## Рекомендуемая литература

1. **Хомоненко А.Д., Цыганков В.М., Мальцев М.Г.** Базы данных: Учебник для вузов/ Под. ред. А.Д. Хомоненко. - М.: Бином-Пресс, 2006.
2. **К. Дж. Дейт** Введение в системы баз данных. — 8-е изд. — М.: Вильямс, 2006.
3. **Т. Коннолли, К. Бегг**. *Базы данных: проектирование, реализация и сопровождение. Теория и практика*. – М. Издательский дом «Вильямс», 2001.
4. **В. Н. Четвериков и др**. *Базы и банки данных*. – М. Высшая школа, 1987.
5. **Дж. Ульман**. *Основы систем баз данных*. – М. Финансы и статистика, 1983.
6. **Лори Ульрих Фуллер, Кен Кук, Джон Кауфельд**. Microsoft Office Access 2007 для “чайников”. : Пер. с англ. – М. : ООО “И.Д.Вильямс“, 2007.

# 

## Предметная область. Информация, данные, знания

Сегодня можно утверждать, что появление баз данных стало самым важным достижением в области информационных технологий. Если еще лет тридцать назад эта сфера была лишь областью фундаментальных научных исследований, то теперь, на исследованиях баз данных основана целая индустрия информационных услуг. Этот факт послужил основой для крупномасштабных разработок коммуникационных систем и телефонии, транспорта и логистики, финансового менеджмента и законотворчества, систем баз знаний и методов доступа к научной литературе. Системы баз данных можно встретить во всех сферах человеческой деятельности: в магазине, в банке, в туристическом агентстве, в библиотеке, в здравоохранении и образовании, наконец, в управлении, т.е. в различных предметных областях.

***Предметная область*** *– это область реального мира, использующая конкретную информационную систему*.

*Под* ***информацией*** *обычно понимают абсолютно любые сведения о каком-либо событии, сущности, процессе или явлении, характерные для рассматриваемой предметной области*.

Традиционно фиксация сведений осуществляется с помощью конкретного средства общения, например, с помощью естественного языка или изображений на конкретном носителе, например бумаге. Обычно эти сведения - факты, явления, события, идеи или предметы (данные) и их интерпретация (семантика) фиксируются совместно, так как естественный язык достаточно гибок для представления того и другого. Примером может служить утверждение "Стоимость авиабилета 9200". Здесь "9200" является данным, а "Стоимость авиабилета" – семантика данного.

Перед тем, как определить понятие «данные», представим следующую абстрактную ситуацию. Имеется некоторая система и информация о состоянии системы, которая представляет особый интерес для некоторого наблюдателя, способного воспринимать состояния системы и в определенной форме фиксировать их в своей памяти. В этом случае говорят, что в памяти наблюдателя находятся «данные», описывающие состояние системы. В общем случае, в качестве таких вот наблюдателей выступают информационные системы.

Таким образом, ***данные*** *– это информация, зафиксированная в определенной форме и пригодная для последующего хранения, обработки и передачи*.

Кроме данных используется понятие знаний. Знания основаны на данных, полученных эмпирическим путем. ***Знания*** *– это закономерности предметной области (принципы, связи, законы), позволяющие ставить и решать задачи в этой области.* ***Знания*** *– это хорошо структурированные данные, или данные о данных, или метаданные*. Знания организуются в базы знаний, являющиеся фундаментом всех интеллектуальных систем, предназначенных для решения неформализованных задач, для которых невозможно представить алгоритм их решения заранее.

## Приложения и данные. Варианты построения приложений

Данные, как было сказано выше, соответствуют зарегистрированным сведениям об объектах или явлениях реального мира. Чтобы в дальнейшем использовать эти данные, нам потребуется их смысловое содержание, т.е. семантика данных. Но нередко данные отделены от своего смысла и представляют собой просто совокупность битов на запоминающем устройстве.

