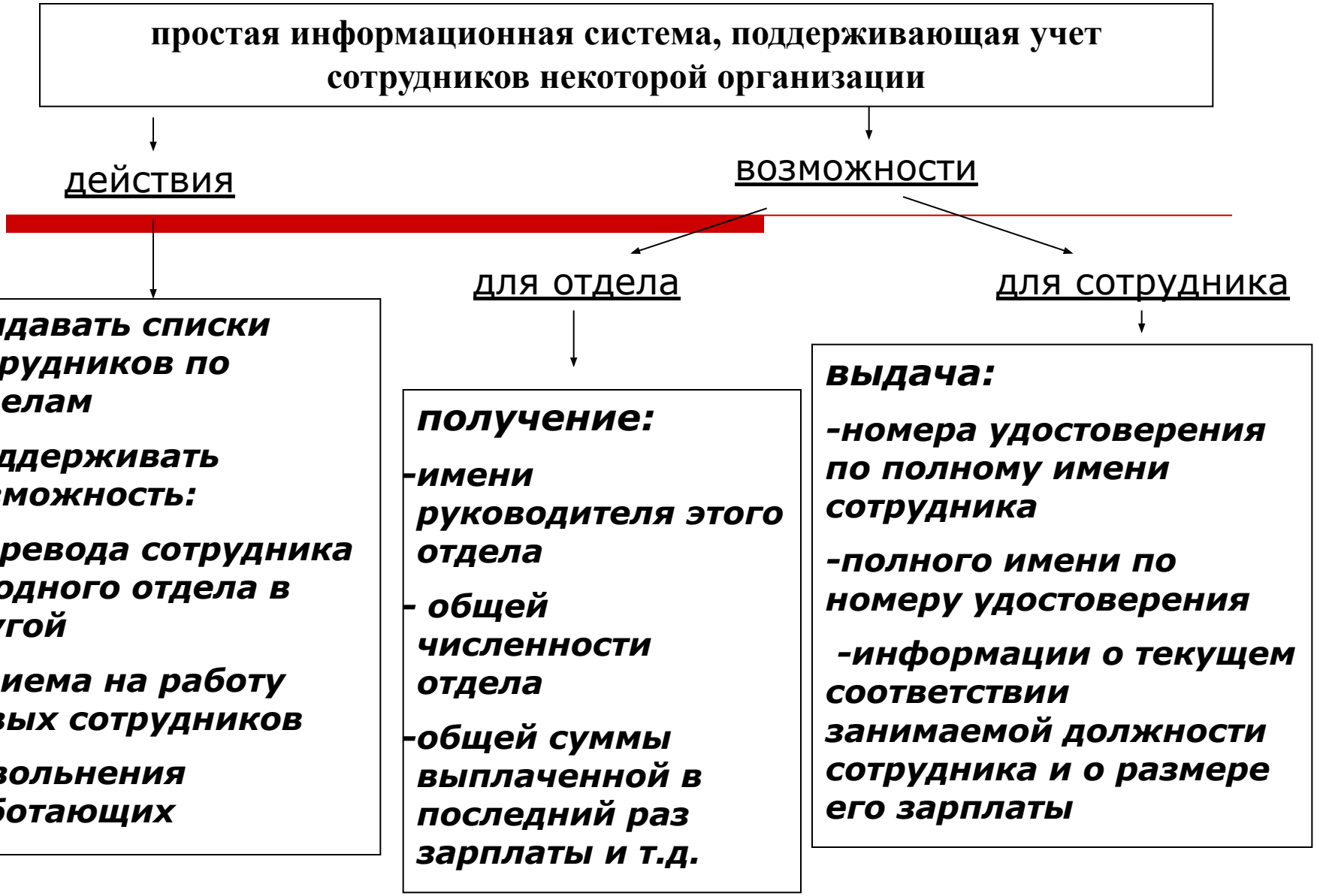
Лекция 2. Уровни представления БД. Свойства БД. Отличия от файловых систем.



