**СОДЕРЖАНИЕ**

Введение 2

1. Системный анализ предметной области 4

1.1. Краткая характеристика предметной области 4

1.2. Описание предметной области 13

2. Инфологическое моделирование 18

2.1.Модель «сущность-связь» 18

2.2. Связи между сущностями инфологической модели 20

Заключение 23

Список литературы 24

# Введение

Целью данной курсовой работы является построение и реализация базы данных защиты распределенной базы данных от несанкционированного доступа.

Основными задачами, поставленными в ходе работы, являлись:

* сбор, анализ и сортирование документов с целью описания предметной области;
* отбор необходимых документов для создания базы данных;
* выявление сущностей инфологической модели и моделирование связей между ними.

В настоящее время практически во всех сферах человеческой деятельности используются базы данных. В том числе решение перечисленных задач позволит достигнуть цели, поставленной в курсовой работе, а именно, реализовать базу данных защиты распределенной базы данных от несанкционированного доступа.

В общем смысле термин «база данных» (БД) можно применить к любой совокупности связанной информации, объединенной вместе по определенному признаку, т.е. к набору структурированных данных (организованных определенным образом). При этом большинство БД использует табличный способ представления, где данные располагаются по строкам (которые называются записями) и столбцам (которые называются полями), все записи должны состоять из одинаковых полей и все данные одного поля должны иметь один тип.

В настоящее время существует множество систем управления базами данных (СУБД) и других программ, выполняющих похожие функции, преобладающей является реляционная базаданных. В компьютерном варианте в реляционной БД информация хранится, как правило, в нескольких таблицах-файлах, связанных между собой посредством одного или нескольких совпадающих в этих таблицах полей (в некоторых компьютерных системах все таблицы одной базы помещаются в один файл). Каждая строка в таблице реляционной БД должна быть уникальна (т.е. не должно быть одинаковых строк-записей). Такие уникальные столбцы (или уникальные группы столбцов), используемые, чтобы идентифицировать каждую строку и хранить все строки отдельно, называются первичными ключами таблицы.

Для проектирования БД одной из концепционных является модель «сущность-связь». С помощью сущности моделируется класс однотипных объектов. Сущность имеет имя, уникальное в пределах моделируемой системы. Так как сущность соответствует некоторому классу однотипных объектов, то предполагается, что в системе существует множество экземпляров данной сущности. Объект, которому соответствует понятие сущности, имеет свой набор *атрибутов* – характеристик, определяющих свойства данного представителя класса. При этом набор атрибутов должен быть таким, чтобы можно было различать конкретные экземпляры сущности.

Отношение «один-ко-многим» можно назвать основным типом отношений, использующимся при проектировании современных БД. Поскольку оно позволяет представлять иерархические структуры данных.

Отношения один-ко-многим могут быть жесткими и нежесткими. Для жестких отношений должно выполнять требование, что каждой записи в родительской таблице должна соответствовать хотя бы одна запись в дочерней таблице.

Таким образом, выбор реляционной БД открывает широкие возможности для пользователя, позволяя легко создать БД, удовлетворяющую требованиям организаций самого разного пользовательского профиля, выполняющих разные по значимости задачи и использующих неравнозначные объемы информации в своей деятельности.

# 1. Системный анализ предметной области

## 

## 1.1. Краткая характеристика предметной области

В зависимости о размещения типовых компонентов приложения по узлам сети информационные системы и соответствующие приложения могут строиться различными способами, такими как системы на основе локальной сети персональных компьютеров (файл-серверные приложения), системы с архитектурой клиент-сервер и др.

**Суть модели файлового сервера** состоит в том, что один из компьютеров в сети считается файловым сервером и предоставляет услуги по обработке файлов другим компьютерам. Файл-сервер работает под управлением сетевой операционной системы и играет роль компонента доступа к информационным ресурсам. На других компьютерах в сети функционирует приложение, в котором функции представления информации и логика прикладной обработки совмещены. Обращение за сервисом управления данными происходит через среду передачи с помощью операторов языка SQL или вызовом функций библиотеки API (Application Programming Interface – интерфейс прикладного программирования). Основное достоинство такой модели состоит в большом обилии готовых СУБД, имеющих SQL-интерфейсы, и существующих инструментальных сердств, обеспечивающих быстрое создание программ клиентской части. Средства разработки чаще всего поддерживают графический интерфейс пользователя в MS Windows, стандарт интерфейса ODBC и средства автоматической генерации кода.

**Недостатки модели файл-сервер**:

высокий сетевой трафик (вследствие того, что вся логика сосредоточена в приложении, а обрабатываемые данные расположены на удаленном узле;



во время работы приложений обычно по сети передаются целые БД);



узкий спектр операций манипуляции с данными;



отсутствие надежных средств безопасности доступа к данным (защита только на уровне файловой системы).



