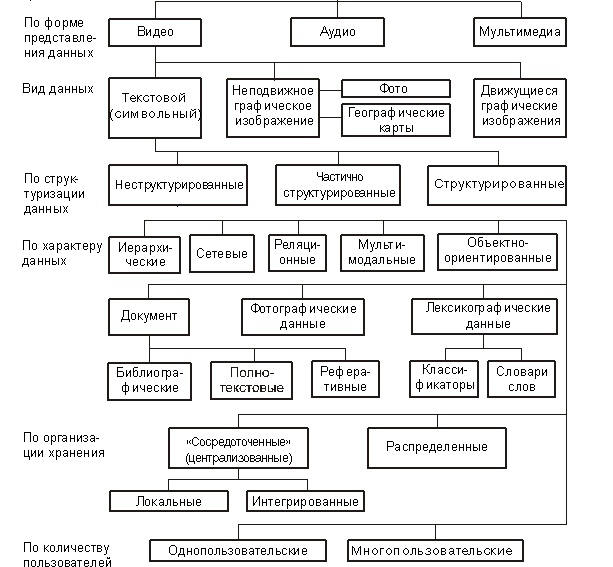
|  |
| --- |
| **Лекция 3. Ранние подходы к организации СУБД.** |
| **Лекция 4. Общие понятия реляционного подхода к организации баз данных. Основные концепции и термины** |

**Цель лекции:** Ознакомиться с комплексом основных понятий классификации *БД* и *СУБД*. Ознакомиться с функциями и функциональными возможностями *СУБД*.

**Классификация** - *разделение множества* на подмножества *по* неформально предложенному признаку. В силу многогранности баз данных и *СУБД* (комплекса технических и программных средств, для хранения, поиска, защиты и использования данных) имеется множество классификационных признаков. Классификация *БД* *по* основным признакам приведена на [рис. 2.1](https://www.intuit.ru/studies/courses/3439/681/lecture/14019?page=1#image.2.1).



**Рис. 2.1.**Классификация баз данных

*Базы данных* могут классифицироваться и с точки зрения экономической: *по* условиям предоставления услуг - бесплатные и платные (бесприбыльные, коммерческие); *по* форме собственности - государственные, негосударственные; *по* степени доступности - общедоступные, с ограниченным кругом пользователей.

**Классификация баз данных**

В мире существует множество *СУБД*. Несмотря на их различие, все они опираются на единый устоявшийся комплекс основных понятий.

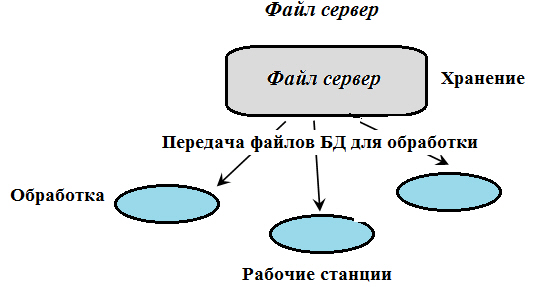
*СУБД* носит централизованный характер. Что предполагает необходимость существования некоторого лица (группы лиц), на которое возлагаются функции администрирования данными, хранимыми в базе.

*По* технологии обработки данных *БД* делятся на *централизованные БД* и *распределённые БД*.

**Централизованная БД** хранится в памяти одной вычислительной системы (применяется в локальных сетях ПК).

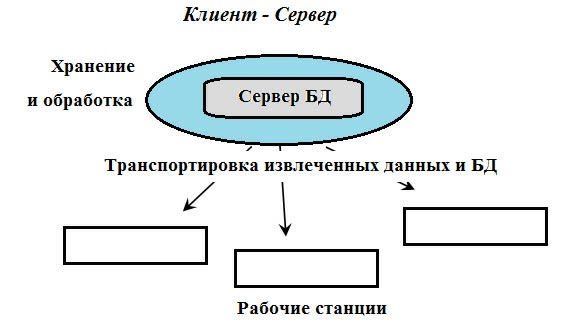
Централизованные *БД* могут быть с *сетевым доступом*.

Архитектуры систем централизованных *БД* с сетевым доступом подразделяются на *файл-сервер* и клиент-*сервер*.



