МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ГРАЖДАНСКОЙ АВИАЦИИ

А.А. Егорова

БАЗЫ ДАННЫХ

Учебно-методическое пособие
по проведению практических занятий

для студентов IV курса
направления 01.03.04
очной формы обучения

Москва
2017

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра прикладной математики
А.А. Егорова

БАЗЫ ДАННЫХ

Учебно-методическое пособие
по проведению практических занятий

для студентов IV курса
направления 01.03.04
очной формы обучения

Москва-2017

ББК 6ф7.3
Е30

Рецензент канд. техн. наук, доц. В.М. Коновалов

Егорова А.А.

Е30 Базы данных: учебно-методическое пособие по проведению практических занятий. – М.: МГТУ ГА, 2017. – 36 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Базы данных» по учебному плану для студентов IV курса направления 01.03.04 очной формы обучения.

Рассмотрено и одобрено на заседании кафедры 24.10.2017 г. и методического совета 24.10.2017 г.

Подписано в печать 10.11.2017 г.

Печать офсетная	Формат 60x84/16	1,55 уч.-изд. л.
2,08 усл.печ.л.	Заказ № 1725/257	Тираж 40 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д.20

ООО «ИПП «ИНСОФТ»
107140, Москва, 3-й Красносельский переулок, д. 21, стр. 1

ПРЕДИСЛОВИЕ

Настоящее пособие содержит справочный материал, необходимый при выполнении заданий на практических занятиях по дисциплине «Базы данных», проводимых у студентов IV курса направления подготовки «Прикладная математика», а также при выполнении курсовой работы по дисциплине.

Навыки, приобретенные на практических занятиях, необходимы студентам в процессе подготовки не только лабораторных работ и курсовой работы по дисциплине, но и выпускной квалификационной работы, а также далее в процессе самостоятельной работы на предприятиях.

В пособии отражены организационно-методические аспекты работы на практических занятиях, цели и задачи, достигаемые в процессе работы, формируемые компетенции.

Пособие охватывает полный курс и содержит цель и задание, выполняемое на каждом занятии, краткие теоретические сведения, контрольные вопросы, список рекомендуемой литературы.

Минимально необходимый справочный материал, содержащийся в пособии, не претендует на полноту, а касается только необходимых в рамках данной дисциплины аспектов, которые могут быть полезны, в том числе и для дальнейшей самостоятельной работы

Настоящее пособие может быть использовано и как справочник при самостоятельной работе по разработке информационной системы.

Пособие имеет прикладной характер, что способствует формированию у студентов компетенций в соответствии с требованиями, содержащимися в рабочей программе по дисциплине «Базы данных», и в целом соответствующие модели компетенций по направлению подготовки «Прикладная математика».

Оглавление

1. Введение

На сегодняшний день применение баз данных приобрело большое значение практически для всех организаций, поскольку без использования компьютерных технологий трудно представить их работу. Базы данных стали основой информационных систем, которые в последние годы были внедрены в производственную деятельность. Развитие технологии баз данных вместе с развитием аппаратных средств привело к созданию весьма мощных и удобных в эксплуатации систем, обеспечив к ним широкий доступ пользователей.

Эффективное хранение, обработка и взаимодействие с данными - важная составляющая управления предприятием, компании инвестируют значительные средства в разработку компьютеризированных системы для эффективного решения этих задач. Один из способов повышения эффективности обработки данных — организовать их эффективное хранение и извлечение. Один из самых распространенных подходов к хранению данных на сегодняшний день – использовать реляционную базу данных.

База данных – это такая совокупность данных, которая организована в соответствии с определёнными правилами и имеющая определённую структуру. Она редактируется при помощи системы управления базами данных (СУБД).

СУБД – это программное обеспечение, которое позволяет создавать БД, редактировать их, выполнять различные манипуляции с ними, а также удалять их.

Реляционная база данных – это совокупность взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Строка таблицы содержит данные об одном объекте, а столбцы таблицы описывают различные характеристики этих объектов – атрибутов. Записи, т. е. строки таблицы, имеют одинаковую структуру – они состоят из полей, хранящих атрибуты объекта. Каждое поле, т. е. столбец, описывает только одну характеристику объекта и имеет строго определенный тип данных. Все записи имеют одни и те же поля, только в них отображаются различные информационные свойства объекта.

Правильно спроектированная база данных (в первую очередь построение логической модели, определяющей перечень таблиц и установление взаимосвязи между ними, а также выявление атрибутов) – залог успеха при ее программировании и функционировании.

Настоящее пособие предназначено для студентов при выполнении заданий по дисциплине «Базы данных» как в процессе практических занятий, так и при самоподготовке, в первую очередь при проектировании базы данных и ее программировании в части формирования запросов.

Целью проведения практических занятий является как закрепление основных теоретических положений, изложенных в лекциях, так и получение

практических навыков по проектированию баз данных, необходимых студентам в том числе при подготовке курсовой работы.

Пособие необходимо студентам в течение всего семестра и является дополнением к пособиям по выполнению лабораторных работ и курсовой работы.

2. Организационно-методические рекомендации

В соответствии с учебным планом подготовки студентов по направлению 01.03.04 «Прикладная математика» (бакалавриат) и рабочей программой по дисциплине «Базы данных» и изложенными в них требованиями к уровню подготовки инженеров для работы в организациях ГА, студенты должны обладать:

- навыками работы с ПК;
- навыками программирования на одном из языков высокого уровня;
- навыками работы с прикладными пакетами программ.

Каждому практическому занятию должна предшествовать лекция по соответствующей теме.

В соответствии с учебной программой продолжительность практических занятий – 12 часов (6 практических занятий по 2 часа).

В процессе выполнения заданий студенты должны научиться:

- проектировать структуру баз данных, приводя их к третьей нормальной форме (или усиленной третьей нормальной форме);
- проводить анализ разработанной структуры с учетом классов принадлежности сущности, используя различные методы;
- ориентироваться в программных продуктах и методах для автоматизированного проектирования процессов, поддерживаемых базой данных (информационной системой);
- формировать запросы и оперировать объектами и данными средствами языка структурированных запросов SQL.

2.1. Компетенции обучающегося, формируемые в результате освоения дисциплины «Базы данных»

Студент:

- имеет способность к самоорганизации и самообразованию (ОК-7);
- готов к самостоятельной работе (ОПК-1);
- способен использовать современные математические методы и современные прикладные программные средства и осваивать современные технологии программирования (ОПК-2);
- готов применять знания и навыки управления информацией (ПК-11).

В результате изучения дисциплины «Базы данных» студент должен:

- по компетенции ОК-7 знать основные направления развития программных систем в части проектирования и поддержки баз данных (ОК-7.1.8).
- по компетенции ОПК-1 знать методы и средства для решения задач, связанных с сбором, систематизацией и организацией информации, в том числе в части построения информационных систем на основе баз данных (ОПК-1.1.15) и уметь самостоятельно принимать решения по выбору средств и методов решения поставленной задачи (ОПК-1.2.13), а также иметь навыки самостоятельного планирования последовательности решения задачи проектирования и оформления информационной системы (ОПК-1.3.7);
- по компетенции ОПК-2 знать основные типы и классификацию ПО для разработки и поддержки баз данных (ОПК-2.1.7) и уметь работать в современных программных системах для разработки баз данных и процедур их поддержки и администрирования (ОПК-2.2.6).
- по компетенции ПК-11 знать:
  - ✓ способы проектирования баз данных, модели баз данных (ПК-11.1.4);
  - ✓ программные системы для разработки баз данных и пользовательского интерфейса (ПК-11.1.5);
- по компетенции ПК-11 уметь:
  - ✓ проектировать реляционные базы данных одним из способов (ПК-11.2.5);
  - ✓ программировать в одной из доступных сред (MS SQL, Delphi, MS Access и т.п.) (ПК-11.2.6);
  - ✓ разрабатывать пользовательский интерфейс и ПО для работы с БД и ее поддержки (ПК-11.2.7).
- по компетенции ПК-11: иметь навыки администрирования баз данных (ПК-11.3.2).

