1СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Информационные системы и их виды

Информационная система (ИС) - это

– система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующих информационную технологию выполнения установленных функций (ГОСТ 34.003-90 Автоматизированные системы);

– система, которая организует хранение и манипулирование информацией о предметной области (ГОСТ 34.321-96 Информационные технологии);

– совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств (ГОСТ Р 50922-2006 Защита информации);

– организационно упорядоченная совокупность документов и информационных технологий, в том числе с использованием средств вычислительной техники и связи, реализующих информационные процессы (ГОСТ Р 54089-2010 Интегрированная логистическая поддержка).

Будем понимать под ИС систему для сбора, передачи, обработки, хранения и выдачи информации пользователям и состоящую из следующих основных компонентов: программное и информационное обеспечения, технические средства, обслуживающий персонал .

ИС по характеру представления и логической организации информации делятся на: фактографические, документальные, геоинформационные.

По функциям и решаемым задачам ИС делятся на справочные, поисковые, расчетные, технологические.

1.2 База данных (БД) и система управления базой данных (СУБД)

Сегодня ни одна ИС не обходится без базы данных и системы управления базой данных.

База данных (БД) - именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

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

Первый этап развития СУБД связан с организацией баз данных на больших машинах типа IBM 360/370, ЕС-ЭВМ и др. В этот период

–базы данных хранились во внешней памяти центральной ЭВМ,

–задачи запускались в пакетном режиме,

–интерактивный режим доступа обеспечивался с помощью терминалов,

–программы доступа к БД писались на различных языках и запускались как обычные числовые программы,

–управление распределением ресурсов в основном осуществляются операционной системой (ОС)..

Второй этап связан с эпохой персональных компьютеров, когда

–появились программы, которые назывались СУБД и позволяли хранить значительные объемы информации;

–все СУБД были рассчитаны на создание БД в основном с монопольным доступом.

Третий этап связан с распространением компьютерных сетей. В этот период

- обеспечение поддержки полной реляционной модели
- большинство СУБД рассчитаны на много платформенную архитектуру,
- практически все СУБД имеют средства подключения клиентских приложений, разработанных с использованием настольных СУБД, и средства экспорта данных из форматов настольных СУБД второго этапа развития

Четвертый этап характеризуется появлением новой технологии доступа к данным — интранет (технологии клиент-сервер), когда отпадает необходимость использования специализированного клиентского программного обеспечения и для работы с удаленной базой данных используется стандартный браузер

1.3 Виды СУБД

Американским комитетом по стандартизации ANSI (American National Standards Institute) предложена трехуровневая система организации СУБД:

- уровень внешних моделей (определяет отдельные приложения);
- концептуальный уровень (объединяет данные, используемые всеми приложениями и отражает обобщенную модель предметной области, для которой создавалась база данных);
- физический уровень (собственно данные, расположенные в файлах на внешних носителях информации).

Такая организация обеспечивает логическую (возможность изменения одного приложения без корректировки других) и физическую (возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений) независимость уровней СУБД

По степени распределённости различают

– локальные СУБД (все части локальной СУБД размещаются на одном компьютере)

– серверные СУБД (файлы данных располагаются централизовано на сервере).

В свою очередь серверные СУБД подразделяются на файл-серверные и клиент-серверные. Для файл-серверных характерно размещение данных на файл-сервере, а СУБД - на каждом клиентском компьютере, а доступ СУБД к данным осуществляется через локальную сеть.

К преимуществам таких СУБД следует отнести низкую нагрузку на процессор файлового сервера, а к недостаткам:

– потенциально высокую загрузка локальной сети;

– затруднённость или невозможность централизованного управления данными;

– затруднённость или невозможность обеспечения таких характеристик как высокая надёжность, высокая доступность и высокая безопасность.

Недостатки файл серверных систем становятся преимуществами клиент-серверных СУБД.

В клиент-серверных на сервере вместе с БД располагается СУБД и доступ к данным осуществляется непосредственно в монопольном режиме. Все клиентские запросы обрабатываются СУБД централизованно. В этом случае предъявляются повышенные требования к серверу, но обеспечивается

– потенциально более низкая загрузка локальной сети;

– удобство централизованного управления данными;

– такие характеристики как высокая надёжность, высокая доступность и высокая безопасность.

Примеры серверных СУБД - Oracle, IBM DB2, MS SQL Server, Sybase, My SQL

1.4 Предметная область и ее описание

Проектирование информационной системы, составным компонентом которой является база данных, начинается с анализа предметной области. Предметная область (ПО) – это область применения конкретной БД. Предметной областью может быть сфера управления предприятиями, транспорт, медицина, научные исследования и т.п.

Прежде всего, должны быть определены границы предметной области и сформулирована главная цель проектирования базы данных.

При определении границ ПО используют :

– подход «от реального мира», когда с помощью экспертов определяются границы ПО (состав объектов, их свойства и отношения с учетом существующего положения дел и развития системы);

– подход «от запросов пользователей», который широко используется для уточнения границ ПО и наибольшее применение получает в период использования ИС, когда накапливается достаточно информации о содержании запросов и необходимо выполнить коррекцию границ ПО и модели системы.

Будем говорить, что предметная область БД определена, если известны существующие в ней объекты, их свойства и отношения (связи).

Объект – это то, о чем должна накапливаться информация. Выбор объектов осуществляется в соответствии с целевым назначением. Объекты могут быть атомарными или составными, причем один и тот же объект может быть в одном приложении выступать как атомарный, в другом – как составной.

Атомарный объект — объект определенного типа, дальнейшее разложение которого на более мелкие объекты в рамках данной предметной области невозможно. Составные объекты включают в себя множества объектов

Каждый объект в конкретный момент времени характеризуется состоянием, которое описывается набором свойств и связей его с другими объектами.

Свойства объекта могут не зависеть от его связей с другими объектами, то есть являются локальными, а могут и зависеть. В последнем случае они являются реляционными.

Между объектами ПО могут существовать отношения подчиненности. Каждая связь (отношение) между объектами по числу входящих в нее объектов характеризуется степенью n=2.3, …, k (бинарная, териарная, …, k-арная).

Чтобы отобразить объекты в информационную сферу, необходимо определить:

– какие объекты важны для данного применения;

– какие свойства могут иметь эти объекты;

– какие связи существуют между объектами;

– какие имена можно присвоить отдельным составляющим объектной системы.

Таким образом, анализ предметной области позволяет определить ее границы и лежит в основе проектирования состава элементов информационной модели БД.

1.5 Инфологическое моделирование

Цель инфологического проектирования -создание структурированной информационной модели предметной области, для которой разрабатывается БД.

Определим требования к такой модели:

1) обеспечение естественных для человека способов сбора и представления информации, которую предполагается хранить в создаваемой БД;

2) адекватное отображение моделируемой предметной области (корректность схемы БД);

3) простота и удобство использования на следующих этапах проектирования, то есть поддерживается известными СУБД (сетевые, иерархические, реляционные и др.);

4) использование языка описания, понятного проектировщикам БД, программистам, администратору и будущим пользователям.

Инфологическая модель выражает информацию о предметной области в виде, независимом от используемой СУБД. Ее называют также семантической, смысловой или концептуальной.

На этапе инфологического моделирования базы данных широко используется модель «сущность-связь». Модель «сущность-связь» предложена в 1976 г Питером Пин-Шен Ченом американским профессором компьютерных наук.

Составные элементы инфологической модели

- сущности,
- их атрибуты и
- связи между ними

Сущность – некоторая абстракция реально существующего предмета, объекта, явления.

Тип сущности - набор однородных личностей, предметов, событий или идей, выступающих как целое.

Экземпляр сущности относится к конкретному объекту

Атрибут – это поименованная характеристика сущности. Атрибуты используются для определения того, какая информация должна быть собрана о сущности, то есть для описания свойства сущности. Это следует из определения. Но атрибут также используется:

- для однозначной идентификации конкретного экземпляра сущности (ключ – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности);
- для представления связей между сущностями, характеризует те объекты между которыми существует связь (отношение).

Связь – средство представления отношений между сущностями.

Графическое представление модели «сущность-связь» носит название ER(Entity Relatioship) -диаграммы.

Множества сущностей изображаются в виде прямоугольников, а множества отношений (связей) - в виде ромбов. Если сущность участвует в отношении, они связаны линией. Атрибуты изображаются в виде овалов и связываются линией с одним отношением или с одной сущностью (см.рисунок 1).

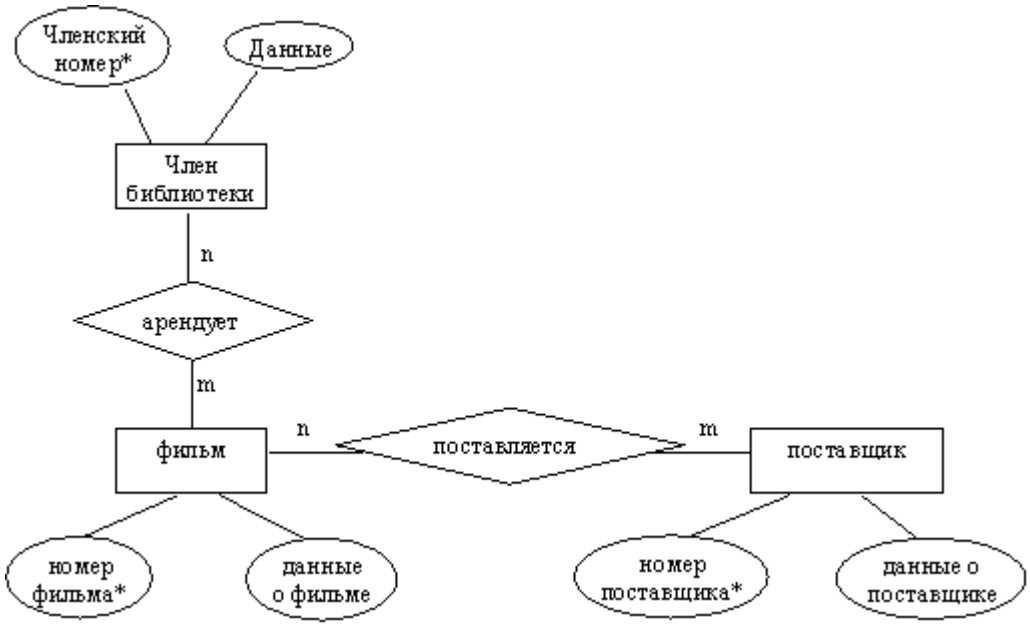


Рисунок 1

1.6 Даталогическое моделирование (иерархическая, сетевая, реляционная модели данных)

Даталогическая модель разрабатывается на основе инфологической модели предметной области с учётом конкретной реализации СУБД.

Существуют различные виды даталогических моделей - иерархическая, сетевая, реляционная.

Иерархическая модель представляет данные в виде иерархии и ориентирована на описание объектов, находящихся между собой в отношении подчинения (см.рисунок 2).

Иерархические модели представляются в форме графов – форме деревьев. Дерево – это связный граф, который не содержит циклов (корень дерева – вершина, в которую не заходит ни одно ребро). На верхнем уровне дерева в имеется один узел – “корень”, на следующем уровне располагаются узлы, связанные с этим корнем. Вершины графа – типы сущностей, а дуги – типы связей между сущностями.

Поиск данных в иерархической системе всегда начинается с корня.

Основные достоинства этой модели - простота описания иерархических структур реального мира и быстрое выполнение запросов,

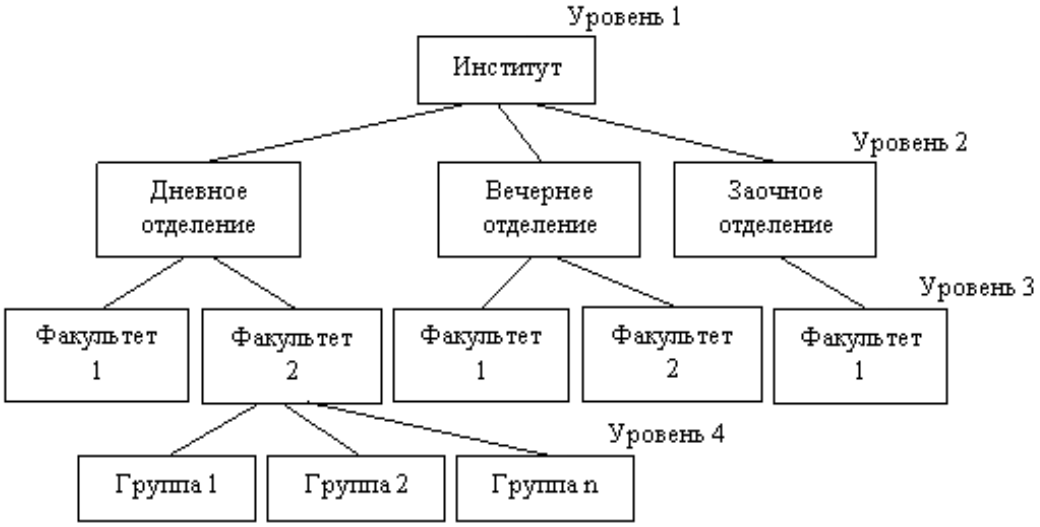


Рисунок 2 – Иерархическая модель

Сетевая модель позволяет описывать более сложные виды взаимоотношений между данными, чем иерархическая, но расширение возможностей достигается за счет большей сложности реализации самой модели и трудности манипулирования данными. Эти модели данных широко применялась в 70-е годы в первых СУБД.