**Рис. 2.2.**БД с сетевым доступом (Файл-сервер)

*Архитектура* систем *БД* с сетевым доступом (*Файл-сервер*) как показано на [рис. 2.2](https://www.intuit.ru/studies/courses/3439/681/lecture/14019?page=1#image.2.2) предполагает выделение одной из машин сети в качестве центральной (*сервер* файлов). На ней хранится совместно используемая централизованная *БД*. Все другие машины сети являются рабочими станциями. Файлы *БД* в соответствии с пользовательскими запросами передаются на рабочие станции, где и производится обработка. При большой интенсивности доступа к одним и тем же данным *производительность* системы падает.



**Рис. 2.3.**БД с сетевым доступом Клиент - сервер

В *архитектуре Клиент-сервер* ([рис. 2.3](https://www.intuit.ru/studies/courses/3439/681/lecture/14019?page=1#image.2.3)) подразумевается, что помимо хранения централизованной *БД* центральная машина (*сервер* *базы данных*) должна обеспечивать выполнение основного объёма обработки данных. *Запрос* на данные клиента, порождает *поиск* и *извлечение данных* на сервере. Извлечённые данные (но не файлы) транспортируются *по* сети от сервера к клиенту.

Пример *БД* - деловой ежедневник, в котором каждому календарному дню выделено *по* странице. Даже в отсутствии там записей, он не перестаёт быть ежедневником, т.к. имеет структуру, отличающую его от записных книжек, рабочих тетрадей и т.п. Другие примеры *БД*: *база данных* больных в поликлинике, *БД* *по* видеофильмам (видеотека), *БД* *по* сотрудникам организации (Ф.И.О., пол, дата рождения, *место*жительство, телефон, состав семьи и т.д.).

**Распределённая БД** состоит из нескольких частей, хранимых в различных ЭВМ вычислительной сети (работа с такой *БД* происходит с помощью *СУБД*).

*По* способу доступа к данным *БД* разделяются на *БД* *с локальным и удаленным доступом*.

*БД* **с локальным доступом** называется, если эта *вычислительная система* является компонентом сети ЭВМ, возможен распределённый *доступ*к такой базе. Такой способ использования *БД* часто применяют в локальных сетях ПК.

*БД* **с удалённым (сетевым) доступом** называется когда, части *БД* могут пересекаться или даже дублироваться, но хранятся в различных ЭВМ вычислительной сети.

Для работы с созданной *БД* пользователю или *администратору БД* следует иметь перечень файлов-таблиц с описанием состава их данных (структуры, схемы).

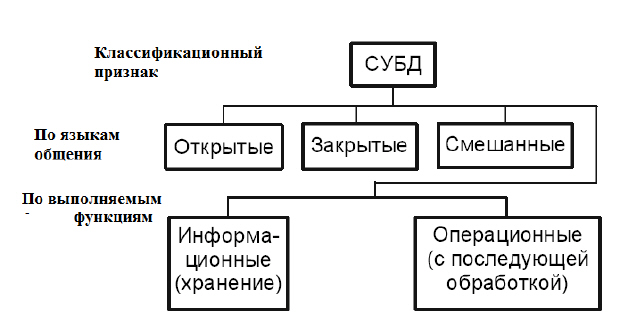
Для этого создается специальный *файл*, называемый *словарем* данных (депозитарием, словарем-справочником, энциклопедией). Описание *БД* относится к метаинформации.

В качестве технических средств могут выступать супер- или персональные компьютеры с соответствующими периферийными устройствами.

**Классификация СУБД**

**Система управления базами данных (СУБД)** - это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования *БД* многими пользователями.

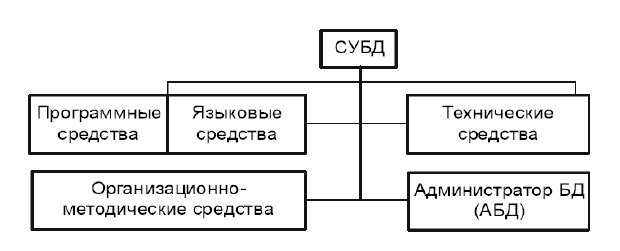
Системы управления базами данных следует классифицировать отдельно ([рис. 2.4](https://www.intuit.ru/studies/courses/3439/681/lecture/14019?page=1#image.2.4)).



