Содержание

1 Лекция 1. Введение в реляционные базы данных

1.1 Понятие базы данных и СУБД

Definition 1.1. Данные

Данные – факты, текст, графики, картинки, звуки, аналоговые или цифровые видеосегменты, представленные в форме, пригодной для хранения, передачи и обработки

Notation 1.1. Задачи технологий работы с данными

- Загрузить
  - Получить
  - Передать между системами
  - Собрать в одном месте
- Сохранить
  - Обеспечить эффективное безопасное хранение
  - Предоставить доступ
  - Обеспечить быстрый поиск
- Обработать
  - Объединить в одной структуре
  - Рассчитать показатели
  - Обработать модель
- Принять решение
  - Отчеты
  - Дашборды
  - Визуализация
  - Предписание в операционном процессе

Notation 1.2. Что должны обеспечивать системы обработки данных?

- Поддержку необходимых требований к данными
  - Сохранность и достоверность данных в соответствии со спецификой предметной области
  - Защиту от искажения, уничтожения данных и несанкционированного доступа
  - Простоту и легкость использования данных
  - Требуемую скорость доступа к данным
- Независимость прикладных программ от данных
  - Возможность использования одних и тех же данных различными приложениями
  - Изменение логического представления данных (добавление/удаление новых элементов данных) не должно приводить к изменению прикладных программ обработки
  - Изменение параметров физической организации данных (характеристик носителя, длины блока, и др.) или методов доступа к ним не должны приводить к изменению прикладных программ

Definition 1.2. Независимость (от) данных

- **Логическая**
  Изменение логического представления данных (добавление/удаление новых элементов данных, разделение на несколько логических сегментов, изменение порядка хранения) не должно приводить к изменению прикладных программ обработки
- **Физическая**
  Изменение параметров физической организации данных (характеристик носителя, длины блока, и др.) или методов доступа к ним не должны приводить к изменению логической структуры данных или прикладных программ их обработки

Notation 1.3. Концепции обработки данных

- **Файловая обработка данных**
  До середины 60-х годов прошлого века – основная концепция построения программного обеспечения обработки данных
- **Базы данных**
  Независимость прикладных программ от данных
- **Объектно-ориентированные базы данных**
  Независимость прикладных программ от данных вместе с процедурами их обработки (объектно-ориентированный подход в программировании)

Remark 1.1. Файловая обработка данных

- Совмещение логического и физического представления данных
  - Физически данные ИС размещаются в файлах операционной системы
  - Прикладные программы работают напрямую с файлами
  - Структура файла зависит от требований конкретной прикладной программы, с ним работающей, и определяется разработчиком приложения
- Зависимость программ обработки от организации данных
  - Возможная избыточность и противоречивость данных
  - Для каждой задачи – свой алгоритм получения данных
- Разграничение доступа и защита данных – на уровне средств ОС

Она не обеспечивает:

- Поддержание логически согласованного набора файлов
- Единого языка манипулирования данными
- Восстановление информации после разного рода сбоев
- Реально параллельной работы нескольких пользователей

Remark 1.2. Базы данных

- Разделение логического и физического представления данных
  - Физически данные как правило размещаются в файлах операционной системы
  - Разрабатывается определённая логическая структура данных, с которой «общаются» прикладные программы
  - Логическая структура данных основана на физической, но не меняется при изменении последней
- Независимость программ обработки от организации данных
  - Сокращение избыточности и противоречивости данных
  - Единые языки для получения и изменения данных
  - Дополнительные средства разграничения доступа и защиты данных, учитывающие их логическую структуру

Remark 1.3. Объектно-ориентированные БД

- Разделение логического и физического представления данных
  - Физические данные как правило размещаются в файлах операционной системы
  - На логическом уровне данные представляют собой объекты
    - Данные $\rightarrow$ свойства объекта (статическая часть)
    - Обработка данных $\rightarrow$ методы объекта (динамическая часть)
- Независимость прикладных программ от объектов
- Концепция повторного использования программного кода
  - Разные прикладные программы используют унифицированные методы обработки одних и тех же данных

1.2 Базы данных и СУБД

Definition 1.3. База данных

База данных – логически структурированная совокупность постоянно хранимых в памяти компьютера данных, характеризующих актуальное состояние некоторой предметной области и используемых прикладными программными системами какого-либо предприятия
База данных (БД) – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных (ГОСТ)

Notation 1.4. Основные характеристики БД

- Компьютерная система – БД хранится и обрабатываемя в вычислительной системе
- Содержит структурированную информацию – данные в БД логически структурированы (систематизированы) с целью обеспечения возможности их эффективного поиска и обработки в вычислительной системе
  - Структурированность БД оценивается не на уровне физического хранения, а на уровне некоторой логической модели данных
- Поддерживает определенный набор операций над данными
  - Структурированность определяет семантику и допустимые операции

Definition 1.4. Система баз данных (банк данных)

База данных является составной частью системы баз данных
Система базы данных – компьютеризированная система обработки данных, хранящихся в БД
Основные функции системы баз данных:

- Добавление новых структур данных (таблиц) в базу данных
- Изменение существующих структур данных (таблиц)
- Добавление новых элементов данных (записей в таблицы)
- Выборка необходимых элементов данных (записей из таблиц)
- Обновление элементов данных (записей в таблицах)
- Удаление элементов данных (записей из таблиц)
- Удаление структур данных (таблиц) из базы данных

Definition 1.5. Системы управления базами данных

СУБД – совокупность программных средств, предназначенная для модификации и извлечения из БД необходимых пользователю (прикладной программе) данных, а также для создания БД, поддержания их в работоспособном состоянии, обеспечения безопасности БД и решения других задач администрирования
Цель: Обеспечивать совместное безопасное использование данных различными приложениями и пользователями
Основные функции СУБД:

- Обеспечение физической и логической независимости данных
- Поддержка связи между логической и физической организацией данных
- Предоставление интерфейса доступа к данным пользователям и прикладным программам
- Обеспечение целостности и непротиворечивости данных при совместной работе нескольких пользователей
- Предоставление механизмов защиты данных
- Предоставление механизмов восстановления данных

