# Лекция №2.
# Функции СУБД.
# Типовая организация СУБД.
# Дореляционные СУБД

- Как было показано в первой лекции, традиционных возможностей файловых систем оказывается недостаточно для построения даже простых информационных систем.
- Мы выявили несколько потребностей, которые не покрываются возможностями систем управления файлами:
  - поддержание логически согласованного набора файлов;
  - обеспечение языка манипулирования данными;
  - восстановление информации после разного рода сбоев;
  - реально параллельная работа нескольких пользователей.
- Можно считать, что если прикладная информационная система опирается на некоторую систему управления данными, обладающую этими свойствами, то эта система управления данными является *системой управления базами данных (СУБД).*

# Основные функции СУБД:

- Непосредственное управление данными во внешней памяти
- Управление буферами оперативной памяти
- Управление транзакциями
- Журнализация
- Поддержка языков БД

# Непосредственное управление данными во внешней памяти

- Эта функция включает обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используются индексы).
- В некоторых реализациях СУБД активно используются возможности существующих файловых систем, в других работа производится вплоть до уровня устройств внешней памяти. Но подчеркнем, что в развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, и если использует, то как организованы файлы. В частности, СУБД поддерживает собственную систему именования объектов БД.

# Управление буферами оперативной памяти

- СУБД обычно работают с БД значительного размера; по крайней мере этот размер обычно существенно больше доступного объема оперативной памяти.
- Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти.
- Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. При этом, даже если операционная система производит общесистемную буферизацию (как в случае ОС UNIX), этого недостаточно для целей СУБД, которая располагает гораздо большей информацией о полезности буферизации той или иной части БД. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

# Управление транзакциями

- Транзакция - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует (COMMIT) изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД.
- Если вспомнить наш пример информационной системы с файлами СОТРУДНИКИ и ОТДЕЛЫ, то единственным способом не нарушить целостность БД при выполнении операции приема на работу нового сотрудника является объединение элементарных операций над файлами СОТРУДНИКИ и ОТДЕЛЫ в одну транзакцию.
- То свойство, что каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к БД. При соответствующем управлении параллельно выполняющимися транзакциями со стороны СУБД каждый из пользователей может в принципе ощущать себя единственным пользователем СУБД (на самом деле, это несколько идеализированное представление, поскольку в некоторых случаях пользователи многопользовательских СУБД могут ощутить присутствие своих коллег).

# Журнализация

- Одним из основных требований к СУБД является надежность хранения данных во внешней памяти.
- Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя.
- Обычно рассматриваются два возможных вида аппаратных сбоев:
  - так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания),
  - и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти.
- Примерами программных сбоев могут быть:
  - аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя)
  - или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной.

- Понятно, что в любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.
- Журнал - это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД.
- Во всех случаях придерживаются стратегии "упреждающей" записи в журнал (так называемого протокола Write Ahead Log - WAL). Грубо говоря, эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. Известно, что если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

# Поддержка языков БД

- Для работы с базами данных используются специальные языки, в целом называемые *языками баз данных*.
- В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка - *язык определения схемы* БД *(SDL - Schema Definition Language)* и *язык манипулирования данными (DML - Data Manipulation Language).* SDL служил главным образом для определения логической структуры БД, т.е. той структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, т.е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные. Мы рассмотрим более подробно языки ранних СУБД в следующей лекции.

# Поддержка языков БД

- В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language).
- Прежде всего, язык SQL сочетает средства SDL и DML, т.е. позволяет определять схему реляционной БД и манипулировать данными. При этом именование объектов БД (для реляционной БД - именование таблиц и их столбцов) поддерживается на языковом уровне в том смысле, что компилятор языка SQL производит преобразование имен объектов в их внутренние идентификаторы на основании специально поддерживаемых служебных таблиц-каталогов. Внутренняя часть СУБД (ядро) вообще не работает с именами таблиц и их столбцов.

# Типовая организация современной СУБД

- мы выделили следующие основные функции СУБД:
  - управление данными во внешней памяти;
  - управление буферами оперативной памяти;
  - управление транзакциями;
  - журнализация и восстановление БД после сбоев;
  - поддержание языков БД.