Сетевая модель также использует графовую форму представления данных, но сетевой модели соответствует произвольный граф (возможно имеющий циклы и петли), см.рисунок 3.

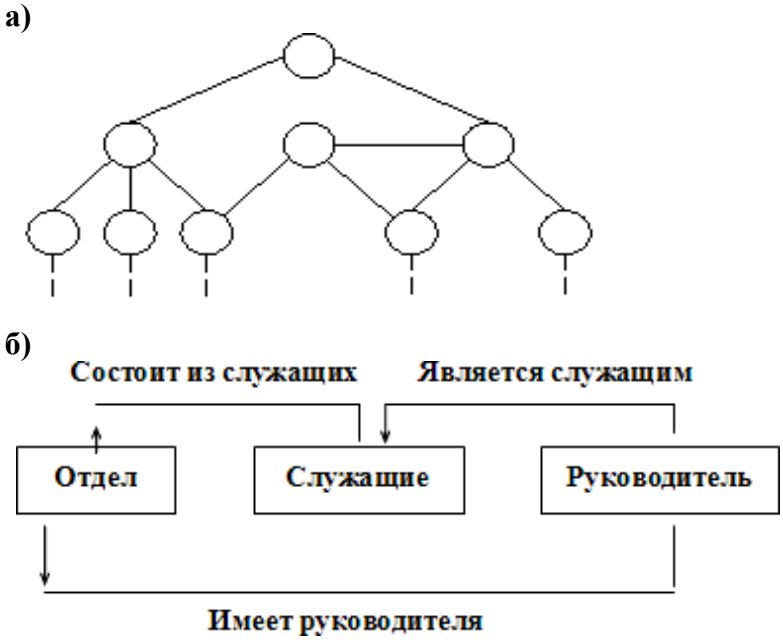


Рисунок 3 – Примеры сетевых моделей данных

В узлах графа помещаются типы записей, а ребра интерпретируются как связи между типами записей, причем объект-потомок может иметь не одного, а любое количество объектов-предков. Кроме того, допускаются любые связи-отношения, в том числе и одноуровневые

Сетевые модели не распространены из-за отсутствия языков, которые бы позволили в разных прикладных ИС одинаковым образом описывать данные сетевой организации.

Реляционная база данных представляет собой хранилище данных, организованных в виде двумерных таблиц .

Любая таблица реляционной базы данных состоит из строк (называемых записями) и столбцов (называемых полями).

Строки таблицы содержат сведения о представленных в ней фактах (или документах, или людях, одним словом, об однотипных объектах). На пересечении столбца и строки находятся конкретные значения содержащихся в таблице данных.

Данные в таблицах удовлетворяют следующим принципам:

- каждое значение на пересечении строки и столбца должно быть атомарным;
- значения данных в одном и том же столбце должны принадлежать к одному и тому же типу;
- каждая запись в таблице уникальна (не существует двух записей полностью совпадающих, что определяется ключом);
- каждое поле имеет уникальное имя ;
- последовательность полей в таблице несущественна;
- последовательность записей в таблице несущественна.

Поле, указывающее на запись в другой таблице, связанную с данной записью, называется внешним ключом.

Связь между двумя таблицами устанавливается путем присвоения значений внешнего ключа одной таблицы значениям первичного ключа другой.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ.

2 ОСНОВЫ РЕЛЯЦИОННОЙ АЛГЕБРЫ

2.1 Терминология реляционной алгебры

Сильную сторону реляционных баз данных представляет развитая математическая теория, лежащая в их основе – реляционная алгебра.

В основе реляционной модели лежит понятие отношения (от англ. Relation), которое выражает не взаимосвязь между таблицами-сущностями, а определение самой таблицы как математического отношения доменов.

Домен – это подмножество элементов.

Кортеж представляет собой упорядоченный конечный набор элементов.

Декартовым произведением доменов D1,D2, …,Dk D=D1xD2x…xDk, называется множество всех кортежей длины k (то есть состоящих из k-элементов) по одному из каждого домена Di: $(d1i_1, d2i_2, \ldots, dki_k)$, где D1={d11, d12, …, d1n1}, D2= {d21, d22, …, d2n2}, …., Dk = {dk1, dk2, …, dknk}.

Пример. Пусть имеется 3 домена: D1={A,2}, D2 = {B,C}, D3 = {4,5, D}. Необходимо определить их декартово произведение D1xD2xD3. Поскольку доменов 3, то строим кортежи длиной 3(берем по одному элементу из каждого домена

D = D1xD2xD3 ={(A, B, 4), (A, B, 5), (A, B, D), (A, C, 4), (A, C, 5), (A, C, D),
(2, B, 4), (2, B, 5), (2, B, D), (2, C, 4), (2, C, 5), (2, C, D)}

Декартово произведение позволяет получить все возможные комбинации элементов исходных множеств.

Определим отношение. Отношение R на множествах D1, D2, …, Dk есть подмножество декартова произведения D1xD2x…xDk, то есть является некоторым подмножеством кортежей арности k (арность – длина кортежа , число элементов нем).

Пример. Имеется 3 домена D1={A,2}, D2 = {B,C}, D3 = {4,5, D}. Их декартово произведение D = D1xD2xD3 ={(A, B, 4), (A, B, 5), (A, B, D), (A, C, 4), (A, C, 5), (A, C, D), (2, B, 4), (2, B, 5), (2, B, D), (2, C, 4), (2, C, 5), (2, C, D)}.

Можно определить отношение

R1={(A,B.4), (A,C,4)} $\subseteq$ D1xD2xD3 или

R2 = {(A,B,4), (A,B,5), (A,B,D), (A,C,4), (2,C,D)} $\subseteq$ D1xD2xD3.

Отношение называется нормализованным, если каждая компонента кортежа является простым, атомарным значением, не состоящим из группы значений.

Нормализованное отношение удобно представлять как таблицу, где

- каждая строка есть кортеж, а
- каждый столбец соответствует одному и тому же компоненту декартова произведения (одному и тому же домену).

Имя таблицы соответствует имени отношения.

Например, отношение R1={(A,B.4), (A,C,4)} $\subseteq$ D1xD2xD3 может быть представлено в виде таблицы

A	B	4
A	C	4

Такая таблица обладает следующими свойствами:

- каждая строка есть кортеж из k значений, принадлежащих k столбцам;
- порядок столбцов фиксирован;
- порядок строк безразличен;
- любые две строки различаются хотя бы одним элементом;
- строки и столбцы таблицы могут обрабатываться в любой последовательности.

Чтобы устранить необходимость фиксированного порядка столбцов в отношении, их именуют.

Столбцы таблицы (отношения) – это атрибуты. Множество значений атрибута есть элементы домена. Список имен атрибутов отношения называется схемой отношения, которая обозначается, как

R(A1, A2, …, Ak),

где Ai, i=1,…,k– атрибуты. По сути здесь представлены имя таблицы (R) и имена столбцов (Ai).

В отношении могут существовать несколько атрибутов, которые однозначно идентифицируют кортеж. Такие атрибуты называются возможными ключами. Один из них выбирается в качестве первичного ключа для обеспечения доступа к кортежам.

Схему реляционной базы можно представить в виде:

R1(A11, A12, …, $A1k_1$)

R2(A21, A22, …, $A2k_2$)

……

Rm(Am1, Am2, …, Amk_m)

2.2 Основные операции реляционной алгебры

Для получения информации из отношений необходим язык манипулирования данными (ЯМД), выполняющий соответствующие операции над отношениями.

Для описания ЯМД используют 3 абстрактных теоретических языка:

- реляционная алгебра,
- реляционное исчисление с кортежами и

– реляционное исчисление с доменами,

По своей выразительности все три языка оказались эквивалентными.

При определении операций реляционной алгебры предполагается, что порядок столбцов в отношении фиксирован, а сами отношения конечны.

Пять операций реляционной алгебры являются основными

– проекция,
– объединение,
– разность,
– декартово произведение и
– селекция.

Другие часто используемые операции пересечения, соединения и деления можно выразить через пять основных операций. Рассмотрим основные операции реляционной алгебры.

1. Объединение отношений R1 и R2 (R=R1UR2) - отношение, состоящее из множества всех кортежей, принадлежащих R1 и R2 за исключением повторяющихся. Эта операция применяется к отношениям одной арности.

R1

а	**б**	**в**
д	**е**	**а**
и	**к**	**л**

R2

и	**к**	**л**
г	**д**	**е**

R1UR2

а	**б**	**в**
д	**е**	**а**
и	**к**	**л**
г	**д**	**е**

Это можно записать и так

R1{(а, б, в), (д, е, а), (и, к, л)}

R2{(и, к, л), (г, д, е)}

R3 = R1UR2 = {(а, б, в), (д, е, а), (и, к, л), (г, д, е)}

2. Разность отношений R1 и R2 (R=R1-R2) - это отношение, состоящее из множества кортежей, принадлежащих R1 и не принадлежащих R2. Операция применяется к отношениям одной арности

R1

а	**б**	**в**
д	е	а
и	к	л

R2

и	**к**	**л**
г	д	е

R1 - R2

а	**б**	**в**
д	**е**	**а**

Это можно записать и так

R1{(а, б, в), (д, е, а), (и, к, л)}

R2{(и, к, л), (г, д, е)}

R3 = R1 - R2 = {(а, б, в), (д, е, а)}

3. Декартово произведение отношений R1 и R2 (R=R1xR2) – это множество кортежей арности (k1+k2), причем первые k1 элементов образуют кортеж из отношения R1, а последние k2 элементов образуют кортеж из отношения R2.

R1

а	**б**	**в**
д	**е**	**а**
и	**к**	**л**

R2

и	**к**	**л**
г	**д**	**е**

R1xR2

а	**б**	**в**	**и**	**к**	**л**
д	**е**	**а**	**и**	**к**	**л**
и	**к**	**л**	**и**	**к**	**л**
а	**б**	**в**	**г**	**д**	**е**
д	**е**	**а**	**г**	**д**	**е**
и	**к**	**л**	**г**	**д**	**е**

Это можно записать и так

R1{(а, б, в), (д, е, а), (и, к, л)}

R2{(и, к, л), (г, д, е)}

R3 = R1xR2 = {(а, б, в, и, к, л), (д, е, а, и, к, л), (и, к, л , и, к, л),

(а, б, в, г, д, е), (д, е, а, г, д, е), (и, к, л , г, д, е)}

4. Проекция отношения R1 на компоненты i1, i2, …, ik заключается в том, что из отношения R1 выбираются указанные столбцы (i1, i2, …, ik) и компонуются в указанном порядке, записывается как $R=\pi_{i1,\, i2,\ldots,ik}(R1)$

R1

а	**б**	**в**
д	**е**	**а**
и	**к**	**л**

$R=\pi_{3,1}(R1)$

в	а
а	д
л	и

Это можно записать и так

R1{(а, б, в), (д, е, а), (и, к, л)}

$R = \pi_{3,1}(R1) = \{(в, а), (а, д), (л, и)\}$

5. Селекция отношения R1 по формуле F $R=\sigma_F(R1)$ - это операция выборки строк (кортежей), удовлетворяющих формуле F, где F – формула, образованная операндами, являющимися номерами столбцов, и логическими операторами ($\wedge$-и, $\vee$-или, $\neg$-не), и арифметическими операторами сравнения ($<, >, =, \geq, \neq, \leq$).

R1

а	б	в
д	е	а
и	к	л

$R=\sigma_{1=а \,\vee\, 1=д}\,(R1)$

а	**б**	**в**
д	**е**	**а**

Это можно записать и так

R1{(а, б, в), (д, е, а), (и, к, л)}

$R=\sigma_{1=а \vee 1=д}(R1) = \{(а, б, в), (д, е, а)\}$

Рассмотрим операцию пересечения отношений R1 и R2. Она выражается через известные операции следующим образом

$$R=R1 \cap R2 = R1- (R1-R2)$$

R1

и	к	а	н
с	а	м	д
и	к	б	в
о	л	р	б
д	в	е	и

R2

и	у	н	а
с	а	м	д
р	к	б	в
о	л	р	б
е	д	в	и

R1- (R1-R2)

с	а	м	д
о	л	р	б

2.3 Требования к табличной форме

Поскольку в реляционной модели данные представлены в двумерных таблицах рассмотрим требования к таблицам.

