**1. Основные сведения о БД и СУБД.**

**1.1. Понятие базы данных.**

**Информация -** это любые сведения о каком либо событии, сущности, процессе.

**Данные -** это информация, представленная в формализованном виде, позволяющая передавать и обрабатывать её при помощи некоторого процесса и соответствующих технических средств.

Под передачей данных подразумевается и процессор хранения, т.к. с теор. хранение = передача данных, не не в пространстве, а во времени.

**База данных -** это именованная совокупность данных отображающих состояние объектов и их отношений в рассматриваемой предметной области.

Особенности организации данных в БД обеспечивают использование одних и тех же данных в различных приложениях, позволяют решать различные задачи планирования, исследования и управления. БД свдят к минимуму дублирование данных, прибегая к дублированию только для ускорения доступа к данным или для обеспечения восстановления БД при её разрушении.

Одна из важных черт БД, независимость данных от особенностей прикладных программ, которые их используют, а также возможность создания этих программ в такой форме, чо изменение особенностей хранения логической структуры или значения данных не требует изменения программ их обработки. Другой важной чертой БД является возможность изменения физических особенностей данных, без изменения их логической структуры.

Соответственно двум понятиям информация и данные в БДД различают 2 аспект рассмотрения вопросов:

1. инфологический аспект - употребление при рассмотрении вопросов, связанных со смысловым содержанием данных независимо от способа их представления в памяти системы;
2. даталогический - при рассмотрении вопросов представления данных в памяти информации системы.

**1.2. Понятие СУБД.**

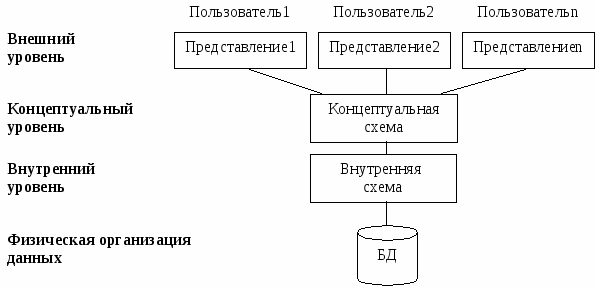
СУБД - это совокупность языковых и программных средств, предназначенных для создания, ведения и сипользования БД многими пользователями. Основной интерес к применению СУБД заключается в том. чтобы предложить пользователям или прикладным процессам абстрактное представление данных, скрыв особенности хранения и управления ими.

Обобщенная структура связи программ и данны используемых СУБД представлена на рисунке.

При выполнении основных функций СУБД должна использовать различные описания данных, в которых должны быть учтены:

1. сущности, интересующей предметной области;
2. атрибуты, характеризующие свойства каждой сущности;
3. связи, ассоциирующие выделенную сущность.

В архитектуре современных СУБД выделяют три уровня абстракции: внешний, концептуальный и внутренний.



Внешний уровень - представление БД с точки зрения конкретных пользователей. Моет включать несколько различных представлений БД со стороны различных групп пользователей. Каждый пользователь имеет дело с представлением предметной области, выраженной наиболее понятно. такое представление содержит только те сущности, атрибуты и связи, которые интересны ему при решении профессиональных задач. Различные представления на внешнем уровне могут пересекаться, т.е. использовать общие писания абстракций предметной области. На внешнем уровне создастся инфологическая модель БД, полностью независимая от платформ. Инфологическая модель является человекоориентированной, средой её хранения может быть память человека, а не ЭВМ.

Концептуальный уровень - обобщающее представление БД, описывающее то, какие данные хранятся в БД, а также связи существующие между ними. На концептуальном уровне необходимо выделить:

1. сущности их атрибуты и связи;
2. ограничения, накладываемые на данные;
3. семантическую информацию о данных;
4. информацию о мерах обеспечения безопасности.

На концептуальном уровне создается даталогическая модель, представляющая собой описание инфологической модели на языке определения данных конкретной СУБД. Эта модель является компьютероориентированной, но не рассматривает вопросы хранения данных.

Внутренний уровень - это физическое представление БД, описывающее методы хранения данных в вычислительной системе. данный уровень описывает физическую реализацию БД и предназначен для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. На внутреннем уровне создается физическая модель БД, которая также является компьютероориентированной.

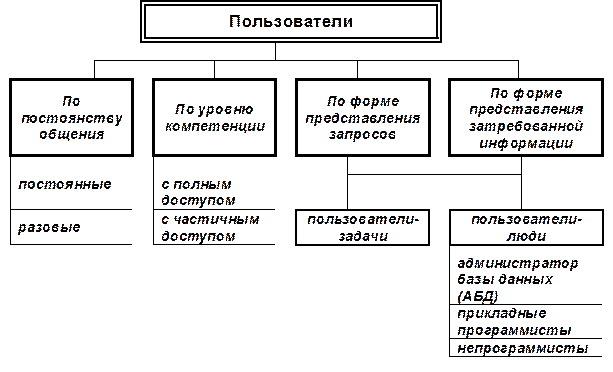
**1.3. Достоинства СУБД.**

1. контроль за избыточностью данных;
2. непротиворечивость данных;
3. большее количество полезной информации при том же объеме хранимых данных;
4. совместное использование данных;
5. поддержка целостности данных;
6. повышенная безопасность;
7. применение стандартов;
8. повышение эффективности с ростом масштабов системы;
9. возможность нахождения компромисса при противоречивых требованиях;
10. повышение доступности данных;
11. улучшение показателей производительности;
12. упрощение сопровождения системы за счет независимости данных;
13. улучшенное управление параллельностью;
14. развитые службы резервного копирования и восстановления.