**Рис. 2.4.**Классификация СУБД

**Состав СУБД и работа БД**

*СУБД* представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнения их данными получается та или иная *база данных*. В связи с этим полезно поговорить о системе программно-технических, организационных и "человеческих" составляющих ([рис. 2.5](https://www.intuit.ru/studies/courses/3439/681/lecture/14019?page=1#image.2.5)). *Программные средства* включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование *БД*, *трансляторы*.



**Рис. 2.5.**Состав СУБД

Базовыми внутренними языками программирования являются языки четвертого поколения. В качестве базовых языков могут использоваться C, C++, *Pascal*, *Object Pascal*. Язык C++ позволяет строить программы на языке *Visual Basic* с широким спектром возможностей, более близком и понятном даже пользователю-непрофессионалу, и на непроцедурном (*декларативном) языке* структурированных *запросов SQL*. Следует отметить, что исторически для системы управления базой данных сложились три языка:

1. *язык описания данных* (ЯОД), называемый также языком описания схем, - для построения структуры ("шапки") таблиц БД;
2. *язык манипулирования данными* (ЯМД) - для заполнения БД данными и операций обновления (запись, удаление, модификация);
3. *язык запросов* - язык поиска наборов величин в файле в соответствии с заданной совокупностью *критериев поиска* и выдачи затребованных данных без изменения содержимого файлов и БД (язык преобразования критериев в систему команд).

В настоящее время функции всех трех языков выполняет язык *SQL*, относящийся к классу языков, базирующихся на *исчислении кортежей*(*кортеж* чаще всего является единицей информации), языки *СУБД* FoxPro, *Visual Basic* for *Application* (*СУБД* Access) и т.д.

Вместе с тем сохранились и *языки запросов*, например *язык запросов* *по* примеру Query By *Example* (*QBE*) класса *исчисления доменов*. Отметим, что эти языки в качестве "информационной единицы" *БД* используют отдельную *запись*. С помощью языков *БД* создаются приложения, *базы данных* и *интерфейс* пользователя, включающий *экранные формы*, *меню*, отчеты. При создании *БД* на базе *СУБД* FoxPro эти элементы (объекты) фиксируются в отдельных файлах, которые, в свою *очередь*, сосредоточиваются в одном файле, называемом **проектом**. После отработки *БД* проект преобразуется в *приложение*. В *СУБД* Access все созданные объекты размещаются в одном файле.

**Основные функции СУБД**

Более точно, к числу *функций СУБД* принято относить следующие:

**1. Непосредственное управление данными во внешней памяти**

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

**2. Управление буферами оперативной памяти**

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

Заметим, что существует отдельное направление *СУБД*, которое ориентировано на постоянное присутствие в оперативной памяти всей *БД*. Это направление основывается на предположении, что в будущем объем оперативной памяти компьютеров будет настолько велик, что позволит не беспокоиться о буферизации. Пока эти работы находятся в стадии исследований.

**3. Управление транзакциями**

**Транзакция** - *это последовательность операций над БД, рассматриваемых СУБД как единое целое.*

Либо *транзакция* успешно выполняется, и *СУБД* фиксирует изменения *БД*, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии *БД*.

Понятие транзакции необходимо для поддержания логической целостности *БД*. Приведем пример информационной системы с файлами СОТРУДНИКИ и ОТДЕЛЫ, единственным способом не нарушить *целостность* *БД* при выполнении *операции* приема на работу нового сотрудника является *объединение* элементарных операций над файлами СОТРУДНИКИ и ОТДЕЛЫ в одну транзакцию. Таким образом, поддержание механизма транзакций является обязательным условием даже однопользовательских *СУБД* (если, конечно, такая система заслуживает названия *СУБД*). Но понятие транзакции гораздо более важно в *многопользовательских СУБД*.

То свойство, что каждая *транзакция* начинается при *целостном состоянии* *БД* и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы *активности пользователя* *по* отношению к *БД*. При соответствующем управлении параллельно выполняющимися транзакциями со стороны *СУБД* каждый из пользователей может в принципе ощущать себя единственным пользователем *СУБД* (на самом деле, это несколько идеализированное *представление*, поскольку в некоторых случаях пользователи *многопользовательских СУБД* могут ощутить присутствие своих коллег).

**4. Журнализация**

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

Понятно, что в любом случае для восстановления *БД* нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в *БД* требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение *журнала изменений* *БД*.