мы решили основывать эту информационную систему на файловой системе и пользоваться при этом ***одним файлом***, расширив базовые возможности файловой системы за счет ***специальной библиотеки функций***

минимальная информационная единица - сотрудник

одна запись для каждого сотрудника

поля, *которые* должна содержать запись

-***полное имя сотрудника*** (СОТР_ИМЯ)

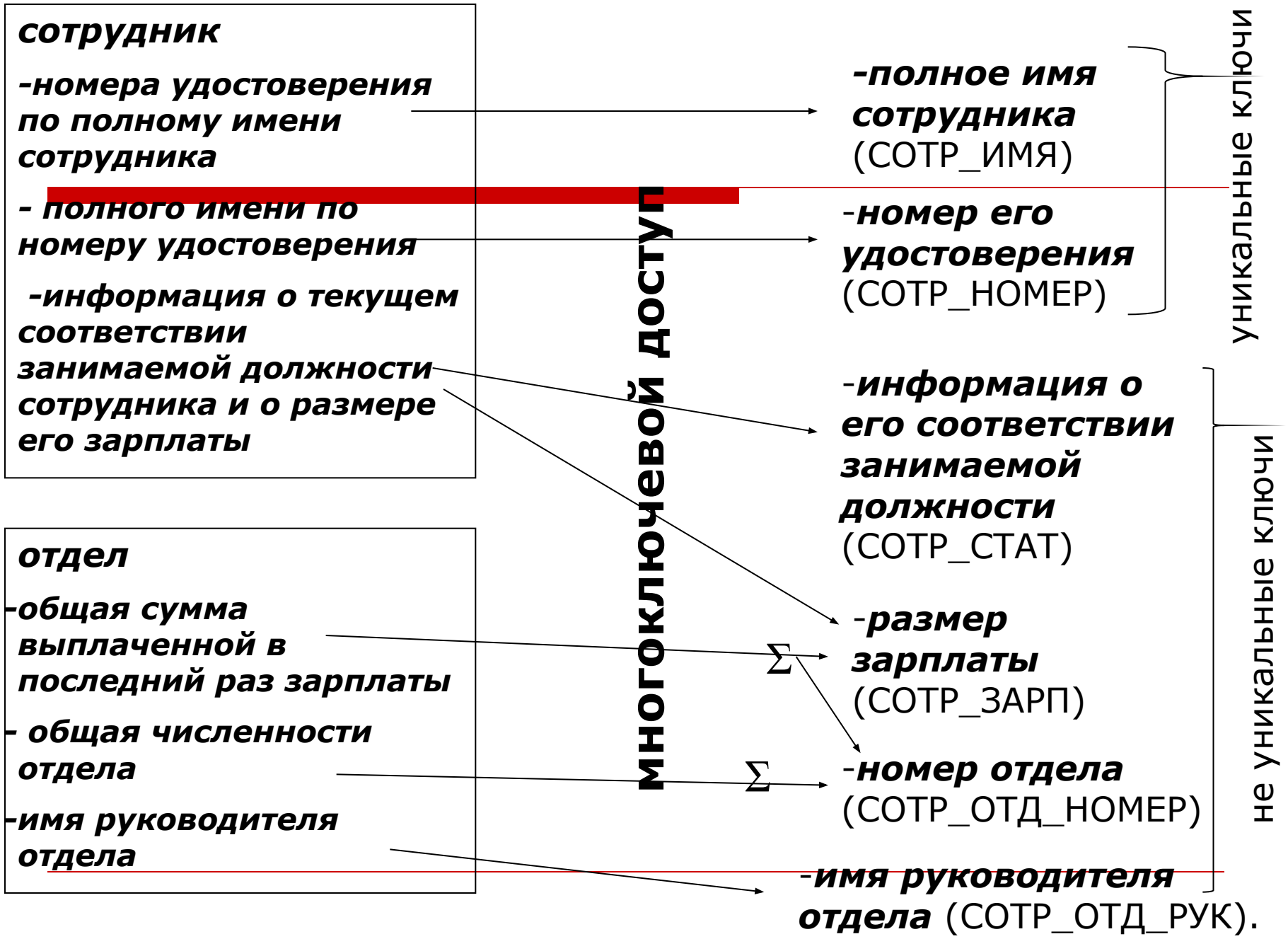
-***номер его удостоверения*** (СОТР_НОМЕР)

-***информацию о его соответствии занимаемой должности*** (для простоты, "да" или "нет") (СОТР_СТАТ)

-***размер зарплаты*** (СОТР_ЗАРП)

-***номер отдела*** (СОТР_ОТД_НОМЕР)

-***имя руководителя отдела*** (СОТР_ОТД_РУК).


