# B1 Файловые системы обработки данных

КП - конечный пользователь

ПП - прикладная программа

# 2.1 Организация файловых систем обработки данных

Вариант “Данные вокруг программ”

* Каждая ПП ведёт самостоятельно и независимо все свои файлы
* Форматы записей определяются автором ПП  
  (это приемлемо, если разные прикладные программы обрабатывают различные данные,  
  и не приемлемо, если обработка одинаковых данных разнородна, не идентична)
* приводит к неуправляемому дублированию

Суть: независимая обработка данных (возможно, пересекающихся и не идентичных по формату + дублированных)

Вариант “Программы вокруг данных”

* состав и форматы записей должны быть согласованы
* наличие общего поля данных (некая централизованность внутри рабочей группы - РГ)
* каждая ПП обрабатывает определенную часть общего поля данных (устраняется дублирование)
* возникает проблема доступа к данным (конфликты)
* необходимость разграничения полномочий (права доступа)

# 2.2 Недостатки ФСОД

## 2.2.1 Неконтролируемая избыточность данных.

(в контексте “Данные вокруг программ” и “Программы вокруг данных с допущением ошибок ПП-мами”)

* неидентичность пересекающихся(дублирующихся) наборов файлов различных ПП
* приводит к дезинформации КП, неверным управленческим решениям

## 2.2.2 Зависимость программ от данных (физических форматов).

(в контексте “Программы вокруг данных”)

* каждая ПП содержит определение физических форматов данных. Тогда изменение структуры записи файла ведёт к перекомпиляции всех ПП (даже тех, которые не обрабатывают измененное поле структуры)

## 2.2.3 Разделение и изоляция данных.

* Объекты (поля) связаны между собой
* Каждый объект дополнительно представлен в виде отдельного файла
* Файлы (записи в них) связаны с другими файлами взаимно однозначным соответствием
* О связях записей знает ПП, отвечает за обработку, поэтому изменение записи в одном файле ведет к обязательному синхронному изменению ПП-й других взаимосвязанных записей
* Идея осуществима, но: у ПП нет возможности получения сведений о структурах записей обрабатываемых файлов

## 2.2.4 Большое количество автономных программ в системе.

(ПП выполняют только предопределенную обработку данных)

* невозможно обрабатывать произвольные запросы
* требуется писать отдельную ПП, компилировать, отлаживать
* но для управления предприятием требуется оперативность, которая не обеспечивается

# 2.3 Чем обусловлены недостатки ФСОД

* Данные, с которыми работает подразделение, рассматриваются как внутренний ресурс - т.е. подразделение хранит, накапливает и обрабатывает данные изолированно, что приводит к избыточности, конфликтности на уровне всего предприятия
* ПП хранят:   
   - определения файлов (структур), что приводит к дублированию определений   
   - ссылки на устройства ввода/вывода (ссылки физического уровня)  
  изменение ОС (платформы), структур, устройств  
  - всё это ведёт к перекомпиляции всех ПП
* все ПП - это автономные функциональные единицы, алгоритм действий которых предопределен, поэтому нельзя в ФСОД отправить моментальный запрос без написания отдельной ПП

# Юра

# B4. База данных

## 3.3.1 Предметная область

* **Базой данных** называют совокупность записей, необходимых для осуществления какой-то организованной деятельности.
* Часть реального мира, сведения о которой представляют интерес с точки зрения владельца базы данных, называется **предметной областью** (ПО) базы данных.

## 3.3.2 Понятие базы данных в информатике

* **База данных В ИНФОРМАТИКЕ** – это динамическая модель предметной области. Её состояние в каждый момент времени отражает текущее состояние ПО.

## 3.3.3 Определение термина «база данных»

* **База данных (БД)** — это самодокументированная интегрированная совокупность записей.
* Самодокументированность означает, что вместе с данными пользователей в БД содержится описание её собственной структуры.

## 3.7 Структурные единицы базы данных

* **Поле** – это элементарная именованная единица логической организации данных, которая соответствует неделимой единице информации — (атрибуту, реквизиту).
* **Запись** — именованная совокупность логически связанных полей.
* **Таблица** — это совокупность экземпляров записей одного типа.

