ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

А.А. Егорова

БАЗЫ ДАННЫХ

ЧАСТЬ I.
ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Учебно-методическое пособие
по проведению практических занятий

для студентов III курса
направления 09.03.01
очной формы обучения

Москва
ИД Академии Жуковского
2020

УДК 004.65
ББК 6Ф7.3
Е30

Рецензент:
Феоктистова О.Г. – д-р техн. наук, доцент

Егорова А.А.

Е30 Базы данных. Часть I. Проектирование баз данных [Текст] : учебно-методическое пособие по проведению практических занятий / А.А. Егорова. – М.: ИД Академии Жуковского, 2020. – 52 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Базы данных» по учебному плану для студентов III курса направления 09.03.01 очной формы обучения.

Рассмотрено и одобрено на заседаниях кафедры 29.08.2020 г. и методического совета 29.08.2020 г.

УДК 004.65
ББК 6Ф7.3

В авторской редакции

Подписано в печать 10.12.2020 г.
Формат 60x84/16 Печ. л. 3,25 Усл. печ. л. 3,02
Заказ № 710/1008-УМП17 Тираж 90 экз.

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

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68
E-mail: zakaz@itsbook.ru

ПРЕДИСЛОВИЕ

Настоящее пособие содержит справочный материал, необходимый при выполнении заданий на практических занятиях по дисциплине «Базы данных», проводимых у студентов III курса направления подготовки 09.03.01 «Информатика и вычислительная техника» в первом семестре изучения дисциплины, 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.

Справочный материал, содержащийся в пособии, не претендует на полноту, а касается только минимально необходимых в рамках данной дисциплины аспектов, которые могут быть полезны, в том числе и для 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 «Информатика и вычислительная техника».

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 часть управления предприятием, компании инвестируют значительные средства в разработку компьютеризированных системы для эффективного решения этих задач. Один из способов повышения эффективности обработки данных — организовать их оптимальное 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 поддерживается (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 выполнении курсовой работы.

Пособие необходимо студентам в течение всего года обучения по дисциплине «Базы данных».

2. Организационно-методические рекомендации

В соответствии с учебным планом подготовки студентов по направлению 09.03.01 «Информатика и вычислительная техника» (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 и операционными системами;
- 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 – 24 часа (12 практических занятий по 2 часа).

Задачами изучения дисциплины «Базы данных» являются:

- изучение теории баз данных;
- изучение приемов проектирования баз данных;
- получение навыков работы с системами управления базами данных;
- получение навыков, инструментов и методов верификации данных.

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»

- ИД-1$_{ОПК-8}$ - Основные языки программирования и работы с базами данных, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий.

- ИД-2$_{ОПК-8}$ - Программирования, отладки и тестирования прототипов программно-технических комплексов задач.

В результате изучения дисциплины «Базы данных» студент должен:

по компетенции ИД-1$_{ОПК-8}$:

знать:

ОПК-8.1.1 - принципы проектирования структур баз данных;

ОПК-8.1.2 - универсальный механизм обмена данных, механизм распределенных информационных баз;

уметь:

ОПК-8.2.1 - разрабатывать и верифицировать структуру баз данных;

ОПК-8.2.2 - использовать эффективные методы обработки, формализации и структурирования и хранения данных;

ОПК-8.2.3 - разрабатывать приложения баз данных;

по компетенции ИД-2$_{ОПК-8}$:

знать:

ОПК-8.1.4 - принципы и современные технологии разработки приложений баз данных

иметь навыки:

ОПК-8.3.1 - разрабатывать модели вычислительных систем, включая модели баз данных.

2.2. Перечень тем практических занятий

ПЗ - 1. Операции реляционной алгебры.

ПЗ - 2. Проектирование реляционных БД.

ПЗ - 3. Проектирование баз данных методом нормальных форм.
ПЗ - 4. Оптимизация баз данных.
ПЗ – 5. Запросы действия в SQL.
ПЗ – 6. Процедуры в SQL.
ПЗ – 7. Выборка данных из базы в SQL.
ПЗ – 8. Подзапросы в SQL.
ПЗ – 9. Логическая оптимизация запросов.
ПЗ – 10. Модели структурного проектирования.
ПЗ – 11. Проектирование баз данных с использованием языка UML.
ПЗ – 12. Архитектура TOGAF.

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.

Большинство заданий, выполняемых в процессе практических занятия, являются подготовительными для выполнения лабораторных работ и отчасти курсовой работы и при определенной доработке и оформлении могут стать частью пояснительной записки.

3. Практическое занятие №1

Операции реляционной алгебры

3.1. Цель занятия

Целью практического занятия является изучение:

- операторов реляционной алгебры:
  - ✓ Базовых операций;
  - ✓ Операций Кодда;
  - ✓ Операций Дейта;
- реляционного исчисления.

3.2. Задание на практическое занятие

1. Выполнить все возможные операции Кодда и Дейта для отношений, приведенных в Приложении 1.

2. Сформулировать запросы реляционного исчисления для тех же отношений, аналогичных операциям Кодда.

3.3. Краткие теоретические сведения

3.3.1. Реляционная алгебра

Реляционная алгебра – это замкнутая система операций над отношениями в реляционной модели данных. Операции реляционной алгебры также называют реляционными операциями. Реляционная алгебра является основанием некоторых языков запросов (в частности

ISBL – Information System Base Language), однако современные языки построены на реляционном исчислении.

Отношение обычно имеет графическую интерпретацию в виде таблицы, столбцы которой соответствуют атрибутам (которые определяются в «шапке» таблицы), а строки — кортежам, в «ячейках» находятся значения атрибутов в кортежах.

Основу реляционной операции составляют восемь операций, предложенных Коддом еще в начале 1970-х годов, которые можно разделить на две группы операторов: базовые теоретико-множественные и специальные реляционные.

Рассмотрим их подробнее.

Базовые теоретико-множественные операции.

– Объединение (UNION)

Результатом объединения отношений A и B будет отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих или A, или B, или обоим отношениям.

Пример:

Персоны

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Персонажи

Имя	Возраст	Вес
Daffy	24	19
Donald	25	23
Scrooge	81	27

Результат объединения:

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80
Daffy	24	19
Donald	25	23
Scrooge	81	27

– Разность (MINUS)

Результатом разности отношений A и B будет отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B.

Пример:

Персоны

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Персонажи

Имя	Возраст	Вес
Daffy	24	19
George	29	70
Donald	25	23
Scrooge	81	27
Sally	28	64

Результат разности:

Имя	Возраст	Вес
Harry	34	80
Helena	54	54
Peter	34	80

– Пересечение (INTERSECT)

Результатом пересечения отношений A и B будет отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям A и B.

Пример:

Персоны

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Персонажи

Имя	Возраст	Вес
Daffy	24	19
George	29	70
Donald	25	23
Scrooge	81	27
Sally	28	64

Результат пересечения:

Имя	Возраст	Вес
George	29	70
Sally	28	64

– Произведение (TIMES)

При выполнении прямого произведения двух отношений формируется отношение, кортежи которого являются конкатенацией (сцеплением) кортежей первого и второго операндов.

Пример:

Мультфильмы

Код_мульта	Название_мульта
0	The Simpsons
1	Family Guy
2	Duck Tales

Каналы

Код_канала	Название_канала
0	CTC
1	2x2

Результат произведения:

Код_мульта	Название_мульта	Код_канала	Название_канала
0	The Simpsons	0	CTC
0	The Simpsons	1	2x2
1	Family Guy	0	CTC
1	Family Guy	1	2x2
2	Duck Tales	0	CTC
2	Duck Tales	1	2x2

Специальные реляционные операции.

– Проекция

Проекция является операцией, при которой из отношения выделяются атрибуты только из указанных доменов, то есть из таблицы выбираются только нужные столбцы, при этом, если получится несколько одинаковых кортежей, то в результирующем отношении остается только по одному экземпляру подобного кортежа.

Пример:

Персоны

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Условие: Проекция Имя и Вес

Результат разности:

Имя	Вес
Harry	80
Sally	64
George	70
Helena	54
Peter	80

– Выборка (селекция, горизонтальная проекция)

(R WHERE f) отношения R по формуле f представляет собой новое отношение с таким же заголовком и телом, состоящим из таких кортежей отношения R, которые удовлетворяют истинности логического выражения, заданного формулой f. Для записи формулы используются операнды — имена атрибутов (или номера столбцов), константы, логические операции (AND — И, OR — ИЛИ, NOT — НЕ), операции сравнения и скобки.