Notation 1.5. Обеспечение логической и физической независимости данных

СУБД обеспечивает необходимые структуры внешней памяти как для хранения пользовательских данных БД, так и для служебных целей
СУБД скрывает от пользователей как используется файловая система и как организованы файлы
Для связи логической и физической структур данных СУБД использует служебную (мета) информация, хранящуюся в словаре данных

Notation 1.6. Управление буферами оперативной памяти

Для увеличения скорости обработки данных используется буферизация данных в оперативной памяти
СУБД поддерживает собственный набор буферов оперативной памяти с собственным алгоритмом замены буферов

Notation 1.7. Предоставление интерфейса доступа к данным

Для работы с базами данных СУБД предоставляет поддержку специальных языков, называемых языками баз данных

- Язык описания данных (DDL) – позволяет создавать и изменять структуру объектов базы данных
- Язык манипулирования данными (DML) – позволяет заносить данные в БД, удалять, модифицировать или выбирать существующие данные

Использование данных языков позволяет пользователям сохранять, извлекать и обновлять данные в БД без необходимости понимания физической реализации системы

Notation 1.8. Обеспечение целостности

Целостность БД – свойство БД, означающее, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область информация
Целостность данных предполагает:

- Отсутствие неточно введенных данных или двух одинаковых записей об одном и том же факте
- Защиту от ошибок при обновлении данных в БД
- Невозможность удаления (или каскадное удаление) записей, которые связаны с другими записями
- Неискажение данных при работе в многопользовательском режиме
- Сохранность данных при сбоях техники (восстановление данных)

СУБД должна обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам

Notation 1.9. Управление параллельной работой пользователей

Для управления параллельной работой пользователей и поддержки целостности данных в СУБД реализован механизм изоляции транзакций

Definition 1.6. Транзакция

Транзакция – некоторая неделимая последовательность операций над данными БД, которая отслеживается СУБД от начала и до завершения
Если по каким-либо причинам (сбои и отказы оборудования, ошибки в программном обеспечении) транзакция остается незавершенной или при выполнении транзакции нарушается целостность данных, то она отменяется

Definition 1.7. Журнализация

СУБД должна иметь возможность восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя
Для восстановления БД нужно располагать некоторой дополнительной информацией, которая должна храниться особо надежно
Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД – журнала транзакций

Notation 1.10. Обеспечение безопасности БД

СУБД должна иметь механизм, гарантирующий возможность доступа к базе данных только санкционированных пользователей
Термин безопасность относится к защите БД от преднамеренного или случайного несанкционированного доступа
Защита данных от несанкционированного доступа может достигаться:

- Созданием ролей и ввдеением системы паролей
- Настройкой разрешений на доступ к данным и выполнение операций с данными
- Шифрованием соединения с прикладными программами
- Шифрованием данных

Notation 1.11. Выбор СУБД

Выбор СУБД – важный шаг при создании ИС, влияющий на эффективность, как проектирования, так и функционирования системы

Учитывая тенденции развития ИС, СУБД должна отвечать следующим требованиям:

- Обеспечивать работу в гетерогенной сетевой среде, включая возможность эффективной работы в Интернете
- Легко переноситься с платформы на платформу
- Обеспечивать работу с большими объемами разнотипных данных
- Быть надежной и эффективной

1.3 Принципы организации базы данных

Notation 1.12. Требования к БД и СУБД

- Высокое быстродействие (малое время отклика на запрос)
- Простота обновления данных
- Защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения
- Поддержка целостности данных
- Независимость данных – возможность изменения логической и физической структуры БД без изменения предсталвений пользователей
- Совместное использование данных многими пользователями
- Стандартизация построения и эксплуатации БД
- Адекватность отображения данных соответствующей предметной области

Notation 1.13. Принципы организации БД

- Разделение различных видов данных
  - Данные пользовательские (приложений)
  - Вспомогательные данные (индексы)
  - Метаданные (словарь данных БД)
  - Служебная информация
- Позволяет повысить безопасность данных и быстродействие работы с ними
- Проектирование логической структуры данных
  Выбор модели данных (в зависимости от требований доступа к данным)
- Позволяет обеспечить:
  - Логическую и физическую независимость данных
  - Адекватность отображения данных соответствующей предметной области
  - Удобство работы и гибкость
- Определение ограничений данных
  - Уменьшение ошибок ввода/изменения данных
  - Зависимость от семантики данных
- Позволяет поддержать целостность данных

1.4 Реляционная модель данных

Definition 1.8. Модель данных

Модель данных – формальное описание представления и обработки данных в системе управления базами данных

Notation 1.14. Компоненты реляционной модели данных

- Структурный аспект
  - Какие структуры данных рассматриваются реляционной моделью
  - Постулируется, что единственной структурой данных, используемой в реляционной модели, являются нормализованные n-арные отношения
- Аспект целостности
  - Ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных
  - Целостность сущностей реального мира
  - Ссылочная целостность
- Аспект обработки
  - Способы манипулирования реляционными данными
  - Реляционная алгебра – базируется на теории множеств
  - Реляционное исчисление – базируется на логическом аппарате исчисления предикатов первого порядка

1.5 Структурный аспект реляционной модели

Definition 1.9. Отношение

Отношение (relation) – класс объектов реального мира, каждый из которых должен быть уникально идентифицирован

- Подмножество R декартового произведения множеств элементов доменов: (не обязательно различных)
- Или множество кортежей, соответствующих одной схеме отношения
- Мощность отношения – количество кортежей в отношении
- На бытовом уровне – тело таблицы

Definition 1.10. Схема отношения

- Именованное конечное множество пар вида {Имя_атрибута: Имя_домена}
- Степень или -арность (схемы) отношения – количество атрибутов
- На бытовом уровне – заголовок таблицы

Definition 1.11. Атрибут

Атрибут (поле, столбец) – характеристика объекта, принимающая значения определенного типа данных

- Подмножество домена (атрибут A_n определен на домене D_n, который содержит множество возможных значений атрибута)
  $A_1(D_1), A_2(D_2), \dots A_n(D_n)$
- На бытовом уровне – столбец таблицы
- Имена атрибутов должны быть уникальны в пределах отношения

Definition 1.12. Тип данных

Тип данных – реляционная модель данных допускает использование только простых типов