Первое требование– это конечность. Конечное число строк и столбцов таблицы. Работать с большими (бесконечными) таблицами неудобно и довольно редко при этом оправдываются затраченные усилия

Второе требование касается имени таблицы. Каждая таблица БД должна обладать уникальным именем (имя таблицы, как правило, указывается в единственном числе);

Заголовок таблицы должен обязательно состоять из одной строки, перечня столбцов, причем с уникальными именами. Многоярусные заголовки не допускаются.

A			**B**		**C**
1	**2**	**3**	**1**	**2**	
...	...	...	...	...	...

Все многоярусные заголовки заменяются одноярусными путем подбора подходящих заголовков.

A_1	A_2	A_3	B_1	B_2	C
...	...	...	...	...	...

Порядок строк и столбцов произвольный , несущественен. Это требование не является строго ограничительным. Если необходимо, можно ввести дополнительный столбец, который будет определять порядок строк. В этом случае от перестановки строк тоже ничего не изменится.

...	...	**Порядок**
...	...	1
...	...	3
...	...	2

Данные во всех столбцах должны быть одного и того же типа.

Сотрудник

ФИО	**Должность**	**Дата**	**Зарплата**
Петров С.С.	инженер	20.05.90	25000
Попов Н.А.	ст.инженер	10.06.91	тридцать тысяч

В столбце Зарплата представлены значения двух разных типов данных, что не допустимо.

Значения в ячейках таблицы должны быть атомарны, ячейка не может содержать множественной информации. Значения должны быть простого типа.

Простой тип данных – это такой тип, значения данных которого не содержат составных частей.

Сотрудник

ФИО	Должность	Дата	Зарплата	Имя_реб	Дата_рожд
Петров С.С.	инженер	20.05.90	25000	Коля, Катя	05.09.11, 10.12.12
Попов Н.А.	ст.инженер	10.06.91	30000		

В столбцах Имя_реб и Дата_рожд содержится множественная информация. Таблицу следует преобразовать. Строку с множественной информации разбиваем на 2 строки и получаем таблицу с атомарными значениями в ячейках.

Сотрудник

ФИО	Должность	Дата	Зарплата	Имя реб	Дата рожд
Петров С.С.	инженер	20.05.90	25000	Коля	05.09.11
Петров С.С.	инженер	20.05.90	25000	Катя	10.12.12
Попов Н.А.	ст.инженер	10.06.91	30000		

Повторение столбцов с одинаковой информацией, например, (Имя_реб1, Дата_р1, Имя_реб2, Дата_р2), не рекомендуется. В этом случае данные лучше выделить в отдельную таблицу.

Ребенок

Имя реб	Дата рожд
Коля	05.09.11
Катя	10.12.12

Еще одно требование касается наличия строк-дубликатов. В таблице не должно быть строк-дубликатов. Если же в таблице встречаются повторяющиеся строки, и они необходимы (например, экземпляры книг), то это можно исправить введением

дополнительного столбца, отвечающего за количество дубликатов каждой строки (записи).

...	...	**Число дубликатов**
...	...	0
...	...	3
...	...	1

Рассмотренный абстрактный язык реляционной алгебры служат основой реальных языков манипулирования данных (ЯМД) реляционных систем.

В общем случае ЯМД выходят за рамки абстрактных языков, так как включают такие команды, как ввод данных, модифицирование данных, удаление данных.

3 НОРМАЛИЗАЦИЯ ТАБЛИЦ БД

3.1 Понятие ключей

Для идентификации строк таблицы используют ключи, В реляционной БД это единственный способ сослаться на строку. Ключи бывают разные. Рассмотрим некоторые из них.

Ключ – это один или нескольких столбцов (полей) таблицы, для которых объявлено условие уникальности значений в строках таблицы.

Условие (или ограничение) уникальности означает, что все значения строк, соответствующих ключам, уникальны, единственны в своем отношении, например номер зачетной книжки (№_зачетной_книжки) в таблице «Студент».

Студент

№ зачетной книжки	Фамилия	Имя	Отчество	...
...	...	...	...	...
...	...	...	...	...

Простой ключ - это ключ, состоящий из одного и не более столбцов.

Составной ключ - это ключ, состоящий из двух и более столбцов. Например, в таблице «Список учебных аудиторий» ключом будет номер аудитории и номер корпуса.

Суперключ (superkey) - это ключ, идентифицирующий каждую запись таблицы, но при этом на него не накладывается условие минимальности числа столбцов. Иными словами, суперключ функционально определяет все атрибуты сущности, и в этом смысле сама схема отношения (таблицы) заведомо является суперключом. Можно сказать, что суперключ таблицы – это любое надмножество любого ключа.

Потенциальный ключ – это один или несколько столбцов, которые позволяют однозначно отличить одну запись таблицы от другой, и при этом среди них не содержится другой набор столбцов, обладающий таким же свойством. Например, в отношении (таблице) Сотрудник (ФИО, Должн, Дата, Зарпл, ИНН, Паспорт) два потенциальных ключа – ИНН и Паспорт.

Первичный ключ – это тот потенциальный ключ, который выбран для реализации в таблице (Primary Key, PK). Важно, что допустимо объявление одного и только одного первичного ключа, и столбцы первичного ключа ни в коем случае не могут принимать неизвестные или неопределенные значения (Null-значения).

Каждая таблица должна содержать первичный ключ, и он может быть в таблице только один. При выборе первичного ключа из набора потенциальных необходимо учитывать, что

– ключ должен содержать как можно меньше столбцов, чтобы быстродействие системы поиска было выше;

– максимальное количество байт информации в столбцах выбранного ключа должно быть как можно меньше;

– столбцы ключа не должны допускать отсутствие информации в ячейках;

– информация ключа не должна изменяться (изменение данных ключа не должно допускаться).

Внешний ключ – это ключ, объявленный в таблице, и который при этом ссылается на первичный или потенциальный ключ той же самой или какой-то другой таблицы. Внешний ключ обозначается как foreign key (FK).

Таблица, на которую ссылается внешний ключ, называется родительской таблицей. Таблица, содержащая внешний ключ, называется дочерней.

Потенциальные ключи, которые не выбраны в качестве первичного, называют альтернативными (Alternative Key – AK).

Суррогатный ключ - искусственный ключ без смысловой нагрузки. Как правило, суррогатный ключ — это числовое поле, в которое последовательно заносятся возрастающие. Во многих СУБД существует специальный тип данных для таких полей, когда при добавлении записи в таблицу автоматически записывается уникальное для этой таблицы числовое значение. Суррогатные ключи могут привести к неправильному заполнению таблицы.

Ребенок

Номер	Имя реб	Дата рожд	Табельный номер
1	Коля	05.09.11	1
2	Катя	10.12.12	1
3	Вася	29.01.13	2
4	Катя	10.12.12	1

3.2 Виды связи

Для того, чтобы связать две таблицы необходимо ввести в структуру одной таблицы внешний ключ, т.е. поле, не являющееся ключевым в этой первой таблице, но являющееся ключевым во второй.

Будем говорить, что некоторая запись таблицы А связана с некоторой записью таблицы В, если в обеих таблицах эти записи содержат одно и то же значение в поле, по которому установлена связь между таблицами.

Различают 4 типа связей между таблицами реляционной базы данных:

– «один-к-одному» (1 - 1);

– «один-ко-многим» (1 - М);

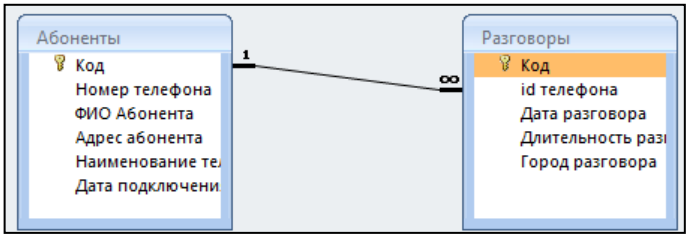
– «многие-к-одному» (М - 1);

– «многие-ко-многим» (М - М).

Таблицы А и В находятся в отношении «один-к-одному», если каждая запись в таблице А имеет не более одной связанной с ней записи в таблице В и наоборот, каждая запись в таблице В имеет не более одной связанной с ней записи в таблице А.

Таблицы А и В находятся в отношении «один-ко-многим», если каждая запись в таблице А может быть связана с 0, 1 или несколькими записями таблицы В, но каждая запись в таблице В не может быть связана более чем с одной записью таблицы А.

Таблица А в этом случае называется главной таблицей, а таблица В – подчиненной.



Один абонент может заказать много разговоров, но каждый разговор заказан конкретным абонентом.

Таблицы А и В находятся в отношении «многие-к-одному», если каждая запись в таблице А связана только с одной записью таблицы В, но каждая запись таблицы В связана с 0, 1 или несколькими записями таблицы А.

Связи «один-ко-многим» и многие-к-одному» являются наиболее распространенными в реляционных БД. Эти виды связей в реляционной модели устанавливаются через введение в таблицах дополнительных полей, которые дублируют ключевые поля связанной таблицы.

Внешний ключ

Сотрудник					
Таб_ном	Фамилия	Имя	Отчест	Должн	№ отд

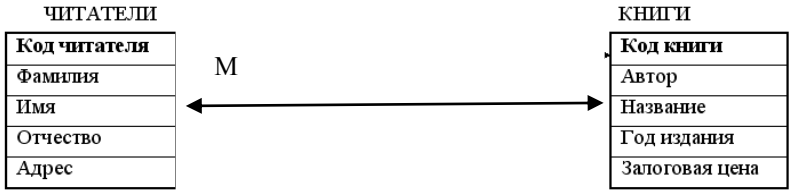
←

Отдел			
№ отд	Наименование	Руков-ль	Кол-во людей

Таблица, к которой связь подходит к первичному ключу, называют главной таблицей (Отдел), а вторую таблицу называется подчиненной или дочерней (Сотрудник).

Таблицы А и В находятся в отношении «многие-ко-многим», если каждая запись таблицы А может быть связана с 0, 1 или несколькими записями в таблице В, и наоборот,

каждая запись таблицы В может быть связана с 0,1 или несколькими записями в таблице А.



Многие СУБД не могут отражать связи типа Многие-ко-многим (Читатели – Книги). Проблема решается введением промежуточной (дополнительной, ассоциативной) таблицы (Абонемент), куда вводятся первичные ключи («Код читателя» и «Код книги») связываемых таблиц (Читатели и Книги).

Связи выполняют более важную роль, чем просто информация о размещении данных по таблицам. Прежде всего, они требуются разработчикам для поддержания целостности баз данных. Правильно настроив связи, можно быть уверенным, что ничего не потеряется.

3.3 Целостность данных

Под целостностью баз данных будем понимать некоторый набор требований, нарушение которых приведет к противоречию между базой данных и предметной областью которую она отражает. Целостность данных - это механизм поддержания соответствия базы данных предметной области.

Конечно, СУБД не может контролировать правильность каждого отдельного значения, вводимого в базу данных. Например, нельзя обнаружить, что вводимое значение 33, представляющее число отработанных часов в неделю, в действительности должно быть равно 35. Но значение 168, явно будет ошибочным и система должна его отвергнуть.

Целостность базы данных может быть нарушена вследствие сбоя оборудования, ошибки пользователя или программной ошибки. В системах со многими пользователями целостность может быть нарушена при одновременном обращении к одним и тем же фрагментам данных.

Целостность обеспечивается путем задания ограничений. Современные СУБД поддерживают возможности соблюдения тех или иных требований целостности.

Ограничение целостности реляционной модели касается

– целостности сущностей (таблиц) и

– целостности ссылок (связей).

Требование целостности сущностей (таблиц) заключается прежде всего в требовании уникальности каждого кортежа (записи). Это достигается:

– отсутствием записей дубликатов (отсутствием дубликатов по полям первичных ключей, что не исключает совпадения значений по другим полям);

– отсутствием полей с множественным характером значений в столбцах, что обеспечивается нормализацией таблиц.

Требование целостности ссылок или ссылочная целостность предполагает, что для любой записи с конкретным значением внешнего ключа должна обязательно существовать запись связанной таблицы с соответствующим значением первичного ключа.

Если существует запись в Таблице «Сотрудник» для Иванова, работающего в отделе №7, то в таблице «Отдел» должна быть запись с соответствующим номером отдела (7).

Сотрудник					
Таб_ном	Фамилия	Имя	Отчест	Должн	№ отд

←

Отдел			
№ отд	Наименование	Руков-ль	Кол-во людей

Ссылочная целостность играет большую роль в организации СУБД. Дело в том, что ввод данных в отдельные таблицы базы данных можно сделать только поочередно. Естественно возникает проблема: в каком порядке вводить данные в случае, когда одному и тому же атрибуту соответствуют столбцы разных таблиц?

Типичное решение этой проблемы в современной СУБД – таблицы с одинаковым столбцом упорядочиваются в цепочку таблиц, и до тех пор, пока не введена полная строка в родительской таблице, СУБД запрещает ввод в дочернюю таблицу .