Пример:

Персоны

Имя	Возраст	Вес
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

Условие: WHERE вес > 65

Результат разности:

Имя	Возраст	Вес
Harry	34	80
George	29	70
Peter	34	80

– Деление

Реляционное деление достаточно нетривиально описать, но на примере его смысл нагляден. В целом, из таблицы A берутся значения строк, для которых присутствуют все комбинации значений из таблицы B.

Пример:

Мультфильмы

Код_мульта	Название_мульта	Название_канала
0	The Simpsons	RenTV
0	The Simpsons	2x2
0	The Simpsons	CTC
1	Family Guy	RenTV
1	Family Guy	2x2
2	Duck Tales	CTC
2	Duck Tales	2x2

Каналы

Название_канала
RenTV
2x2

Результат деления:

Код_мульта	Название_мульта
0	The Simpsons
1	Family Guy

– Соединение (JOIN)

Операция соединения есть результат последовательного применения операций декартового произведения и выборки. Если в отношениях и имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать.

Пример:

Мульфильмы

Код_мульта	Название_мульта	Название_канала
0	The Simpsons	2x2
1	Family Guy	2x2
2	Duck Tales	RenTV

Каналы

Код_канала	Частота
RenTV	3,1415
2x2	783,25

Соединим их с выборкой по «Название_канала = Код_канала» после Произведения.

Первый этап - произведение:

Код_мульта	Название_мульта	Название_канала	Код_канала	Частота
0	The Simpsons	2x2	RenTV	3,1415
0	The Simpsons	2x2	2x2	783,25
1	Family Guy	2x2	RenTV	3,1415
1	Family Guy	2x2	2x2	783,25
2	Duck Tales	RenTV	RenTV	3,1415
2	Duck Tales	RenTV	2x2	783,25

Второй этап, выборка:

Код_мульта	Название_мульта	Название_канала	Код_канала	Частота
0	The Simpsons	2x2	2x2	783,25
1	Family Guy	2x2	2x2	783,25
2	Duck Tales	RenTV	RenTV	3,1415

По мнению Дейта этих восьми операций было недостаточно для построения реальной реляционной СУБД. Требуются расширения. Дейтом были предложены операции:

– Переименование

RENAME <исходное отношение> <старое имя атрибута> AS <новое имя атрибута>,

где <исходное отношение> задается именем отношения либо выражением реляционной алгебры (в последнем случае выражение заключается в круглые скобки).

Пример:

RENAME Город_П AS Город_Поставщика

– Расширение

EXTEND <исходное отношение> ADD <выражение> AS <новый атрибут>,

где к исходному отношению добавляется <новый атрибут>, подсчитываемый по правилам, заданным <выражением>. Исходное отношение может быть задано именем отношения и с помощью выражения реляционной алгебры, заключенного в круглые скобки.

Пример:

EXTEND (D JOIN PD) ADD (Вес*Количество) AS Общий_Вес

– Подведение итогов

SUMMARIZE <исх.отн.> BY (<список атрибутов>) ADD <выр.> AS <новый атрибут>, где исходное отношение задается или именем отношения или заключенным в круглые скобки выражением реляционной алгебры, <список атрибутов> представляет собой разделенными запятыми имена атрибутов исходного отношения A1, A2, …, AN, <выр.> - скалярное выражение, аналогичное выражению операции EXTEND, а <новый атрибут> - имя формируемого атрибута.

Пример. Пусть требуется вычислить количество поставок по каждому из поставщиков (В отношении PD Приложения 1):

SUMMARIZE PD BY (ID_P) ADD COUNT AS Количество_поставок

– Присвоение

<выражение_цель>:= <выражение_источник>,

где оба выражения задают совместимые (точнее, эквивалентные) по структуре отношения. Например, в левой части – имя отношения, а в правой – некоторое выражение реляционной алгебры.

Пример:

P:=P UNION {{<ID_P:’S6’>, <Имя: 'Борисов'>, <Статус:’40’>, <Город_П:'Томск'>}}

– Вставка

INSERT <выражение_источник> INTO <выражение_цель>

Пример:

INSERT (P WHERE Город_П='Москва') INTO Temp

– Обновление

UPDATE <выражение_цель> <список элементов>,

где <список элементов> - последовательность разделенных запятыми операций присвоения <атрибут>:=<скалярное выражение>. Результатом является отношение, полученное после присвоения соответствующих значений атрибутам отношения, заданного целевым выражением.

Пример:

UPDATE D WHERE Тип='мягкий' Город:=’Москва’.

– Удаление

DRLETE <выражение_цель>,

где <выражение_цель> - реляционное выражение, описывающее удаляемые кортежи.

Пример:

DELETE P WHERE Статус=10

– Сравнение (реляционное)

Реляционное сравнение используется для сравнения двух отношений:

<выражение1> <операнд сравнения> <выражени2>,

где оба выражения задают совместимые по структуре отношения,

а <операнд сравнения> – один из следующих операторов сравнения:

= (равно),

≠ (не равно),

≤ (собственное подмножество),

< (подмножество),

≥ (надмножество),

> (собственное надмножество).

Пример:

P [Город_П] = D [Город_Д]

Сравнивает, совпадают ли города поставщиков и города хранения деталей.

3.3.2. Реляционное исчисление

Реляционное исчисление — прикладная ветвь формальной теории, носящей название «исчисления предикатов первого порядка». Или реляционное исчисление — это адаптация логики первого порядка для написания запросов.

В основе исчисления лежит понятие переменной с определенной для неё областью допустимых значений и понятие правильно построенной формулы, опирающейся на переменные, предикаты и кванторы. Фактически для получения искомого результата требуется указать свойства указанного отношения без конкретизации способа его получения. Такой подход называется описательным (в отличии от реляционной алгебры, где подход предписывающий).

Пример: получить номера и города поставщиков, выпускающих деталь Р3. Запрос будет выглядеть следующим образом: «Получить атрибут ID_P и Город_П для таких поставщиков, для которых существует поставка в отношении PD с тем же значением атрибута ID_P и со значением Р3 атрибута ID_D».

В зависимости от того, что является областью определения переменной, различают:

- исчисление кортежей (областями определения переменных являются отношения базы данных, т.е. допустимым значением каждой переменной является кортеж некоторого отношения),
- исчисление доменов (областями определения переменных являются домены, на которых определены атрибуты отношений базы данных, т.е. допустимым значением каждой переменной является значение некоторого домена).

Рассмотрим исчисление кортежей подробнее.

Для определения кортежной переменной используется оператор RANGE. Например, для того, чтобы определить переменную ПОСТАВЩИК, областью определения которой является отношение Р, нужно употребить конструкцию

RANGE ПОСТАВЩИК IS P.

При использовании кортежных переменных в формулах можно ссылаться на значение атрибута переменной. Например, для того, чтобы сослаться на значение атрибута Имя переменной ПОСТАВЩИК, нужно употребить конструкцию ПОСТАВЩИК.Имя.

Правильно построенные формулы служат для выражения условий, накладываемых на кортежные переменные. Их основой являются простые сравнения. Более сложные варианты формул строятся с помощью логических связок NOT, AND, OR и IF ... THEN. Допускается построение формул с помощью кванторов (существования EXISTS и всеобщности FORALL).

Переменные, входящие в формулы, могут быть свободными или связанными. Все переменные, входящие в формулу, при построении которой не использовались кванторы, являются свободными. Т.е. если для какого-то набора значений свободных кортежных переменных при вычислении формулы получено значение true, то эти значения кортежных переменных могут входить в результирующее отношение.

Если же имя переменной использовано сразу после квантора при построении формулы вида EXISTS var (form) или FORALL var (form), то в этой формуле и во всех формулах, построенных с ее участием, var - это связанная переменная. Такая переменная не видна за пределами минимальной формулы, связавшей эту переменную. При вычислении значения такой формулы используется не одно значение связанной переменной, а вся ее область определения.