**Журнал** - *это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД.* В разных *СУБД* изменения *БД* журнализуются на разных уровнях: иногда *запись* в журнале соответствует некоторой логической *операции* изменения *БД* (например, *операции* удаления строки из таблицы реляционной *БД*), иногда - минимальной внутренней *операции* модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода.

Во всех случаях придерживаются стратегии "упреждающей" записи в журнал (так называемого протокола *Write* Ahead *Log* - WAL). Грубо говоря, эта стратегия заключается в том, что *запись* об изменении любого объекта *БД* должна попасть во *внешнюю память* журнала раньше, чем измененный *объект* попадет во *внешнюю память* основной части *БД*. Известно, что если в *СУБД* корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления *БД* после любого сбоя.

Самая простая ситуация восстановления - индивидуальный *откат* транзакции. Строго говоря, для этого не требуется общесистемный *журнал изменений* *БД*. Достаточно для каждой транзакции поддерживать локальный журнал операций модификации *БД*, выполненных в этой транзакции, и производить *откат* транзакции, путем выполнения обратных операций, следуя от конца локального журнала. В некоторых *СУБД*так и делают, но в большинстве систем локальные журналы не поддерживают, а индивидуальный *откат* транзакции выполняют *по*общесистемному журналу, для чего все записи от одной транзакции связывают обратным списком (от конца к началу).

**5. Поддержка языков БД**

Для работы с базами данных используются специальные языки, в целом называемые языками баз данных. В ранних *СУБД* поддерживалось несколько специализированных *по* своим функциям языков. Чаще всего выделялись два языка

* *язык определения схемы БД* (SDL - Schema Definition Language) и
* *язык манипулирования данными* (*DML* - *Data Manipulation Language*).

SDL служил главным образом для определения *логической структуры* *БД*, т.е. той структуры *БД*, какой она представляется пользователям. *DML*содержал набор операторов манипулирования данными, т.е. операторов, позволяющих заносить данные в *БД*, удалять, модифицировать или выбирать существующие данные.

В современных *СУБД* обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с *БД*, начиная от ее создания, и обеспечивающий базовый пользовательский *интерфейс* с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных *СУБД* является *язык запросов SQL* (Structured *Query Language*).

**Язык SQL** содержит специальные средства определения *ограничений целостности* *БД*. Опять же, *ограничения целостности* хранятся в специальных таблицах-каталогах, и обеспечение *контроля целостности* *БД* производится на языковом уровне, т.е. при компиляции операторов модификации *БД* *компилятор* *SQL* на основании имеющихся в *БД* *ограничений целостности* генерирует соответствующий программный код.

Специальные *операторы* языка *SQL* позволяют определять так называемые представления *БД*, фактически являющиеся хранимыми в *БД*запросами (результатом любого запроса к реляционной *БД* является *таблица*) с именованными столбцами. Для пользователя *представление*является такой же таблицей, как любая *базовая таблица*, хранимая в *БД*, но с помощью представлений можно ограничить или наоборот расширить видимость *БД* для конкретного пользователя. Поддержание представлений производится также на языковом уровне.

Наконец, *авторизация доступа* к объектам *БД* производится также на основе специального набора *операторов SQL*. Идея состоит в том, что для выполнения *операторов SQL* разного вида *пользователь* должен обладать различными полномочиями. *Пользователь*, создавший таблицу *БД*, обладает полным набором полномочий для работы с этой таблицей. В число этих полномочий входит полномочие на передачу всех или части полномочий другим пользователям, включая полномочие на передачу полномочий. Полномочия пользователей описываются в специальных таблицах-каталогах, *контроль* полномочий поддерживается на языковом уровне.

### Функциональные возможности СУБД

*По* степени универсальности различают два класса *СУБД*:

* *системы общего назначения* - реализованные как программный продукт, способный функционировать на ЭВМ в определённой операционной системе и поставляемый пользователям как коммерческое изделие;
* специализированные системы - создаваемые в случаях невозможности или не целесообразности использования СУБД общего назначения.

*СУБД* общего назначения - это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией *БД* информационной системы.

Рынок программного обеспечения ПК располагает большим числом разнообразных *по* своим функциональным возможностям коммерческих систем *СУБД* общего назначения.

*СУБД* - лидеры на рынке программ:

* dBASE IV, компании Borland International;
* Microsoft Access 2007;
* Microsoft FoxPro 2.6 for DOS;
* Microsoft FoxPro for Windows, Microsoft Corp:
* Paradox for DOS 4.5:
* Paradox for Windows, версия 4.5 Borland.

