КР1. Основные вопросы.

▼ **В1. Организация файловых систем обработки данных**

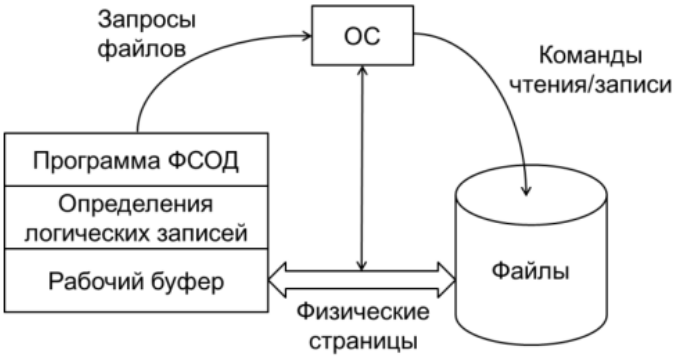
Изначально: каждый отдел - автономная в информационном плане единица. Программы каждой рабочей группы ведут и обрабатывают свои файлы. Предполагается, что наборы данных различных отделов, не пересекаются.

Варианты организации прямого доступа:
1) Данные вокруг программы (каждая программа самостоятельно ведёт все файлы, содержащие необходимые ей данные. Форматы записей файлов определялись автором программы. Это вполне приемлемо, если двум различным программам не приходится обрабатывать одни и те же данные. В противном случае одни и те же по смыслу данные, возможно, в различных формах, придётся сохранять в нескольких независимо обрабатываемых файлах. Идентичность этих "копий" не гарантирована.)

2)Программы вокруг данных (состав и форматы записей файлов согласовывались внутри рабочей группы. Создавалось общее для всех программ отдела поле данных. Каждая программа обрабатывала определённую часть общих данных. При такой организации удаётся избежать неуправляемого дублирования данных, но возникают другие проблемы. В частности, проблемы управления параллельным доступом программ к данным и разграничения полномочий доступа.)

Каждая программа:

- содержит в своём теле определения всех необходимых файлов.
- самостоятельно сканирует файлы, чтобы извлечь или обновитьь их.
- использует методы доступа конкретной ОС.



▼ Недостатки ФСОД

▼ **Неконтролируемая избыточность**

Возможная противоречивость результатов работы различных программ. Информационные интересы пользователей различных программ пересекаются. Поэтому различные, независимо обрабатываемые файлы должны содержать дубликаты общих данных. ФСОД не имеет средств контроля идентичности дубликатов, а люди, вводящие в файлы значения данных, не могут работать безошибочно. Поэтому "дубликаты" нередко оказываются неидентичными. Это приводит к дезинформации пользователей и к неверным управленческим решениям.
Принципиальное **решение** этой проблемы — "обобществление" данных предприятия, создание единого хранилища данных для всех подразделений.

▼ **Зависимость программ от физических форматов данных**

Каждая программа ФСОД содержит определения физических форматов всех используемых файлов. Если потребуется изменить структуру записи какого-то файла, то придётся перекомпилировать и заново отладить все программы, обрабатывающие этот файл. Придётся модифицировать даже те программы, которые не обрабатывают изменённое поле. Изменять структуры файлов приходится довольно часто, для увеличения скорости обработки данных или в связи с изменениями требований пользователей.

Поэтому бóльшая часть прикладных программ ФСОД постоянно находится в состоянии отладки.

▼ **Разделение и изоляция данных**

Каждый файл ФСОД содержит записи о каких-то объектах или фактах одного типа. Отношения между объектами представлены в системе обработки данных какими-то связями записей. Т
О связях записей различных файлов знают только прикладные программы ФСОД, выполняющие их совместную обработку. В общем случае любая программа ФСОД, извлекающая информацию из нескольких разнотипных записей, должна выполнять синхронную обработку соответствующих файлов. Синхронная обработка файлов — типовое действие. Конкретные особенности реализации скрываются в параметрах синхронизации, т.е. в связях записей.Функция синхронной обработки должна получать откуда-то сведения о структурах записей обрабатываемых файлов. ФСОД не имеет службы, которая могла бы предоставить эти сведения.

▼ **Невозможность обработки произвольных запросов**

Каждая программа ФСОД создаётся для поддержки какой-то одной функции или нескольких связанных функций. Все программы ФСОД обрабатывают предопределённые запросы к данным. Накопленные в файлах ФСОД данные используются в процессе оперативного управления предприятием. Информационные потребности управляющего зависят от текущей ситуации. Поэтому система должна уметь принимать и обрабатывать произвольные запросы на выборку данных. В рамках ФСОД для каждого запроса приходится писать специальную программу. К моменту её окончательной отладки управляющего уже не интересуют данные, которые она может извлечь. Изменилась ситуация.

▼ Чем обусловлены недостатки ФСОД

Данные, с которыми работает подразделение, рассматриваются как его внутренний ресурс.

Считается, что каждое подразделение самостоятельно накапливает, хранит и обрабатывает всё необходимое. Такой подход приводит к избыточности данных. Опасным следствием этого является возможность появления
противоречий в полной совокупности хранимых данных. ФСОД не имеет никаких средств контроля согласованности данных различных подразделений.

Программы ФСОД получают доступ к данным, используя методы доступа конкретной операционной системы (ОС).

Определения файлов хранятся в телах программ и неизбежно дублируются в различных программах. Кроме того, программы содержат ссылки
на конкретные устройства ввода/вывода. Поэтому любое изменение структур
файлов, обновление ОС или внешних устройств приводит к необходимости
изменения и перекомпиляции программ

Программы ФСОД реализуются и исполняются как автономные функциональные единицы, выполняющие только предопределённую обработку данных.

▼ B2. Концепция системы с базами данных (СБД)

Не ориентированность на конкретное предприятие.

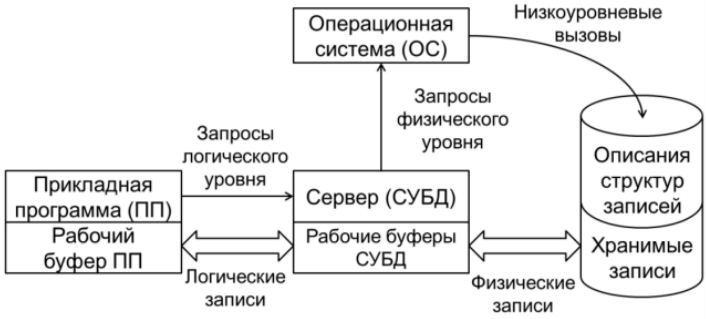
Система БД - человеко-машинная система, предназначенная для поддержания динамической модели ПО и коллективного многоцелевого использования данных.

Детали организации данных во внешней памяти и методов доступа к данным должны быть **скрыты от прикладных программ.**

Описания структур данных должны **сохраняться отдельно** от программ обработки данных.

Прикладные программы будут содержать только ссылки **логического уровня.** Все физические записи будет обрабатывать сервер, формируя из них

запрошенные приложением логические записи.



▼ Система баз данных

▼ Компоненты СБД

БД и СУБД образуют ядро системы баз данных (СБД). Помимо БД и СУБД в систему включаются: аппаратура, приложения документация и пользователи.

