Теория баз данных

Лекция 1. Введение

Е. П. Моргунов
Сибирский федеральный университет
г. Красноярск
Институт космических и информационных технологий
emorgunov@mail.ru

Версия от 29.01.2018

1.1. Основы и история
1.1.1. Подход на основе файлов

Данные хранились в плоских файлах (flat files). Структура: записи, разделенные на поля фиксированной длины.

Проблемы:

- Трудно организовать совместную обработку данных из разных файлов.
- Дублирование данных (это затраты времени на повторный ввод и обработку данных, расход дисковой памяти на их хранение, рассогласование данных, которые в одном месте могли быть изменены, а в другом – нет).
- Зависимость приложений от данных (физическая структура и методы доступа к данным определялись непосредственно в программном коде, поэтому при изменении структуры данных необходимо откорректировать все модули приложения, в которых эти данные используются, а также нужно провести реструктуризацию самих данных, возможно, написав специальные программы).
- Несовместимость форматов файлов при использовании разных языков программирования.
- Невозможность выполнения разовых (ad hoc) запросов, которые изначально не были предусмотрены.
- Нет гарантий защищенности и целостности.
- Проблемы с восстановлением данных после сбоя.
- Сложность организации совместного доступа к данным.

Итак, коротко:

- Определения данных встроены непосредственно в код прикладных программ, а не хранятся отдельно и независимо от них.
- Нет никакого контроля за доступом к данным, кроме того, который налагается прикладными программами.

1.1.2. Подход на основе баз данных

- **База данных** – коллекция логически связанных данных и их описаний, предназначенных для удовлетворения информационных потребностей предприятия (организации).
- **База данных** — это некоторый набор перманентных (постоянно хранимых) данных, используемых прикладными программными системами какого-либо предприятия.
- **Описания данных** – системный каталог, или словарь данных, или метаданные (данные о данных).
- **Система управления базами данных (СУБД)** – программная система, позволяющая пользователям определять, создавать, поддерживать данные и управлять доступом к базе данных.
- **Система баз данных** — это компьютеризированная система хранения записей, т. е. компьютеризированная система, основное назначение которой — хранить информацию, предоставляя пользователям средства ее извлечения и модификации.
- Главные компоненты системы: данные, аппаратное обеспечение, программное обеспечение, процедуры и пользователи.
- **Данные**
  Данные в БД являются интегрированными и разделяемыми.
  Интегрированные – объединяют различные данные, исключается избыточность хранения данных.
  Разделяемые – обеспечивается совместный доступ к данным (возможно, параллельный).
- **Аппаратное обеспечение**
- **Программное обеспечение**
  - сервер базы данных (database server) или, что более привычно, система управления базами данных, СУБД (Database Management System — DBMS)
  - утилиты
  - средства разработки приложений
  - средства проектирования
  - генераторы отчетов

1.1.2. Подход на основе баз данных (продолжение)

- **Процедуры**
  - вход в систему
  - запуск и останов сервера баз данных
  - выполнение резервного копирования базы данных
  - обработка сбоев и т. д.
- **Пользователи**
  - прикладные программисты
  - конечные пользователи
  - администраторы баз данных (АБД) (database administrators (DBA))
  - администраторы данных (АД) (data administrators (DA))
  - проектировщики базы данных
- **Администратор данных**
  - администратор должен разбираться в данных
  - понимать нужды предприятия по отношению к данным *на уровне высшего управляющего звена* в руководстве предприятием
- **Администратор *базы* данных,** или АБД
  Администратор базы данных, в отличие от администратора данных, должен быть профессиональным специалистом в области информационных технологий. Это *технический* специалист, ответственный за реализацию решений администратора данных. Он отвечает за физическую реализацию БД, обеспечение целостности, защищенности, производительности.

1.1.3. История баз данных

- Середина 1960-х гг. Иерархические СУБД (IMS компании IBM)
- Середина 1960-х гг. Сетевые СУБД (IDMS/R компании Computer Associates)
- 1970 г. Доктор E. F. Codd предложил реляционную модель.
- 1970-е гг. Предложены первые реляционные СУБД (Ingres в Berkeley и System R в компании IBM, язык SQL).
- 1976 г. П. Чен предложил модель «сущность–связь» (entity–relation) – ER-модель.
- 1979 г. Появление коммерческих реляционных СУБД Oracle, Ingres, DB2
- 1987 г. Стандарт ISO языка SQL (последующие выпуски стандарта: 1989, 1992 (SQL2), 1999 (SQL:1999), 2003 (SQL:2003), 2008 (SQL:2008), 2011 (SQL:2011)
- 1990-е гг. Появление объектно-ориентированных СУБД и объектно-реляционных СУБД
- 1990-е гг. Появление хранилищ данных (data warehousing)
- Середина 1990-х гг. Интеграция web с базами данных
- Середина 1990-х гг. – по настоящее время. Развитие open source СУБД (PostgreSQL, MySQL и др.)