- Логически в современной реляционной СУБД можно выделить
  - наиболее внутреннюю часть - ядро СУБД (часто его называют Data Base Engine),
  - компилятор языка БД (обычно SQL),
  - подсистему поддержки времени выполнения,
  - набор утилит.
- В некоторых системах эти части выделяются явно, в других - нет, но логически такое разделение можно провести во всех СУБД.

- Ядро СУБД отвечает за
  - управление данными во внешней памяти,
  - управление буферами оперативной памяти,
  - управление транзакциями
  - и журнализацию.
- Соответственно, можно выделить такие компоненты ядра (по крайней мере, логически, хотя в некоторых системах эти компоненты выделяются явно), как
  - менеджер данных,
  - менеджер буферов,
  - менеджер транзакций
  - и менеджер журнала.
- Как можно было понять из первой части этой лекции, функции этих компонентов взаимосвязаны, и для обеспечения корректной работы СУБД все эти компоненты должны взаимодействовать по тщательно продуманным и проверенным протоколам. Ядро СУБД обладает собственным интерфейсом, не доступным пользователям напрямую и используемым в программах, производимых компилятором SQL (или в подсистеме поддержки выполнения таких программ) и утилитах БД. Ядро СУБД является основной резидентной частью СУБД. При использовании архитектуры "клиент-сервер" ядро является основной составляющей серверной части системы.

- Основной функцией компилятора языка БД является компиляция операторов языка БД в некоторую выполняемую программу.
- Основной проблемой реляционных СУБД является то, что языки этих систем (а это, как правило, SQL) являются непроцедурными, т.е. в операторе такого языка специфицируется некоторое действие над БД, но эта спецификация не является процедурой, а лишь описывает в некоторой форме условия совершения желаемого действия. Поэтому компилятор должен решить, каким образом выполнять оператор языка прежде, чем произвести программу. Применяются достаточно сложные методы оптимизации операторов.
- Результатом компиляции является выполняемая программа, представляемая в некоторых системах в машинных кодах, но более часто в выполняемом внутреннем машинно-независимом коде. В последнем случае реальное выполнение оператора производится с привлечением подсистемы поддержки времени выполнения, представляющей собой, по сути дела, интерпретатор этого внутреннего языка.

- Наконец, в отдельные утилиты БД обычно выделяют такие процедуры, которые слишком накладно выполнять с использованием языка БД, например,
  - загрузка и выгрузка БД,
  - сбор статистики,
  - глобальная проверка целостности БД и т.д.
- Утилиты программируются с использованием интерфейса ядра СУБД, а иногда даже с проникновением внутрь ядра.

УРА!

ПЕРЕМЕНА!

# Дореляционные СУБД

- Системы, основанные на инвертированных списках,
- иерархические
- и сетевые СУБД.
- Сильные места и недостатки ранних систем

# Зачем нам знать о таких системах?

- В этом есть смысл по трем причинам:
  - во-первых, эти системы исторически предшествовали реляционным, и для правильного понимания причин повсеместного перехода к реляционным системам нужно знать хотя бы что-нибудь про их предшественников;
  - во-вторых, внутренняя организация реляционных систем во многом основана на использовании методов ранних систем;
  - в-третьих, некоторое знание в области ранних систем будет полезно для понимания путей развития постреляционных СУБД.

# Общие характеристики ранних систем

- Эти системы активно использовались в течение многих лет, дольше, чем используется какая-либо из реляционных СУБД. На самом деле некоторые из ранних систем используются даже в наше время, накоплены громадные базы данных, и одной из актуальных проблем информационных систем является использование этих систем совместно с современными системами.
- Все ранние системы не основывались на каких-либо абстрактных моделях. Как мы упоминали, понятие модели данных фактически вошло в обиход специалистов в области БД только вместе с реляционным подходом. Абстрактные представления ранних систем появились позже на основе анализа и выявления общих признаков у различных конкретных систем.
- В ранних системах доступ к БД производился на уровне записей. Пользователи этих систем осуществляли явную навигацию в БД, используя языки программирования, расширенные функциями СУБД. Интерактивный доступ к БД поддерживался только путем создания соответствующих прикладных программ с собственным интерфейсом.
- Навигационная природа ранних систем и доступ к данным на уровне записей заставляли пользователя самого производить всю оптимизацию доступа к БД, без какой-либо поддержки системы.
- После появления реляционных систем большинство ранних систем было оснащено "реляционными" интерфейсами. Однако в большинстве случаев это не сделало их по-настоящему реляционными системами, поскольку оставалась возможность манипулировать данными в естественном для них режиме.