- СБД является центральным хранилищем информации предприятия и интсрументальными средствами поиска и анализа информации (вся совокупность данных предприятия должна рассматриваться как единый информационный ресурс. Управление этим ресурсом должно быть централизованным.)
- СБД предоставляет доступ к данным одновременно и независимо многим пользователям (каждый пользователь имеет иллюзию индивидуальной работы)

▼ Пользователи СБД

- **конечные пользователи** (работники предприятия, использующие данные для выполнения служебных обязанностей. Не имеет прямого доступа к БД организации. Для каждого КП создаётся прикладная программа (приложение), поддерживающая специально для него разработанный интерфейс (интерфейс конечного пользователя). Интерфейс КП, во-первых, создаёт удобную рабочую среду, во-вторых, ограничивает возможности доступа к данным.);

- **прикладные программисты** (пользователи системы, создающие и сопровождающие программы для конечных пользователей. Они не работают
  с данными непосредственно. Их задачи – реализация алгоритмов обработки
  данных и создание удобного интерфейса конечного пользователя)
- **администратор базы данных** (группа специалистов, проектирующих, реализующих и сопровождающих систему. АБД несёт всю ответственность за функционирование и развитие системы)

▼ Документация

Документация содержит различные инструкции и правила, которые должны учитываться при проектировании и эксплуатации базы данных. Она обязательно должна включать описания:
- правил регистрации пользователей в системе;
- правил использования инструментов СУБД и приложений;
- процедуры запуска и останова системы;
- процедуры создания резервных копий БД;
- процедур обработки сбоев аппаратуры и программ;
- процедур реструктурирования и реорганизации БД;
- способов улучшения производительности системы;
- методов архивирования данных.

Документация создаётся вместе с системой и поддерживается в актуальном состоянии администратором базы данны

▼ Приложения

Прииложение обеспечивает интерфейс КП с СУБД. Создаёт проекцию данных на проблемную область пользователя. СУБД обеспечивает доступ приложений к БД на логическом уровне.

Приложением - входящий в состав СБД комплекс программных средств, предназначенный для поддержки служебных функций определённой группы конечных пользователей. Состоит из запросов, форм, отчётов, меню и прикладных программ.

Запрос — это требование на выборку из БД подмножества данных, удовлетворяющих определённым условиям. Запрос записывается на языке манипулирования данными, сохраняется в БД и может быть в дальнейшем активирован пользователем или прикладной программой по мере необходимости.

Форма — это способ отображения данных пользователя в виде, удобном для восприятия. Формы используются для просмотра, ввода и редактирования записей.
Отчёт — это результат обработки одного или нескольких запросов, отображённый в стандартной форме документа. Отображаемые данные сгруппированы в разделы и часто агрегированы. Отчёт содержит заголовки, колонтитулы и прочие атрибуты оформления документа. Приложение может напечатать отчёт, сохранить в архиве, отправить по электронной почте, НО оно не может изменить какие-либо значения данных в отчёте. Отчёт отображает состояние БД на момент его создания.

Меню — это средства доступа к функциям приложения.
Прикладные программы пишутся на входном языке СУБД.

▼ Аппаратное обеспечение

Аппаратное обеспечение — это совокупность технических средств, обеспечивающих функционирование программ системы.

- **Система с локальным доступом** разворачивается на одном компьютере, к которому подключаются рабочие станции (PC) пользователей. Представляет собой простой терминал, например, дисплей с клавиатурой, не способный выполнять какую-либо обработку данных. Он используется лишь для запуска приложения и ввода/отображения данных.
- **Системы с удалённым доступом** разворачиваются на компьютерных сетях. Можно выделить два класса таких систем: системы централизованных баз данных (СЦБД) и системы распределённых баз данных (СРБД):
- **Системы централизованных баз данных**
  СЦБД имеет специальный выделенный серверный узел сети, на

котором размещается база данных, доступная для любого узла – клиента. При этом любой клиентский узел может иметь собственную БД, недоступную для других узлов.

- **с файловым сервером** на каждом клиентском узле размещается копия СУБД и приложение пользователя. База данных размещена на серверном узле. Файлы БД контролируются ОС сервера.*Вся обработка содержимого файлов выполняется СУБД клиента*. Файл является единицей сетевого обмена. Сетевой трафик в СЦБД с файловым сервером очень высок. Время реакции системы на запрос клиента может оказаться неприемлемым для пользователя.
- **с сервером БД** на клиентских узлах разворачиваются только приложения пользователей. На серверном узле хранится централизованная БД и функционирует единственная в системе копия СУБД. *Сервер БД обеспечивает выполнение основного объёма обработки данных*. Запрос, выдаваемый клиентом, порождает выборку данных на сервере. Результаты выборки (но не файлы!) передаются по сети от сервера к клиенту.
- **с сервером приложений** на серверном узле, кроме СУБД, размещаются и прикладные программы. Клиенты направляют по сети запросы на запуск приложений. *Сервер выполняет всю обработку данных и направляет клиентам результаты работы приложений.* Нагрузка на клиентские узлы сводится к функциям отображения этих результатов. Сетевой трафик в СЦБД с сервером приложений значительно ниже, чем в СЦБД с сервером базы данных, однако требования к вычислительным ресурсам серверного узла значительно выше.
- **Системы распределённых баз данных** не имеют выделенного узла. Все узлы
  сети равноправны. На каждом развёрнута полноценная локальная СБД (ЛСБД). Пользователь любого узла имеет доступ к данным, размещённым на любом узле, и работает с ними, как с собственными.

Распределённая БД — это виртуальная база данных. Её «реальные» части физически размещены на удалённых узлах. Каждый узел имеет собственных локальных пользователей, собственную локальную СУБД и,

кроме того, программный компонент, обеспечивающий согласованную работу локальных СБД. Комбинацию этого компонента и локальной СУБД называют распределённой системой управления базами данных.

▼ Структурные единицы базы данных

Поле – это элементарная именованная единица логической организации данных, которая соответствует неделимой единице информации — (атрибуту, реквизиту).

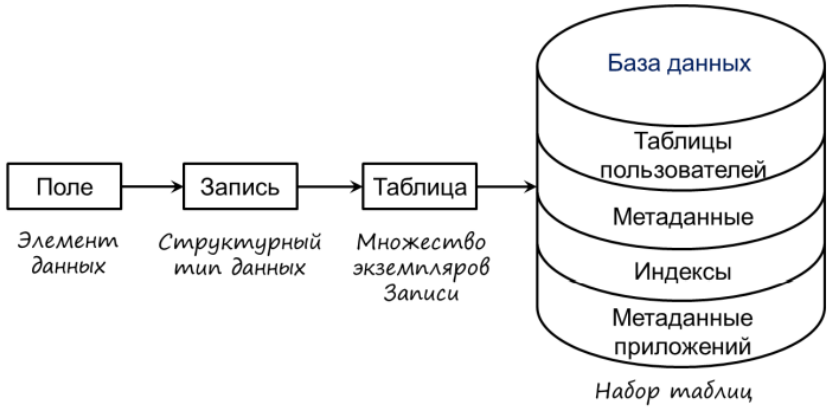
Для описания поля обязательны следующие характеристики:
- имя;
- тип (множество допустимых значений);
- длина, (максимально возможное количество символов);
- точность (для числовых данных).

Запись — именованная совокупность логически связанных полей

Экземпляр записи – это отдельная реализация записи, содержащая конкретные значения её полей (значение структурного типа данных.

Таблица — это совокупность экземпляров записей одного типа.

Структурные единицы БД представлены на рис. 3.10.



▼ **В3. 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. С её помощью централизованно происходит создание баз данных, их поддержка и обеспечение доступа пользователей к данным в СБД.

(Задачи СУБД) СУБД должна обеспечить:

1. доступ прикладным программам к данным на логическом уровне;
2. контроль согласованности (целостности) данных;
3. управление многопользовательским доступом к данным;
4. контроль доступа/защиту от несанкционированного доступа;
5. защиту данных от разрушений вследствие аварий;
6. поддержку среды разработки ПП;
7. поддержку среды обслуживания набора данных.