3.4 Нормализация таблиц

Этап проектирования структуры БД является процессом творческим, неоднозначным, а с другой стороны, его основные моменты могут быть формализованы.

Одной из таких формализаций является требование, согласно которому реляционная база данных должна быть нормализована. Процесс нормализации имеет

своей целью устранение избыточности данных и заключается в приведении таблиц БД чаще всего к третей нормальной форме (3НФ).

С содержательной точки зрения нормализация таблиц – это доработка концептуальной (инфологической) модели базы данных.

С формальной точки зрения нормализация - процесс последовательного разбиения и преобразования некоторого исходного набора таблиц для получения набора взаимосвязанных таблиц в нормальных формах.

Обычно выделяют следующие нормальные формы:

- первая нормальная форма (1НФ);
- вторая нормальная форма (2НФ);
- третья нормальная форма (3НФ);
- нормальная форма Бойса-Кодда (НФБК);
- четвертая нормальная форма (4НФ);
- пятая нормальная форма, или нормальная форма проекции-соединения (5НФ).

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле лучше предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

Первая нормальная форма (1НФ) требует, чтобы значения всех полей таблицы были простые (атомарные, неделимые), то есть значение поля не должно быть множеством или повторяющейся группой.

Неделимость поля означает, что значение поля не должно делиться на более мелкие значения. Например, если в поле "Подразделение" содержится название факультета и название кафедры, требование неделимости не соблюдается и необходимо из данного поля выделить или название факультета, или кафедры в отдельное поле.

Сотрудник

Личн. №	Наименование мероприятия	Награда	Фамилия	Звание	Кабин.	Сл.тел
001	Операция «Ы»	Премия	Пронин	Майор	110	11-22-33
001	Операция «Бриллиантовая рука»	Отпуск	Пронин	Майор	110	11-22-33
002	Операция «Берн»	-	Исаев	Полковник	110	11-22-33
007	Операция «Золотой глаз»	Ягуар	Бонд	Капитан	С-110	33-22-11
007	Операция «Багамы»	Феррари	Бонд	Капитан	С-110	33-22-11

Суть первой нормальной формы – атомарность (неделимость) полей и единственность значений по полям в реляционной модели данных.

Таблицы в 1НФ могут

– содержать ситуации дублирования данных (см.поля Фамилия, Звание, Кабинет, Сл.телефон),

– включать аномалии схемы отношений.

Аномалии бывают:

– аномалии вставки (возникают в случаях, когда информацию в таблицу нельзя поместить, пока она неполная; например, нельзя образовать запись для сотрудника, не участвовавшего ни в одной операции);

– аномалии удаления (состоят в том, что при удалении какого-либо данного из таблицы может пропасть и другая информация, которая не связана напрямую с удаляемым данным; например, удаляя запись об участии определенного сотрудника в определенной операции, можно удалить информацию о том, что он вообще работает в определенном подразделении);

– аномалии изменения (проявляются в том, что изменение значения одного данного может повлечь за собой просмотра всей таблицы и соответствующее изменение некоторых других записей таблицы).

Чтобы устранить эти аномалии, необходимо разбить исходную таблицу и перейти ко второй нормальной форме.

Вторая нормальная форма 2НФ использует понятие полной функциональной зависимости.

Отношение (таблица) находится во 2НФ, если оно находится в 1НФ, и каждый неключевой атрибут функционально полно зависит от ключа.

Сотрудник

Личн. №	Наименование мероприятия	Награда	Фамилия	Звание	Кабин.	Сл.тел
001	Операция «Ы»	Премия	Пронин	Майор	110	11-22-33
001	Операция «Бриллиантовая рука»	Отпуск	Пронин	Майор	110	11-22-33
002	Операция «Берн»	-	Исаев	Полковник	110	11-22-33
007	Операция «Золотой глаз»	Ягуар	Бонд	Капитан	С-110	33-22-11
007	Операция «Багамы»	Феррари	Бонд	Капитан	С-110	33-22-11

В таблице «Сотрудник» поле Награда целиком зависит от составного ключа, а остальные поля функционально зависят от части составного ключа - Личн.№.

Для перевода таблицы из 1НФ в 2НФ надо:

- образовать проекцию исходной таблицы на составной ключ и на поля, находящиеся в полной функциональной зависимости от составного ключа;

T1

Лич №	Наименование мероприятия	Награды
001	Операция Ы	Оклад
001	Операция “Бриллиантовая рука”	Отпуск
002	Операция Берн	-
007	Операция “Золотой глаз”	Ягуар
007	Операция “Багамы”	Феррари

- построить еще одну или несколько проекций на часть составного ключа с полями функционально зависящими от этой части ключа.

T2

Лич №	Фамилия	Звание	Кабинет	Сл.тел.
001	Пронин	Майор	110	11-22-33
002	Исаев	Полковник	110	11-22-33
007	Бонд	Капитан	С-110	33-22-11

Третья нормальная форма 3НФ требует отсутствия функциональных зависимостей между неключевыми атрибутами.

Например, в таблице Т2 имеется цепочка функциональной зависимости неключевых атрибутов, которая имеет название транзитивной зависимости:
Лич№ → Кабинет → Сл.телефон.
Имеется транзитивная зависимость поля Сл.телефон от поля Лич№

Отношение (таблица) находится в 3НФ, если оно находится в 2НФ и в нем отсутствуют транзитивные зависимости неключевых атрибутов от ключа (или функциональные зависимости между неключевыми атрибутами).

Для преобразования таблицы из 2НФ в 3НФ

– таблицу разделяют на 2 или более проекций так, чтобы конечные поля в цепочках транзитивной зависимости вынести в отдельные таблицы,

T3

Лич №	Фамилия	Звание	Кабинет
001	Пронин	Майор	110
002	Исаев	Полковник	110
007	Бонд	Капитан	С-110

T4

Кабинет	Сл.телефон
110	11-22-33
С-110	33-22-11

– таблицы связывают внешними ключами по полям, находящимся внутри цепочек транзитивной зависимости (например, по полю Кабинет)

На практике обычно таблицы доводят до третьей нормальной формы.

3.5 Индексные массивы

Индексы в БД – это вспомогательные структуры (массивы), ускоряющие доступ к данным. Индекс позволяет избежать последовательного или пошагового просмотра файла данных в поисках необходимых записей.

Различают линейные и нелинейные структуры индексных массивов.

Примером линейных структур индексных массивов является инвертированный список, который строится по схеме таблицы с 2-мя колонками:

- значение индексируемого поля и
- номера строк.

Значение индексируемого поля	Номера строк
1952	3
1959	5, 17, 123, 256
1960	31, 32
1961	11, 45, 59, 167, 251
1962	7, 8, 9, 10, 234

Для доступа к нужной строке сначала в упорядоченном списке отыскивается строка с требуемым значением поля (например, 1960), затем считывается номер (номера) строки (31 и 32) и по нему осуществляется доступ к искомой строке базовой таблицы.

При добавлении новой строки в базовую таблицу значение ее индексируемого поля ищется в ранее составленном списке, и если такое значение есть (например, 1960), номер новой строки (261) базовой таблицы добавляется в инвертированный список.

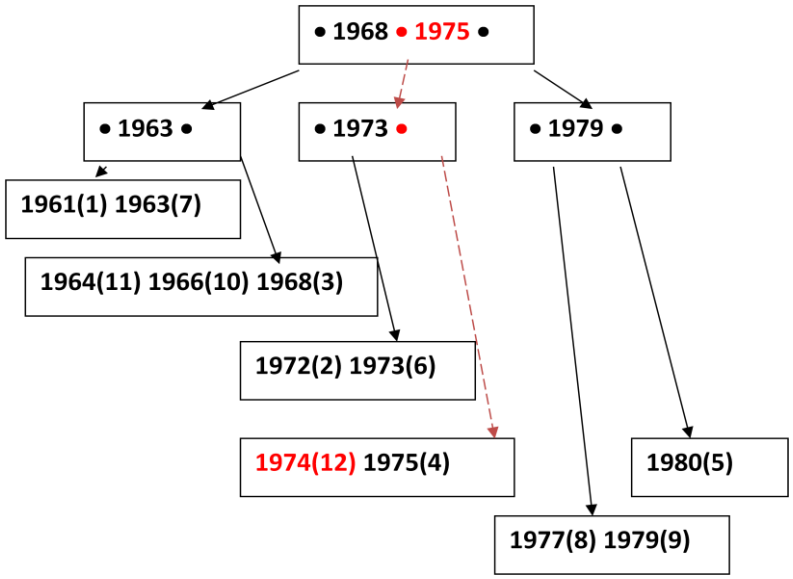
Значение индексируемого поля	Номера строк
1952	3
1959	5, 17, 123, 256
1960	31, 32, 261
1961	11, 45, 59, 167, 251
1962	7, 8, 9, 10, 234

Нелинейный структуры индексных массивов используют B-деревья, которые представляют собой корневые сбалансированные сильно ветвистые деревья.

Рассмотрим пример для базовой таблицы, приведенной ниже, и для индексируемого поля ее – «Год рождения».

Фамилия	Год рождения
1. Иванов	1961
2. Петров	1972
3. Сидоров	1968
4. Лаврентьев	1975
5. Егоров	1980
6. Максимов	1973
7. Сергеев	1963
8. Григорьев	1977
9. Матвеев	1979
10. Владимиров	1966
11. Николаев	1964
12. Федоров	1974

B-дерево для этого поля имеет следующий вид.



Каждая внутренняя вершина содержит информацию о различных последовательно возрастающих значениях (P_i X_i)индексируемого поля (например, •1968•1975•). Точкой обозначен указатель P_i на вершину, содержащую значение индексируемого поля, меньшие или равные X_i (например, первый указатель на вершину со значениями ≤ 1968, второй – на вершину со значениями ≤ 1975 и > 1968, третий – на вершину со значениями > 1975).

Листовая вершина содержит информацию о нахождении страницы в файле базы данных с записями, имеющими соответствующие значения индексируемого поля.

Пусть ищется запись с значением индексируемого поля 1974.

В оперативную память сначала считывается страница с корневой вершиной и последовательно просматривается до первого значения, превышающего значение индексируемого поля нужной записи (1975), при этом определяется ссылка (номер) страницы-потомка (внутренней или листовой).

В оперативную память считывается страница потомок (•1975•), если она внутренняя, ее обработка производится аналогично, если она является листовой (1974(12) 1975(4)), то она просматривается до нахождения нужного значения индексируемого поля и определяется номер страницы файла данных, которая содержит нужную запись (12).

Таковы основы работы индексов.

Индекс обязательно создается в автоматическом режиме для столбцов, на которые наложено ограничение уникальности, то есть для первичных ключей.

При внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу.

4 ЯЗЫКИ БАЗ ДАННЫХ

4.1 Общая характеристика языков БД

Для работы с базами данных используются специальные языки, в целом называемые языками баз данных

К ним относится язык описания данных на внешнем уровне. Он используется для описания требований пользователей и прикладных программ, а также для создания инфологической(концептуальной) модели БД.

Язык описания данных (DDL - Data Definition Language), предназначен для описания данных на разных уровнях абстракции: внешнем, логическом и внутреннем, и в большинстве СУБД является языком описания схем.

Языки общения с БД включают языки манипулирования данными и языки запросов.

Язык манипулирования данными (DML - Data Manipulation Language) используется для обработки данных, их преобразований и написания программ.

Базовый язык DML - это один из традиционных языков программирования (BASIC, C, FORTRAN и др.). Этот язык характерен для систем открытых.

Автономный язык DML - это собственный язык СУБД, который дает возможность выполнять различные операции с данными, и в этом случае системы являются закрытыми.

Для упрощения процедур поиска данных в БД предусмотрен язык запросов. Выделяют

– язык запросов SQL (Structured Query Language - структурированный язык запросов), который был создан фирмой IBM в рамках работы над проектом построения системы управления реляционными базами данных в начале 70-х годов;

– язык запросов QBE (Query By Example) - это реализация запросов по образцу в виде таблиц. Для определения запроса к БД пользователь должен заполнить предоставленную системой таблицу QBE и определить в ней критерии поиска и выбора данных.

Язык ведения диалога обеспечивает· поддержку интерфейса с пользователями, и по мере развития и совершенствования СУБД этот интерфейс становится все более дружественным.

4.2 Характеристика языка запросов QBE на примере MS Access

4.2.1 Основные объекты MS Access

Таблицы хранят данные и структуру базы.

Таблицу образуют строки и столбцы. Их аналоги в БД – записи и поля.

Запросы служат для извлечения данных из таблиц и представления их пользователю в удобном виде. Особенность запросов в том, что они используют данные из базовых таблиц и создают на их основе временную результирующую таблицу.

