УДК 004(075.8) Д-931

ББК з973-083я73

Дьяков И.А.

Иванова О.Г.

«Проектирование баз данных»

для студентов всех форм обучения

Тамбов 2017

ВВЕДЕНИЕ

Развитие компьютерной техники и программного обеспечения приводит к увеличению объёмов хранимой информации. Одновременно возрастают требования к быстродействию информационных систем, оперативности доступа к данным. Основополагающими категориями в базах данных являются модели данных и модели доступа к данным. Правильное выполнение этапов проектирования баз данных не гарантирует получение целостной и работоспособной базы. Оптимизация данных позволяет получить эффективные, неизбыточные с точки зрения данных базы. Таким образом, в жизненном цикле баз данных, на этапе разработки, необходимо выполнять оптимизационные процедуры, основанные на теоретико-множественных и специальных операторах.

Учебное пособие состоит из двух частей, где рассматриваются теоретические основы и практические приемы разработки баз данных. В первой части основное внимание уделено теории баз данных, их моделям и математическим основам. Во второй части рассматривается применение специального программного обеспечения для машинной реализации баз данных.

Предназначено для студентов вузов, обучающихся по направлениям: 09.03.01 – «Информатика и вычислительная техника», 09.03.02 – «Информационные системы и технологии», 27.03.03 – «Системный анализ и управление», 27.03.04 – «Управление в технических системах», 38.03.05 – «Бизнес-информатика»

В результате освоения дисциплины «Базы данных» у обучающихся должны быть сформированы *общепрофессиональные компетенции ОПК-3* и *профессиональные компетенции ПК-6* (табл. 1).

Таблица 1 –Формируемые компетенции и результаты обучения

| № | Индекс компетенции / Структурной составляющей компетенции | Формулировка компетенции /  Структурные составляющие компетенции (результаты обучения) |
| --- | --- | --- |
| **1** | **2** | **3** |
| **1** | **ОПК-3** | ***Способностью работать с компьютером как средством управления информацией, работать с информацией из различных источников, в том числе в глобальных компьютерных*** |
|  | С5-(ОПК-3) | знание информационных моделей |
|  | С6-(ОПК-3) | умение структурировать и обрабатывать информацию |
|  | С7-(ОПК-3) | владение основными методами, способами и средствами получения, хранения, переработки информации |
| **2** | **ПК-6** | ***Управление контентом предприятия и Интернет-ресурсов, процессами создания и использования информационных сервисов (контент-сервисов)*** |
|  | С1-(ПК-6) | знание инфологических моделей данных |
|  | С2-(ПК-6) | умение использовать датологические модели при создании информационных сервисов |
|  | С3-(ПК-6) | владение процедурой нормализации баз данных при управлении контентом предприятия |
|  | С7-(ПК-6) | знание современных средств разработки и реализации информационных сервисов |
|  | С8-(ПК-6) | умение обеспечивать целостность и безопасность данных контент-сервисов |
|  | С9-(ПК-6) | владение автоматизированными программными средствами разработки и реализации информационных сервисов |

**1 ЖИЗНЕННЫЙ ЦИКЛ БАЗЫ ДАННЫХ**

**Жизненный цикл базы данных** — это совокупность этапов, которые проходит база данных на своём пути от создания до окончания использования. Последовательность этапов приведена на рисунке и соответствует ГОСТ 34.

В учебном пособии будут рассмотрены некоторые этапы проектирования реляционных баз данных и приведен пример разработки.

**2 ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

Инфологические модели данных используются на начальных этапах проектирования. Основная задача инфологических моделей - описание структур данных и связей между ними.

Цель инфологического моделирования (проектирования) – обеспечение понятных человеку способов сбора информации и её формальное представление.

**Модель «Сущность-Связь»(ER)**

Важную роль в инфологическом проектировании играет наглядность представляемых моделей данных. В настоящее время большой популярностью разработчиков пользуются средства, основанные на графических нотациях. Самым распространенным средством данного типа являются *диаграммы "сущность-связь"* (entity-relationship, ER), которые соответствуют объектно-ориентированному подходу.

Модель "сущность-связь" была предложена в 1976 г. Питером Пин-Шэн Ченом. Типовая разработка БД начинается с ER-моделирования, с последующей трансляцией в реляционную модель, подлежащую физической реализации.

*Компоненты диаграмм "сущность-связь":*

**Сущности** – объекты реального мира.

**Атрибуты** – значения, описывающие свойства сущности.

**Связи** – соединения между двумя или более множествами сущностей.

Существует несколько нотаций, т.е. вариантов обозначения элементов диаграмм "сущность-связь". Рассмотрим основные нотации.

**Нотация Чена**

В нотации Чена присутствуют независимые и зависимые сущности, однозначные, многозначные, наследуемые и ключевые атрибуты, а так же связи. В таблице показаны графические обозначения элементов диаграмм и их смысловая интерпретация.

|  |  |
| --- | --- |
| **Обозначение** | **Пояснение** |
| Имя сущности | Независимая сущность |
| Имя сущности | Зависимая сущность |
|  | Атрибут |
|  | Многозначный атрибут |
|  | Получаемый (наследуемый) атрибут в иерархических связях |
|  | Ключевой атрибут (первичный ключ) |
| Имя связи | Связь |

Связь, включает линии и ромб и соединяет с ассоциируемые сущности. Над линией цифрами указывается класс принадлежности (степень связи).Внутри ромба записывается название связи, при этом учитывается её направленность. Например, для сущностей А и В, мы хотим показать связь в которой сущность А является определяющей, тогда название связи строится как АВ.

Бинарные связи:

- Степень один к одному (1:1). Читать следует так «в каждый момент времени каждому экземпляру сущности A соответствует 1 или 0 экземпляров сущности B».

На рисунке показана единичная степень связи для каждой сущности.

А

АВ

В

1

1

- Степень один ко многим (1:N). Читать следует так «в каждый момент времени каждому экземпляру сущности A соответствует 2 и более экземпляров сущности B». Графически степень связи N может интерпретироваться "древообразной" структурой. На рисунке показано графическое обозначение, соответствующее нотации.

А

АВ

В

1

N

Интерпретация выглядит следующим образом:

А

АВ

1 экземпляр В

1

1

АВ

2 экземпляр В

1

АВ

N экземпляр В

1

1

1

- Степень многие к одному (N:1). Читать следует так «в каждый момент времени каждому экземпляру сущности B соответствует 2 и более экземпляров сущности А». Графически степень связи N может интерпретироваться "древообразной" структурой. На рисунке показано графическое обозначение, соответствующее нотации.

А

АВ

В

N

1

Интерпретация выглядит следующим образом:

В

АВ

1

АВ

1

АВ

1

1 экземпляр А

2 экземпляр А

N экземпляр А

1

- Степень многие ко многим (М:N). Читать следует так «в каждый момент времени 2м и более экземплярам сущности A соответствует 2 и более экземпляров сущности B». Графически степень связи N может интерпретироваться "сетевой" структурой. На рисунке показано графическое обозначение, соответствующее нотации.

А

АВ

В

М

N

Интерпретация выглядит следующим образом:

АВ

АВ

АВ

1 экземпляр А

2 экземпляр А

M экземпляр А

1 экземпляр В

2 экземпляр В

N экземпляр В

Подобный вид связи трудно реализуем в реляционных базах данных и на практике стараются изменить структуру БД, так, чтобы степень связи была меньше, даже за счёт избыточности данных

### Нотация Мартина

### Список атрибутов, в нотации Мартина, записывается внутри прямоугольника, обозначающего сущность. Ключевые атрибуты подчеркиваются.

|  |  |
| --- | --- |
| **Обозначение** | **Пояснение** |
| имя | Независимая сущность |
| имя | Зависимая сущность |
| имя | Родительская сущность в иерархической связи |

Связи изображаются линиями без ромба. Со стороны сущности на линии связи графически (линиями) указывается кардинальность. Обязательный или необязательный класс принадлежности также указывается графически (кружочком или чертой). Название связи записывается в центре рядом с линией связи или над ней.

|  |  |
| --- | --- |
| **Обозначение**  **по Мартину** | **Кардинальность** |
|  | нет |
|  | 1,1 |
|  | 0,1 |
|  | N,M |
|  | 0,N |
|  | 1,N |

### Нотация IDEF1X

В нотации IDEF1X атрибуты также как и в нотации Мартина записываются внутри прямоугольника сущности. Атрибуты, составляющие ключ сущности, группируются в верхней части прямоугольника и отделяются горизонтальной чертой.

|  |  |
| --- | --- |
| **Обозначение** | **Интерпретация** |
| имя | Независимая сущность |
| имя | Зависимая сущность |

Обозначения связей:

|  |  |
| --- | --- |
| **Элемент диаграммы** | **Тип связи** |
|  | Идентифицирующая связь |
|  | Неидентифицирующая связь |

Обозначение кардинальности связей:

|  |  |
| --- | --- |
| **Элемент диаграммы** | **Кардинальность и принадлежность** |
|  | 1,1 |
|  | 0,M |
| Z | 0,1 |
| P | 1,M |
| N | точно *N* (*N* - произвольное число) |

Пример ER-диаграммыв нотации Чена для студента исследователя выполняющего научную работу в рамках научного исследования приведён ниже.

Сущности «Магистрант», «Грант», «Научная работа». Ведущая (главная) сущность «Грант» связана с подчинёнными сущностями «Научная работа» и «Магистрант». Грант состоит из нескольких научных работ в соответствии с разделами общей темы исследований. Поэтому установлена связь «Содержит». Читаем как «Грант» содержит несколько «Научных работ». Каждая научная работа может выполняться группой студентов – исследователей, в нашем случае, магистрантами. Следовательно, устанавливается связь «Выполняет». Читаем как «Научная работа» выполняется «Магистрантамии».

Сущность «Грант» характеризуется темой исследований и областью применения. Сущность «Научная работа» содержит название, автора исследований и результаты экспериментов. Сущность «Магистрант» характеризуется фамилией, именем и отчеством студента и его телефоном.

**Грант**

**Содержит**

**Научная работа**

связь

сущность

атрибут

**Выпол-няет**

**Магистрант**

1

N

N

1

**3 ДАТАЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ**

В даталогической модели изображаются логические взаимосвязи между элементами данных безотносительно их физической организации. При этом даталогическая модель разрабатывается с учетом специфики конкретной СУБД, также с учетом специфики конкретной предметной области на основе ее инфологической модели.

**СИНТЕЗ И ДЕКОМПОЗИЦИЯ ОТНОШЕНИЙ**

В настоящее время применяются два направления проектирования БД:

- Синтез. Компоновка заданных элементарных зависимостей.

- Декомпозиция (разбиение). Исходное множество отношений заменяется другим множеством отношений. Их число возрастает.

Декомпозиция основана на понятии нормальной формы отношений, основной метод – нормализация.

Нормализация – метод создания набора отношений с заданными свойствами на основе некоторых требований к данным. Процесс нормализации предложен Коддом (1972 г.) – формальный метод для оптимизации столбцов отношений и устранения аномалий.

## Синтез реляционных баз данных

Пусть дан документ в виде приходного ордера:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Приходный ордер №123 | | | | |
| Дата | Поставщик | Адрес | | |
| 14.10.2017 | НПП «Миландр» | г. Москва, ул. Автозаводская, д.20 | | |
|  | | | | |
| Получены детали | Количество | Единицы | Цена за единицу | Стоимиость |
| Д1 | 100 | шт. | 150 | 15000 |
| Д2 | 10 | упак. | 1000 | 10000 |
| Д3 | 50 | шт | 70 | 3500 |

Воспользуемся методом синтеза БД, применив определения нормальных форм и алгоритм, описанный выше.

|  |  |
| --- | --- |
| **Имя атрибута** | **Семантика** |
| A | № накладной |
| B | дата |
| C | поставщик |
| D | дом |
| E | улица |
| F | город |
| G | имя детали |
| K | количество |
| L | единица измерения |
| M | цена за единицу |
| N | стоимость |

Отсюда получаем следующие функциональные зависимости:

A→B - дата зависит от № накладной;

C→DEF - дом, улица и город зависят от поставщика;

G→ML - единица измерения и цена за единицу зависят от имени детали;

ABCG→K - количество может зависеть от № накладной, даты, поставщика и имени детали;

GKM→N - общая стоимость зависит от имени детали, количества и цены за единицу.

Исключим последнюю ФЗ для получения неизбыточного покрытия, т.к. стоимость может быть посчитана как произведение *Количество\* Цена\_за\_ед.*

Для построения редуцированного покрытия необходимо из левых и правых частей ФЗ удалить посторонние атрибуты.

Редуцирование левых частей:

A→B

C→DEF

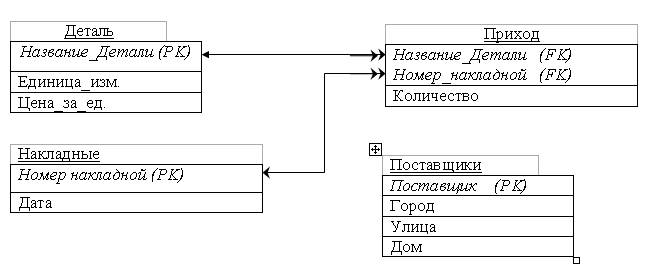
G→ML

AG→K

Редуцирование правых частей:

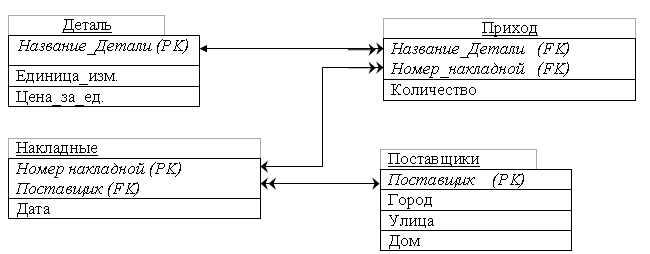
Данное множество ФЗ является редуцированным справа.