▼ Основные подсистемы СУБД

▼ Подсистема проектирования

Предназначена для создания баз данных, обеспечения интерфейсов всех категорий пользователей и проектирования приложений.
- 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.
Многие современные СУБД имеют визуальные средства определения структур и связей данных, создания форм, запросов и отчётов.

▼ Подсистема обработки

Подсистема обработки 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.*

▼ Ядро СУБД

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

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.*
Ядро участвует в управлении транзакциями, санкционировании доступа к данным, резервном копировании и восстановлении БД.

▼ B4. База данных

▼ Предметная область

Базой данных называют совокупность записей, необходимых для осуществления какой-то организованной деятельности. В этих записях содержится представляющая интерес для владельца информация — сведения об объектах и отношениях объектов, попадающих в сферу деятельности.

Часть реального мира, сведения о которой представляют интерес с точки зрения владельца базы данных, называется предметной областью (ПО) базы данных.

▼ Понятие базы данных в информатике

База данных полезна, если она:
- содержит всю необходимую с точки зрения владельца информацию о предметной области;
- обеспечивает возможность упорядочивать информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков.

Для того чтобы это было возможно, данные (независимо от способов хранения!) структурируют, выделяют поисковые признаки и создают необходимые для сортировки и поиска указатели (индексы).

В предметной области происходят события, изменяющие её состояние. Появляются новые экземпляры объектов и отношений, исчезают существовавшие ранее, изменяются свойства существующих экземпляров. Эти события отражаются на состоянии базы данных. Создаются или удаляются экземпляры записей, изменяются значения полей существующих экземпляров.

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

▼ Определение термина «база данных»

Компьютерная БД отличается от любого другого набора записей тем, что наряду с данными пользователей содержит своё собственное описание. В информатике принято следующее определение термина:

База данных (БД) — это самодокументированная интегрированная совокупность записей.

Самодокументированность означает, что вместе с данными пользователей в БД содержится описание её собственной структуры. Это описание называется метаданными. Метаданные сохраняются в специальном разделе БД. Он называется каталогом данных или, что то же самое, словарём данных.

Интегрированность означает, что БД наряду с записями пользователей содержит сведения о связях записей. Обычно связи записей представляют индексами. Индекс – это служебная запись. Кроме индексов многие современные системы сохраняют
в БД метаданные приложений – определения структур форм ввода данных и
отчётов и т.п.

В общем случае БД предприятия отражает представления множества различных пользователей о различных аспектах ПО. Поэтому часто БД предприятия организуется как совокупность БД отдельных подразделений. Однако, в отличие от ФСОД, все эти БД подчинены единому управлению.

▼ Структурные единицы базы данных

Поле – это элементарная именованная единица логической организации данных, которая соответствует неделимой единице информации — (атрибуту, реквизиту).

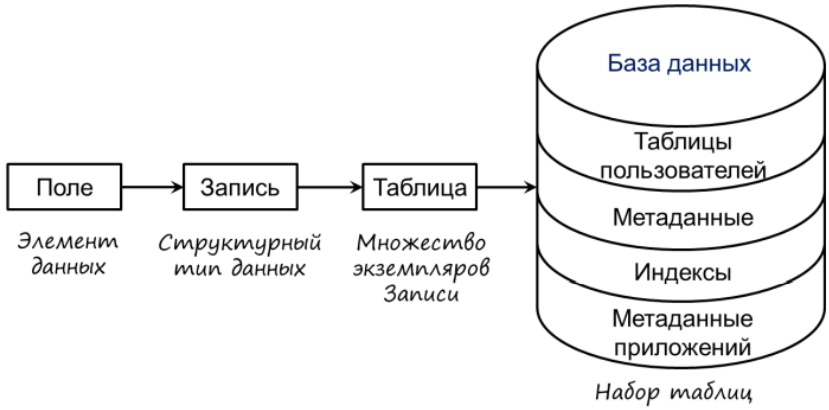
Для описания поля обязательны следующие характеристики:
- имя;
- тип (множество допустимых значений);
- длина, (максимально возможное количество символов);
- точность (для числовых данных).

Запись — именованная совокупность логически связанных полей

Экземпляр записи – это отдельная реализация записи, содержащая конкретные значения её полей (значение структурного типа данных.

Таблица — это совокупность экземпляров записей одного типа.

Структурные единицы БД представлены на рис. 3.10.



▼ Ключи

В *структуре записи выделяют поля (группы полей), значения которых используются для идентификации экземпляров записи.* Различают первичные и вторичные ключи.

Первичный ключ (Primary Key, PK) – это поле (или группа полей), значения которого принципиально не могут быть одинаковыми в различных одновременно существующих экземплярах записи; уникальный идентификатор записи.

В качестве вторичного ключа (поискового поля) может быть выбрано любое поле, по значениям которого приходится часто искать или

сортировать экземпляры записи.

Внешний ключ (Foreign Key, FK) – это поле (группа полей) записи R1, значения которого являются ссылками на существующие экземпляры записи R2.

▼ Представление метаданных

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.

▼ Индексы

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

Индекс поддерживает прямой доступ к записям по ключу (PK, FK, SK) и *последовательный просмотр записей в порядке возрастания или убывания*
значений ключа. 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.*
Индексы бывают *уникальными и неуникальными. Уникальные* создаются для полей, значения которых не могут дублироваться в существующих записях таблицы. *Неуникальные могу*т создаваться для любых других полей, если по их значениям нужно часто сортировать или отфильтровывать записи.

Индексы бывают простыми и составными. Составной индекс для базовой таблицы создаётся, если нужно выполнять поиск и сортировку записей по наборам значений какой-то группы полей, например, по составным первичным или внешним ключам.

▼ B5. Архитектура ANSI/SPARC

▼ 4.1 Уровни представления данных (Архитектура ANSI/SPARC)

БД и программные средства их создания и ведения (СУБД) имеют трёхуровневое строение (см. рис. 4.1). Различают концептуальный, внутренний и внешний уровни представления данных БД, которым соответствуют одноимённые схемы (описания).