сотрудник
-номера удостоверения по полному имени сотрудника
- полного имени по номеру удостоверения
-информация о текущем соответствии занимаемой должности сотрудника и о размере его зарплаты
отдел
-общая сумма выплаченной в последний раз зарплаты
- общая численности отдела
-имя руководителя отдела
многоключевой доступ
-полное имя сотрудника (СОТР_ИМЯ)
-номер его удостоверения (СОТР_НОМЕР)
-информация о его соответствии занимаемой должности (СОТР_СТАТ)
-размер зарплаты (СОТР_ЗАРП)
-номер отдела (СОТР_ОТД_НОМЕР)
-имя руководителя отдела (СОТР_ОТД_РУК).
уникальные ключи
не уникальные ключи

Недостатки:

-требуется создание достаточно сложной надстройки для многоключевого доступа к файлам

-существенная избыточность хранения (для каждого сотрудника одного отдела повторяется имя руководителя, номер отдела)

-требуется проведение массовой выборки и вычислений для получения суммарной информации об отделах

-если в ходе эксплуатации системы нам захочется, выдавать списки сотрудников, получающих заданную зарплату, то придется либо полностью просматривать файл, либо реструктуризовывать его, объявляя ключевым поле СОТР_ЗАРП.

Решение - поддерживать ***два многоключевых файла***: ***СОТРУДНИКИ и ОТДЕЛЫ***. Первый файл должен содержать поля СОТР_ИМЯ, СОТР_НОМЕР, СОТР_СТАТ, СОТР_ЗАРП и СОТР_ОТД_НОМЕР, а второй - ОТД_НОМЕР, ОТД_РУК, ОТД_СОТР_ЗАРП (общий размер зарплаты) и ОТД_РАЗМЕР (общее число сотрудников в отделе). Большинство неудобств, перечисленных в предыдущем абзаце, будут преодолены. Каждый из файлов будет содержать только ***недублируемую информацию***, необходимости в динамических вычислениях суммарной информации не возникает.

Система должна теперь знать, что она работает с ***двумя информационно связанными файлами***, должна знать структуру и смысл каждого поля (например, что СОТР_ОТД_НОМЕР в файле СОТРУДНИКИ и ОТД_НОМЕР в файле ОТДЕЛЫ означают одно и то же), а также понимать, что в ряде случаев ***изменение информации в одном файле должно автоматически вызывать модификацию во втором файле,*** чтобы их общее содержимое было согласованным. Например, если на работу принимается ***новый сотрудник***, то необходимо добавить запись в файл СОТРУДНИКИ, а также соответствующим образом ***изменить поля*** ОТД_ЗАРП и ОТД_РАЗМЕР в записи файла ОТДЕЛЫ, описывающей отдел этого сотрудника.

Понятие ***согласованности данных является ключевым понятием баз данных.*** Фактически, если информационная система (даже такая простая, как в нашем примере) поддерживает согласованное хранение информации в нескольких файлах, можно говорить о том, что она поддерживает базу данных. Если же некоторая вспомогательная система управления данными позволяет работать с несколькими файлами, обеспечивая их согласованность, можно назвать ее системой управления базами данных. Уже ***только требование поддержания согласованности данных в нескольких файлах не позволяет обойтись библиотекой функций***: такая система должна иметь некоторые собственные данные ***(метаданные)*** и даже знания, определяющие ***целостность данных.***

В нашем примере неудобно реализовывать такие запросы как "выдать общую численность отдела, в котором работает Петр Иванович Сидоров". Было бы гораздо проще, если бы СУБД позволяла сформулировать такой запрос на близком пользователям языке. Такие языки называются ***языками запросов к базам данных***. Например, на языке SQL наш запрос можно было бы выразить в форме:

```
SELECT ОТД_РАЗМЕР
FROM СОТРУДНИКИ, ОТДЕЛЫ
WHERE СОТР_ИМЯ = "ПЕТР ИВАНОВИЧ СИДОРОВ"
AND СОТР_ОТД_НОМЕР = ОТД_НОМЕР
```