**1.4. Недостатки СУБД.**

1. сложность;
2. стоимость;
3. дополнительные затраты на аппаратное обеспечение;
4. затраты на преобразования;
5. серьезные последствия при выходе системы из строя.

**1.5. Категории пользователей БД.**



**Постоянные пользователи -** такие пользователи, которые регулярно пользуются услугами БД и им можно спрогнозироать типы запросов, определяющих круг их интересов.

**Разовые пользователи -** не имеют постоянных запросов, но могут обращаться к системе с произвольными по содержанию запросами.

При разделении пользователей по уровню компетенции речь идет о защите определенной части данных от тех пользователей, которые не должны иметь возможность их получения или изменения.

**Пользователи - задачи**  обращаются к БД с регламентированными по форме содержанию запросами.

**Пользователи - люди** обращаются с произвольными или регламентированными запросами.

**Пользователи непрограммисты -** это наиболее многочисленная группа для удовлетворения информационных потребностей которые создаются БД.

**Администратор -** это человек или группа лиц, имеющие полное представление об одной или нескольких БД и контролирующие их проектирование и использование. Отвечает за состояние БД на протяжении его жизненного цикла.

**2. Проектирование БД.**

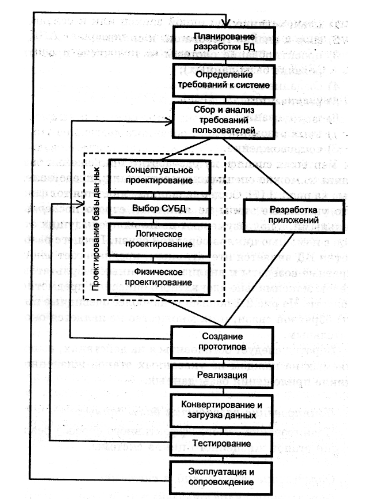
**2.1. Жизненный цикл информационной системы.**

Жизненный циклю любой сложной системы и безусловно информационной системы основанный на БД обычно состоит из нескольких этапов:

1. Планирование;
2. Сбор и анализ требований;
3. Проектирование системы;
4. Создание проотипа;
5. Реализация;
6. Тестирование;
7. Преобразование;
8. Сопровождение.

Указанные этапы не являются строго последовательными, а подразумевают повторы предыдущих этапов с помощью циклов обратной связи.

Процесс разработки БД является интерактивным, предполагает многократные возвраты и анализ результатов с целью максимально адекватного описания предметной области.



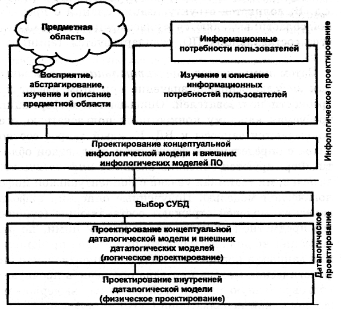
**2.2. Подходы и этапы проектирования БД.**

Процесс проектирования БД представляет собой процесс проектирования отображения описания предметной области в схему внутренней модели БД. Этот процесс является последовательностью более простых обычно итеративных процессов проектирования, менее сложных отображений между промежуточных моделей данных, т.е. последовательностью проектирования моделей уровней абстрагирования.

Основные уровни абстрагирования:

1. Информационный;
2. Внешний;
3. Концептуальный;
4. Внутренний.

В процессе проекирования БД, разрабатывали схемы моделей названных уровней, проверяется возможность отображения объектов одной модели в объекты другой модели. Только небольшие организации могут обобществить данные в одной полностью интегрированной БД. Чаще администратор Бд практически не в состоянии охватить и осмыслить все информационные требования сотрудников организации.



Процесс проектирования БД может быть разбит на инфологическое и даталогическое проектирование. Даталогическое проектирование делится на логическое и физическое проектирование. Различают концептуальную инфологическую и даталогическую модель. Внешнюю инфологическую и внешнюю даталогическую модель.

Задача инфологического моделирования БД получение семантических моделей, отражающих информационное содержание конкретной предметной области на этом этапе выполняется восприятие реальной действительной, абстрагирование, изучение описания из воспринимаемой реальности предметная область, определение ее границы, происходит абстрагирование от не существенных частей для данного конкретного применения БД. В результате этих действий определяются объекты, их свойства и связи, которые будут существенны для будущих пользователей системы. После этого осуществляется предметная область, накапливаются знания о ней, эти знания представляются в какой - либо языковой системе, обычно это неформализованное описание с использованием естественного языка, математических формул, диаграмм и т.д. Выполняется структуризация знаний предметной области, выделяются и классифицируются множества составляющей предметной области, стандартизуется терминология.

Основное значение при этом имеют потрености пользователей. описывается информация, требуемая каждому конкретному пользователю, т.е. описываются запросы БД. Каждый запрос соотносится с определенным фрагментом предметной области. Формируется описание внешних инфологических моделей, их взаимная увязка с концептуальной инфологической моделью. Получение описания инфологических моделей отражают сущности предметной области, связи между ними, но эти описания не должны зависеть от методов представления данных в конкретных СУБД.

Концептуальная инфологическая модель призвана обеспечить прочную, долговременную работы системы. призвана выдерживать замену одной СУБД на другую.

Задача логического этапа это организация данных, выделенных на предыдущем этапе в форму конкретной СУБД. Другими словами, требуется разработать схему концептуальной модели и схемы внешни моделей данных по предметной области, пользуясь только теми типами модели данных и их особенностями, которые поддерживаются этой СУБД.