- Соответствует понятию типа данных в языках программирования
- Реляционная модель данных допускает использование только простых типов (логические, символьные, числовые ...), т.к. в реляционных операциях не должна учитываться внутренняя структура данных

Definition 1.13. Домен

Домен – произвольное логическое выражение (опционально), определяющее набор допустимых значений атрибута

- Базовый тип данных + произвольное логическое выражение (опционально)
- Если вычисление заданного логического выражения для элемента данных заданного типа дает результат "истина" , то он является элементом домена
- Домены ограничивают сравнения: некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип данных (например, возраст сотрудника и количество его детей)

Definition 1.14. Кортеж

Кортеж (запись, строка) – набор связанных значений атрибутов, относящихся к одному объекту (сущности)

- Множества пар вида { Имя_атрибута: Значение_атрибута }
- На бытовом уровне – строка таблицы
- Значение атрибута должно быть в пределах заданного домена
- Каждый кортеж отношения должен быть уникально идентифицирован

Definition 1.15. Схема БД

Схема БД (в структурном смысле) – набор именованных схем отношений

Definition 1.16. Реляционная база данных

Реляционная база данных – набор отношений, имена которых совпадают с именами схем отношений в схеме БД

Notation 1.15. Свойства отношений

- В отношении нет кортежей-дубликатов
  Т.к. отношение – это множество кортежей, а каждое множество (в классической теории множеств) состоит из различных элементов
  Во многих РСУБД может нарушаться для отношений, являющихся результатами запросов
- Кортежи не упорядочены
  Множество не упорядочено
- Атрибуты не упорядочены
  Каждый атрибут имеет уникальное имя в пределах отношения, поэтому порядок атрибутов не имеет значения
- Значение атрибута должно быть атомарным (неразделяемым на несколько значений)
  В современных РСУБД в ячейки таблиц можно поместить что угодно – массивы, структуры и даже другие таблицы
- Домены ограничивают сравнения
  Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип данных

1.6 Аспект целостности реляционной модели

Definition 1.17. Целостность базы данных

Целостность базы данных (database integrity) – соответствие имеющейся в базе данных информации ее внутренней логике, структуре и всем явно заданным правилам
Каждое такое правило, налагающее ограничение на возможное состояние базы данных, называется ограничением целостности (integrity constraint)
Задача аналитика и проектировщика БД – возможно более полно выявить все имеющиеся ограничения целостности и задать их в базе данных
СУБД может (и должна) контролировать целостность БД

Notation 1.16. Сущностная целостность

Каждый кортеж отношения должен быть уникально идентфиицирован по значениям его атрибутов
Потенциальный ключ обладает следующими свойствами:

- Свойством уникальности – в отношении не моежт быть двух различных кортежей с одинаковым значением потенциального ключа
- Свойством неизбыточности – никакое подмножество в потенциальном ключе не обладает свойством уникальности, т.к. если из потенциального ключа убрать любой атрибут, он утратит свойство уникальности

Отношение может иметь несколько потенциальных ключей:

- Один из потенциальных ключей объявляется первичным – Primary Key, а остальные – альтернативными – Alternate Key
- С точки зрения реляционной модели данных, нет оснований выделять таким образом один из потенциальных ключей

Notation 1.17. Правила для поддержки сущностной целостности

Потенциальный ключ может быть:

- Простым – состоит из одного атрибута
- Составным – состоит из нескольких атрибутов

Потенциальные ключи фактически являются идентификаторами. Если бы идентификаторы могли содержать NULL значения, невозможно было бы дать ответ "да" или "нет" на вопрос, совпдаают ли два идентификатора
Это определяет следующее правило: значения атрибутов, входящих в состав некоторого потенциального ключа не могут быть NULL (во многих СУБД выполняется только для первичного ключа)

Definition 1.18. NULL

Для того чтобы обойти проблему неполных или неизвестных данных, каждый тип данных в БД может быть дополнен NULL
NULL – это не значение, а некий маркер, показывающий, что значение неизвестно
В ситуации, когда возможно появление неизвестных или неполных данных, разработчик имеет на выбор два варианта:

- Ограничиться использованием обычных типов данных и не использовать NULL, а вместо неизвестных данных вводить либо нулевые значения, либо значения специального вида
- Использовать NULL вместо неизвестных данных

Наличие NULL приводит к использованию трехзначной логики:

- Три возможных значений выражений: TRUE (T), FALSE (F), UNKNOWN (U)
- NULL = NULL → U (NOT NULL = NULL → U)
- NULL !+ NULL → U (NOT NULL != NULL → U)
- F OR NULL → U (T OR NULL → T)
- T AND NULL → U (F AND NULL → F)

Definition 1.19. Внешние ключи

Различные объекты предметной области, информация о которых хранится в базе данных, взаимосвязаны друг с другом
Для реализации взаимосвязи между родительским и дочерним отношениями в реляционных БД используются внешние ключи – foreign key (FK)

Notation 1.18. Требования к FK

Подмножество атрибутов FK отношения R будем называть внешним ключом, если

- Существует отношения S (R и S не обязательно различны) с потенциальным ключом K
- Каждое значение FK в отношении R всегда совпадает со значением K для некоторого кортежа из S, либо является NULL

Remark 1.4. Замечания относительно FK

- Отноешние S называется родиетльским отношением, отношение R называется дочерним отношением
- FK, также как и потенциальный, может быть простым и Составным
- FK должен быть определен на тех же доменах, что и соответствующих потенциальный ключ родительского отношения
- FK, как правило, не обладает свойством уникальности (тип связи – один ко многим)
- Для FK не требуется, чтобы он был компонентом некоторого потенциального ключа
- NULL для значений атрибутов FK допустимы только в том случае, когда атрибуты FK не входят в состав никакого потенциального ключа

Definition 1.20. Связь

Связь – ассоциирование двух или более сущностей (или копий одной и той же сущности)
Одно из основным требований к организации БД – это обеспечение возможности поиска одних сущностей по значениям других, для чего необходимо установить между ними определенные связи
Типы связей:

- Связь 1:1. Один экземпляр сущности одного класса связан с одним экземпляром сущности другого класса
- Связь 1:M. Один экземпляр сущности одного класса связан со многими экземплярами сущности другого класса
- Связь M:N. Несколько экземпляров сущности одного класса связаны с несколькими экземплярами сущности другого класса