2.2. Перечень тем практических занятий

ПЗ - 1. Проектирование баз данных методом нормальных форм.
ПЗ - 2. Использование модели «сущность-связь» при проектировании баз данных.
ПЗ - 3. Автоматизация процесса разработки баз данных.
ПЗ - 4. Основные операторы языка SQL.
ПЗ – 5. Формирование запросов к базам данных на языке SQL.
ПЗ – 6. Формирование триггеров и курсоров средствами языка SQL.

2.3. Подготовка к практическому занятию

Для успешного выполнения заданий преподавателя на практическом занятии студенту необходимо:

- изучить лекционный материал, предшествующий практическому занятию по теме, определенной в настоящем пособии для каждого занятия;
- получить у преподавателя вариант задания на выполнение лабораторных работ и курсовой работы для отработки приемов работы на примере своего варианта;
- подготовить вопросы преподавателю по уточнению задания (для отработки навыков проведения интервью Заказчика информационной системы);
- сформулировать назначение информационной системы и роли ее пользователей.

Все задания, выполняемые в процессе практического занятия, являются подготовительными для выполнения курсовой работы и при определенной доработке и оформлении являются частью пояснительной записки.

2.4. Отчет по выполнению заданий на практическом занятии

В конце практического занятия студент демонстрирует преподавателю выполненное задание, отвечает на контрольные вопросы, поясняет процесс выполнения задания, принятые решения, демонстрирует используемые приемы.

Все продемонстрированные задания преподаватель учитывает при выставлении оценки по курсовому проектированию, так как они являются, большей частью, подготовительными для его корректного выполнения.

3. Практическое занятие №1

Проектирование баз данных методом нормальных форм

3.1. Цель занятия

Целью практического занятия является приобретение навыков:

- проектирования баз данных методом нормальных форм;
- практической работы с «представителем заказчика» и интервьюирования его;
- выделения основных сущностей;
- формирования атрибутов сущностей с учетом входных и выходных документов, а также бизнес правил.

3.2. Задание на практическое занятие

Разработать структуру базы данных авиаперевозок в части управления экипажами воздушных судов, приведенной к третьей нормальной форме.

База данных должна содержать сведения о следующих объектах:

- Расписание авиарейсов – номер, период выполнения, а/п вылета, а/п прилета, время вылета, время прилета, дни оперирования (недели).
- Авиарейсы – номер, дата, тип ВС, количество членов летного и кабинного экипажей.
- Аэропорты – страна, город, наименование аэропорта, категория посадки.
- ВС – количество и должности членов летного экипажа, количество членов кабинного экипажа (по классам обслуживания).
- Члены летных экипажей – ФИО, должность, тип ВС, категории посадки (I, II или III), даты отпусков на текущий год, дата годового медосмотра, больничные листы (номер, период нетрудоспособности), контактный телефон.
- Члены кабинных экипажей – ФИО, должность, типы ВС (не более 5), даты отпусков на текущий год, дата годового медосмотра, больничные листы (номер, период нетрудоспособности), контактный телефон.

Выходные документы:

- Расписание на месяц конкретного члена летного или кабинного экипажа,
- Задание на полет, выдаваемое командиру ВС,
- Расписание на день по всем рейсам (для диспетчера) с телефонами членов экипажа.

Бизнес-правила:

- Считаем, что классов обслуживания только 2 (экономический и бизнес класс),
- Между рейсами у членов экипажа не может быть менее 12 часов,
- Нельзя ставить на 2 подряд ночных рейса,
- Ночью считается период с 23 часов до 6 утра по московскому времени,
- В месяц налет одного человека не должен превышать 80 часов, в год - 800.

3.3. Краткие теоретические сведения

3.3.1. Элементы реляционной модели данных

Сущность – объект, сведения о котором нужно сохранить; фактически – таблица. Реляционная база данных состоит из сущностей (таблиц), находящихся в некотором отношении друг с другом. (Термин «реляционный» означает «основанный на отношениях»).

Атрибут — свойство некоторой сущности. Часто называется полем таблицы.

Домен атрибута — множество допустимых значений, которые может принимать атрибут.

Кортеж — конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность (строка таблицы).

Отношение — конечное множество кортежей (таблица).

Схема отношения — конечное множество атрибутов, определяющих некоторую сущность. Иными словами, это структура таблицы, состоящей из конкретного набора полей.

Проекция — отношение, полученное из заданного путём удаления и (или) перестановки некоторых атрибутов.

Первичный ключ — поле или комбинация полей, которые единственным образом идентифицируют каждую строку таблицы. Если ключ состоит из нескольких полей, он называется составным. Ключ должен быть уникальным и однозначно определять запись. По значению ключа можно отыскать единственную запись. Ключи служат также для упорядочивания информации в БД.

Внешний ключ – поле таблицы, предназначенное для хранения значения первичного ключа другой таблицы с целью организации связи между этими таблицами.

Индекс (вторичный индекс) - объект базы данных, создаваемый с целью повышения производительности поиска данных. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и, таким образом, позволяет искать строки, удовлетворяющие критерию поиска.

Над реляционными таблицами возможны следующие **операции**:

- Объединение таблиц. Результат - общая таблица: сначала первая, затем вторая (конкатенация).
- Пересечение таблиц. Результат - записи, которые находятся в обеих таблицах.
- Вычитание таблиц. Результат - записи, которых нет в вычитаемом.
- Выборка (горизонтальное подмножество). Результат - записи, отвечающие определенным условиям.
- Проекция (вертикальное подмножество). Результат - отношение, содержащее часть полей из исходных таблиц.
- Декартово произведение двух таблиц Записи результирующей таблицы получаются путем объединения каждой записи первой таблицы с каждой записью другой таблицы.

Объединение, пересечение и вычитание выполняются для таблиц с одинаковой структурой.

Реляционные таблицы могут быть связаны друг с другом, а данные – извлекаться одновременно из нескольких таблиц. Связь каждой пары таблиц обеспечивается при наличии в них одинаковых столбцов.

Существуют следующие **типы информационных связей**:

- один-к-одному (1 : 1);
- один-ко-многим (1 : М);
- многие-ко-многим (М : М).

Связь 1 : 1 предполагает, что одному атрибуту первой таблицы соответствует только один атрибут второй таблицы и наоборот.

Связь 1 : М предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы.

Связь М : М предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы и наоборот.

3.3.2. Зависимости между атрибутами

Метод нормальных форм основан на фундаментальном в теории реляционных баз данных понятии зависимости между атрибутами отношений.

Рассмотрим основные виды зависимостей: функциональные, транзитивные, многозначные.

Понятие функциональной зависимости является базовым, так как на его основе формулируются определения всех остальных видов зависимостей.

Функциональная зависимость между атрибутами (множествами атрибутов) А и В означает, что для любого допустимого набора кортежей в данном отношении: если два кортежа совпадают по значению А, то они совпадают по значению В. Обозначается функциональная зависимость А -> В.

Частичной функциональной зависимостью называется зависимость неключевого атрибута от части составного ключа.

Полная функциональная зависимость - зависимость неключевого атрибута от всего составного ключа.

О **транзитивной зависимости** можно говорить, когда атрибут С зависит от атрибута В, который в свою очередь зависит от атрибута А. Или если для атрибутов А, В и С выполняются условия А – > В и В -> С, но обратная зависимость отсутствует. Например, ФИО -> должность -> оклад.

Многозначная зависимость имеет место в отношении, если каждому значению атрибута А соответствует множество значений В, не связанных с другими атрибутами этого отношения. Многозначные отношения могут быть 1 : М и М : М, которые обозначаются соответственно А => В и А <=> В. Иногда рассматривают отношение М : 1 (А <= В).

Существует также понятие взаимно независимых атрибутов. Два и более атрибутов называются взаимно независимыми, если ни один из этих атрибутов не является функционально зависимым от других атрибутов.