# Основные особенности систем, основанных на инвертированных списках

- Рассмотрим только самое важное:
  - Структуры данных
  - Манипулирование данными
  - Ограничения целостности

# Системы, основанные на инвертированных списках: Структуры данных

- База данных, организованная с помощью инвертированных списков, похожа на реляционную БД, но с тем отличием, что хранимые таблицы и пути доступа к ним видны пользователям. При этом:
  - Строки таблиц упорядочены системой в некоторой физической последовательности.
  - Физическая упорядоченность строк всех таблиц может определяться и для всей БД.
  - Для каждой таблицы можно определить произвольное число ключей поиска, для которых строятся индексы. Эти индексы автоматически поддерживаются системой, но явно видны пользователям.

# Системы, основанные на инвертированных списках: Манипулирование данными

- Поддерживаются два класса операторов:
  - Операторы, устанавливающие адрес записи, среди которых:
    - прямые поисковые операторы (например, найти первую запись таблицы по некоторому пути доступа);
    - операторы, находящие запись в терминах относительной позиции от предыдущей записи по некоторому пути доступа.
  - Операторы над адресуемыми записями
- Типичный набор операторов:
  - LOCATE FIRST - найти первую запись таблицы T в физическом порядке; возвращает адрес записи;
  - LOCATE FIRST WITH SEARCH KEY EQUAL - найти первую запись таблицы T с заданным значением ключа поиска K; возвращает адрес записи;
  - LOCATE NEXT - найти первую запись, следующую за записью с заданным адресом в заданном пути доступа; возвращает адрес записи;
  - LOCATE NEXT WITH SEARCH KEY EQUAL - найти следующую запись таблицы T в порядке пути поиска с заданным значением K; должно быть соответствие между используемым способом сканирования и ключом K; возвращает адрес записи;
  - LOCATE FIRST WITH SEARCH KEY GREATER - найти первую запись таблицы T в порядке ключа поиска K со значением ключевого поля, большим заданного значения K; возвращает адрес записи;
  - RETRIVE - выбрать запись с указанным адресом;
  - UPDATE - обновить запись с указанным адресом;
  - DELETE - удалить запись с указанным адресом;
  - STORE - включить запись в указанную таблицу; операция генерирует адрес записи.

# Системы, основанные на инвертированных списках: Ограничения целостности

- Общие правила определения целостности БД отсутствуют.
- В некоторых системах поддерживаются ограничения уникальности значений некоторых полей, но в основном все возлагается на прикладную программу.