1.1.4. Преимущества и недостатки баз данных

Преимущества

- Возможность совместного доступа к данным
- Сокращение избыточности данных (допустима контролируемая избыточность)
- Обеспечение согласованности данных (Data consistency), т. е. устранение противоречивости данных (до некоторой степени) (это следствие предыдущего пункта)
- Возможность поддержки транзакций при параллельной работе разных пользователей и программ

  Транзакция (transaction) — это логическая единица работы (точнее, логическая единица работы базы данных), обычно включающая несколько операций базы данных (в частности, несколько операций модификации данных).
- Обеспечение целостности данных

  Ограничения целостности (integrity constraints), которые будут применяться при любой попытке внести какие-либо изменения в соответствующие данные. Ограничения могут касаться одной записи или взаимосвязанных записей.
- Организация защиты данных
- Возможность согласования противоречивых требований
- Возможность введения стандартизации
- Обеспечение независимости от данных (data independency)
- Экономия на масштабе за счет централизации данных
- Баланс конфликтующих требований
- Улучшенный доступ к данным (возможность писать ad hoc запросы)
- Повышение продуктивности разработчиков за счет наличия стандартных процедур
- Более надежные процедуры резервного копирования данных и восстановления после сбоев

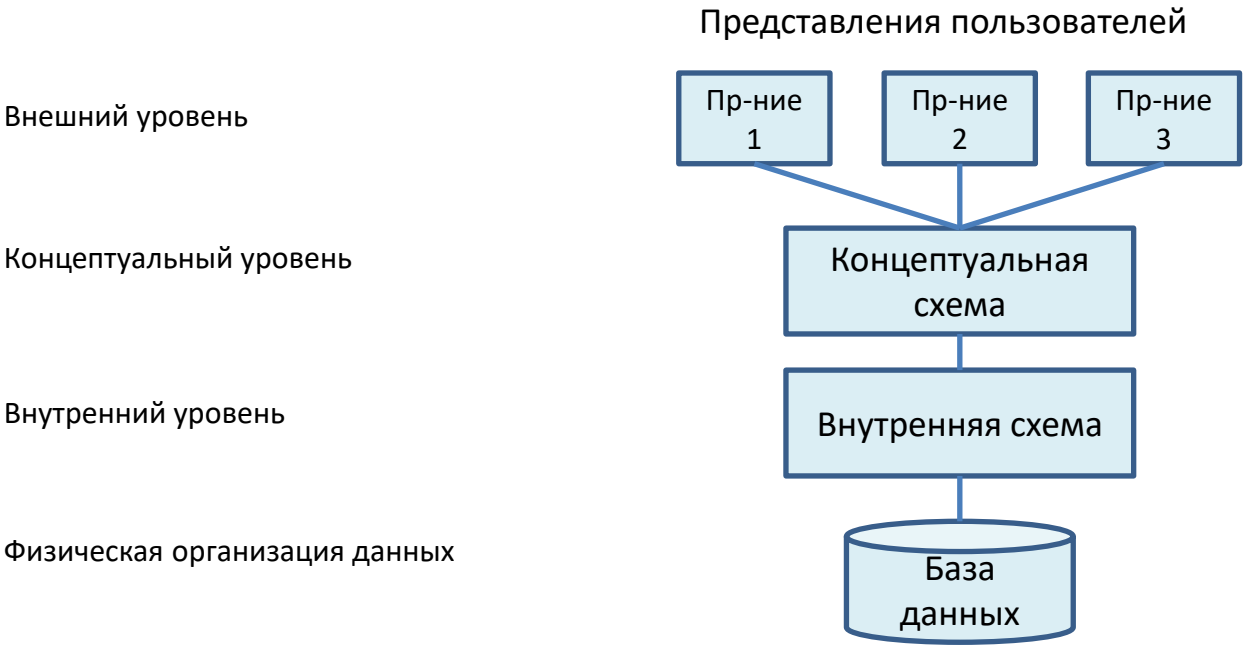
1.1.4. Преимущества и недостатки баз данных

Недостатки

- Сложность (нужны специалисты)
- Большой масштаб (следствие – требуется мощное аппаратное обеспечение, много памяти, быстрые процессоры)
- Высокая стоимость коммерческих СУБД (но есть альтернатива – свободное ПО, например, PostgreSQL)
- Высокая стоимость перехода от устаревших систем, основанных на файлах, к современным СУБД
- Более низкая скорость работы, чем у специализированных приложений, написанных с учетом специфики решаемой задачи
- Больший масштаб проблем в случае сбоя сервера баз данных

1.2. Системы баз данных
1.2.1. Архитектура ANSI/SPARC

- В 1971 г. подразделением DBTG (Data Base Task Group) организации CODASYL (Conference on Data Systems Language) был предложен двухуровневый подход: системное представление (system view), называемое *схемой* и пользовательские представления (user views), называемые *подсхемами.*
- Комитет под названием Standards Planning and Requirements Committee (SPARC) Американского национального института стандартов (The American National Standards Institute (ANSI)), т. е. ANSI/X3/SPARC, в 1975 г. предложил трехуровневую архитектуру.
- Хотя архитектура ANSI/SPARC не стала стандартом, она является базисом для понимания функциональности СУБД.
- Архитектура ANSI/SPARC включает три уровня: внутренний, концептуальный, внешний.



1.2.1.1. Уровни архитектуры ANSI/SPARC

Разделение на уровни желательно по следующим причинам:

1. Если разные пользователи имеют возможность получать одни и те же данные, то они должны иметь возможность видеть их в различных представлениях. Изменение способа отображения данных для одного пользователя не должно влиять на способ отображения этих же данных для другого пользователя.
2. Пользователи не должны взаимодействовать напрямую с деталями физического представления данных.
3. ДБА должны иметь возможность изменения концептуальной структуры базы данных или структур хранения базы данных без влияния на представление данных пользователям.

Три уровня: внешний, концептуальный и внутренний

- **Внешний уровень** (называемый также *пользовательским логическим)* наиболее близок к пользователям, т. е. связан со способами представления данных для отдельных пользователей.
- Одни и те же данные могут быть представлены разным пользователям по-разному.

