ∪ B;

if( замыкание не применялось за последнюю итерацию)

Break;

Пример:



Построим оболочку множества атрибутов CG. Начиная с CG пробегаем все множествао и выбираем два.



CG - является потенциальным ключом отношения . Атрибуты CG однозначно определяют все атрибуты отношения .

Исходя из вышесказанного следует , если *Х+ = U* то потенциальный ключ отношения содержится в X, чтобы его выделить поступаем так:

1. Из X удаляем произвольный атрибут *Х1 = Х\А*

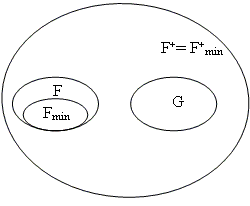
2. Если вновь * = U* , то атрибут А лишний и процедуру повторяем.

3. Продолжаем до тех пор пока не получим такое множество из которого нельзя выбросить ни одного атрибута .Это и будет потенциальный ключ.

Теперь решаем исходную задачу при помощи введенных операций . Необходимо определить истинно ли условие : *X* —> *Y* ∈ *F+* Оно будет верным , если будет выполнятся следующее условие *:Y* ⊆ *Х+.*

### 9.Неприводимое множество функциональных зависимостей.

Решим задачу : Найти минимальное множество функциональных зависимостей , из которых можно получить все остальные функциональные зависимости .рисунок



Пусть F1 и *F2 -* два множества функциональных зависимостей . Если ** то говорят что множество *F2* покрывает множество ** Иначе говоря , любая функциональная зависимость ,выводимая из ** может быть выведена и из *.*

Если ** покрывает **и наоборот, то говорят что множества **и **эквивалентны.



Множество функциональных зависимостей является неприводимым , если выполняются свойства :

1. Часть каждой функциональной зависимости создает только 1 атрибут.

2. Левая часть любой функциональной зависимости является неприводимой . Это значит что если любой атрибут левой части опускается , то замыкание *F+* изменяется.

3.Ни одна функциональная зависимость в F не может быть выброшена из F без изменения *F+.*

Алгоритм построения неприводимого множества функциональных зависимостей.

1. Переписать функциональную зависимость т.о. , чтобы каждой из них в правой части имела единственный атрибут.

*А->В⇒А-> В,А->С*

То , что такое преобразование допустимо доказывается при помощи аксиом Армстронга.

2. Проверить можно ли опустить какой - либо атрибут из левой части функциональной зависимости.

Способ: Пусть имеется функциональная зависимость *:Х* — > *А, В* ∈ *X*

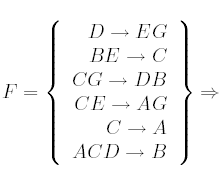
*А* ∈ *(Х\В)+*

*(Х\В)* — > *А* ∈*F+* и В можно опустить из левой части функциональной зависимости .

3. Проверить можно ли опустить какую-либо функциональную зависимость.

Способ: Пусть имеется функциональная зависимость *X* — > *А.* Построим *Х+* над *{F\X* -> *А}.* Если *А* ∈ *Х+* , то *X -> А* ∈*{F\X -> A}+* и эту функциональную зависимость можно исключить .

Пример: Пусть имеется множество :

шаг2 : Функциональная зависимость СЕ , функционально определяет



А, исключается , т.к. Е м.б. опущен

 включает атрибут B CG→B можно исключить



Шаг 3:



### 10.Алгоритм поиска потенциальных ключей.

Цель :Найти множество неприводимых зависимостей.

Рассмотрим следующую задачу: Дано множество функциональных зависимостей F и функциональная зависимость *X —>Y.*

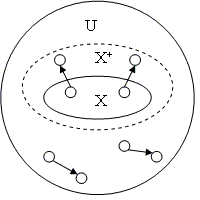
Необходимо определить : принадлежит ли *X* —> *Y* к замыканию *F+.* Иначе говоря выводима ли *X* —> У из множества F.

*F -> F+*

*X ->Y*

Для решения этой задачи используется замыкание атрибутов.

**Определение :** Пусть F - множество функциональных зависимостей , определенных на множестве атрибутов U (все атрибуты , которые могут встретится на F) и X подмножество U . Тогда оболочки X по отношению к F (обозначение *X*+) есть множество атрибутов А , таких что функциональные зависимости *X* —> *А* могут быть выведены из F при помощи Аксиом Армстронга.



**Алгоритм построения замыкания атрибутов**

Алгоритм приводится на псевдо коде.

Замыкание= *X*

hile (1)

для каждого *А* —> *В ∈F*

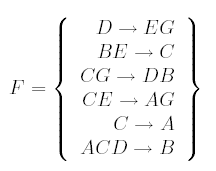
if *(А* ⊆Замыкание)

Замыкание = Замыкание ∪ B;

if( замыкание не применялось за последнюю итерацию)

Break;

Пример:



Построим оболочку множества атрибутов CG. Начиная с CG пробегаем все множествао и выбираем два.



CG - является потенциальным ключом отношения . Атрибуты CG однозначно определяют все атрибуты отношения .

Исходя из вышесказанного следует , если *Х+ = U* то потенциальный ключ отношения содержится в X, чтобы его выделить поступаем так:

1. Из X удаляем произвольный атрибут *Х1 = Х\А*

2. Если вновь * = U* , то атрибут А лишний и процедуру повторяем.

3. Продолжаем до тех пор пока не получим такое множество из которого нельзя выбросить ни одного атрибута .Это и будет потенциальный ключ.

Теперь решаем исходную задачу при помощи введенных операций . Необходимо определить истинно ли условие : *X* —> *Y* ∈ *F+* Оно будет верным , если будет выполнятся следующее условие *:Y* ⊆ *Х+.*

### 11.Декоипозиция реляционных схем .Восстановимая декомпозиция.

### 12.Декоипозиция реляционных схем .Условия восстановимой декомпозиции.

Декомпозиция реляционной схемы *R = {A1,A2,...,An}* есть множество схем *р = {Ri, R2, . . . , Rk}* таких, что *.*

Цель декомпозиции: устранение избыточности, которая приводит к аномалиям.

Пример:

Пусть есть реляционная схема Поставщики (Название , телефон, предмет, цена)=НТПЦ. В схеме имеются следующие функциональный зависимости: *F =* {Н — > Т, НП — > Ц}. Чтобы устранить избыточность, НТПЦ-схема разделяется на две реляционные схемы:

, R1=НТ, R2=НПЦ

Пусть имеется отношение реляционной схемы *R =* НТПЦ. После декомпозиции получаем проекции **r**  в схемы НТ и НПЦ: , Можно ли сказать, что r нт и r нпц содержат такую же информацию, что и  **r** ? Как восстановить обратно?

Чтобы получить исходное отношение г из проекций, следует выполнить обратную операцию соединения.



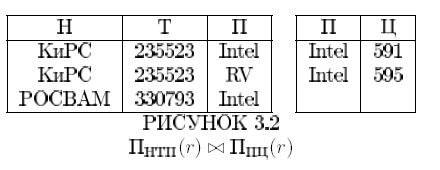
Необходимо ответить на следующий вопрос: при каких условиях **S=r**, - т.е., когда отношение восстановимо из своих проекций?

Пример:

Восстановимая декомпозиция

*R =* {НТПЦ}

|  |  |  |  |
| --- | --- | --- | --- |
| н | Т | П | Ц |
| КиРС КиРС РОСВАМ | 235523 235523 330793 | Intel RV  Intel | 591 596 595 |



|  |  |  |  |
| --- | --- | --- | --- |
| Н | Т | П | Ц |
| КиРС | 235523 | Intel | 591 |
| КиРС | 235523 | Intel | 595 |
| КиРС | 235523 | RV | 596 |
| РОСВАМ | 330793 | Intel | 591 |
| РОСВАМ | 330793 | Intel | 595 |

В общем случае, что подтверждает данный пример, соединение проекций не равно исходному отношению,



Чтобы получить восстановимую декомпозицию, т.е. когда отношение равно соединению его проекций, следует учитывать его функциональные зависимости. Верна следующая теорема:

Пусть *р ={Ri,R2} -* декомпозиция схемы R, F - множество функциональных зависимостей в R, тогда *р* имеет восстановимое соединение по отношению к F тогда и только тогда, когда  принадлежит *F+*

Пример:

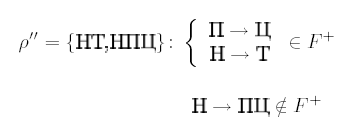
*R =* {НТПЦ} *F =* {Н -> Т, НП -> Ц}

1.