## 3.8 Ключи

* **Первичный ключ** (Primary Key, PK) – это поле (или группа полей), значения которого принципиально не могут быть одинаковыми в различных одновременно существующих экземплярах записи. Это **уникальный** идентификатор экземпляров записи.
* В ряде случаев в структуре записи выделяют также поисковые поля, иногда называемые **вторичными ключами** (Secondary Key, SK). В качестве вторичного ключа может быть выбрано **любое** поле, по значениям которого приходится **часто** искать или сортировать экземпляры записи.
* **Внешний ключ** (Foreign Key, FK) – это поле записи ТАБЛИЦЫ 1, значения которого являются ссылками на существующие экземпляры записи ТАБЛИЦЫ 2.

## 3.9 Представление метаданных

* **Метаданные** – это информация об объектах БД пользователя.
* **Метаданные**, как и данные пользователей, сохраняются в таблицах. Их называют системными таблицами. Они составляют часть системного каталога. В них содержатся, в частности, описания полей и типов записей БД пользователя, сведения об отношениях таблиц, индексах, хранимых процедурах, триггерах и других объектах.

## 3.10 Индексы

* **Индекс** – сопоставленная базовой таблице служебная структура, обеспечивающая быстрый поиск записей БД пользователя.

Основной идеей индекса является хранение **упорядоченного** списка значений ключа с привязкой к каждому значению списка указателей — физических идентификаторов записей, содержащих это значение. Эти избыточные данные, используются для улучшения производительности системы и доступности БД.

# 

# В5. Архитектура ANSI/SPARC

КП - конечный пользователь

ПП - прикладная программа

# 4.1 Уровни представления данных (Архитектура ANSI/SPARC)

## 4.1.1 Концептуальный уровень

(Обобщённое представление пользователей)

Это множество объектов концептуальных записей

* логическое представление данных (без физических ссылок)
* состав: определения типов концептуальных записей, связи записей, ограничения целостности данных и т.д. - в общем, значимые по смыслу поля (содержательные)  
  + смысловая информация для обеспечения безопасности (целостности) данных
* Определения этого уровня выполняются на языке обработки данных и хранятся в системных таблицах (метаданных), доступ к которым могут получить любой пользователь
* СУБД использует определения структур при обработке поступающих запросов, обновляет метаданные при обновлении концептуальной схемы (структуры)
* Т.о. обеспечивается актуальность и целостность смысловой и логической структуры данных

## 4.1.2 Внутренний уровень

(Представление данных во внешней/физической памяти)

Это множество физических данных, которые запрашиваются Операционной Системой

1. Внутренняя схема:

* Образует физическое представление данных (уровня реализации)
* Состав: определения типов внутренних записей, файлов физической памяти, индексов и др.
* Определения описываются на языке внутреннего управления СУБД
* между внутренними и концептуальными записями нет взаимно однозначного соответствия (в т.ч. типов и длин полей записей) - разные поля одной внутренней записи могут соответствовать разным полям нескольких концептуальных записей, и наоборот (одна концептуальная - много внутренних)  
  НО: между концептуально и внутренней схемами есть взаимно-однозначное соответствие - для формирования экземпляра концептуальной записи из значений полей внутренних записей (и наоборот)
* Содержание записи: смысловые, системные и служебные поля (личные, для разраба БД).

1. Другая информация: о распределении дискового пространства, расположении файлов (физических адресов), сжатии и методах шифрования

## 4.1.3 Внешний уровень

(Представление пользователей)

Внешнее представление данных (интерфейс) для конкретного пользователя(обрезанные данные) в рамках его прав (обеспечивается санкционированных доступ).

Компоненты:

1. Внешняя схема:

* это подмножество концептуальной схемы (или некоторая функция от концептуальной схемы)
* формируется из полей концептуальной схемы как часть сущностей, атрибутов и связей, полезных для конкретного пользователя
* при этом могут быть представлены как производные полей концептуальных записей, и одно и то же поле для разных КП отображается по разному (в зависимости от целей КП), и наоборот (несколько концептуальных записей собираются в одну внешнюю для КП)

1. Определения форм, отчетов, запросов и т.д.