Основной способ определения наличия функциональных зависимостей – анализ семантики атрибутов. Практически в каждом отношении существует некоторое количество функциональных зависимостей между атрибутами. Причем, если в некотором отношении существует одна или несколько функциональных зависимостей, то можно вывести другие функциональные зависимости, существующие в этом отношении. Для выявления всех функциональных зависимостей, существующих в отношении, необходимо знать ряд правил (или аксиом) вывода одних функциональных зависимостей из других. Существует 8 основных аксиом: рефлексивности, пополнения, транзитивности, расширения, продолжения, псевдотранзитивности, объединения и декомпозиции.

Далее представлены восемь аксиом вывода функциональных зависимостей.

1. Рефлексивность. Если $X \subseteq U, Y \subseteq U, Y \subseteq X$, то ФЗ $X \to Y$ следует из F. Иначе $X, X \to X$.
2. Пополнение. Если $X \subseteq U, Y \subseteq U, Z \subseteq U$ и задана ФЗ $X \to Y$ из F, то имеет место ФЗ $X \cup Z \to Y \cup Z$.
3. Транзитивность. Если $X \subseteq U, Y \subseteq U, Z \subseteq U$ и задана ФЗ $X \to Y, Y \to Z$ из F , то имеет место ФЗ $X \to Z$.
4. Расширение. Если $X \subseteq U, Y \subseteq U$ и задана ФЗ $X \to Y$, то $Z \subseteq U$ имеет место ФЗ $X \cup Z \to Y$.
5. Продолжение. Если $X \subseteq U, Y \subseteq U, W \subseteq U, Z \subseteq U$, и задана ФЗ $X \to Y$, то $\forall W \subseteq U$ имеет место ФЗ $X \cup Z \to Y \cup W$.
6. Псевдотранзитивность. Если $X \subseteq U, Y \subseteq U, W \subseteq U, Z \subseteq U$, и заданы ФЗ $X \to Y$ и ФЗ $Y \cup W \to Z$, то имеет место ФЗ $X \cup W \to Z$.
7. Объединение. Если $X \subseteq U, Y \subseteq U, Z \subseteq U$, и заданы ФЗ $X \to Y$ и ФЗ $X \to Z$, то имеет место ФЗ $X \to Y \cup Z$.
8. Декомпозиция. Если $X \subseteq U, Y \subseteq U, Z \subseteq U$ и $Z \subseteq Y$, и задана ФЗ $X \to Y$, то имеет мето ФЗ $X \to Z$.

После того, как выделены все функциональные зависимости, следует проверить их согласованность с данными исходного отношения.

3.3.3. Нормальные формы

Нормальная форма — требование, предъявляемое к структуре таблиц баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Процесс проектирования БД с использованием метода нормальных форм (НФ) является итерационным и заключается в последовательном переводе отношения из первой НФ в НФ более высокого порядка по определенным правилам. Каждая следующая НФ ограничивается определенным типом функциональных зависимостей и устранением соответствующих аномалий при выполнении операций над отношениями БД, а также сохранении свойств предшествующих НФ.

Аномалией называется такая ситуация в таблице БД, которая приводит к противоречию в БД (или существенно усложняет её обработку). Причиной

является излишнее дублирование данных в таблице, которое вызывается наличием функциональных зависимостей от неключевых атрибутов.

Типы аномалий:

- аномалии-модификации (проявляются в том, что изменение одних данных может повлечь просмотр всей таблицы и изменение ряда записей таблицы);
- аномалии-удаления (при удалении какого либо кортежа из таблицы может пропасть информация, которая не связана на прямую с удаляемой записью);
- аномалии-добавления (информацию в таблицу нельзя поместить, пока она неполная или вставка записи требует дополнительного просмотра таблицы).

Метод нормальных форм состоит в сборе информации о объектах задачи в рамках одного отношения и последующей декомпозиции этого отношения на несколько взаимосвязанных отношений на основе процедур нормализации отношений.

Цель нормализации: исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении кортежей.

Первая нормальная форма: отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Вторая нормальная форма: отношение находится во 2НФ, если оно находится в 1НФ и каждый неключевой атрибут неприводимо зависит от первичного ключа. (Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.)

Третья нормальная форма: отношение находится в 3НФ, когда находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа. Т.е. второе правило требует выносить все неключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Примеры.

Определите, в какой форме находится отношение. Если не в третьей, что нужно сделать, чтобы привести к третьей.

1.

Модель	Салон	Телефон
BMW	Автомир	(903) 888-33-99
Audi	Автомир	(903) 888-33-99
Nissan	Джепен-Авто	(916) 999-44-88

2.

Модель	Фирма	Цена	Скидка
M5	BMW	5500000	5%
X5M	BMW	6000000	5%

M1	BMW	2500000	5%
GT-R	Nissan	5000000	10%

3.

Модель	Год выпуска	Фирма	Цена	Скидка
M5	2016	BMW	5500000	5%
M5	2017	BMW	6500000	5%
X5M	2017	BMW	6000000	5%
M1	2017	BMW	2500000	5%
GT-R	2017	Nissan	5000000	10%

4.

Модель	Салон	Телефон
BMW, Audi	Автомир	(903) 888-33-99
Audi	Автопродажа	(495) 777-22-77
Nissan, Mazda	Джепен-Авто	(916) 999-44-88

На практике 3НФ в большинстве случаев является вполне достаточной и приведением к ней процесс проектирования базы данных заканчивается. Однако, 3НФ упрощает решение проблем контроля избыточности данных, контроля за операциями модификации данных только если в отношениях отсутствуют какие-либо другие ФЗ, в частности обратные ФЗ неключевого атрибута на один из атрибутов составного первичного ключа или многозначные ФЗ. Иначе упомянутые проблемы остаются неразрешенными. Для устранения проблем, связанных с существованием обратных ФЗ неключевых атрибутов на часть составного ключа, была предложена **усиленная 3НФ или НФ Бойса-Кодда** (НФБК).

Отношение находится в НФБК, если оно находится в 3НФ, и в нем отсутствовали зависимости ключевых атрибутов от неключевых атрибутов. Схема отношения в НФБК обладает теми же достоинствами, что и схема в 3НФ, но устраняет некоторые дополнительные аномалии, не устраняемые 3НФ. Например, в отношение (Город, Адрес, Почтовый_индекс), находящееся в 3НФ, невозможно записать кортеж для города с известным почтовым индексом, если неизвестен адрес с этим почтовым индексом. Данное отношение не находится в НФБК, так как имеет место ФЗ Почтовый_индекс -> Город, а атрибут почтовый_индекс не является ключом этого отношения. В отличие от 3НФ, исходные отношения не всегда могут быть приведены в НФБК.

Четвертая нормальная форма: отношение находится в 4НФ в том и только том случае, когда существует многозначная зависимость A => B, а все остальные атрибуты функционально зависят от A. Или: В отношении R (A, B, C) существует **многозначная зависимость** R.A => R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

Пятая нормальная форма: отношение находится в 5НФ (или нормальной форме проекции-соединения), в том и только в том случае, если когда любая зависимость соединения в отношении следует из существования некоторого возможного ключа в этом отношении (отсутствуют сложные зависимые соединения между атрибутами).

3.4. Список рекомендуемой литературы

1. Шнырев С.Л. **Шнырев** С.Л. / Базы данных: Учебное пособие / Москва / МИФИ / 2011 http://www.iqlib.ru/
2. Вайнейкис Л.А. Базы данных : Учебное пособие / - М. : МГТУ ГА. - 92с.
3. Нестеров С.А. Базы данных. Учебник. -2013. - http://mexalib.com/view/61873
4. Голицына О.Л. Максимов Н.В. Попов И.И. Базы данных. / Форум, Инфа-М. 2009, - http://mexalib.com/view/166338

3.5. Контрольные вопросы

1. Назовите подходы к проектированию баз данных
2. В чем состоит избыточное и неизбыточное дублирование данных?
3. Что такое аномалия? Назовите основные виды и их характеристики.
4. Как формируется исходное отношение при проектировании базы данных?
5. Назовите виды зависимостей между атрибутами отношений.
6. Приведите примеры отношений с зависимыми атрибутами.
7. Охарактеризуйте нормальные формы.
8. Дайте определение первой нормальной формы.
9. Дайте определение второй нормальной формы.
10. Дайте определение третьей нормальной формы.
11. Дайте определение третьей усиленной нормальной формы.
12. Дайте определение четвертой нормальной формы.
13. Дайте определение пятой нормальной формы.
14. Сформулируйте основное правила по созданию таблиц сущностей.
15. Поясните назначение первичного ключа, внешнего ключа, индекса.