На данном этапе проектирования обычно не прорабатываются вопросы связанные с организацией хранения и доступа к данным на внутреннем уровне.

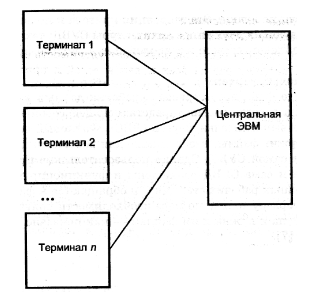
Задача физического этапа проектирования - это выбор рациональной структуры хранения данных и методов доступа к ним исходя из набора методов и средств, которые представляются разработчику СУБД. Основная цель проектирования БД - это сокращение избыточности хранения данных. Следовательно экономия используемой памяти. уменьшение затрат на многократные операции обновления избыточностей и устранения возможностей возникновения противоречий из - за хранения в разных местах сведений об одном и том же объекте.

**3. Архитектура СУБД.**

При реализации многопользовательских СУБД применяют следующие типовые архитектурные решения: телеобработка, файловый сервер, технология "клиент - сервер".

**3.1. Телеобработка.**

Традиционный архитектурой многопользовательской системы раньше считалась схема - телеобработка. При которой один компьютер с единственным процессором был соединен с несколькими терминалами, как это показано на рисунке.

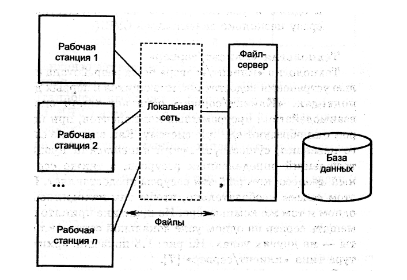


При этом вся обработка выполнялась в рамках единственного компьютера, а присоединения к нему пользовательские терминалы были типичными не интеллектуальными устройствами не способными функционировать самостоятельно.

С центральным компьютером терминалы связаны с помощью кабелей, по которым они посылали сообщение пользовательским приложениям. В свою очередь пользователи приложения обращались к службам СУБД. Таким образом сообщения возвращались назад на пользовательский терминал. Недостаток что при такой архитектуре основная нагрузка была на центральный компьютер. который должен выполнять не только действия от программ СУБД, но и значительную работу по обслуживанию терминала. Например форматирование данных выводимых на экран - терминалах.

**3.2. Файловый сервер.**

В среде файлового сервера разработка данных распределена в сети, представляющей собой локальную вычислительную сеть. ФС содержит файлы, необходимые для работы приложений и самой СУБД размещены и функционируют на отдельны рабочих станциях и обращаются к ФС только по мере необходимости получения доступа к нужным им файлам, как показано на следующем рисунке:

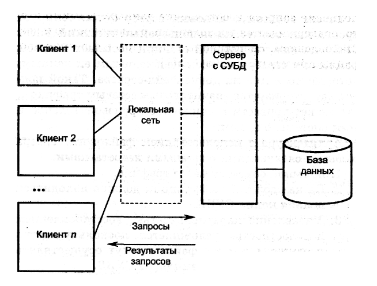


Таким образом файловый сервер функционирует на подобии совместного используемого жесткого диска. СУБД на каждой рабочей станции посылает запросы ФС по всем необходимым ей данным, которые хранятся на диске файл - сервера. Такой подход характеризуется значительным сетевым трафиком, что может привести к снижению производительности всей системы в целом. Архитектура с использованием файлового сервера обладает недостатками:

1. большой объем сетевого трафика;
2. на каждой рабочей станции должна быть полная копия СУБД;
3. управление параллельностью, восстановлением и целостностью усложняется, т.к. доступ к один и тем же файлам могут осуществлять сразу несколько экземпляров СУБД.

**3.3. Технология "клиент - сервер".**

Она была разработана с целью устранения недостатков имеющихся в предыдущих подходах. Клиент - сервер означает такой способ взаимодействия программных компонентов при которых они образуют единую систему. Как видно из названия, существует некий клиентский процесс, требующий определенных ресурсов, а также серверный процесс, которые эти ресурсы предоставляет. При этом не обязательно, чтобы они находились на одном и том же ПК. Архитектура показана на следующем рисунке:



Этот тип архитектуры обладает следующими преимуществами:

1. обеспечивается широкий доступ к БД;
2. повышается производительность системы;
3. снижается стоимость;
4. сокращаются коммуникационные расходы;
5. повышается уровень непротиворечивости данных.

**4. Инфологическое проектирование БД.**

Все этапы проектирования БД подразумевают создание данных об интересующей предметной области. Моделирование данных упрощает понимание смысла элементов данных, способствует более плодотворному общению пользователей и разработчиков.

**4.1. Модель "сущность - связь".**

Цель инфологического моделирования обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить создавая на БД. Поэтому инфологическая модель данных строят на доступном широкому кругу пользователей и разработчиков языке. Известны следующие средства создания инфологических моделей:

1. Семантические сети;
2. Язык инфологического моделирования;
3. ER - диаграммы.

Наибольшей популярностью из - за доступности и наглядности обладает "сущность - связь". Основными элементами этой модели являются:

1. Сущности;
2. Атрибуты;
3. Связи.

Сущность представляет собой различимое множество объектов (экземпляров сущности) реального мира с одинаковым набором свойств (атрибутов). Сущнсть идентифицируется именем и списком атрибутов. Каждый экземпляр сущности обладает уникальным набором значений атрибутов. На ER - диаграммах сущность представляется прямоугольником с именем сущности внутри.