*Производительность* *СУБД* оценивается:

* временем выполнения запросов;
* скоростью поиска информации;
* временем выполнения операций импортирования данных из других форматов;
* скоростью выполнения таких операций как обновления, вставка, удаление данных;
* максимальным числом параллельных обращений к данным в многопользовательском режиме;
* временем генерации отчёта.

На *производительность* *СУБД* оказывают влияния 2 фактора:

* правильное проектирование
* построения БД.

*СУБД*, которые следят за соблюдением целостности данных, несут дополнительную нагрузку, которую не испытывают другие программы;

*Целостность* данных подразумевает наличие средств, позволяющих удостовериться, что *информация* в *БД* всегда остаётся корректной и полной.

*Операции*, обеспечивающие *безопасность*:

* шифрование прикладных программ;
* шифрование данных;
* защита паролем;
* ограничение уровня доступа

Хороший уровень безопасности в *СУБД* dBase IV, Access

Для сохранения информации используется двойной подход. Некоторые *операции* сохранения происходят в обход операционной системы

*Целостность* должна обеспечиваться независимо от того, каким образом данные заносятся в *память*, не конкретных действий пользователей, пробоев сети и т.п.

Он предусматривает назначение паролей для индивидуальных пользователей или групп пользователей и присвоение различных прав доступа отдельно таблицам, запросам, отчётам на уровне пользователя или группы.

### Краткие итоги

Рассмотрены вопросы классификации *БД* и *СУБД*.

*По* технологии обработки данных *БД* делятся на централизованные *БД* и распределённые *БД*. Централизованные *БД* могут быть с сетевым доступом. Архитектуры систем централизованных *БД* с сетевым доступом подразделяются на *файл-сервер* и клиент-*сервер*. Распределённая *БД*разделяется *по* способу доступа к данным *БД* с локальным и удаленным доступом.

*СУБД* - классифицируются *по* языкам общения и *по* выполняемым функциям.

Для системы управления базой данных сложились три языка: язык описания данных (ЯОД), язык манипулирования данными (ЯМД), *язык запросов*.

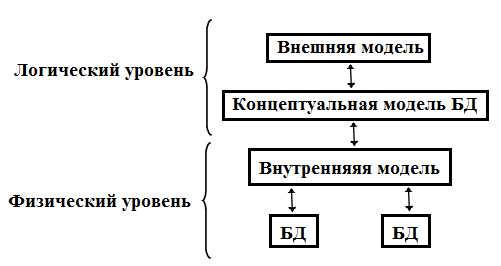
Основные *функции СУБД*: непосредственное *управление данными* во внешней памяти, управление буферами оперативной памяти, *управление транзакциями*, *журнализация*, *поддержка* языков *БД*.

*По* степени универсальности различают два класса *СУБД*: *системы общего назначения*, специализированные системы.

### 

### Архитектура СУБД

*СУБД* имеет многоуровневую структуру, в которой реализуется принцип относительной независимости *логической* и *физической* организации данных.



**Рис. 2.6.**Структура СУБД

Различают *концептуальный*, *внутренний* и *внешний* *уровни представления данных* *БД*, которым соответствуют модели аналогичного назначения.

*Концептуальная модель* состоит из *множества* экземпляров различных типов данных, имеющих структуру в соответствии с требованиями *СУБД* к *логической структуре* *БД*.

*СУБД* имеет два режима работы:

* *проектировочный* - предназначен для создания или изменения структуры базы и создания её объектов;
* *пользовательский* - использование ранее подготовленных объектов для наполнения базы или получения данных из нее.

Проектирование *БД* состоит в построении комплекса взаимосвязанных моделей данных.

**ЭТАПЫ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ**

* 1. На этапе формулирования и *анализа требований* устанавливаются цели организации, определяются требования к *БД*. Они состоят из общих требований, определенных выше, и специфических требований. Для формирования специфических требований обычно используется методика интервьюирования персонала различных уровней управления. Все требования документируются в форме, доступной конечному пользователю и проектировщику *БД*.
  2. Этап концептуального проектирования заключается в описании и синтезе информационных требований пользователей в первоначальный проект *БД*. Исходными данными могут быть совокупность документов пользователя при классическом подходе или алгоритмы приложений (алгоритмы бизнеса) при современном подходе. Результатом этого этапа является высокоуровневое *представление* (в виде системы таблиц *БД*) информационных требований пользователей на основе различных подходов.