В итоге получили следующую совокупность таблиц:



Данную БД необходимо привести к 3НФ и установить связи между таблицами.

Полученное множество отношений уже находится в 1НФ, т.к. дальнейшее деление атрибутов не возможно. Продолжим нормализацию. Отношение находится во 2НФ, тогда и только тогда, когда оно находится в 1НФ. Все поля зависят от первичного ключа. Первичный ключ неизбыточен. Отношение находится в 3НФ, тогда и только тогда, когда оно находится во 2НФ. В таблице не должно быть транзитивных зависимостей между неключевыми полями. Или другими словами значение неключевого поля не зависит от любого неключевого поля. Установим связь: «Поставщики» - «Накладные» (1:М) и получим следующее:



Данная база данных находится в 3НФ, что нам и требовалось.

**Пример декомпозиции**

Пусть дан документ в виде приходного ордера:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Приходный ордер №123 | | | | |
| Дата | Поставщик | Адрес | | |
| 14.10.2017 | НПП «Миландр» | г. Москва, ул. Автозаводская, д.20 | | |
|  | | | | |
| Получены детали | Количество | Единицы | Цена за единицу | Стоимиость |
| Д1 | 100 | шт. | 150 | 15000 |
| Д2 | 10 | упак. | 1000 | 10000 |
| Д3 | 50 | шт | 70 | 3500 |

Выполним последовательную декомпозицию пошагово, применив определения нормальных форм, которые даны выше.

***1НФ.*** Каждое поле таблицы БД должно быть неделимым и не содержать повторяющихся (одинаковых по смыслу) групп. Неделимость значит, что значения поля не делятся на более мелкие. Повторяющиеся - это поля, содержащие одинаковые по смыслу значения.

Применив определение, выберем из документа названия полей. Получим отношение «Приход»:

|  |
| --- |
| Приход |
| Номер\_накладной |
| Дата |
| Поставщик |
| Город |
| Улица |
| Дом |
| Название\_Детали |
| Количество |
| Единица\_изм. |
| Цена\_за\_ед. |
| Стоимость |

Поле «Дата» не делится на день, месяц и год, т.к. существует такой тип переменной. При необходимости значения можно получить любую часть даты из общего значения. Напротив «Адрес» является составным однотипным элементом и требует разделения на «Город», «Улица» и «Дом».

Считаем, что таблица «Приход» находится в 1НФ и дальнейшее деление атрибутов невозможно.

***2НФ.*** Отношение находится во 2НФ, тогда и только тогда, когда оно находится в 1НФ. Все поля зависят от первичного ключа. Первичный ключ неизбыточен.

а) Определим атрибуты первичного ключа. Выберем поле «Деталь» как первичный ключ. Это поле определяет характеристики детали. Но другие поля не определяет. Значит ключ составной. Добавим к нему поле «Поставщик», характеризующее поставщика и поля «Номер накладной» и «Дата», характеризующие документ.

Получим отношение (первый шаг в выборе ключа):

б) Если ключ избыточен – уточнить состав полей в первичном ключе. Проведем дополнительный анализ зависимостей внутри составного ключа. Увидим, что «Номер накладной» однозначно определяет поля «Дата» и «Поставщик». Следовательно, эти поля вернем обратно в разряд неключевых атрибутов.

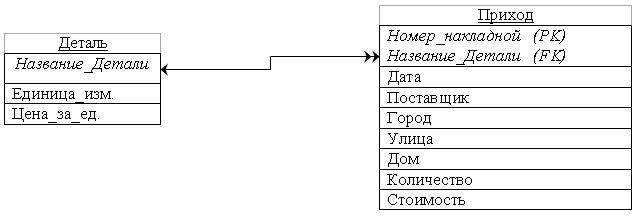
|  |
| --- |
| Приход |
| *Название\_Детали*  *Поставщик*  *Номер\_накладной*  *Дата* |
| Город |
| Улица |
| Дом |
| Количество |
| Единица\_изм. |
| Цена\_за\_ед. |
| Стоимость |

Получим новое отношение (второй шаг в выборе ключа):

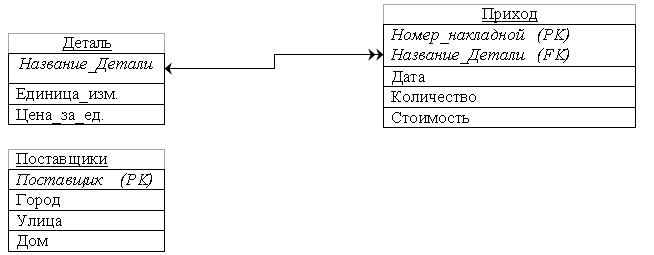
|  |
| --- |
| Приход |
| *Название\_Детали*  *Номер\_накладной* |
| Дата |
| Поставщик |
| Город |
| Улица |
| Дом |
| Количество |
| Единица\_изм. |
| Цена\_за\_ед. |
| Стоимость |

в) Если ключ неизбыточен, проверить, что значения всех полей однозначно зависят от значения первичного ключа.

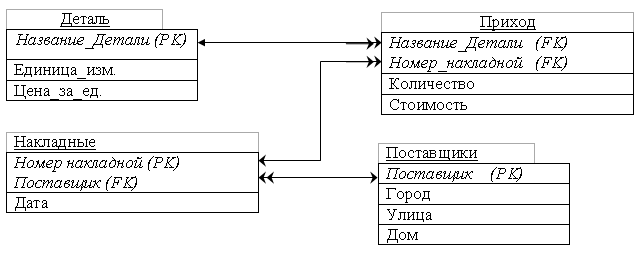
Видим, что «Цена\_за\_ед.» и «Единица\_изм.» функционально полно зависят от поля «*Название\_Детали*». Выделим эти поля в отдельную таблицу «Деталь». Так как одно наименование детали может присутствовать в различных накладных, установим связь один ко многим:



Заметим, что поле «Поставщик» не зависит от «Номер\_накладной», от поле «Поставщик» однозначно определяет «Город», «Улица» и «Дом». Выделим эти поля в отдельную таблицу «Поставщики» (декомпозиция на три отношения):



Однако, связь не устанавливается. Далее определим, что «Дата» зависит от номера накладной. Выделим это поле в отдельную таблицу «Накладные» и установим связи: «Поставщики» - «Накладные» (1:М) и «Накладные» - «Приход» (1:М) (декомпозиция на четыре отношения):

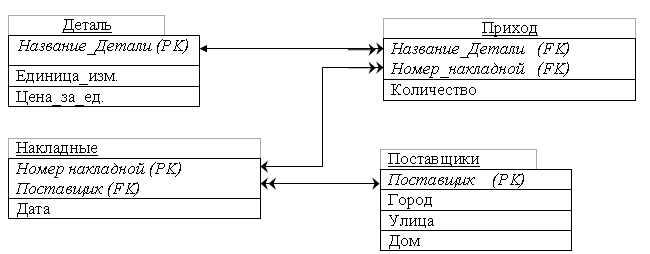


Полученное множество отношений находится в 2НФ.

***3НФ.*** Отношение находится в 3НФ, тогда и только тогда, когда оно находится во 2НФ. В таблице не должно быть транзитивных зависимостей между неключевыми полями. Или другими словами значение неключевого поля не зависит от любого неключевого поля.

В таблице «Приход» поле «Стоимость» зависит от поля «Количество» и существует формула вычислений «Стоимость» = «Количество» \* «Цена\_за\_ед.». То, что поле «Цена\_за\_ед.» находится в другой таблице не имеет значения.

Исключим поле «Стоимость». Получим множество отношений (декомпозиция на четыре отношения в 3НФ):



Вывод - база данных нормализована. Здесь возможна декомпозиция только до 3НФ.

**РЕЛЯЦИОННАЯ АЛГЕБРА**

Реляционная алгебра представляет собой набор операторов, использующих отношения в качестве аргументов, и возвращающие отношения в качестве результата.

Традиционно, вслед за Коддом, определяют восемь реляционных операторов, объединенных в две группы.

**Теоретико-множественные операторы:**

-объединение;

-пересечение;

-вычитание;

-декартово произведение.

**Специальные реляционные операторы:**

-выборка;

-проекция;

-соединение;

-деление.

**Стандартные реляционные операции**

*Объединением* двух совместимых по типу отношений A и B называется отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих или A, или B, или обоим отношениям.

Синтаксис операции объединения

**A union B**

С = (A union B) | ti Є С tj Є А & ti Є С tj Є В

*Пример.* Пусть даны два отношения А (детали изготовлены из стали) и В (детали весом больше 0,5):

Отношение A

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 3 | Сталь | 0,5 |

Отношение B

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Медь | 1,0 |
| 4 | Алюминий | 0,7 |

Тогда результирующее отношение будет состоять из деталей, которые – или изготовлены из стали, или имеют вес больше 0,5:

С = A UNION B

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 3 | Сталь | 0,5 |
| 2 | Медь | 1,0 |
| 4 | Алюминий | 0,7 |

*Пересечением* двух совместимых по типу отношений A и B называется отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям A и B.

Синтаксис операции пересечения

**A intersect B**

C = (A intersect B) | ti Є C | ti Є A & ti Є B

*Пример.* Пусть даны два отношения А (детали изготовлены из стали) и В (детали весом больше 0,5):

Отношение А

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 3 | Сталь | 0,5 |

Отношение В

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 4 | Алюминий | 0,7 |

Тогда результирующее отношение будет состоять из деталей, изготовленных из стали и весом больше 0,5:

С = A INTERSECT B

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |

*Вычитанием*двух совместимых по типу отношений A и B называется отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B.

Синтаксис операции вычитания

**A** **minus** **B**

C = (A minus B) | ti Є C | ti Є A & ti Є B

*Пример.* Для тех же отношений A и B, что и в предыдущем примере вычитание имеет вид:

Отношение А

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 3 | Сталь | 0,5 |

Отношение В

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 4 | Алюминий | 0,7 |

Результат: детали изготовленные из стали и не весят больше 0,5:

С = A MINUS B

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 3 | Сталь | 0,5 |

Следует заметить, что операция вычитания учитывает порядок следования операндов С = A MINUS B не одно и тоже, что С = В MINUS А

*Декартовым произведением* двух отношений A(A1, A2, …, An) и B(B1, B2, …, Bn) называется отношение, заголовок которого является сцеплением заголовков отношений A и B, а тело которого состоит из кортежей, являющихся сцеплением кортежей отношений A и B таких, что .

Синтаксис операции произведения

**A times B**

C = (A times B) | ai Є A & bi Є B & ai ≠ bi ∃ t | t = a∪b

*Пример.* Пусть даны два отношения А и В с информацией о поставщиках и деталях:

Отношение В

|  |  |
| --- | --- |
| **Номер**  **детали** | **Наименование**  **детали** |
| 1 | Болт |
| 2 | Гайка |
| 4 | Винт |

Отношение А

|  |  |
| --- | --- |
| **Номер**  **поставщика** | **Имя**  **поставщика** |
| 1 | Иванов |
| 2 | Петров |
| 3 | Сидоров |

С = A TIMES B

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер поставщика** | **Имя поставщика** | **Номер**  **детали** | **Наименование**  **детали** |
| 1 | Иванов | 1 | Болт |
| 1 | Иванов | 2 | Гайка |
| 1 | Иванов | 3 | Винт |
| 2 | Петров | 1 | Болт |
| 2 | Петров | 2 | Гайка |
| 2 | Петров | 3 | Винт |
| 3 | Сидоров | 1 | Болт |
| 3 | Сидоров | 2 | Гайка |
| 3 | Сидоров | 3 | Винт |

**Специальные операции**

*Делением* отношений А на В называется отношение с заголовком (Х1, Х2, …, Хn) и телом, содержащим множество кортежей (x1, x2, …, xn), таких, что для всех кортежей (y1, y2, …, yn) Є B в отношении A найдется кортеж (x1, x2, …, xn, y1, y2, …, yn).

Отношение A выступает в роли делимого, отношение B выступает в роли делителя. Деление отношений аналогично делению чисел с остатком. Например, деление бинарного отношения на унарное возвращает отношение, содержащее все значения одного атрибута бинарного отношения, которые соответствуют всем значениям в унарном отношении.

Синтаксис операции деления:

**A divide by B**

**Замечание:** Типичные запросы, реализуемые с помощью операции деления, обычно в своей формулировке имеют слово "все" - "какие поставщики поставляют все детали?".

*Пример.* Пусть есть отношение исходное AP и делители Bi для i=1, 2, 3. Выполним деление отношений AP divide by B1 и AP divide by B2.