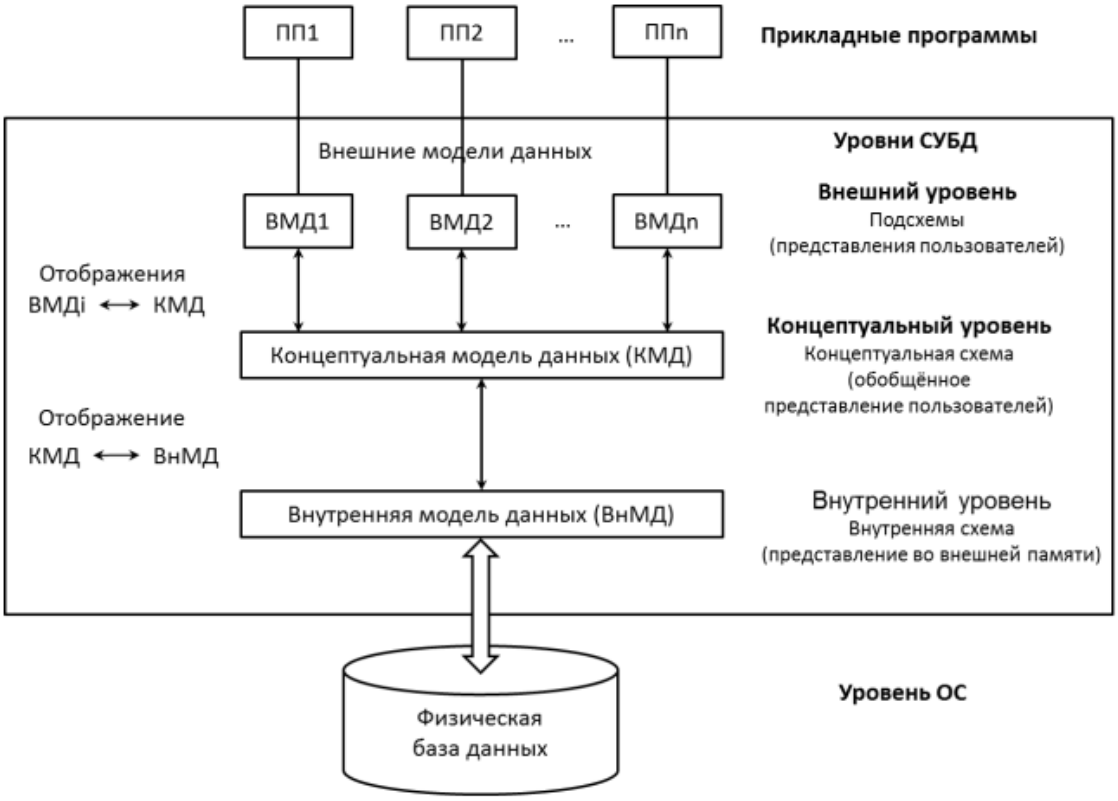


Рис. 4.1 – Архитектура ANSI/SPARC

ANSI – American National Standard Institute (Национальный институт стандартизации США), SPARC – Standards Planning and Requirement Committee (Комитет планирования стандартов и норм).

▼ 4.1.1 Концептуальный уровень

Центром архитектуры является концептуальный уровень. Он соответствует логическому представлению данных ПО в обобщённом виде без какихлибо ссылок на реализацию.

Концептуальная схема состоит из определений типов концептуальных записей, их связей, ограничений целостности данных и других типов объектов.

> В состав концептуальной записи входят только семантически значимые поля. Никаких служебных полей, видимых пользователю, в её структуре нет.

На концептуальном уровне представлена также семантическая информация о данных и сведения о мерах по обеспечению безопасности и поддержки целостности данных.

> Концептуальный уровень представления данных можно понимать как множество экземпляров концептуальных записей. Они физически не существуют. СУБД воспроизводит их из экземпляров внутренних записей, обрабатывая запросы пользователей.

▼ 4.1.2 Внутренний уровень

Внутренний уровень описывает организацию данных в среде хранения и соответствует физическому представлению данных.

> Внутренняя схема состоит из определений типов внутренних (хранимых) записей, файлов внешней памяти, индексов и других компонентов уровня реализации. Эти определения выполняются на входном ЯВУ СУБД.

Все семантически значимые поля концептуальных записей представлены на внутреннем уровне. Однако в общем случае между внутренними и концептуальными записями не существует соответствия «один к одному». Различные поля одной и той же внутренней записи могут соответствовать полям нескольких концептуальных записей. И наоборот, различные поля одной и той же концептуальной записи могут соответствовать полям

различных внутренних записей. Могут различаться типы и длины соответственных полей. Внутренние записи кроме семантически значимых и системных полей содержат служебные поля, созданные разработчиками БД для своих целей.

Решения о составе и структуре внутренних записей принимают исходя из соображений эффективности обработки запросов. Они могут пересматриваться в процессе эксплуатации системы. Однако в целом между концептуальной и внутренней схемами существует взаимно однозначное соответствие. Поэтому СУБД всегда может сформировать требуемый экземпляр концептуальной записи из значений полей внутренних записей и наоборот.

Кроме схемы на внутреннем уровне представлена информация

о распределении дискового пространства,

о размещении файлов на физических носителях,

о сжатии данных и методах шифрования.

> Таким образом, множество экземпляров внутренних записей и есть база данных с «точки зрения» СУБД. Именно их она запрашивает у операционной системы.

▼ 4.1.3 Внешний уровень

Внешний уровень поддерживает частные представления данных, необходимые конкретному пользователю. Каждая внешняя схема (подсхема) является частью (подмножеством) концептуальной схемы или выводится из неё посредством однозначных преобразований.

Внешние записи формируются из полей концептуальных записей, но они представляют только те сущности, атрибуты и связи ПО, которые представляют интерес для конкретного КП. Этот КП может даже не подозревать о том, что в БД есть и другие сведения.

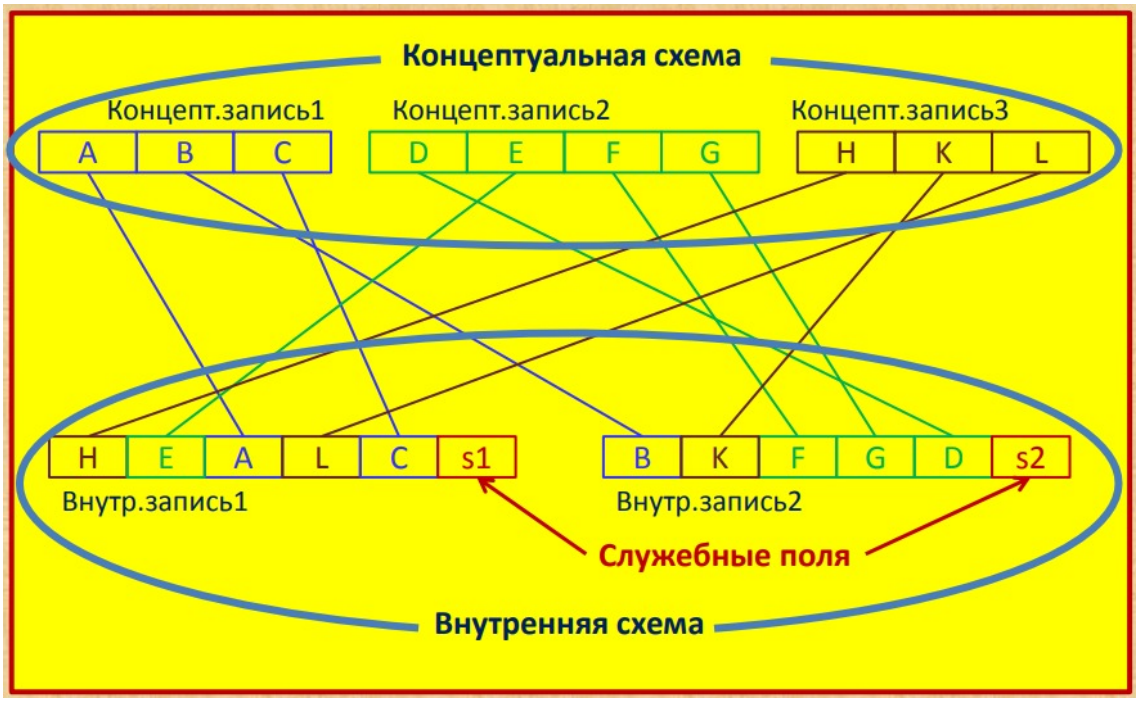
Различные внешние схемы могут по-разному отображать одни и те же данные. Внешние записи могут включать вычислимые и производные поля, не хранящиеся в БД как таковые. Поля различных концептуальных

записей могут отображаться в одной внешней. Структуру и состав внешних записей проектируют исходя из требований конечного пользователя.

Кроме внешних схем на этом уровне представлены определения форм, отчётов, запросов и другие компоненты приложений. С помощью внешних схем поддерживается интерфейс конечного пользователя и санкционированный доступ к данным и приложениям.