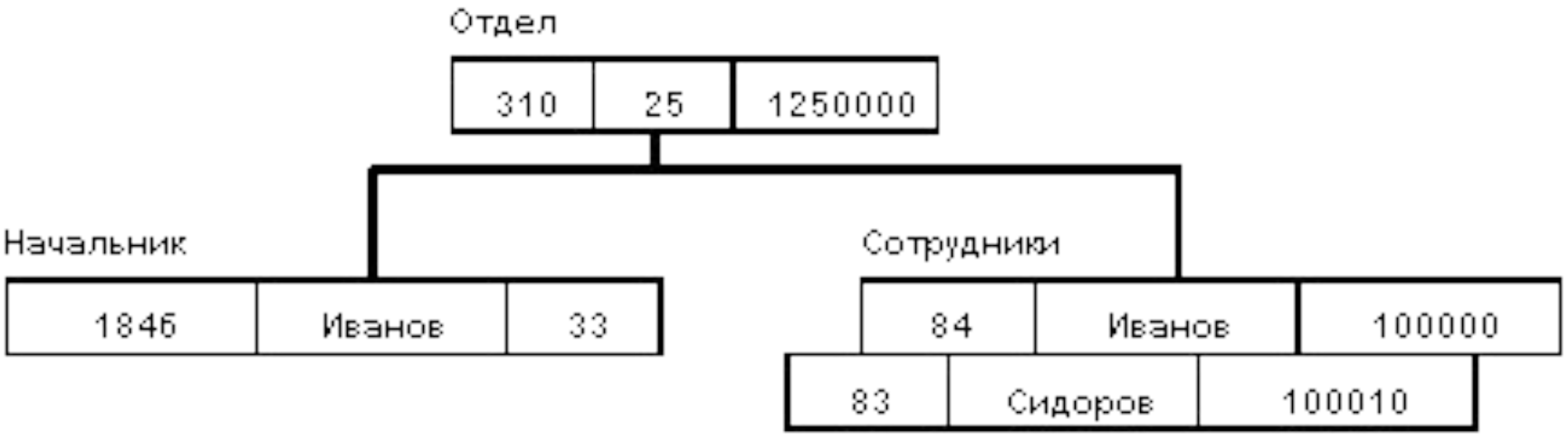
# Иерархические системы:
## Структуры данных

- Иерархическая БД состоит из упорядоченного набора деревьев; более точно, из упорядоченного набора нескольких экземпляров одного типа дерева.
- Тип дерева состоит из одного "корневого" типа записи и упорядоченного набора из нуля или более типов поддеревьев (каждое из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор типов записи.

# Пример


- Здесь Отдел является предком для Начальник и Сотрудники, а Начальник и Сотрудники - потомки Отдел. Между типами записи поддерживаются связи.

# Пример: один экземпляр дерева




- Все экземпляры данного типа потомка с общим экземпляром типа предка называются близнецами. Для БД определен полный порядок обхода - сверху-вниз, слева-направо.

# Иерархические системы:
## Манипулирование данными

- Примерами типичных операторов манипулирования иерархически организованными данными могут быть следующие:
  - Найти указанное дерево БД (например, отдел 310);
  - Перейти от одного дерева к другому;
  - Перейти от одной записи к другой внутри дерева (например, от отдела - к первому сотруднику);
  - Перейти от одной записи к другой в порядке обхода иерархии;
  - Вставить новую запись в указанную позицию;
  - Удалить текущую запись.

# Иерархические системы:

## Ограничения целостности

- Автоматически поддерживается целостность ссылок между предками и потомками.
- *Основное правило: никакой потомок не может существовать без своего родителя.*
- Заметим, что аналогичное поддержание целостности по ссылкам между записями, не входящими в одну иерархию, не поддерживается

# Сетевые системы:
## Структуры данных

- Сетевой подход к организации данных является расширением иерархического. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков.
- Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно, из набора экземпляров каждого типа из заданного в схеме БД набора типов записи и набора экземпляров каждого типа из заданного набора типов связи.
- Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора экземпляров типа записи потомка. Для данного типа связи L с типом записи предка P и типом записи потомка C должны выполняться следующие два условия:
  - Каждый экземпляр типа P является предком только в одном экземпляре L;
  - Каждый экземпляр C является потомком не более, чем в одном экземпляре L.

# Простой пример сетевой схемы БД:

# Сетевые системы:
## Манипулирование данными

- Примерный набор операций может быть следующим:
  - Найти конкретную запись в наборе однотипных записей (инженера Сидорова);
  - Перейти от предка к первому потомку по некоторой связи (к первому сотруднику отдела 310);
  - Перейти к следующему потомку в некоторой связи (от Сидорова к Иванову);
  - Перейти от потомка к предку по некоторой связи (найти отдел Сидорова);
  - Создать новую запись;
  - Уничтожить запись;
  - Модифицировать запись;
  - Включить в связь;
  - Исключить из связи;
  - Переставить в другую связь и т.д.

# Сетевые системы:

## Ограничения целостности

- В принципе их поддержание не требуется, но иногда требуют целостности по ссылкам (как в иерархической модели).

Спасибо за внимание!

ВОПРОСЫ?