4. Практическое занятие №2

Использование модели «сущность-связь» при проектировании баз данных

4.1. Цель занятия

Целью практического занятия является приобретение навыков:

- структурного анализа с использованием модели «сущность-связь»;
- построения диаграмм ER-типа и ER-экземпляра;
- выявления классов принадлежности сущностей и их использования при проектировании.

4.2. Задание на выполнение работы

Разработать структуру базы данных авиаперевозок в части управления экипажами воздушных судов, представленной в п.3.2, используя метод «сущность-связь». Сравнить получившуюся структуру с полученной при проектировании методом нормальных форм. Сформулировать принципиальные различия в методах, их преимущества и недостатки.

4.3. Краткие теоретические сведения

4.3.1. Основные понятия метода

Метод сущность-связь (также называемый методом ER-диаграмм) основывается на использовании диаграмм, которые называют диаграммами ER-типа и диаграммами ER-экземпляров.

Основными понятиями метода сущность – связь являются следующие:

- сущность - объект, информация о котором хранится в базе данных; в качестве сущностей обычно используются существительные;
- атрибут сущности - свойство сущности, которое представляет собой логически неделимый элемент структуры информации, характеризующийся множеством атомарных значений (аналогично понятию «атрибут» в отношении);
- ключ сущности - атрибут или набор атрибутов, который используется для идентификации экземпляра сущности;
- связь между сущностями показывает зависимость между их атрибутами; названием связи обычно является глагол;
- степень связи характеризует связь между сущностями; степени могут быть четырех типов: 1:1, 1:M, M:1, M:M;
- класс принадлежности экземпляров сущности может быть двух видов: обязательный и необязательный (O и H); обязательным класс принадлежности сущности является в случае, когда все экземпляры данной сущности обязательно участвуют в данной связи, иначе класс принадлежности сущности является необязательным.

Для наглядности используются графические компоненты:

- диаграммы ER-экземпляров, пример представлен в таблице;
- диаграммы ER-типа (или ER-диаграмма), пример представлен на рисунке.

Диаграмма ER-экземпляров:

Преподаватель	***Ведет***	***Дисциплина***
Егорова А.А.		Базы данных
Иванов И.И.		Линейная алгебра
Козлов С.С.		Программирование
Кузнецов В.Л.		Математическое моделирование
Петров П.П.		Дискретная математика
		Математический анализ

Диаграмма ER-типа для диаграммы, представленной в таблице (1:M, H:O):

4.3.2. Этапы проектирования

Процесс проектирования базы данных является итерационным, допускающим в к предыдущим этапам и пересмотр ранее принятых решений и включает следующие этапы:

- Выделение сущностей и связей между ними,
- Построение диаграмм ER-типа,
- Формирование набора предварительных отношений,
- Добавление неключевых атрибутов в отношения,
- Приведение предварительных отношений к нормальной форме,
- Пересмотр ER-диаграмм.

4.3.3. Правила формирования отношений

Правила формирования отношений основываются на учете:

- степени связи между сущностями;
- класс принадлежности (КП) экземпляров сущности: обязательный (О) или необязательный (Н).

Формирование отношений для связи 1:1.

Правило 1.

Если класс принадлежности сущностей О:О, то формируется одно отношение. Первичным ключом может быть ключ любой из сущностей.

Правило 2.

Если класс принадлежности сущностей О:Н, то под каждую из сущностей формируется отношение с первичными ключами, являющимися ключами

соответствующих сущностей. Далее к отношению с обязательным КП добавляется ключ сущности с необязательным КП.

Правило 3.

Если класс принадлежности сущностей H:H, то необходимо использовать 3 отношения. Два отношения соответствуют связываемым сущностям, а третье отношение связывает первые два, и его первичный ключ объединяет ключевые атрибуты связываемых сущностей.

Формирование отношений для связи 1:M.

Правило 4.

Если класс принадлежности M-связной сущности O, то формируются два отношения. Ключ односвязной сущности добавляется как атрибут (внешний ключ) в отношение, соответствующее M-связной сущности.

Правило 5.

Если класс принадлежности M-связной сущности H, то необходимо формирование трех сущностей. Два отношения соответствуют связываемым сущностям, а третье отношение связывает первые два, и его первичный ключ объединяет ключевые атрибуты связываемых сущностей.

Формирование отношений для связи M:M.

Правило 6.

Независимо от класса принадлежности необходимо использовать 3 отношения. Два отношения соответствуют связываемым сущностям, а третье отношение связывает первые два, и его первичный ключ объединяет ключевые атрибуты связываемых сущностей.

4.4. Список рекомендуемой литературы

1. Шнырев С.Л. **Шнырев** С.Л. / Базы данных: Учебное пособие / Москва / МИФИ / 2011 http://www.iqlib.ru/
2. Вайнейкис Л.А. Базы данных : Учебное пособие / - М. : МГТУ ГА. - 92с.
3. Нестеров С.А. Базы данных. Учебник. -2013. - http://mexalib.com/view/61873
4. Голицына О.Л. Максимов Н.В. Попов И.И. Базы данных. / Форум, Инфа-М. 2009, - http://mexalib.com/view/166338

4.5. Контрольные вопросы

1. Перечислите основные понятия метода «сущность – связь».
2. Что такое ключ сущности?
3. Что такое диаграмма ER-экземпляра?
4. Что такое диаграмма ER-типа?

5. Что определяет степень связи между сущностями?
6. Что такое класс принадлежности и на что он влияет?
7. Как на диаграммах обозначаются степень связности и класс принадлежности?
8. Назовите этапы проектирования базы данных.
9. Сформулируйте правила формирования от ношений для связей 1:1, 1:M, M:M.

5. Практическое занятие №3

Автоматизация процесса разработки баз данных

5.1. Цель работы

Целью практического занятия является приобретение навыков:

- использования распространенных моделей и диаграмм графического представления, используемых при структурном анализе и проектировании;
- описания процессов функционирования информационных систем в нотации case-систем;
- декомпозиции при проектировании баз данных и разработки информационных систем.

5.2. Задание на выполнение работы

Сформировать функциональную схему информационной системы на основе исходных данных п.3.2 в методологии SADT.

Выполнить описание основной функции системы в нотации UML, используя диаграммы активности, состояний и прецедентов использования.

5.3. Краткие теоретические сведения

5.3.1.Основные определения

Графическое представление информационных систем при их ручной разработке весьма трудоемко, что послужило одной из причин появления программно-технологических средств, получивших название Case-средств и реализующих Case-технологию создания и сопровождения информационных систем.

Термин Case (Computer Aided Software Engineering) переводится как разработка программного обеспечения с помощью компьютера, но в настоящее время получил более широкий смысл, означающий автоматизацию разработки информационных систем.

CASE-средства – это программно-технологические средства, реализующие технологию создания и сопровождения информационных систем.

Case-система – набор средств определенного функционального назначения и выполненного в рамках единого программного продукта.

Case-технология – методология проектирования ИС плюс инструментальные средства, позволяющие моделировать предметную область на всех этапах разработки и сопровождения ИС.

5.3.2. Модели жизненного цикла информационных систем (программного обеспечения)

Каждая Case-система ориентирована на определенную модель жизненного цикла программного обеспечения (ЖЦ ПО).

Жизненный цикл программного обеспечения — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент завершения его эксплуатации.

Модель жизненного цикла программного обеспечения — структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта (на всех этапах ЖЦ ПО).

Основные этапы ЖЦ ПО:

- анализ,
- проектирование,
- реализация,
- внедрение,
- сопровождение.

Отдельно могут рассматриваться этапы определения требований, тестирования, внедрения версий и т.п.