Формы – это средства для ввода данных. Формы позволяют

- предоставить пользователю средства для заполнения только тех полей, которые ему заполнить положено;
- расположить специальные элементы управления для автоматизации ввода: переключатели., кнопки, флажки и прочее.

Отчеты - это форматированное представление данных, которое выводится на экран, в печать или файл. Они позволяют извлечь из базы нужные сведения и представить их в виде, удобном для восприятия

Макрос - структурированное описание одного или нескольких действий, которые должен выполнить Access в ответ на определенное событие (например, в ответ на выбор некоторого элемента в основной форме открывает другую форму).

Модуль содержит программы, написанные на языке Visual Basic для приложений.

Страницы доступа служат для обеспечения доступа к данным, содержащимся в базе, удалённой от потребителя (например, через *Интернет*).

4.2.2 Работа с таблицами

Существуют два основных режима работы с таблицами: режим Таблицы (Datasheet View) и режим Конструктора (Design View),

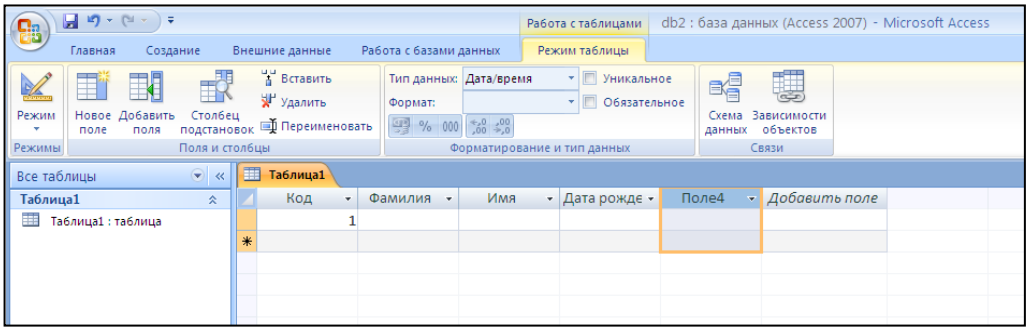
В режиме Таблицы осуществляется работа с данными, находящимися в таблице: просмотр, редактирование, добавление, сортировка и т. п. В этом режиме можно и создавать таблицы.

Access предлагает шаблон таблицы. таблицы вначале имеют имена, данные по умолчанию: Поле1, Поле2 и т.д.

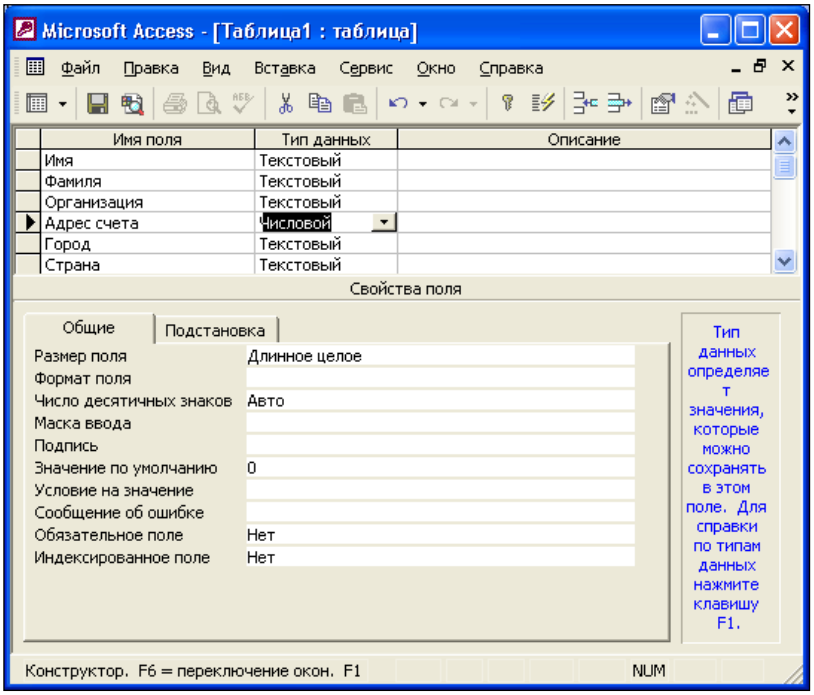
Пользователь может присвоить полям новые имена, которые будут нести смысловую нагрузку, выделив столбец и осуществив щелчок по кнопке Переименовать. Можно изменить ширину столбца или высоту строки.

Символ * обозначает новую запись таблицы.

К этому режиму всегда можно вернуться, открыв таблицу.



В режиме Конструктора таблиц пользователю предлагается графический бланк для создания и редактирования структуры таблицы.



В верхней части указываются: имена полей таблицы, тип данных для каждого поля (столбца), описание или характеристика поля. Заполнение двух первых столбцов является обязательным. Ключевое поле таблицы помечается специальным значком - ключик в поле выделения в левой части окна.

В нижней части указываются значения свойств выделенного поля таблицы.

Больше всего свойств имеет текстовое поле. Свойства текстового поля:

– Размер – может находиться в пределах от 1 то 255 символов;

– Формат – спец.символы, которые задают вид и размер выводимых строк;

– Маска ввода - последовательность кодовых символов для управления вводом ;

– Подпись – это второй идентификатор поля, используется в табличной форме для создания заголовка столбца;

– Значение по умолчанию – автоматически подставляет заданное значение во все вновь создаваемые поля;

– Условие на значение - позволяет создать фильтр, который разрешит вводить в поле только то, что удовлетворяет определенному условию;

– Обязательное поле - является логическим и принимает значения Да или Нет;

– Пустые строки - логическое, определяет, разрешены или нет в данном поле пустые строки, то есть поле, содержащее пробелы;

– Индексированное поле - позволяет установить индекс на поле.

Кроме текстового поля в Access представлены и другие типы данных, каждый из которых имеет свои свойства:

– числовой тип - применяется для хранения числовых данных, используемых в математических расчетах;

– дата/время - тип для представления даты и времени;

– денежный - тип данных для хранения данных с точностью представления от 1 до 4 десятичных знаков;

– счетчик - поле содержит число, определяемое автоматически для каждой новой записи либо случайным образом, либо путем увеличения предыдущего значения на 1;

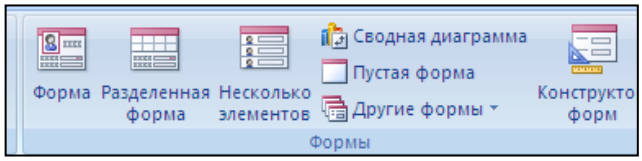
– логическое поле, которое может содержать только два значения, интерпретируемых как Да/Нет, Истина/Ложь, Включено/Выключено;

– поле объекта OLE - содержит ссылку на OLE-объект (лист Microsoft Excel, документ Microsoft Word, звук, рисунок и т. п.).

4.2.3 Работа с формами

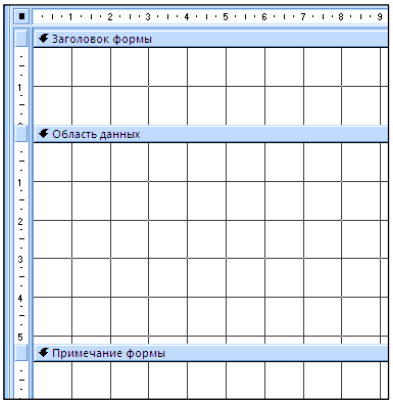
Работа с формами. Формы используются в приложении для ввода и отображения данных. Формы могут применяться для управления доступом к данным - с их помощью можно определять, какие поля или строки данных будут отображаться. Для автоматизации часто выполняемых действий в форму можно добавить кнопки и другие функциональные элементы.

Самым простым способом создания форм является средства автоматического создания форм на основе таблицы или запроса.



Созданная с помощью этих средств форма готова к использованию, но можно улучшить ее внешний вид, изменив некоторые параметры в режиме Конструктора. В этом режиме можно изменить источник данных для формы, количество отображаемых полей,

внешний вид формы и элементов управления, добавить или удалить элементы управления, настроить их свойства.



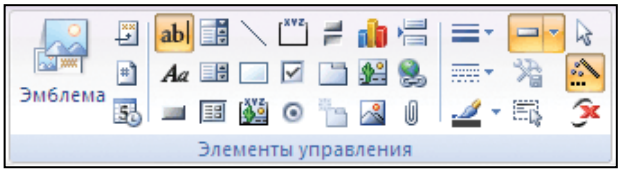
В режиме Конструктора формы могут содержать следующие разделы:

– заголовок (имеет оформительское значение и содержит название формы),

– область данных (имеет содержательное значение, здесь представлены элементы управления для ввода данных или их отображения),

– примечание, имеет оформительское значение.

Элементы управления – элементы, улучшающие интерфейс пользователя, используются для отображения данных или выполнения других действий (позволяют просматривать данные и работать с ними). Часто используются поле, надписи, флажки и др.элементы



Элементы управления могут быть

– присоединенными - источник данных представляет собой поле таблицы или запроса;

– свободными - не имеющими источника данных;

– вычисляемыми - источник данных есть выражение.

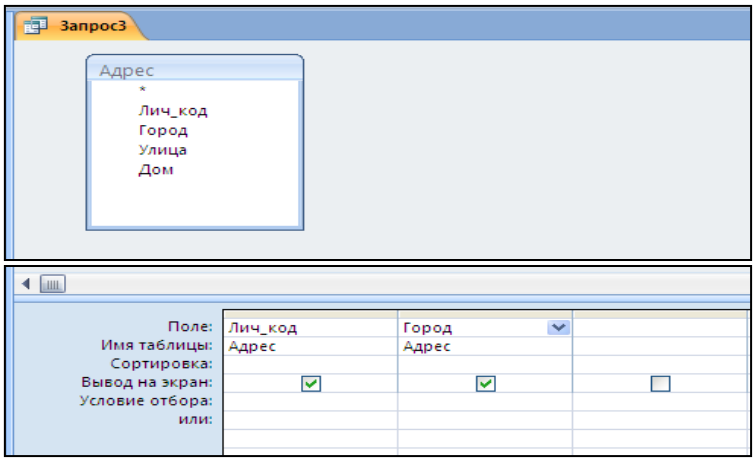
Значения свойств всей формы, ее разделов и каждого элемента управления в форме задаются в окнах свойств соответствующих элементов.

Основной целью форм является создание удобного интерфейса для ввода и изменения данных в одной или более таблицах, являющихся источниками данных формы.

4.2.4 Работа с запросами, отчетами

Запросы используются для просмотра, анализа и изменения данных в одной или нескольких таблицах. Например, можно использовать запрос для отображения данных из одной или нескольких таблиц и отсортировать их в определенном порядке, выполнить вычисления над группой записей, осуществить выборку из таблицы по определенным условиям. Запросы могут служить источником данных для форм и отчетов. Запрос не содержит данных, но позволяет выбирать данные из таблиц.

При создании *запроса в режиме Конструктора* в верхней части размещается структура таблиц, к которым адресован запрос, а в нижней – столбцы результирующей таблицы, которая будет сформирована в результате выполнения запроса



Адрес

Лич_код	Город	Улица	Дом
л-001	Серово	Еловая	24
л-011	Москва	Лесная	112
к-001	Москва	Русская	67
к-001	Киев	Манежная	45
к-002	Петербург	Серая	12
к-002	Минск	Моховая	11

Исходная таблица

Адрес | Запрос на выборку

Лич_код	Город
к-001	Москва
к-001	Киев
к-002	Петербург
к-002	Минск
л-001	Серово
л-011	Москва

Результат запроса

Запросы на выборку – самые простые и наиболее распространенные. Существуют другие виды запросов:

- запросы с параметрами;
- итоговые запросы;
- запросы на изменение или запросы действия;
- перекрестные запросы.

4.2.5 Работа с отчетами

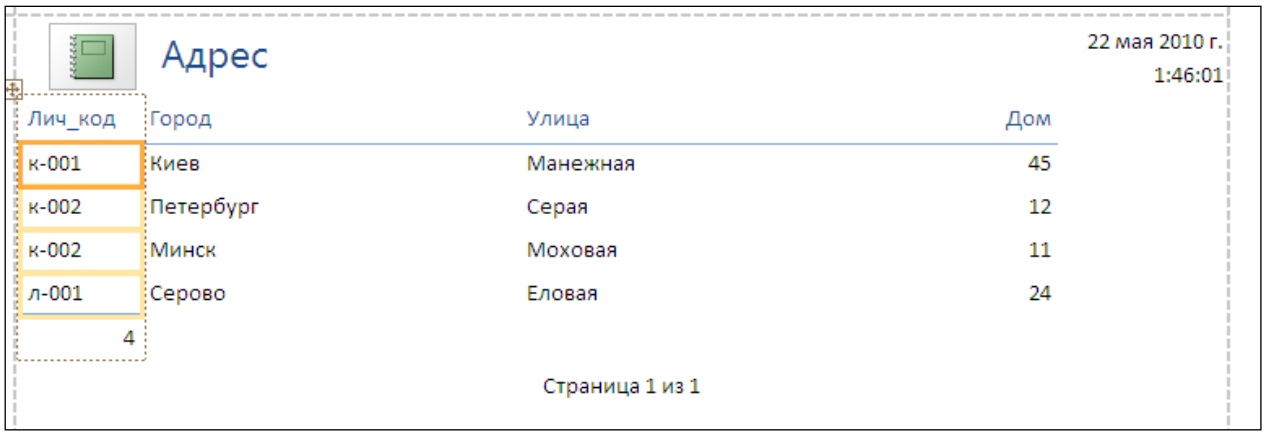
Отчеты позволяют выбрать из базы данных требуемую пользователем информацию и оформить ее в виде документов, которые можно просмотреть и напечатать. Источником данных для отчета может быть таблица или запрос. Кроме данных, полученных из таблиц, в отчете могут отображаться вычисленные по исходным данным значения.