*Сначала выбирается модель БД. Затем с помощью ЯОД создается структура БД, которая заполняется данными с помощью команд ЯМД, систем меню, экранных форм или в режиме просмотра таблиц БД. Здесь же обеспечивается защита и целостность (в том числе ссылочная) данных с помощью СУБД или путем построения триггеров.*

* 1. В процессе *логического проектирования* высокоуровневое *представление* данных преобразуется в структуру используемой *СУБД*. Основной целью этапа является устранение избыточности данных с использованием специальных правил нормализации.

*Цель нормализации - минимизировать повторения данных и возможные структурные изменения БД при процедурах обновления. Это достигается разделением (декомпозицией) одной таблицы в две или несколько с последующим использованием при запросах операции навигации*.

Полученная логическая структура БД может быть оценена количественно с помощью различных характеристик (число обращений к логическим записям, объем данных в каждом приложении, общий объем данных). На основе этих оценок логическая структура может быть усовершенствована с целью достижения большей эффективности.

Специального обсуждения заслуживает процедура управления *БД*. Она наиболее проста в однопользовательском режиме. В многопользовательском режиме и в распределенных *БД* процедура сильно усложняется. При одновременном доступе нескольких пользователей без принятия специальных мер, возможно, *нарушение целостности*. Для устранения этого явления используют систему транзакций и режим блокировки таблиц или отдельных записей.

**Транзакция** - процесс изменения файла, записи или *базы данных*, вызванный передачей одного входного сообщения.

* 1. На этапе физического проектирования решаются вопросы, связанные с производительностью системы, определяются *структуры хранения* данных и методы доступа.

Взаимодействие между этапами проектирования и словарной системой необходимо рассматривать отдельно. Процедуры проектирования могут использоваться независимо в случае отсутствия словарной системы. Сама словарная система может рассматриваться как элемент автоматизации проектирования.

Средства проектирования и оценочные критерии используются на всех стадиях разработки. В настоящее время неопределенность при выборе критериев является наиболее слабым местом в проектировании *БД*. Это связано с трудностью описания и идентификации большого числа альтернативных решений.

Проще обстоит дело при работе с количественными критериями, к которым относятся время ответа на *запрос*, *стоимость* модификации, *стоимость* памяти, время на создание, *стоимость* на реорганизацию. Затруднение может вызывать противоречие критериев друг другу.

В то же время существует много *критериев оптимальности*, являющихся неизмеримыми свойствами, трудно выразимыми в количественном представлении или в виде *целевой функции*.

К качественным критериям могут относиться гибкость, адаптивность, доступность для новых пользователей, совместимость с другими системами, возможность конвертирования в другую вычислительную среду, возможность восстановления, возможность распределения и расширения.

Процесс проектирования является длительным и трудоемким и обычно продолжается несколько месяцев. Основными ресурсами проектировщика *БД* являются его собственная интуиция и *опыт*, поэтому качество решения во многих случаях может оказаться низким.

Основными причинами низкой эффективности проектируемых *БД* могут быть:

* недостаточно глубокий анализ требований (начальные этапы проектирования), включая их семантику и взаимосвязь данных;
* большая длительность процесса структурирования, делающая этот процесс утомительным и трудно выполняемым при ручной обработке.

В этих условиях вопросы автоматизации разработки становятся первостепенными.

### Основные этапы разработки БД

#### Этап 1. Уточнение задач

На первом этапе составляется список всех основных задач, которые в принципе должны решаться этим приложением, - включая и те, которые не нужны сегодня, но могут появиться в будущем. Под "основными" задачами понимаются функции, которые должны быть представлены в формах или отчетах приложения.

#### Этап 2. Последовательность выполнения задач

Для того, чтобы приложение работало логично и удобно, лучше всего объединить основные задачи в тематические группы и затем упорядочить задачи каждой группы так, чтобы они располагались в порядке их выполнения. Может получиться так, что некоторые задачи будут связаны с разными группами или, что выполнение некоторой задачи должно предшествовать выполнению другой, принадлежащей к иной группе.