1.2.1.1. Уровни архитектуры ANSI/SPARC (продолжение)

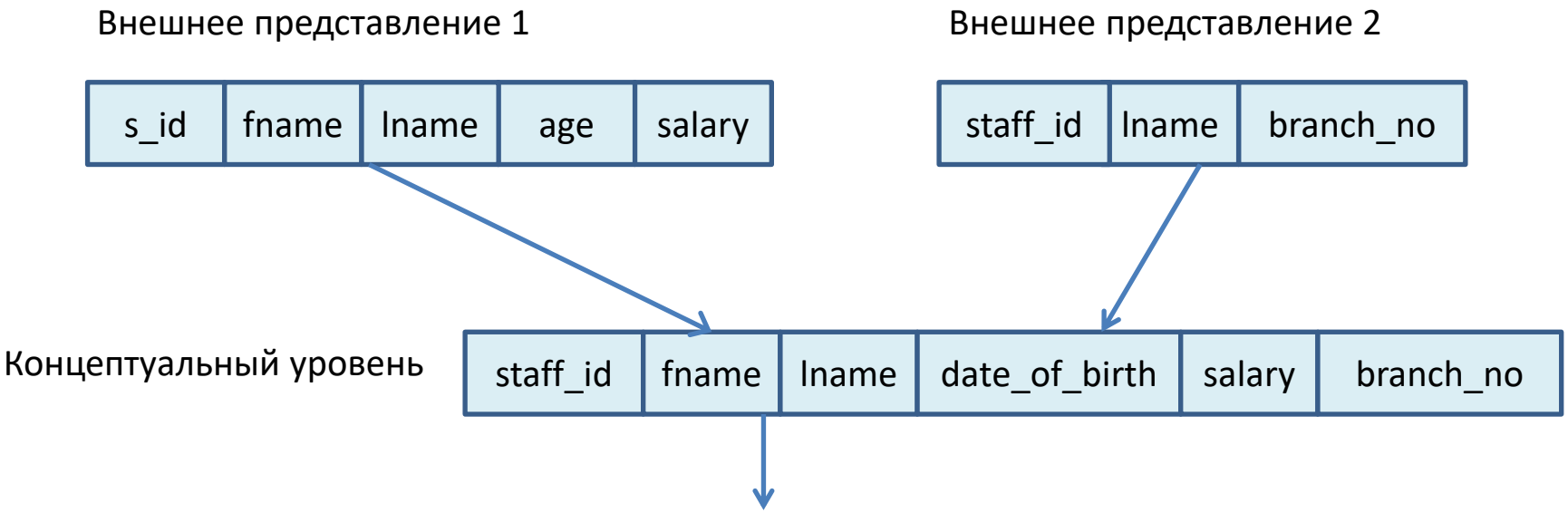
- **Концептуальный уровень** (называемый также *общим логическим* или просто *логическим,* без дополнительного определения) является «промежуточным» уровнем между двумя первыми. Он описывает, какие данные хранятся в базе данных.
- Если внешний уровень связан с *индивидуальными* представлениями пользователей, то концептуальный уровень связан с *обобщенным* представлением пользователей. Он представляет:
  - все сущности, их атрибуты и связи между ними
  - ограничения, накладываемые на данные
  - смысловую информацию о данных
  - информацию о защищенности данных и их целостности

- Концептуальное представление существенно отличается от представления данных какого-либо отдельного пользователя. Вообще говоря, концептуальное представление — это представление данных в том виде, какими они являются на самом деле, а не в том, какими их вынужден рассматривать пользователь в рамках, например, определенного языка.
- Чтобы добиться независимости от данных, нельзя включать в определения концептуального языка какие-либо указания о структурах хранения или методах доступа. Определения концептуального языка должны относиться *только* к содержанию информации. Это означает, что в концептуальной схеме не должно быть никакого упоминания о представлении хранимого файла, упорядоченности хранимых записей, индексировании, хэш-адресации, указателях или других подробностях хранения данных или доступа к ним.
- Концептуальное представление — это представление всего содержимого базы данных.

1.2.1.1. Уровни архитектуры ANSI/SPARC (продолжение)

- **Внутренний уровень** (иногда называемый также *физическим*. См. ниже) наиболее близок к физическому хранилищу информации, т. е. связан со способами сохранения информации на физических устройствах. Он описывает, каким образом данные хранятся в базе данных.
- Внутренний уровень отвечает за следующие вещи:
  - выделение пространства для хранения данных и индексов
  - описания хранимых записей (с указанием размеров хранимых элементов данных)
  - размещение записей
  - методы сжатия и шифрования данных
- Внутреннее представление описывается с помощью **внутренней схемы,** которая определяет не только различные типы хранимых записей, но также
  - существующие индексы
  - способы представления хранимых полей
  - физическую упорядоченность хранимых записей и т. д.
- Внутренняя схема формируется с использованием еще одного языка определения данных — **внутреннего.**
- Ниже внутреннего уровня лежит **физический уровень.** Он может реализовываться операционной системой при участии СУБД.

1.2.1.2. Схемы, отображения, экземпляры



Внутренний уровень

```
struct STAFF {
  int staff_id,
  int branch_no,
  char fname[ 15 ],
  char lname[ 15 ],
  struct date date_of_birth,
  float salary,
  struct STAFF *next   /* указатель на следующую запись */
};
index staff_id; index branch_no;  /* индексы */
```

1.2.1.2. Схемы, отображения, экземпляры (продолжение)

- Общее описание базы данных – **схема** базы данных (intension).
- Совокупность данных в базе данных в конкретный момент времени – **экземпляр** базы данных (extension).
- В соответствии с уровнями абстракции:
  - внешние схемы (подсхемы). Их может быть много (больше одной)
  - концептуальная схема
  - внутренняя схема
- СУБД отвечает за отображение между этими схемами:
  - «концептуальная/внутренняя»
  - «внешняя/концептуальная»

1.2.1.3. Независимость от данных

- Приложения зависимы от данных, если сведения об организации данных и способе доступа к ним встроены в саму логику и программный код приложения.
- **Независимость от данных** можно определить как невосприимчивость приложений к изменениям в физическом представлении данных и в методах доступа к ним.
- **Хранимое поле** — это наименьшая единица хранимых данных.