Notation 1.19. Допустимая кратность связей

В реляционных БД допустимыми являются связи типа 1:M и 1:1 (значения внешнего ключа – уникальны)
Механизм реализации допустимых взаимосвязей состоит в том, что на дочернее отношение добавляются атрибуты, являющиеся ссылками на ключевые атрибуты родительского отношения
Невозможно ссылаться на несуществующие объекты $\rightarrow$ значения атрибута внешнего ключа дочернего отношения должны иметь соответствие среди значений атрибутов отношения потенциального ключа родительского
Взаимосвязи типа M:N реализуются использованием нескольких взаимосвязей типа 1:M

1.7 Знакомство с PostreSQL (реклама)

Definition 1.21. PostreSQL

PostreSQL – это открытая, BSD-лицензированная система управления объектно-ориентированными реляционными базами данных

Notation 1.20. История PostreSQL

- 1986 – старт проекта PostreSQL на факультете компьютерных наук Калифорнийсокго университета в Беркли
  Первоначальное название проекта – POSTGRES (развитие старой БД Ingres)
- 1996 – проект POSTGRES переименован в PostreSQL, для отражения поддержки SQL
  Global Development Group PostreSQL, специализированное сообщество участников, продолжает выпускать релизы проекта с открытым исходным кодом
- Первоначально PostreSQL был разарботан для работы на UNIX-подобных платформах
  Сейчас PostreSQL поддерживает различны платформы, такие как Windows, macOS и Solaris

Notation 1.21. Преимущества и особенности СУБД PostreSQL

- Надежность
- Производительность
- Расширяемость
- Поддержка SQL
- Поддержка многочисленных типов данных

Notation 1.22. Расширяемость

PostreSQL спроектирован с рассчетом на расширяемость
Прикладные программисты могут:

- Создавать собственные типы данных на основе уже имеющихся (составные типы, диапазоны, масисвы, перечисления)
- Писать хранимые процедуры и функции для обработки данных в БД (в том числе триггеры)
- Писать расширения (языке программирования Си), которые добавляют необходимый функционал и, обычно, могут подключаться даже к работающему серверу

Если вам не нравится какая-либо часть системы, вы всегда можете разработать собственный плагин

Remark 1.5. Важный факт

Подавляющее большинство СУБД:

- Представляет собой сервис (демон в *nix-системах), который взаимодействует с внешним миром по специальным протоколам (чаще всего, построенным поверх TCP/IP)
- Не имеет никакого человеческого интерфейса
- Общение осуществляется на специализированном языке через специальные библиотеки

MySQL Workbench, Microsoft SQL Serber Management Studio, Oracle SQL Developer и им подобные – это не СУБД, это лишь клиентское программное обеспечение, позволяющее нам взаимодействовать с СУБД

Notation 1.23. Упрощенная архитектура PostreSQL

PostreSQL – это СУБД клиент-серверного типа с многопроцессной архитектурой, работающая на одном хосте

Сбой в одном из процессов не повлияет на остальные и система продолжит функционировать

Набор нескольких процессов, совместно управляющих одним кластером БД, называется "сервером PostreSQL"

Один сервер PostreSQL может управлять несколькими конкурентными клиенсткими подключениями

Notation 1.24. Основные процессы

- FrontEnd процессы – клиентские приложения:
  - Используют PostreSQL в качестве менеджера баз данных
  - Соединение может происходить через TCP/IP или локальные сокеты
- Демон postres (postmaster) – это основной процесс PostreSQL:
  - Прослушивание через порт/сокет входящих клиенстких подключений
  - Создание BackEnd процессов и выделение им ресурсов
- BackEnd процессы:
  - Аутентификация клиенстких подключений
  - Управление запросами и отправка результатов клиенстким приложениям
  - Выполнение внутренних задач (служебные процессы)

Notation 1.25. Процесс взаимодействия с БД

1. Клиент (FrontEnd)
   (a) Используя язык БД формулирует требования к результату
   (b) Опирается на знание логической структуры БД
2. СУБД (postmaster и BackEnd)
   (a) Получает запрос от клиента
   (b) Анализирует разрешения
   (c) Анализирует и оптимизирует запрос
   (d) Создает план выполнения запроса, опираясь на знание физической структуры данных
   (e) Выполняет запрос

Notation 1.26. Взаимодействие с БД

Для работы с реляционной СУБД существует два основных подхода:

- Работа с библиотекой, которая соответствует конкретной СУБД и позволяет использовать для работы с БД язык БД
- Работа с ORM, которая использует объектно-ориентированный подход для работы с БД и автоматически генерирует код на языке БД

Notation 1.27. Подключение к СУБД с использованием клиентской библиотеки

Строка подключения включает:

- Имя сервера БД (или IP адрес и порт)
- Имя базы данных
- Учетную запись пользователям
- И другие параметры, необходимые для установки исходного подключения

1.8 SQL и NoSQL

Notation 1.28. Модели данных

- Иерархическая (файловая система)
- Сетевая (социальные сети)
- Документ-ориентированная (системах управления содержимым)
- Реляционная (банковские системы)
- Объектно-ориентированная (естественное отображение ООП кода на БД, уменьшающее impedance mismatch)
- Многомерная (аналитические системы)

Notation 1.29. Преимущества РБД

- Совместное использование данных
  - Улучшенное управление паралелльной работой
  - Повышенная безопасность
  - Контроль доступа к данным
- Поддержка целостности данных
  - Контроль за избыточностью данных и их непротиворечивостью
  - Обеспечение поддержки бизнес-правил
- Эффективное управление
  - Упрощение сопровождения системы за счет неазависимости от данных
  - Эффективное резервное копирование и восстановление данных
- Применение стандартов

Notation 1.30. Недостатки РБД

- Сложность
  Затраты на преобразование данных на входе и выходе
- Уязвимость
  Централизация ресурсов повышает уязвимость системы
- Высокие финансовые затраты
  - Стоимость СУБД
  - Стоимость сопровождения
  - Дополнительные затраты на аппаратное обеспечение