Например, расписание движения самолетов может быть представлено в виде таблицы (рис.2), где данные явно отделены от их смыслового содержания.

***Смысловое содержание***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер рейса | Дни недели | Пункт отправления | Время вылета | Пункт назначения | Время прибытия | Тип самолета | Стоимость билета |

***Данные***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 56 | 3, 6 | Москва | 7.20 | Киев | 9:25 | ТУ-154 | 3200.00 |
| 242 | 1 - 7 | Сочи | 14.10 | Москва | 16:15 | ИЛ-86 | 2600.50 |
| 138 | 2, 4, 7 | Баку | 21.12 | Москва | 0:52 | ТУ-154 | 3100.00 |

Компьютер главным образом имеет дело с данными как таковыми. Большая часть смыслового содержания данных вообще не фиксируется в явной форме (компьютер не "знает", является ли "14.10" временем вылета или временем прилета или стоимостью авиабилета).

Почему такое произошло? Во-первых, компьютеры никогда не обладали и до сих пор не обладают достаточными возможностями для обработки текстов на естественном языке – языке, наиболее полно выражающем семантику данных. Во-вторых, стоимость памяти компьютера первоначально была весьма велика. Она использовалась лишь для хранения самих данных, а интерпретация, то есть наделение данных смыслом, традиционно возлагалась на пользователя. Пользователь закладывал интерпретацию данных в свою программу, которая "знала", например, что четвертое вводимое значение связано с временем отправления самолета, а шестое – с временем его прибытия. Это, безусловно, существенно повышало роль программы и пользователя.

Кроме того, нередки случаи, когда пользователи создают и используют в своих программах разные наборы данных, содержащие сходную информацию. Это связано с тем, что пользователь не знает, либо не захотел узнать, что другой сотрудник уже давно ввел в программу нужные данные. Разработчики прикладных программ, написанных, например, на Бейсике, Паскале или Си, размещают нужные им данные в файлах, организуя их наиболее удобным для себя образом. При этом одни и те же данные могут иметь в разных приложениях совершенно разную организацию (разную последовательность размещения в записи, разные форматы одних и тех же полей и т.п.). Обобществить такие данные чрезвычайно трудно: например, любое изменение структуры записи файла, производимое одним из разработчиков, приводит к необходимости изменения другими разработчиками тех программ, которые используют записи этого файла.

Такая жесткая зависимость между данными и использующими их приложениями создает серьезные проблемы в ведении данных и делает их использование менее гибкими.

Существуют два возможных варианта построения приложений, использующих данные:

1. Приложение самостоятельно определяет формат своих собственных данных. При этом структура данных описывается внутри приложения, внутри исполняемого файла. При таком варианте построения приложения нередки случаи, когда пользователи создают и используют в своих приложениях разные наборы данных, содержащие сходную информацию. Типичный пример таких приложений – это файловые системы, использовавшиеся ранее для ведения картотек.
2. Приложение использует данные, формат которых определен ранее специальной системой: системой управления базами данных. В этом случае, структура данных описывается вне программы пользователя и от нее не зависит, а скорее наоборот, программа зависит от структуры данных. При таком варианте построения приложения случаи, указанные выше практически исключены, поскольку имеет место централизованное хранение данных, то есть, имеем одну базу данных, с которой взаимодействуют все приложения пользователей.

## Файловые системы

Прежде, чем начать разговор о базах данных, дадим небольшой обзор их предшественниц – файловых систем или систем, основанных на файлах.

***Файловая система*** *– это набор программ, которые выполняют для пользователей необходимые операции*. Каждая программа определяет свои данные и управляет ими. Несмотря на то, что файловые системы уже давно устарели, все же мы их очень кратко рассмотрим, сделав акцент на ограничениях, присущих файловым системам.