Различают модели:

- каскадную (строгая последовательность реализации каждого этапа),
- с промежуточным контролем (допускает возврат с каждого этапа на любой предыдущий этап),
- спиральную (несколько раз проходит все этапы, так как основана на разработке прототипов систем).

5.3.3. Методология SADT

Методология SADT (Structured Analysis and Design Technique – методология структурного анализа и проектирования) представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели системы. Часть этой методологии, называемой IDEF0 (Icam DEFinition), принята во многих странах в качестве федерального стандарта на разработку программного обеспечения. В России IDEF0 рекомендована для использования Госстандартом РФ и активно применяется в отечественных госструктурах.

Данная методология позволяет:

- описывать любые системы, а не только информационные;
- создать описание системы и ее внешнего окружения до определения окончательных требований к ней. С помощью данной методологии можно постепенно выстраивать и анализировать систему даже тогда, когда трудно еще представить ее воплощение.

IDEF0 может применяться на ранних этапах создания широкого круга систем. В то же время она может быть использована для анализа функций существующих систем и выработки решений по их улучшению.

Основу методологии IDEF0 составляет графический язык описания процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Типы диаграмм IDEF0:

- контекстные;
- диаграммы декомпозиции;
- диаграммы дерева узлов;
- диаграммы только для экспозиции.

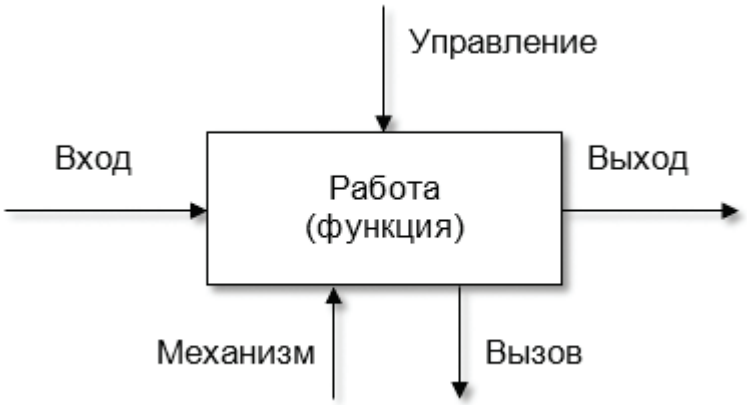
Контекстная диаграмма (диаграмма верхнего уровня) - вершина древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т. е. определяются функции, из которых состоит основная.

Далее функции делятся на подфункции и так до достижения требуемого уровня детализации исследуемой системы. Диаграммы, которые описывают каждый такой фрагмент системы, называются **диаграммами декомпозиции**. После выполнения декомпозиции эксперты предметной области указывают на соответствие реальных процессов созданным диаграммам. Найденные несоответствия устраняются, после чего проводят дальнейшую детализацию процессов.

Диаграмма дерева узлов показывает иерархическую зависимость функций (работ). Их может быть несколько, поскольку дерево можно построить на произвольную глубину и с произвольного узла.

Диаграммы для экспозиции строятся для иллюстрации отдельных фрагментов модели с целью отображения альтернативной точки зрения на происходящие в системе процессы.

Основные элементы модели представлены на рисунке:



Прямоугольник представляет собой работу (процесс, деятельность, функцию или задачу), которая имеет фиксированную цель и приводит к некоторому конечному результату. Имя работы должно выражать действие.

Взаимодействие работ между собой и внешним миром описывается в виде стрелок. В IDEF0 различают 5 видов стрелок:

- **Вход** – материал или информация, которые используются и преобразуются работой для получения результата (выхода). Вход отвечает на вопрос «Что подлежит обработке?». В качестве входа может быть и материальный объект, и нематериальный (например, запрос). Допускается, что работа может не иметь ни одной стрелки входа.
- **Управление** – регламентирующие и нормативные данные, которые руководят работой. Управление отвечает на вопрос «В соответствии с чем выполняется работа?». Управление выступает в качестве ограничения. В качестве управления могут быть правила, стандарты, нормативы, расценки и даже устные указания.
- **Выход** – материал или информация, которые представляют результат выполнения работы. Выход отвечает на вопрос «Что является результатом работы?». В качестве выхода может быть как материальный объект (деталь, документы…), так и нематериальный (выборка данных из БД, ответ на вопрос,…).
- **Механизм** – ресурсы, которые выполняют работу. Механизм отвечает на вопрос «Кто выполняет работу или посредством чего?». Механизм может быть одушевленным и неодушевленным (персонал предприятия, оборудование, программа).
- **Вызов** – стрелка указывает, что некоторая часть работы выполняется за пределами рассматриваемого блока.

Типы связей между работами (функциями):

- **Иерархическая связь** (связь «часть» – «целое») - связь между функцией и подфункциями, из которых она состоит.
- **Регламентирующая** (управляющая, подчиненная) связь отражает зависимость одной функции от другой, когда выход одной работы направляется

на управление другой. Функцию, из которой выходит управление, следует считать регламентирующей или управляющей, а в которую входит – подчиненной. Различают прямую связь по управлению, когда управление передается с вышестоящей работы на нижестоящую, и обратную связь по управлению, когда управление передается от нижестоящей к вышестоящей.

- **Функциональная** - связь, когда выход одной функции служит входными данными для следующей функции. Различают прямую связь по входу (выход передается с вышестоящей работы на нижестоящую), и обратную связь по входу (выход передается с нижестоящей к вышестоящей).
- **Потребительская** - связь, когда выход одной функции служит механизмом для следующей функции (одна функция потребляет ресурсы, вырабатываемые другой).
- **Логическая** связь наблюдается между логически однородными функциями (например, функции, выполняют одну и ту же работу, но разными способами или используют разные исходные данные/материалы).
- **Коллегиальная** - связь между функциями, алгоритм работы которых определяется одним и тем же управлением.
- **Ресурсная** = связь между функциями, использующими для своей работы одни и те же ресурсы. (Ресурсно-зависимые функции, как правило, не могут выполняться одновременно).
- **Информационная** - связь между функциями, использующими в качестве входных данных одну и ту же информацию.
- **ВременнАя** - связь между функциями, которые должны выполняться одновременно до или после другой функции.
- **Случайная** - связь, когда конкретная связь между функциями мала или полностью отсутствует.

При объединении функций в модули наиболее желательными являются первые пять видов связей. Функции, связанные последними пятью связями, лучше реализовывать в отдельных модулях.

Правила построения диаграмм IDEF0:

1. Перед построением модели необходимо определиться, какая модель (модели) системы будет построена: AS-IS, TO-BE или SHOULD-BE, и определить позицию, с «точки зрения» которой строится модель (место человека или объекта, на которое надо встать, чтобы увидеть систему в действии). Обычно выбирается одна точка зрения, наиболее полно охватывающая все нюансы работы системы, и при необходимости для некоторых диаграмм декомпозиции строятся диаграммы FEO, отображающие альтернативную точку зрения.

2. На контекстной диаграмме отображается один блок, показывающий назначение системы. Для него рекомендуется отображать по 2–4 стрелки, входящие и выходящие с каждой стороны.

3. Количество блоков на диаграммах декомпозиции рекомендуется 3–6. Если на диаграмме декомпозиции два блока, то она, как правило, не имеет смысла. При наличии большого количества блоков диаграмма становится перенасыщенной и трудно читаемой.

4. Блоки на диаграмме декомпозиции следует располагать слева направо и сверху вниз. Такое расположение позволяет более четко отразить логику и последовательность выполнения работ. При этом маршруты стрелок будут менее запутанными и иметь минимальное количество пересечений.

5. Отсутствие у функции одновременно стрелок управления и входа не допускается. Это означает, что запуск данной функции не контролируется и может произойти в любой произвольный момент времени либо вообще никогда.

6. У каждого блока должен быть как минимум один выход.

7. При построении диаграмм следует минимизировать число пересечений, петель и поворотов стрелок.

8. Обратные связи и итерации (циклические действия) могут быть изображены с помощью обратных дуг.

9. Каждый блок и каждая стрелка на диаграммах должны обязательно иметь имя. Допускается использовать ветвление (декомпозицию) или слияние (композицию) стрелок.