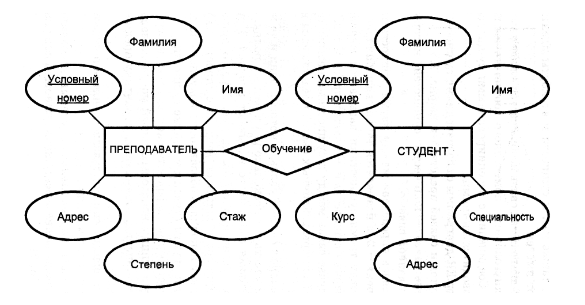
**Атрибут** - это неотъемлемое свойство сущности или связи. Именно по значениям атрибутов можно идентифицировать экземпляр сущности. Значение атрибутов представляют основную часть сведений, хранящихся в БД. На ER - диаграммах атрибут изображается овалом, соединенным с соответствующей сущностью линией и с именем атрибута внутри. С понятием атрибута тесно связано понятие домена. **Домен -** это множество значений, которые может принимать атрибут.

Атрибуты делятся на простые, составные, однозначные, многозначные и производные. *Простой атрибут* состоит из одного компонента с независимым существованием. *Составной атрибут* состоит из нескольких компонентов, каждый из которых характеризуется независимым существованием. *Однозначный атрибут*содержит одно значение для одного экземпляра сущности. *Многозначный атрибут*может содержать несколько значений для одного экземпляра сущности. *Производный атрибут*представляет значение, производное (вычисляемое) от значения связанного с ним атрибута или некоторого множества атрибутов, принадлежащих некоторой сущности.

**Ключ -** это минимальный набор атрибутов, по значению которых можно идентифицировать экземпляр сущности. В наборе атрибутов сущности можно выделить несколько потенциальных ключей.

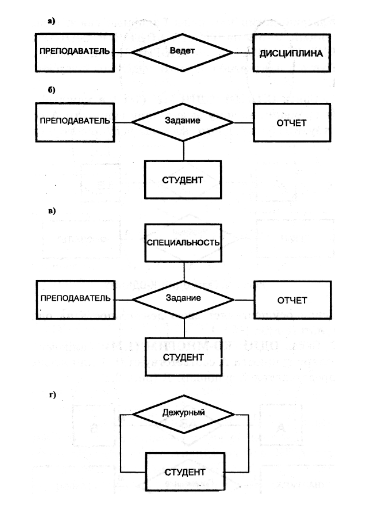
Потенциальный ключ, используемый для идентификации экземпляров сущности называется первичным ключом. На ER - диаграммах подчеркивается. Ассоциирование двух и более сущностей называется связью.

Связи также идентифицируются именем. На ER - диаграммах связь изображается в виде ромба с именем внутри. Соединение с ассоциированными сущностями производится линиями.



**Степень связи -** это количество сущностей, которые охвачены данной связью. Если связь определена между двумя сущностями, то её степень 2, а связь называется бинарной. Связь между тремя сущностями называется тернарной, между четырьмя кватернарной и т.д.

**Рекурсивная связь (унарная) -** это связь в которой одни и те же сущности участвуют несколько раз в разных ролях.

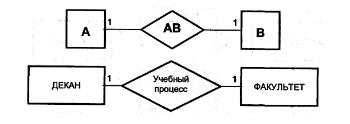


а) – бинарной; б) – тернарной; в) – кватернарной; г) – унарной (рекурсивной)

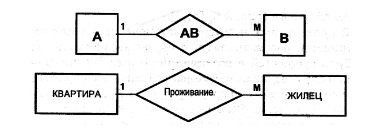
При построении инфологической модели на сущности могут накладываться ограничения, отражающие семантику (смысл) предметной области. С этими ограничениями связано понятие показателя кардинальности связи.

Показатель кардинальности указывает количественное соотношение экземпляров сущности для каждой связи. Классическими признаны бинарные связи с показателями кардинальности 1:1, 1:многим, многие : многим.

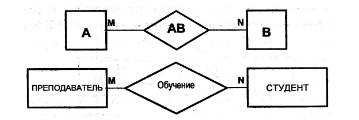
Пусть в предметной области существуют сущности A и B. Связь 1:1 в каждый момент времени каждому экземпляру сущности A соответствует не более одного экземпляра сущности B.



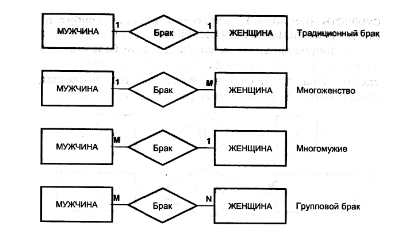
Связь один ко многим: каждому экземпляру сущности А соответствует 0,1 или несколько представителей сущности В.



Связь многие ко многим: каждому экземпляру сущности А соответствует 0, 1 или несколько представителей сущности В, а каждому экземпляру сущности В соответствует 0, 1 или несколько представителей сущности А.



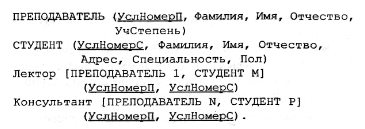
Если связь между сущностями мужчина и женщина называется брак, то существует 4 возможных представления такой связи.



Характер связи может оказаться более сложный:



В пиведенных примерах не показаны атрибуты сущностей и ассоциации во всех ER - диаграммах. Язык ER - диаграммы для построения небольших модулей и иллюстрации отдельных фрагментов больших моделей. Для представления полных инфологических моделей предметной области применяется, менее наглядный, но более содержательный, язык инфологического моделирования.



Для определенных связей между сущностями необходимо выделить в интересующей предметной области сами сущности. Но это не простая задача, так как в разных предметных областях один и тот же объект может быть сущностью, атрибутом или связью.