Поэтому предпочтительно применять **технологию клиент-сервер**, когда сервер базы данных используется не только для хранения информации, но и для обработки запросов к базе данных. Запросы рабочей станции обрабатываются сервером базы данных и обратно возвращается только результат выполнения запроса. Такой подход уменьшает поток данных в сети. Кроме того, обработка запросов сервером базы данных осуществляется быстрее, чем на рабочей станции, так как:

в качестве сервера базы данных используется гораздо более мощный компьютер



СУБД, используемая в качестве сервера базы данных, обладает более совершенными средствами обработки данных



Так как обработка запросов осуществляется на сервере базы данных, а не на рабочей станции, рабочая станция называется клиентом сервера базы данных. При работе в режиме клиент-сервер серверная часть системы управления базами данных устанавливается на файл-сервере, а клиентская часть — на рабочей станции. В ряде случаев клиентская и серверная части являются отдельными компонентами одной СУБД (например, Oracle или SQLbase). В других случаях в качестве клиентской части используются настольные СУБД или специальные системы разработки приложений клиент/сервер (например, PowerBuilderили SQLWindows), а в качестве сервера базы данных — мощная СУБД типа Oracle или SQL Server.

Основная задача, которая должна быть надежно решена при разработке многопользовательского приложения – это управление возможными столкновениями пользователей при одновременной модификации одних и тех же данных. Эта проблема должна быть решена на двух уровнях:

**первый** – это сведение количества таких конфликтов к минимуму и

**второй** – разработка четкого алгоритма их разрешения.

Таблица 1

|  |  |
| --- | --- |
| Низкая конкуренция (случай 1) | Высокая конкуренция (случай 2) |
| Блокировка записи | Запись данных в переменные |
| Чтение записи | Чтение переменных |
| Полноэкранное редактирование | |
| Запрос на сохранение данных | |
| Сохранение данных | Блокировка записи |
| -- | Запись данных из переменных в БД |
| Снятие блокировки с записи | |

В таблице 1 приведены два возможных алгоритма модификации данных в многопользовательских системах с низким и высоким уровнями конкуренции пользователей при попытке доступа к одним и тем же данным. В первом случае мы получаем несомненно более стабильную систему за счет захвата нужной записи и тем самым полного устранения возможности одновременной модификации одних и тех же данных. В то же время захваченные данные могут оставаться недоступными долгое время в случае, если пользователь вдруг заметил какую-то ошибку и надолго застрял в поисках путей ее исправления или просто решил отдохнуть в момент подноэкранного редактирования. Для избежания подобных ситуаций можно поставить ограничение на время редактирования, но тогда данные могут вдруг изчезнуть с экрана прямо на глазах изумленного зазевавшегося пользователя.

При втором варианте продожительность блокировки не зависит от поведения пользователя. Это время будет определяться только продолжительностью записи данных. Во время редактирования данные продолжают оставаться доступными для других пользователей. Хорошо ли это? Отлично, мы достигли потрясающей гибкости! Но в то же время получили массу забот, так как во время редактирования данных одним пользователем их может изменить и другой. При таком подходе есть риск, что после того, как первый пользователь успешно модифицирует запись, ее тут же освежит и второй работник, который в глаза не видел обновленных первым абонентом сети данных. Ведь у него на экране были данные из БД до их изменения первым пользователем.

Для решения этих проблем в СУБД предлагается использовать **буферизацию данных**. Рассмотрим типичный набор блокировок:

отсутствие буферизации.



пессимистическая буферизация записи;



оптимистическая буферизация записи;



пессимистическая буферизация таблицы;



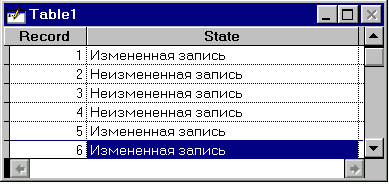
оптимистическая буферизация таблицы.



Буферизация на уровне записи означает, что перед началом редактирования содержимое текущей записи будет сохранено во внутреннем буфере СУБД. Буферизация нв уровне таблицы сохраняет в буфере содержание всех отобранных для редактирования записей. Оптимистическая буферизация обеспечивает блокировку записей только на время сохранения содержимого буфера в файле, что соответствует приведенному выше случаю 2. Пессимистическая буферизация работает как в случае 1, то есть блокирует запись перед копированием ее содержимого в буфер.

Использование буферизации позволяет автоматизировать процесс переноса данных из полей БД в переменные и обратно. При этом группа функций позволяет получить исчерпывающую информацию о состоянии буферизованной таблицы, что дает возможность организовать очень эффективный алгоритм разрешения возможных конфликтов при изменении данных (схема их взаимодействия приведена на рисунке 1).

**Рис.1 Функции для работы с буферизованной таблицей**



|  |
| --- |
| ТАБЛИЦА |

|  |
| --- |
| Редактирование |

|  |
| --- |
| Сравнение данных |

|  |
| --- |
| Обновление данных |

|  |
| --- |
| Отмена изменений |

|  |
| --- |
| **OLDVAL()** |

|  |
| --- |
| **CURVAL()** |

|  |
| --- |
| **TABLEUPDATE()** |

|  |
| --- |
| **TABLEREVERT()** |

**OLDVAL()** – возвращает первоначальное значение поля, которое было модифицировано, но не обновлялось.

**CURVAL()** – возвращает значение поля непосредственно с диска или из удаженного источника.

**TABLEUPDATE()** – фиксирует изменения, внесенные в буферизованную запись либо в буферизованную таблицу или курсор.

**TABLEREVERT()** – сбрасывает изменения, внесенные в буферизованную запись либо в буферизованную таблицу или курсор и восстанавливает содержимое по данным OLDVAL().