  Следует различать тип поля и экземпляр поля.
- **Хранимая запись** — это набор взаимосвязанных хранимых полей.

  Следует различать тип записи и экземпляр записи.
- **Хранимый файл** — это набор всех существующих в настоящий момент экземпляров хранимых записей одного и того же типа.

- В базах данных *логическая* (с точки зрения разработчика приложения) запись не всегда совпадает с соответствующей *хранимой* записью.
- База данных должна иметь возможность **расти** и развиваться, не нарушая логическую структуру существующих приложений. Например, добавленные новые хранимые поля будут невидимы для существующих приложений.

1.2.1.3. Независимость от данных (продолжение)

- Главная цель трехуровневой архитектуры – обеспечение независимости от данных.
- **Логическая независимость от данных** – невосприимчивость внешних схем к изменениям в концептуальной схеме.

  Добавление или удаление сущностей, атрибутов и связей должно быть возможно *без необходимости* изменения существующих внешних схем или переписывания прикладных программ. Пользователи, для которых эти изменения предназначены, должны быть осведомлены о них. Важно, что остальным пользователям об этих изменениях знать не нужно.
- **Физическая независимость от данных** – невосприимчивость концептуальной схемы к изменениям во внутренней схеме.

  Использование другой организации файлов или структур хранения данных, использование других устройств хранения данных, модификация индексов или алгоритмов хеширования должны быть возможны без необходимости изменения концептуальной или внешних схем. С точки зрения пользователя возможен только один заметный эффект – изменение производительности. Зачастую снижение производительности является причиной для проведения изменений во внутренней схеме.

1.2.2. Языки баз данных

- Используется термин подъязык данных (sublanguage).
- Любой подъязык данных является комбинацией по крайней мере двух подчиненных языков:
  - языка определения данных (Data Definition Language — DDL), который позволяет формулировать определения или объявления объектов базы данных (таблиц, представлений, индексов и др.)
  - языка манипулирования данными (Data Manipulation Language — DML), который поддерживает операции с такими объектами или их обработку:
    - добавление данных
    - изменение
    - выборка (извлечение)
    - удаление
- Подъязык данных – как правило, SQL (Structured Query Language).
- Подъязык данных может быть встроен в базовый язык или использоваться интерактивно.
- DML как правило бывают непроцедурными, т. е. требуют только указать, *какие* данные требуется получить, но не нужно указывать, *каким образом* это сделать.

1.2.3. Данные и модели данных

- **Модель данных** — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих *абстрактную машину доступа к данным,* с которой взаимодействует пользователь. Упомянутые объекты позволяют моделировать *структуру* данных, а операторы — *поведение* данных.
- Еще одно определение **модели данных** — интегрированная совокупность концепций для описания данных (и манипулирования ими), взаимосвязей между данными и ограничений на данные, имеющих место в организации.
- Модель данных включает три компонента:
  - Структурная часть — правила конструирования базы данных
  - Манипуляционная часть — определяет типы операций, которые допустимо выполнять над данными
  - Множество ограничений целостности — требования, предъявляемые к данным и их соотношениям
- **Реализация** (implementation) заданной модели данных — это физическое воплощение на реальной машине компонентов абстрактной машины, которые в совокупности составляют эту модель.
- Все модели данных можно разделить на три группы:
  - объектные (object-based)
  - основанные на записях (record-based)
  - физические (physical) модели

Физические модели данных

- unifying model
- frame memory

1.2.3.1. Объектные модели данных

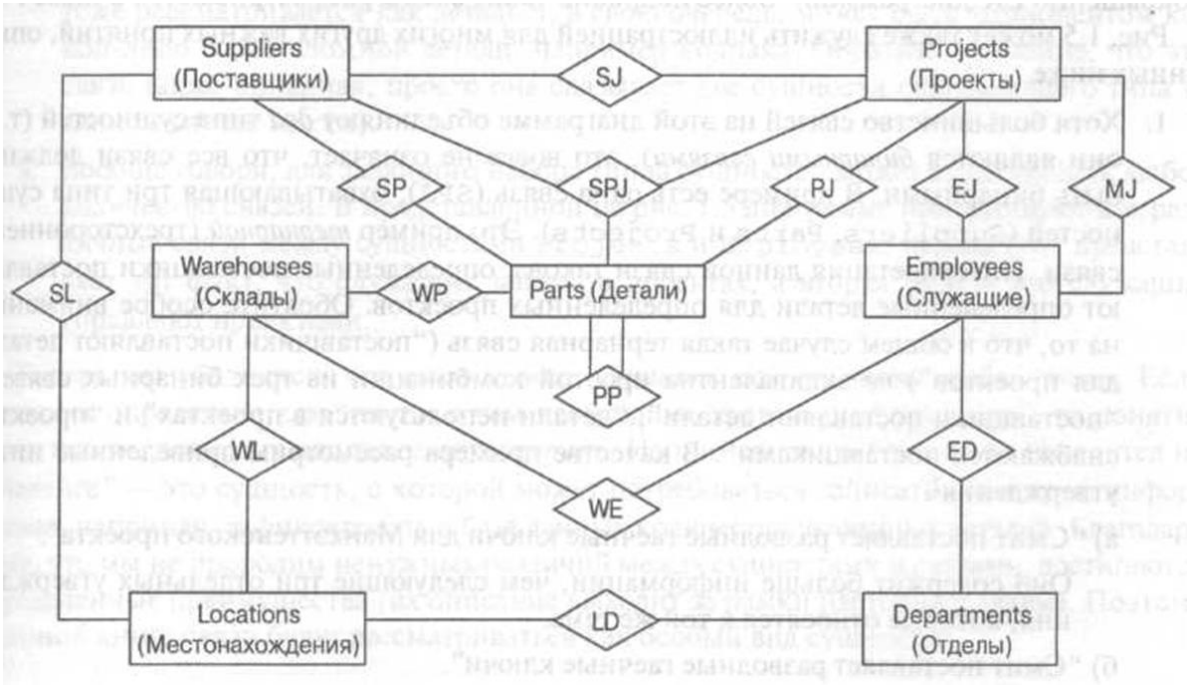
Эти модели основаны на следующих понятиях:

- **Сущность** – любой объект, информация о котором может (должна) быть представлена в базе данных.
- **Связь** – соединяет две или более сущностей и указывает вид взаимодействия между ними. Связи также должны быть представлены в базе данных. Связь можно понимать как сущность особого типа.
- **Свойства** – описывают (детализируют) сущности и связи.

Виды моделей

- Entity–Relationship (ER-модель)
- Семантические
- Функциональные
- Объектно-ориентированные

Диаграмма «сущность–связь» (ER-диаграмма)



1.2.3.2. Модели данных, основанные на записях

- В таких моделях база данных состоит из некоторого количества записей фиксированного формата, возможно, различных типов. Каждый тип записей определяет фиксированное число полей, возможно, фиксированной длины.
- Три главных типа таких моделей:
  - реляционная (relational data model)
  - сетевая (network data model)
  - иерархическая (hierarchical data model)
- **Иерархическая модель.** Данные представлены в виде коллекций **записей**, а связи представлены множествами (sets). Записи организованы в виде дерева. При этом записи являются узлами (nodes) графа, а связи – ребрами. Связи на уровне реализации представлены в виде указателей. В отличие от сетевой модели, каждый узел может иметь только один родительский узел.
- **Сетевая модель.** Является расширением иерархической модели. Записи организованы в виде графа. Каждый узел может иметь более одного родительского узла.
- Эти две модели требуют от пользователя знать физическую организацию базы данных, чтобы работать с данными. Таким образом снижается уровень независимости от данных.
- Системы, реализованные на основе таких моделей, являются *навигационными*. Это означает, что для получения данных нужно указать, каким образом их нужно извлекать.
- В настоящее время СУБД, построенные на основе иерархической и сетевой моделей, встречаются редко.

1.2.3.3. Реляционная модель данных

- **Реляционная модель данных** – данные представлены посредством строк в таблицах.
- В теории баз данных эти таблицы называют **отношениями (relations)** – поэтому и базы данных называются **реляционными**. Отношение – это математический термин. При определении свойств таких отношений используется теория множеств. В терминах данной теории строки таблицы будут называться **кортежами (tuples)**, а колонки – **атрибутами**. Отношение имеет заголовок, который состоит из атрибутов, и тело, состоящее из кортежей. Количество атрибутов называется **степенью отношения**, а количество кортежей – **кардинальным числом**.
- В распоряжении пользователя имеются операторы для выборки данных из таблиц, а также для вставки новых данных, обновления и удаления имеющихся данных.
- Реляционные системы используют *декларативный* подход к получению данных для обработки. Это означает, что требуется только указать, какие данные нужны, но не нужно предписывать способ их получения.
- Реляционная модель требует, чтобы данные воспринимались в виде таблиц. Однако это требование относится только к логической структуре базы данных, т. е. только к внешнему и концептуальному уровням архитектуры ANSI/SPARC. На физическом уровне могут использоваться какие угодно структуры хранения данных и механизмы доступа.

1.2.3.3. Реляционная модель данных (продолжение)

Таблица «Студенты»

Номер зачетной книжки	Ф. И. О.	Серия паспорта	Номер паспорта
55500	Иванов Иван Петрович	0402	645327
55800	Климов Андрей Иванович	0402	673211
55865	Новиков Николай Юрьевич	0202	554390

Таблица «Успеваемость»

Номер зачетной книжки	Предмет	Учебный год	Семестр	Оценка
55500	Физика	2017/2018	1	5
55500	Математика	2017/2018	1	4
55800	Физика	2017/2018	1	4
55800	Физика	2017/2018	2	5

1.2.3.3. Реляционная модель данных (продолжение)

Основные понятия реляционной модели: ограничения

При работе с базами данных часто приходится следовать различным **ограничениям**, которые могут быть обусловлены спецификой конкретной предметной области:

- номер зачетной книжки состоит из пяти цифр и не может быть отрицательным
- серия документа, удостоверяющего личность, является четырехзначным числом, а номер документа, удостоверяющего личность — шестизначным числом
- номер семестра может принимать только два значения — 1 (осенний семестр) и 2 (весенний семестр)
- оценка может принимать только три значения — 3 (удовлетворительно), 4 (хорошо) и 5 (отлично)

1.2.3.3. Реляционная модель данных (продолжение)

Основные понятия реляционной модели: ключи

- Ключи служат для идентификации строк в таблицах и для связи таблиц между собой.
- Потенциальный ключ — это комбинация атрибутов таблицы, позволяющая уникальным образом идентифицировать строки в ней.
- Ключ может состоять и только лишь из одного атрибута таблицы.
  Например, в таблице «Студенты» таким идентификатором может быть атрибут «Номер зачетной книжки».
  В качестве потенциального ключа данной таблицы могут также служить два ее атрибута, взятые вместе: «Серия паспорта» и «Номер паспорта». Ни один из них в отдельности не может использоваться в качестве уникального идентификатора. В таком случае ключ будет составным.
- Потенциальный ключ должен быть не избыточным, т. е. никакое подмножество атрибутов, входящих в него, не должно обладать свойством уникальности.