10. Все стрелки, входящие и выходящие из блока, при построении для него диаграммы декомпозиции должны быть отображены на ней. Имена стрелок, перенесенных на диаграмму декомпозиции, должны совпадать с именами, указанными на диаграмме верхнего уровня.

11. Если две стрелки проходят параллельно (начинаются из одной и той же грани одной работы и заканчиваются на одной и той же грани другой работы), то по возможности следует их объединить и называть единым термином.

12. Каждый блок на диаграммах должен иметь свой номер. Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Блок на диаграмме верхнего уровня обозначается 0, блоки на диаграммах второго уровня – цифрами от 1 до 9 (1, 2, …, 9), блоки на третьем уровне – двумя цифрами, первая из которых указывает на номер детализируемого блока с родительской диаграммы, а вторая номер блока по порядку на текущей диаграмме (11, 12, 25, 63) и т. д. Контекстная диаграмма имеет обозначение «А – 0», диаграмма декомпозиции первого уровня – «А0», диаграммы декомпозиции следующих уровней – состоят из буквы «А», за которой следует номер декомпозируемого блока (например, «А11», «А12», «А25», «А63»).

5.3.4. Нотация UML

UML (Unified Modeling Language) - единый язык моделирования, предназначенный для спецификации, визуализации, конструирования и документирования материалов программных систем, а также для моделирования бизнеса и других непрограммных систем (графический, унифицированный).

Типы диаграмм UML:

- Классов (class),
- Компонентов (component),
- Составных структур (composite structure),
- Объектов (object),
- Пакетов (package),
- Активности или деятельности (activity),
- Состояния или автомата (statechart),
- Прецедентов использования (use case),
- Следования (sequence),
- Сотрудничества или коммуникации (collaboration),
- Размещения или развертывания (deployment).

Рассмотрим некоторые из них.

Диаграмма активности:

Диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов (action), соединённых между собой потоками, которые идут от выходов одного узла к входам другого.

Пример диаграммы активности показан на рисунке:

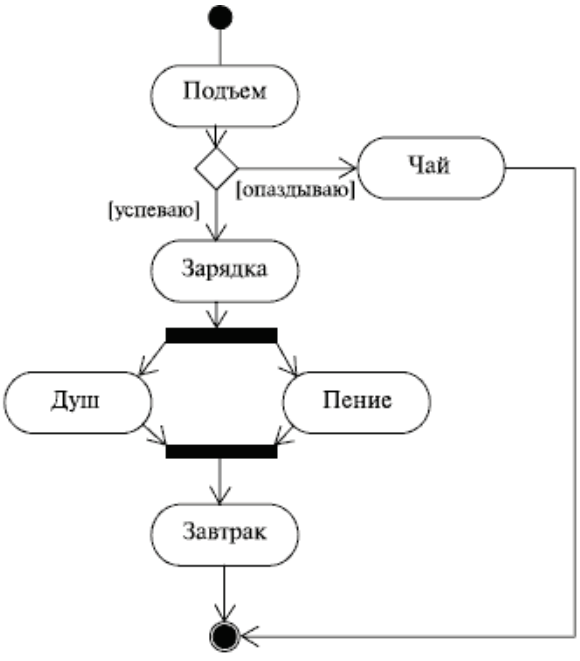


Диаграмма состояний - спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события.

Пример:

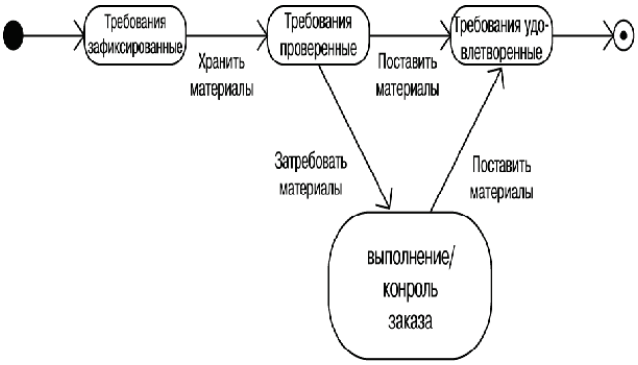


Диаграмма прецедентов использования описывают функциональность ИС, видимую пользователями системы.

Пример:

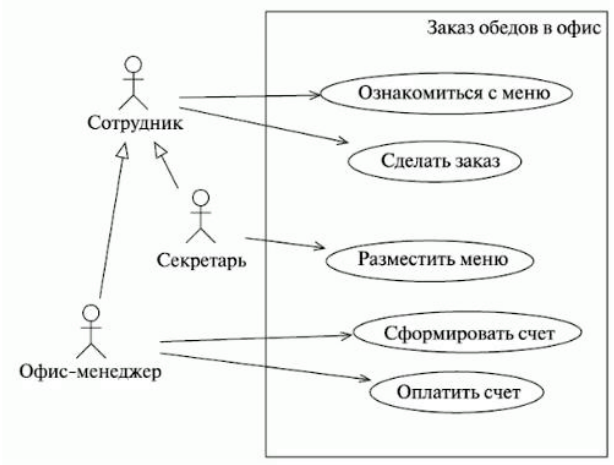


Диаграмма классов - статическая структурная диаграмма, описывающая структуру системы, демонстрирующая классы системы, их атрибуты, методы и зависимости между классами.

Пример:

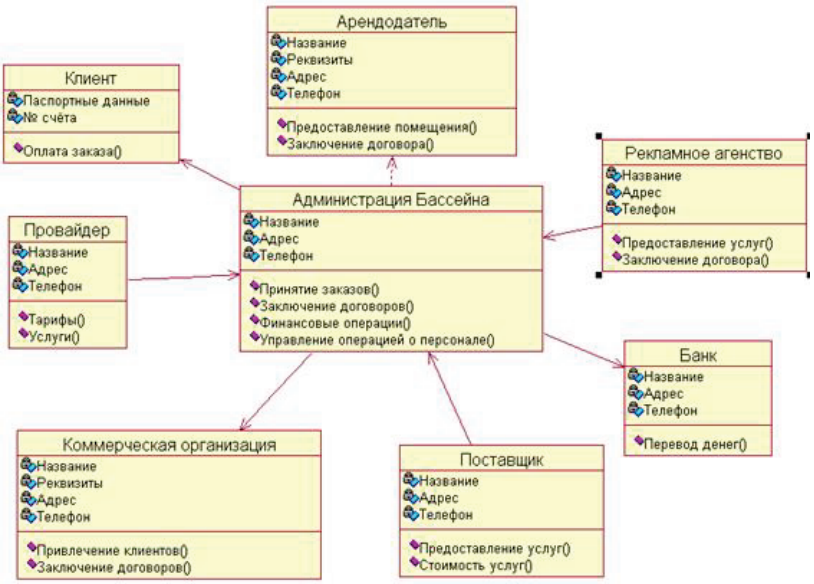
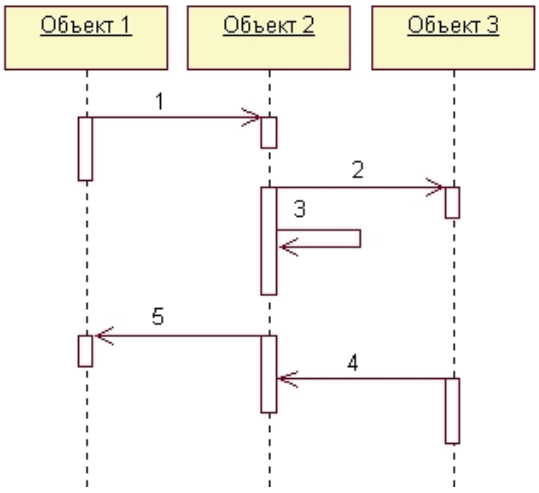


Диаграмма следования предназначена для моделирования взаимодействия объектов системы во времени, а также обмена сообщениями между ними.

Пример:



Инструментальные средства UML можно разделить на профессиональные (дорогие, ориентированные на коллективную работу) и полупрофессиональные (относительно недорогие или распространяющиеся условно-бесплатно, иногда широкого назначения, но в том числе поддерживающие нотацию). К первым относятся:

- Rational Rose,
- Platinum,
- Select Enterprise,
- Visual Modeler.