## 4.1.4 Отображения

Уровни СУБД связаны между собой взаимно-однозначным отображением:

* Концептуальные поля и записи соответствуют внутренним полям и записям => при изменении внутренних структур меняется отображение так, чтобы концептуальные остались неизменны
* Концептуальные поля и записи соответствуют внешним полям и записям => при изменении концептуальных обновляется отображение

## 4.1.5 Системный каталог

Системный каталог СУБД - это внутренняя БД СУБД для согласованного управления системой баз данных.

Состав системного каталога:

* определение типов записей всех уровней
* описание правил отображений
* данные о пользователях и их прав (привилегий) на доступ к данным
* информация о программных и аппаратных ресурсах системы

## 4.1.6 Независимость приложений от данных

Отсутствие влияния изменений нижнего слоя на верхний слой представления данных за счёт модификации отображений (таблиц системного каталога, которые хранят соответствие схем) одного уровня на другую.

Типы независимости:

* Логическая независимость - Полная защищённость внешнего представления от изменений на концептуальном уровне. Означает, что добавление или удаление типа концептуальной записи, поля или связи не влияет на уже существующие внешние схемы. Это видно только для КП, для которых обновление предназначалось.
* Физическая независимость - Полная защищённость концептуального представления от изменений на внутренней уровне. Причина изменений на внутреннем уровне: повышение производительности через слияние или разделение внутренних записей, изменение индексов, файловых систем, внешних устройств хранения…без изменения концептуальной схемы.

# B6. Операции в базе данных

## 4.2 Дисциплина доступа приложений к хранимым данным

## Шаг 1. СУБД получает запрос прикладной программы (ПП) в терминах внешней модели.

* Шаг 2. определяет, какие записи концептуального уровня необходимы для формирования требуемых внешних записей
* Шаг 3. определяет, какие внутренние записи необходимы для формирования затребованных записей
* Шаг 4. Выдает ос запрос на чтение необходимых записей физической бд
* Шаг 5. ОС считывает записи и помещает в буферы СУБД
* Шаг 6. СУБД формирует внешние записи
* Шаг 7. Пересылает их в рабочую область
* Шаг 8. Пересылает прикладной программе сообщение об исполнении

## 4.3 Операции обработки данных

* Извлечение записей (**RETRIEVE**) – считывание совокупности записей ФБД.
* Добавление записей (**INSERT**) – создание в рабочем буфере новых записей, содержащих введённые приложением новые значения данных.
* Изменение значений (**UPDATE**) – замена существующих значений указанных полей в извлечённом множестве записей новыми значениями, введёнными приложением
* Удаление записей (**DELETE**) – уничтожение в рабочем буфере подмножества извлечённых записей, указанного приложением.

## 4.4 Дисциплина обменов с внешней памятью

* СУБД, приняв запрос приложения, проверяет, есть ли в буфере данные, необходимые для его исполнения.
* Если нужных данных нет в буфере, то система проверяет, достаточно ли в нём места для размещения требуемых физических записей.
* Если места достаточно, то нужные записи извлекаются, и запрос приложения исполняется.

## 4.5 Операции обновления и целостность данных.

* На введенные данные есть ограничения и правила.
* СУБД не в состоянии гарантировать истинность вводимых значений данных. Она может лишь гарантировать их соответствие заданным ограничениям целостности.

## 4.6 Транзакции в базе данных.

* **транзакция** – это последовательность действий над базой данных, выполняемая по запросам одного пользователя или приложения.
* 

# 

# 

# В7. Управление доступом к данным

## 4.7.1 Принципы ограничения доступа.

1. Служащий имеет право доступа только к тем сведениям, которые необходимы для исполнения его служебных обязанностей.

2. Служащий имеет право выполнять только те манипуляции доступными данными, которые обусловлены его служебными обязанностями.

## 4.7.2 Авторизация пользователей.

* Подсистема доступа обеспечивает предоставление **прав** (привилегий) доступа к системе и её объектам.

## 4.7.3 Аутентификация.

* **аутентификации** - проверка того, является ли пытающийся подключиться тем, за кого он себя выдаёт

## 4.7.4 Привилегии доступа

* Системная **привилегия** это право создания и модификации объектов БД – схем, таблиц, приложений, правил и т.п. Пользователь, создавший объект, является его владельцем.
* **объектная привилегия** есть тройка (идентификатор авторизации, объект БД, действие).