Отчеты и формы Access имеют много общего. Но, в отличие от форм, отчеты не предназначены для ввода и правки данных в таблицах. Они позволяют лишь просматривать и печатать данные.

Разделы отчета подобны разделам формы и включают заголовок и примечание отчета, область данных, а также верхний и нижний колонтитулы. В примечание отчета часто помещают поля с итоговыми значениями.

Наиболее простым способом создания отчетов является использование средств автоматического создания отчета. Средство «Отчет» — это самый быстрый способ создания отчета, потому что с его помощью отчет формируется сразу же, без запроса дополнительной информации.

В отчете будут представлены все записи базовой таблицы или запроса.



4.3 Характеристика языка запросов SQL

4.3.1 Стандарты языка. Типы данных

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана реляционная СУБД System R (англ.), для которой был создан специальный язык SEQUEL, позволявший просто управлять данными в этой СУБД. SEQUEL расшифровывалась как англ. Structured English QUEry Language — «структурированный английский язык запросов». Позже язык SEQUEL был переименован в SQL.

Цель – создать язык запросов к БД, с которым мог бы работать любой пользователь не имеющий навыков программирования.

В 1986г Американский национальный институт стандартизации ANSI (American National Standards Institute) принял первый стандарт языка SQL-0 .

В 1989 стандарт был уточнен SQL (level 2) → SQL-1.

В 1992г стандарт ANSI SQL-92 (SQL-2).

В 1999г SQL-99 или SQL-3 и 2003г (SQL-4).

Преимущества SQL:

- простота использования;

– независимость от СУБД (запрос выполняется и в Oracle и Mysql);

– полноценность языка как языка управления данными.

Диалект – это надмножества некоторого подмножества стандартов. Сегодня таких диалектов возникло много.

Стандарт состоит из 2-х частей – обязательной и рекомендательной.

Рассмотрим типы данных в SQL Oracle. Тип данных определяет множество значений, набор операций над этими значениям и способ реализации хранения значений и выполнения операций над этими данными.

1 символьные - хранят алфавитно-цифровые данные:

CHAR или CHAR(n) -символьные строки фиксированной длины,

VARCHAR(n) - символьная строка переменной длины (Varchar2);

2 числовые – используются для хранения нуля и положительных или отрицательных чисел с фиксированной и плавающей точкой (целочисленные и нецелочисленные):

NUMBER (точность, масштаб)

где точность – общее число цифр, масштаб – число десятичных знаков;

3 дата/время - хранит значения в виде точек времени– по умолчанию используется формат DD-MMM-YY HH:MI:SS ;

4 большие двоичные объекты (BLOB - Binary Large Object) - тип данных, предназначенный для хранения изображений, аудио и видео, а также компилированного программного кода;

5 большие текстовые объекты (CLOB - Character Large Object) – позволяют хранить в базе данных до четырех гигабайтов символьных данных.

4.3.2 Операторы SQL

Последовательность символов для определения объектов языка Oracle получила название идентификатора. Объекты языка - это таблицы, столбцы таблиц, пользователи и др. Имя объекта (идентификатор):

- должно начинаться с буквы латинского алфавита;

- может включать символы латинского алфавита, цифры и 3 специальных символа: _ (подчеркивание), # (решетка), $;

- не может использовать - (дефис), (пробел), скобки, !, (.), (,) ;

- не может использовать зарезервированные слова языка;

- может использовать ограничители – двойные кавычки, например, “My table”;

- имеет длину <= 30 байт в Oracle;

- не используют спецсимволы.

К элементам языка SQL относятся операторы. Операторы – это синтаксические конструкции языка для выполнения определенных действий.

1 Арифметические операторы. Арифметические операции используются в выражениях для отрицания, сложения, вычитания, умножения и деления числовых величин. Стандартный набор арифметических операторов

+ – * /

2 Операторы сравнения. Операторы сравнения применяются в условиях, сравнивающих одно выражение с другим.

<, <=, >, >=, =, <> (не равно), != (не равно)

Оператор BETWEEN осуществляет проверку на принадлежность значения диапазону значений

Значение BETWEEN Нижн_гран AND Верх_гран

Границы попадают в допустимый диапазон. Верхняя и нижняя границы могут быть числом, датой, символьным выражением.

Оператор IN производит проверку на принадлежность множеству значений.

Значение IN (список допустимых значений через запятую)

Оператор LIKE сопоставляет значение с шаблоном. В шаблоне используются 2 метасимвола: % (любой набор символов) и _ (1 любой символ).

Значение LIKE ‘шаблон’

Если в какой-то ячейке в таблице базы данных нет значения, то считается, что в этой ячейке находится специальное значение NULL. Это не 0 и не пустая строка. Для работы с Null предусмотрены два специальных условия:

- IS NULL — это условие вернет True, если проверяемое значение равно NULL;
- IS NOT NULL — это условие вернет NULL, если проверяемое значение не равно NULL.

3 Логические операторы. Логические операторы проверяют истину некоторого условия – AND, OR, NOT.

Условие1 OR Условие2 AND Условие3

4 Оператор конкатенации соединяет символьные выражения – II.

4.3.3 Функции SQL

Функция – это правило, по которому каждому элементу одного множества ставится в соответствие некоторый элемент другого множества.

1 Математические функции

SQRT(n) –квадратный корень

SQUARE(n) – квадрат указанного числа

ABS(n) – абсолютное значение

COS(n) – косинус

SIGN(n) – знак n

FLOOR(n) возвращает наибольшее целое, меньшее или равное n

MOD(n1, n2) – остаток от деления n1 на n2

ROUND(n [,m]) - округление n до количества разрядов m. Если m не указан, то округление производится до целого значения.

TRUNC(n [,m]) - усечение до m знаков после десятичной точки, m может не указываться, тогда n усекается до целого

Функции ROUND и TRUNC могут применяться к датам.

2 Функции символьные

UPPER(str) – преобразует все символы строки str в прописные.

LOWER(str) – преобразует все символы строки в строчные.

INITCAP(str) – каждая первая буква каждого слова станет заглавной.

LENGTH(str) – длина строки в символах.

ASCII(str) – возвращает ASCII-код первого символа строки str .

CHR(n) – возвращает символ по его коду.

SUBSTR – извлечение подстроки из строки

SUBSTR (символьное_выражение, число 1, число 2)

число 1 – начало позиции выделения подстроки;

число 2 – число выделяемых символов

INSTR – определяет, на какой позиции находится заданный символ в символьном выражении:

INSTR (символьное_выражение, символ, число 1, число 2)

число 1 – с какой позиции надо искать;

число 2 – какое вхождение символа ищется

3 Агрегатные или групповые функции позволяют проводить операции по значениям в столбце или значениям части столбца.

SUM ([Distinct] имя_столбца) – сумма значений столбца. [Distinct] – необязательный параметр, означает «различных значений»;

AVG ([Distinct] имя столбца) – среднее значение.

Пример. SELECT SUM (Distinct Зарплата), SUM (Зарплата) FROM Сотрудник;

Сотрудник

Таб_ном	ФИО	Зарплата
1	Панов	20000
2	Павлов	30000
3	Путов	30000
4	Пронин	

SUM (Distinct Зарплата)	SUM (Зарплата)
50000	80000

MAX (имя_столбца) – минимальное значение столбца:

MIN (имя_столбца) – максимальное значение столбца:

COUNT – считает число непустых значений.

В большинстве запросов должна быть ссылка на таблицу. Как правило, для таких запросов используется таблица DUAL. Таблица DUAL очень проста. Она состоит из одного поля и содержит одно значение.

4.3.4 Команды SQL

Команды языка определения данных – команды DDL (Data Definition Language). Язык DDL не работает с данными, а работает с объектами базы данных.

Команда создания объекта:

```
CREATE тип_объекта имя_объекта
Параметры;
```

Параметры зависят от того, какой объект создается.

Пример 1.
```
CREATE Table Отделы
(Номер_отд Number(5) Primary KEY,
Название Varchar2(30) NotNull UNIQUE,
Телефон Varchar2(20) Default '111-11-11');
```

Команда изменения свойств объекта:

```
ALTER тип_объекта имя_объекта
Параметры;
```

Пример 2.
```
ALTER TABLE t1
ADD (pole1 char(10));        ←добавление в таблицу t1 столбца pole1
```

Пример 3.
```
ALTER TABLE t1
MODIFY (pole1 char(20));     ←изменение типа данных столбца
```

Пример 4.
```
ALTER TABLE t1
DROP COLUMN pole1;           ←удаление столбца
```

Команда удаления объекта:

DROP тип_объекта имя_объекта;

<u>Команды языка манипулирования данными</u> - команды DML (Data Manupulation Language). Это команды работы с данными. Они позволяют добавлять, изменять, удалять данные таблиц.

Команда вставки данных (записи) в таблицу:

INSERT INTO [TABLE] имя_таблицы [(список столбцов через запятую)]
VALUES(список значений через запятую для каждого столбца);

Пример 5. INSERT INTO Отдел
VALUES (101, ‘Плановый’, ‘555-55-55’, ‘Петров С.С.’);

Команда изменения уже внесенных данных:

UPDATE имя_таблицы
SET имя_столбца=значение
[WHERE условие];

Пример 6. UPDATE Отдел
SET Телефон=’535-77-12’
WHERE Номер_отд=101;

Команда удаление записи из таблицы:

DELETE имя_таблицы
[WHERE условие];

Пример 7. DELETE Employees
WHERE Salary>20000;

<u>Команда языка запросов</u> – команда DQL (Date Query Language) – состоит из одной команды Select. Рассмотрим основные разделы этой команды.

1 WITH – задаются подзапросы и псевдонимы для них;

2 SELECT – указываются выводимые параметры

SELECT [DISTINCT] список_столбцов_через_запятую

3 FROM – указываются таблицы, из которых извлекаются данные

FROM список_таблиц

4 WHERE – задается условие отбора выводимых записей

[WHERE условие_отбора_записей]

В разделе WHERE агрегатные функции использовать нельзя.

5 GROUP BY – указываются параметры группировки данных

[GROUP BY список_столбцов_для_группировки]

6 HAVING – задаются условия отбора сгруппированных записей

[HAVING условие_отбора_сгруппированных_записей]

7 ORDER BY – сортировка записей

[ORDER BY имя_столбца [ASC/DESC], 2-й столбик,3-й столбик . . .] ;

ASC - сортировка по возрастанию, по умолчанию ASC; DESC - сортировка по убыванию. Квадратные скобки означают, что раздел или параметр может отсутствовать, то есть является необязвтельным.

Особенности команды SELECT:

- можно пропускать разделы, но порядок следования должен сохраняться;
- каждый раздел повторяется только один раз;
- каждый раздел должен писаться заглавными буквами;
- каждый раздел пишется с новой строки;

4.3.5 Примеры использования команды Select

Список столбцов в разделе Select может включать * (все столбцы таблицы), имя_столбца, имя_таблицы.имя_столбца, имя_столбца [AS] псевдоним, функцию (например, Sysdate), выражение (например, преобразование столбцов – Зарплата*2), константу, подзапрос.

1 Применение раздела WHERE

Сотрудник

Таб_ном	ФИО	Зарплата	Ном_отд	Долж
1	Антонов	20000	10	инж
2	Алексеев	25000	10	стинж
3	Амосов	25000	20	стинж
4	Алтунин	30000	20	стинж

Вывести ФИО, слово Должность, зарплату для старших инженеров, у которых буква “т” на третьей позиции

```
SELECT ФИО, Должн AS Должность, Зарплата
FROM Сотрудник
WHERE Должн=’стинж’ AND ФИО LIKE ‘_ _т%’;
```

2 Применение раздела GROUP BY

Какова средняя зарплата по отделам?

```
SELECT Ном_отд, AVG(Зарплата) СрЗарплата
FROM Сотрудник
GROUP BY Ном_отд;
```

Если есть раздел GROUP BY, то в разделе SELECT должен быть столбик из GROUP BY и агрегатные функции.