К средствам, широко доступным для использования, относятся:

- StarUML,
- NClass,
- Torgether UML Designer,
- Poseidon,
- MS Visio Professional.

5.4. Список рекомендуемой литературы

1. Джим Арлоу, Айла Нейштадт. UML 2 и унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. Символ плюс, 2007. http://forcoder.ru/uml
2. Фаулер М., Скотт К. UML. Руководство по унифицированному языку моделирования.http:/mexalib.com/view/23266
3. С.В. Маклаков "BPwin и Erwin. CASE-средства для разработки информационных систем". ДИАЛОГ-МИФИ

5.5. Контрольные вопросы

1. Что такое case-средство?
2. Что такое case-технология?
3. Что такое модель жизненного цикла программного обеспечения?
4. Охарактеризуйте спиральную модель жизненного цикла программного обеспечения.
5. Охарактеризуйте каскадную модель жизненного цикла программного обеспечения.
6. Перечислите распространенные модели и диаграммы графического представления, используемые при структурном анализе и проектировании.
7. Приведите пример диаграммы потоков данных.
8. Что представляет унифицированный язык моделирования UML?
9. Назовите типы диаграмм UML.
10. Для чего служит диаграмма прецедентов использования?
11. Для чего служит диаграмма классов?
12. Для чего служит диаграмма следования?
13. Назовите основные case-системы, поддерживающие UML.

6. Практическое занятие №4

Основные операторы языка SQL

6.1. Цель работы

Целью практического занятия является приобретение навыков:

- работы с основными операторами SQL;
- формирования баз данных в среде, поддерживающей SQL;
- использования различных типов, поддерживаемых SQL;
- создания доменов в SQL.

6.2. Задание на выполнение работы

Для структуры базы данных, разработанной в п.3 или п.4, напишите запросы на языке SQL для:

- создания базы данных,
- модификации базы банных и данных таблиц,
- удаления данных и таблиц.

Обоснуйте выбор вторичных ключей и доменов.

6.3. Краткие теоретические сведения

6.3.1. Структурированный язык запросов SQL

SQL (Structured Query Language — язык структурированных запросов) — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. Является информационно-логическим языком, а не языком программирования. Основан на реляционном исчислении.

Преимущества SQL:

- независимость от конкретной СУБД,
- наличие стандартов,
- полноценность как языка для управления данными.

Недостатки SQL:

- частичная нереляционность,
- отступления от стандартов,
- сложность работы с иерархическими структурами.

Основные части SQL:

- операторы определения данных (Data Definition Language, DDL);
- операторы манипуляции данными (Data Manipulation Language, DML);
- операторы определения доступа к данным (Data Control Language, DCL).

6.3.2. Основные операторы языка SQL

Рассмотрим минимальное подмножество языка SQL, основываясь на его реализации в стандартном интерфейсе ODBC фирмы Microsoft.

Команды языка DDL:

- CREATE TABLE – создание таблицы,
- ALTER TABLE – изменение таблицы,
- DROP TABLE – удаление таблицы,
- CREATE INDEX – создание индекса,
- DROP INDEX – удаление индекса,
- CREATE VIEW – создание представления,
- DROP VIEW – удаление представления.

Команды языка DCL:

- GRAND – назначение привилегий,
- REVOKE- удаление привилегий.

Команды языка DML:

- SELECT – выборка записей,
- INSERT – вставка новых записей,
- UPDATE – изменение записей,
- DELETE – удаление записей.

Рассмотрим форматы некоторых команд.

1.CREATE TABLE имя_табл. (имя_столб. тип_дан. [NULL | NOT NULL] [,...n])
2. ALTER TABLE имя_таблицы
{[ALTER COLUMN имя_столбца {новый_тип_данных [(точность[,масштаб])] [NULL | NOT NULL]}]
| ADD { [имя_столбца тип_данных]
| имя_столбца AS выражение } [,...n]
| DROP {COLUMN имя_столбца}[,...n] }
3.DROP TABLE имя_табл.
4.CREATE [UNIQUE] INDEX имя_индекса ON имя_табл.
(имя_столбца [ASC|DESC]
[, имя_столбца [ASC|DESC]…])
5. CREATE VIEW имя_представления [(имя_столбца [,имя_столбца…])]
AS оператор_SELECT.
6. DROP VIEW имя_представления
7. UPDATE имя_табл. SET имя_столбца = {выражение|NULL}
[, SET имя_столбца = {выражение|NULL}…]
[WHERE условие]

6.4. Список рекомендуемой литературы

1. Ульман Л. MySQL / **Ульман** Л. / MySQL / Москва / ДМК Пресс / 2012 http://www.iqlib.ru/
2. Вайнейкис Л.А. Базы данных : Учебное пособие / - М. : МГТУ ГА. - 92с.
3. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. SQL. Полное руководство / Вильямс, 2015 – 959 с. http://forcoder.ru/sql/sql-polnoe-rukovodstvo-1485
4. Душан Петкович. Microsoft SQL Server 2012. Руководство для начинающих / БХВ-Петербург, 2013. – 816 с. http://forcoder.ru/sql

6.5. Контрольные вопросы

1. Охарактеризуйте язык SQL.
2. Каково назначение языка SQL.
3. На чем основан язык SQL?
4. Назовите преимущества и недостатки языка SQL.
5. Перечислите составные части языка SQL.
6. Каково назначение оператора Select?
7. Поясните форму записи операторов SQL.
8. Что такое представление?
9. Что такое курсор в SQL?
10. Перечислите типы данных в SQL.
11. Как определить домен в SQL?

12. Перечислите основные объекты структуры базы данных SQL.
13. Перечислите основные операторы SQL.

7. Практическое занятие №5

Формирование запросов к базам данных на языке SQL

7.1. Цель работы

Целью практического занятия является приобретение навыков:

- формирования запросов действия к базам данных на языке SQL;
- формирования запросов с помощью оператора Select;
- использования шаблонов при формировании запросов;
- использования нестандартных запросов.

7.2. Задание на выполнение работы

Для структуры базы данных, разработанной в п.3 или п.4 и созданной в п.6, напишите запросы на языке SQL для извлечения, получения выборки и проекции по различным поисковым признакам (поисковых запросов) с использованием оператора Select.

Все запросы должны иметь смысл с учетом ограничений, выходных документов и бизнес-правил. Необходимо использовать все предложения Select (можно в разных запросах), включая поиск по шаблону, результирующие функции и нетривиальные запросы.

Обоснуйте выбор разработанных запросов.

7.3. Краткие теоретические сведения

7.3.1. Оператор Select

Оператор Select наиболее важный из всех операторов SQL. Его функциональные возможности и его применение огромны. Основное назначение оператора – производить выборку и вычисления над данными одной или нескольких таблиц. Результатом выполнения оператора является таблица.

Формат оператора:

SELECT [ALL | DISTINCT] {*|[имя_столбца [AS новое_имя]]} [,...n] FROM имя_таблицы [[AS] псевдоним] [,...n] [WHERE <условие_поиска>]
[GROUP BY имя_столбца [,...n]]
[HAVING <критерии выбора групп>]
[ORDER BY имя_столбца [,...n]]

Последовательность обработки элементов оператора Select:

- FROM – определяются имена используемых таблиц;
- WHERE – выполняется фильтрация строк объекта в соответствии с заданными условиями;

- GROUP BY – образуются группы строк , имеющих одно и то же значение в указанном столбце;
- HAVING – фильтруются группы строк объекта в соответствии с указанным условием;
- SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;
- ORDER BY – определяется упорядоченность результатов выполнения операторов (ASC – по возрастанию, DESC – по убыванию).