Файловые системы были первой попыткой компьютеризировать известные всем ручные картотеки. Подобная картотека, как правило, содержала все внутреннюю и внешнюю документацию. Поначалу, такие ручные картотеки позволяли достаточно успешно справляться со всеми поставленными задачами, если количество хранимых данных было невелико. При большом количестве хранимых объектов использование ручных картотек было целесообразным только в случае, если эти объекты нужно было только хранить и извлекать.

Но со временем, клиентам и менеджерам требовалось все больше и больше информации. Кроме того, перед ними вставали весьма непростые вопросы, связанные с обработкой хранимых данных. Дальнейшее использование ручных картотек становилось неэффективным. Так вот, в ответ на потребность в получении более эффективных способов доступа к данным, были разработаны файловые системы.

Однако в них сразу же было заложено противоречие. Вместо организации централизованного хранения данных предприятия (как это было в ручных картотеках), был использован децентрализованный подход, при котором сотрудники каждого отдела работают только со своими собственными данными и хранят их только в своем отделе.

Это противоречие впоследствии привело к обострению ряда проблем, способствовавших переходу от файловых систем к базам данных. Такими проблемами являлись:

1. **Разделение и изоляция данных**. Когда необходимые данные хранятся в разных файлах, в разных отделах, крайне сложно организовать синхронную выборку даже из двух файлов, не говоря уже о большем количестве файлов.
2. **Дублирование данных**. Из-за децентрализованной работы с данными, проводимой в каждом отделе, независимо от других отделов, в файловой системе фактически поощряется бесконтрольное дублирование данных, и это неизбежно. Это сулит не только неэффективное расходование ресурсов машины, но и может привести к нарушению их целостности, когда данные в разных отделах будут противоречить друг другу.
3. **Зависимость от данных**. Как указывалось ранее, физическая структура и способ хранения записей файлов данных жестко зафиксированы в коде программ приложений. Это означает, что изменить структуру хранимых данных достаточно сложно. Так, даже увеличение длины поля всего на один символ приведет к необходимости перекомпиляции всего приложения и созданию безошибочно работающих одноразовых конверторов, преобразующих структуру файлов без потери данных.
4. **Несовместимость форматов файлов**. Поскольку структура файлов определяется кодом приложения, она зависит от языка программирования. Так, файл, определенный программой, написанной на языке Паскаль, может существенно отличаться от файла, определенного программой, написанной на языке C. Такая прямая несовместимость файлов затрудняет процесс их совместной обработки.
5. **Фиксированные запросы**. Все требуемые запросы и отчеты должны быть созданы программистом заранее и жестко прописаны в самой программе. Это приводит к ограничению количества возможных запросов и принципиально исключает возможность создания незапланированных или произвольных запросов. Хотя, в любом отделе кадров крупного предприятия количество возможных вариантов запросов вполне может достигать несколько десятков или даже сотен, а предусмотреть их все заранее крайне сложно, практически невозможно.

## Концепция баз данных

Все перечисленные выше ограничения явились следствием двух факторов:

1. Определение данных содержится внутри приложений, а не хранится отдельно и независимо от них.
2. Помимо приложений не предусмотрено никаких других инструментов доступа к данным и их обработки.

Активная деятельность по отысканию приемлемых способов обобществления непрерывно растущего объема информации привела к созданию в начале 60-х годов специальных программных комплексов, называемых "*Системы управления базами данных*" (СУБД).

Основная особенность СУБД - это наличие процедур для ввода и хранения не только самих данных, но и описаний их структуры. Файлы, снабженные описанием хранимых в них данных и находящиеся под управлением СУБД, стали называть "*Базы данных*" (БД).

***База данных*** *– это совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояния объектов и их взаимосвязей в рассматриваемой предметной области*.

По сути, база данных – это единое, большое хранилище данных, которое однократно определяется, а затем многократно используется одновременно многими пользователями из разных подразделений. Вместо разрозненных файлов с избыточными данными, здесь все данные собраны вместе с минимальной долей избыточности. База данных – уже не собственность какого-либо одного отдела, а общий корпоративный ресурс.