#### Этап 3. Анализ данных

После формирования списка задач, наиболее важным этапом является составление подробного перечня всех данных, необходимых для решения каждой задачи. Некоторые данные понадобятся в качестве исходных и меняться не будут. Другие данные будут проверяться и изменяться в ходе выполнения задачи. Некоторые элементы данных могут быть удалены или добавлены. И наконец, некоторые данные будут получены с помощью вычислений: их вывод будет частью задачи, но в базу данных вноситься они не будут.

#### Этап 4. Определение структуры данных

После предварительного анализа всех необходимых элементов данных нужно упорядочить их по объектам и соотнести объекты с таблицами и запросами базы данных. Для реляционных баз данных типа Access используется процесс, называемый нормализацией, в результате которого вырабатывается наиболее эффективный и гибкий способ хранения данных.

#### Этап 5. Разработка макета приложения и пользовательского интерфейса

После задания структуры таблиц приложения, в Microsoft Access легко создать его макет с помощью форм и связать их между собой, используя несложные макросы или процедуры обработки событий. Предварительный рабочий макет легко продемонстрировать заказчику и получить его одобрение еще до детальной реализации задач приложения.

#### Этап 6. Создание приложения

В случае очень простых задач созданный макет является практически законченным приложением. Однако довольно часто приходится писать процедуры, позволяющие полностью автоматизировать решение всех намеченных в проекте задач. Поэтому, понадобится создать специальные связующие формы, которые обеспечивают переход от одной задачи к другой.

#### Этап 7. Тестирование и усовершенствование

После завершения работ по отдельным компонентам приложения необходимо проверить функционирование приложения в каждом из возможных режимов. Необходимо проверить работу макросов, для этого использовав пошаговый *режим отладки*, при котором будет выполняться одна конкретная *макрокоманда*. При использовании Visual Basic для приложений в вашем распоряжении имеются разнообразные средства отладки, позволяющие проверить работу приложения, выявить и исправить ошибки.

По мере разработки автономных разделов приложения желательно передать их заказчику для проверки их функционирования и получения мнения о необходимости внесения тех или иных изменений. После того как заказчик ознакомится с работой приложения, у него практически всегда возникают дополнительные предложения по усовершенствованию, какой бы тщательной не была предварительная проработка проекта. Пользователи часто обнаруживают, что некоторые моменты, о которых в процессе постановки задач, они говорили как об очень важных и необходимых, на самом деле не играют существенной роли при практическом использовании приложения. Выявление необходимых изменений на ранних стадиях разработки приложения позволяет существенно сократить время на последующие переделки.

### Краткие итоги

Рассмотрены вопросы проектирования *БД*. Существуют два подхода при построении *БД* классический и современный.

*Архитектура* *СУБД*. *СУБД* имеет многоуровневую структуру, в которой реализуется принцип относительной независимости *логической* и *физической* организации данных.

*СУБД* имеет два режима работы проектировочный и пользовательский.

Рассматривается совокупность процедур проектирования централизованной *БД*, которые можно объединить в четыре этапа. Этап формулирования и *анализа требований*, этап концептуального проектирования, этап *логического проектирования* и этап физического проектирования.

Основные этапы разработки *БД*. Разработка *БД* состоит из следующих этапов: уточнение задач, последовательность выполнения задач, *анализ*данных, *определение* структуры данных, разработка макета приложения и пользовательского интерфейса, создание приложения и тестирования.

### Вопросы для самопроверки

* Назовите два подхода применяющиеся при построении БД и дайте их характеристику.
* Назовите уровни архитектуры СУБД.
* Какие режимы работы СУБД Вы знаете?
* Этапы проектирования БД.
* Кратко расскажите, что происходит на каждом этапе проектирования БД.
* Понятие транзакции.
* Основные причины низкой эффективности проектируемых БД.
* Чем отличается проектирование БД от разработки БД.
* Назовите этапы разработки БД.
* Кратко расскажите, что происходит на каждом этапе разработки БД.
* Дать определения понятий "предметная область", "приложение", "программа", ЯОД, ЯМД.
* Как происходит деление БД по технологии обработке данных?
* Чем отличается архитектура БД клиент - сервер от файл-сервер?
* Как классифицируются СУБД.
* Охарактеризуйте основные функции СУБД.
* Дайте понятие Журнала СУБД и его назначение.
* Как различаются по степени универсальности СУБД?
* Чем отличаются системы общего назначения от специализированных систем?

CRUD