7.3.2. Типы условий поиска

Можно выделить следующие типы условий поиска:

- Сравнение: сравниваются результаты вычисления одного выражения с результатами вычисления другого (=, <>, >=, <=, <, >).
- Диапазон: проверяется, попадает ли результат вычисления выражения в заданный диапазон значений (BETWEEN).
- Принадлежность множеству: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений (IN).
- Соответствие шаблону: проверяется, отвечает ли некоторое строковое значение заданному шаблону:
  - ✓ Символ % – вместо этого символа может быть подставлено любое количество произвольных символов.
  - ✓ Символ _ заменяет один символ строки.
  - ✓ [] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.
  - ✓ [^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.
- Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

Для формирования нетривиальных запросов используются специальные функции. Например, функция Year(Сделка.Дата) из атрибута Дата отношения Сделка выбирает только год, а функция Month(Сделка.Дата) – только месяц. А функция Left(Имя,1) из атрибута Имя берет один символ справа.

7.3.3. Результирующие функции

При формировании запросов можно использовать результирующие функции. Рассмотрим основные.

Count (Выражение) - определяет количество записей в выходном наборе SQL-запроса;

Min/Max (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;

Avg (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом

записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.

Sum (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.

7.4. Список рекомендуемой литературы

5. Ульман Л. MySQL / **Ульман** Л. / MySQL / Москва / ДМК Пресс / 2012 http://www.iqlib.ru/
6. Вайнейкис Л.А. Базы данных : Учебное пособие / - М. : МГТУ ГА. - 92с.
7. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. SQL. Полное руководство / Вильямс, 2015 – 959 с. http://forcoder.ru/sql/sql-polnoe-rukovodstvo-1485
8. Душан Петкович. Microsoft SQL Server 2012. Руководство для начинающих / БХВ-Петербург, 2013. – 816 с. http://forcoder.ru/sql

7.5. Контрольные вопросы

1. Какие бываю запросы действия в SQL?
2. Что такое запросы модификации? Назовите их типы.
3. Поясните формат оператора Select.
4. Определите последовательность обработки элементов в операторе Select.
5. Каковы типы условий поиска в операторе Select? (Предложение Where)/
6. Что такое шаблон в SQL? Для чего он используется? Приведите примеры шаблонов.
7. Как упорядочить данные представления?
8. Поясните использование значения Null при заполнении базы данных и поиске.
9. Поясните использование итоговых функций при составлении запросов.
10. Приведите примеры нетривиальных запросов. В каких случаях они могут использоваться?

8. Практическое занятие №6

Формирование триггеров и курсоров средствами языка SQL

8.1. Цель работы

Целью практического занятия является приобретение навыков:

- использования курсоров при работе с базами данных на языке SQL;
- формирования хранимых процедур на языке SQL;

– формирования триггеров разных типов на языке SQL.

8.2. Задание на выполнение работы

Для структуры базы данных, разработанной в п.3 или п.4 и созданной в п.6, напишите запросы на языке SQL для:

– работы с курсором,
– формирования и использования хранимых процедур,
– формирования и использования триггеров.

При проектировании триггеров исходить из соображений поддержки целостности базы данных.

8.3. Краткие теоретические сведения

8.3.1. Целостность данных

Целостность данных - это свойство базы данных, означающее, что она содержит полную, непротиворечивую и адекватно отражающую предметную область информацию.

Различают целостность:

– физическую (означает наличие физического доступа к данным и что данные не утрачены);
– логическую (означает отсутствие логических ошибок в базе данных, к которым относятся нарушение структуры данных, ее объектов, удаление или изменение установленных связей между объектами и т.п.).

Целостное состояние задается с помощью ограничений.

Типы ограничений:

– ограничения значений атрибутов отношений,
– структурные ограничения на кортежи отношений.

Для введения ограничений используются, в том числе, хранимые курсоры, процедуры, триггеры.

8.3.2. Курсор в SQL

Курсор в SQL - это область в памяти базы данных, которая предназначена для хранения последнего оператора SQL. Если текущий оператор – запрос к базе данных, в памяти сохраняется и строка данных запроса, называемая текущим значением, или текущей строкой курсора. Указанная область в памяти поименована и доступна для прикладных программ.

Рассмотрим операции управления курсором:

DECLARE – создание или объявление курсора;
OPEN – открытие курсора, т.е. наполнение его данными;
FETCH – выборка из курсора и изменение строк данных с помощью курсора;
CLOSE – закрытие курсора;
DEALLOCATE – освобождение курсора, т.е. удаление курсора как объекта.

Формат команды для создания курсора:
DECLARE имя_курсора [INSENSITIVE][SCROLL]
CURSOR FOR SELECT_оператор [FOR { READ_ONLY | UPDATE
[OF имя_столбца[,...n]]}]

8.3.3. Хранимая процедура в SQL

Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде.

Типы хранимых процедур:

- системные,
- пользовательские,
- временные.

Формат команды для создания и изменения хранимой процедуры:

{CREATE | ALTER } PROC[EDURE] имя_процедуры [;номер] [{@имя_параметра тип_данных } [VARYING] [=default][OUTPUT]][,...n] [WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION }]
[FOR REPLICATION] AS sql_оператор [...n]

Выполнение хранимой процедуры:

[[EXEC [UTE] имя_процедуры [;номер] [[@имя_параметра=]{значение | @имя_переменной} [OUTPUT]|[DEFAULT]][,...n]

8.3.4. Триггер в SQL

Триггер – это специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблицах, с которыми триггеры связаны. Каждый триггер привязывается к конкретной таблице.

Создание триггеров:

CREATE TRIGGER имя_триггера
BEFORE | AFTER <триггерное_событие> ON <имя_таблицы>
[REFERENCING <список_старых_или_новых_псевдонимов>] [FOR EACH { ROW | STATEMENT}] [WHEN(условие_триггера)] <тело_триггера>

Параметры, определяющие поведение триггеров:

- AFTER (после события),
- INSTEAD OF (вместо события).

8.4. Список рекомендуемой литературы

1. Ульман Л. MySQL / **Ульман** Л. / MySQL / Москва / ДМК Пресс / 2012 http://www.iqlib.ru/
2. Вайнейкис Л.А. Базы данных : Учебное пособие / - М. : МГТУ ГА. - 92с.
3. Егорова А.А. Пособие к выполнению лабораторных работ по дисциплине «Базы данных». М.: МГТУ ГА– 38 с.

4. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. SQL. Полное руководство / Вильямс, 2015 – 959 с. http://forcoder.ru/sql/sql-polnoe-rukovodstvo-1485

8.5. Контрольные вопросы

1. Что такое целостность данных?
2. Поясните типы целостности.
3. Поясните использование в SQL курсоров.
4. Назовите категории курсоров.
5. Поясните операции управления курсором.
6. Поясните выборку данных из курсора.
7. Поясните использование в SQL процедур (хранимых процедур).
8. Какие бывают типы хранимых процедур? Охарактеризуйте каждый тип.
9. Поясните операции создания и выполнения хранимых процедур.
10. Определите параметры хранимых процедур.
11. Поясните использование в SQL триггеров.
12. Поясните операции создания и удаления триггера.
13. Поясните параметры, определяющие поведение триггера.
14. Назовите типы триггеров.
15. Поясните назначение и содержание таблиц Inserted и Deleted.

9. Заключение

Качество проектирования информационной системы в целом и базы данных в частности оказывает огромное влияние на успешность системы на протяжении всего жизненного цикла и обеспечивает минимальные доработки в будущем. Кроме того, производительность - главный фактор, определяющий эффективность компьютерной системы, а проектирование - основа хорошей производительности. Если база данных изначально спроектирована плохо, то приложения едва ли смогут работать эффективно. Проектирование повышает вероятность того, что система будет удовлетворять заданным требованиям с учетом ограничений. Хорошее проектирование существенно облегчает сопровождение приложений, в том числе внесение в них изменений. Поскольку построение идеальной системы, полностью соответствующей реальному объекту, невозможно, то задача проектировщика - максимально эффективно выполнить свою работу в рамках этих ограничений и указать, где можно пойти на компромисс. Поэтому и данное пособие направлено, в первую очередь на выработку навыков, связанных с проектированием баз данных. Выполнение каждого задания на практическом занятии должно быть отражено в пояснительной записке по курсовой работе, встроено в логику всей работы и пояснено во время защиты.