*р' =* {НТП,НПЦ}:

Используя ранее рассмотренные методы можно доказать, что рассмотренные ранее зависимости не принадлежат оболочке и данная декомпозиция не будет восстановимой.

2.



### 13.Нормальные формы реляционных схем.1и 2НФ.

Нормализация - это пошаговый процесс устранения избыточности в реляционной схеме.

Нормальные формы - это реляционные схемы, в которых устранен некоторый тип избыточности.

В процессе нормализации реляционная схема будет находится в соответствующей нормальной форме.

Реляционная схема находится в первой нормальной форме, если никакое значение атрибута не может быть множеством значений или повторяющейся группой.

Пример:

Рассмотрим следующую ситуацию - студент несколько раз сдает экзамен и каждой пересдаче соответствует отдельная оценка.

|  |  |  |
| --- | --- | --- |
| Id студента | Id курс | Оценка |
| 254  254  256 | 17  18  17 | 2,4  5 2,3,4 |

Нормализованное:

|  |  |  |  |
| --- | --- | --- | --- |
| Id студента | Id курс | Номер пересдачи | Оценка |
| 254  254  254  256 | 17  17  18  17 | 1  2  1  1 | 2  4  5  2 |

Если для обработки даты необходимо учитывать год, месяц и день, то под каждое из них заводится отдельное поле.

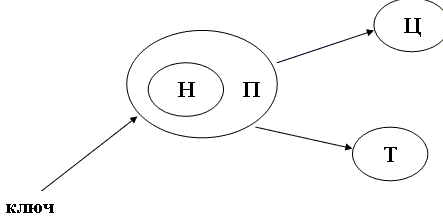
**Вторая нормальная форма**

Реляционная форма находится во второй нормальной форме, она находится в первой нормальной форме и никакой не ключевой атрибут не зависит от части ключа.

Функциональная зависимость, в которой часть ключа функционально определяет не ключевой атрибут, называется частичной.

Вторая нормальная форма устраняет частичные зависимости.

Пример:

 Н=НТПЦ, ключ - НП; (см рисунк)

**Определение функциональной зависимости**: Пусть R - отношение . X,Y- произвольные подмножества множества атрибутов R. X функционально опредеяет Y. Y функционально зависит от X. Если для 2 кортежей с одинаковыми значениями атрибутов X , атрибуты Y будут также равны. *X* —> *Y*

*Аномалии*

1. избыточная информация (уменьшение эффективности запроса).

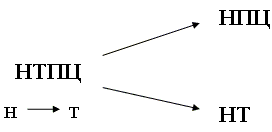
2. Аномалии изменения : бд потенциально может содержать противоречивую информацию -

3. Аномалии вставки :.

4. аномалия удаления:

Аномалии - следствие избыточности.

Приведение ко второй нормальной форме состоит в том, что реляционная схема, содержащая частичные зависимости, декомпозируется на две: первая схема состоит из атрибутов частичной зависимости; вторая - из исходной схемы за вычетом атрибутов из правой части атрибутов функциональной зависимости.

****

### 14.Нормальные формы реляционных схем.3НФ.

Нормализация - это пошаговый процесс устранения избыточности в реляционной схеме.

Нормальные формы - это реляционные схемы, в которых устранен некоторый тип избыточности.

В процессе нормализации реляционная схема будет находится в соответствующей нормальной форме.

**Определение функциональной зависимости**: Пусть R - отношение . X,Y- произвольные подмножества множества атрибутов R. X функционально определяет Y. Y функционально зависит от X. Если для 2 кортежей с одинаковыми значениями атрибутов X , атрибуты Y будут также равны. *X* —> *Y*

Отношение находится в третей нормальной форме, если оно находится во второй нормальной форме, и каждый не ключевой атрибут не транзитивно зависит от первичного ключа.

Пример:

Поставки (Номер Накладной, Банковский счет, Название фирмы) = НБФ; *F =* {Н —> Б, Б —> Ф} - функциональная зависимость; Н —> Б —> Ф - транзитивная зависимость (номер накладной определяет все атрибуты, а банковский счет определяет название фирмы, не являясь ключом)

Аномалии вызываемые транзитивной зависимостью:

• вставки : нельзя включить данные о фирме, если от нее не поступит товар;

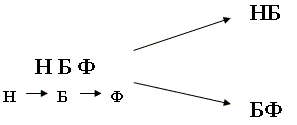
• удаления : если удалить все накладные некоторой фирмы, то пропадет информация о ней;

• изменения: если название фирмы изменилось, то надо сделать изменения во всех накладных этой фирмы.

Приведение к третьей нормальной форме состоит в том, что реляционная схема, содержащая транзитивную зависимость *X* —> *Y* —> *Z* декомпозируется на две: ХУ, YZ.

Пример:

Можно сказать, что схема находится в третьей нормальной форме, если она состоит ключевых атрибутов, которые идентифицируют некоторую сущность и взаимно независимых атрибутов, которые описывают эту сущность. Атрибут называется взаимозависимым, если ни один из них функционально не зависит от какой-либо комбинации остальных атрибутов.



### 15.Многозначные функциональные зависимости.

**Определение функциональной зависимости**: Пусть R - отношение . X,Y- произвольные подмножества множества атрибутов R. X функционально определяет Y. Y функционально зависит от X. Если для 2 кортежей с одинаковыми значениями атрибутов X , атрибуты Y будут также равны. *X* —> *Y*

В базе данных может существовать избыточность, для описания которой не достаточно функциональных зависимостей.

Многозначная зависимость является обобщение функциональной и позволяет описать новый тип избыточности. Благодаря декомпозиции получается четвертая нормальная форма.

Если *X* — > *Y -* функциональная зависимость.

**Определение многозначной зависимости**: Пусть задана реляционная схема R и X,Y - множества атрибутов R. X многозначно определяет Y, или Y многозначно зависит от X, *X (X* — > У), если для любых двух кортежей S и Т таких, что *S[X] = Т[Х]* в отношении г будут существовать два кортежа U и V таких, что 1) *U[X] = V[X] = S[X] = Т[Х}-*

2) *U[Y] = T[Y];* U[R *\X\Y] = S[R\X\Y];*

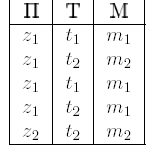
*3)* V[Y]=S[Y] ,V[R *\X\Y] = T[R\X\Y].*

Иначе говоря, если в отношении г имеются кортежи и то в отношении г будут иметься также следующие два кортежа: , .

Пример

Сбыт (Предприятия, Товары, Магазины)=С(П,Т,М);

Будем считать, что если предприятие выпускает какой-либо товар, то оно отправляет его во все магазины, с которыми связано.



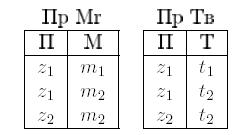
В этой схеме имеется многозначная зависимость П→>Т П→>М . Рассмотрим аномалии, присутствующие в данной схеме:

1. Аномалия вставки: если в таблицу следует занести вставку, что товар изменил название - следует изменить столько кортежей, со сколькими магазинами связано данное предприятие.

2. Аномалия удаления: если из таблицы следует отразить факт, что предприятие перестало выпускать какой-либо товар, то следует удалить столько кортежей, со сколькими магазинами связано данное предприятие.

Аномалии можно устранить следующей декомпозицией:

Из примера видно, что избыточность имеется.



### 14.Нормальная форма реляционных схем. 4НФ

Нормализация - это пошаговый процесс устранения избыточности в реляционной схеме.

Нормальные формы - это реляционные схемы, в которых устранен некоторый тип избыточности.

В процессе нормализации реляционная схема будет находится в соответствующей нормальной форме.

Свойства многозначных зависимостей можно описать аксиомами, аналогично аксиомам Армстронга.

1. *XY \= X(R\X\Y)*

2. *XY, YZ* |= *X(Z\Y)*

Как и в случае функциональных зависимостей, вводится понятие оболочка множества многозначных зависимостей, неприводимого множества многозначнойфункциональных зависимостей и т. д. С их помощью можно дать определение восстановимого соединения.

Затем доказывается следующая теорема:

ТЕОРЕМА.

Пусть R-реляционная схема и *р = {R1,* R2} декомпозиция. Пусть D-множество функциональных и многозначных зависимостей над атрибутами R. Тогда *р* имеет восстановимое соединение тогда и только тогда, когда *,* если эти зависимости принадлежат *D+.* Введем обобщение нормальных форм.