Пусть ПОСТАВЩИК1 и ПОСТАВЩИК2 - две кортежные переменные, определенные на отношении Р. Тогда, формула EXISTS ПОСТАВЩИК2 (ПОСТАВЩИК1.СТАТУС > ПОСТАВЩИК2. СТАТУС) для текущего кортежа переменной ПОСТАВЩИК1 принимает значение true в том и только в том случае, если во всем отношении Р найдется кортеж (связан-

ный с переменной ПОСТАВЩИК2) такой, что значение его атрибута СТАТУС удовлетворяет внутреннему условию сравнения. Формула FORALL ПОСТАВЩИК2 (ПОСТАВЩИК1.СТАТУС > ПОСТАВЩИК2.СТАТУС) для текущего кортежа переменной ПОСТАВЩИК1 принимает значение true в том и только в том случае, если для всех кортежей отношения P (связанных с переменной ПОСТАВЩИК2) значения атрибута СТАТУС удовлетворяют условию сравнения.

3.4. Список рекомендуемой литературы

1. Хомоненко А., Цыганков В., Мальцев М. Базы данных: учебник для высших учебных заведений. – М: КОРОНА принт, 2017 – 672 с.

2. Дейт, Дж.К. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.: ил.

3.5. контрольные вопросы

1. Назовите операции реляционной алгебры, предложенные Коддом.
2. Дайте определение операциям реляционной алгебры, предложенным Коддом.
3. Приведите графическую интерпретацию операциям реляционной алгебры, предложенным Коддом.
4. Назовите и охарактеризуйте дополнительные операции реляционной алгебры, предложенные Дейтом.
5. Охарактеризуйте варианты реляционного исчисления.
6. Что такое свободные и связанные переменные?
7. Какие кванторы используются в реляционном исчислении?

4. Практическое занятие №2

Проектирования реляционных БД

4.1. Цель занятия

Целью практического занятия является приобретение навыков:

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

4.2. Задание на практическое занятие

Разработать структуру базы данных авиаперевозок в части управления экипажами воздушных судов, приведенную 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 планировать 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.

4.3. Краткие теоретические сведения

4.3.1. 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.

4.3.2. 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.

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

4.5. Контрольные вопросы

1. Назовите подходы к проектированию баз данных
2. В чем состоит избыточное и неизбыточное дублирование данных?
3. Как формируется исходное отношение при проектировании базы данных?
4. Назовите виды зависимостей между атрибутами отношений.
5. Приведите примеры отношений с зависимыми атрибутами.
6. Сформулируйте основное правила по созданию таблиц сущностей.
7. Поясните назначение первичного ключа, внешнего ключа, индекса.

5. Практическое занятие №3

Проектирование баз данных методом нормальных форм

5.1. Цель занятия

Целью практического занятия является:

- изучение нормальных форм баз данных и принципов нормализации;
- приобретение навыков проектирования баз данных методом нормальных форм;
- приобретение навыков анализа баз данных на предмет соответствия нормальным формам (в первую очередь третьей).

5.2. Задание на выполнение работы

I. 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

II. Дано отношение «Проекты».

Номер_проекта	Код_сотрудника	Задание_сотрудника
001	05	1
001	05	2
001	05	3
004	02	1
004	02	2
004	03	1
004	03	2
004	05	1
004	05	2
007	06	1

Предполагается, что каждый сотрудник, участвующий в проекте, выполняет все задания по этому проекту. Привести отношение к четвертой нормальной форме.

III. Дано отношение «сотрудники-отделы-проекты». Пусть в этом отношении один сотрудник может работать в нескольких отделах и он может принимать участие в нескольких проектах. Но каждый проект выполняет только один сотрудник.

Номер_проекта	Код_сотрудника	Задание_сотрудника
01	РД	036
02	АД	004
03	УП	004
04	АД	019
05	ЛС	001
06	ЛС	004
07	УП	007
08	ВЦ	013
09	ВЦ	014
10	СЖ	013

Привести отношение к пятой нормальной форме.

5.3. Краткие теоретические сведения

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.

Третья усиленная нормальная форма: отношение находится в БКНФ (Бойеса-Кодда НФ), если оно находится в 3НФ и в нем отсутствуют зависимости ключей от неключевых атрибутов.

Четвертая нормальная форма: отношение находится в 4НФ в том и только том случае, когда существует многозначная зависимость A=> B, а все остальные атрибуты отношения функционально зависят от A.

Пятая нормальная форма: отношение находится в 5НФ (или нормальной форме проекции соединения) в том и только том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R.

Доменно-ключевая нормальная форма: переменная отношения находится в ДКНФ тогда и только тогда, когда каждое наложенное на неё ограничение является логическим следствием ограничений доменов и ограничений ключей, наложенных на данную переменную отношения.

Шестая нормальная форма: переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ.

5.4. Список рекомендуемой литературы

3. Шнырев С.Л. Шнырев С.Л. / Базы данных: Учебное пособие / Москва / МИФИ / 2011 http://www.iqlib.ru/

4. Хомоненко А., Цыганков В., Мальцев М. Базы данных: учебник для высших учебных заведений. – М: КОРОНА принт, 2017 – 672 с.

5. Дейт, Дж.К. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.: ил.

6. Вайнейкис Л.А. Базы данных: Учебное пособие / - М. : МГТУ ГА. - 92с.

5.5. Контрольные вопросы

1. Охарактеризуйте нормальные формы.
2. Дайте определение первой нормальной формы.
3. Дайте определение второй нормальной формы.
4. Дайте определение третьей нормальной формы.

5. Дайте определение третьей усиленной нормальной формы.
6. Дайте определение четвертой нормальной формы.
7. Дайте определение пятой нормальной формы.
8. Дайте определение доменно-ключевой нормальной формы
9. Дайте определение шестой нормальной формы.
10. Что такое аномалия? Назовите основные виды и их характеристики.

6. Практическое занятие №4

Оптимизация баз данных

6.1. Цель работы

Целью практического занятия является:

- изучение понятия и основных функциональных зависимостей;
- изучение понятия замыкания атрибутов;
- овладение навыком декомпозиции атрибутов и навыком доказательства оптимальности базы данных и обоснованности ее ключевых атрибутов.

6.2. Задание на выполнение работы

1. Изучить аксиомы функциональных зависимостей.
2. Выполнить вычисление замыкания множества атрибутов на примере Базы данных Приложения 2.
3. Для Базы данных Приложения 2 доказать, что она находится в третьей нормальной форме и является «хорошей».

6.3. Краткие теоретические сведения

6.3.1.Функциональные зависимости

Пусть R определение функциональной зависимости. $R = (A_1, A_2, \ldots, A_N)$ - универсальная схема отношений, т.е. схема состоит из всех атрибутов предметной области.

X и Y – некое подмножество этой схемы;

$X \rightarrow Y$ (X функционально определяет Y), если в любом экземпляре r не существует двух кортежей, совпадающих по атрибутам X и не совпадающих по атрибутам Y.

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 $X \rightarrow Y$ следует из F. Иначе $X, X \rightarrow X$.

2. Пополнение.

   Если $X \subseteq U, Y \subseteq U, Z \subseteq U$ и задана ФЗ $X \rightarrow Y$ из F, то имеет место ФЗ $X \cup Z \rightarrow Y \cup Z$.

3. Транзитивность.
   Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ и задана ФЗ $X \rightarrow Y, Y \rightarrow Z$ из F , то имеет место ФЗ $X \rightarrow Z$.

4. Расширение.
   Если $X \subseteq U$, $Y \subseteq U$ и задана ФЗ $X \rightarrow Y$, то $Z \subseteq U$ имеет место ФЗ $X \cup Z \rightarrow Y$.

5. Продолжение.
   Если $X \subseteq U$, $Y \subseteq U$, $W \subseteq U$, $Z \subseteq U$, и задана ФЗ $X \rightarrow Y$, то $\forall\ W \subseteq U$ имеет место ФЗ $X \cup Z \rightarrow Y \cup W$.

6. Псевдотранзитивность.
   Если $X \subseteq U$, $Y \subseteq U$, $W \subseteq U$, $Z \subseteq U$, и заданы ФЗ $X \rightarrow Y$ и ФЗ $Y \cup W \rightarrow Z$, то имеет место ФЗ $X \cup W \rightarrow Z$.

7. Объединение.
   Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$, и заданы ФЗ $X \rightarrow Y$ и ФЗ $X \rightarrow Z$, то имеет место ФЗ $X \rightarrow Y \cup Z$.