Definition 1.22. Базы данных NoSQL

Базы данных NoSQL хорошо подходят для приложений, которые должны быстро, с низкой временной задержкой (low latency) обрабатывать большой объем данных с разной структурой

- Гибкость. Благодаря использованию гибких моделей данных БД NoSQL хорошо подходят для частично структурированных и неструктурированных данных. Эффективность работы с разреженными данными
- Масштабируемость. БД NoSQL рассчитаны на масштабирование с использованием распределенных кластеров аппаратного обеспечения. Широко используются в облачных решениях в качестве полностью управляемых сервисов
- Высокая производительность. БД NoSQL оптмизированы для конкретных моделей данных и шаблонов доступа, что позволяет достичь более высокой производительности по сравнению с реляционными базами данных
- Широкие функциональные возможности. БД NoSQL предоставляют API и типы данных с широкой функциональностью, которые специально разработаны для соответствующих моделей данных

Definition 1.23. Нереляционные БД

- Хранилища ключей и значений
  - Поддерживают высокую разделяемость и обеспечивают беспрецендентное горизонтальное масштабирование
  - Игровые, рекламные прилоежния и приложения IoT (Amazon DynamoDB, Redis, Riak)
- Колоночные
  - Данные хранятся не по строкам, а по столбцам
  - Хорошо подходят для BigData (Hbase, Clickhouse, Vertica)
- Документоориентированные
  - Хранение коллекций документов с произвольным набором атрибутов (полей)
  - Каталоги, пользовательские профили и системы управления контентом, где каждый документ уникален и изменяется со временем (CouchDB, Couchbase, MongoDB)
- Графовые
  - Упор на установление произвольных связей между данными
  - Социальные сети, сервисы рекомендаций, системы выявления мошенничества и графы знаний (OrientDB, Neo4j)

Параметр	Реляционные (SQL)	NoSQL
Подходящие рабочие нагрузки	OLTP и OLAP	Приложения с низкой задержкой доступа к данным
Модель данных	Нормализованная реляционная модель обеспечивает целостность ссылочных данных в отношениях между таблицами	Предоставляют разнообразные модели данных, оптимизированные для высокой производительности и масштабируемости
Струткура	Жесткая схема: таблицы (строки, столбцы)	Гибкие модели: документы, ключ-значение, графы, колоночные
Изменение схемы	Требует ALTER TABLE, миграции	Гибкость (документы без фиксированной схемы)
Типы данных	Строгая типизация	Диинамические (JSON, BLOB и другие)
Связи	JOIN, внешние ключи, ACID-транзакции	Часто денормализация, ссылки или вложенные данные
Производительность	Зависит от дисковой подсистемы. Требуется оптимизация запросов, индексов и струткур таблицы	Зависит от размера кластера базового аппаратного обеспечения, задержки сети и вызывающего приложения
Чтение/запись	Быстрые сложные запросы (OLTP), но JOIN могут замедлять	Высокая скорость для простых операция (ключ-значение)
Оптимизация	Индексы, нормализация	Денормализация, распределенные вычисления
Масштабирование	Масштабируются путем увеличения вычислительных возможностей аппаратного обеспечения или добавления отдельных копий для рабочих нагрузок чтения	Поддерживают высокую разделяемость благодаря шаблонам доступа с возможностью масштабирования на основе распределенной архитектуры
Горизонтальное	Сложно (шардинг требует усилий)	Оптмиизировано (например, Cassandra, DynamoDB)
Вертикальное	Стандартный подход (увеличение сервера)	Возможно, но реже используется

1.9 Классификация баз данных

Notation 1.31. Классификация БД

- Классификация БД по характеру организации данных
  - Неструктурированные
    БД, хранят данные в виде обычного текста или гипертекстовой разметки
    * рпоще зафиксировать (как есть)
    * Очень трудно искать конкретные данные, поскольку они не структурированы
    * Очень трудно анализировать, поскольку данных как правило качественные (семантические)
  - Структурированные
    БД, хранящие данные в организованном виде в отформатированном хранилище
    * Требуют предварительного проектирования и описания структуры БД
    * Только после этого БД такого типа могут быть заполнены данными
    * Очень простой поиск и нахождение данных в базе данных или наборе данных
    * Очень легко анализировать данные, поскольку они как правило количественные
- Классификация БД по характеру хранимой информации
  - Документальные
    * Предназначены для хранения слабо структурированных данных. Единицей хранения является документ, заданный конечным (но не фиксированным) набором полей в общем случае произвольной длины. Значение поля может иметь сложную структуру и зависеть от контекста использования
    * Пользователю в ответ на его запрос выдается либо ссылка на документ, либо сам документ, в котором он может найти интересующую информацию
    * Использование: гипертекстовые документы в сети Интернет
    * Информационно-справочные или информационно-поисковые системы
  - Фактографические
    * Ориентированы на хранение хорошо структурированных данных. Единицей информации служит описание факта конечным, четко определенным множеством свойств. Каждое свойство факта (объекта) имеет атомарное значение, которое не зависит от контекста использования
    * Использование: БД оперативной обработки транзакций (OLTP) – операционные БД, БД оперативной аналитической обработки (OLAP) – хранилища данных (Data Warehouse)
- Классификация БД и СУБД по структуре организации данных
  Структурированные БД различаются по типу используемой модели представления данных
  - Сетевые
  - Иерархические
  - Реляционные
  - Многомерные
  - Объектно-ориентированные

Definition 1.24. Модель представления данных

Модель данных – интегрированный набор понятий для описания и обработки данных, связей между ними и ограничений, накладываемых на данные в рамках предметной области

Модель данных можно рассматривать как сочетание трех компонентов:

- Структурная часть – набор правил, по которым может быть построена БД
- Управляющая часть – определяет типы допустимых операций с данными: для обновления и извллечения данных, для изменения структуры данных
- Набор ограничений (необязательный) для поддержки целостности данных, гарантирующих корректность используемых данных (полноту, непротиворечивость и адекватное отражение предметной области)

Definition 1.25. Иерархическая модель данных

Иерархическая модель данных – это модель данных, где используется представление БД в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней – родителей-потомков

В иерархической модели узел может иметь только одного родителя

- Самый верхний узел называется корневым узлом
- Все узлы дерева, за исключением корневого, должны иметь родительский узел
- Связи между отдельными узлами дерева отражаются с помощью направленных ребер графа – от родителя к ребенку