АР

|  |  |  |  |
| --- | --- | --- | --- |
| **К** | **Р** | **К** | **Р** |
| К1 | Р1 | К2 | Р1 |
| К1 | Р2 | К2 | Р2 |
| К1 | Р3 | К3 | Р2 |
| К1 | Р4 | К4 | Р3 |
| К1 | Р5 | К4 | Р4 |
| К1 | Р6 | К4 | Р5 |

AP divide by B1

|  |
| --- |
| **Р** |
| Р3 |
| Р4 |

AP divide by B2

|  |
| --- |
| **К** |
| К1 |
| К4 |

В1

|  |
| --- |
| **Р** |
| Р1 |

В2

|  |
| --- |
| **К** |
| К1 |
| К2 |

*Выборкой (ограничением, селекцией****)*** на отношении A с условием C называется отношение с тем же заголовком, что и у отношения A, и телом, состоящем из кортежей, значения атрибутов которых при подстановке в условие C дают значение ИСТИНА. C представляет собой логическое выражение, в которое могут входить атрибуты отношения A и (или) скалярные выражения.

Синтаксис операции выборки:

**A where C** или **A where X Θ Y**

В простейшем случае условие C имеет вид X Θ Y, где Θ - один из операторов сравнения, а X и Y - атрибуты отношения A или скалярные значения. Такие выборки называются Θ-выборки (тэта-выборки) или Θ-ограничения, Θ-селекции.

*Пример.* Для отношения А сформируем выборку.

Отношение А

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 1 | Сталь | 0,8 |
| 2 | Сталь | 1,0 |
| 3 | Сталь | 0,5 |

A where Вес >= 1,0

|  |  |  |
| --- | --- | --- |
| **Номер детали** | **Материал** | **Вес** |
| 2 | Сталь | 1,0 |

*Проекцией* отношения A по атрибутам X, Y, …, Z, где каждый из атрибутов принадлежит отношению A, называется отношение с заголовком (X, Y, …, Z) и телом, содержащим множество кортежей вида (x, y, …,z), таких, для которых в отношении A найдутся кортежи со значением атрибута X равным x, значением атрибута Y равным y, …, значением атрибута Z равным z.

Синтаксис операции проекции:

**A [X,Y,…,Z]**

**Замечание.** Операция проекции дает "вертикальный срез" отношения, в котором удалены все возникшие при таком срезе дубликаты кортежей.

*Пример*. Пусть дано отношение с информацией о поставщиках, включающих наименование и месторасположение (отношение А), выполнить проекцию «Город поставщика»:

Отношение А (Поставщики)

|  |  |  |
| --- | --- | --- |
| **Номер**  **поставщика** | **Имя**  **поставщика** | **Город**  **поставщика** |
| 1 | Иванов | Уфа |
| 2 | Петров | Москва |
| 3 | Сидоров | Москва |
| 4 | Сидоров | Челябинск |

Проекция А (Город поставщика)

|  |
| --- |
| **Город поставщика** |
| Уфа |
| Москва |
| Челябинск |

*Соединение.*Операция соединения имеет несколько вариантов: это наиболее важное *естественное* соединение и *Θ – соединение.*

*Естественное соединение* . Пусть даны отношения A(A1, A2,…,An, X1, X2,…, Xp) и B(B1, B2,…,Bn, X1, X2,…,Xm), имеющие одинаковые атрибуты X1, X2,…, Xp (т.е. атрибуты с одинаковыми именами и определенные на одинаковых доменах).

Тогда *естественным соединением* отношений A и B называется отношение с заголовком A(A1, A2,…,An, X1, X2,…, Xp, B1, B2,…, Bm) и телом, содержащим множество кортежей (a1, a2,…, an, x1, x2,…, xp, b1, b2,…, bm), таких, что (a1, a2,…, an, x1, x2,…, xp) Є A и (x1, x2,…, xp, b1, b2,…, bm) Є B.

Синтаксис операции естественного соединения:

**A JOIN B**

В синтаксисе естественного соединения не указываются, по каким атрибутам производится соединение. Естественное соединение производится по **всем** одинаковым атрибутам.

Можно выполнять последовательное естественное соединение нескольких отношений.

*Пример.* Пусть имеем таблицу деталей С и таблицу поставщиков Р:

Отношение С

|  |  |  |  |
| --- | --- | --- | --- |
| **Название детали** | **Материал** | **Вес** | **Город** |
| D1 | Сталь | 0,8 | Москва |
| D2 | Сталь | 1,0 | Москва |
| D3 | Сталь | 0,5 | Пенза |
| D3 | Алюминий | 0,7 | Липецк |

Отношение Р

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N\_пост** | **Название**  **завода** | **Город** | **Улица** | **Дом** |
| Р1 | … | Москва | … | … |
| Р2 | … | Омск | … | … |
| Р3 | … | Саратов | … | … |
| Р4 | … | Москва | … | … |
| Р5 | … | Тамбов | … | … |

С JOIN Р

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Название**  **детали** | **Материал** | **Вес** | **Город** | **N\_пост** | **Название**  **завода** | **Улица** | **дом** |
| D1 | Сталь | 0,8 | Москва | Р1 | … | … | … |
| D1 | Сталь | 0,8 | Москва | Р4 | … | … | … |
| D2 | Сталь | 1,0 | Москва | Р1 | … | … | … |
| D2 | Сталь | 1,0 | Москва | Р4 | … | … | … |

*Θ – соединение****.*** Пусть отношение А содержит атрибут Х, отношение В содержит атрибут Y, а Θ - один из операторов сравнения (=, <, > и т.д.). Тогда *Θ - соединением* отношения A по атрибуту X с отношением B по атрибуту Y называют отношение:

**(A TIMES B) WHERE X Θ Y**

Эта операция используется в редких случаях, когда надо соединить два отношения на основе некоторых условий, отличных от эквивалентности. Θ – соединение эквивалентно двум операциям: нахождению декартова произведения от двух отношений и последующему выполнению указанной выборки из полученного результата.

**Основные свойства операций**

Операции объединения, пересечения, декартова произведения, естественное соединение обладают свойствами:

- ассоциативность:

(A union B) union C ~ A union (B union C) => A union B union C;

- коммутативность:

A union B ~ B union A;

A intersect B ~ B intersect A;

A times B ~ B times A.

Указанные свойства не выполняются для операции вычитания.

Все операции обладают свойством замкнутости – результат каждой операции над отношением есть отношение.

**ПРИМЕР ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ**

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Описание предметной области

Объектом автоматизации данной курсовой работы является фермерское хозяйство ООО «Зеленый луч», которое занимается растениеводством и животноводством. Организационная структура организации состоит из ряда взаимосвязанных и подчиненных общим целям отделам. К таким отделам относится отдел кадров.

Предметной областью автоматизации является учет кадров предприятия.

Функции отдела кадров ограничены следующими локальными задачами:

- оперативный контроль и учёт кадров на предприятии;

- выдача справок и выходных документов о сотрудниках, налоговых льготах и других сведений;

- анализ и обработка входной информации о персонале компании. Организационная структура отдела кадров ООО «Зеленый луч» представлена на рисунке 1.

Отдел кадров

Сектор приема

Сектор увольнения

Сектор по работе с сотрудниками

Сектор учета

Рисунок 1 – Организационная структура отдела кадров

Отдел управления персоналом занимается персональным учетом работников, который включает прием и выбытие. Основные задачи кадрового учета следующие:

- планирование профессиональной подготовки рабочих и специалистов (обучение, переподготовка);

- планирование повышения квалификации (обучение и переобучение) руководящих кадров;

- рациональная расстановка кадров в соответствии с уровнем их квалификации;

- аттестация и рационализация рабочих мест;

- статистический и оперативный учет и отчетность;

- анализ движения кадров;

- анализ качественного состава кадров;

- ведение архива.

Отдел управления персоналом ведёт учёт персонального состава работников, движение кадров, обучение и переобучение кадров. Также функцией отдела является составление должностных инструкций, формирование отчетности и обработка анкет. Отдел управления персоналом также обязан следить за трудовой дисциплиной, а также за продвижением по служебной лестнице сотрудников.

Отдел кадров наряду с расчетом численности активной части трудовых ресурсов фирмы, ведет расчет численности управленческого персонала. Базируясь на представленных показателях плановой численности, кадровая служба фирмы планирует прием и увольнение сотрудников, а также распределение и регулирование принятых рабочих и служащих по рабочим местам. В процессе изготовления продукции рабочие получают плановые задания, фактические результаты выполнения которых фиксируются либо в бумажных документах, либо на машинных носителях и передаются в бухгалтерию. Бухгалтерия в свою очередь, обрабатывает полученную информацию и вычисляет показатели оценки трудовой деятельности в денежном эквиваленте. До начала расчетов отдел кадров информирует бухгалтерию о текущих изменениях относительно отдельных работников (данные о пропущенном рабочем времени по причине временной нетрудоспособности о внутреннем и внешнем движении кадров, о нарушениях трудовой дисциплины и т.д.). Отдельные показатели такого рода передаются в отдел кадров, который использует их впоследствии при решении своих задач.

Специалист по кадрам обеспечивает соответствие осуществляемых операций с персоналом законодательству РК, контроль за  
движением персонала и выполнением обязательств перед ним. Кадровик назначается на должность и освобождается от должности генеральным директором. Прием и сдача дел при назначении и освобождении специалиста оформляются актом после проверки состояния кадрового учета и отчетности. Кадровик подчиняется генеральному директору фирмы.

В результате проектного исследования установлено, что отдел кадров нуждается в автоматизации своей деятельности с целью оперативной обработки данных о сотрудниках, что упрощает доступ к информации по иждивенцам, налоговым льготам, а также других данных о личном составе по бухгалтерии, на основе которых составляются таблицы с соответствующей информацией.

1.2 Документационное обеспечение процессов управления

Среди задач, решаемых кадровой службой, наиболее традиционной представляется документирование трудовых правоотношений. В процессе их возникновения и установления возникает значительный объем разнообразных документов. Наибольшая часть документов кадровой службы связана с движением кадров. Документирование движения кадров – неотъемлемая обязанность любой организации (предприятия, учреждения) независимо от масштабов его деятельности и организационно-правовой формы. Под движением кадров специалисты понимают:

* прием на работу;
* перевод на другую работу;
* увольнение;
* предоставление отпусков;
* командирование.

Документооборот отдела кадров в организации представлен таблицей 1.

Таблица 1 – Документооборот отдела кадров

|  |  |  |
| --- | --- | --- |
| Структурное подразделение | Полученные документы | Предоставленные документы |
| Бухгалтерия | - справки о заработной плате для оформления пенсии;  - материалы для выдачи справок работникам о работе на предприятии, занимаемой должности и размере заработной платы | - сведения о приеме, перемещении и увольнении работников;  - проекты приказов о приеме, увольнении и перемещении материально ответственных лиц;  - табель учета рабочего времени;  - график отпусков;  - листки временной нетрудоспособности к оплате |
| Отдел организации и оплаты труда | - штатное расписание;  - схемы должностных окладов, доплат, надбавок к заработной плате;  - показатели по труду и заработной плате;  - положения о премировании работников;  - расчеты фондов заработной платы и численности;  - расчеты потребности в рабочих и служащих | - сведения о приеме, перемещении и увольнении работников;  - сведения о списочной численности работников;  - данные о текучести кадров |
| Юридический отдел | - сведения об изменениях трудового законодательства, законодательства о социальном обеспечении;  - разъяснения действующего законодательства и порядка его применения | - проекты трудовых договоров с руководящими работниками предприятия;  - заявки на поиск необходимых нормативно-правовых документов, и на разъяснение действующего законодательства;  - приказы для визирования |

## АНАЛИЗ СУЩЕСТВУЮЩИХ ПОДСИСТЕМ

Возможным продуктом по кадровому учету, который мог бы стать в ООО «Зеленый луч» основным, является «1С:Управление персоналом 8.0.». Типовая конфигурация «Зарплата и Управление персоналом» предполагается в качестве инструмента для реализации кадровой политики, а также для автоматизации различных служб предприятия, начиная от службы управления персоналом и линейных руководителей до работников бухгалтерии.

Но программа еще сложна для использования в данной организации, и требует больших затрат для внедрения и реализации. Для 1С:8.0 требуются хорошие компьютеры, с мощным техническим обеспечением, что и является основными затратами на внедрение программы. Для организации, где еще нет автоматизированной подсистемы управления кадров, описываемая выше программа является достаточно сложной и не выгодной. Поэтому в дальнейшем разрабатываемая подсистема отдел кадров ООО «Зеленый луч» в MS Access представляет простоту и удобство использования, содержит всю необходимую информацию по кадровому учету.

- получение информации о сотрудниках;

- информация о дате рождения;

- информация о семейном положении и детях сотрудника;

- ведомость по заработным платам сотрудников;

- информация о членах семьи сотрудника;

- сведения о сотрудниках, оклад которых ниже заданного.

Для удобства использования базы данных «Отдел кадров предприятия » будет предусмотрен поиск по следующим критериям:

- поиск сотрудника по фамилии;

- поиск сотрудника по профессии.

На базе любого поиска необходимо предусмотреть возможность печати результатов, создание отчетов по каждому запросу. Также в приложении необходимо создать возможность редактировании (коррекции) данных, обновления базы данных, получение сведений о сотруднике или об определенном отделе.

Кроме того, в результате решения задачи необходимо выполнить разработку подсистемы ведения данных: ввод новых данных, удаление из БД данных об уволившихся по различным причинам сотрудниках. Кроме того, для удобства работы будет создан удобный и понятный для пользователя интерфейс, с целью предоставить возможность работы с проектом даже неопытному пользователю.

4ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ

## 4.1 Информационное обеспечение

Декомпозиция основана на понятии нормальных форм и переходе от одной нормальной формы (НФ) к другой последовательно. Декомпозиция должна сохранять эквивалентность схем баз данных (БД) при замене одной на другую, т.е. не нарушать обратимость исходной схемы.

Рассмотрим основные понятия, для того, чтобы определения нормальных форм не вызывали вопросов.

Функциональная зависимость (ФЗ) – это соотношение проекций R[A] и R[B] при котором в каждый момент времени любому элементу R[A] соответствует только один элемент проекции R[B], входящий в вместе в какой-либо кортеж отношения R.

Полная функциональная зависимость (ПФЗ). Набор атрибутов В функционально зависит от А, и не зависит функционально от любого подмножества А.

Транзитивная функциональная зависимость (ТФЗ). Пусть существует набор атрибутов С, таких что:

1. С не является подмножеством А;
2. С не включает в себя В;
3. Существует функциональная зависимость С от А ();
4. Не существует функциональная зависимость А от С ();
5. Существует функциональная зависимость B от C ().

Детерминант отношения. Если в отношении существуют несколько ФЗ, то каждый атрибут или набор атрибутов, от которого зависит другой атрибут, называется детерминантом отношения.

Аксиомы Армстронга.

1. Рефлексивность: если В является подмножеством А, то .
2. Дополнение: если , то .
3. Транзитивность: если и , то .

Первая нормальная форма. Обозначение 1НФ или 1NF. варианты определений первой нормальной формы:

1. Отношение находится в 1НФ, тогда и только тогда, когда на пресечении каждого столбца и каждой строки находятся только элементарные значения атрибутов.
2. Отношение находится в 1НФ, тогда и только тогда, когда все используемые домены содержат только скалярные значения.
3. Каждое поле таблицы БД должно быть неделимым и не содержать повторяющихся (одинаковых по смыслу) групп.

Вторая нормальная форма. Обозначение 2НФ или 2NF. варианты определений второй нормальной формы:

1. Отношение находится во 2НФ, тогда и только тогда, когда оно находится в 1НФ и не содержит неполных ФЗ не первичных атрибутов от атрибутов первичного ключа.
2. Отношение находится во 2НФ, тогда и только тогда, когда оно находится в 1НФ и каждый неключевой атрибут неприводимо зависит от первичного ключа.
3. Отношение находится во 2НФ, тогда и только тогда, когда оно находится в 1НФ. Все поля зависят от первичного ключа. Первичный ключ неизбыточен.

Третья нормальная форма. Обозначение 3НФ или 3NF. варианты определений второй нормальной формы:

1. Отношение находится в 3НФ, тогда и только тогда, когда оно находится во 2НФ и не содержит транзитивных зависимостей.
2. Отношение находится в 3НФ, тогда и только тогда, когда оно находится во 2НФ и каждый неключевой атрибут не транзитивно зависит от первичного ключа.
3. Отношение находится в 3НФ, тогда и только тогда, когда оно находится во 2НФ. В таблице не должно быть транзитивных зависимостей между неключевыми полями.

1НФ. Каждое поле таблицы БД должно быть неделимым и не содержать повторяющихся (одинаковых по смыслу) групп. Неделимость значит, что значения поля не делятся на более мелкие. Повторяющиеся - это поля содержащие одинаковые по смыслу значения.

Применив определение, выберем из документа названия полей. Получим отношение «Сотрудник» (таблица 2).

Таблица 2 – «Сотрудник» в 1НФ

|  |
| --- |
| ИД сотрудник  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |
| Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения  Серия  Номер  Кем выдан  Дата выдачи  Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

Считаем, что таблица «Сотрудник» находится в 1НФ и дальнейшее деление атрибутов невозможно.

2НФ. Отношение находится во 2НФ, тогда и только тогда, когда оно находится в 1НФ. Все поля зависят от первичного ключа. Первичный ключ неизбыточен.

Выбрала поле «ИД сотрудник» как первичный ключ. Это поле определяет характеристики сотрудника.

Видно, что «Название учебного заведения», «Тип документа», «Начало обучения», «Конец обучения» функционально полно зависят от поля «ИД сотрудник». Выделим эти поля в отдельную таблицу «Обучение». Так как один сотрудник может присутствовать в различных документах, установим связь один ко многим (рисунок 2).

|  |  |  |
| --- | --- | --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона | |  | | --- | | ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения | |
| Серия  Номер  Кем выдан  Дата выдачи  Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

Рисунок 2 - Декомпозиция на два отношения

Затем анализируя таблицу «Сотрудник» можем выделить в отдельную таблицу «Паспорт» поля «Серия», «Номер», «Кем выдан» и «Дата выдачи».

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |
| Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

Рисунок 3 - Декомпозиция на три отношения

Далее устанавливается связь таблицы «Сотрудник» и новой образованной таблицы «Состав семьи».

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |
| Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

Рисунок 4 - Декомпозиция на четыре отношения

В последующих таблицах поступаем аналогичным образом, продолжая разбивать таблицы на части, и устанавливая связи между ними.

Все связи между таблицей «Сотрудник» и таблицами «Паспорт», «Обучение», «Состав семьи» будут 1:М.

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |
| Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

|  |
| --- |
| ИД сотрудник (FK)  Номер пп (PK)  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска |

Рисунок 5 - Декомпозиция на пять отношений

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |
| Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

|  |
| --- |
| ИД сотрудник (FK)  Номер пп (PK)  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска |

|  |
| --- |
| ИД сотрудник (FK)  ИД предприятия (PK)  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы |

Рисунок 6 - Декомпозиция на шесть отношений

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

|  |
| --- |
| ИД сотрудник (FK)  ИД зарплата (PK)  Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п |

|  |
| --- |
| ИД сотрудник (FK)  Номер пп (PK)  Вид отпуска  Период с  Период до  Начало отпуска  Конец отпуска |

|  |
| --- |
| ИД сотрудник (FK)  ИД предприятия (PK)  Подразделение  Номер приказа тд  Дата приказа тд  Начало работы |

Рисунок 7 - Декомпозиция на семь отношений

Далее добавляются таблицы «Учебные заведения», «Вид отпуска» и «Должность». «Учебные заведения» связана с таблицей «Обучение» связью один-ко-многим.

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

Учебные заведения(PK)

Вид отпуска(PK)

|  |
| --- |
| ИД сотрудник (FK)  Номер пп (PK)  Вид отпуска(FK)  Период с  Период до  Начало отпуска  Конец отпуска |

Должность(PK)

|  |
| --- |
| ИД сотрудник (FK)  ИД зарплата (PK)  Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п |

|  |
| --- |
| ИД сотрудник (FK)  ИД предприятия (PK)  Подразделение  Должность (FK)  Номер приказа тд  Дата приказа тд  Начало работы |

Рисунок 8 - Декомпозиция на десять отношений

Полученное множество отношений находится в 2НФ.

3НФ. Отношение находится в 3НФ, тогда и только тогда, когда оно находится во 2НФ. В таблице не должно быть транзитивных зависимостей между неключевыми полями. Или другими словами значение неключевого поля не зависит от любого неключевого поля.

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

Учебные заведения(PK)

|  |
| --- |
| ИД сотрудник (FK)  ИД зарплата (PK)  Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п |

Вид отпуска(PK)

Должность(PK)

|  |
| --- |
| ИД сотрудник (FK)  Номер пп (PK)  Вид отпуска(FK)  Период с  Период до  Начало отпуска  Конец отпуска |

|  |
| --- |
| ИД сотрудник (FK)  ИД предприятия (PK)  Подразделение  Должность (FK)  Номер приказа тд  Дата приказа тд  Начало работы |

Рисунок 9 – Декомпозиция на десять отношений в 3НФ

База данных нормализована. Здесь возможна декомпозиция только до 3НФ.

ER-модель удобна при проектировании [баз данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). С её помощью можно выделить ключевые сущности, присутствующие в модели, и обозначить отношения, которые могут устанавливаться между этими сущностями. На рисунке 10 изображена схема базы данных подсистемы отдел кадров ООО «Зеленый луч».

|  |
| --- |
| ИД сотрудник (FK)  ИД паспорт(PK)  Серия  Номер  Кем выдан  Дата выдачи |

|  |
| --- |
| ИД сотрудник (PK)  Фамилия  Имя  Отчество  Табельный номер  Пенсионный код  Пол  Дата рождения  Образование  Должность  Квалификация  Семейное положение  Адрес  Номер телефона |

|  |
| --- |
| ИД сотрудник (FK)  ИД обучение (PK)  Название уч. заведения  Специальность  Тип документа  Начало обучения  Конец обучения |

|  |
| --- |
| ИД сотрудник (FK)  ИД семьи(PK)  Степень родства  Адрес род-ка  Фамилия род-ка  Имя род-ка  Отчество род-ка  Дата рождения род-ка |

Учебные заведения(PK)

Вид отпуска(PK)

|  |
| --- |
| ИД сотрудник (FK)  Номер пп (PK)  Вид отпуска(FK)  Период с  Период до  Начало отпуска  Конец отпуска |

Должность(PK)

|  |
| --- |
| ИД сотрудник (FK)  ИД зарплата (PK)  Вид зарплата  Величина зарплаты  Начало действия  Номер приказа о з/п  Дата приказа з/п |

|  |
| --- |
| ИД сотрудник (FK)  ИД предприятия (PK)  Подразделение  Должность (FK)  Номер приказа тд  Дата приказа тд  Начало работы |

Рисунок 10 - схема базы данных подсистемы отдел кадров ООО «Зеленый луч»

С точки зрения здравого смысла хорошим следует считать интерфейс, при работе с которым пользователь получает именно то, что он ожидал. Представление пользователя о функциях интерфейса можно описать в виде пользовательской модели интерфейса.

Пользовательская модель интерфейса - это совокупность обобщенных представлений конкретного пользователя или некоторой группы пользователей о процессах, происходящих во время работы программы или программной системы. Эта модель базируется на особенностях опыта конкретных пользователей, который характеризуется:

1)уровнем подготовки в предметной области разрабатываемого про­граммного обеспечения;

2)интуитивными моделями выполнения операции в этой предметной об­ласти;

3)уровнем подготовки в области владения компьютером;

4)устоявшимися стереотипами работы с компьютером.

При создании программной модели интерфейса также следует иметь в виду, что изменить пользовательскую модель непросто. Повышение профес­сионального уровня пользователей и их подготовки в области владения ком­пьютером в компетенцию разработчиков программного обеспечения не вхо­дит, хотя часто грамотно построенный интерфейс, который адекватно отоб­ражает сущность происходящих процессов, способствует росту квалифика­ции пользователей.

Язык SQL является важным звеном в архитектуре систем управления базами данных. SQL является инструментом, предназначенным для обработки и чтения данных, содержащихся в компьютерной базе данных. SQL - это сокращенное название структурированного языка запросов (Structured Query Language). Как следует из названия, SQL является языком программирования, который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных одного определенного типа, называемых реляционными.

4.3 Программное обеспечение

Операционная система Windows XP (кодовое название при разработке — Whistler; внутренняя версия — Windows NT 5.1) — операционная система семейства Windows NT корпорации [Microsoft](http://ru.wikipedia.org/wiki/Microsoft). Windows XP — наиболее широко используемая операционная система в мире.

За последние несколько лет вырос уровень потребительских качеств систем управления базами данных (СУБД): разнообразие поддерживаемых функций, удобный для пользователя интерфейс, сопряжение с программными продуктами, в частности с другими СУБД, возможности для работы в сети и т.д. СУБД позволяет сводить воедино информацию из самых разных источников (электронные таблицы, другие базы данных) и помогает быстро найти необходимую информацию, донести ее до окружающих с помощью отчетов, графиков или таблиц.

К настоящему времени накоплен значительный опыт проектирования БД, предназначенных для управления производством, это позволяет сделать процесс создания БД более эффективным.

Одной из наиболее распространенных СУБД является MS Access. Широкое применение именно этой СУБД для небольших офисных программ связано с тем, что она интегрирована в пакет прикладных программ MS Office, не требует большого объема памяти и достаточно проста в использовании.

СУБД Microsoft Access основана на использовании реляционной базы данных. Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Кроме описания структуры таблиц, обычно задаются связи между таблицами. Связи в реляционных базах данных определяются по совпадению значений полей в разных таблицах. База данных является основой информационной системы, которая позволяет пользователю хранить, обрабатывать и анализировать данные, обеспечивает удобный интерфейс, предоставляет ряд других возможностей.

СУБД Microsoft Access предоставляет разработчику интегрированные средства проектирования базы данных, создания запросов, создания и изменения форм, отчетов и страниц данных.

Прикладным программным обеспечением в данной курсовой работе будет являться разработанная база «отдел кадров.mdb».

Операционная система Microsoft Windows XP с пакетом обновления 2 (SP2) или более поздняя версия.

Компьютер и процессор. ПК с процессором 500 МГц или более, 256 или более МБ ОЗУ; дисковод для DVD-дисков; ПК с процессором 1 ГГц и 512 МБ ОЗУ (или более мощный компьютер) необходим для работы с Microsoft Office Outlook 2007 с диспетчером контактов.