Таким образом, при ***формулировании запроса СУБД позволит не задумываться о том, как будет выполняться этот запрос***. Среди ее метаданных будет содержаться информация о том, что поле СОТР_ИМЯ является ключевым для файла СОТРУДНИКИ, а ОТД_НОМЕР - для файла ОТДЕЛЫ, и система сама воспользуется этим. Если же возникнет потребность в получении списка сотрудников, не соответствующих занимаемой должности, то достаточно предъявить системе запрос

```
SELECT СОТР_ИМЯ, СОТР_НОМЕР
FROM СОТРУДНИКИ
WHERE СОТР_СТАТ = "НЕТ",
```

и система сама выполнит необходимый полный просмотр файла СОТРУДНИКИ, поскольку поле СОТР_СТАТ не является ключевым.

Далее, представьте себе, что в нашей первоначальной реализации информационной системы, основанной на использовании библиотек расширенных методов доступа к файлам, обрабатывается операция регистрации нового сотрудника. Следуя требованиям согласованного изменения файлов, информационная система вставила новую запись в файл СОТРУДНИКИ и собиралась модифицировать запись файла ОТДЕЛЫ, но именно в этот момент произошло аварийное выключение питания. Очевидно, что ***после перезапуска системы ее база данных будет находиться в рассогласованном состоянии***. Потребуется выяснить это (а для этого нужно явно проверить соответствие информации с файлах СОТРУДНИКИ и ОТДЕЛЫ) и привести информацию в согласованное состояние. Настоящие СУБД берут такую работу на себя. Прикладная система не обязана заботиться о корректности состояния базы данных.

Наконец, представим себе, что мы хотим обеспечить ***параллельную (например, многотерминальную) работу с базой данных сотрудников***. Если опираться только на использование файлов, то для обеспечения корректности на все время модификации любого из двух файлов доступ других пользователей к этому файлу будет блокирован (вспомните возможности файловых систем для синхронизации параллельного доступа). Таким образом, зачисление на работу Петра Ивановича Сидорова существенно затормозит получение информации о сотруднике Иване Сидоровиче Петрове, даже если они будут работать в разных отделах. Настоящие СУБД обеспечивают гораздо более тонкую синхронизацию параллельного доступа к данным.

Отличия БД от файловых систем

Таким образом, БД решают ***множество проблем***, которые затруднительно или вообще невозможно решить при использовании файловых систем:

1). Упорядоченный (структурированный) подход к хранению данных. Метаданные (данные о структуре).

2). Исключение дублирования данных и контроль целостности.

3). Возможность организации расширенных средств поиска информации. Наличие языков запросов к БД.

4). Независимость программ, данных и операций.

5). Защищенность от сбоев.

6). Возможность организации параллельного доступа множества пользователей.

Упорядоченный (структурированный) подход к хранению данных. Метаданные (данные о структуре).

БД хранит данные в структурированном (упорядоченном) и специально организованном виде, исключающем совместное хранение разнородной информации. **В БД структура данных строго фиксирована** и определяется стандартом используемой **модели данных**. Описание структуры данных (**метаданные**) хранятся **отдельно** от самих данных в так называемом, словаре (**системном каталоге**) данных. Таким образом, любая СУБД может работать с разными наборами данных, поскольку структура их хранения доступна при чтении данных. Для сравнения, в **файловой системе способ хранения данных - дело каждой программы**, осуществляющей хранение и обработку данных. Структура данных встроена в программу доступа и не может быть прочитана другими программами.

Исключение дублирования данных и контроль целостности.

К примеру, в текстовых файлах на порядок размещения данных не накладывается сколько-нибудь серьезных ограничений, и данные могут быть расположены произвольно. Некоторые данные могут неоднократно повторяться. В электронных таблицах данные по строкам и столбцам располагаются уже упорядочение, но все еще достаточно произвольно. В БД структура данных строго фиксирована.

Возможность организации расширенных средств поиска информации. Наличие языков запросов к БД.

Наличие интерфейса, не зависящего ни от структуры, ни от содержания БД. Существование мощных языков для реализации самых сложных запросов (SQL).

Независимость программ, данных и операций.