▼ 4.1.4 Отображения

Уровни представления данных связаны однозначными отображениями. Отображение определяет соответствие между представлениями верхнего и нижнего уровней. Отображение концептуальный внутренний ставит в соответствие концептуальным полям и записям хранимые. Это соответствие является взаимно однозначным. При изменении структур хранения отображение концептуальный внутренний изменяется так, чтобы концептуальная схема осталась неизменной. Аналогично отображение внешний концептуальный ставит в соответствие концептуальным полям и записям внешние.



▼ 4.1.5 Системный каталог

Определения данных всех уровней сохраняются в системном каталоге СУБД. Он представляет собой внутреннюю базу данных СУБД, предметной областью которой является система баз данных. В системном каталоге сохраняются:

- определения типов записей всех уровней;
- описания отображений;
- сведения о пользователях системы и их привилегиях (правах на использование данных и приложений);
- сведения о программных и аппаратных ресурсах системы;
- и другие сведения, необходимые для управления системой.

> Системный катало́г СУБД – это информационное ядро системы, обеспечивающее согласованную работу всех её компонентов.

▼ 4.1.6 Независимость приложений от данных

Основное назначение трёхуровневой архитектуры – обеспечить независимость приложений от данных.

> Требование независимости от данных означает, что никакие изменения на нижних уровнях не должны влиять на верхние уровни представления данных.

Различают два типа независимости: логическую и физическую.

> Логическая независимость - есть полная защищённость существующих внешних схем от любых изменений, вносимых в концептуальную схему.

Такие изменения концептуальной схемы, как добавление или удаление типа концептуальной записи, поля или связи должны производиться без

необходимости изменения существующих внешних схем или приложений. Они должны быть видны только тем пользователям, для которых предназначались. Все прочие пользователи не должны даже подозревать о них.

> Физическая независимость - есть полная защищённость концептуальной схемы от любых изменений, вносимых во внутреннюю
> схему.

Наиболее распространённой причиной внесения изменений во внутреннюю схему является недостаточная производительность системы. Для её повышения приходится вносить во внутреннюю схему такие изменения, как расщепление или слияние внутренних записей, модификация индексов, использование других файловых систем или устройств хранения. Все эти изменения должны осуществляться без необходимости внесения изменений в концептуальную схему.

Независимость достигается за счёт модификации отображений при изменении схем нижних уровней. Отображение реализуется с помощью таблиц системного каталога, хранящих сведения о соответствии полей и записей верхнего и нижнего уровней. Для модификации отображения достаточно обновить данные в этих таблицах.

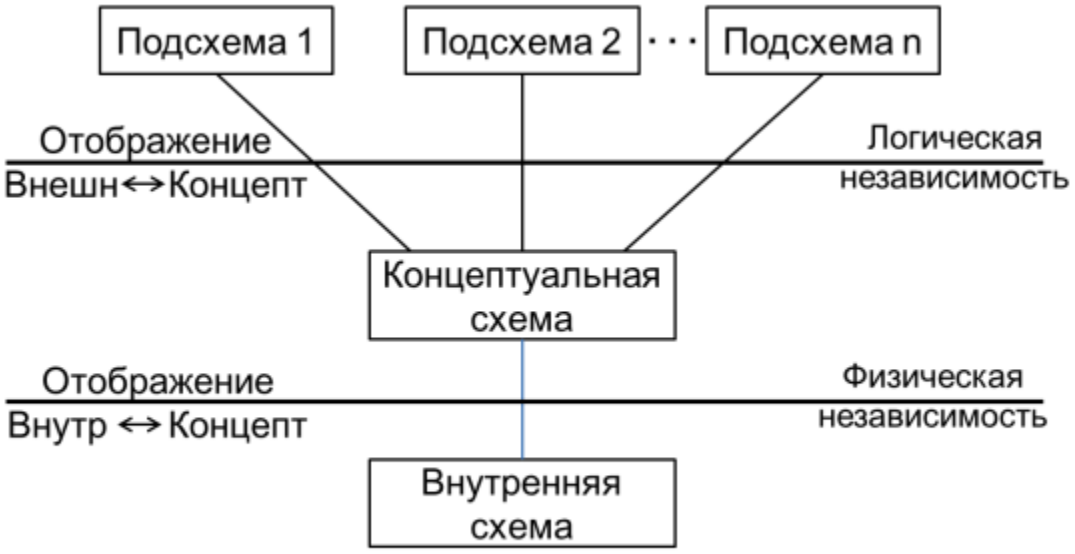


Рис. 4.3 – Уровни независимости от данных

▼ B6. Операции в базе данных

▼ 4.2 Дисциплина доступа приложений к хранимым данным

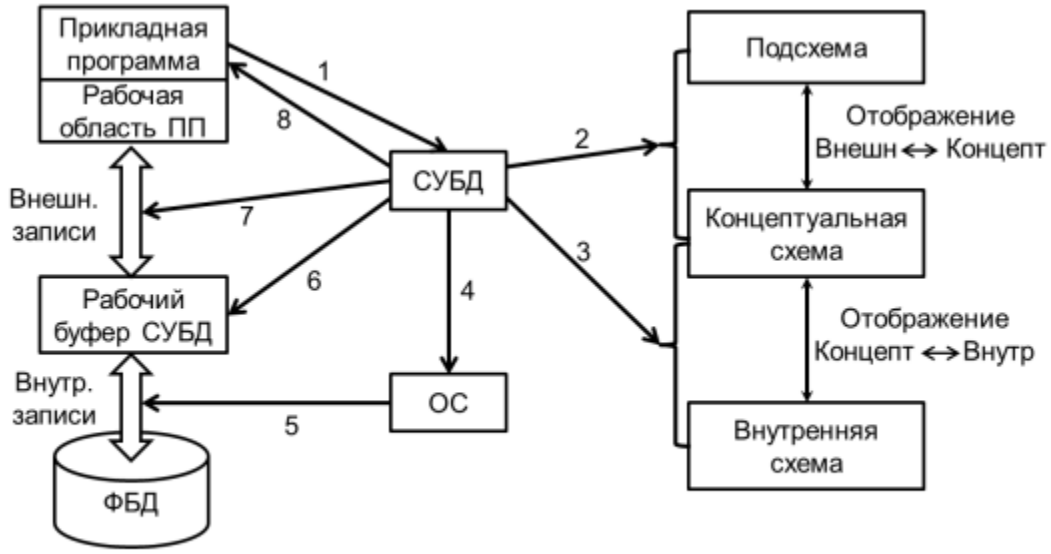


Рис. 4.4 – Схема обработки запроса на извлечение данных

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

Шаг 2. СУБД, используя внешнюю и концептуальную схемы и описание отображения внешний концептуальный, определяет, какие записи концептуального уровня необходимы для формирования требуемых внешних записей.

Шаг 3. СУБД, используя концептуальную и внутреннюю схемы и описание отображения концептуальный внутренний, определяет, какие внутренние записи необходимы для формирования затребованных концептуальных записей и совокупность физических записей, которые должны быть для этого считаны с физического носителя.

Шаг 4. СУБД выдаёт ОС запрос на считывание в свои буферы необходимых записей физической базы данных (ФБД).

Шаг 5. ОС считывает затребованные записи и помещает их в системные буферы СУБД.

Шаг 6. На основании схем и описаний отображений СУБД формирует в своём буфере затребованные внешние записи.

Шаг 7. СУБД пересылает сформированные внешние записи в рабочую область (РО) ПП.

Шаг 8. СУБД передаёт ПП сообщение об исполнении запроса.