**4.2. Проблемы ER - моделирования.**

В процессе создания инфологической модели на языке ER-диаграмм, могут возникать нежелательные ситуации, которые в литературе называются *ловушками соединения.*Причины этих проблем кроются в неправильной интерпретации семантики предметной области, в том числе смысла некоторых связей между выделенными сущностями. Очень важно своевременно выявлять в модели данных ловушки соединения, иначе они могут привести к неадекватному описанию предметной области и необходимости перестройки всей концептуальной модели.

Наиболее распространенными являются 2 вида ловушек соединения:

1. Ловушки разветвления;
2. Ловушки разрыва.

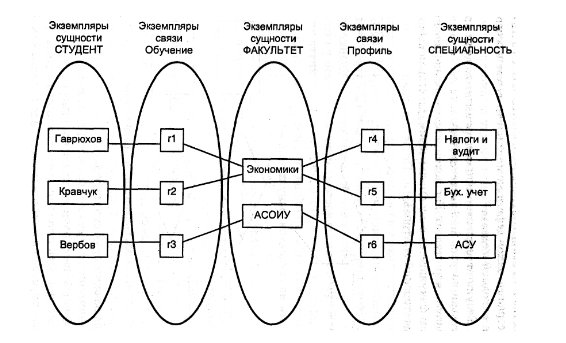
*Ловушка разветвления*имеет место в том случае, если модель отображает связь между сущностями, но путь между отдельными экземплярами этих сущностей однозначно не определяется.

*Ловушка разветвления* возникает в случае, когда две или больше связей ОДИН-КО-МНОГИМ разветвляются из одной сущности.

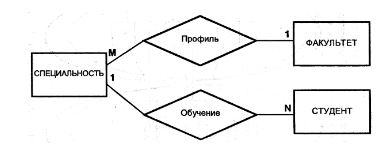


Проанализировав модель можно сделать вывод, что на одном факультете осуществляют обучение по нескольким специальностям и на факультете учатся множество студентов. Проблема может возникнуть при попытке выяснить по какой специальности обучается каждый из студентов факультета.

Для обнаружения этой проблемы удобно пользоваться семантическими сетями . С помощью семантической сетевой модели на конкретном примере невозможно дать однозначный ответ на вопрос: «По какой специальности обучается студент Гаврюхов?» - это ловушка разветвления. Эта неприятность произошла из-за неправильной трактовки связей между сущностями ФАКУЛЬТЕТ, СПЕЦИАЛЬНОСТЬ, СТУДЕНТ. Устранить такой дефект можно только путем перестройки исходной модели.



Результат адекватного преобразования модели представлен на следующем рисунке:



Если проверить полученную структуру на уровне отдельных экземпляров сущностей можно убедиться, что по преобразованной модели легко дать однозначный ответ на поставленный выше вопрос.

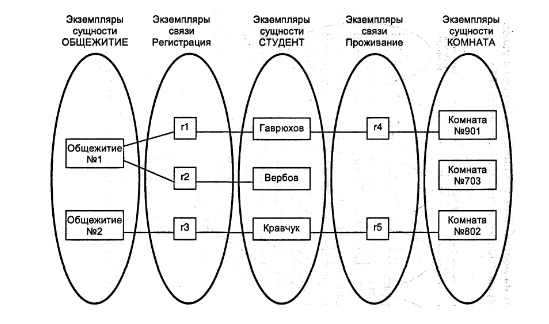
*Ловушка разрыва*появляется в том случае, если в модели предполагается наличие связи между сущностями, но не существует пути между отдельными экземплярами этих сущностей.

Ловушка разрыва возникает при наличии связи, образующей часть пути между связанными сущностями.

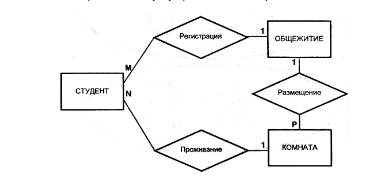


С помощью семантической сети ER-модели невозможно дать ответ на вопрос: «В каком общежитии находится комната под условным номером 703?». Это причина проявления ловушки разрыва, возникающей из-за неправильной интерпретации связей между сущностями ОБЩЕЖИТИЕ, СТУДЕНТ и КОМНАТА.

Появление ловушки разрыва возникающей неправильной интерпретации связей между сущностями общежитие, студент, комната.



Для представления правильного взаимоотношения между сущностями. Преобразованная ER - модель показана на рисунке:



Если исследовать новую структуру на уровне отдельных, то можно дать ответ на поставленный вопрос: «Комната с условным номером 703 находится в общежитии №1».

**5. Введение в язык SQL.**

**5.1. Краткая история языка SQL.**

Язык SQL, предназначенный для взаимодействия с базами данных, появился в середине 70-х гг. (первые публикации датируются 1974 г.) и был разработан в компании IBM в рамках проекта экспериментальной реляционной СУБД System R. Исходное название языка SEQUEL (Structured English Query Language) только частично отражало суть этого языка. Конечно, язык был ориентирован главным образом на удобную и понятную пользователям формулировку запросов к реляционным БД. Но, в действительности, он почти с самого начала являлся полным язы­ком БД, обеспечивающим помимо средств формулирования запросов и манипулирования БД следующие возможности:

• средства определения и манипулирования схемой БД;

• средства определения ограничений целостности и триггеров;

• средства определения представлений БД;

•  средства определения структур физического уровня, поддерживающих эффективное выполнение запросов;

• средства авторизации доступа к отношениям и их полям;

• средства определения точек сохранения транзакции и выполнения фиксации и откатов транзакций.