Реляционная схема R находится в 4НФ, если она содержит только такие функциональные многозначные зависимости *XY* что, если многозначная зависимость не тривиальна, т. *е Y⊄X , X∪Y≠R*

и не приводима слева, то ее детерминант X является потенциальным ключом. Если реляционная схема находится в 4НФ, то она находится и в НФБК. Это объясняется тем, что если в реляционной схеме имеется функциональная зависимость *X* —> У, то в ней будет и многозначная функциональная зависимость (тривиальная *XY).*

### 17.Зависимости соединения.5НФ.

До сих пор предполагалось, что для устранения избыточности достаточно декомпозировать схему на две. Такая декомпозиция будет восстановимой. Существуют схемы с избытком, которые нельзя восстановимо декомпозировать на две схемы, но можно восстановимо декомпозировать на три и более. Такие схемы называются n-декомпозированными. Отношение называется три- декомпозируемым, если выполняется следующее условие: Пусть в отношении : присутствуют следующие кортежи:

()

() ( \*)

()

Тогда в отношении будет присутствовать кортеж

()

**Пример**. Рассмотрим опять отношение Сбыт . Сбыт (Предприятие, Товары, Магазины).

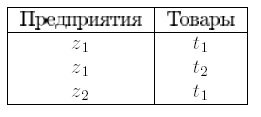
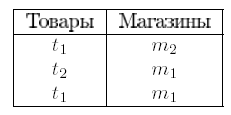
Предполагается, что в этом отношении всегда выполняется следующее условие: Если Предприятие X1 выпускает товар Y1 и Предприятие X1 делает поставки в Магазин Z1, то Предприятие X1, выпуская товар Y1 и поставляет его в Магазин Z1. Это для схемы (\*). Тогда это отношение будет трех декомпозируемым.

|  |  |  |
| --- | --- | --- |
| Предприятия | Товары | Магазины |
| Z1  Z1  Z2  Z1 | T1  T2  T1  T1 | M2  M1  M1  M1 |

Аномалия удаления. Если удаляется кортеж; (), то должен быть удален еще один кортеж;. Неясно какой надо именно удалить кортеж;.

Аномалия вставки. Если вставить кортеж. (), то надо еще добавить ()

Чтобы устранить избыточность, это отношение делится на следующие три:



N-декомпозируемые соединения содержат избыточность. Она описывается зависимостью соединения. Пусть R-отношение, a A,B,...,Z - множество атрибутов

отношения R. Отношение R удовлетворяет зависимости соединения \*(A,B,...,Z) тогда и только тогда, когда оно равносильно соединению своих проекций



Многозначная и функциональная зависимости - это частные случаи зависимости соединения. Зависимость соединения - это наиболее общая форма функциональной зависимости, т. е. для отношений имеющих восстановимое соединение нет более общей зависимости.

Устранение избыточности, порожденной зависимостью соединения, приводит к 5НФ. 5НФ является окончательной в том смысле, что она не содержит аномалий, которые могли бы быть устранены декомпозицией на проекции.

### 18.Трехуровневая архитектура и SQL. Операторы Create Schema и Create View .

Принцип отделения логического уровня представления от физического лёг в основу обобщённой **СБД,** Эта структура получила название трёхуровневая архитектура: в ней имеется три уровня абстракций, на которых рассматривается **СБД:**

1. Концептуальный - описывает логическую структуру всей **БД.**

2. Внутренний - связан со способом хранения информации на физических устройствах хранения.

3. Внешний - связан со способом представления данных для пользователей.

. SQL- комбинация исчисления кортежей и реляционной алгебры.

***В SQL база данных -* это множество схем**. Каждая схема состоит из множества определений таблиц, представлений и прав доступа. Схема определяется следующим образом:

**Create Schema Authorization user [элемент\_ схемы {, элемент\_ схемы}];**

элемент\_ схемы: =определение\_ таблицы | определение\_ вида | права\_ доступа

Создание таблиц осуществляется командой:

Create Table имя\_ таблицы (определение\_ столбца {, определение\_ столбца});

определение\_ столбца:= имя\_ столбца тип\_ данных [NOT NULL];

тип\_ данных:= СНАRАСТЕR[(длина)] | NUMERIC[(precision[, scale])] | INTEGER | SMALLINT | FLOAT | REAL | DOUBLE PRECISION;

***Пример****:* Создать таблицу сотрудников, для которых хранится следующая информация: номер паспорта, год рождения, заработная плата.

Create Table Сотрудники(

Паспорт Character(15),

Фамилия Character(3O),

Имя Character(30),

ГодРождения Numeric(4),

должность Character(40), зарплата Numeric(lO));

Уровню внешнего представления в языке SQL соответствуют логические таблицы (виды), которые задаются командой Create View, которая имеет вид:

**Create View имя\_ вида[(определение\_ столбца{, определение\_ столбца})]**

**As select\_ предложение [With Check Option];**

Внешний уровень позволяет взять реально существующие таблицы и выбрать те данные на которые имеет права пользователь.

***Пример****:* Необходимо для пользователя создать вид, в котором из таблицы сотрудники удалены должность и заработная плата и в котором видны только те сотрудники, которые родились раньше 1975 года.

Create View сотрудники2 (Паспорт Character(15), Фамилия Character(30),

Имя Character(3O), Отчество Character(3O), Год\_ Рождения Numeric(4))

As Select Паспорт, Фамилия, Имя, Отчество, ГодРождения From сотрудники

Where ГодРождения<1975;

With Check Option - если указаны эти ключевые слова, то при дополнении и модификации View будет проверятся принадлежит ли новая запись виду.

*Пример:* При добавлении сотрудника с годом рождения 1980 запись не будет добавляться. На модификацию вида есть определенные ограничения.

*Модификации вида возможны если:*

1. Колонки вида - это колонки из других таблиц (нет вычислений над значениями из других таблиц).

2. В From части упоминаются таблицы, которые можно изменять.

3. Where часть не содержит подзапросов.

4. Нет Group By и Having частей.

### 19. SQL . Реализация правил целостности.

### 20. SQL .Реализация ссылочной целостности внешних ключей.

Правила целостности можно задать в SQL двумя способами:

1. Декларативно

2. Процедурально

1) Декларативно задается при объявлении таблиц. В предложении Create Table. В языке SQL объявляется в виде ограничений. Ограничения могут задаваться для всей таблицы или для столбца. Ограничения имеют синтаксис:

[Constraint имя\_ ограничения] ограничение

*Ограничения бывают:*

1. ***Unique*** *-* определение ключа-кандидата. Выделенный в ограничение столбец либо группа, должны содержать значения встречающиеся в таблице единственный раз. Если ключ-кандидат охватывает несколько столбцов, то его необходимо пометить в конце таблицы.

#### Для таблицы :

#### CREATE TABLE АУДИТОРИИ( КОРПУС CHAR(2), НОМЕР CHAR(4), ТИП CHAR(5), UNIQUE (КОРПУС, НОМЕР) )

**Для столбца** CREATE TABLE СОТРУДНИКИ(НомерПаспорта CHAR(20) UNIQUE, ФАМИЛИЯ CHAR(30),)

**2) *Primary Key****.* В таблице может быть единственный первичный ключ. Задается в двух формах: для таблицы, для столбца.

***3) Foreign Key*** *-* объявление внешнего ключа. Объявляет группу столбцов таблицы внешним ключом, который ссылается на другую таблицу. Допустима только форма задания для всей таблицы (для столбца отсутствует).

Foreign Key (перечень\_ столбцов) Reference имя\_ таблицы

[(перечень\_ столбцов)] ссылочная\_ спецификация;

Перечень столбцов можно не указывать если имена совпадают в первой и второй таблице. Ссылочная спецификация задает, что будет происходить при изменении (OnUpdate) или при удалении (OnDelete) родительского ключа. Возможные опции:

Cascade - распространяет удаление на все подчиненные. Если изменяется, то изменяются и все подчиненные.

SetNull - установить пустое значение в подчиненных записях.

SetDefault - установить значения по умолчанию в подчиненных записях.

**4) *Check*** *-* предикат проверяет выполнение условия. Можно устанавливать для столбца и для таблицы.

*Пример:* Create Table A(X Integer Check(X>=0), Y Integer Check(y>=0), Constraint MyCheck Check(x+y>0));

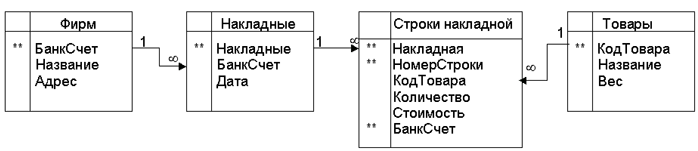
Первые две конструкции Check - для столбцов, третья - для таблицы. Foreign Key - отношение один ко многим. А как создать отношение один к одному?