Definition 1.26. Сетевая модель данных

Для описания сетевой модели данных используют понятия "запись" и "связь"

Связь определяется для двух записей: предка и потомка

В сетевой модели данных запись-потомок может иметь произвольное число записей-предков

В сетевой структуре каждый элемент может быть связан с любым другим элементом

Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию

Несмотря на то, что эта модель решает некоторые проблемы, связанные с иерархической моделью, выполнение простых запросов остается достаточно сложным процессом

Также, поскольку логика процедуры выборки данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения. Другими словаи, если необходимо изменить структуру данных, то нужно изменить и приложение

Definition 1.27. Реляционная модель данных

Эта модель данных основана на понятии математических отошений (relation)

В реляционной модели данные представлены в виде плоских таблиц, связанных между собой. Необходимо помнить, что таблица есть понятие нестрогое и часто означает не отношение как абстрактное понятие, а визуальное представление отношения на бумаге или экране. В частности – таблицы обычно предполагают упорядоченное хранение данных, в то время как отношения – не обладают этой характеристикой

Между отношениями поддерживаются связи один-к-одному или один-ко-многим

Definition 1.28. Многомерная модель данных

Данные представлены в виде многомерного куба (массива). Измерение (атрибут в реляционной модели) – размерность куба. Факт (агрегированная числовая характеристика) – содержимое ячейки
Агрегаты по всем срезам куба высчитываются один раз и хранятся в базе

Definition 1.29. Объектно-ориентированная модель данных

Данные представлены в виде классов и относящихся к ним объектов

- Класс – тип объекта
- Атрибут – свойство объекта
- Метод – операция над объектом

Инкапсуляция структурного и функционального описания объектов
Наследуемость внешних свойств объектов на основе соотношения "класс-подкласс"

1.10 Классификация СУБД

Definition 1.30. Словарь данных

Словарь данных – набор доступных для выборки всем пользователям базы данных системных таблиц, в которых хранятся метаданные (данные о данных)

Notation 1.32. Классификация СУБД

- Классификация по количеству пользователей
  - Однопользовательские
    - Реализуются на автономном ПК без использования сетей связи
    - Рассчитаны на работу одного пользователя или группы пользователей, разделяющих по времени одно рабочее место
    - Настольные или локальные СУБД
  - Многопользовательские
    - Ориентированы на коллективное использование информации
    - Строятся на базе локальной вычислительной сети
    - Могут быть распределены по нескольким узлам (хостам)
- Классификация по степени распределенности
  - Локальные СУБД – все части размещаются на одном компьютере
  - Распределенные СУБД – части СУБД могут размещаться не только на одном, но на двух и более компьютерах
    - Файл-серверные
    - Клиент-серверные
      - Двухзвенные (СУБД и БД, клиентские приложения)
      - Многозвенные (СУБД и БД, сервер приложений, клиентские приложения)
- Классификация по способу доступа к БД

Notation 1.33. Недостатки файл-серверной архитектуры

СУБД не располагает информацией о том, что происходит на компьютере где хранятся данные:

- Невозможно считать из БД только ту часть данных, которые запрашивает пользователь – считывается файл целиком (блокировка файла)
- Большой объем сетевого трафика (передача по сети множества блоков и файлов, необходимых приложению)
- Узкий спект операций манипулирования с данными, определяемый только файловыми командами
- Отсутствие адекватных средств безопасности доступа к данным (защита только на уровне файловой системы)
- Недостаточно развитый аппарат транзакций служит потенциальным источником ошибок в плане нарушения смысловой и ссылочной целостности информации при одновременном внесении изменений в одну и ту же запись

Definition 1.31. Архитектура клиент-сервер (двухзвенная)

На сервере: база данных, серверная часть СУБД – взаимодействует с БД, обеспечивая выполнение запросов клиентской части
На клиенте: прикладные программы, клиентская часть СУБД – обеспечивает взаимодействие с пользователем и формирование запросов к БД и передача их на сервер
Преимущества:

- СУБД располагает информацией о наборе файлов БД
- Обеспечение разграничения доступа к данным нескольких пользователей
- Считывание только необходимой пользователю информации из файла (блокировка блока данных)
- Обеспечивает корректную параллельную раоту всех пользователей с единой БД

Недостатки:

- Очень большая загрузка на сервер, так как он обслуживает множество клиентов и выполняет всю основную обработку данных
- Нагрузка с обработкой полученных данных дублируется на клиентские хосты

Definition 1.32. Архитектура клиент-сервер (трехзвенная)

Схема: тонкий клиент $\rightarrow$ сервер приложений $\rightarrow$ сервер базы данных
В функции клиентской части (тонкий клиент) входи только интерактивное взаимодействие с пользователем
Вся логика обработки данных (прикладные программы) вынесена на сервер приложений, который и обеспечивает формирование запросов к БД, передаваемых на выполнение серверу БД
Сервер приложений может являться специализированной программой или обычным web-сервером
Преимущества:

- Снижается нагрузка на сервер БД – он занимается исключительно функциями СУБД
- При изменении бизнес-логики нет необходимости изменять клиенсткие приложения
- Максимально снижаются требования к аппаратуре пользователей
- Данная модель обладает большей гибкостью, чем двухуровневые модели

Недостатки:

- Более высокие затраты ресурсов компьютеров на обмен информацией между компонентами приложений по сравнению с двухуровневыми моделями

Definition 1.33. Встраиваемые СУБД

Встаиваемая СУБД – поставляется как составная часть некоторого программного продукта, не требующая процедуры самостоятельной установки
Предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети
Физически чаще всего реализуется в виде подключаемой библиотеки
Доступ к данным со стороны приложения может происходить через язык запросов либо через специальные программные интерфейсы

1.11 Аспект обработки реляционной модели

Notation 1.34. Математические аппараты для манипулирования данными

Виды:

- Реляционная алгебра – основана на теории множеств. Описывает порядок выполнения операций, позволяющих из исходных выражений получить результат
- Реляционное исчисление – основано на логике предикатор первого порядка. Описывает результат в терминах исходных отношений

Свойство замкнутости операций на множестве отношений. Выражения реляционной алгебры и формулы реляционного исчисления определяются над отношениями реляционных БД и результатом вычисления также являются отношения

В современных РСУБД не используется в чистом виде ни реляционная алгебра, ни реляционное исчисление. Фактическим стандартом доступа к реляционным данным стал язык SQL (Structured Query Language), который представляет собой смесь операторов реляционной алгебры и выражений реляционного исчисления, использующий синтакси, близкий к фразам английского языка и расширенный лополнительными отсутствующими в упомянутых аппаратах

Definition 1.34. Реляционная алгебра (РА)

Реляционная алгебра – это формальный язык операций над отношениями (таблицами), включающий SQL – декларативный язык запросов, который включает не только операции РА, но и дополнительные конструкции (агрегацию, рекурсию, модификацию данных и др.)