1.2.3.3. Реляционная модель данных (продолжение)

Основные понятия реляционной модели: ключи (продолжение)

- При наличии в таблице более одного потенциального ключа один из них выбирается в качестве так называемого первичного ключа, а остальные будут являться альтернативными ключами.
- Предположим, что в таблице «Студенты» нет строки с номером зачетной книжки 55900, тогда включать строку с таким номером зачетной книжки в таблицу «Успеваемость» не имеет смысла. Таким образом, значения столбца «Номер зачетной книжки» в таблице «Успеваемость» должны быть согласованы со значениями такого же столбца в таблице «Студенты».
- Атрибут «Номер зачетной книжки» в таблице «Успеваемость» является примером того, что называется внешним ключом. Таблица, содержащая внешний ключ, называется ссылающейся таблицей (referencing table). Таблица, содержащая соответствующий потенциальный ключ, называется ссылочной (целевой) таблицей (referenced table). В таких случаях говорят, что внешний ключ ссылается на потенциальный ключ в ссылочной таблице.
- Внешний ключ может быть составным, т. е. может включать более одного атрибута. Внешний ключ не обязан быть уникальным.

1.2.3.3. Реляционная модель данных (продолжение)

Основные понятия реляционной модели: ссылочная целостность

- Проблема обеспечения того, чтобы база данных не содержала неверных значений внешних ключей, известна как проблема ссылочной целостности. Ограничение, согласно которому значения внешних ключей должны соответствовать значениям потенциальных ключей, называется ограничением ссылочной целостности (ссылочным ограничением).
- Обеспечением выполнения ограничений ссылочной целостности занимается СУБД, а от разработчика требуется лишь указать атрибуты, служащие в качестве внешних ключей.

- При проектировании баз данных часто предусматривается, что при удалении строки из ссылочной таблицы соответствующие строки из ссылающейся таблицы должны быть также удалены, а при изменении значения столбца, на который ссылается внешний ключ, должны быть изменены значения внешнего ключа в ссылающейся таблице. Этот подход называется каскадным удалением (обновлением).

1.2.3.4. Что такое язык SQL?

- Язык SQL — это непроцедурный язык, который является стандартным средством работы с данными во всех реляционных СУБД.
- Операторы (команды), написанные на этом языке, лишь указывают СУБД, какой результат должен быть получен, но не описывают процедуру получения этого результата. СУБД сама определяет способ выполнения команды пользователя.
- В языке SQL традиционно выделяются группа операторов определения данных (Data Definition Language — DDL), группа операторов манипулирования данными (Data Manipulation Language — DML) и группа операторов, управляющих привилегиями доступа к объектам базы данных (Data Control Language — DCL).
- К операторам DDL относятся команды для создания, изменения и удаления таблиц, представлений и других объектов базы данных.
- К операторам DML относятся команды для выборки строк из таблиц, вставки строк в таблицы, обновления и удаления строк.

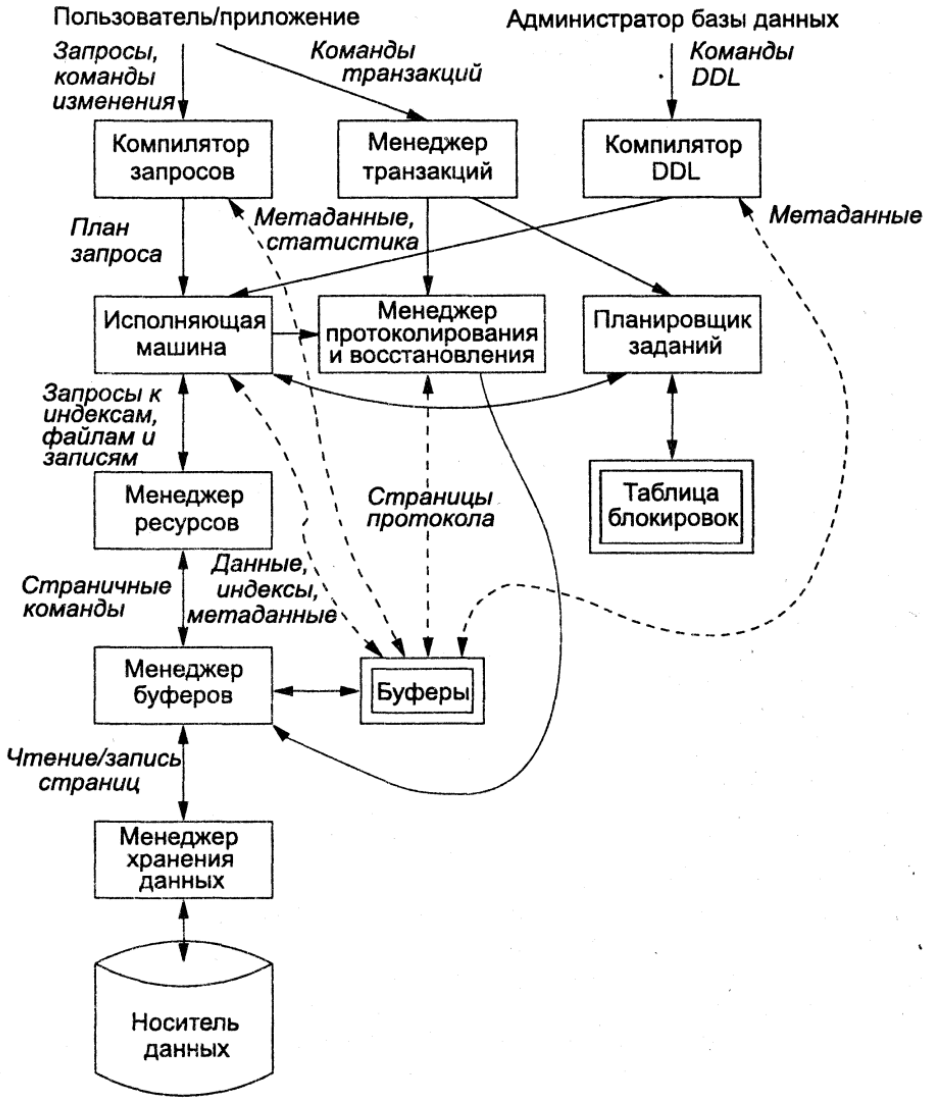
1.2.4. Функции системы управления базой данных