Группировка может осуществляться по нескольким столбцам, например, требуется вычислить среднюю заработную плату по должностям каждого отдела.

```
SELECT  Ном_отд, Долж, AVG(Зарплата) СрЗарплата
FROM  Сотрудник
GROUP BY  Ном_отд, Долж;
```

3 Применение раздела ORDER BY.

Вывести список должностей отсортированных по зарплате.

```
SELECT  DISTINCT  Должн
FROM  Сотрудник
ORDER  BY  Зарплата  DESC;
```

Параметр DISTINCT приводит к тому, что дубли значений не выводятся.

4 Выборка данных из нескольких таблиц и включения этих данных в один результирующий набор реализуется с помощью оператора JOIN. Различают внутреннее и внешнее соединение таблиц.

Сотрудник

Таб_ном	ФИО	Зарплата	Ном_отд	Долж
1	Антонов	20000	10	инж
2	Алексеев	25000	10	стинж
3	Амосов	25000	20	стинж
4	Алтунин	30000	20	стинж
5	Авдеев	30000		

M : 1

Ном_отд	Название
10	Плановый
20	Проектный
30	Внедрение

При выполнении внутреннего соединения 2-х таблиц соединяются (склеиваются) только записи, имеющие общие значения в столбце связи, то есть внутреннее соединение описывает связанные записи.

```
SELECT  ФИО, Название
FROM  Сотрудник  INNER  JOIN  Отдел
ON  Сотрудник.Ном_отд = Отдел.Ном_отд;
```

Внешнее соединение включает в себя внутреннее соединение, а также некоторые строки, отсутствующие в результате внутреннего соединения. Типы внешних соединений:

- левое внешнее соединение - внутреннее соединение и строки левой таблицы, которым нет соответствия в правой таблице;
- правое внешнее соединение - внутреннее соединение и строки правой таблицы, которым нет соответствия в левой таблице;
- полное внешнее соединение - внутреннее соединение и строки правой таблицы, которым нет соответствия в левой таблице, и строки правой таблицы, которым нет соответствия в левой таблице.

```
SELECT  ФИО, Название
```

FROM Сотрудник LEFT OUTER JOIN Отдел

ON Сотрудник.Ном_отд = Отдел.Ном_отд; ←левое внешнее соединение

RIHGT OUTER JOIN ←правое внешнее соединение

FULL OUTER JOIN ←полное внешнее соединение

5 Кросс-соединение (декартово соединение) дает в результате всевозможные сочетания записей двух таблиц.

SELECT ФИО, Название

FROM Сотрудник CROSS JOIN Отдел;

6 Подзапрос в разделе WHERE.

Вывести фамилии всех студентов, стипендия которых выше средней стипендии.

SELECT Фамилия

FROM Студенты

WHERE Стипендия > (SELECT AVG(Стипендия) FROM Студенты);

7 Подзапрос в разделе SELECT.

Вывести фамилии студентов, стипендию и процент, который составляет эта стипендия от общей суммы.

SELECT Фамилия, Стипендия, Стипендия*100/

(SELECT SUM(Стипендия) FROM Студенты) AS Процент

FROM Студенты;

8 Подзапрос в разделе FROM.

Вывести фамилии всех студентов, стипендия которых выше средней стипендии их группы.

SELECT Фамилия FROM Студенты INNER JOIN

(SELECT Группа, AVR(Стипендия) Ср_ст

FROM Студенты

GROUP BY Группа) Ср_ст_гр

ON Студенты.Группа = Ср_ст_гр. Группа

WHERE Стипедия > Ср_ст;

4.3.6 Операторы работы с множеством записей

Преподаватель

Таб_ном	Фамилия	Зарплата
1	Попов	10000
2	Петров	12000
3	Антонов	8000

Студент

Номер_с	Фамилия_с	Стипендия
1	Алексеев	1500
2	Андреев	2000
3	Антонов	8000

1 Объединение UNION. Объединением двух множеств называется множество всех элементов, принадлежащих какому-либо одному или обоим исходным множествам. Две таблицы совместимы по объединению (и к ним может быть применен оператор объединения UNION) тогда и только тогда, когда:

– они имеют одинаковое число полей;

– поля первой таблицы и поля второй таблицы имеют в одинаковый тип данных.

```
SELECT  Фамилия FROM  Преподаватель
UNION
SELECT Фамилия_с  FROM Студент;
```

2 UNION ALL позволяет, в отличие от UNION, разрешить выборку повторяющихся значений, то есть выводит все дубли строк (все записи первого SELECT, затем второго).

Вывести фамилии преподавателей и студентов, их зарплаты и стипендии с указанием их статуса – преподаватель или студент.

```
SELECT  Фамилия, Зарплата “Зарпл\Стип”, ‘Преподаватель’ Статус
FROM  Преподаватели
UNION ALL
SELECT Фамилия_с, Стипендия, ‘Студент’
FROM Студенты;
```

3 INTERSECT – оператор пересечения множества записей, результат его работы - те записи первого SELECT, которые присутствуют по втором SELECT.

```
SELECT  Фамилия, Зарплата “Зарпл\Стип”
FROM  Преподаватели
INTERSECT
SELECT Фамилия_с, Стипендия
FROM Студенты;
```

4 MINUS – оператор вычитания 2-х множеств, при этом остаются записи первого SELECT за исключением тех записей, которые есть во втором SELECT

5 ПРОЕКТИРОВАНИЕ БД. CASE-СИСТЕМЫ

5.1 Проектирование БД. Основные этапы

Проектирование БД – сложная задача, связанных с созданием информационной системы. Кратко рассмотрим основные этапы проектирования БД.

1. Описание предметной области БД. Описание БД дается словесное, в нем формулируются требования к БД, особенности ее функционирования и ограничения, которые накладываются на ее отдельные объекты Необходимо определить:
   - какие объекты важны для данного применения;
   - какие свойства могут иметь эти объекты;
   - какие связи существуют между объектами;
   - какие имена можно присвоить отдельным объектам и атрибутам.
2. Построение инфологической модели (логической), то есть модели БД, которая не привязана к конкретной СУБД (СУБД-независимая модель). Разрабатывается модель предметной области БД в терминах ER (Entity Relationship)-диаграммы (модель «сущность-связь»).
3. Построение даталогической (физической) модели. Это уже СУБД-ориентированная модель, жестко связанная с конкретной СУБД. Здесь появляются таблицы и реальные типы данных, ограничения на них.
4. Прямое моделирование конкретной БД, ее реализация.

При изображении ER-диаграммы используют различные нотации (обозначения). Будем использовать нотацию IE (Information Engineering). В этой нотации сущности изображаются прямоугольниками, в верхней части которых указывают первичные ключи, отделенные от неключевых атрибутов горизонтальной чертой.

Связи между сущностями бинарные: «один-к-одному», «один-ко-многим», «многие-к-одному», «многие-ко-многим». При изображении связей используются следующие графические элементы, указывающие на мощность связи:

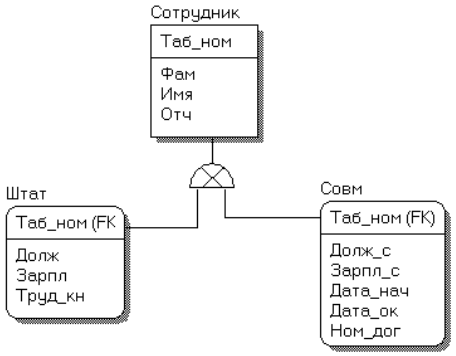
- – ноль;
- | – один;
- || – один и только один (обычно указывается со стороны родительской таблицы);
- $\prec$ – много (больше 0, элемент иногда называют «вороньей (куриной) лапкой»).

Связь, когда вторичный ключ мигрирует в список неключевых атрибутов второй сущности, называют неидентифицирующей связью и обозначают пунктиром. Дочерняя сущность в этом случае будет независимой и изображается прямоугольником.

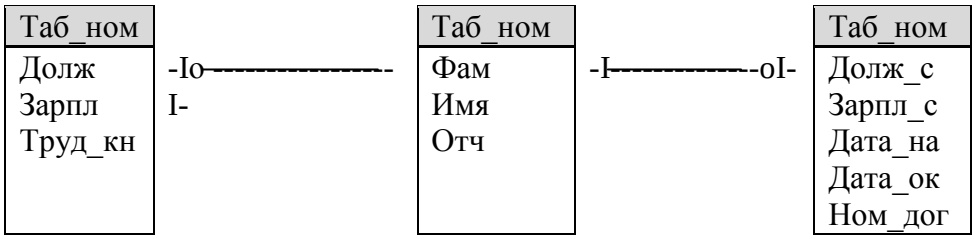
Связь, когда вторичный ключ попадает в состав атрибутов первичного ключа второй сущности, называют идентифицирующей и отображают сплошной линией.

Дочерняя сущность в этом случае будет зависимой сущностью, изображается прямоугольником со скругленными уголками.

Существуют разные способы оценки сущностей и связей между ними. Например, сущность Сотрудник в общем случае характеризует сотрудников предприятия, которые могут быть в его штате и/или работать по совместительству на условиях почасовой оплаты (всего 12 атрибутов – Таб_ном, Фам, Имя, Отч, Долж, Зарпл, Труд_кн, Долж_с, Зарпл_с, Дата_нач, Дата_ок, Ном_дог). В этом случае целесообразно выделить общие атрибуты в одну сущность Сотрудник (супертип), а остальные, различающиеся по своим характеристикам атрибуты, разбить на 2 сущности – Штат и Совм (подтипы). Этот вид связи получи название супертип-подтип.



На уровне даталогического проектирования этот вид связи преобразуется в связи один-к-одному.



В процессе проектирования базы данных используют опросов будущих пользователей, помогающие определить сущности, атрибуты и связи. А также значительное количество информации можно получить при исследовании бизнес-процессов, отчетов и документов, используемых на предприятии в его повседневной деятельности.

5.2 Системы разработки приложений баз данных. CASE-системы

CASE-средства (Computer-Aided System Engineering) представляют собой набор инструментов и методов программной инженерии для проектирования программного обеспечения.

Достоинством CASE-средств, ориентированных на проектирование БД, является поддержка инфологического (концептуального) проектирования. Но ни одна CASE-система не может гарантировать соответствия построенной концептуальной модели предметной области. Это определяется только квалификацией разработчиков, их пониманием предметной области и умением отобразить ее в модели.

Как правило, такие Case-системы содержат средства проверки моделей, помогающие устранить ошибки, связанные, в основном, с невнимательностью - отсутствие идентификатора у сущности, отсутствие связи объекта с другими объектами, неправильное задание имени др.

К CASE-системам, предназначенным для автоматизации этапов анализа и проектирования программного обеспечения, а также для генерации кодов на различных языках и выпуска проектной документации относятся Rational Rose, Oracle Designer, Power Designer, All Fusion Modeling Suite.

All Fusion Modeling Suite включает 5 основных компонентов:

– All Fusion Process Modeler (BPWin), позволяет моделировать бизнес-процессы на предприятии;

– All Fusion Erwin Date Modeler (ERWin), осуществляет концептуальное моделирование;

– All Fusion Date Model Validator (ERWin Examiner), позволяет проверить модель, построенную с помощью ERWin;

– All Fusion Date Model Manager (ModelMart – MM) – используется при коллективной разработке БД;

– All Fusion Component Modeler – позволяет автоматизировать процесс создания пользовательских интерфейсов.