8. Декомпозиция.
   Если $X \subseteq U$, $Y \subseteq U$, $Z \subseteq U$ и $Z \subseteq Y$, и задана ФЗ $X \rightarrow Y$, то имеет место ФЗ $X \rightarrow Z$.

6.3.2. Замыкание множества функциональных зависимостей

Пусть R – универсальная схема отношения;

F – заданное множество функциональных зависимостей на этой схеме.

Замыканием F называется множество функциональных зависимостей, которые логически следуют из F.

Или иначе:

Пусть заданы отношение R, множество Z атрибутов этого отношения и некоторое множество ФЗ F, выполняемых для R. Тогда замыканием Z над F называется наибольшее множество F^+ таких атрибутов Y отношения R, что ФЗ $Z \rightarrow Y$ входит в $F^{+/}$

Функциональная зависимость логически следует из F, если её можно получить с помощью аксиом Армстронга. Обозначается F^+.

Аксиомы Армстронга (правила вывода) – это аксиомы рефлексивности, пополнения и транзитивности.

Пример построения замыкания:

Дано: R =(A, B, C) и F=($A \rightarrow B$, $B \rightarrow C$)

F + -?

1) Построение тривиальных функциональных зависимостей:

$A \rightarrow A$, $B \rightarrow B$, $C \rightarrow C$, $AB \rightarrow A$, $AB \rightarrow B$, $AC \rightarrow A$, $AC \rightarrow C$, $BC \rightarrow C$, $BC \rightarrow B$, $ABC \rightarrow A$, $ABC \rightarrow C$, $ABC \rightarrow B$, $ABC \rightarrow AC$, $ABC \rightarrow AB$, $ABC \rightarrow BC$, $ABC \rightarrow ABC$, $AB \rightarrow AB$, $AC \rightarrow AC$, $BC \rightarrow BC$.

2) Используем вторую аксиому Армстронга

$A \rightarrow AB$, $AC \rightarrow BC$, $B \rightarrow BC$

$AB \rightarrow AC$, $AC \rightarrow ABC$, $AB \rightarrow ABC$

3) Используем третью аксиому Армстронга

Т.к. $A \rightarrow B$, $B \rightarrow C$, то $A \rightarrow C$. Аналогично, $AB \rightarrow BC$, $A \rightarrow AC$, $A \rightarrow ABC$

6.3.3. Третья нормальная форма

Схема отношения R находится в 3НФ, если не существует ключа X, множества $Y \subseteq R$, и не первичного атрибута H ($H \notin Y$), для которых выполнялись бы следующие условия:

1. $X \rightarrow Y \in F^+$
2. $Y \rightarrow H \in F^+$
3. $Y \rightarrow X \notin F^+$

Из определения следует, что если можно подобрать X, Y и H для которых выполняются условия 1–3, то схема отношения не находится в 3НФ.

6.3.4. Пример доказательства, что схема отношения находится в 3НФ

Дано:

R = (фирма, адрес, товар, цена) = (A, B, C, D) - универсальная схема отношения;

F = ($A \rightarrow B$, $AC \rightarrow D$) – функциональные зависимости;

r = (AB, ACD) = (R_1, R_2) – схема БД.

Задача: Используя определение 3НФ доказать, что схема отношения находится в 3НФ (Доказать что R_1 , R_2 находятся в 3НФ).

Решение:

1 этап. R_1 = AB

а) X = A – ключ, так как $A \rightarrow B$ по условию задачи;

б) подберем Y для которого $X \rightarrow Y$, $Y \nrightarrow X$

Y = B ($(B)^+ = B$, $A \notin B$, $B \nrightarrow A$);

в) Здесь нельзя подобрать не первичный атрибут H, который $H \notin Y$ (единственный не первичный атрибут – B, но $B \in Y$).

Мы не сумели подобрать X, Y, H для которых были бы справедливы условия 1–3 из определения 3НФ, следовательно, по определению, R_1 находится в 3НФ.

2 этап. R_2 = ACD

1) X = AC – ключ

2) найдём Y для которого $X \rightarrow Y$, $Y \nrightarrow X$

Можно убедиться, что таким условиями удовлетворяют следующие Y:

а) A б) C

в) D г) AD

д) CD

3) Попытаемся подобрать не первичный атрибут H, для которого $H \notin Y$, $Y \rightarrow H$.

Единственным претендентом является D, так как AC – ключ, D – не первичный.

а) Y = A, но $Y \rightarrow H$, потому что $A \nrightarrow D$, $(A)^+ = AB$, $D \notin AB$.

б) Y = C, но $Y \rightarrow H$, потому что $C \nrightarrow D$, $(C)^+ = C$, $D \notin C$.

в-д) не проходят, так как здесь не первичный атрибут (D) $H \in Y$.

Методом логических исключений можно сделать вывод, что здесь нельзя подобрать X, Y, H для которых были бы справедливы условия 1–3 из определения 3НФ, следовательно, по определению, R_2 находится в 3НФ.

Поэтому вывод: R находится в 3НФ.

6.4. Список рекомендуемой литературы

1. Хомоненко А., Цыганков В., Мальцев М. Базы данных: учебник для высших учебных заведений. – М: КОРОНА принт, 2017 – 672 с.

2. Дейт, Дж.К. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.: ил.

3. В. Дунаев. «Базы данных. Язык SQL для студента». - BHV, 2012 г. – 370 с.

6.5. Контрольные вопросы

1. Что такое функциональная зависимость?
2. Назовите аксиомы Армстронга.
3. Сформулируйте дополнительные аксиомы функциональных зависимостей
4. Сформулируйте понятие «Замыкание множества функциональных зависимостей».
5. Алгоритм построения замыкания множества атрибутов.
6. Свойства «хорошей» базы данных.
7. Свойство соединения без потерь для схемы базы данных.
8. Свойство сохранения зависимостей для схемы базы данных.

7. Практическое занятие №5

Запросы действия в SQL

7.1. 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.

7.2. 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.

7.3. Краткие теоретические сведения

7.3.1. 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).