Для этого:

1. В одной из таблиц группа полей объявляется внешним ключом (For­eign Key).

2. Потом объявляется Unique.

*Пример:* Пусть имеются следующие таблицы.



Описание таблиц на SQL имеет вид:

Create Table Фирмы(БанкСчет Char(12) Primary Key, Название Char(20), Адрес Char(40));

Create Table Товары(КодТовара Char(15) Primary Key, Название Char(30), Bee Decimal(lO));

Create Table Накладные (Накладные Char(20), БанкСчет Char(12), Дата Char(lO), Primary Кеу(Накладные,БанкСчет), Foreign Key (БанкСчет), Ref­erences Фирмы OnUpdate Cascade OnDelete Cascade);

Create Table СтрокиНакладной(БанкСчет Char(12), Накладная Char(20), НомерСтроки Decimal(3), КодТовара Char(15), Количество Decimal(lO), Стоимость Decimal(lO), Primary Кеу(БанкСчет,Накладные, НомерСтроки), Foreign Key (КодТовара), References Товары, Foreign Кеу(БанкСчет,Накладные), References Накладные);

### 21. SQL .Реализация Соединения таблиц в операторе SELECT.

**SELECT оператор**

С его помощью можно выбрать данные из СУБД. Данные можно выбрать из нескольких таблиц. Оператор SELECT имеет следующий вид:

Select [Distinct] список \_ столбцов From источники

Where предикат Group By список\_ столбцов

Having предикат Order By {Столбец [Asc|Desc]};

*Результат запроса -* таблица, столбцы которой заданы в части Select.

1. Часть *From* определяет одну либо несколько таблиц, из которых извлекаются данные.

2. В части *Where* определяются критерии, которым должны удовлетворять строки, появляющиеся в итоговой таблице.

3. Часть *Group By* группирует выходные данные по одинаковым значениям перечисленных столбцов. Часть Group By применяется для выполнения агрегатных функций над группами строк.

4. Часть *Having* определяет критерии, которым должны удовлетворять группы строк, формируемые предложением Group By.

5. С помощью *Order By* можно расположить результаты запроса в определенном порядке.

6. Если задано ключевое слово *Distinct,* то если в результате выполнения запроса появляются одинаковые строки, то дубликаты убираются

*Пример 1:* Выбрать все товары вес которых больше 1 кг.

Select \* From Товары Where вес>1;

или

Select КодТовара, Название From Товары Where вес>1;

В первом случае \* значит выбрать все столбцы таблицы. Во втором случае указаны конкретные имена столбцов.

*Пример 2:* Выбрать названия товаров, код накладной, номер в строке для всех строк накладной в которых более 1000 единиц товаров.

Select СтрокиНакладной.Накладная, СтрокиНакладной.НомерСтроки,

Товары.Название From СтрокиНакладной, Товары

Where СтрокиНакладной.КодТовара=Товары.КодТовара

And СтрокиНакладной.Количество>1000;

*Альтернативный синтаксис* для соединения - использование ключевого слова Join в части From.

TablA [Natural] [тип\_ соединения] Join TablB.

В обеих таблицах выбираются столбцы с одинаковыми именами. В итоге помещаются только те строки из декартового соединения в которых столбцы с одинаковыми именами имеют одинаковые значения.

*Типы соединения:*

*• Inner -* соединяются только те строки, где найдены совпадения значений (используется по умолчанию).

• *Left -* результат все строки из tablA + строки из tablB, для которых значения совпадают.

• *Right -* все строки из tablB + строки из tablA с совпадающими значениями.

3. Соединение посредством предиката.

TabА [тип\_ соединения] ТаbВ On предикат

В соединение попадают только те пары строк, которые удовлетворяют предикату. Вернемся к примеру 2:

*Пример:*

Select Накладная, НомерСтроки, Название

From СтрокиНакладной Join Товары Where Количество>1000;

*Пример 1:* Вычислить вес товаров, перечисленных в строках накладной (для каждой).

Select Накладная, НомерСтроки, Название, Количество\*Вес As

Вес\_ В\_ Строке From СтрокиНакладной Join Товары;

*Пример 2:* Напечатать все номера накладных в которых упоминается "сахар".

Select Distinct Накладная From СтрокиНакладной Join Товары

Where Название="сахар";

### 22. SQL .Коррелированный и не коррелированный подзапрос.

*Пример :* Выдать цену сахара и его количество для каждой строки накладной.

Where Название="сахар";

или

Select Количество, Стоимость From СтрокиНакладной

Where КодТовара= (Select КодТовара From Товары Where Название="сахар")

Это пример запроса с подзапросом. Сначала выполняется подзапрос, а затем основной запрос. Подзапрос должен при использовании его в операторах сравнения возвращать единственное значение в единственном столбце.

**В корелированном подзапросе есть ссылки на таблицу, которая имеется во внешнем запросе.**

*Пример 1:* Найти все фирмы которые сделали поставки 11.11.2002.

Вариант 1 (не коррелированный):

Select distinct Название From Фирмы Join Накладные

Where data=11.11.02;

Вариант 2 (корелированный):

Select Название From Фирмы Where 11.11.02 In

(Select Дата From Накладные Where Накладные.БанкСчет=Фирмы.БанкСчет~

*Порядок выполнения запроса следующий:*

Сначала выбирается какаялибо фирма. Для нее выполняется подзапрос в котором находится множество дат, в которые фирма делала поставки. Затем проверяется условие (то что заданная 11.11.02 находится в этом множестве) и решается поступит ли фирма в окончательное множество или нет. Эта процедура выполняется для каждой фирмы. Коррелированные подзапросы сильно влияют на производительность, но иногда без них трудно обойтись.

*Пример 2:* Выдать названия фирм, которые ничего не поставили.

Select Название From Фирмы Where Not Exists ( Select \* From Накладные Where БанкСчет=Фирмы.БанкСчет);

Предикат Exists(Not Exists) проверяет является ли множество, возвращаемое подзапросом не пустым (пустым).

*Пример 3:* Выбрать названия товаров у которых названия одинаковые, а коды разные.

Select Distinct Название From Товары Where Exists(

Select \* From Товары Товары2 Where Товары.Название=Товары2.Название

And Товары.Код<>Товары2.Код);

Новая синтаксическая конструкция - переименование (alias).

Если необходимо, например, соединить саму с собой таблицу, то возникает следующая проблема: необходимо синтаксически различать 2 различных ее появления. Для этого и используется переименование. В нашем примере товары в коррелированном подзапросе переименованы в Товары2. Теперь можно различать таблицу товары из внешнего запроса и подзапроса.

*Пример 4-'* Выдать количество поставок, которые фирмы сделали в сентябре.

Select Название, Count (Накладная) As КоличествоПоставок

From Фирмы Join Накладные Where Дата Between 01.09.02 And 31.09.02

Group By БанкСчет;

Рассмотрим множество строк, которое возвращается оператором Select без части Group By. Все это множество разбивается на подмножества таким образом, что все записи с одинаковым банковским счетом попадают в одно подмножество. Над каждым подмножеством выполняется агрегирующая функция (для этого примера - количество элементов подмножества) результат которой и возвращается в запросе.

*Пример 5:* Расчитать вес поставки.

Select Накладная, Sum(Количество\*Вес) As ВесПоставки

From СтрокиНакладной Join Товары Group By Накладные;

*Пример 6:* Для каждой фирмы подсчитать число поставок вес которой превышает одну тонну.

Select Название, Count (Накладные) As КоличествоБолыпихПоставок

From Фирмы Join Накладные Where Накладная In (

Select Накладная From СтрокиНакладной Join Товары Group By Накладная

Having Sum (Количество\*Вес)> 1000)

Group By БанкСчет;

В *Having* части записываются те условия, в которых используется агрегирующая функция.

### 23.SQL,Оператор Select.Функции подведения итогов. ?

С его помощью можно выбрать данные из СУБД. Данные можно выбрать из нескольких таблиц. Оператор SELECT имеет следующий вид:

Select [Distinct] список \_ столбцов From источники

Where предикат Group By список\_ столбцов

Having предикат Order By {Столбец [Asc|Desc]};

*Результат запроса -* таблица, столбцы которой заданы в части Select.

1. Часть *From* определяет одну либо несколько таблиц, из которых извлекаются данные.

2. В части *Where* определяются критерии, которым должны удовлетворять строки, появляющиеся в итоговой таблице.

3. Часть *Group By* группирует выходные данные по одинаковым значениям перечисленных столбцов. Часть Group By применяется для выполнения агрегатных функций над группами строк.

4. Часть *Having* определяет критерии, которым должны удовлетворять группы строк, формируемые предложением Group By.