- Хранение, извлечение и обновление данных
- Поддержание словаря данных (системного каталога)
  - имена и типы элементов данных
  - имена таблиц
  - ограничения целостности, накладываемые на данные
  - имена авторизованных пользователей СУБД и их права доступа к объектам базы данных
  - статистика использования объектов базы данных
- Поддержка транзакций
  - Транзакция — одно из важнейших понятий теории баз данных.
  - Она означает набор операций над базой данных, рассматриваемых как единая и неделимая единица работы, выполняемая полностью или не выполняемая вовсе, если произошел какой-то сбой в процессе выполнения транзакции.
  - Таким образом, транзакции являются средством обеспечения согласованности данных.
  - В нашей простой базе данных транзакцией могут быть, например, две операции: удаление строки из таблицы «Студенты» и удаление связанных по внешнему ключу строк из таблицы «Успеваемость».

1.2.4. Функции системы управления базой данных (продолжение)

- Поддержка параллельности (concurrency) в использовании базы данных
- Восстановление данных после сбоев
- Авторизация пользователей
- Обеспечение доступа к базе данных с компьютеров локальной или глобальной сети
- Защита и поддержка целостности данных
- Реализация принципа независимости от данных
  - Приложения не должны зависеть от фактической структуры базы данных
- Обслуживание базы данных (с помощью различных утилит)
- Оптимизация и выполнение запросов к базе данных