7.3.2. 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]
```

7.4. 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

7.5. 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. Перечислите типы данных в SQL.
10. Как определить домен в SQL?
11. Перечислите основные объекты структуры базы данных SQL.
12. Перечислите основные операторы SQL.

8. Практическое занятие №6

Процедуры в SQL

8.1. Цель работы

Целью практического занятия является приобретение навыков:

- формирования хранимых процедур на языке SQL;
- использования курсоров при работе с базами данных 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]]}] , где

INSENSITIVE определяет курсор, который создает временную копию данных для использования курсором;
SCROLL указывает, что доступны все параметры выборки.

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

9. Практическое занятие №7

Выборка данных из базы в SQL

9.1. 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.

9.2. Задание на выполнение работы

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

2. Постройте запросы для базы данных, представленной в Приложении 2.

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.

9.3. Краткие теоретические сведения

9.3.1. 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).
- ALL | DISTINCT - предикат DISTINCT следует применять в тех случаях, когда требуется отбросить блоки данных, содержащие дублирующие записи в выбранных полях. Значения для каждого из приведенных в инструкции SELECT полей должны быть уникальными, чтобы содержащая их запись смогла войти в выходной набор.

9.3.2. Типы условий поиска

Можно выделить следующие типы условий поиска:

- Сравнение: сравниваются результаты вычисления одного выражения с результатами вычисления другого (=, <>, >=, <=, <, >).

SELECT * FROM Student WHERE YearBorn >2000

- Диапазон: проверяется, попадает ли результат вычисления выражения в заданный диапазон значений (BETWEEN).

SELECT SurName, Rost FROM Student WHERE Rost Between 180 And 190

- Принадлежность множеству: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений (IN).

SELECT SurName, City FROM Student WHERE City in ("Москва", "Самара")

- Соответствие шаблону: проверяется, отвечает ли некоторое строковое значение заданному шаблону:
  - ✓ Символ % – вместо этого символа может быть подставлено любое количество произвольных символов.

SELECT SurName, PhoneNumber FROM Student WHERE PhoneNumber Like "7%"

(найти и вывести фамилии и телефоны студентов, чей номер телефона начинается с цифры 7).

✓ Символ _ заменяет один символ строки.

SELECT SurName, PhoneNumber FROM Student WHERE PhoneNumber Like "_0%"

(Вывести фамилии и телефоны студентов, в номере телефона которых вторая цифра 0).

✓ [] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.

SELECT SurName, PhoneNumber FROM Student WHERE SurName Like "К[а,о,у]чкин"

(Найти и вывести фамилию и телефон студента, фамилия которого то ли Качкин, то ли Кочкин, то ли Кучкин).

✓ [^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.

SELECT SurName, PhoneNumber FROM Student WHERE SurName Like "К[^у]чкин"

(Найти и вывести фамилию и телефон студента, фамилия которого известна приблизительно, но точно понятно, что не Кучкин).

– Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение).

SELECT SurName FROM Student WHERE PhoneNumber Is Null

(Вывести фамилии студентов, номер телефона которых неизвестен).

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 слева.

Например, вывести номер зачетки и фамилию с инициалами всех студентов из отношения ***Student:***

SELECT Zach_Number, SurName+""+ Left(Name,1) +"." +Left(FatherName,1) +"." AS FIO FROM Student

Или вывести информацию о студентах из отношения ***Student***, родившихся в декабре:

SELECT SurName, Name FROM Student WHERE Month (YearBorn)=12

9.3.3. 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.

9.4. 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
5. https://sql-academy.org/ru/trainer?sort=byId

9.5. 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?

10. Практическое занятие №8

Подзапросы в SQL

10.1. Цель работы

Целью практического занятия является приобретение навыков:

- использования подзапросов в SQL;
- формирования подзапросов, вложенных в подзапросы в SQL;
- формирования подзапросов в инструкциях INSERT, UPDATE и DELETE.

10.2. Задание на выполнение работы

1. Для структуры базы данных, разработанной в п.4, напишите запросы на языке SQL с вложенными запросами и вложенными подзапросами, которые должны быть использованы в инструкции SELECT, INSERT, UPDATE или DELETE.

2. Постройте запросы с подзапросами для базы данных Приложения 2.

10.3. Краткие теоретические сведения

Вложенный запрос — это запрос, который используется внутри инструкции SELECT, INSERT, UPDATE или DELETE или внутри другого вложенного запроса. Подзапрос может быть использован везде, где разрешены выражения.

SQL подзапрос — это запрос, вложенный в другой запрос.

Подзапрос может использоваться:

- В инструкции SELECT;
- В инструкции FROM;
- В условии WHERE.

Подзапрос обычно добавляется в условие WHERE оператора SELECT. Можно использовать операторы сравнения, такие как >, <, или =. IN, ANY или ALL. Подзапрос также называется внутренним запросом. Оператор, содержащий подзапрос, называется внешним или

родительским. Внутренний запрос выполняется перед родительским запросом, чтобы результаты его работы могли быть переданы внешнему.

Задачи подзапроса:

- сравнение выражения с результатом запроса;
- определения того, включено ли выражение в результаты запроса;
- проверка, выбирает ли запрос любые строки.

Синтаксис подзапроса:

(SELECT [DISTINCT] аргументы_подзапроса_для_отбора
FROM {имя_таблицы | имя_представления} { имя_таблицы | имя_представления } ...
[WHERE условия_поиска][GROUP BY выражение_объединения
[,выражение_объединения] ...]
[HAVING условия_поиска])

Рассмотрим пример. Пусть даны два отношения: Student и Rate (рейтинг). Составить запрос, определяющий всех студентов (фамилии), которые имеют рейтинг выше среднего.

Student

ID_Student	Surname
ЭВМ-001	Иванов
ЭВМ-002	Петров
ЭВМ-003	Сидоров
ЭВМ-004	Турков

Rate

ID_Student	Rate
ЭВМ-001	3,4
ЭВМ-002	4,5
ЭВМ-003	5
ЭВМ-004	1,5

SELECT SurName,Rate FROM Student, Rate WHERE Student.ID_Student=Rate.ID_Student and Rate.Rate > (SELECT AVG(Rate.Rate) FROM Rate.

Рекомендации по применению подзапросов:

- подзапрос должен быть заключен в круглые скобки;
- подзапрос должен указываться в правой части оператора сравнения;
- подзапросы не могут обрабатывать свои результаты, поэтому в подзапрос не может быть добавлено условие ORDER BY;
- используйте однострочные операторы с однострочными подзапросами;
- если подзапрос возвращает во внешний запрос значение NULL, внешний запрос не будет возвращать никакие строки при использовании операторов сравнения в условии WHERE.

Типы подзапросов:

- Однострочный подзапрос: возвращает ноль или одну строку;
- Многострочный подзапрос: возвращает одну или несколько строк;
- Многостолбцовый подзапрос: возвращает один или несколько столбцов;
- Коррелированные подзапросы: указывают один или несколько столбцов во внешней инструкции SQL. Такой подзапрос называется коррелированным, поскольку он связан с внешней инструкцией SQL;
- Вложенные подзапросы: подзапросы помещенные в другой подзапрос.

Подзапросы с инструкцией INSERT

Синтаксис:

```
INSERT INTO имя_таблицы [ (столбец1 [, столбец2 ]) ]
SELECT [ *|столбец1 [, столбец2 ]
FROM таблица1 [, таблица2 ]
[ WHERE VALUE OPERATOR ];
```

Пример (БД Приложения 2):
Вставить диски из таблицы ‘CD‘ в таблицу ‘CD1‘, у которых затраты меньше среднего.

```
INSERT INTO CD1
SELECT * FROM CD WHERE Затраты < AVG(Затраты)
```

Подзапросы с инструкцией UPDATE

```
UPDATE таблица  SET имя_столбца = новое_значение
[ WHERE OPERATOR [ VALUE ]
(SELECT COLUMN_NAME
FROM TABLE_NAME)
[ WHERE) ]
```

Подзапросы с инструкцией DELETE

```
DELETE FROM TABLE_NAME
[ WHERE OPERATOR [ VALUE ]
(SELECT COLUMN_NAME
FROM TABLE_NAME)
[ WHERE) ]
```

10.4. 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

10.5. Контрольные вопросы

1. Что такое вложенный запрос?
2. Где может использоваться вложенный запрос?
3. Какие задачи решаются с использованием подзапроса?
4. Какие бывают типы подзапросов.
5. Для чего и как используются подзапросы с инструкцией INSERT?
6. Для чего и как используются подзапросы с инструкцией UPDATE?
7. Для чего и как используются подзапросы с инструкцией DELETE?

11. Практическое занятие №9

Логическая оптимизация запросов

11.1. Цель работы

Целью практического занятия является приобретение навыков оптимизации запросов в SQL, в том числе и содержащих подзапросы.

11.2. Задание на выполнение работы

1. Для разработанных для выполнения п.9.2 и 10.2 запросов определить количество выполняемых операций при условии, что в каждой таблице 100 запросов. Минимизировать количество операций, поменяв запрос или доказать, что уменьшение невозможно.

2. Сформировать план выполнения запроса с использованием операций реляционной алгебры.

11.3. Краткие теоретические сведения

Оптимизацию запросов можно рассматривать в двух аспектах: как функцию СУБД, осуществляющую поиск оптимального плана выполнения запросов из всех возможных для заданного запроса, и как процесс изменения запроса и/или структуры БД с целью уменьшения использования вычислительных ресурсов при выполнении запроса. Рассмотрим только второй аспект в рамках занятия.

План выполнения запроса – это последовательность выполняемых оптимизатором действий, которые обеспечивают выбранные пути доступа.

Один и тот же результат может быть получен СУБД различными способами (планами выполнения запросов), которые могут существенно отличаться и по затратам ресурсов, и по времени выполнения. Задача оптимизации заключается в нахождении оптимального способа.

Во время процесса оптимизации запросов определяются пути доступа для всех типов команд SQL, но команда SELECT представляет наибольшую сложность в решении задачи выбора пути доступа. Поэтому этот процесс чаще называют оптимизацией запроса, а не оптимизацией путей доступа к данным. При этом термин «оптимизация запросов» нельзя считать точным, так как не факт, что в процессе оптимизации запроса будет действительно получен оптимальный путь доступа. Более точным был бы термин «улучшение запроса».

Ранние версии реляционных СУБД обрабатывали запросы без какой-либо попытки оптимизировать как запрос сам по себе, так и путь доступа. Результат - неудовлетворительно большое время обработки запроса. Для того чтобы увеличить скорость обработки запросов, были проведены исследования путей поиска способов повышения эффективности обработки запросов. И в настоящее время понятие «оптимизация запросов» можно определить как сумму всех технических приемов, которые применяются для повышения эффективности обработки запросов. К ним относят синтаксическую оптимизацию, оптимизацию, основанную на правилах и оптимизацию, основанную на вычислении стоимости. Рассмотрим только первую.

Синтаксическая оптимизация - способ переформулирования запроса таким образом, что новое представление запроса обеспечивало тот же результат, но было более эффективно для обработки СУБД.

Пример. Рассмотрим запрос, который делает выборку данных из таблиц Student и Group, а именно выборку студентов заданной группы, родившихся после 2000 года:

```
SELECT Student.Surname, Group.Name
FROM Student, Group
WHERE Student.YearBorn > 2000 AND
        Student.Group = Group.Name AND Group. Group_ID = “EVM-3-1”
```

Наиболее очевидный путь обработки этого запроса состоит в следующем.

- Формируем декартово произведение таблиц Student и Group.
- Ограничиваемся в результирующей таблице строками, которые удовлетворяют условию поиска в предложении WHERE.
- Выполняем проекцию результирующей таблицы на список колонок, указанный в предложении SELECT.

Процесс обработки этого запроса оценим в терминах операций ввода-вывода. Пусть для определенности таблица Student содержит 1000 строк, а таблица Group — 50 строк. Тогда формирование декартова произведения потребует 50000 операций чтения и операций записи (в результирующую таблицу). Для ограничения результирующей таблицы потребуется более 50000 операций чтения и, если 20 строк удовлетворяют условиям поиска, то 20 операций записи. Выполнение операции проекции вызовет еще 20 операций чтения и 20 операций

записи. Таким образом, обработка этого запроса обойдется системе более, чем в 100000 операций чтения и записи.

Основная идея синтаксической оптимизации - применение эквивалентных алгебраических преобразований. Каждый оператор SELECT эквивалентен некоторой формуле языка манипулирования множествами, значит можно использовать набор алгебраических правил для тождественных преобразований формул над множествами. Для рассматриваемого примера ограничение по условию поиска (по группе) должно быть выполнено как можно раньше, для того чтобы ограничить число строк, которые могут быть обработаны позже.

Применяя это правило к запросу, приведенному выше, определим следующий процесс обработки запроса:

1. Ограничение по условию поиска во второй таблице (Group.Group_ID = “EVM-3-1”) приведет к 1000 операций чтения и 20-ти операциям записи.
2. Выполнение соединения полученной на 1 шаге результирующей таблицы потребует 20 операций чтения результирующей таблицы, 100 операций чтения из таблицы STUDENT и 20 операций записи в новую результирующую таблицу

Обработка запроса в этом случае потребует 1120 операций чтения и 40 операций записи для получения того же самого результата, что и в первом случае.

Существует много формул для выполнения таких преобразований, и все современные оптимизаторы запросов используют правила преобразований для представления запроса в более эффективной форме для обработки.

11.4. Список рекомендуемой литературы

1. Ульман Л. MySQL / **Ульман** Л. /MySQL / Москва / ДМК Пресс / 2012 http://www.iqlib.ru/
2. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. SQL. Полное руководство / Вильямс, 2015 – 959 с. http://forcoder.ru/sql/sql-polnoe-rukovodstvo-1485

11.5. Контрольные вопросы

1. Для чего нужна логическая оптимизация запросов?
2. Назовите критерии оптимального запроса.
3. Назовите стратегии поиска оптимального плана.
4. Какие факторы учитываются при выполнении плана запроса?
5. Какие можно получить показатели при оценке числа извлекаемых строк?

12. Практическое занятие №10

Модели структурного проектирования

12.1. Цель работы

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;
- 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.

12.2. Задание на выполнение работы

Сформировать диаграммы ER-типа и ER-экземпляра для информационной системы на основе исходных данных п.4.2.

Сформировать функциональную схему информационной системы на основе исходных данных п.4.2 в методологии SADT.

12.3. Краткие теоретические сведения

12.3.1. Основные понятия метода «сущность-связь»

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; 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):

12.3.2. Этапы проектирования

Процесс проектирования базы данных является итерационным, допускающим 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.

12.3.3. Правила формирования отношений

Правила формирования отношений основываются на учете:

- степени связи между сущностями;
- класса 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 Н:Н, 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:М.

Правило 4.

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

Правило 5.

Если класс принадлежности М-связной сущности Н, 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 М:М.

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.

12.3.4. Методология SADT

Методология SADT (Structured Analysis and Design Technique – методология структурного анализа и проектирования) представляет собой совокупность методов, правил и проце-

дур, 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 Росстандартом 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.

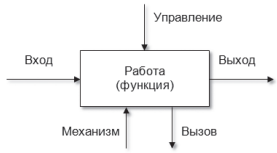
Контекстная диаграмма (диаграмма верхнего уровня) - вершина древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция,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

дующих уровней – состоят из буквы «А», за которой следует номер декомпозируемого блока (например, «А11», «А12», «А25», «А63»).

12.4. Список рекомендуемой литературы

1. С.В. Маклаков "BPwin и Erwin. CASE-средства для разработки информационных систем". ДИАЛОГ-МИФИ

12.5. 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.
10. Перечислите распространенные модели и диаграммы графического представления, используемые при структурном анализе и проектировании.
11. Сформулируйте правила построения диаграмм IDEF0.
12. Определите типы связей между работами (функциями).

13. Практическое занятие №11

Проектирование баз данных с использованием языка UML.

13.1. Цель занятия

Целью практического занятия является приобретение навыков использования распространенных моделей и диаграмм графического представления UML.

13.2. Задание на практическое занятие

Выполнить описание основной функции системы для информационной системы на основе исходных данных п.4.2. в нотации UML, используя диаграммы активности, состояний и прецедентов использования.

13.3. Краткие теоретические сведения

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:

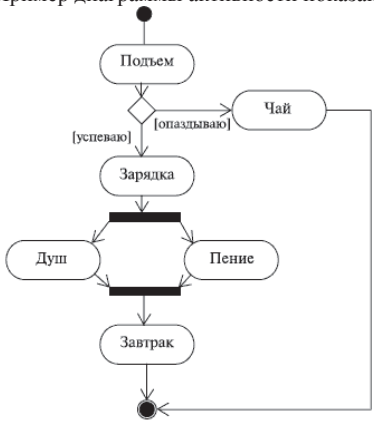


Диаграмма состояний - спецификация последовательности состояний, через которые проходит объект или взаимодействие в ответ на события своей жизни, а также ответные действия объекта на эти события.

Пример:

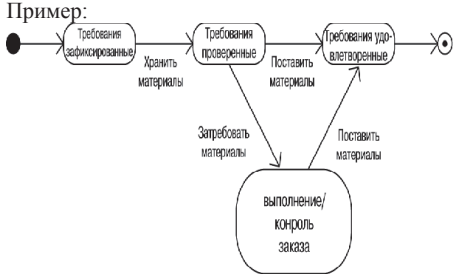


Диаграмма прецедентов использования описывают функциональность ИС, видимую пользователями системы.

Пример:

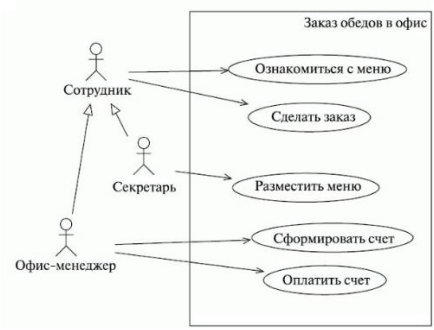


Диаграмма классов - статическая структурная диаграмма, описывающая структуру системы, демонстрирующая классы системы, их атрибуты, методы и зависимости между классами.

Пример:

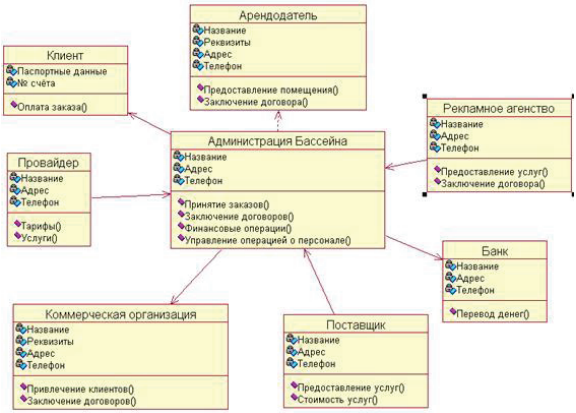
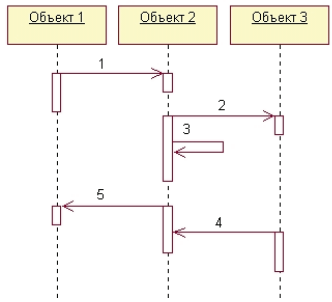


Диаграмма следования предназначена для моделирования взаимодействия объектов системы во времени, а также обмена сообщениями между ними.

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,
– Creately,
– Magic Draw,
– Torgether UML Designer,
– Poseidon,
– MS Visio Professional.