5. С помощью *Order By* можно расположить результаты запроса в определенном порядке.

6. Если задано ключевое слово *Distinct,* то если в результате выполнения запроса появляются одинаковые строки, то дубликаты убираются

*Пример 1:* Выбрать все товары вес которых больше 1 кг.

Select \* From Товары Where вес>1;

или

Select КодТовара, Название From Товары Where вес>1;

В первом случае \* значит выбрать все столбцы таблицы. Во втором случае указаны конкретные имена столбцов.

EXISTS - это оператор, который производит верное или неверное значение, Это означает что он может работать автономно в предикате или в комбинации с другими выражениями. Он берет подзапрос как аргумент и оценивает его как верный если тот производит любой вывод или как неверный если тот не делает этого.

Этим он отличается от других операторов предиката, в которых он не может быть неизвестным.

Not Exists Предыдущий пример дал понять что EXISTS может работать в комбинации с операторами Бул. Конечно, то что является самым простым способом для использования и вероятно наиболее часто используется с EXISTS - это оператор NOT.

Запросы могут производить обобщенное групповое значение полей точно также как и значение одного пол. Это делает с помощью агрегатных функций. Агрегатные функции производят одиночное

значение для всей группы таблицы. Имеется список этих функций:

COUNT -производит номера строк или не-NULL значения полей которые выбрал запрос.

SUM -производит арифметическую сумму всех выбранных значений данного пол.

AVG -производит усреднение всех выбранных значений данного пол.

MAX -производит наибольшее из всех выбранных значений данного пол.

MIN производит наименьшее из всех выбранных значений данного пол.

### 24.SQL,Оператор Select.Предложение Having.

**SELECT оператор**

С его помощью можно выбрать данные из СУБД. Данные можно выбрать из нескольких таблиц. Оператор SELECT имеет следующий вид:

Select [Distinct] список \_ столбцов From источники

Where предикат Group By список\_ столбцов

Having предикат Order By {Столбец [Asc|Desc]};

*Результат запроса -* таблица, столбцы которой заданы в части Select.

1. Часть *From* определяет одну либо несколько таблиц, из которых извлекаются данные.

2. В части *Where* определяются критерии, которым должны удовлетворять строки, появляющиеся в итоговой таблице.

3. Часть *Group By* группирует выходные данные по одинаковым значениям перечисленных столбцов. Часть Group By применяется для выполнения агрегатных функций над группами строк.

4. Часть *Having* определяет критерии, которым должны удовлетворять группы строк, формируемые предложением Group By.

5. С помощью *Order By* можно расположить результаты запроса в определенном порядке.

6. Если задано ключевое слово *Distinct,* то если в результате выполнения запроса появляются одинаковые строки, то дубликаты убираются

*Пример 1:* Выбрать все товары вес которых больше 1 кг.

Select \* From Товары Where вес>1;

или

Select КодТовара, Название From Товары Where вес>1;

В первом случае \* значит выбрать все столбцы таблицы. Во втором случае указаны конкретные имена столбцов.

*Пример 2:* Выбрать названия товаров, код накладной, номер в строке для всех строк накладной в которых более 1000 единиц товаров.

Select СтрокиНакладной.Накладная, СтрокиНакладной.НомерСтроки,

Товары.Название From СтрокиНакладной, Товары

Where СтрокиНакладной.КодТовара=Товары.КодТовара

And СтрокиНакладной.Количество>1000;

Having может использовать аргументы которые имеют одно значение на группу вывода

*Пример 6:* Для каждой фирмы подсчитать число поставок вес которой превышает одну тонну.

Select Название, Count (Накладные) As КоличествоБолыпихПоставок

From Фирмы Join Накладные Where Накладная In (

Select Накладная From СтрокиНакладной Join Товары Group By Накладная

Having Sum (Количество\*Вес)> 1000)

Group By БанкСчет;

В *Having* части записываются те условия, в которых используется агрегирующая функция.

### 25.SQL,Операторы изменения данных.

Добавление записей осуществляется оператором *INSERT:*

*Синтаксис:*

Insert Into имя\_ таблицы[(имя\_ столбца,...)] запрос | конструктор значений

*Пример 1:* Форма с конструктором значений.

Insert Into Товары(Код,Название,Вес) Values(2367,"Сахар",!);

*Пример 2:* Форма с выражением запроса.

Insert Into Товары Select КодТов,Назв,Вес From СписокТоваров Where Назв Like "С\*";

Этот запрос копирует записи из одной таблицы в другую.

Оператор удаления записей *Delete*

*Синтаксис:*

Delete From имя\_ таблицы [Where предикат];

*Пример:* Delete From Накладные Where data<"05.05.1995";

*Оператор модификации данных*

*Синтаксис:*

Update имя\_ таблицы set имя\_ столбца=выражение,... Where предикат;

*Пример:* Update Товары Set Ha3BaHHe="Caxap"Where Название="Сахар";

# Практический раздел

ИНДИВИДУАЛЬНЫЕ ПРАКТИЧЕСКИЕ РАБОТЫ, ИХ ХАРАКТЕРИСТИКА

### Введение

По результатам работы студентом должен быть представлен и защищен отчет. Содержание отчета включает:

1. Введение, содержащее постановку задачи.
2. Распечатку программы.
3. Вывод.

### ИПР №1

**PL/SQL: переменные, анонимный блок.**

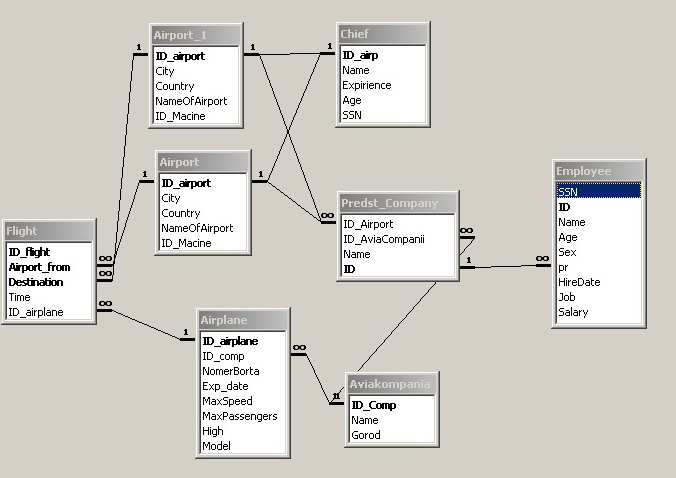
**Цель:** Целью работы является изучение основных элементов программы на PL/SQL, приобретение практических навыков написания простейших программ, использующих выборку данных из таблиц БД.

**Общая постановка задачи:** Применить основные элементы программы на PL/SQL, написать простейшие программы, использующие выборку данных из таблиц БД.

**Практическая часть:**

**Вариант 1**

«Аэропорт»



1. Создайте и выполните пустой анонимный блок.
2. Добавьте в блок команду вывода строки «This is my first string»
3. Добавьте декларативную секцию и объявите переменную just\_a\_number числового типа, присвоив ей значение по умолчанию 10; объявите константу типа строка, присвоив ей значение “Hello world!”.
4. Объявите следующие переменные: две переменные с типом столбца Name таблицы Employee, две переменные с типом столбца HireDate таблицы Employee , переменную-запись с типом строки таблицы Employee.
5. С помощью неявного курсора получите в переменные c типом столбца Name и HireDate таблицы Emloyee соответствующие значения из строки со значением поля ID равным 1 и 3, а в переменную с типом строки таблицы Employee — значение строки со значением поля ID равным 2.
6. Необходимо вывести следующую строку используя переменные из предыдущего задания :

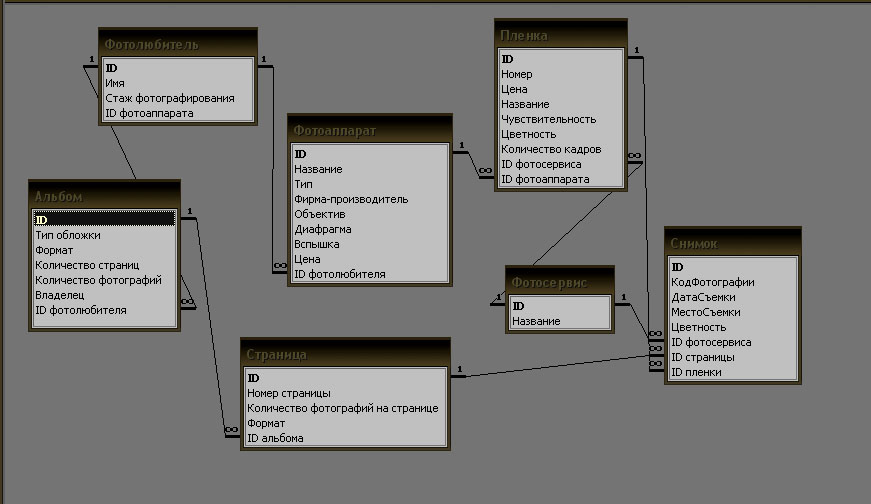
“Разница в трудоустройстве [имя\_1] и [имя\_2] составляет [дней]”

Где вместо [имя\_1] требуется подставить соответствующее значение имени сотрудника, у которого ID равен 1; [имя\_2] следует подставить имя сотрудника, у которого ID равен 3. А вместо [дней] вычислить и вывести количество дней, на сколько один сотрудник работает больше или меньше другого.