Причем, база данных способна хранить не только рабочие оперативные данные организации, но и их описания. То есть, в отличие от файловых систем, здесь описание структуры находится не в приложении, а в самой базе данных. При таком подходе, добавление новых структур данных или изменение существующих никак не влияет на приложения, при условии, что они не зависят непосредственно от изменяемых компонентов.

База данных не может существовать сама по себе. Необходимо обязательно наличие некоторой специальной программы, осуществляющей управление базой данных. Такая программа называется *системой управления базой данных* (СУБД).

***СУБД*** – *это специальной программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ*.

Пусть, например, требуется хранить расписание движения самолетов (рис.2) и ряд других данных, связанных с организацией работы аэропорта (БД "Аэропорт"). Используя для этого "русифицированную" СУБД, можно подготовить следующее описание расписания:

СОЗДАТЬ ТАБЛИЦУ Расписание

(Номер\_рейса Целое

Дни\_недели Текст (8)

Пункт\_отправления Текст (24)

Время\_вылета Время

Пункт\_назначения Текст (24)

Время\_прибытия Время

Тип\_самолета Текст (8)

Стоимость\_билета Денежный(15.4);

и ввести его вместе с данными в БД "Аэропорт".

*Язык запросов* СУБД позволяет обращаться за данными, как из программ, так и с терминалов. Сформировав запрос:

ВЫБРАТЬ Номер\_рейса, Дни\_недели, Время\_вылета

ИЗ ТАБЛИЦЫ Расписание

ГДЕ Пункт\_отправления = 'Москва'

И Пункт\_назначения = 'Киев'

И Время\_вылета > 6;

получим данные из таблицы

|  |  |  |
| --- | --- | --- |
| Номер рейса | Дни недели | Время вылета |
| 56 | 3, 6 | 7.20 |

А по запросу:

ВЫБРАТЬ КОЛИЧЕСТВО(Номер\_рейса)

ИЗ ТАБЛИЦЫ Расписание

ГДЕ Пункт\_ назначения = 'Москва'

получим количество рейсов прилетающих в Москву (2).

Типичная структурная схема организации информационной системы на предприятии с использованием СУБД представлена на рис.1.

*База* *данных*

*Сервер подразделения*

*СУБД*

*Корпоративный сервер*

*Сервер подразделения*

*СУБД*

*Конечный пользователь*

*Конечный пользователь*

*Конечный пользователь*

*Терминалы*

*Конечный пользователь*

*Конечный пользователь*

*Конечный пользователь*

*Терминалы*

Рис. 1. Типичная схема организации информационной системы на предприятии

Здесь терминалы конечных пользователей подключены к серверу соответствующего подразделения, где размещается СУБД. В свою очередь, серверы всех подразделений подключены к корпоративному серверу, где размещается централизованная корпоративная база данных. В более простом случае, если серверы подразделений отсутствуют, то СУБД расположена на компьютерах пользователей.

На базе указанной выше типичной схемы организации аппаратных и программных средств, возможно построение двух архитектур распределенной информационной системы: **файл – сервер** и **клиент – сервер**. Исторически первой является архитектура «файл - сервер». На рис.2 схематически изображено функционирование информационной системы такой архитектуры.

*Возврат использованных файлов*

*Запрос*

*Передача необходимых файлов*

*Обработка*

*файлов*

*Вывод результата*

**Рис. 2. Функционирование информационной системы архитектуры «файл-сервер»**

При использовании архитектуры «файл-сервер», по запросу, посылаемому, от персонального компьютера пользователя центральному серверу, осуществляется передача всех необходимых для выполнения этого запроса файлов с сервера на компьютер пользователя, где и производится их обработка. Результат обработки отображается на экране монитора, а все файлы, которые были нужны для выполнения запроса, отправляются обратно на сервер. Недостаток архитектуры - высокая интенсивность передачи данных и, как следствие, большая загрузка сети. Причем, зачастую передаются явно избыточные данные: вне зависимости от того, сколько записей из базы данных требуется пользователю, файлы базы данных передаются целиком. Достоинство - центральный сервер может быть не очень мощным.