Жесткий диск. Для установки необходимо 2ГБ; часть этого объема будет освобождена после установки, когда исходный установочный файл будет удален.

Разрешение экрана. Минимум 800x600 точек; рекомендовано 1024x768 или более.

Подключение к Интернету. Для загрузки и активации продуктов понадобится широкополосное подключение к Интернету со скоростью 128 кбит/с и выше.

**ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЙ**

Созданный файл базы данных получит наименование программы: Отдел кадров. Программа будет предназначена для организации работы информационной системы Отдел кадров.

Интерфейс пользователя - элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением. Для взаимодействия пользователя с базой данных подсистемы отдел кадров КИС ООО «Зеленый луч» разработан оконный интерфейс программы. Общий вид интерфейса показан на рисунке 11.

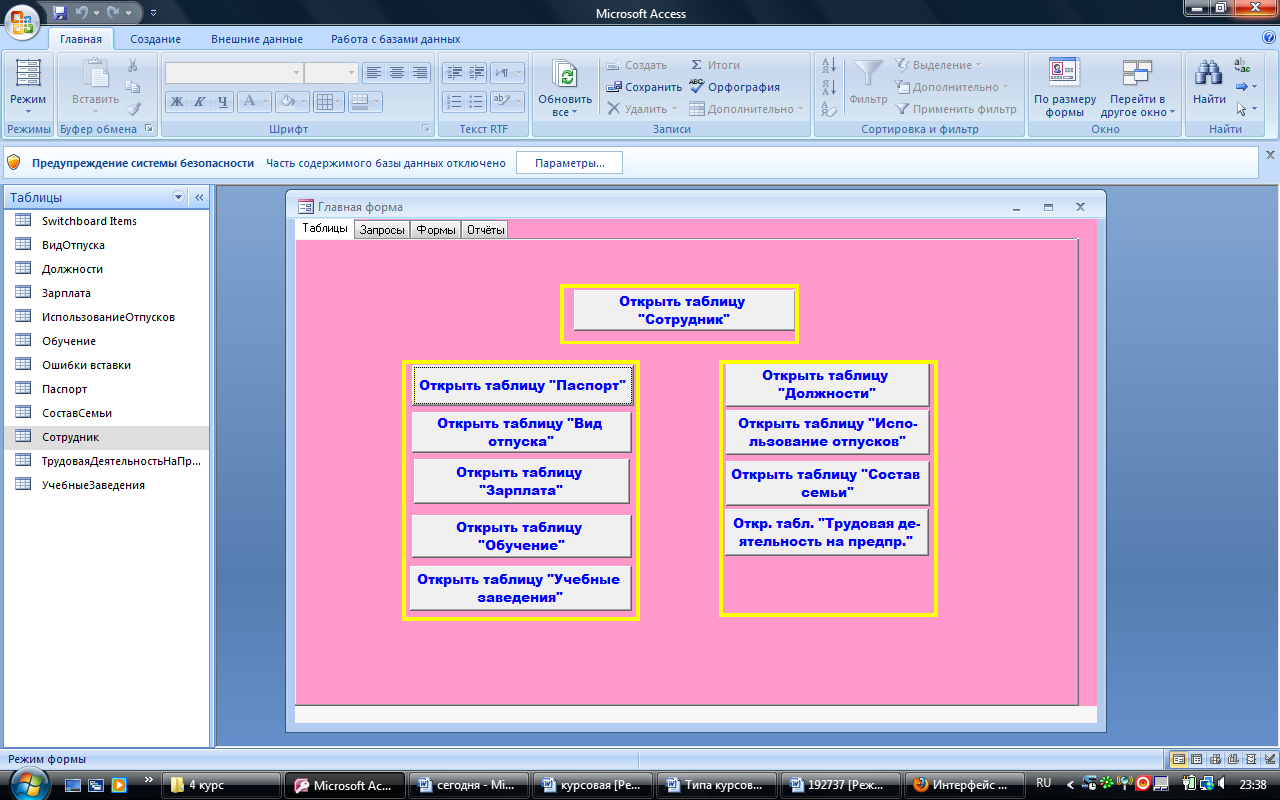


Рисунок 11 – Общий вид интерфейса подсистемы отдел кадров ООО «Зеленый луч»

В начале программы организуется цикл «Главная форма» и выводится меню с пунктами:

* таблицы;
* запросы;
* отчёты;
* формы.

Если выбран пункт «Таблицы», то открываются кнопки с всеми таблицами данной базы на их открытие.

Если выбран пункт «Запросы», то открывается окно с кнопочной формой на открытие запросов по отделу кадров.

Если выбран пункт «Отчеты», то вызывается модуль «Отчеты», который обеспечивает просмотр и печать отчетов, необходимых отделу кадров.

Теперь можно просмотреть форму в режиме форм. Готовая форма представлена на рисунке 12.

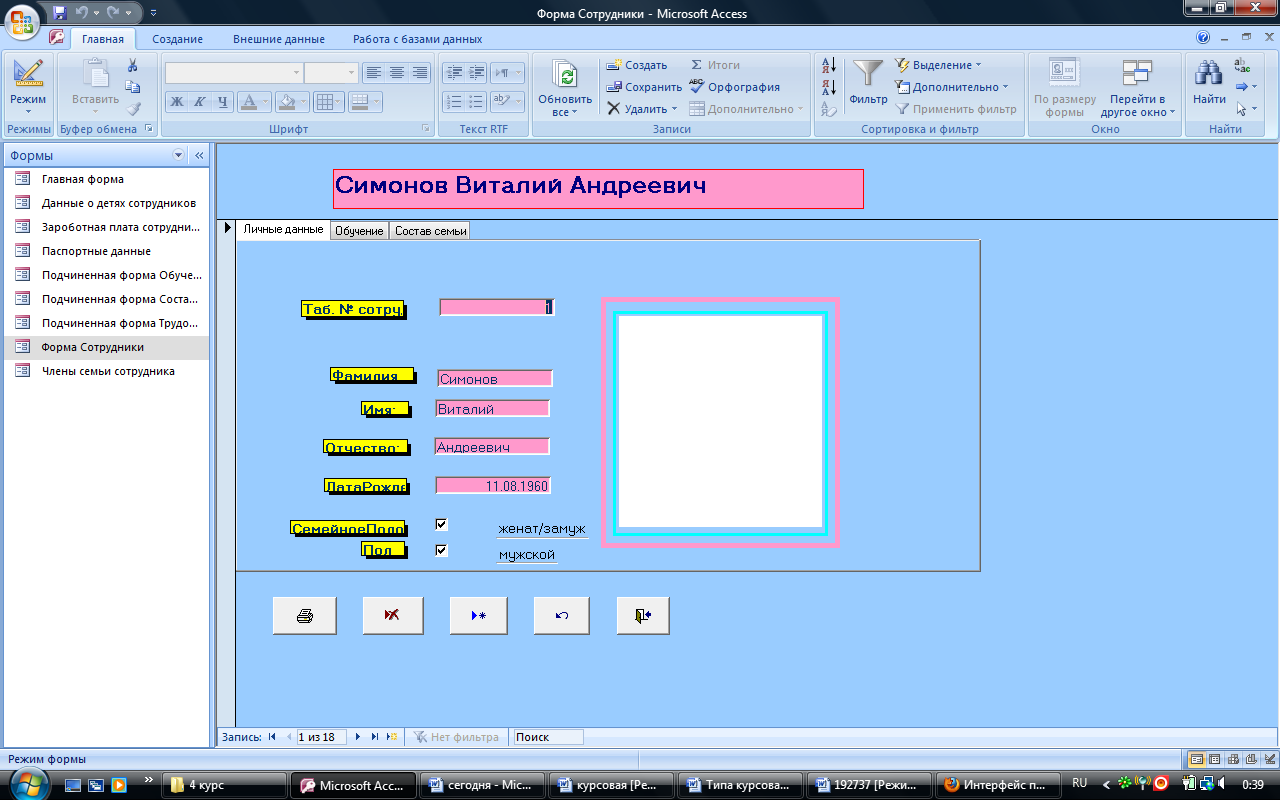


Рисунок 12 - Экранная форма для ведения справочника Сотрудники

Данная экранная форма и все последующие формы позволяют:

- просматривать и корректировать существующие данные;

- добавлять новые и удалять ненужные;

- осуществлять поиск по заданному условию и отправлять отчет на печать.

Кроме того, в данной базе данных существуют формы:

- Данные о детях сотрудника;

- Заработанная плата сотрудника;

- Члены семьи сотрудника.

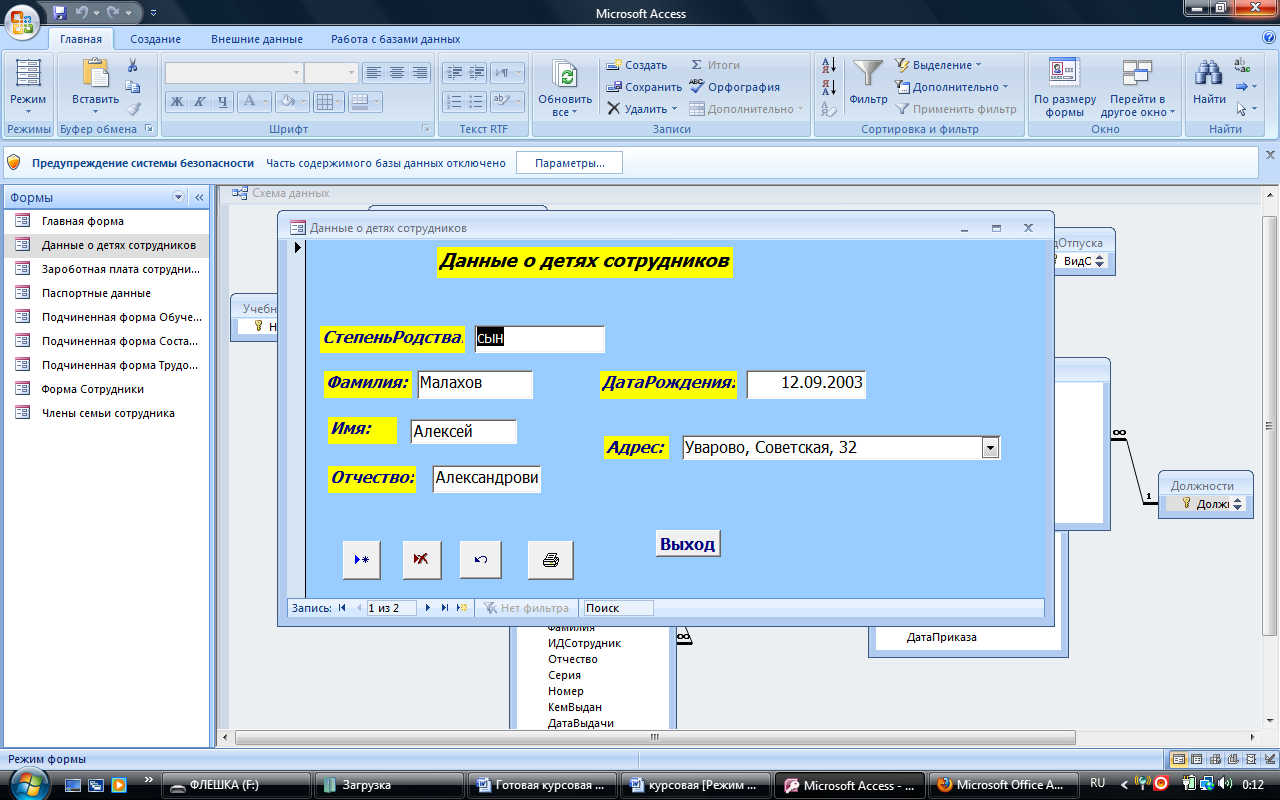


Рисунок 13 – экранная форма «Данные о детях сотрудников»

Данная форма открывается с помощью выбора фамилии сотрудника.