1.2.5. Компоненты СУБД



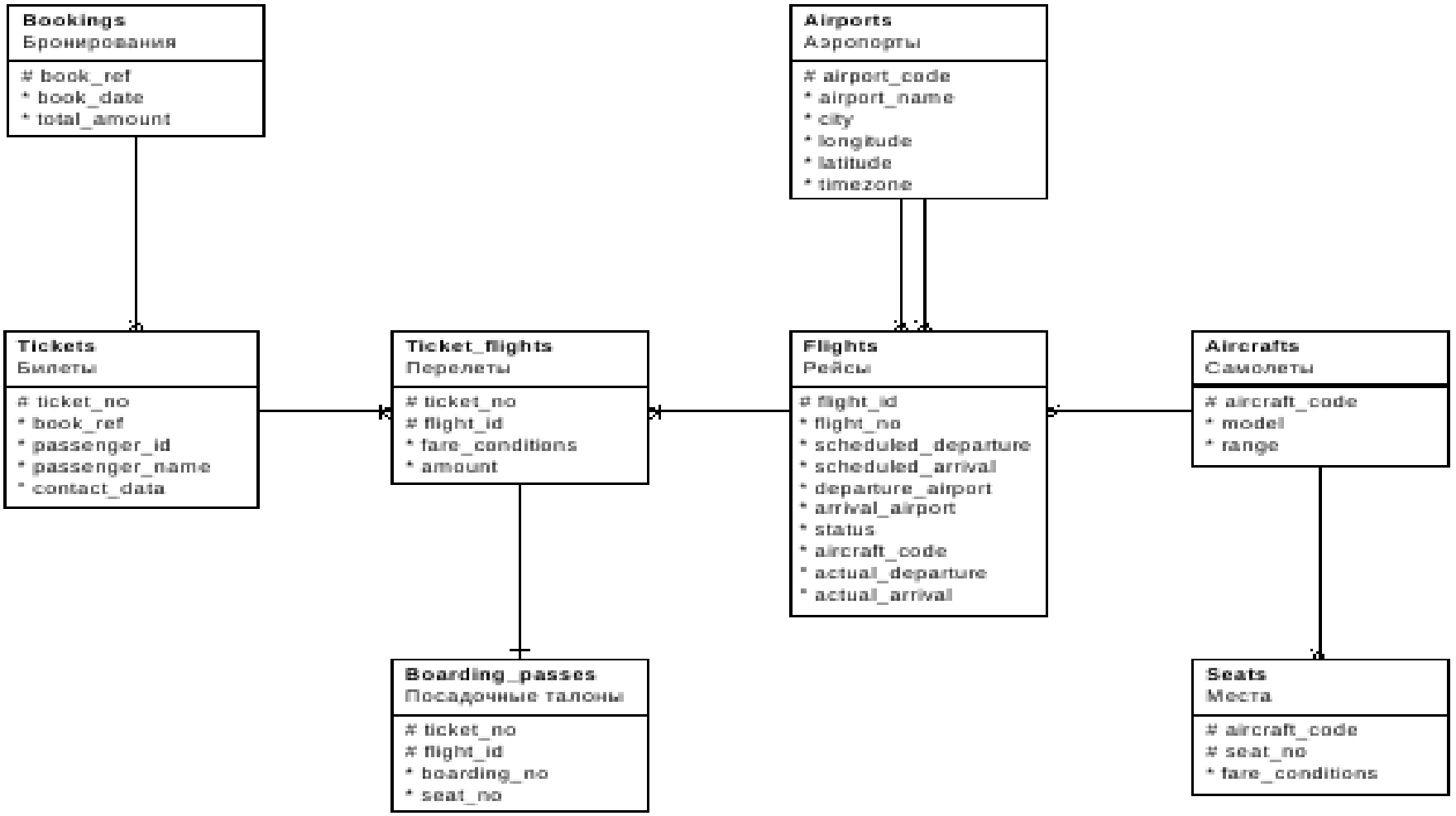
1.3. Описание предметной области

- В качестве предметной области выберем пассажирские авиаперевозки. Ее оригинальное описание и описание базы данных «Авиаперевозки» можно найти по адресам https://postgrespro.ru/education/demodb и https://postgrespro.ru/docs/postgrespro/current/demodb-bookings.html.
- Компания имеет парк самолетов. Компоновки салонов у самолетов одной модели одинаковые. Каждый аэропорт имеет трехбуквенный код, название аэропорта не всегда совпадает с названием города.
- Формируются маршруты перелетов между городами. Конечно, каждый такой маршрут требует указания не только города, но и аэропорта, поскольку в городе может быть и более одного аэропорта
- На основе перечня маршрутов формируется расписание полетов (или рейсов). В расписании указывается плановое время отправления и плановое время прибытия, а также тип самолета, выполняющего этот рейс.
- При фактическом выполнении рейса возникает необходимость в учете дополнительных сведений, а именно: фактического времени отправления и фактического времени прибытия, а также статуса рейса: Scheduled (можно бронировать), On Time (идет регистрация), Delayed (задержан), Departed (вылетел), Arrived (прибыл), Cancelled (отменен).

1.3. Описание предметной области (продолжение)

- Полет начинается с бронирования электронного авиабилета. Каждый такой билет имеет уникальный номер, состоящий из 13 цифр. В рамках одной процедуры бронирования может быть оформлено несколько билетов на различных пассажиров, но каждая такая процедура имеет уникальный шестизначный номер (шифр) бронирования, состоящий из заглавных букв латинского алфавита и цифр. Для каждой процедуры бронирования записывается дата бронирования и рассчитывается общая стоимость оформленных билетов.
- В каждый билет, кроме его тринадцатизначного номера, записывается идентификатор пассажира, а также его имя и фамилия (в латинской транскрипции) и контактные данные. В качестве идентификатора пассажира используется номер документа, удостоверяющего личность.
- В каждый электронный билет может быть вписано более одного перелета, например: Красноярск — Москва, Москва — Анапа, Анапа — Москва, Москва — Красноярск. Для каждого перелета указывается номер рейса, аэропорты отправления и назначения, время вылета и время прибытия, а также стоимость перелета. Кроме того, указывается и так называемый класс обслуживания: экономический, бизнес и др.
- Когда пассажир прибывает в аэропорт отправления и проходит регистрацию билета, оформляется так называемый посадочный талон. Этот талон связан с авиабилетом: в талоне указывается такой же номер, который имеет электронный авиабилет данного пассажира. Кроме того, в талоне указывается номер рейса и номер места в самолете. Указывается также и номер посадочного талона — последовательный номер, присваиваемый в процессе регистрации билетов на данный рейс.

1.3.1. Схема базы данных



Литература

1. Гарсиа-Молина, Г. Системы баз данных. Полный курс : пер. с англ. / Гектор Гарсиа-Молина, Джеффри Ульман, Дженнифер Уидом. – М. : Вильямс, 2003. – 1088 с.
2. Грофф, Дж. SQL. Полное руководство : пер. с англ. / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. – 3-е изд. – М. : Вильямс, 2015. – 960 с.
3. Дейт, К. Дж. Введение в системы баз данных : пер. с англ. / Крис Дж. Дейт. – 8-е изд. – М. : Вильямс, 2005. – 1328 с.
4. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика : пер. с англ. / Томас Коннолли, Каролин Бегг. – 3-е изд. – М. : Вильямс, 2003. – 1436 с.
5. Кузнецов, С. Д. Основы баз данных : учеб. пособие / С. Д. Кузнецов. – 2-е изд., испр. – М. : Интернет-Университет Информационных Технологий ; БИНОМ. Лаборатория знаний, 2007. – 484 с.
6. Лузанов, П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Лёвшин ; Postgres Professional. – М., 2017. – 146 с.
7. Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие. / Е. П. Моргунов ; под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с.
8. PostgreSQL [Электронный ресурс] : официальный сайт / The PostgreSQL Global Development Group. – http://www.postgresql.org.
9. Postgres Professional [Электронный ресурс] : российский производитель СУБД Postgres Pro : официальный сайт / Postgres Professional. – http://postgrespro.ru.

Задание

Для выполнения практических заданий необходимо использовать книгу:

Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие / Под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с. https://postgrespro.ru/education/books/sqlprimer

1. Прочитать введение и главу 1.
2. Установить ОС Linux (Debian или другой). Указания по установке СУБД PostgreSQL приведены в главе 2, параграф 2.1. Можно воспользоваться виртуальной машиной VirtualBox или аналогичной. Можно использовать уже настроенную ОС Debian (в виде виртуальной машины), полученную у преподавателя.
3. Развернуть учебную базу данных «Авиаперевозки» https://postgrespro.ru/education/demodb (см. главу 2, параграф 2.3). Использовать версию БД от 13.10.2016.
4. Изучить материал главы 3. Запросы к базе данных выполнять с помощью утилиты psql, описанной в главе 2, параграф 2.2.