1. Как осуществить действия пунктов 5-6 с помощью только SQL-запроса?
2. Требуется удалить сотрудников, у кого заработная плата составляет менее 100 у.е. Выполните внутри блока DELETE таблицы Employee.
3. Добавьте команду, которая выводит количество строк, обработанных запросом из п.8.

**Вариант 2**

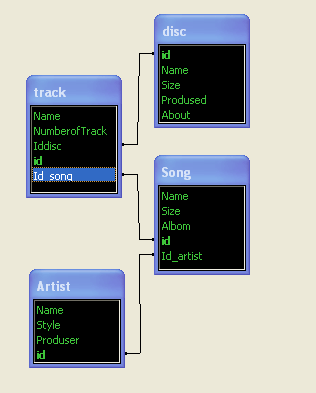
«Фотоальбом»

****

1. Создайте и выполните пустой анонимный блок.
2. Добавьте в блок команду вывода строки «Hello world»
3. Добавьте декларативную секцию и объявите переменную Price числового типа, присвоив ей значение по умолчанию 100; объявите константу типа строка.
4. Объявите следующие переменные: переменную с типом ShootingPlace, переменную с типом ShootingDate, переменную-запись с типом всей строки из таблицы Photograph.
5. С помощью неявного курсора получите в переменные ShootingPlace и ShootingDate таблицы Photograph соответствующие значения из строки со значением ID, равным 2, а в переменную-запись — значение строки со значением ID, равным 3.
6. Выведите строку типа: Между съемками в местах [место1] и [место2] прошло [разница между датами].(Комбинация двух текстовых значений из переменной ShootingPlace и переменной-записи и разница значений из переменной ShootingDate и переменной-записи)
7. Как осуществить действия пунктов 5-6 с помощью только SQL-запроса?
8. Выполните внутри блока команду DELETE для строк с датой съемки раньше 1 января 2000 года.
9. Выведите количество строк, обработанных запросом из п.8.

**Вариант 3**

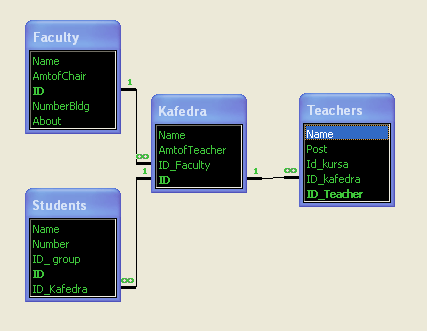
«Музыкальные диски»

****

* 1. Создайте и выполните пустой анонимный блок.
  2. Добавьте в блок команду вывода строки «Hello world»
  3. Добавьте декларативную секцию и объявите переменную Size числового типа присвоив ей значение по умолчанию 512; объявите констану Artname типа VARCHAR2(10).
  4. Объявите следующие переменные: переменную Discname с типом поля Name таблицы Disc, переменную Disccost с типом поля Cost таблицы Disc, переменную-запись Discrecord с типом всей строки из той же таблицы
  5. С помощью неявного курсора получите в переменные Discname и Disccost соответствующие значения из строки со значением id равным 3, а в переменную Discrecord — значение строки со значением поля id равным 1
  6. Выведите следующую строку «Чек на суму 12000 включает в себя цену за диск Nightwish и Green\_Day».
  7. Как осуществить действия пунктов 5-6 с помощью только SQL-запроса?
  8. Выполните внутри блока команду обновления (UPDATE) для дисков с размером больше 680 изменить цену на 7000.
  9. Выведите количество строк, обработанных запросом из п.8.

**Вариант 4**

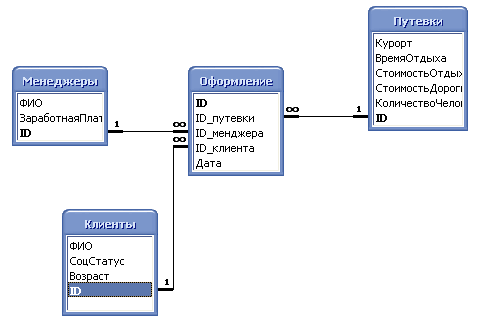
«Университет»

****

1. Создайте и выполните пустой анонимный блок.
2. Добавьте в блок команду вывода строки «Есть надо вкусно и полезно».
3. Добавьте декларативную секцию и объявите переменную Нечто числового типа, присвоив ей 275, как значение по умолчанию; объявите константу типа дата, присвоив ей значение – 12 июля 1985 года.
4. Объявите переменную ReciepeAuthor с типом столбца Автор из таблицы Рецепт, переменную Time из столбца Время Приготовления и переменную-запись ReciepeInfo из этой же таблицы.
5. С помощью неявного курсора получите в переменные ReciepeAuthor и ReciepeInfo соответствующие значения из таблицы Рецепт из строк с названиями рецептов «Борщ» и «Плов», а ReciepeInfo – значение строки с названием «Манты».
6. Выведите строку «Блюда авторов [ReciepeAuthor1] и [ReciepeAuthor2] готовятся за [TimeElapsed] минут», используя переменные из предыдущего задания, а вместо TimeElapsed выведите сумму времени приготовления двух блюд.
7. Как осуществить действия пунктов 5-6 с помощью только SQL-запроса?
8. Всем авторам, рожденным после 1984 года, добавьте по 10 минут к времени приготовления их рецептов, внутри созданного блока выполните UPDATE для поля ВремяПриготовления таблицы Рецепт.
9. Выведите количество строк, обработанных запросом из пункта 8.

**Вариант 5**

«Турфирма»

****

1. Создайте и выполните пустой анонимный блок.
2. Добавьте в блок команду вывода строки «Увидеть Париж и умереть».
3. Добавьте декларативную секцию и объявите переменную SomeNumber числового типа, присвоив ей 512, как значение по умолчанию; объявите константу типа дата, присвоив ей значение – 20 февраля 2001 года.
4. Объявите переменную ClientName с типом столбца ФИО из таблицы Кленты, переменную Age из столбца Возраст и переменную-запись ClientInfo из этой же таблицы.
5. С помощью неявного курсора получите в переменные ClientName и ClientInfo соответствующие значения из таблицы Клиенты для клиентов с ID равными 7 и 12, а ReciepeInfo – значение строки для клиента с ID 15.
6. Выведите строку «Клиенты [ClientName1] и [ClientName2] имеют средний возраст [AverAge]», используя переменные из предыдущего задания, а вместо AverAge выведите средний возраст двух клиентов.
7. Как осуществить действия пунктов 5-6 с помощью только SQL-запроса?
8. Всем клиентам, у которых возраст меньше 21-го года измените СоцСтатус на «холост/не замужем», выполните UPDATE для поля СоцСтатус таблицы Клиенты.
9. Выведите количество строк, обработанных запросом из пункта 8.

### ИПР №2

**PL/SQL: курсоры, циклы, условия.**

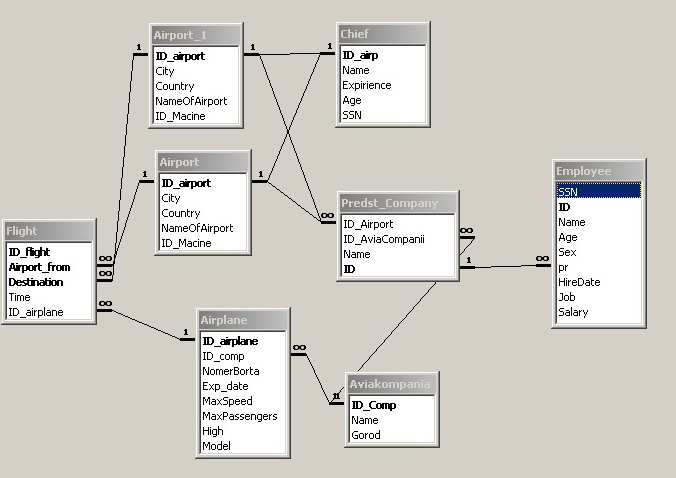
**Цель:** Целью работы является изучение основных управляющих конструкций программы на PL/SQL, приобретение практических навыков написания программ, использующих ветвление и циклическую обработку и последовательное извлечение данных многострочных выборок с использованием явного курсора.