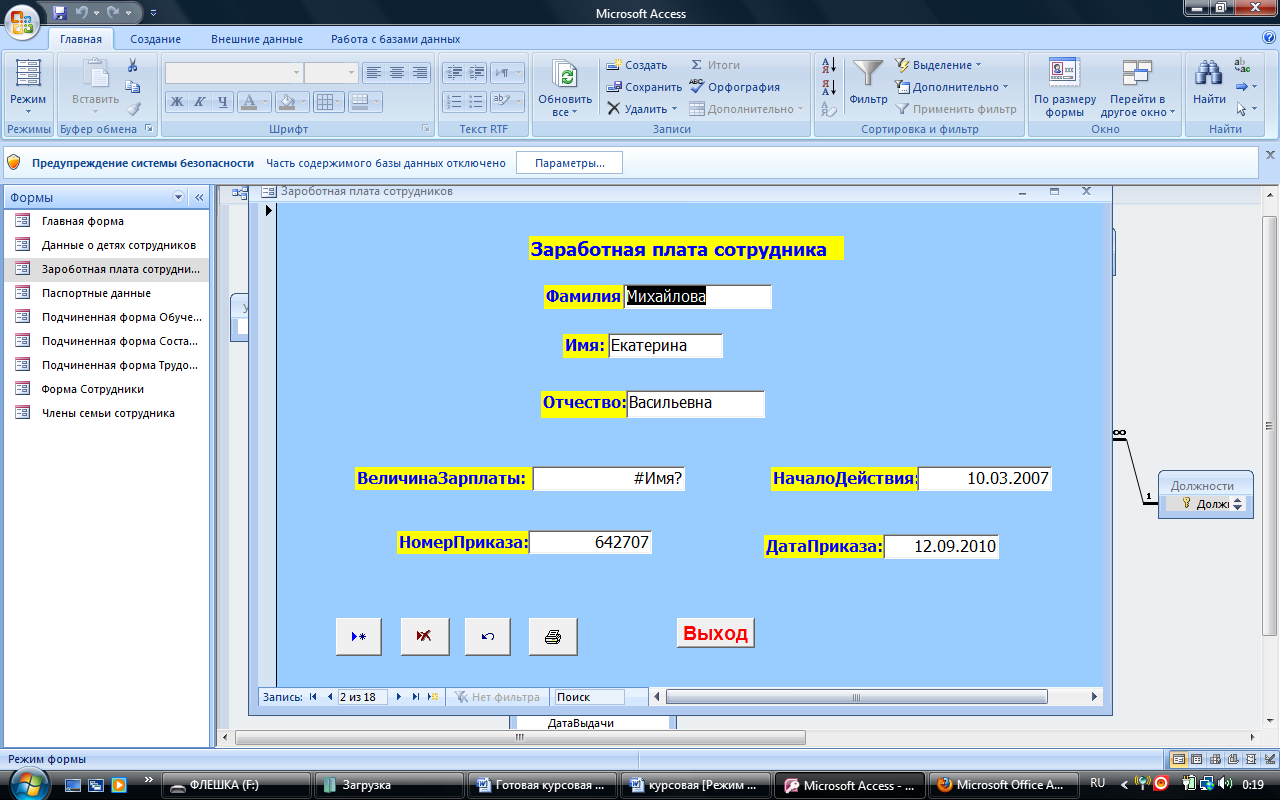


Рисунок 14 - экранная форма «Заработанная плата сотрудника»

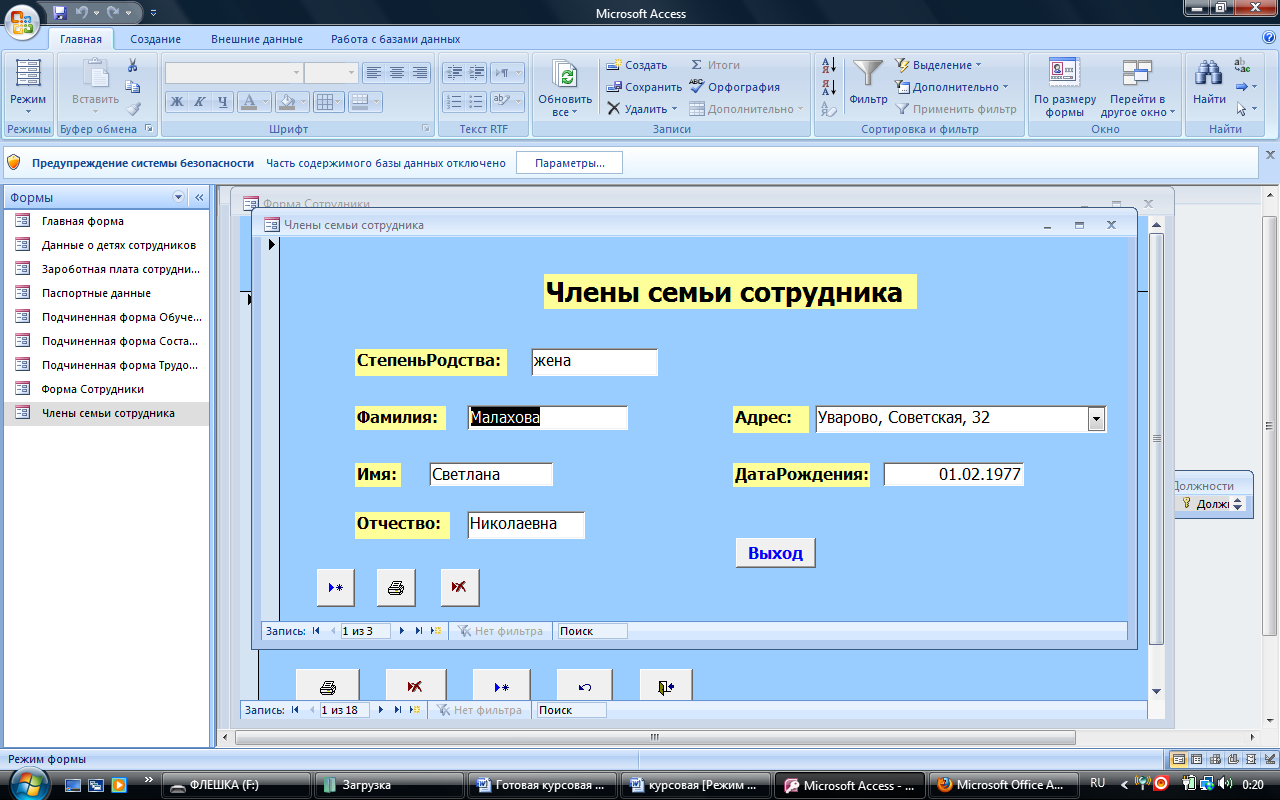


Рисунок 15 – экранная форма «Члены семьи сотрудника»

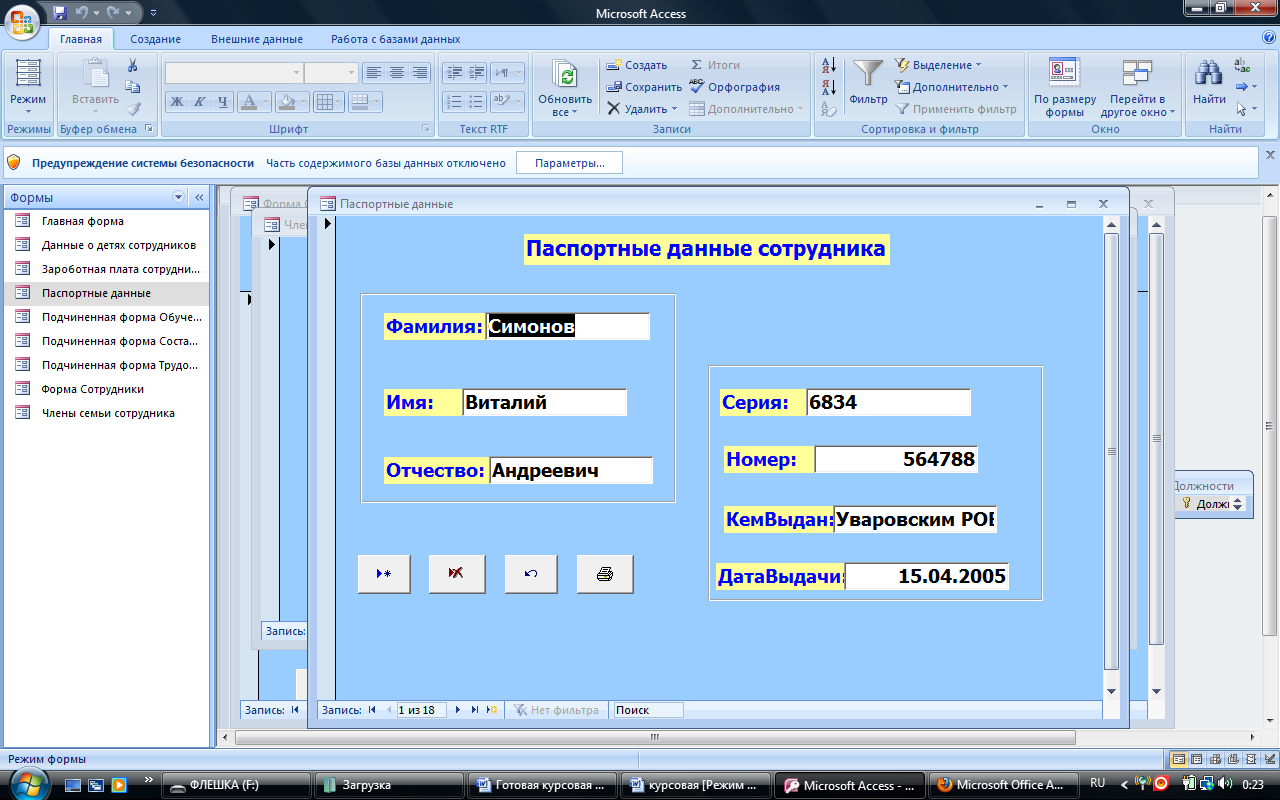


Рисунок 16 – экранная форма «Паспортные данные сотрудника»

Все экранные формы удобны в использовании и имеют кнопки печати, удаления информации, восстановления и добавления информации, а также перелистывание.

Также база данных осуществляет печать отчетов .В данном случае Это – данные паспорта сотрудника, зарплата, дата рождения, личные данные, использование отпусков, трудовая деятельность на предприятии, члены семьи, семейное положение. Пример отчета показан на рисунке 17.

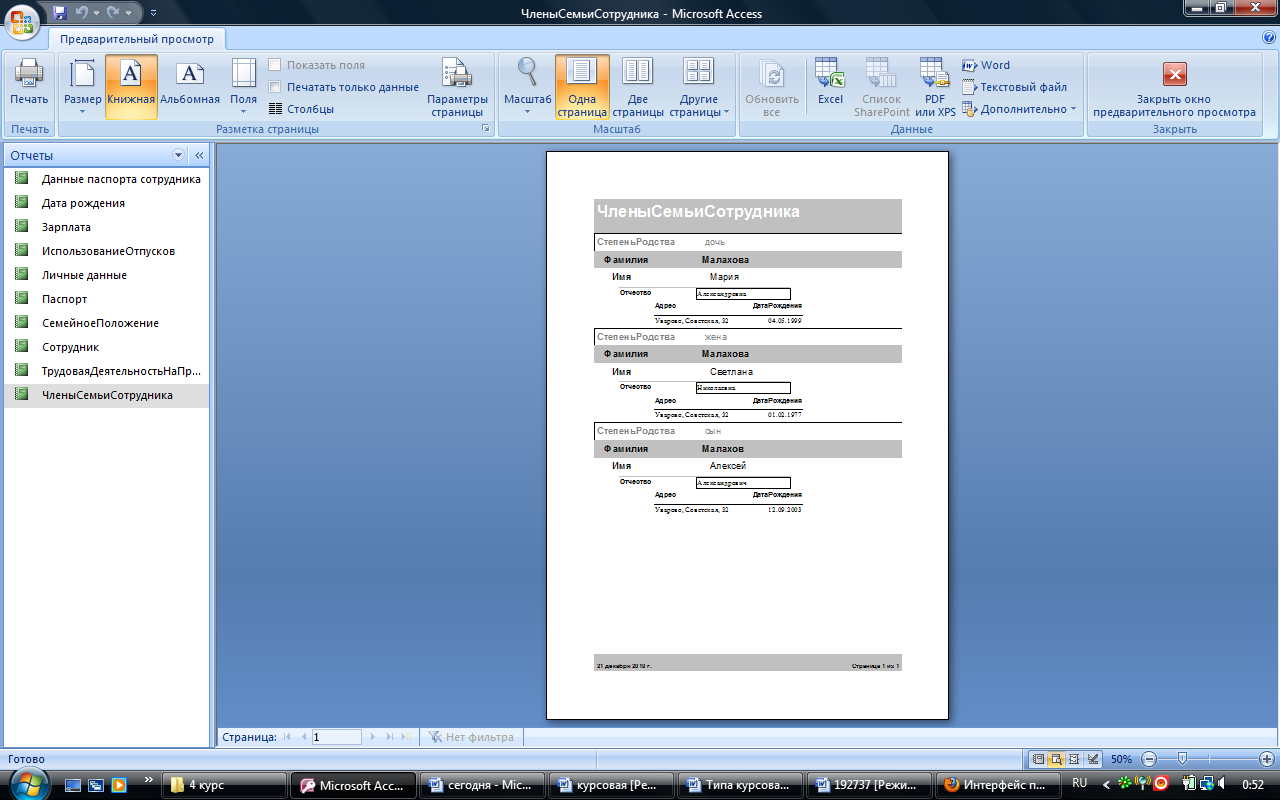
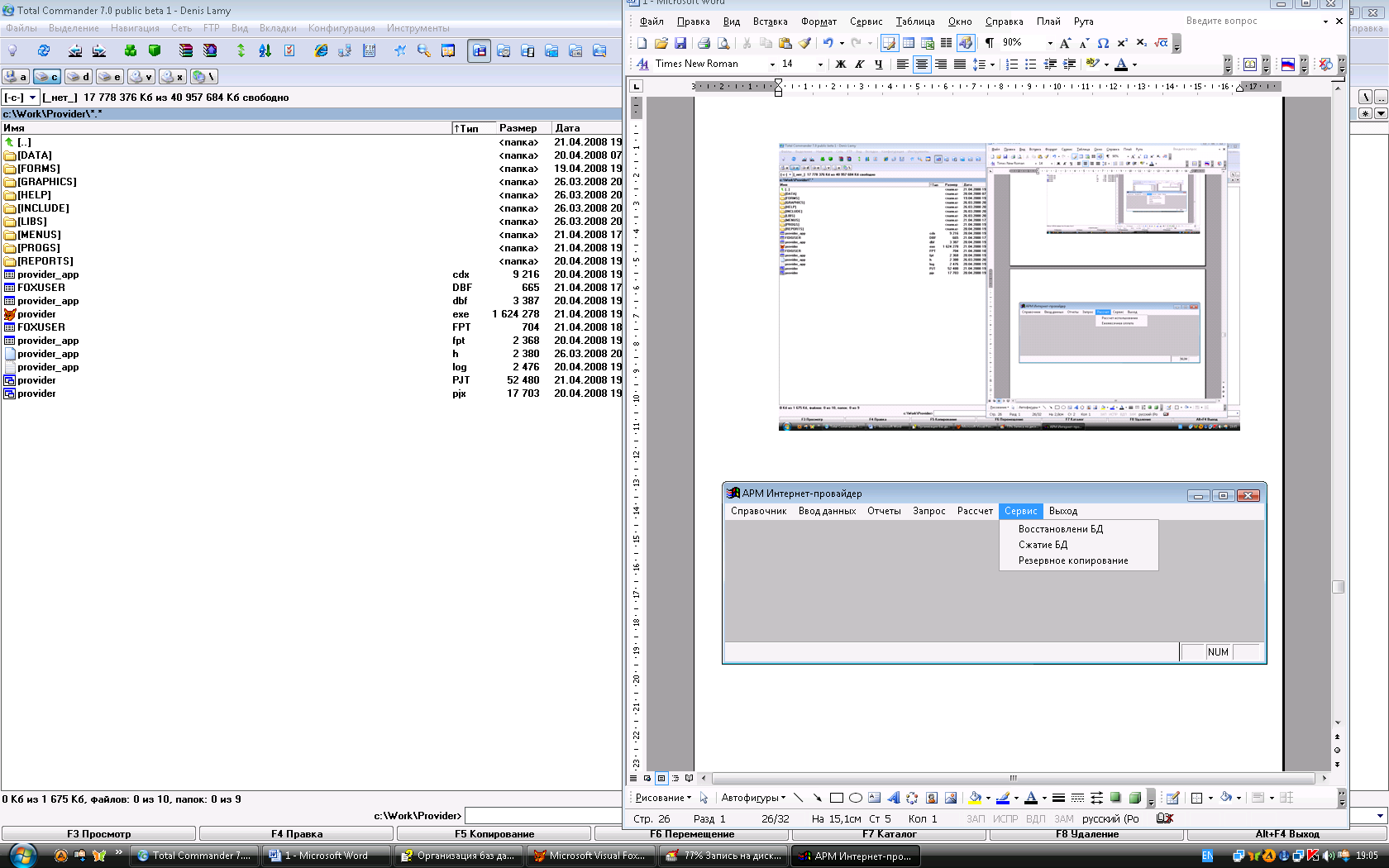