▼ 4.3 Операции обработки данных

В процессе исполнения запросов приложений СУБД выполняет четыре разновидности высокоуровневых (описанных средствами входного ЯМД) операций обработки данных.

Извлечение записей (RETRIEVE) – считывание в рабочий буфер совокупности записей ФБД и формирование в буфере совокупности внешних записей, затребованных приложением.
Возможны: сортировка, группирование записей, агрегирование данных

Извлечённое множество записей может обновляться приложением. СУБД выполняет следующие операции обновления

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. При этом записи могут либо физически уничтожаться, либо помечаться как удалённые, но физически сохраняться в буфере.

> Все эти операции изменяют только состояние рабочего буфера СУБД.

В физической базе данных новое состояние может быть зафиксировано, только если оно удовлетворяет всем ограничениям целостности.

▼ 4.4 Дисциплина обменов с внешней памятью

Записи ФБД, считанные в рабочий буфер СУБД по запросу какого-либо приложения, не удаляются из буфера по исполнении запроса.
СУБД, приняв запрос приложения, проверяет, есть ли в буфере данные, необходимые для его исполнения. Если данные есть, то они не извлекаются повторно.

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. В противном случае содержимое рабочего буфера выталкивается во внешнюю память и буфер частично очищается. Из него удаляются редко используемые данные. Для того чтобы иметь возможность сделать это, система накапливает статистику обращений к данным.

Такая дисциплина обменов позволяет существенно уменьшить среднее время реакции системы на запрос приложения

▼ 4.5 Операции обновления и целостность данных.

Данные в БД должны быть полными и достоверными; гарантировать это невозможно, однако полные и достоверные данные всегда имеют

осмысленную интерпретацию, обратное является неверным.

Интерпретируемый в терминах ПО набор данных называют целостным.

Целостный набор данных ограничен:

- свойствами объектов ПО,
- свойствами отношений объектов,
- правилами ПО (бизнес-правилами)
- соображениями здравого смысла.

> Ограничение целостности (ОЦ) - предикат, определённый на объектах БД.

СУБД может обеспечить целостность БД в пределах заданных ОЦ.

Уровни ОЦ:

- Уровень атрибута
- Уровень экземпляра записи
- Уровень таблицы
- Уровень базы данных

СУБД должна гарантировать целостность набора данных пользователя в пределах заданных ОЦ.

Механизмы поддержки целостности:

- Внутренний
  Реализован в ядре СУБД. Обеспечивает контроль целостности атрибута, уникальности записи и целостности ссылок.
  ОЦ объявляются средствами DDL (Date Definition Language), сохраняются как объекты БД. Используются СУБД при попытках обновления состояния буфера.
- Внешний
  Реализуется проектировщиком БД. Обеспечивает контроль ОЦ, которые невозможно объявить средствами DDL.

Виды ОЦ:

Немедленно проверяемое - Проверяется в момент завершения операции обновления, которая может нарушить это ограничение.

С отложенной проверкой - Заведомо нарушается одиночной операцией обновления. Проверка возможна только по завершении серии обновлений.

В общем случае целостность БД может быть проверена лишь по исполнении логически завершённой последовательности операций обновления.

▼ 4.6 Транзакции в базе данных.

Транзакция - последовательность действий над базой данных, выполняемая по запросам одного пользователя или приложения и переводящая БД из согласованного начального состояния в согласованное конечное состояние.

Может быть представлена:

- отдельной программой
- частью программы
- последовательностью операторов ЯМД
- отдельным оператором

Транзакция есть неделимая единица работы в БД.

Подчеркнём, что промежуточные состояния БД могут быть несогласованными.

Способы завершения транзакции:

COMMIT TRANSACTION (фиксация) - произведённые транзакцией обновления данных становятся частью БД.

ROLLBACK TRANSACTION (откат) - произведённые транзакцией обновления данных отменяются. БД возвращается в исходное состояние.

Свойства транзакции (свойства АСИД):

- Атомарность

Гарантирована неделимость транзакции. Либо выполняются ВСЕ её операции, либо не выполняется НИ ОДНА.

- Согласованность
  Гарантирована согласованность состояния БД к моменту начала транзакции, и после её завершения.
- Изолированность
  Гарантирована эквивалентность результатов параллельного исполнения любого набора транзакций результатам последовательного исполнения того же набора в любой последовательности.
- Долговечность
  Гарантировано сохранение результатов работы транзакции, завершившейся фиксацией.

СУБД должна обеспечивать поддержку свойств АСИД. Иначе она не является системой управления базами данных.

▼ B7. Управление доступом к данным

4.7 Управление доступом к данным

▼ 4.7.1 Принципы ограничения доступа.

Право служащего предприятия получать данные из БД и манипулировать ими естественно ограничивается его служебными обязанностями. Любая система санкционирования доступа к данным базируется на двух принципах.

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.

В компьютерной системе эти принципы реализуются механизмом авторизации (подсистемой доступа). Подсистема доступа обеспечивает предоставление прав (привилегий) доступа к системе и её объектам.

Пользователем считается зарегистрированный в системном каталоге идентификатор авторизации (User’s ID), который может быть присвоен лицу, группе лиц или прикладной программе.

Привилегия – это право пользователя выполнять определённое действие над определённым объектом БД: запускать приложение, просматривать таблицу, вставлять строки в таблицу и т.п.

Ответственность за авторизацию пользователей возлагается на администратора системы. В его обязанности входит создание учётных записей пользователей. В учётной записи указывается:

идентификатор авторизации – имя, под которым пользователь будет известен системе;

тип пользователя – лицо, группа или прикладная программа;

описание – ФИО, должность лица, наименование группы, имя программы, её назначение и т.п.

Учётные записи сохраняются в системном каталоге

▼ 4.7.3 Аутентификация.

С каждым идентификатором связывается пароль, выбираемый пользователем и известный только ему. Пароль сохраняется в зашифрованном виде
в защищённой части системного каталога и недоступен для просмотра никакому пользователю, в том числе и владельцу. Пароль используется при подключении пользователя к системе для аутентификации, т.е., для проверки того, является ли пытающийся подключиться тем, за кого он себя выдаёт.

Существуют и другие способы аутентификации, однако парольный наиболее распространён, хотя и не даёт абсолютной гарантии аутентификации. Чтобы уменьшить вероятность раскрытия пароля, обычно вводят ограничения на его минимальную длину, на состав символов, обязывают пользователей периодически изменять пароли, запрещают использовать в паролях собственные имена и применяют другие административные ограничения.

▼ 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. Он может использовать его в любых операциях. **Объектная** привилегия это право использования объекта в операциях определённого типа. Например, право просмотра конкретной таблицы или её отдельных столбцов, или некоторого подмножества её строк, право вставки строк в конкретную таблицу, право запуска приложения и т.п.

Говоря обобщённо, объектная привилегия есть тройка (идентификатор авторизации, объект БД, действие).

Сведения о привилегиях сохраняются в системном каталоге в виде матрицы управления доступом. Здесь двоичные числа – это коды наборов привилегий. Отдельные типы привилегий закодированы так:

RETRIEVE	UPDATE	INSERT	DELETE	ALL
0001	0010	0100	1000	1111

Код набора привилегий есть сумма кодов типов привилегий.

▼ 4.7.5 Подсхемы

Подсхема (внешняя схема) представляет собой набор виртуальных записей, реально не существующих в БД. Эти записи создаются в системном буфере тогда, когда пользователь запускает своё приложение. Подсхема является мощным и гибким средством ограничения доступа к данным. Она отображает для пользователя только ту часть БД, которая необходима ему для работы.