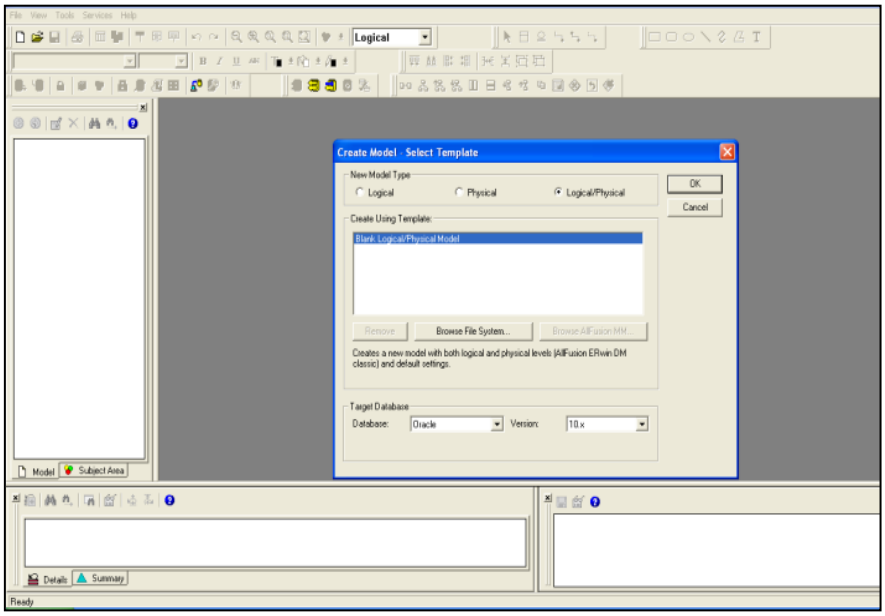
5.3 Моделирование структур данных средствами CASE-систем

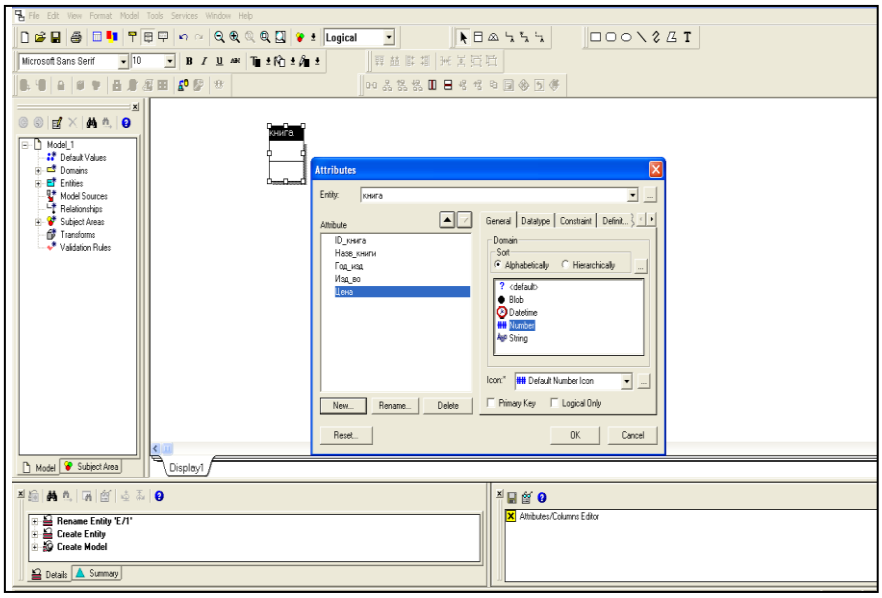
5.3.1 Моделирование в All Fusion Erwin Date Modeler

All Fusion Erwin Date Modeler позволяет строить логическую модель БД, которая не зависит от СУБД. Пользователь описывает структуру данных визуально. Задает служащие прообразами реляционных таблиц сущности с их атрибутами и при помощи мыши "натягивает" между ними связи, которые являются прототипами реляционных отношений.

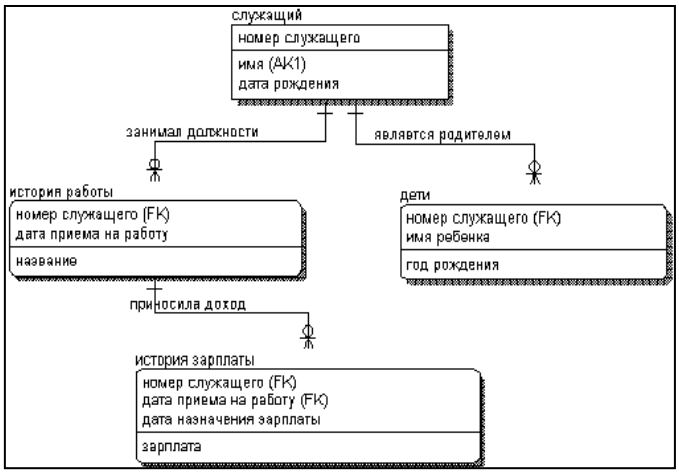
Существует 2 уровня представления и моделирования - логический и физический. Выбор между логическим и физическим уровнем отображения определяет линейку

инструментов и строку меню. Инструменты для создания модели - доступны как из меню, так и через окно инструментов:



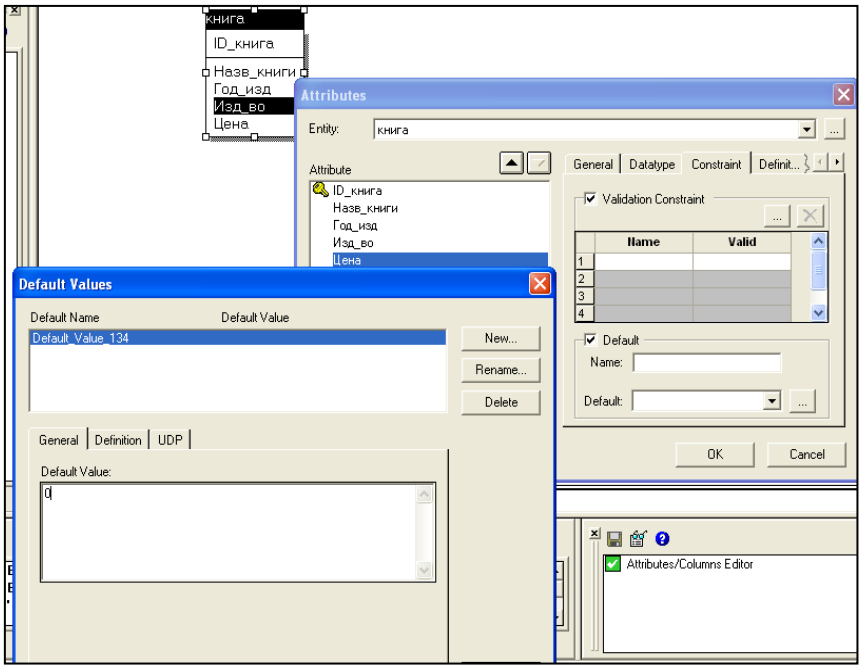


Используют различные режимы отображения модели: режим "сущности", режим "атрибуты", режим "первичные ключи", режим "показ глагольной фразы".

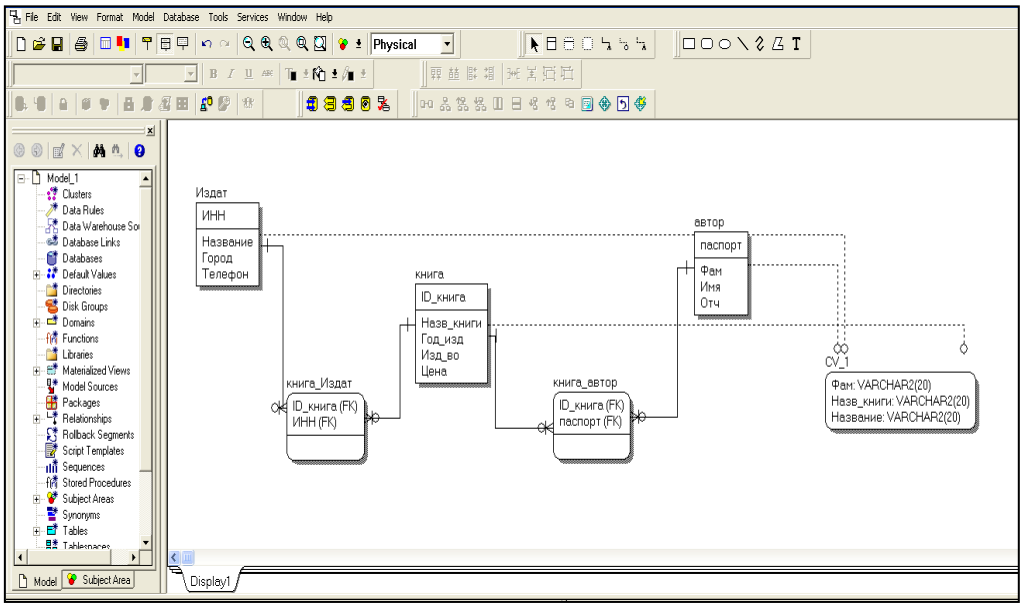


Логическая модель создается на уровне сущностей, здесь допускаются связи «многие-ко-многим» и супертип-подтип.

Задаются также значения по умолчанию и ограничения.



All Fusion Erwin Date Modeler позволяет по логической модели получать физическую (даталогическую) модель, а также построить представление (View, объект - представление на основе команды Select).



Обычно модели ERwin сохраняются на диск в виде файла.

Таким образом, All Fusion Erwin Date Modeler позволяет

– поддерживать работу более 20 СУБД (Oracle, SQL-server, Access, Progress, MySQL, Tecado, RedBreek);

– поддерживать проектирование БД в составе коллектива разработчиков (можно создавать проект в Erwin и сохранить его в специальной БД ModelMart);

– взаимодействовать с Validator (можно проверить созданную модель);

– передать модель в Oracle Designer;

– поддерживать режим прямого проектирования (Forward Engineering);

– поддерживать режим Reverse Engineering (возможность построения БД(ее структуры) на основе реальной информационной системы в любой СУБД);

– документировать модель, документация строится автоматически.

5.3 Ошибки моделирования. All Fusion Date Model Validator

All Fusion Date Model Validator позволяет осуществлять поиск ошибок проектирования и исправлять некоторые из них

Источником для поиска ошибок может служить физическая модель Erwin скрипт, БД (набор команд SQL), реальная БД, модель из ModelMart (если сохранили модель там).

Для визуального представления модели имеется несколько окон.

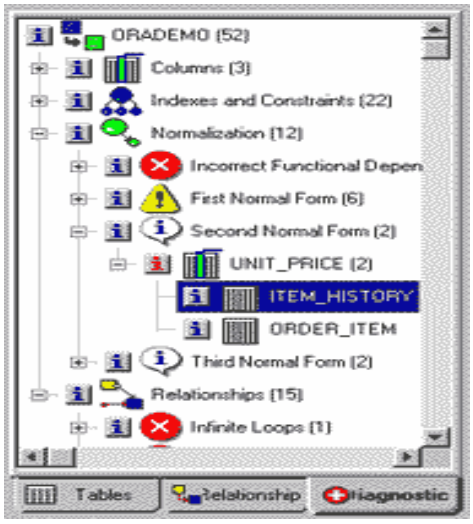
В левой части главного окна три закладки: Tables, Relationships и Diagnostics. Каждая из них содержит иерархических список, например, ракладка Tables содержит следующий иерархический список:

– первый уровень - список открытых моделей,

– второй — список таблиц,

– остальные — список колонок и первичных ключей.

Закладка Relationships содержит информацию о связях: Закладка Diagnostics – информацию об ошибках и недостатках модели данных



– первый уровень - список моделей,

– второй — список категорий ошибок.

Подробное описание ошибки можно получить щелчком левой кнопкой мыши по кнопке «i» слева от имени ошибки.

Ошибки объединены в четыре категории, которые определяет Validator:

– ошибки моделирования столбцов;

– ошибки моделирования индексов и ограничений;

– ошибки моделирования связей;

– ошибки нормализации.

Ошибки моделирования столбцов:

1) противоречия в определении типа данных (Inconsistent Definition), столбцы имеют одинаковые названия в разных таблицах, а типы данных у них разные, в результате могут сохраниться неверные данные;

2) таблица не имеет колонок (No Column in a table);

3) в модели таблицы с одинаковыми именами (Duplicate Table Names in Model), то есть имя таблицы не уникально;

4) достигнуто максимальное количество символов в имени таблицы (Name Length exceeds Maximum);

6) значение по умолчанию конфликтует с типом данных или с ограничением (Inconsistence Default Value), несогласованное с типом данных значение по умолчанию;

7) имена колонок совпадают с зарезервированными словами – Date, Name, Select …

Ограничения – это средства, контролирующие вносимую в БД информацию. К ограничениям можно отнести задание Primary Rey, Foreign Key, Unique, Not Null и Check.

Помимо ограничений существуют специальные структуры, позволяющие ускорить поиск данных в таблице - индексы. Oracle поддерживает более полутора десятков индексов разных типов, например, B-tree (индекс сбалансированного дерева), Bitmap – битовый индекс, индекс по функциям, Spatial индекс (для графической информации).

Ошибки моделирования ограничений и индексов:

1) таблица не имеет кластерного индекса (Tables with No Clustered Index). Вид индексов не поддерживает Oracle. Неправильно установлена БД, то есть начата проверка другой БД (не Oracle);

2) таблица не имеет ограничений, гарантирующих уникальность записи (Tables without a Candidate Key), не заданы Primary Key, Unique Constraint или Unique Index;

3) колонки с одинаковыми именами имеют в разных таблицах разные ограничения (Different Check Constrains);

4) колонки потенциальных и альтернативных ключей допускают значение Null (Candidate Keys with All Nullable Columns);

5) таблица содержит слишком много индексов (Table with Too Many Indexes), максимальное кол-во индексов – 6.

Ошибки моделирования связей:

1) Incorrect Recursive RelationShip (некорректно определенная рекурсивная связь, связь таблицы с самой собой);

2) бесконечные циклы (Infinite Loops).

Ошибки нормализации (Validator проверяет столбики с одинаковыми именами в одной или нескольких таблицах и определяет нарушение 1 НФ (First Normal Form), 2 НФ (Second Normal Form), 3 НФ (Fird Normal Form).