Куда более перспективной является архитектура «клиент – сервер». На рис.3 схематически изображено функционирование информационной системы этой архитектуры.

*Обработка*

*запроса*

*Запрос*

*Возврат результата*

*Вывод результата*

**Рис. 3. Функционирование информационной системы архитектуры «клиент-сервер»**

При использовании архитектуры «клиент-сервер» по запросу, посылаемому, от персонального компьютера пользователя центральному серверу, осуществляется выполнение этого запроса непосредственно на сервере. Результат выполнения запроса отсылается обратно на персональный компьютер пользователя, где он отображается на экране монитора. Достоинством такой архитектуры является низкая загрузка сети, так как в этом случае по ней передаются только запрос и результат его выполнения, а не перекачиваются огромные объемы данных, как в файл – серверной архитектуре. Кроме этого, отмечаем удачное сочетание централизованного хранения, обслуживания и коллективного доступа к общей корпоративной информации с индивидуальной работой пользователей над персональной информацией. Недостатком является потребность в чрезвычайно мощном центральном сервере, который бы быстро отрабатывал запросы, поступающие от нескольких пользователей сразу.

# Основные понятия

## Задачи и требования к базам и банкам данных

Приведенное выше определение банка данных как некоторой обеспечивающей подсистемы в составе большой автоматизированной системы очерчивает границу его применения. Эта граница определяется задачами поддержания информационной модели в требуемом состоянии и обеспечения информационных запросов пользователей.

Задача поддержания информационной модели в необходимом состоянии требует, чтобы в банке данных, а соответственно и в базе данных, выполнялись операции хранения и адекватной модификации информационной модели в соответствии с возникающими изменениями в состоянии объектов предметной области.

Задача обеспечения информационных запросов пользователей требует, чтобы, во-первых, банк данных обеспечивал автоматизированную систему всей необходимой информацией, а в идеальном случае и той, которая может потребоваться при дальнейшем расширении автоматизированной системы. Во-вторых, банк данных должен быть ориентирован на эффективное обслуживание запросов внешних пользователей. Это означает, что банк данных должен:

1. Удовлетворять актуальным информационным потребностям внешних пользователей;
2. Обеспечивать заданный уровень достоверности данных;
3. Обеспечивать возможность поиска данных по нескольким признакам;
4. Обеспечивать доступ к данным только пользователей с соответствующими полномочиями.
5. Обеспечивать возможность одновременного обслуживания большого числа внешних пользователей.

## Архитектура базы данных

**Архитектура** - одна из наиболее всеобъемлющих областей человеческой деятельности, занимающаяся организацией пространства и решающая любые пространственные задачи, от разработки стратегий развития до дизайна. В 1975 году подкомитет SPARC (Standards Planning and Requirements Committee) ANSI использовал это понятие для определения принципа, согласно которому рекомендуется строить СУБД. Этот термин стал использоваться в информационных технологиях.

**Архитектура базы данных** имеет трехуровневую структуру, состоящую из следующих уровней:

* внешний (пользовательский);
* концептуальный (промежуточный);
* внутренний (физический).

Схематично, архитектура базы данных представлена на рис. 4.

*Пользователи*

*Прикладные программы*

***Внешний уровень***

***Концептуальный уровень***

***Внутренний уровень***

*Физическая база данных*

База данных

Представление 1

Представление 2

Представление 3

Концептуальная модель

Внутренняя модель

Рис. 4. Трехуровневая архитектура базы данных

Внешний и концептуальный уровни являются логическими уровнями представления данных, а внутренний уровень – физическим уровнем представления данных. Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления.