Вывод: классическая реляционная алгебра и базовый SQL (без агрегации, рекурсии, оконных функци итд) эквивалентны по выразительной силе в рамках запросов к базе данных. Однако SQL строго мощнее, если учитывать все его возможности (например, рекурсивные запросы, агрегацию, модификацию данных)

Операции реляционной алгебры:

- Теоретико-множественные
  - Объединение отношений
  - Пересечение отношений
  - Вычитание отношений
  - Декартово произведение отношений
- Специальный
  - Выборка (ограничение) отношений
  - Проекция отношения
  - Соединение отношений
  - Деление отношения
- Дополнительные
  - Присваивание (сохранение результатов вычисления)
  - Переименование атрибутов отношения

Notation 1.35. Совместимость по типу

Некоторые реляционные операции требуют, чтобы отношения были совместимы по типу
Отношения являются совместимыми по типу, если их схемы идентичны
Отношения имеют одно и то же множество имен атрибутов, т.е. для любого атрибута в одном отношении найдется атрибут с таким же именем в другом отношении
Атрибуты с одинаковыми именами определены на одних и тех же доменах
Степени схем отношений (количество атрибутов) совпадают

Definition 1.35. Объединение отношений

Объединением двух совместимых по типу отношений A и B называется отношение S с той же схемой, что и у отношений A и B, и состоящее из кортежей, принадлежащих или A, или B, или обоим отношениями
Синтаксис: A UNION B или A $\cup$ B

Remark 1.6.

Объединение, как и любое отношение, не может содержать одинаковых кортежей
Если некоторый кортеж входит и в отношение A, и в отношение B, то в объединение он входит один раз

Definition 1.36. Вычитание отношений

Вычитанием двух совместимых по типу отношений A и B называется отношение S с той же схемой, что и у отношений A и B, и состоящее из кортежей, принадлежащих отношению A и не принадлежащих отношению B
Синтаксис: A EXCEPT B или A $\setminus$ B

Definition 1.37. Пересечение отношений

Пересечением двух совместимых по типу отношений A и B называется отношение S с той же схемой, что и у отношений A и B, и состоящее из кортежей, принадлежащих одновременно обоим отношениям
Синтаксис: A INTERSECT B или A $\cap$ B

Remark 1.7.

Пересечение может быть выражено через операцию вычитания: $A \cap B = A \setminus (A \setminus B)$

Definition 1.38. Декартово произведение отношений

Декартовым произведением двух отношений $A = (A_1, A_2, \dots A_n)$ и $B = (B_1, B_2 \dots B_m)$ называется отношение S, со схемой, состоящей из атрибутов отношений A и B: $(A_1, A_2 \dots A_n, B_1, B_2 \dots B_m)$ и являющееся результатом конкатенации (сцепления) каждого кортежа из отношения A с каждый кортежем из отношения B. В результате получаем набор кортежей $(a_1, a_2 \dots a_n, b_1, b_2 \dots b_m)$, таких, что $(a_1 \dots a_n) \in A$, а $(b_1 \dots b_m) \in B$
Синтаксис: A CROSS JOIN B или $A * B$

Remark 1.8.

Мощность произведения равен произведению мощностей отношений A и B
Если в отношениях A и B имеются атрибуты с одинаковыми наименованиями, то перед выполнением операции декартового произвдеения такие атрибуты необходимо переименовать

Definition 1.39. Выборка (ограничение) отношений

Выборкой (ограничением) называется подмножество кортежей отношения R, удовлетворяющих определенному условию (предикату)
Результат выборки – горизонтальный срез отношения по некоторому условию
Предикат – логическое выражение, в которое могут входить атрибуты отношения R и/или скалярные выражения
Синтаксис: R WHERE или σ R

Definition 1.40. Проекция отношения

Проекцией называется вертикальное подмножество кортежей отношения R, создаваемое посредством извлечения значений указанных атрибутов $A_1 \dots A_n$ отношения
Результат проекции – вертикальный срез отношения, в котором удалены все возникшие при этом дубликаты кортежей
Синтаксис: $R[A_1 \dots A_n]$ или $\prod_{A_1 \dots A_n} R$

Definition 1.41. Соединение отношений

Θ-соединение (тэта-соединение) – определяет отношение S, которое содержит кортежи из декартового произведения отношений A и B, удовлетворяющих предикату Θ
Синтаксис: (A JOIN B) ON Θ

Remark 1.9. Частные случаи

- Экви-соединение – предикат содержит только оператор равенства
- Естественное соединение – эквисоединение отношений A и B, выполненное по всем общим атрибутам, из результатов которого исключается по одному экземпляру каждого общего атрибута
- Левое внешнее соединение – соединение, при котором кортежи отношения A, не имеющие совпадающих значений в общих атрибутах отношения B, также включаются в общее отношение

Definition 1.42. Деление

Пусть даны отношения $A(X_1 \dots X_n, Y_1 \dots Y_p)$ и $B(Y_1 \dots Y_p)$, причем атрибуты $(Y_1 \dots Y_p)$ – общие для двух отношений
Результатом деления отношения A на B является отношение со схемой $S(X_1 \dots X_n)$, содержащее множество кортежей $(x_1 \dots x_n)$, таких, что для всех кортежей $(y_1 \dots y_p) \in B$ в отношении A найдется кортеж $(x_1 \dots x_n, y_1 \dots y_p)$
Отношение A выступает в роли делимого, отношение B выступает в роли делителя
Все атрибуты отношения B должны входить в состав схемы отношения A
Синтаксис: A DIVIDE BY B или A ÷ B

Remark 1.10.