13.4. 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

13.5. Контрольные вопросы

1. Что представляет унифицированный язык моделирования UML?
2. Назовите типы диаграмм UML.
3. Приведите пример диаграммы потоков данных.
4. Для чего служит диаграмма прецедентов использования?
5. Для чего служит диаграмма классов?
6. Для чего служит диаграмма следования?
7. Назовите основные case-системы, поддерживающие UML.
8. Назовите примеры инструментальных средств UML (профессиональных и полупрофессиональных).

14. Практическое занятие №12
Архитектура TOGAF

14.1. Цель занятия

Целью практического занятия является изучение методов архитектурного проектирования баз данных TOGAF (The Open Group Architecture Framework) и приобретение навыков описания архитектуры предприятия.

14.2. Задание на практическое занятие

Выполнить описание архитектуры предприятия (одного из вариантов) в части бизнес-архитектуры и архитектура информации. Варианты предприятия: вуз, школа, детский сад, библиотека, поликлиника, ресторан, спортивная школа, музей.

14.3. Краткие теоретические сведения

TOGAF (The Open Group Architecture Framework) - это инфраструктура архитектуры предприятия, которая появилась в начале 1990-х годов с целью стать стандартом разработки архитектуры предприятия.

Архитектура предприятия (АП) определяет общую структуру и функции систем (бизнес и ИТ) в рамках всей организации в целом и обеспечивает общую рамочную модель (framework), стандарты и руководства для архитектуры уровня отдельных проектов. Общее видение, обеспечиваемое архитектурой предприятия, создает возможность единого проектирования систем, адекватных, с точки зрения обеспечения потребностей организации, и способных к взаимодействию и интеграции там, где это необходимо. В более узком смысле под архитектурой предприятия понимают корпоративную архитектуру ИТ – видение, принципы и стандарты, которыми организации руководствуются при разработке и внедрении информационных технологий.

TOGAF разработан и развивается за счет коллективной работы членов консорциума «The Open Group».

Сначала TOGAF включала только технические аспекты архитектуры (версии с 1 по 7), однако когда в эту инфраструктуру была добавлена предметная область архитектуры бизнеса (с версии 8, Enterprise Edition), TOGAF стала наиболее используемым программным продуктом для формирования вариантов инфраструктур архитектуры предприятий, и является наиболее распространенным и узнаваемым в мировой практике фреймворком.

В настоящее время актуальной версией является TOGAF 9.1.

TOGAF предполагает разделение на четыре домена для описания целостной архитектуры предприятия:

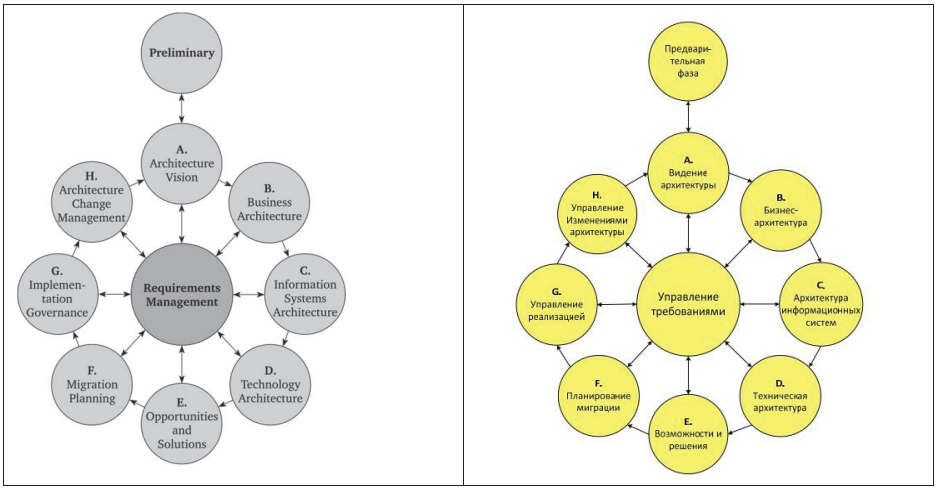
- бизнес-архитектура (Business Architecture);
- архитектура приложений (Application Architecture);
- архитектура данных (Data Architecture);
- технологическая архитектура (Technical Architecture).

Архитектура данных — в области информационных технологий архитектура данных состоит из моделей, политик, правил или стандартов, которые определяют, какие данные собираются, и как они хранятся, размещаются, интегрируются и используются для использования в системах данных и в организациях. Элементы должны быть определены на этапе проектирования схемы архитектуры данных. В частности, должна быть описана административная структура, которая будет создана для управления ресурсами данных, методологии, которые будут использоваться для хранения данных. Должно быть создано описание используемой технологии базы данных и описание процессов, которые будут обрабатывать данные. Во время работы над архитектурой предприятия создают документы, схемы, презентации, планы и т.п. Весь этот объем информации образует информационный контент архитектуры предприятия. Метамодель контента — это инструмент организации архитектурной информации таким образом, чтобы она была сконцентрирована вокруг потребностей заинтересованных сторон.

Основное отличие от других методов построения архитектуры — наличие метода разработки архитектуры (Architecture Development Method —ADM), который фактически дает алгоритм комплексного решения задачи построения АП. Материалы TOGAF содержат большое количество примеров и шаблонов архитектурных артефактов, создаваемых на разных этапах ADM, широкий наборо архитектурных продуктов: концепции, видение, модели, описания, диаграммы, базы технических стандартов для конкретных решений, каталоги и ряд других.

Метод разработки архитектуры (Architecture Development Method — ADM) методологии TOGAF предоставляет законченный набор инструкций для реализации и выполнения АП

в организации. Этот процесс, состоящий из нескольких последовательных фаз, замкнутых в цикл, представлен на рисунке:



Задача подготовительной фазы (Preliminary Phase) — выявление заинтересованных в процессе реализации лиц и обсуждение с ними задач АП, выработка руководящих принципов архитектуры, которые основываются на бизнес-принципах организации и описывают процессы и критерии для наблюдения за процессом реализации АП.

Фаза A (Видение архитектуры) предназначена для выражения видения (концепции) АП, чтобы обозначить цель действий по созданию АП и создать описания первого среза для базовой и целевой среды. Результат - специальный документ под названием «Задание на разработку архитектуры», очерчивающий область действия и условия АП.

Фаза B (Бизнес-архитектура) предназначена для детальной разработки архитектуры предметной области бизнеса. И базовая, и целевая архитектура детализируются, чтобы получить входные данные для технического анализа.

Фаза C (Архитектура информационных систем) связана с созданием архитектуры предметных областей «Приложение» и «Данные (Информация)». На этой фазе производится описание текущей и разработка целевой архитектуры информационных систем. Используются специальные методики, подходы и нотации, связанные с данными и приложениями.

Фаза D (Техническая архитектура) фокусируется на описании и разработке технологической архитектуры (сети, вычислительные устройства, узлы, технологические интерфейсы и т. п.).

Фаза E (Возможности и решения) — выяснение возможностей, предлагаемых целевой архитектурой, и создание эскиза потенциального решения. Создается план реализации и внедрения, определяются основные проекты.

В фазе F (Планирование миграции) расставляются приоритеты проектов реализации, выполняются детализированное планирование и анализ просчетов процесса миграции. Обновляется список проектов, детализируется «План реализации и внедрения».

В фазе G (Управление реализацией) формулируются более конкретные условия и рекомендации для каждого из проектов реализации. Устанавливается связь между управляющей архитектурой (TOGAF) и разрабатывающей организацией. На выходе этой фазы - «Архитек-

турные контракты», утверждаемые организацией-разработчиком, решения, совместимые с архитектурой.

В фазе H (Управление изменениями архитектуры) акцент ставится на управление изменением архитектуры, которая достигается поставкой реализованных решений.