## 4.7.5 Подсхемы

* **Подсхема** (внешняя схема) представляет собой набор виртуальных записей, реально не существующих в БД.

## 4.7.6 Сеанс

* Сеанс – это промежуток времени между моментом подключения пользователя к системе и моментом отключения.

# 

# В8. Управление параллелизмом

## 4.8.1 Необходимость управления параллелизмом

* при чередовании операций различных транзакций результаты параллельно выполняемых обновлений данных могут **оказаться неверными**, несмотря на то, что каждая транзакция выполнялась вполне корректно

**Чтение «грязных» данных** - Чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится).

**Потеря обновлений** - Ситуация, когда при одновременном изменении одного блока данных разными транзакциями одно из изменений теряется.

**Неповторяющееся чтение** - Проявляется, когда при повторном чтении в рамках одной транзакции, ранее прочитанные данные, оказываются изменёнными. Данный эффект может наблюдаться при уровне изоляции ниже, чем REPEATABLE READ.

**Фантомное чтение** - Можно наблюдать, когда одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям.

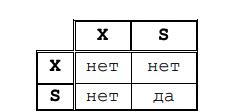
## 4.8.2 Изолированность транзакций

* Read uncommitted (чтение незафиксированных данных) - Блокировка записи данных на время выполнения изменения, т.е. команды изменения одних и тех же строк выполнятся последовательно
* Read committed (чтение фиксированных данных) - блокирует строки на чтение пока с ними работает тразанкция
* repeatable read - гарантия что незафиксированные данные не будут прочитаны, а данные, находящиеся в обработке ну будут изменены.
* serializable - все тразнакции обрабатываются одна за другой.



## 4.8.3 Двухфазный протокол блокировки

* Прежде чем выполнять какую-либо операцию над объектом базы данных А, транзакция Т должна запросить блокировку (захват) А.
* S (Shared lock) – разделяемый захват, необходимый для выполнения операции чтения; X (eXclusive lock) – монопольный захват, необходимый для выполнения операций добавления, удаления и модификации объекта



Никита

# B3. Система управления базами данных (СУБД)

## 3.3.4 Система управления базами данных (СУБД)

**Система управления базами данных —** это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

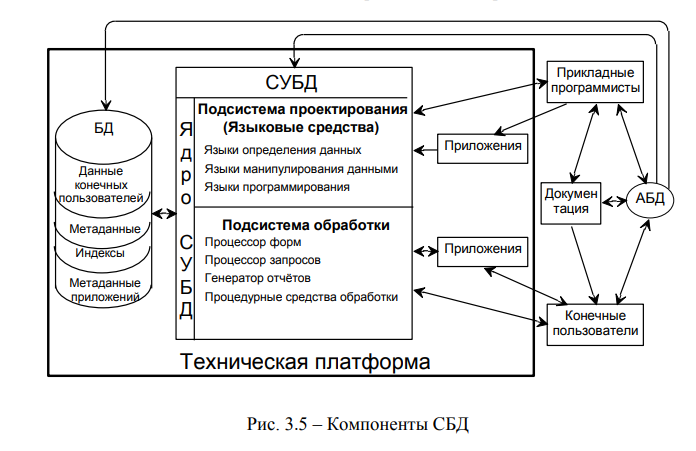
Создание баз данных, их поддержка и обеспечение доступа пользователей к данным осуществляется в СБД централизованно с помощью специальных программных средств — **системы управления базами данных.**

## **3.5 Основные подсистемы СУБД**

## 3.5.1 Подсистема проектирования

* Язык определения данных (ЯОД) предоставляет средства описания элементов и структур данных, экранных форм и других параметров приложений.
* Язык манипулирования данными (ЯМД) обеспечивает навигацию в БД и формулирование запросов к данным.
* Язык программирования предназначен для написания прикладных программ, обрабатывающих данные.

## 3.5.2 Подсистема обработки



**Подсистема обработки** (см рис 3.5) занимается обработкой компонентов приложения. Так, при открытии формы процессор форм запрашивает её определение из раздела БД «Метаданные приложений», воспроизводит элементы формы на экране, связывает поля с элементами хранимых данных и отображает в них текущие значения данных.