**Общая постановка задачи:** Применить основные управляющие конструкций программы на PL/SQL, написать программы, использующие ветвление и циклическую обработку и последовательное извлечение данных многострочных выборок с использованием явного курсора.

**Практическая часть:**

**Вариант 1**

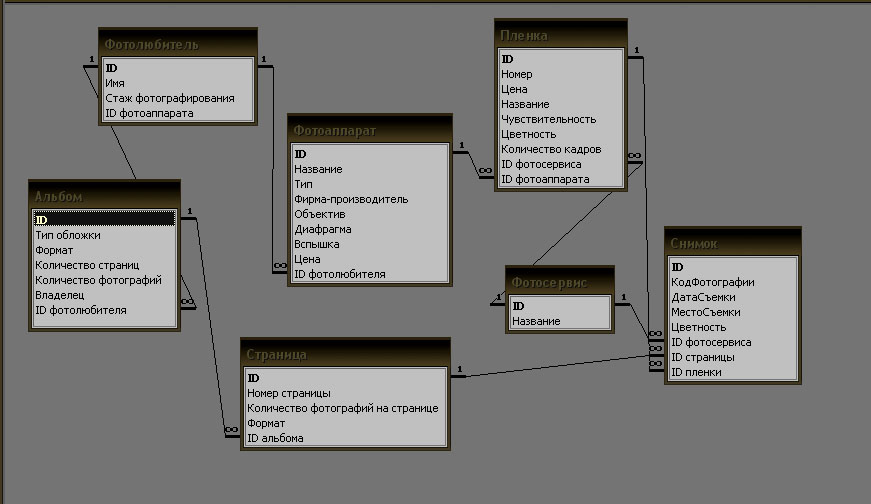
«Аэропорт»



1. Напишите анонимный блок, вычисляющий значение на основании значений строки, задаваемой значением первичного ключа, таблицы Employee в соответствии с приведенным алгоритмом: если работник проработал менее 10 лет, вывести сообщение “[Name] - young employee ” . Если же сотрудник проработал 10 и более лет, вывести сообщение “[Name] – old employee”. Значение первичного ключа запросите через подстановочную переменную. Выведите на экран полученную строку и стаж сотрудника в формате “[кол\_во\_дней]:[месяцев]:[лет]”.
2. Используя явный курсор, перепишите блок так, чтобы значение вычислялось последовательно для строк, в которых возраст сотрудника составляет подстановочную переменную. Используйте явное открытие/закрытие/выборку данных из курсора.
3. Перепишите блок, используя курсорный цикл FOR.

**Вариант 2**

«Фотоальбом»

****

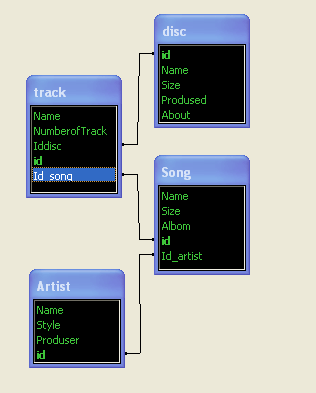
1. Напишите анонимный блок, вычисляющий цену пленки с различной надбавкой в зависимости от количества кадров, на основании значений строки, задаваемой значением первичного ключа таблицы Film в соответствии с приведенным алгоритмом:

Если в названии пленки фигурирует слово «Kodak», то цена пленки увеличивается на 10 единиц за каждые три кадра сверх 24-х. Для пленок с иными названиями цена пленки увеличивается в 1,5 раза, если кадров больше 24-х. Значение первичного ключа запросите через подстановочную переменную. Выведите на экран названия пленок и соответствующие вычисленные значения.

1. Используя явный курсор, перепишите блок так, чтобы цена пленки с различной наценкой в зависимости от количества кадров вычислялась последовательно для строк со значением поля Sensibility из подстановочной переменной. Используйте явное открытие/закрытие/выборку данных из курсора.
2. Перепишите блок используя курсорный цикл FOR.

**Вариант 3**

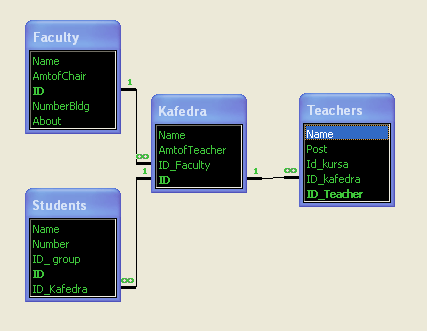
«Музыкальные диски»

****

1. Напишите анонимный блок, вычисляющий цену диска на основании значений строки, задаваемой значением первичного ключа таблицы Disc в соответствии с приведенным алгоритмом: Если поле About содержит значение «пиратская версия» то цену диска уменьшить в полтора раза, если у поля Prodused значение «NewCD» то цену диска увеличить на 30%. Значение первичного ключа запросите через подстановочную переменную. Выведите на экран Название диска и его цену.
2. Используя явный курсор, перепишите блок так, чтобы цена диска вычислялась последовательно для дисков сделанных в промежутке между двумя датами, которые запросите через подстановочные переменные. Используйте явное открытие/закрытие/выборку данных из курсора.
3. Перепишите блок используя курсорный цикл FOR.

**Вариант 4**

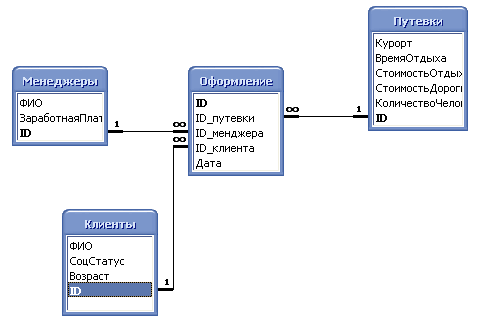
«Университет»

****

1. Напишите анонимный блок, вычисляющий размер стипендии на основании значений строки, задаваемой значением первичного ключа, таблицы Students в соответствии с приведенным алгоритмом: Если средний бал превышает 4 бала то размер стипендии увеличивается в 1.5 раз. В случае если средний бал ниже 3 то стипендия уменьшается в 1.2 раза, а если Фамилия студента «Иванов» то размер стипендии не меняется в любом случае. Значение первичного ключа запросите через подстановочную переменную. Выведите на экран Фамилию студента и его размер стипендии.
2. Используя явный курсор, перепишите блок так, чтобы размер стипендии вычислялся последовательно для студентов , которые учатся в определенной группе .Номер группы запросите через подстановочную переменную. Используйте явное открытие/закрытие/выборку данных из курсора.
3. Перепишите блок используя курсорный цикл FOR.

**Вариант 5**

«Турфирма»

****

1. Напишите анонимный блок, вычисляющий результат на основании значений строки, задаваемой через первичный ключ таблицы Рецепт, по следующей схеме:  
   - если год рождения автора меньше 1984, записать как “[Автор] – профессионал по стажу”;  
   - если же больше или равный 1984, записать как “[Автор] – новичок”;  
   - значение первичного ключа запросите через подстановочную переменную;  
   - также отобразите возраст автора.
2. Используя явный курсор перепишите блок так, чтобы значение вычислялось последовательно для строк, в которых время приготовления блюда задается подстановочной переменной. Используйте явное открытие/закрытие/выборку данных из курсора.
3. Перепишите блок, используя курсорный цикл FOR

## КОНТРОЛЬНЫЕ РАБОТЫ

### *Контрольное работа*

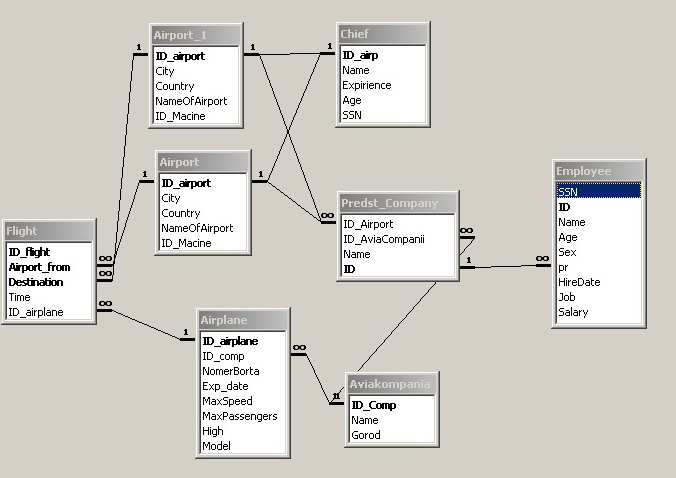
SQL: подзапросы, сложные запросы

**Общая постановка задачи:**

Применить подзапросы в операторе SELECT для построения сложных выборок.