Деление может быть выражено через операции декартова произведения и вычитания
Типичные запросы, реализуемые с помощью операции деления, обычно в своей формулировке имеют слово все

Example 1.1.

1. Получить имена поставщиков, поставляющих деталь номер 2:
   ((DP JOIN P) WHERE DNUM=2)[PNAME]
2. Получить имена поставщиков, поставляющих по крайней мере одну гайку
   (((D JOIN DP) JOIN P) WHERE DNAME=Гайка)[PNAME] (((D WHERE DNAME=Гайка) JOIN DP) JOIN P)[PNAME]
3. Получить имена поставщиков, поставляющих все детали
   ((DP[PNUM,DNUM] DIVIDE BY D[DNUM]) JOIN P)[PNAME]

1.12 Типы данных

Definition 1.43. PostreSQL и типы данных

SQL – язык со строгой типизацией. Каждый элемент данных имеет некоторый тип, определяющий его поведение и допустимое использование
PostreSQL наделен расширяемой системой типов, более универсальной и гибкой по сравнению с другими реализациями SQL

Notation 1.36. Типы данных

- Символьные
- Числовые
- Дата и время
- Логические
- Двоичные
- Специальные

Definition 1.44. Символьные данные

varchar(n), char(n), text
Константные зачения. Последовательность символов, заключенная в апострофы. Две строковые константы, разделенные пробельными символами и минимум одним переводом строки, объединяются в одну

- Константы со спецпоследовательностями в стиле C
  Начинаются с буквы E (заглавной или строчной)
- Строковые константы со спецпоследовательностями Unicode
  Позволяют включать в строки символы Unicode по их кодам
  Начинается с U& (строчная или заглавная U и амперсанд)
  Символы Unicode можно записывать двумя способами:
  - \ и код символа из четырех шестнадцатеричных цифр
  - \+ и код символа из шести шестнадцатеричных цифр
- Строковые константы, заключенные в доллары
  Используются для работы со строками, содержащими много апострофов или обратных косых черт. Позволяют избежать необходимости зеркалирования служебных символов. Делают строки более читабельными. Обрамляются $[тэг]$

Definition 1.45. Точные числовые данные

Целочисленные типы – smallint (int2), integer (int4), bigint (int8)
Числа фиксированной точности – numeric (precision, scale) и decimal (precision, scale)

Definition 1.46. Числовые данные с плавающей точкой

real, double precision и float(p)
Поддерживают специальные значения Infinity, -Infinity и NaN
Если точность вводимого числа выше допустимой – будет выполняться округление значения. При вводе слишком большого или очень маленького значения будет генерироваться ошибка
Внимание: сравнение двух чисел с плавающей точкой на предмет равенства их значений может привести к неожиданным результатам

Definition 1.47. опследовательные типы

serial (int4), bigserial (int8) и msallserial (int2)
Реализованы как удобная замена целой группы SQL-команд: создание объекта SEQUENCE – генератор уникальных целых чисел, генерация и получение значений последовательности
Часто используются в качестве значений суррогатного первичного ключа (Primary Key)
Нет неоюходимости указывать явное значение для вставки в поле PK

Notation 1.37. Функции для работы с последовательностями

Функция	Тип результата	Описание
currval('name')	bigint	Возвращает последнее сгенерированное значение указанной последовательности (которое было возвращено при последнем вызове функции nextval)
lastval()	bigint	Возвращает последнее сгенерированное значение любой последовательности (которое было возвращено при последнем вызове функции nextval)
nextval('name')	bigint	Генерит и возвращает новое значение последовательности
setval('name', bigint)	bigint	Устанавливает текущее значение последовательности
setval('name', bigint, boolean)	bigint	Устанавливает текущее значение последовательности и флаг is-called, указывающий на то, что это значение уже использовалось

Definition 1.48. Дата и время

date, time и time with time zone (timetz)
Даты обрабатываются в соответствии с григорианским календарем
time хранит время внутри суток. time with time zone хранит время с учетом смещения, соответствующего часовому поясу
При вводе значений их нужно заключать в одинарные кавычки, как и текстовые строки

Definition 1.49. Временная метка (интегральный тип)

timestamp, timestamp with time zone (timestamptz)
Получается в результате объединения типов даты и времени
Оба типа занимают 8 байтов
Значения типа timestamptz хранятся приведенными к нулевому часовому поясу (UTC), а перед выводом приводятся к часовому поясу пользователя

Definition 1.50. Тип interval

Представляет продолжительность отрезка времени
Формат: quantity unit [quantity unit ...] direction
Стандарт ISO 8601: P[yyyy-mm-dd][Thh:mm:ss]
Значение типа interval можно получить при вычитании одной временной метки из другой

Notation 1.38. Операторы даты/времени

- date +/- integer – добавляет/вычитает к дате заданное число дней
- date +/- interval – добавляет/вычитает к дате интервал
- date +/- time – добавляет/вычитает к дате время
- interval +/- interval – складывает/вычисляет интервалы
- timestamp +/- interval – добавляет/вычитает к метке времени интервал
- date - date – возвращает разницу между датами в днях
- timestamp - timestamp – вычитает из одной отметки времени другую (преобразуя 24-часовые интервалы в дни)

Definition 1.51. Логический тип

boolean
Может иметь три состояния: true, false, NULL. Реализует трехзначную логику

Definition 1.52. Двоичные типы данных

bytea
Позволяют хранить байты с кодом 0 и другими непечатаемыми значениями (значения вне десятичного диапазона 32..126)
В операциях с двоичными строками обрабатываются байты в чистом виде
Поддерживает два формата ввода и вывода (параметр bytea-output):

- hex (шестнадцатеричный) – '\x коды символов в 16-ой системе'
- escape (спецпоследовательностей) – '\коды символов в 8-ой системе'

Notation 1.39. Приведение типов

Приведение типов в PostreSQL – это осуществление преобразования одного типа информации в другой
Для приведения типов данных в PostreSQL используется: функция CAST, выражение::тип, тип выражения
Неявные преобразования, производимые PostreSQL, могут влиять на результат запроса