Для обеспечения этого вводят некоторую **концептуальную модель данных**, которая будет отражать для пользователей информационное содержание базы данных, ее логическую структуру, но подробности организации физического хранения данных в ней будут отсутствовать. Концептуальная модель описывает следующее:

1. Все сущности, их атрибуты и связи между собой;
2. Накладываемые на данные ограничения;
3. Семантическая информация о данных;

Кроме этого, концептуальная модель должна содержать описание отображения (преобразования) данных из физической базы данных в данные для принятой концептуальной модели.

Далее, обычно при проектировании СУБД не разрабатывают специальные программы манипулирования данными на физическом уровне, а используют программы методов доступа операционной системы. Для этого СУБД необходимо описание того, как информация хранится физически. Такое физическое представление базы данных располагается на **внутреннем уровне** архитектуры. Здесь база данных может быть представлена в виде совокупностей хранимых файлов, для которых известна структура хранимых записей, определены служебные поля, реализующие связи между записями, известны методы доступа СУБД к этим записям.

Эти два уровня уже решают вопрос обеспечения независимости прикладных программ от данных. Однако наличие глобальной концептуальной модели данных вынуждает каждого внешнего пользователя при написании программы вначале ознакомиться с содержимым всей модели данных, что часто оказывается неприемлемым. Во-первых, каждый отдельный пользователь имеет отношение лишь к небольшой, вполне определенной части данных и ему нет никакой необходимости знакомиться со всем информационным содержанием. Во-вторых, необходимо обеспечить защиту данных, не имеющих отношение к конкретному пользователю, от его некомпетентных действий.

Поэтому вводится дополнительный уровень логического представления данных – **внешний уровень**, разделенный на несколько представлений: для каждого пользователя имеется свое собственное представление базы данных. Таким образом, этот уровень описывает ту часть базы данных, которая непосредственно относится к каждому конкретному пользователю. Внешнее представление содержит только те сущности, атрибуты и связи реального мира, которые интересны ему. Другие сущности, атрибуты или связи, которые ему не интересны, также могут быть представлены в базе данных, но пользователь может даже не подозревать об их существовании.

Между внешней и концептуальной моделями должно быть реализовано соответствующее отображение.

## Независимость от данных

Как было указано выше, основным назначением трехуровневой архитектуры является обеспечение независимости от данных, которая означает, что изменения на нижних уровнях никак не должны влиять на верхние уровни. При этом различают два типа независимости: логическую и физическую.

***Логическая независимость от данных*** *означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему*. Такие изменения концептуальной модели, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без необходимости внесения изменений в уже существующие внешние модели или переписывания прикладных программ (при учете, что состояния новых сущностей не должны непосредственно отображаться на внешнем уровне).

***Физическая независимость от данных*** *означает защищенность концептуальной модели от изменений, вносимых во внутреннюю модель*. Такие изменения внутренней модели, как использование различных файловых систем или структур хранения, разных устройств хранения, модификация индексов или хеширование, должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю модели.

## Назначение СУБД

Согласно определению, СУБД представляет собой специальное программное обеспечение, которое управляет доступом к базе данных.

Назначение СУБД состоит главным образом в следующем:

1. **Непосредственное управление данными во внешней памяти**. Эта функция предполагает создание необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для увеличения скорости доступа к данным (обычно для этого используются индексы).
2. **Управление буферами оперативной памяти**. Реальные СУБД обычно работают с базами данных значительного размера. В этом случае практически единственным способом увеличения скорости работы является буферизация данных в оперативной памяти. При этом даже если операционная система производит общесистемную буферизацию этого явно недостаточно для целей СУБД и она, как правило, поддерживает свой собственный набор буферов оперативной памяти с собственной дисциплиной их замены.
3. **Управление транзакциями**. ***Транзакция*** *- это последовательность операций над базой данных, рассматриваемых СУБД как единое целое*. При использовании транзакций возможны лишь два исхода: либо транзакция успешно выполняется, и СУБД фиксирует во внешней памяти изменения базы данных, произведенные этой транзакцией, либо ни одно из производимых транзакцией действий никак не отражается на состоянии базы данных. Заметим, что понятие транзакции используется не только в многопользовательских СУБД, но и в однопользовательских. Оно необходимо для поддержания логической целостности базы данных. Однако в многопользовательских системах транзакция приобретает дополнительное значение. То свойство, что каждая транзакция начинается при целостном состоянии базы данных и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к базе данных. При соответствующем управлении параллельно выполняющимися транзакциями со стороны СУБД каждый из пользователей может в принципе ощущать себя единственным пользователем СУБД.
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 базы данных 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 и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Понятно, что в любом случае для восстановления базы данных нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в базе 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 базы данных. ***Журнал*** *- это особая часть базы данных, недоступная пользователям СУБД и поддерживаемая с особой тщательностью, в которую поступают записи обо всех изменениях основной части базы данных*.
5. **Поддержка языков базы данных**. Для работы с базами данных используются специальные языки, в целом называемые **языками баз данных**. Чаще всего выделялись два языка - **язык определения схемы базы данных (ЯОД)** и **язык манипулирования данными (ЯМД)**. Первый служит главным образом для определения логической структуры базы данных, то есть той структуры базы данных, какой она представляется пользователям. Второй содержит набор операторов манипулирования данными, т.е. операторов, позволяющих заносить данные в базу, удалять, модифицировать или выбирать существующие данные.
6. **Поддержка словаря данных**. Словарь данных сам является базой данных. Он содержит «данные о данных» или «метаданные», то есть содержит определения объектов системы, их свойств и отношениях между ними для данной конкретной предметной области. Если словарь данных интегрирован в определяемую им базу данных, то должен содержать описание самого себя.

## Администратор базы данных

***Администратор базы данных*** *– это лицо, или группа лиц, отвечающее за выработку требований к базе данных, её проектирование, реализацию, эффективное использование и сопровождение, включая управление учетными записями пользователей БД и защиту от несанкционированного доступа.*

1. **Определение концептуальной модели**. После того, как администратор данных проведет логическое проектирование базы данных, то есть укажет те объекты, информацию о которых следует хранить в базе данных, администратор базы данных создает соответствующую концептуальную модель данных с помощью некоторого концептуального языка определения данных.
2. **Определение внутренней модели**. Администратор базы данных должен также решать вопросы о том, как данные должны быть представлены в хранимой базе данных. Этот процесс называют физическим проектированием базы данных. После завершения физического проектирования администратор базы данных с помощью внутреннего языка определения данных должен создать соответствующую структуру хранения. Кроме этого, он должен определить соответствующее отображение между внутренней и концептуальной моделями.
3. **Взаимодействие с пользователями**. В обязанности администратора базы данных также входит взаимодействие с пользователями, обеспечение наличия необходимых им данных и написание необходимых внешних схем с помощью прикладного внешнего языка определения данных. Кроме этого, администратор определяет отображения между каждым внешним представлением и концептуальной моделью.
4. **Определение правил безопасности и целостности данных**. Задача администратора базы данных более полно выявить все имеющиеся правила безопасности и ограничения целостности данных и задать их в базе данных. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности. Целостность БД не гарантирует достоверности содержащейся в ней информации, но обеспечивает, по крайней мере, правдоподобность этой информации, отвергая заведомо невероятные, невозможные значения.
5. **Координация всех действий по проектированию базы данных**. Администратор базы данных должен проводить координацию разработчиков на всех этапах проектирования базы данных.
6. **Определение процедур резервного копирования и восстановления**. В случае повреждения какой-либо части базы данных по той или иной причине очень важно иметь возможность восстановить данные с минимальной задержкой и с наименьшим воздействием на остальную часть системы. Данные, которые не были повреждены, совсем не должны затрагиваться процедурой восстановления. Вследствие этого, администратор базы данных должен определить и реализовать наиболее подходящую схему восстановления потерянных данных, использующую, например, периодическую выгрузку базы данных на устройство резервного копирования и процедуры, при необходимости, загружающие базу данных с последней копии.
7. **Выполнение работ по ведению словаря данных**. Администратор базы данных должен контролировать избыточность и противоречивость данных, их достоверность.
8. **Управление производительностью и реагирование на изменяющиеся потребности**. Администратор базы данных непосредственно отвечает за такую организацию системы, при которой можно получить наибольшую для всего предприятия производительность.