Метод является масштабируемым и может использоваться как для разработки архитектуры компании в целом, так и для конкретного ИТ-решения. Возможно использование в совокупности с другими методами, более специализированными на конкретных задачах, а также с отраслевыми методами и стандартами.

Принципы в области управления данными:

- Бизнес-структуры (отделы, департаменты, ведомства), являющиеся владельцами данных, отвечают за целостность и распространение данных.
- Данные уровня отдельных бизнес-структур (департамента, региона, города) должны быть явно описаны и доступны всем остальным бизнес-структурам (департаментам, ведомствам).
- Ведомства собирают только самый необходимый минимум данных и стремятся уменьшить нагрузку на тех, кто должен предоставлять данные.
- Данные вводятся в информационные системы один раз, и тут же выполняется проверка их корректности.
- Информация является ценным ресурсом, который передан в управление менеджерам (государственным служащим), и этим ресурсом необходимо соответствующим образом управлять.

14.4. Список рекомендуемой литературы

1. https://pubs.opengroup.org/architecture/togaf9-doc/arch/
2. https://ru.qaz.wiki/wiki/The_Open_Group_Architecture_Framework
3. https://docplayer.ru/38594475-The-open-group-architecture-framework-togaf-osnovy-arhitekturnogo-standarta-the-open-group.html
4. http://www.interface.ru/home.asp?artId=5082

14.5. Контрольные вопросы

1. Расшифруйте TOGAF.
2. Что такое TOGAF?
3. Назовите домены архитектуры TOGAF.
4. Что такое архитектура информации?
5. Что такое бизнес-архитектура?
6. Назовите компоненты TOGAF.
7. Принципы в области управления данными.
8. Перечислите общие принципы архитектуры.

15. Задачи для подготовки к контрольной работе

Задача 1.

Дана таблица «ПОСТАВЩИКИ» с данными о поставщиках, в которой имеются следующие столбцы:

- Город;
- Номер;
- Название.

Необходимо сконструировать SQL-запрос, который возвращал бы поставщиков, в названии которых встречается слово «сбыт», если они находятся в заданном городе.

Задача 2.

Дана таблица «КОРАБЛИ» с данными, в которой имеются следующие столбцы:

- Название;
- Владелец;
- Порт приписки;
- Количество членов команды;
- Год спуска на воду.

Необходимо сконструировать SQL-запросы, которые возвращали бы:

- Владельца и название самого старого корабля;
- Общее количество членов всех команд;

Задача 3.

Дана таблица «ПОСТАВЩИКИ» с данными о поставщиках, в которой имеются следующие столбцы:

- Город;
- Номер поставщика;
- Название поставщика.

Дана таблица ПОСТАВКИ с данными о поставках, в которой имеются следующие столбцы:

- Номер детали;
- Номер поставщика;
- Название детали.

Необходимо сконструировать SQL-запрос, который возвращал бы названия поставщиков, которые поставляют только одну деталь.

16. Вопросы для подготовки к зачету

1. Информационная система и её компоненты.
2. Разновидности архитектур информационных систем. Достоинства. Недостатки.
3. Классификация СУБД. Типы данных, используемые в СУБД.
4. Низкоуровневые функции СУБД
5. Модели данных, используемые в СУБД.
6. Локальные информационные системы.
7. Иерархическая модель представления данных.
8. Сетевая модель представления данных.
9. Реляционная модель представления данных.
10. Постреляционная модель представления данных.
11. Многомерная модель представления данных.
12. Объектно-ориентированная модель представления данных.
13. Средства автоматизации проектирования.
14. Модели структурного проектирования.
15. Жизненный цикл программного обеспечения. Модели жизненного цикла.
16. Классификация CASE-средств.
17. Декартово произведение. Отношение. Предикат отношения.
18. Структура реляционной модели. Типы данных реляционной модели.
19. Отношение. Атрибуты и кортежи. База данных. Схема БД. Свойства отношений. Первая нормальная форма.

20. Первичный ключ. Целостность сущностей.
21. Внешний ключ. Целостность внешних ключей.
22. Операции, могущие нарушить ссылочную целостность. Стратегии поддержания ссылочной целостности.
23. Замкнутость реляционной алгебры. Совместимость отношений по типу. Её достижение.
24. Теоретико-множественные операторы реляционной алгебры. Реляционное исчисление.
25. Выборка. Проекция. Соединение.
26. Естественное соединение. Деление. Зависимые реляционные операторы. Примитивные реляционные операторы.
27. Операторы SQL. Примеры.
28. Оператор SELECT. Использование агрегатных функций. Использование группировок.
29. Использование имён корреляции. Использование сортировок.
30. Оператор SELECT. Использование подзапросов. Использование объединения, пересечения и разности
31. Оператор SELECT. Предикат EXIST. Предикат LIKE. Предикат IN. Предикат BETWEEN. Предикат NULL.
32. Порядок выполнения оператора SELECT.
33. Этапы разработки базы данных. Критерии оценки качества логической модели.
34. Первая нормальная форма. Функциональная зависимость.
35. Вторая нормальная форма.
36. Третья нормальная форма. Алгоритм нормализации.
37. Третья усиленная нормальная форма.
38. Четвертая нормальная форма.
39. Пятая нормальная форма.
40. Шестая нормальная форма.
41. Доменно-ключевая нормальная форма.
42. Элементы модели «Сущность-Связь».

17. 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 заданий на практических занятиях в соответствии с логикой работы должно быть применено при выполнении курсовой работы, отражено в пояснительной записке и пояснено во время её защиты.

Приложение 1. Отношения для выполнения реляционных операций

P

ID_P	Имя	Статус	Город_П
S1	Сергей	20	Москва
S2	Иван	10	Сочи
S3	Борис	30	Сочи
S4	Николай	20	Москва
S5	Андрей	30	Минск

D

ID_D	Название	Тип	Вес	Город_Д
P1	гайка	каленый	12	Москва
P2	болт	мягкий	17	Сочи
P3	винт	твердый	17	Ростов
P4	винт	каленый	14	Москва
P5	гвоздь	твердый	12	Сочи
P6	шпилька	каленый	19	Москва

P2

ID_P	Имя	Статус	Город_П
S0	Денис	20	Москва
S2	Иван	10	Сочи
S6	Максим	30	Сочи
S7	Олег	20	Москва
S8	Артём	30	Минск

PD

ID_P	ID_D	Количество
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

PRODUCTS

ID	NAME	COMPANY	PRICE
123	Кофе	ООО "Лунатик"	190
156	Чай	ООО "Лунатик"	60
235	Бананы	ОАО "Фрукты"	100
623	Баклажаны	ООО "Овощи"	130

DRIVERS

COMPANY	DRIVER
ООО "Темная сторона"	Иванов
ООО "Темная сторона"	Михайлов
ОАО "Фрукты"	Русланов
ООО "Овощи"	Владимиров

SELLERS

ID	SELLER
123	ООО "Дарт"
156	ОАО "Ведро"
235	ЗАО "Овоще База"
623	ОАО "Фирма"

Приложение 2. База данных «CD-диск»

Song

Song_ID	Song_name	Song_length	ID_CD
1	Интро	3.50	9
2	Пар	4.10	9
4	Я или Ты	4.25	9
5	Троллейбус	5.10	6
6	Космос	4.37	6
7	Через все времена	5.42	5
8	Город	7.12	5
9	Блики солнца на воде	6.11	5

Artist

Artist_ID	Name
1	Баста
2	Браво
3	Мельница
4	Пикник
5	Кипелов
6	Би-2
7	Звери
8	Земфира
9	Ария
10	Алиса

Record_label

R_L_ID	RL_Title	Adress
1	Студия Владимира Матецкого	Проспект Мира, 21
2	Игорь Крутой компани	ул. Тверская, 15
3	Апрель	ул. Полянка, 17

CD

CD_ID	CD_Title	CD_artist	Year	Тираж	Затраты	Stud_ID
2	Районы-кварталы	7	2004	203	1505	1
3	Навсегда	2	2017	777	672	2
4	Live in Studio	9	2013	2000	996	1
5	Через все времена	9	2015	1000	1234	1
6	Баста 5. Часть	1	2016	345	3211	3
7	Баста 2	1	2007	1121	456	3
9	Баста 4	1	2013	222	333	3