Рисунок 17 – Пример отчета «Члены семьи сотрудника»

По окончанию работы, необходимо закрыть все активные формы и выйти из программы. Для корректного закрытия базы данных и всего приложения в целом используется опция главного меню. Выходили кнопка закрытия окна программы с изображение крестика .

**ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ С ПРИМЕНЕНИЕМ ЯЗЫКА SQL**

**СОЗДАНИЕ И УДАЛЕНИЕ ТАБЛИЦ. ДОБАВЛЕНИЕ, ИЗМЕНЕНИЕ, УДАЛЕНИЕ ЗАПИСЕЙ В ТАБЛИЦЕ**

Для создания таблицы необходимо воспользоваться оператором create table, общий вид которого выглядит следующим образом:

create table <имя\_таблицы> [external [file] “Имя\_файла”]

(

<название\_столбца1> <тип1>,

<название\_столбца2> <тип2>,

…

<название\_столбцаN> <типN>

[<ограничения>,…]

)

где <имя\_таблицы> - название таблицы;

external [file] “Имя\_файла” – таблица может не храниться в базе данных, все ее строки могут помещаться в отдельный текстовый файл, находящийся вне базы данных;

<название\_столбца1>…<название\_столбцаN> - имя текущего поля;

<тип1>…<типN> - тип данных текущего поля;

<ограничения> - разнообразные ограничения, например, not null, что означает, что текущее поле не может принимать значение «null»;

[ ] – необязательный элемент.

Пример. Пусть необходимо создать таблицу с названием Planets. Таблица состоит из 8 столбцов: planetid – порядковый номер планеты, planetname – ее имя, radius – радиус планеты, rotateperiod – период обращения, temper – температура, excentr – эксцентриситет, g – ускорение свободного падения на планете, system – система, в которой находится эта планета.

Оформим таблицу в виде:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Planets | | | | | | | |
| Planetid | Planetname | Radius | Rotateperiod | Temper | Excentr | G | System |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |

Определим тип каждого столбца, в соответствии с хранимыми данными. Столбцы planetid и temper – целые беззнаковые, planetname и system – строки из максимум 30 символов, radius, excentr, g, rotateperiod – вещественные беззнаковые числа.

Применим оператор create table:

create table planets

(

planetid int identity(1,1) primary key,

planetname varchar(30) not null,

radius float check (radius>1000),

rotateperiod float,

temper int,

excentr float default 0,

g float,

system varchar(30) not null

)

В примере используется также много дополнительных параметров, таких как not null, check, default. Результат выполнения этого оператора – пустая таблица planets с указанными выше полями.

После создания таблицы нужно заполнить ее записями. Для внесения записи в таблицу используется инструкция insert into. Ее общий вид:

insert into <имя\_таблицы> values (<знач1>,<знач2>,… <значN>)

где <имя\_таблицы> - название таблицы;

<знач1>…<значN> - значения полей таблицы соответствующего типа.

Пример для таблицы planets:

insert into planets values ('Меркурий', 2440, 58.646, 623, 0.20530294, 3.7, 'Солнечная')

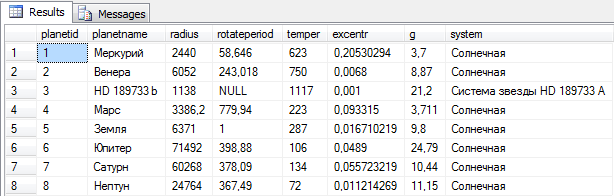
Заполнив таблицу, можно, используя оператор select, просмотреть все те записи, которые были в нее занесены. Оператор select для этого случая обладает следующей конструкцией:

select \* from <имя\_таблицы>

Выполнив запрос

select \* from planets

Получим:



Чтобы удалить запись таблицы, нужно воспользоваться следующим запросом:

delete from <имя\_таблицы> [where <условие\_поиска>]

Например, требуется удалить планеты с ускорением свободного падения g=3.7

delete from planets where g=3.7

Чтобы изменить какое либо поле в записи, пользуются инструкцией update.

Пример: update planets set g=4 where planetid=5

Этот запрос изменит значение поля g на 4 в записи, planetid которой – 5.

**КЛЮЧИ. СВЯЗЬ ДВУХ ТАБЛИЦ ПО ВНЕШНЕМУ КЛЮЧУ.**

Чтобы наглядно показать, как две таблицы можно связать по ключу, создадим новую таблицу planetsstats, используя запрос с уже знакомой нам конструкцией create table:

create table planetsstats

(

planetname varchar(30) primary key,

radius float not null check (radius>1000),

rotateperiod float,

temper int not null,

excentr float not null default 0,

g float not null

)

Для указания ключевого поля используется ограничение primary key, которое означает, в данном случае, что поле planetname будет ключом. Ключ не может принимать значение null. Ключ уникален, двух одинаковых в таблице быть не может.

Чтобы теперь связать с новой таблицей planetsstats другую таблицу, необходимо в новой таблице указать, какое поле станет внешним ключом. Внешний ключ должен быть того же типа, что и ключ первой таблицы. Кроме того, во вторую таблицу нельзя будет добавить записи, если в первой таблице поле primary key не содержит запись с таким значением.

Общий вид задания внешнего ключа выглядит следующим образом:

foreign key (<список\_столбцов\_внешнего\_ключа>) references <имя\_таблицы> [<список\_столбцов\_таблицы>] [on delete { no action | cascade | set default | set null }] [on update { no action | cascade | set default | set null]

Создадим теперь еще одну таблицу planetiden, связав ее с таблицей planetsstats внешним ключом:

create table planetiden

(

planetname varchar(30) foreign key references planetsstats(planetname) on delete cascade on update cascade,

albedo float not null check(albedo>0),

planetsystem varchar(30) not null default 'Солнечная'

)

Конструкция

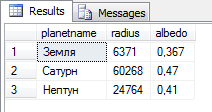
planetname varchar(30) foreign key references planetsstats(planetname) on delete cascade on update cascade,

означает, что поле planetname второй таблицы является внешним ключом (foreign key), связанным с соответствующим полем planetname (primary key) первой таблицы planetsstats. Предложение on delete cascade означает, что, удалив запись и первой таблицы, удалится также запись с соответствующим полем planetname во второй таблице. Аналогично работает on update cascade.

Теперь, чтобы проверить связь по ключу между двумя таблицами, составим запрос select, выводящий на экран поля из разных таблиц.

select planetsstats.planetname, planetsstats.radius, planetiden.albedo from planetsstats, planetiden where planetsstats.planetname = planetiden.planetname

Результат выполнения данного запроса:



Также, для сокращения и упрощения записи запроса можно использовать так называемые псевдонимы таблиц. Псевдонимы таблиц – короткие имена, которые можно использовать вместо много кратного повторения таких длинных названий как, например, planetsstats. Использованный выше запрос select можно записать теперь следующим образом:

select one.planetname, one.radius, two.albedo from planetsstats one, planetiden two where one.planetname = two.planetname

Использование псевдонимов *one* для planetsstats и *two* для planetiden позволило существенно сократить запись. Как видно из конструкции запроса, псевдоним пишется через пробел после названия таблицы, которой данный псевдоним дается.

**ИСПОЛЬЗОВАНИЕ ОПЕРАТОРА SELECT**.

Оператор select может употребляться с использованием различных дополнительных ограничений. Он позволяет произвести выборку данных, преобразуя полученные результаты, реализуя сложные условия выбора. Общий вид оператора select можно описать следующим образом:

select [distinct|all] {\*|<знач1>,…} from <табл1>,…

[where <условия\_поиска>]

[group by <столбец>]

[having <условия\_поиска>]

[union <оператор select>]

[plan <план\_выполнения\_запроса>]

[order by <список\_столбцов>]

где [distinct|all] означает, будут ли выводиться на печать повторяющиеся записи в таблицах.

<знач1>… - конкретные имена столбцов, которые будут выведены при выполнении оператора select.

<табл1>… - имена таблиц, из которых будут выводиться данные.

[where <условия\_поиска>] задает условие выборки из таблицы данных – выведены будут только те записи, которые удовлетворяют заданному после ключевого слова where условию.

[group by <столбец>] указывает, будут ли выведенные данные сгруппированы по какому-либо столбцу.

[having <условия\_поиска>] аналогично where, но с тем отличием, что в условии поиска должна стоять агрегатная функция, например, sum. Having является дополнительным параметром оператора order by.

[union <оператор select>] позволяет объединить несколько столбцов в один и вывести их на печать в виде одного столбца с заданным названием.

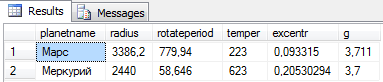
[order by <список\_столбцов>] упорядочивает данные в таблице по какому-то определенному столбцу. По умолчанию упорядочивание производится по возрастанию, но можно задать упорядочивание по убыванию с помощью ключевого слова desc.

Приведем некоторые примеры, используя созданные в предыдущей лабораторной работе таблицы planetsstats и planetiden:

select \* from planetsstats where planetname like 'М%'

Данный оператор выведет все данные из таблицы planetsstats, проверяя их на условие, указанное после ключевого слова where. Ограничение like означает, что из таблицы planetsstats будут выбраны все планеты, название которых начинается с буквы «М».

Результат выполнения этого запроса:



Также можно выводить определенный процент данных из таблицы

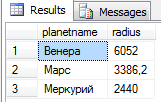
select top(50) percent \* from planetsstats

Конструкция top(50) означает, что из таблицы planetsstats будут показаны первые 50% записей.

select planetname, sum(radius) as radius from planetsstats where planetname like 'М%' or planetname like 'В%' or planetname like 'Н%' group by planetname having sum(radius)<6200

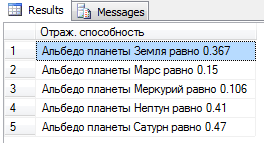
Этот запрос покажет имя и радиус тех планет из таблицы planetsstats, название которых начинается с «М», «В» или «Н», а радиус меньше 6200. Также записи будут сгруппированы по названию.

Результат выполнения выглядит следующим образом:



select distinct 'Альбедо планеты '+planetname+' равно '+convert(char(5), albedo) as 'Отраж. способность' from planetiden

Этот запрос выведет на печать таблицу следующего вида:



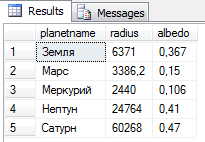
distict предотвращает использование в запросе повторяющихся записей из таблицы planetiden.

Агрегатная функция convert переводит данные из какого-либо столбца таблицы из одного типа в другой. В данном случае – из float в char(5)

select S.planetname, S.radius, I.albedo from planetsstats S inner join planetiden I on S.planetname = I.planetname

Данный запрос реализует *внутреннее соединение таблиц (inner join)*, то есть на печать выведется таблица, состоящая из различных полей из обеих таблиц – planetsstats и planetiden (в данном случае – из полей planetname, radius и albedo).