Использование метаданных для описания структуры данных, а также универсальных языков позволяет обеспечить:
1). Независимость программ и данных (program-data independent).
2) Независимость программ и операций (program-operation independent). Свойство наличия у операции интерфейса (имя и список аргументов) и тела (имплементация)- может быть изменена без поправок в интерфейсе.

Защищенность от сбоев.

Ведение журнала изменений и режим транзакций (начало и конец транзакции – целостная БД). Возможность производить возврат к БД до сбоя внутри одной транзакции.

Возможность организации параллельного доступа множества пользователей.

Также обеспечивается режимом транзакций. Подробнее будет рассмотрена далее.

Уровни представления БД

Так как любая упорядоченность накладывает серьезные ограничения на способ хранения и использования данных, то были предприняты действия, направленные на повышение гибкости доступа к данным. Результатом этих действий стала предложенная в 1978 г. ***трехуровневая архитектура построения баз данных***. Данная схема была разработана как стандарт представления данных (ANSI/SPARC) и в настоящий момент ее поддерживает большинство коммерческих СУБД.

Цель трехуровневой архитектуры заключается в ***отделении пользовательского представления БД от ее физического представления***, т. е, обеспечении независимости от данных.

Первый уровень - внутренний (internal). Определяется физической моделью данных, которая описывает размещение данных на физических носителях и способы доступа к ним, структуру файлов, индексов и отдельных информационных единиц.
Второй уровень - концептуальный (conceptual). Определяется концептуальной моделью данных, которая описывает логическую структуру данных без указания деталей их физического хранения.
Третий уровень внешний, или уровень представлений (интерфейса). Выводит необходимые данные в требуемом формате, скрывая остальную часть БД. Внешнее представление - это содержимое БД, каким его видит определенный пользователь. Пользователь может также изменять свое представление, не оказывая влияния на другие представления. Внешний уровень предоставляет также свободу выбора языка общения с БД. Рядовой пользователь может применять язык интерфейса, т. е. меню и другие запрограммированные действия. Опытный пользователь может воспользоваться языком запросов SQL. Все эти уровни связаны между собой программами отображения одного уровня в другой путем трансляции запросов. Запрос от конечного пользователя на требуемое данные должен быть интерпретирован на концептуальном уровне и затем преобразован в конечный запрос на извлечение требуемых данных на физическом уровне. Затем эти данные должны быть преобразованы к виду, запрашиваемому пользователем.

Свойства БД

Независимость данных

Независимость данных должна обеспечиваться на ***логическом и физическом*** уровнях. ***Логическая независимость*** предполагает независимость приложении (внешнего представления) от изменения логической структуры данных. Независимость данных на логическом уровне обеспечивается тем, что внешнее представление данных развязано от способа организации данных. Поэтому изменения на первом и втором уровнях не повлияют на конечное представление данных, т. е. однажды разработанный интерфейс не придется переписывать заново.

Физическая независимость данных предполагает независимость данных от их конкретного размещения на физических носителях, их типа, организации и способа доступа. Независимость данных на этом уровне обеспечивается развязкой первого и второго уровней. При изменении физической модели не потребуется производить изменения на концептуальном и внешнем уровнях. В настоящее время этот уровень независимости обеспечивается СУБД и операционной системой.

Контролирование избыточности данных.

Избыточность данных в первую очередь связана с хранением повторяющейся информации. Наличие повторяющейся информации приводит не только к увеличению размера БД, но и к возможности появления несогласованного состояния данных. Дублирование информации приводит к следующим серьезным проблемам:

а) приходится тратить лишние усилия на ввод повторяющихся данных;

б) при вводе повторяющихся данных возрастает вероятность ошибки набора;

в) при изменении сведений об объекте необходимо корректировать все записи, содержащие эти сведения;

г) согласованность данных может быть нарушена либо при наличии копий одинаковых данных, либо при обновлении не всех повторяющихся данных (аномалия модификации), либо непреднамеренном обновлении информации, которая напрямую не связана с удаляемыми данными (аномалия удаления); при добавлении неполных данных (аномалия добавления);

д) неоправданное увеличение размера данных приведет к снижению скорости выполнения запросов, поиска.

Основное требование контролируемой избыточности данных заключается в том, что каждая логическая единица данных должна быть сохранена лишь однажды. Данные должны быть избыточными настолько, насколько это требуется для нормальной работы системы, так как некоторая избыточность требуется для повышения эффективности работы, ускорения поиска и восстановления информации после сбоев.