▼ 4.7.6 Сеанс

Сеанс – это промежуток времени между моментом подключения пользователя к системе и моментом отключения. Подключив пользователя, система готова предоставить ему все свои ресурсы в рамках его привилегий. Момент начала, и момент окончания сеанса пользователя регистрируются в системном журнале. Кроме того, система протоколирует в журнале все действия пользователя в течение сеанса во всех деталях.

▼ **B8. Управление параллелизмом**

▼ 4.8.1 Необходимость управления параллелизмом

СУБД одновременно контролирует сеансы и обрабатывает транзакции многих пользователей. Операции различных транзакций чередуются, за счёт чего достигается их параллельное исполнение.

Параллелизм - Одновременный доступ к БД нескольких приложений.

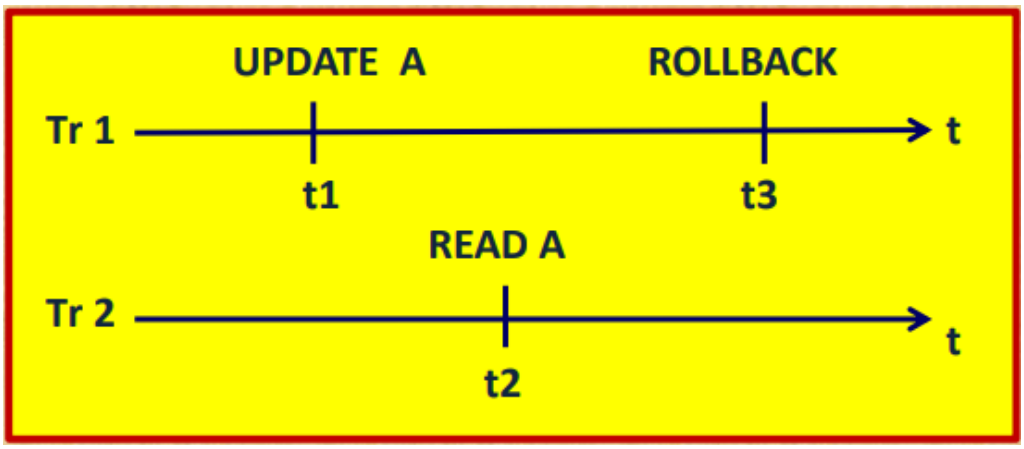
Типы конфликтов параллельного доступа:

- Запись – Запись (W-W) - Транзакция T1 изменила объект и не закончилась. Транзакция T2 изменяет этот объект. Результат - потеря обновления.
- Чтение–Запись (R-W) - Транзакция T1 прочитала объект и не закончилась. Транзакция T2 изменяет этот объект. Результат - неповторяемое считывание
- Запись – Чтение (W-R) - Транзакция T1 изменила объект и не закончилась. Транзакция T2 читает этот объект. Результат - чтение "грязных" данных.

▼ Конфликты параллельного доступа (примеры): (фиг знает нужны или нет)

- Чтение «грязных» данных

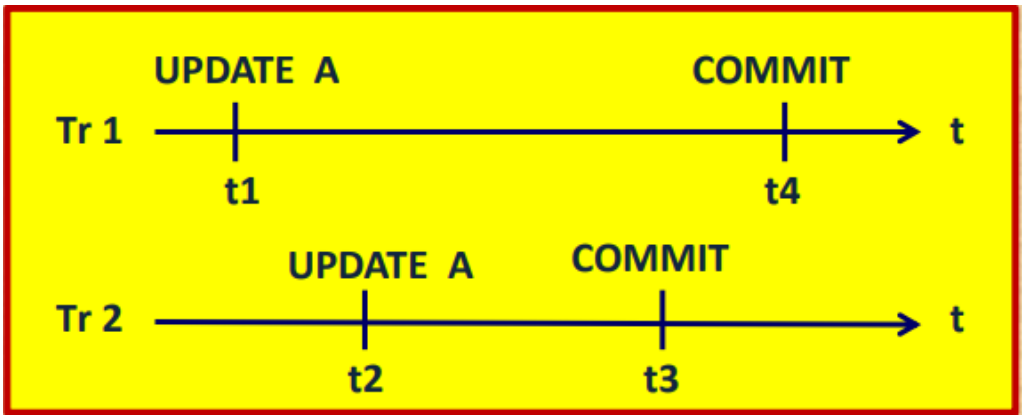
  Эта ситуация может возникнуть, если транзакция имеет доступ к промежуточным результатам другой транзакции.



Грязные чтения невозможны, если до завершения транзакции, изменяющей некоторый объект, никакая другая транзакция не сможет читать этот объект.

- Потеря обновлений

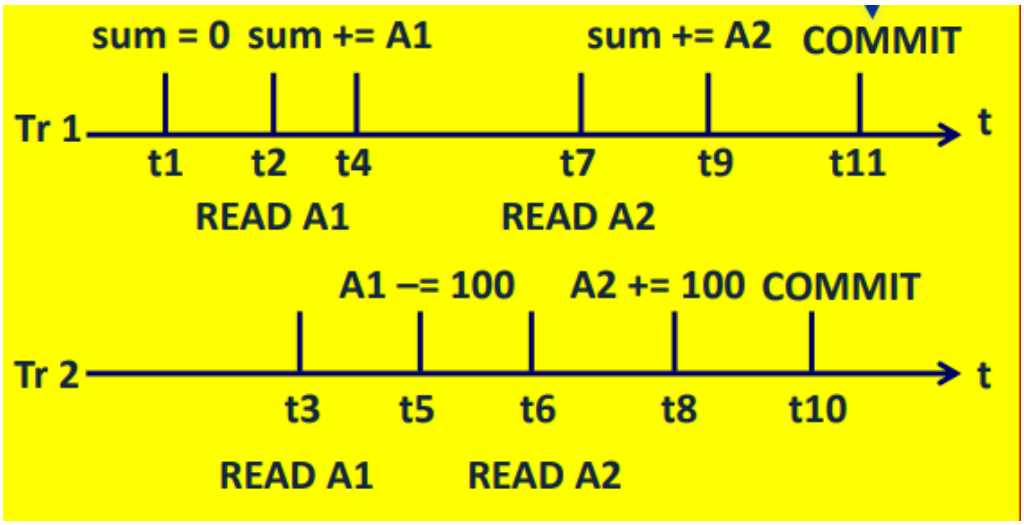
  Эта ситуация может возникнуть, если две транзакции пытаются параллельно обновлять один и тот же объект.



  Избежать потери обновлений можно, если до завершения транзакции, изменяющей объект, никакая другая транзакция не сможет
  его изменять.

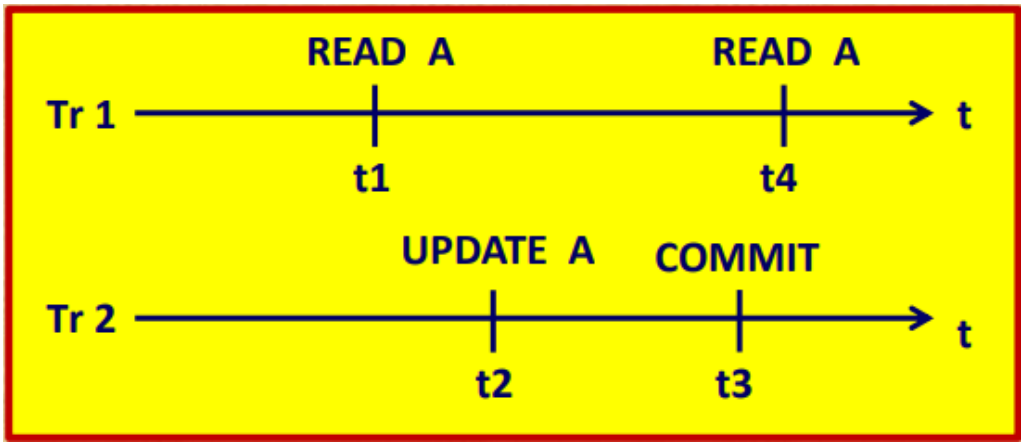
- Несогласованная обработка

  Проблемы с параллельным доступом возникают и в том случае, если одна из двух параллельно исполняемых транзакций обновляет состояние БД, а другая только извлекает данные, и выполняет какие-либо преобразования.