Аналогичные функции выполняют процессор запросов и генератор отчётов. Процедурные средства предназначены для обработки запросов прикладных программ на чтение и запись данных.

## 3.5.3 Ядро СУБД

**Ядро СУБД** — это постоянно загруженная часть системы. Оно обеспечивает интерфейс между БД и двумя другими компонентами СУБД.

Например, подсистема обработки принимает запрос приложения на считывание данных. Запрос сформулирован в терминах приложения. Подсистема обработки формулирует его в терминах логических единиц организации данных (например, таблиц) и передаёт ядру. Ядро преобразует запрос в последовательность команд операционной системы, считывающих данные с физического носителя.

Кроме этого, ядро участвует в управлении транзакциями, санкционировании доступа к данным, резервном копировании и восстановлении БД.

# B9. Восстановление базы данных

# **4.9 Восстановление БД**

## 4.9.1 Необходимость восстановления

В процессе эксплуатации СБД могут возникать ситуации, в результате которых база данных может быть частично или даже полностью разрушена.

Рассмотрим их.

**Локальный сбой** – это аварийное прекращение одной транзакции.

Причиной может быть, например, попытка деления на ноль или нарушение ограничений целостности. В этот же ряд следует поставить явное завершение транзакции оператором ROLLBACK и взаимную блокировку транзакций. Если транзакция выполняла обновление данных, то состояние БД в момент локального сбоя может оказаться несогласованным. Для восстановления согласованности необходимо устранить изменения данных, произведённые прерванной транзакцией – выполнить индивидуальный откат транзакции.

**Мягкий сбой системы** может произойти, например, вследствие аварийного отключения питания или при возникновении неустранимого сбоя процессора и т.п. В этом случае теряется содержимое оперативной памяти. Аварийно прерываются все существующие транзакции.

Могут оказаться не зафиксированными в ФБД результаты транзакций, завершившихся оператором COMMIT. При перезагрузке системы должен быть выполнен откат всех, не завершившихся к моменту сбоя транзакций. Зафиксированные транзакции, результаты которых к моменту сбоя не попали во внешнюю память, должны быть автоматически исполнены повторно.

**Жёсткий сбой** – это физическое разрушение базы данных.

Ситуация весьма маловероятная, но её последствия для организации-владельца данных могут быть катастрофическими. Поэтому система должна быть в состоянии восстановить базу данных даже в этом случае

4.9.2 Системный журнал

*Для восстановления согласованного состояния БД при программном откате транзакции или после локального сбоя должна сохраняться полная информация обо всех операциях обновления, выполненных транзакцией*.

Эта информация хранится в **системном журнале** **(СЖ)**. Записи журнала содержат детальные сведения обо всех операциях каждой транзакции. В них для каждой транзакции фиксируется

* идентификатор транзакции;
* идентификатор пользователя, запустившего транзакцию;
* время начала транзакции;
* время и способ завершения транзакции.

Для каждой операции фиксируется

* идентификатор транзакции, в которой выполнялась операция;
* время начала операции;
* идентификатор обрабатываемого элемента данных;
* тип операции;
* копия элемента данных до операции (для операций обновления и удаления);
* копия элемента данных после операции (для операций обновления значения и вставки).

Имея эту информацию, можно выполнить откат. Для этого нужно

1) выбрать из СЖ все записи об операциях транзакции;

2) расположить их в порядке, обратном хронологическому;

3) последовательно просматривая обратный список операций, выполнять над текущим состоянием рабочего буфера операции, обратные по смыслу.

Кроме того, каждой транзакции, завершившейся штатно (т.е. операторами COMMIT или ROLLBACK), сопоставляется запись об окончании.

Но в случае мягкого сбоя этого мало, так как мягкий сбой приводит к потере содержимого рабочих буферов БД. Нужно ещё располагать состоянием БД на какой-то момент времени **t0** и знать, какие транзакции в этот момент существовали в системе. Тогда, если в момент **tf > t0** произойдёт мягкий сбой, то для восстановления системы достаточно будет загрузить в оперативную память зафиксированное в момент **t0** состояние рабочих буферов БД, проанализировать записи СЖ, сделанные в интервале **[t0, tf]**, и восстановить (выполнить повторно) успешно завершённые или откатить незавершённые транзакции.