Обеспечение целостности и правильности данных.

Обеспечение целостности БД составляет необходимое условие ее успешного функционирования. ***Целостность*** есть свойство БД, означающее, что в ней содержится полная, непротиворечивая, согласованная и адекватно отражающая предметную область информация.
Поддержание целостности включает ее проверку и восстановление в случае обнаружения противоречий в БД. Целостное состояние БД описывается с помощью ограничителей целостности в виде условий, которым должны удовлетворять хранимые в базе данные.
Ограничители целостности (constraints) бывают трех типов:

• ограничители значений. К ним относятся задание типа и формата, позволяющего ввод только определенных данных; задание диапазона значений; задание списка значений;
• ссылочная целостность. Обеспечивается контролем отношений между связанными данными и введением каскадного удаления и обновления связанных записей;
• целостность записи. Обеспечивается проверкой на уникальность некоторых данных и объявлением обязательных данных.

Другим важным механизмом поддержания целостности является введение транзакций. ***Транзакцией называется некоторая неделимая последовательность операций над данными БД, которая отслеживается от начала и до завершения.*** Если по каким-либо причинам (сбои или ошибки) транзакция остается незавершенной, то производится отмена всех операций, входящих в ее состав. Транзакции присущи следующие свойства:

- атомарность (выполняются вес входящие в транзакцию операции или ни одна);
- согласованность (любая транзакция должна переводить БД из одного согласованного состояния и другое согласованное состояние);
- изолированность (транзакции выполняются независимо друг от друга);
- безопасность (даже аварийное завершение работы не приводит к потере данных).

Контроль транзакций особенно важен в **многопользовательских БД**, где транзакции могут быть запущены параллельно.

Так как вследствие ограниченности своих ресурсов (один центральный процессор) компьютер не может обрабатывать параллельно выполняемые процессы, то обычно прибегают к разбиению выполняемых процессов на сравнительно небольшие части и к их поочередному выполнению. Если две или более транзакции читают или модифицируют разные данные, то это не приводит к возникновению каких-либо проблем. Другое дело, когда доступ осуществляется к одним и тем же данным. Тогда порядок выполнения частей транзакций может играть важную роль, так как от него будет зависеть конечное состояние данных.

В последнем случае говорят о **сериализации** транзакций, т. е. о составлении такого плана их выполнения (сериального плана), при котором суммарный эффект реализации транзакции эквивалентен эффекту их последовательного выполнения. При параллельном выполнении транзакций возможно возникновение конфликтов (блокировок). При обнаружении таких случаев обычно производится откат одной иди нескольких транзакций.

Обеспечение безопасности и секретности.

Практически всегда БД представляет собой важный ресурс, который должен быть надежно защищен. Потенциальными опасностями являются:

- похищение и фалъсификация данных;
- утрата конфиденциальности;
- утрата целостности;
- потеря доступности;
- непредумышленное и умышленное повреждение данных.

В отношении опасностей могут быть предприняты самые разные контрмеры, начиная от **компьютерных систем наблюдения, защиты и восстановления и заканчивая правовыми и административными процедурами**.

Обеспечение безопасности достигается защитой объектов БД и программного кода от модификации, запрещением редактирования по умолчанию, поддержкой блокировок, уровней изолированности транзакции и средств восстановления информации после сбоев (создание контрольных точек, ведение журнала, протоколирование и создание архивных копий).

Обеспечение секретности достигается шифрованием программ и данных. защитой паролем, авторизацией и аутентификацией пользователей, поддержкой различных уровней доступа к БД и отдельным ее элементам (таблицам, формам, отчетам и т, д.).

Таким образом, можно выделить **основные характеристики**, определяющие **БД как технологию хранения данных и доступа к ним**.

1. Структурированное хранение данных при их минимальной избыточности.

2. Использование метаданных для хранения описания БД.

3. Независимость программ и данных и независимость программ и операций.

4. Наличие встроенных средств обеспечения целостности, безопасности и секретности данных.

5. Поддержка разнообразных способов отображения (представления) и выборки данных.

6. Наличие расширенных средств поиска информации.