Результат выполнения этого запроса:



select count(distinct planetname) as 'Кол-во планет' from planetiden

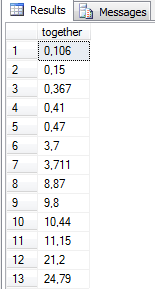
Агрегатная функция count показывает количество записей в таблице. Данный запрос показывает количество планет с различным названием из таблицы planetiden.

select S.g as together from planetsstats S

union

select I.albedo from planetiden I

Запрос реализует объединение столбцов. В результате значения столбца g из таблицы planetsstats и столбца albedo из таблицы planetiden запишутся в один столбец результирующей таблицы:

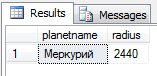


Теперь перейдем к рассмотрению вложенного запроса. Он представляет собой несколько операторов select, объединенных в один. Вложенные запросы употребляются в том случае, если нужно задать сложное, составное условие поиска.

Пример:

select S.planetname, S.radius from planetsstats S where S.radius in (select S1.radius from planetsstats S1 where excentr in(select max(S2.excentr) from planetsstats S2))

Вложенный запрос начинается после ключевого слова in. В данном случае в основной запрос select вложен еще один, в который, в свою очередь, вложен третий запрос. Запрос выведет на печать название планеты и ее радиус, если ее эксцентриситет (поле excentr) является максимальным:



**ИСПОЛЬЗОВАНИЕ ПРОСМОТРОВ (VIEWER) И РАЗЛИЧНЫЕ ОПЕРАЦИИ С НИМИ.**

Просмотры – это специальные виртуальые таблицы, которые могут объединять в себе несколько или все записи из одной или нескольких таблиц. Существуют несколько причин использования просмотров:

- однажды определив просмотр, не требуется всякий раз формировать определенный запрос select.

- просмотры усиливают безопасность данных базы.

Чтобы создать просмотр, необходимо воспользоваться оператором create view.

Общий вид оператора выглядит следующим образом:

create view <имя\_просмотра> [(<ст1>[,<ст2>,..])] as <опер. select> [with check option]

где <имя\_просмотра> - имя просмотра, которое мы хотим задать.

<ст1>… - список конкретных столбцов

as <опер. select> - каждый просмотр соответствует какой-либо выборке, которая указывается после ключевого слова as.

Вертикальный срез таблицы – выборка одного или нескольких столбцов таблицы для всех записей. Например, для таблиц planetstats и planetiden, использовавшихся в лабораторных работах 2 и 3, можно составить следующий запрос:

create view Vertical

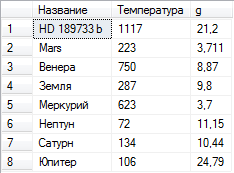
as

select planetname as Название, temper as Температура, g from planetsstats

Данный запрос создаст просмотр Vertical, в состав которого будут входить столбцы planetname, temper и g из таблицы planetsstats. После этого можно с помощью оператора select увидеть данные просмотра:

select \* from Vertical

Результатом выполнения запроса будет следующая таблица:



Горизонтальный срез таблицы – это несколько столбцов таблицы для одной ее записи. Чтобы создать просмотр, реализующий горизонтальный срез таблицы planetsstats, составим следующий запрос создания просмотра:

create view Horizontal

as

select excentr as Эксцентриситет, radius as Радиус from planetsstats where planetname='Нептун'

Данный запрос создаст просмотр Horizontal, в состав которого будут входить столбцы excentr и radius из таблицы planetsstats для записи с полем planetname равным «Нептун». После этого можно с помощью оператора select увидеть данные просмотра:

select \* from Horizontal

Результатом выполнения запроса будет следующая таблица:



Комбинированный срез – объединение предыдущих двух вариантов. Чтобы создать его, сформируем следующий запрос:

create view Verhor

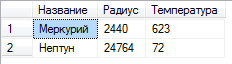
as

select planetname as Название, radius as Радиус, temper as Температура from planetsstats where planetname like 'М%' or planetname like 'Н%'

Созданный просмотр будет содержать в себе поля planetname, radius и temper из таблицы planetsstats для тех записей, имя (planetname) которых начинается с букв «М» или «Н». Выведем данные просмотра:

select \* from Verhor

Получим следующую таблицу:



Теперь объединим в одном просмотре данные разных таблиц. Создадим просмотр Summar, используя следующий запрос:

create view Summar

as

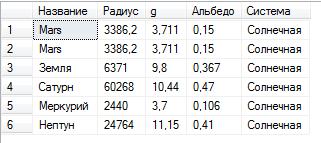
select ps.planetname as Название, ps.radius as Радиус, ps.g, pli.albedo as Альбедо, pli.planetsystem as Система from planetsstats ps, planetiden pli

where ps.planetname = pli.planetname

Также были использованы псевдонимы таблиц для краткости записи. Теперь с помощью оператора select выведем, что содержит в себе запрос:

select \* from Summar

Получим таблицу:



Чтобы удалить просмотр, нужно использовать оператор drop view:

drop view <имя\_просмотра>

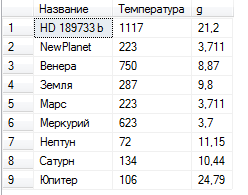
Например:

drop view Vertical

Теперь попробуем добавить данные в таблицу planetsstats и затем вывести данные просмотра Vertical, чтобы проанализировать произошедшие изменения. Добавим новую планету «NewPlanet»:

insert into planetsstats values ('NewPlanet', 3386.2, 779.94, 223, 0.093315, 3.711)

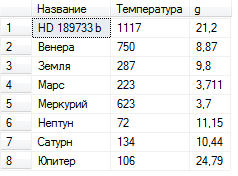
После чего выведем на печать данные просмотра Vertical:



Как можно заметить, изменение таблицы ведет к изменению просмотра. Добавление данных в таблицу добавляет их также и в просмотр. Точно так же работает удаление данных из таблицы:

delete from planetsstats where planetname='NewPlanet'

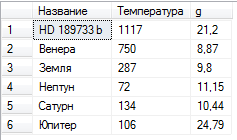
Получим в просмотре Vertical:



Теперь попробуем добавить и удалить данные из просмотров. Чтобы удалить данные из просмотра Vertical, используем следующий запрос:

delete from Vertical where Название like 'М%'

После чего выведем данные просмотра:



Как можно заметить, записи, название которых начиналось с буквы «М», удалились из просмотра. Но при попытке добавления записи в просмотр программа будет выдавать ошибку. Составим следующий запрос, чтобы проверить это:

insert into Vertical values ('New',1000,19.7)

Результат запроса:

*Cannot insert the value NULL into column ‘radius’, table ‘master.dbo.planetsstats’; column does not allow nulls. INSERT fails.*

*The statement has been terminated.*

**РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ**

Клиент-серверное приложение – это приложение, которое связывается с сервером – базой данных, позволяя не знающим языка SQL пользователям выполнять запросы для базы данных, добавлять и удалять оттуда данные.

Чтобы облегчить пользователю работу, нужно создать планируемые запросы – запросы для ваших таблиц, которые использовались бы чаще всего. Такие запросы необходимо выделить в отдельные кнопки.

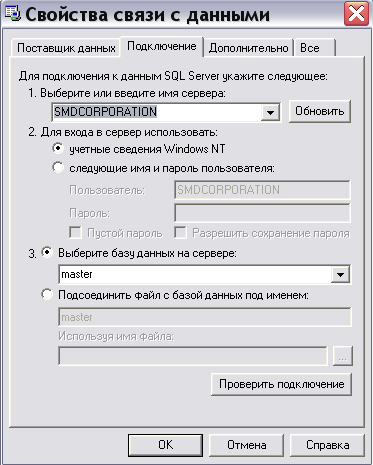
Пример – клиент-серверное приложение на языке С++.

Чтобы создать клиент-серверное приложение, прежде всего нужно связаться с созданной базой данных. Используем ту же базу данных с теми же таблицами, что и в прошлых лабораторных работах. В языке С++ это можно сделать, добавив на форму компонент под названием «ADOConnection» на вкладке «ADO».



Это – соединение, позволяющее подключиться к базе данных. После добавления на форму этого элемента, нужно указать ему имя вашего компьютера. Для этого в параметре элемента «ConnectionString» нужно нажать на кнопку со значком многоточия, нажать кнопку «Build» и указать имя вашего компьютера:





Следующий шаг – добавление элементов «ADOQuery» и «DataSource» со вкладок «ADO» и «Data Access» соответственно.

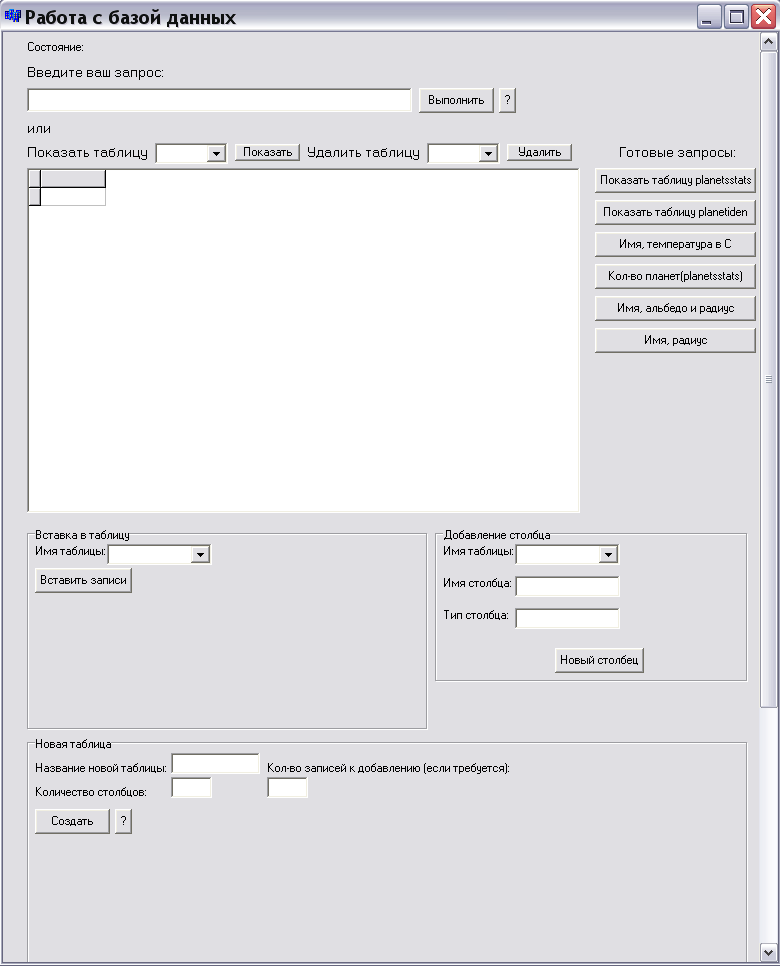
В параметрах «ADOQuery» в строке «Connection» необходимо указать имя вашего элемента «ADOConnection».



Соответственно, для параметра «DataSet» элемента «DataSource» необходимо указать имя вашего элемента «ADOQuery».



Затем создается интерфейс приложения. Интерфейс должен быть удобным и проработанным, чтобы пользователи могли работать и легко ориентироваться в нем. Интерфейс может быть, например, таким:



Чтобы выполнить какую-либо команду на языке SQL, в языке С++ существуют два оператора:

ADOQuery1->SQL->Text = resquery;

ADOQuery1->ExecSQL();

Где resquery – текстовая строка, содержащая в себе текст SQL-запроса. Стоит отметить, что подобный оператор языка С++ употребляется, если выполняемый запрос каким-либо образом изменяет таблицу. Если же требуется только вывести данные из таблицы с помощью оператора select, пользуются следующим оператором:

ADOQuery1->SQL->Add(resquery);

Где resquery – строка, содержащая в себе запрос с оператором select.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Пушников А.Ю. Введение в системы управления базами данных. Режим доступа: [http://www.ishodniki.ru/booksonline/db/books/books.php?to=dblearn08]
2. Дьяков И.А. Базы данных. Язык SQL: Учеб. пособие. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2004. 80 с. Режим доступа: [ http://window.edu.ru/window/library]
3. Дж. Ульман. Основы систем баз данных. Режим доступа: [http://reslib.com/book/Osnovi\_sistem\_baz\_dannih].
4. Кириллов В.В. Основы проектирования реляционных баз данных. Режим доступа: [http://citforum.ru/database/dbguide/index.shtml].
5. Зеленков Ю.А. Введение в базы данных. Режим доступа: [http://www.mstu.edu.ru/education/materials/zelenkov/toc.html]
6. Филлипович А. Принципы взаимных функциональных зависимостей. Режим доступа: [http://iu5.bmstu.ru/~philippovicha/Articles/MFD.htm]
7. Дьяков И.А. Базы данных. Язык SQL. Учебное пособие. Зарегестрировано в мультимедийной учебно-информационной системе VitaLMS интерактивного обучения ТГТУ в 2008 г. Режим доступа: [http://vitalms.tstu.ru/].
8. **Дьяков, И.А.** Базы данных. Язык SQL. Учебное пособие. Тамбов. Издательство ТГТУ, 2012. (exe-файл)