В языке отсутствовали средства явной синхронизации доступа к объек­там БД со стороны параллельно выполняемых транзакций: с самого начала предполагалось, что необходимую синхронизацию неявно выполняет СУБД.

В настоящее время язык SQL реализован во всех коммерческих реля­ционных СУБД и почти во всех СУБД, которые изначально основывались не на реляционном подходе. Все компании-производители провозглаша­ют соответствие своей реализации стандарту SQL, и на самом деле реали­зованные диалекты SQL очень близки. Этого удалось добиться не сразу.

Наиболее близки к System R были две системы компании IBM – SQL/DS и DB2. Разработчики обеих систем использовали опыт проекта System R, а СУБД SQL/DS напрямую основывалась на программном коде System R. Отсюда предельная близость диалектов SQL, реализованных в этих системах, к SQL System R. Из SQL System R были удалены только те части, которые были недостаточно проработаны (например, точки сохра­нения) или реализация которых вызывала слишком большие технические трудности (например, ограничения целостности и триггеры). Можно на­звать этот путь к коммерческой реализации SQL движением сверху вниз.

Другой подход применялся в таких системах, как Oracle, Informix и Sybase. Несмотря на различие в способах разработки систем, реализация SQL везде происходила «снизу вверх». В первых выпущенных на рынок версиях этих систем использовалось ограниченное подмножество SQL System R. В частности, в первой известной нам реализации SQL в СУБД Oracle в операторах выборки не допускалось использование вложенных подзапросов и отсутствовала возможность формулировки запросов с со­единениями нескольких отношений.

Тем не менее, несмотря на эти ограничения и на очень слабую, на первых порах, эффективность СУБД, ориентация компаний на поддержку разных ап­паратных платформ и заинтересованность пользователей в переходе к реляци­онным системам позволили компаниям добиться коммерческого успеха и приступить к совершенствованию своих реализаций. В текущих версиях Oracle, Informix, Sybase и Microsoft SQL Server поддерживаются достаточно мощные диалекты SQL, хотя реализация иногда вызывает сомнения.

Особенностью большинства современных коммерческих СУБД, за­трудняющей сравнение существующих диалектов SQL, является отсутст­вие единообразного описания языка. Обычно описание разбросано по раз­ным руководствам и перемешано с описанием специфических для данной системы языковых средств, не имеющих прямого отношения к SQL. Тем не менее, можно сказать, что базовый набор операторов SQL, включающий операторы определения схемы БД, выборки и манипулирования данными, авторизации доступа к данным, поддержки встраивания SQL в языки про­граммирования и операторы динамического SQL, в коммерческих реализа­циях устоялся и более или менее соответствует стандарт

Деятельность по стандартизации языка SQL началась практически од­новременно с появлением его первых коммерческих реализаций. В 1982 г. комитету по базам данных Американского национального института стан­дартов (ANSI) было поручено разработать спецификацию стандартного языка реляционных баз данных. Первый документ из числа имеющихся у автора проектов стандарта датирован октябрем 1985 г. и является уже не первым проектом стандарта ANSI. Стандарт был принят ANSI в 1986 г., а в 1987 г. одобрен Международной организацией по стандартизации (ISO). Этот стандарт принято называть SQL/86.

Понятно, что в качестве основы стандарта нельзя было использовать SQL System R. Во-первых, этот вариант языка не был должным образом тех­нически проработан. Во-вторых, его слишком сложно было бы реализовать (кто знает, как бы сложилась судьба SQL, если бы все идеи проекта System R были реализованы полностью). Поэтому за основу был взят диалект языка SQL, сложившийся в IBM к началу 1980-х гг. В сущности, этот диалект пред­ставлял собой технически проработанное подмножество SQL System R.

К 1989 г. стандарт SQL/86 был несколько расширен, и был подготов­лен и принят следующий стандарт, получивший название ANSI/ISO SQL/89. Анализ доступных документов показывает, что процесс стандар­тизации SQL происходил очень сложно с использованием не только науч­ных доводов. В результате SQL/89 во многих частях имеет чрезвычайно общий характер и допускает очень широкое толкование. В этом стандар­те полностью отсутствуют такие важные разделы, как манипулирование схемой БД и динамический SQL. Многие важные аспекты языка в соот­ветствии со стандартом определяются в реализации.

Возможно, наиболее важными достижениями стандарта SQL/89 яв­ляются четкая стандартизация синтаксиса и семантики операторов вы­борки данных и манипулирования данными и фиксация средств ограни­чения целостности БД. Были специфицированы средства определения первичного и внешних ключей отношений и так называемых провероч­ных ограничений целостности, которые представляют собой подмноже­ство немедленно проверяемых ограничений целостности SQL System R. Средства определения внешних ключей позволяют легко формулировать требования так называемой ссылочной целостности БД. Это распростра­ненное в реляционных БД требование можно было сформулировать и на основе общего механизма ограничений целостности SQL System R, но фор­мулировка на основе понятия внешнего ключа более проста и понятна.

Осознавая неполноту стандарта SQL, на фоне завершения разработ­ки этого стандарта специалисты различных компаний начали работу над стандартом SQL2. Эта работа также длилась несколько лет, было выпу­шено множество проектов стандарта, пока наконец в марте 1992 г. не был принят окончательный проект стандарта (SQL/92). Этот стандарт суще­ственно полнее стандарта SQL/89 и охватывает практически все аспекты, необходимые для реализации приложений: манипулирование схемой БД, управление транзакциями (появились точки сохранения) и сессиями (сес­сия – это последовательность транзакций, в пределах которой сохраняются временные отношения), подключения к БД, динамический SQL. Наконец, были стандартизованы отношения-каталоги БД, что вообще-то не связано непосредственно с языком, но очень сильно влияет на реализацию.