Несогласованная обработка невозможна, если до завершения транзакции, читающей некоторый объект, никакая другая транзакция
не может изменять его.

- Неповторяющиеся чтения



▼ 4.8.2 Изолированность транзакций

Транзакции являются не только единицами работы в БД, но и единицами управления. СУБД должна обеспечить такой режим обработки транзакций, при котором результат их параллельного исполнения совпадал бы с результатом последовательного исполнения в каком-либо порядке. В этом случае транзакции будут изолированы, т.е., никакая из них не будет влиять на результаты других.

Для обеспечения изолированности транзакций система должна поддерживать определённую дисциплину блокировок объектов.

Уровни изолированности транзакций: (на них субд гарантирует)

1. Первый - Невозможность потери обновлений. До завершения транзакции, изменяющей объект A никакая другая транзакция не может его изменять.
2. Второй - Невозможность чтения “грязных данных”. До завершения транзакции, изменяющей объект A никакая другая транзакция не может его читать.
3. Третий - Отсутствие неповторяющихся чтений. До завершения транзакции, читающей объект A никакая другая транзакция не может его изменять.

▼ 4.8.3 Двухфазный протокол блокировки

Прежде чем выполнять какую-либо операцию над объектом базы данных A, транзакция T должна запросить блокировку (захват) A. В зависимости от вида предполагаемой операции объект может быть заблокирован в одном из двух режимов:

S (Shared lock) – разделяемый захват, необходимый для выполнения операции чтения;
X (eXclusive lock) – монопольный захват, необходимый для выполнения операций добавления, удаления и модификации объекта.

Совместимость захватов:

	X	S
X	нет	нет
S	нет	да

Фаза наложения блокировок - Перед выполнением любой операции над объектом R транзакция должна запросить блокировку R в подходящем режиме.

Фаза снятия блокировок - Если транзакция сняла хотя бы одну наложенную ранее блокировку, то она не может запрашивать новые.

Если все транзакции соблюдают двухфазный протокол блокировки, то коллизии потерянных изменений, «грязных» чтений и несогласованной обработки невозможны.

▼ B9. Восстановление базы данных

▼ 4.9.1 Необходимость восстановления

Локальный сбой – это аварийное или принудительное 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. Для восстановления согласованности необходимо устранить изменения данных, произведённые прерванной транзакцией – выполнить индивидуальный откат транзакции.

Мягкий сбой - аварийное прекращение ВСЕХ транзакций. Может произойти, например, вследствие аварийного отключения питания или при возникновении неустранимого сбоя процессора и т.п. В этом случае теряется содержимое оперативной памяти. 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

Служебный файл. Содержит информацию, необходимую для восстановления БД.

О каждой транзакции:

- идентификатор транзакции
- идентификатор инициатора

- время начала
- время завершения
- способ завершения

О каждой операции:

- идентификатор транзакции
- идентификатор объекта обработки
- время начала операции
- копия объекта до операции (UPDATE и DELETE)
- копия объекта после операции (UPDATE и INSERT)

Правила ведения системного журнала:

1. Записи журнала создаются в специальном буфере журнала.
2. Каждая запись имеет временнȕю метку.
3. Буфер журнала принудительно выталкивается во внешнюю память при завершении транзакции.

Протокол WAL (Write Ahead Log) - Результаты зафиксированной транзакции не могут попасть в ФБД раньше, чем соответствующие ей записи журнала.

▼ 4.9.3 Индивидуальный откат транзакции

Сводится к отмене изменений в рабочем буфере СУБД, произведённых прерванной транзакцией.

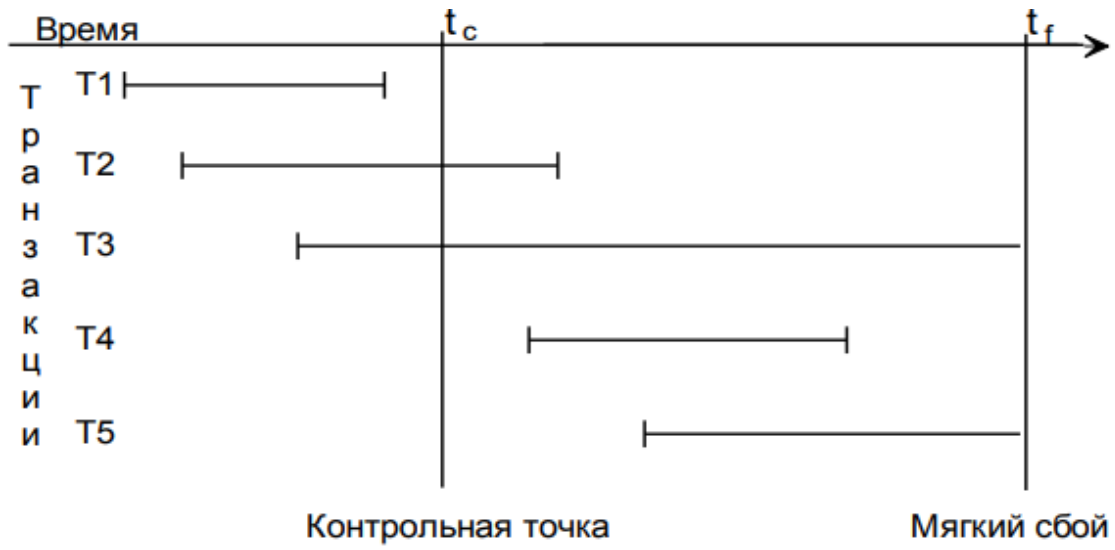
Действия:

- Выбрать из журнала все записи от прерванной транзакции.
- Расположить их в порядке, обратном хронологическому.
- Выполнить в рабочем буфере обратные по смыслу операции.

Процесс восстановления является транзакцией и протоколируется в системном журнале.

▼ 4.9.4 Восстановление после мягкого сбоя

Потеряно содержимое рабочего буфера СУБД. Восстановление возможно, если периодически сохраняется состояние буфера «КАК ЕСТЬ» (Контрольная точка).



При перезагрузке системы СУБД должна

- откатить все прерванные в момент сбоя транзакции и
- выполнить повторно все успешно завершившиеся, но не зафиксированные в ФБД.

Основа для восстановления – состояние буфера на момент принятия контрольной точки.

▼ 4.9.5 Восстановление после жесткого сбоя

Возможно при наличии резервной копии ФБД.

Создание резервной копии по достижении “жёлтой зоны” в файле журнала.

Действия СУБД:

- Запуск новых транзакций не производить.
- Дождаться завершения всех существующих транзакций.
- Вытолкнуть рабочие буферы журнала и БД во внешнюю память.
- Скопировать состояние ФБД на резервный носитель.
- Очистить файл журнала.

Восстановление:

- Восстановить состояние ФБД на момент последнего копирования.
- По текущему журналу повторно исполнить все транзакции, успешно завершившиеся до момента сбоя.