**Вариант 1**

«Аэропорт»



1. Вывести сотрудников, которые были приняты на работу после принятия на работу сотрудника, имя которого “Ivan Petrov”.

2. Найти должность сотрудника, у которой средняя заработная плата такая же, как и у всех сотрудников из таблицы Employee.

3. Вывести название представительства авиакомпании, в котором работают только лица женского пола и возрастом до 30 лет.

4. Вывести имя и номер социального страхования сотрудника , у которого заработная плата меньше средней заработной платы сотрудников таблицы Employee и возраст превышает средний возраст сотрудников.

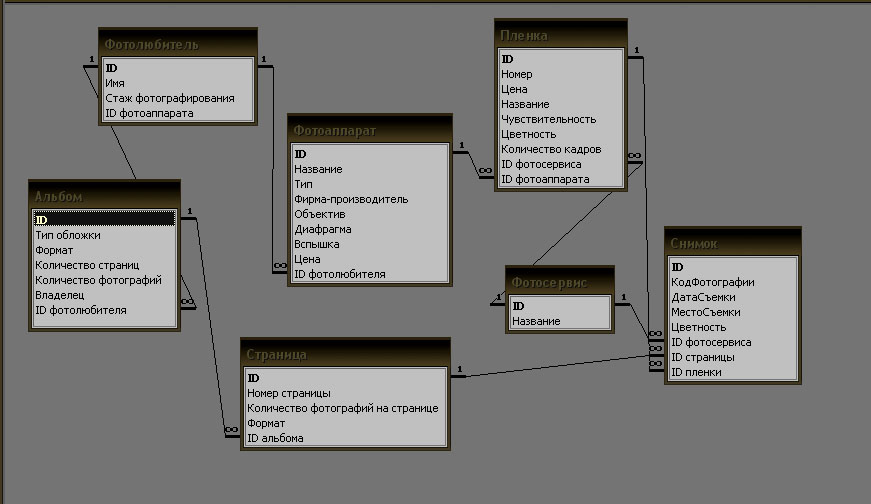
5. Вывести сотрудников, у которых заработная плата отличается от заработной платы любого из сотрудников должности “manager”.

6. Вывести имя и номер социального страхования сотрудников, которые были приняты на работу в один и тот же день и получают одинаковую заработную плату с сотрудником, который занимает должность “manager”.

7. Вывести имя, возраст, пол и дату приема на работу сотрудника, у которого заработная плата превышает среднюю заработную плату сотрудников таблицы Employee.

**Вариант 2**

«Фотоальбом»

****

1. В таблице Photograph найти строки со значением столбца ShootingPlace таким же, как у снимков с датой съемки позже 1 января 2006.

2. Вывести название самой дорогостоящей пленки.

3. Найти фотоаппараты, у которых нет владельца по имени Василий.

4. Отобразить строки из таблицы Film, у которых цена ниже средней, и а в названии есть слово «Kodak».

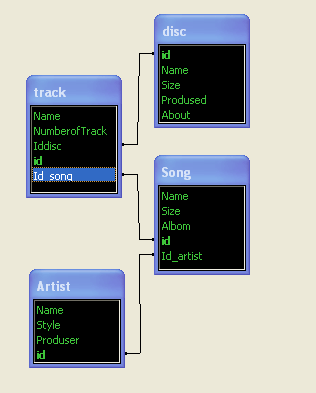
5. Выбрать альбомы, в которых число фотографий отлично от числа фотографий в альбомах Василия, считая пустые значения поля NumberOfPhotos за 0.

6. Выбрать пленки, которые имеют такое же значение цены и количества кадров, что и пленка с названием «Konica».

7. Вывести название, количество кадров, чувствительность пленки, цена которой не превышает среднюю цену пленки среди пленок с такой же чувствительностью. Используйте запрос с подзапросом в секции FROM.

**Вариант 3**

«Музыкальные диски»

****

1. Выбрать название диска у которого фирма-производитель отличается от фирмы которая сделала диск Blink\_182 .

2. Выбрать название и минимальную цену для дисков у которых минимальная цена больше чем у дисков сделанных фирмой newCD.

3. Выбрать название дисков у которых размер отличается от размера дисков у которых в поле About написано «пиратская серия».

4. Выбрать название дисков и фирму-производитель для тех дисков у которых размер меньше чем у дисков фирмы newCD, а цена больше чем у дисков у которых в поле About написано «пиратская серия».

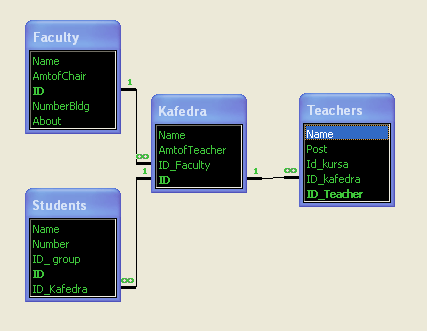
5. Выбрать название для дисков, у которых цена отличается от цены любого из дисков фирмы «Звездная\_серия».

6. Выбрать название всех дисков у которых размер и цена такая же как у дисков фирмы newCD.

7. Выбрать название и размер для всех дисков имеющих больший размер чем средний размер для дисков сделанной той же фирмой (использую подзапрос в секции WHERE).

**Вариант 4**

«Университет»

****

1. Выбрать название факультетов расположенных по тому же адресу что и что и факультет под номером 2 .

2. Вывести группы студентов с максимальной суммой стипендии большей чем у всех студентов факультета КСиС.Запрос сгруппировать по номеру группы.

3. Выбрать названия кафедр где не учится студент «Иванов».

4. Выбрать название факультета где количество кафедр больше чему у факультета ФРЭ , а адрес такой же как у факультета КСиС.

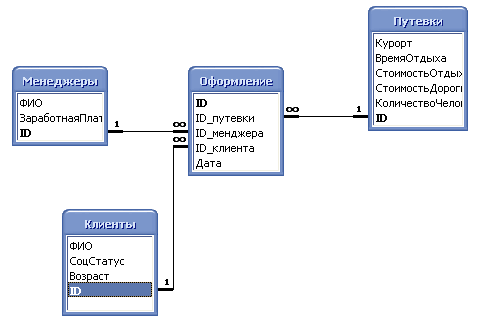
5. Выбрать студентов, у которых стипендия отличается от стипендии любого из студентов в группе 252002.

6. Выбрать всех студентов, которые учатся в той же группе и получают такую же стипендию что и студенты имеющие средний бал 5.

7. Выбрать имя и размер стипендии для всех студентов имеющих большую стипендию чем средняя стипендия на их кафедре(использую подзапрос в секции WHERE).

**Вариант 5**

«Турфирма»

****

1. Выведите имена всех менеджеров, у которых зарплата такая же как и у Розенкранца.

2. выведите названия курортов и имена менеджеров, у которых зарплата выше средней.

3. Выведите список курортов, на которые не ездят холостяки.

4. Выведите информацию о клиентах, которых обслуживали менеджеры с зарплатой больше чем у Горацио и стоимостью отдыха меньше 1000 у.е.

5. Выведите информацию о менеджерах и о клиентах, которые не оформляли путевок в дни, когда это делал Карпович.

6. Выведите списки клиентов для всех менеджеров, у которых оклад такой же как и у Горацио, и которые оформляли путевки с ним в те же дни.

7. Выведите список курортов, на которых побывало больше половины клиентов, и стоимость отдыха на которых стоит больше средней цены (подзапрос должен находиться в секции FROM).

## Задание на курсовую работу, ее характеристика.

1. Программа и пользовательский интерфейс работы с БД по теме лабораторных работ («Аэропорт», «Университет», «Библиотека» и т.д.)
2. Сравнительный анализ быстродействия современных серверных СУБД.
3. Сравнительный анализ областей применения серверных СУБД.
4. Сравнительный анализ безопасности современных серверных СУБД.
5. Программа-wizard визуального создания/изменения схемы БД / заполнения БД данными.
6. Программа экспорта/импорта различных форматов БД (Access<->Oracle, Oracle<-> MS SQL Server и т.д.)