## Инфологический и даталогический аспекты проектирования БД

Соответственно двум понятиям – «информация» и «данные», рассмотренных нами на предыдущей лекции, различают два аспекта проектирования базы данных: инфологический и даталогический.

Инфологический аспект употребляется при рассмотрении вопросов, связанных со смысловым содержанием данных независимо от способов их представления в памяти системы. На этапе инфологического проектирования информационной системы выделяется часть реального мира, определяющая информационную потребность системы, ее предметную область. При этом решаются следующие вопросы:

1. О каких объектах или явлениях реального мира требуется накапливать и обрабатывать информацию в системе;
2. Какие их основные характеристики и взаимосвязи между собой будут учитываться;
3. Наконец, уточняются все вводимые в информационную систему понятия об объектах и явлениях, их характеристиках и взаимосвязях.

Даталогический аспект употребляется при рассмотрении вопросов представления данных в памяти информационной системы. При даталогическом проектировании системы, исходя из возможностей имеющихся средств ввода, хранения и обработки информации, разрабатываются соответствующие формы представления информации в системе, а также приводятся модели и методы представления и преобразования данных, формулируются правила их смысловой интерпретации.

## Этапы проектирования базы данных

Процесс проектирования базы данных представляет собой достаточно сложный процесс перевода отображения предметной области во внутреннюю модель базы данных. Его сложность заключается не в том, что он подразумевает собой глубокие теоретические знания, а в том, что до сих пор отсутствует сколь-нибудь единый и четкий алгоритм проектирования базы данных, который гарантировал бы получение качественного проекта. Процесс проектирования базы данных все еще остается творческим процессом и во многом определяется личностью, опытом и знаниями конкретного разработчика. Тем не менее, всегда этот процесс разбивают на два последовательных этапа: инфологическое и даталогическое проектирование, представленные на рис.5.

На этапе инфологического проектирования создается инфологическая модель, которая отображает реальный мир в некоторые понятные человеку концепции, полностью независимые от параметров среды хранения данных. Существует множество подходов к построению таких моделей: графовые модели, семантические сети, модель "сущность-связь" и т.д. Наиболее популярной из них оказалась модель "сущность-связь", которая будет рассмотрена на наших лекциях.

В процессе даталогического проектирования инфологическая модель отображается в компьютеро-ориентированную даталогическую модель, "понятную" СУБД. В процессе развития теории и практического использования баз данных, а также средств вычислительной техники создавались СУБД, поддерживающие различные даталогические модели.

Восприятие предметной области, абстрагирование, определение ее границ, изучение объектов, их свойств и связей. Накопление знаний: правил, описывающих поведение объектов.

В результате: семантическое описание предметной области

Предметная область

Изучение и описание информационных потребностей пользователей (запросы к базе данных).

Информационные потребности пользователей

Проектирование концептуальной инфологической модели и внешних инфологических моделей предметной области

Проектирование концептуальной даталогической модели и внешних даталогических моделей в соответствии с типами моделей данных, принятыми в выбранной СУБД

(логическое проектирование)

Проектирование внутренней даталогической модели

(физическое проектирование)

Выбор рациональных структур хранения данных и методов доступа к ним

Выбор СУБД

Инфологическое проектирование

Даталогическое проектирование

Рис. 5. Этапы проектирования базы данных