В 1995 г. стандарт был дополнен спецификацией интерфейса уровня вызова (Call-Level Interface – SQL/CLI). SQL/CLI представляет собой на­бор спецификаций интерфейсов процедур, вызовы которых позволяют вы­полнять динамически задаваемые операторы SQL. По сути дела, SQL/CLI представляет собой альтернативу динамическому SQL. Интерфейсы про­цедур определены для всех основных языков программирования: С, Ada, Pascal, PL/1 и т. д. Следует заметить, что стандарт SQL/CLI послужил осно­вой для создания повсеместно распространенных сегодня интерфейсов ODBC (Open Database Connectivity) и JDBC (Java Database Connectivity).

В 1996 г. к стандарту SQL/92 был добавлен еще один компонент — SQL/PSM (Persistent Stored Modules). Основная цель этой спецификации состоит в том, чтобы стандартизировать способы определения и исполь­зования *хранимых процедур,*т. е. специальным образом оформленных про­грамм, включающих операторы SQL, которые сохраняются в базе дан­ных, могут вызываться приложениями и выполняются внутри СУБД.

Незадолго до завершения работ по определению стандарта SQL2 была начата разработка стандарта SQL3. Первоначально планировалось завер­шить проект в 1995 г. и включить в язык некоторые объектные возможности: определяемые пользователями типы данных, поддержку триггеров, поддержку темпоральных свойств данных и т. д. Реально работу над новым стан­дартом удалось частично завершить только в 1999 г., и по этой причине (а также в связи с проблемой 2000 года) стандарт получил название SQL: 1999.

Приведем краткую характеристику текущего состояния стандарта SQL: 1999 и перспектив его развития. Прежде всего, заметим, что каждый новый вариант стандарта языка SQL был существенно объемнее предыду­щих версий. Так, если стандарт SQL/89 занимал около 600 страниц, то объем SQL/92 составлял на 300 с лишним страниц больше. Самые первые проекты SQL3 занимали около 1500 страниц. Это вполне естественно, по­тому что язык усложняется, а его спецификации становятся более деталь­ными и точными. Но разработчики SQL3 пришли к выводу, что при таких объемах стандарта вероятность его принятия и последующей успешной поддержки заметно уменьшается. Поэтому было принято решение раз­бить стандарт на относительно независимые части, которые можно было бы разрабатывать и поддерживать по отдельности.

В 1999 г. были приняты пять первых частей стандарта SQL: 1999. Первая часть (SQL/Framework) посвящена описанию концептуальной структуры стандарта. В этой части приводится развернутая аннотация следующих четырех частей и формулируются требования к реализациям, претендующим на соответствие стандарту.

Вторая часть SQL: 1999 (SQL/Foundation) образует базис стандарта. Вводится система типов языка, формулируются правила определения функциональных зависимостей и возможных ключей, определяются син­таксис и семантика основных операторов SQL:

• операторов определения и манипулирования схемой базы данных;

• операторов манипулирования данными;

• операторов управления транзакциями;

• операторов управления подключениями к базе данных и т. д.

Третью часть занимает уточненная по сравнению с SQL/92 специфи­кация SQL/CLL В четвертой части специфицируется SQL/PSM — синтак­сис и семантика языка определения хранимых процедур. Наконец, в пя­той части - SQL/Bindings - определяются правила связывания SQL для стандартных версий языков программирования FORTRAN, COBOL, PL/1, Pascal, Ada, С и MUMPS.

В стандарт SQL: 1999 должны были войти еще несколько частей. Среди них спецификации следующих средств:

• управление распределенными транзакциями (SQL/Transaction);

• поддержка темпоральных свойств данных (SQL/Temporal);

• управление внешними данными (SQL/MED);

• связывание с объектно-ориентированными языками программирова­ния (SQL/OLB);

• поддержка оперативной аналитической обработки (SQL/OLAP).

В конце 2003 г. был принят и опубликован новый вариант междуна­родного стандарта SQL:2003. Многие специалисты считали, что в вариан­те стандарта, следующем за SQL: 1999, будут всего лишь исправлены неточности SQL: 1999- Но на самом деле, в SQL:2003 специфицирован ряд новых и важных свойств, часть из которых мы затронем в этом курсе.

Претерпела некоторые изменения общая организация стандарта. Стандарт SQL:2003 состоит из следующих частей:

• 9075-1, SQL/Framework;

• 9075-2, SQL/Foundation;

• 9075-3, SQL/CLI;

• 9075-4, SQL/PSM;

• 9075-9, SQL/MED;

• 9075-10, SQL/OLB;

• 9075-11, SQL/Schemata;

• 9075-13, SQL/JRT;

• 9075-14, SQL/XML.

Части 1-4 и 9-10 с необходимыми изменениями остались такими же, как и в SQL: 1999 (разд. 7.4). Часть 5 (SQL/Bindings) перестала существовать; соответствующие спецификации включены в часть 2. Раздел части 2 SQL: 1999, посвященный информационной схеме, выделен в отдельную часть 11. Появились две новые части — 13 и 14. Часть 13 полностью называ­ется «SQL Routines and Types Using the Java Programming Language» («Ис­пользование подпрограмм и типов SQL в языке программирования Java»). Появление такой части стандарта оправдано повышенным вниманием к языку Java со стороны ведущих производителей SQL-ориентированных СУБД. Наконец, последняя часть SQL.2003 посвящена спецификациям языковых средств, позволяющих работать с XML-документами в среде SQL.