*Итак, для обеспечения возможности восстановления системы после мягкого сбоя состояние рабочих буферов БД должно периодически фиксироваться во внешней памяти, и каждая запись журнала транзакций должна иметь временнỳю метку.*

В буфере СЖ размещаются записи от многих транзакций. Велика вероятность того, что к моменту мягкого сбоя он не будет сброшен во внешнюю память автоматически, и часть записей будет утеряна. Поэтому буфер журнала принудительно выталкивается во внешнюю память при завершении каждой транзакции. Только после этого транзакция считается завершённой.

*Обновления, выполненные транзакцией, не могут попасть во внешнюю память раньше, чем соответствующие ей записи СЖ.*

Это правило составляет суть протокола предварительной записи в журнал (протокол WAL – Write Ahead Log).

4.9.3 Индивидуальный откат транзакции

**Индивидуальный откат** транзакции производится либо по явно заданному оператору **ROLLBACK**, либо вследствие локального сбоя.

* Для осуществления отката
* создаётся список записей журнала от данной транзакции.
* Элементы списка размещены в порядке, обратном хронологическому.
* Список последовательно просматривается и для каждой записи выполняется противоположная по смыслу операция, восстанавливающая предыдущее состояние объекта базы данных.
* Начальным состоянием для процедуры отката является состояние буферов БД в момент прекращения транзакции.

С точки зрения системы процедура отката является транзакцией. Поэтому обратные операции также регистрируются в журнале. Это «перестраховка» на случай мягкого сбоя системы в процессе отката. Имея эти записи, система сможет «дооткатить» транзакцию, откат которой был прерван.

4.9.4 Восстановление после мягкого сбоя

При перезагрузке после мягкого сбоя система должна

* откатить все прерванные транзакции
* выполнить повторно все успешно завершившиеся к моменту сбоя, но не зафиксированные в ФБД.

В процессе восстановления выполняются следующие действия.

1) Создаётся два списка транзакций: отменяемых (UNDO) и исполняемых повторно (REDO). В список UNDO включаются все транзакции, указанные в записи контрольной точки. Список REDO остаётся пустым.

2) Выполняется анализ записей журнала регистрации, начиная с записи контрольной точки. 3) Если обнаружена запись о начале транзакции Т, то эта транзакция добавляется в список UNDO.

4) Если обнаружена запись о завершении транзакции Т оператором COMMIT, то эта транзакция добавляется в список REDO.

5) По достижении конца файла журнала списки анализируются с целью различения транзакций типа Т2 – Т4 и Т3 – Т5.

6) Из списка UNDO исключаются транзакции, попавшие в список REDO.

7) Системный журнал просматривается от конца до записи контрольной точки, и отменяются транзакции из списка UNDO.

8) Системный журнал просматривается от записи контрольной точки до конца, и выполняются повторно транзакции из списка REDO. По окончании этой процедуры система готова к работе.

4.9.5 Восстановление после жесткого сбоя

При жёстком сбое физическая база данных оказывается разрушенной. Поэтому для восстановления необходимо иметь её **резервную копию**. Обычно резервное копирование ФБД выполняется по факту переполнения системного журнала. Для этого в файле журнала устанавливается так называемая «*жёлтая зона*», по достижении которой запуск новых транзакций не производится. Система дожидается окончания всех существующих транзакций. После этого рабочие буферы журнала и базы данных выталкиваются во внешнюю память. Созданное таким образом состояние ФБД копируется на резервный носитель, а файл журнала очищается. Может быть также создана резервная копия журнала.

**При восстановлении после жёсткого сбоя восстанавливается состояние ФБД на момент последнего копирования, а затем по текущему журналу регистрации повторно исполняются все транзакции, успешно завершившиеся до момента сбоя.**

Поскольку жёсткий сбой не сопровождается потерей буферов, после восстановления можно даже продолжить исполнение не завершившихся до сбоя транзакций, но обычно все незавершённые транзакции откатываются