На мой взгляд, текущее состояние процесса стандартизации языка SQL отражает текущее состояние технологии SQL-ориентированных баз данных. Ведущие поставщики соответствующих СУБД (сегодня это ком­пании IBM, Oracle и Microsoft) стараются максимально быстро реагиро­вать на потребности и конъюнктуру рынка и расширяют свои продукты все новыми и новыми возможностями. Очевидна потребность в стандар­тизации соответствующих языковых средств, но процесс стандартизации явно не поспевает за происходящими изменениями.

**5.2. Структура языка SQL.**

Язык SQL, соответствующий последним стандартам SQL:2003, SQL: 1999 (и даже SQL/92), – это очень богатый и сложный язык, все возможности которого трудно сразу осознать и тем более понять. Поэтому приходится разбивать язык на уровни, или слои, такие, что каждый уро­вень языка включает все конструкции, входящие в более низкие уровни. В стандарте определяется несколько способов разбиения языка на уров­ни. В одной из классификаций язык разбивается на *«базовый» (entry), «промежуточный» (intermediate)*и *«полный» (full)*уровни.

Эта классификация ориентирована, прежде всего, на производите­лей СУБД, в которых поддерживается SQL. Реализация базового уровня языка является обязательным условием хотя бы какого-то соответствия стандарту. Реализация промежуточного уровня желательна, и обычно именно такой уровень языка поддерживается ведущими компаниями-производителями SQL-ориентированных СУБД. Наконец, полный уро­вень языка является целью, к достижению которой следует стремиться. В данной классификации критерием отнесения той или иной возможности языка к некоторому уровню является оцениваемая создателями стандар­та SQL (большая часть которых является сотрудниками ведущих компа­ний, производящих SQL-ориентированные СУБД) техническая слож­ность реализации этой возможности. Конечно, такая классификация важна и для программистов приложений баз данных, но только для того, чтобы оценить реальные возможности конкретной СУБД. Для понима­ния языка SQL это разбиение на уровни несущественно.

Другая классификация показана на рис. 11.1. Среди всех конструкций языка SQL можно выделить такие конструкции, которые можно использо­вать при *«прямом» (direct)*взаимодействии конечного пользователя с СУБД (например, в интерактивном режиме). В некотором смысле этот уровень также является базовым, поскольку соответствующие средства языка в на­ибольшей степени отражают его ориентированность на работу с мультим­ножествами. На следующем уровне, уровне *«встраиваемого» (embedded)*SQL, язык расширяется конструкциями, позволяющими использовать воз­можности прямого SQL в программах, написанных на традиционных языках программирования.

Наконец, на уровне *«динамического» (dynamic)*SQL во встраиваемый SQL добавляются конструкции, позволяющие приложе­ниям обращаться к СУБД с конструкциями прямого SQL, которые дина­мически образуются во время выполнения программы.

Нам кажется, что вторая классификация является более полезной для читателя, постигающего основы языка SQL. По нашему мнению, до­полнительные возможности, присутствующие во встраиваемом и в дина­мическом SQL, не слишком сильно влияют на модельное представление языка. Конечно, возможности встраиваемого и динамического SQL необ­ходимо хорошо знать разработчикам приложений SQL-ориентированных баз данных. Но поскольку задачей этого курса не является обучение ис­пользованию языка SQL при программировании приложений баз дан­ных, мы не будем затрагивать эти темы. Обратимся к прямому SQL, при­чем не в полном объеме стандартов SQL:2003 и SQL: 1999 (этого не позво­ляет сделать объем курса). Обсудим только наиболее важные аспекты.

**5.3. Типы данных SQL.**

Данные, хранящиеся в столбцах таблиц SQL-ориентированной базы данных, являются типизированными, т.е. представляют собой значения од­ного из типов данных, предопределенных в языке SQL или определяемых пользователями путем применения соответствующих средств языка. Для этого при определении таблицы каждому ее столбцу назначается некоторый тип данных (или домен), и в дальнейшем СУБД должна следить, чтобы в каждом столбце каждой строки каждой таблицы присутствовали только до­пустимые значения. В этом разделе мы обсудим систему типов языка SQL.

Все допустимые в SQL типы данных, которые можно использовать при определении столбцов, разбиваются на следующие категории:

• точные числовые типы *(exact numerics);*

• приближенные числовые типы *(approximate numerics);*

• типы символьных строк *(character strings);*

• типы битовых строк *(bit strings)\*;*

• типы даты и времени *(datetimes);*

• типы временных интервалов *(intervals);*

• булевский тип *(Booleans);*

• типы коллекций *(collection types);*

• анонимные строчные типы *(anonymous row types);*

• типы, определяемые пользователем *(user-defined types);*

• ссылочные типы *(reference types).*

В столбцах таблиц, определенных на любых типах данных, наряду со зна­чениями этих типов, допускается сохранение неопределенного значения, ко­торое обозначается ключевым словом NULL. В языке определено, что результа­том выражений вида *х*а\_ор NULL, NULL a\_op *x,*NULL а\_ор NULL является NULL для всех арифметических операций а\_ор (+, - и т. д.), допустимых для типа данных выражения х (выражение NULL a\_op NULL является допустимым для лю­бой арифметической операции а\_ор). Также по определению полагается, что значением выражений х comp\_op NULL, NULL comp\_\_op x,NULL comp\_op NULL для всех операций сравнения (-, ^, >, < и т. д.), определенных для типа выра­жения х, является третье логическое значение *unknown*(выражение null comp\_op NULL является допустимым для любой операции сравнения сопр\_ор).