При буферизации таблицы мы имеем возможность добавлять и удалять записи в буфере. При добавлении новая запись помещается в конец буфера и получает номер с отрицательным значением. Доступ к этой записи может быть выполнен с помощью функции RECNO() с отрицательным параметром, например –1.

**RECNO()** - возвращает номер текущей записи в текущей или заданной таблице

При написании многопользовательских приложений необходимо учитывать, что целый ряд команд при их использовании выполняет автоматическую блокировку.

Если вы решили управлять блокировкой вручную, то придерживайтесь следующего алгоритма:

проверьте состояние блокировки записи или таблицы;



если блокировки нет, то требуемые ресурсы можно заблокировать;



если ресурсы блокированы, попробуйте еще раз, но при этом следует избегать слишком частых попыток. Кроме чрезмерной загрузки сети это ни к чему не приведет.



При разработке алгоритма блокировки старайтесь придерживаться четкой последовательности событий. Например, в следующем примере при борьбе за несколько ресурсов будет наблюдаться патовая ситуация (тупик):

|  |  |
| --- | --- |
| **Пользователь 1** | **Пользователь 2** |
| Попытка блокировать запись 1 | Попытка блокировать запись 4 |
| Попытка блокировать запись 4 | Попытка блокировать запись 1 |
| Если попытка неудачна, ждлем освобождения ресурсов | Если попытка неудачна, ждлем освобождения ресурсов |
| Ждем, ждем, ждем … | Ждем, ждем, ждем … |

Существуют два простых правила, помогающих избежать данной ситуации. Во-первых, необходимо выполнять все действия в одинаковой последовательности, т.е. 2-й пользователь должен пытаться блокировать записи, также начиная с первой. Во-вторых, всегда необходимо ставить ограничения по времени на попытки блокировки, чтобы дать возможность хотя бы одному пользователю закончить работу.

Ряд команд и функций автоматически обеспечивают снятие блокировки:

Закрытие таблицы



Завершение транзакции



Завершение сеанса работы с СУБД



Фиксация изменений, внесенных в буферизованную запись или таблицу



Если вы хотите обеспечить защиту изменяемых данных и возможность восстановления первоначальных значений на протяжении определенного периода испольнения программы, используйте механизм встроенных транзакций. При использовании транзакций с момента выдачи команды «Начать транзакцию» все изменения сначала сохраняются в памяти компьютера или на диске и только при завершении транзакции переносятся в таблицу. При этом таблица обязательно должна быть включена в базу данных. Если в процессе работы выяснилась нецелесообразность использования сделанных изменений, до выполнения команды «Завершить транзакцию» всегда остается возможность вернуться к первоначальному состоянию таблицы, выдав команду «Отменить изменения, внесенные в ходе текущей транзакции». Для организации логических групп по обновлению данных можно использовать вложенные транзакции.

В общем случае при использовании транзакций предпочтительной является буферизация записи по сравнению с буферизацией таблицы.

Использование транзакций не может гарантировать сохранение измененных данных, например, при выключении компьютера. В этом случае автоматически выполнится откат к состоянию таблицы до начала транзакции.

Интерфейс прикладного программирования ODBC API предоставляет общие методы доступа SQL как к реляционным, так и к нереляционным (ISAM) источникам данных.

В ANSI SQL входит интерфейс на уровне вызовов (CLI - call-level interface), который используется ODBC для обеспечения доступа и работы с данными во многих системах управления базами данных. Интерфейс CLI соответствует требованиям, установленным в 1991 году группой SQL Access Group, которые определяют общий синтаксис SQL и интерфейса API. Иметь общий метод доступа к источникам данных удобно потому, что тогда база данных на сервере становится прозрачной для приложений, которые написаны в соответствии с некоторым заданным уровнем совместимости ODBC.

Интерфейс ODBC API реализован как набор расслоенных DLL-функций для Windows. Динамическая библиотека ODBC.DLL - это основная библиотека управления драйверами ODBC, которая вызывает специализированные драйверы для разных поддерживаемых системой баз данных. Каждый драйвер совместим со своим уровнем CLI и относится к одной из двух категорий: одноуровневые или многоуровневые драйверы.

Одноуровневые драйверы предназначены для использования при работе с теми источниками данных, которые не могут быть обработаны ANSI SQL. Обычно это локальные базы данных на персональных компьютерах, такие как dBase, Paradox, FoxPro и др. Драйверы, соответствующие этим базам данных, переводят грамматику ANSI SQL в инструкции низкого уровня, которые непосредственно обрабатывают составляющие базу данных файлы.

Многоуровневые драйверы используют сервер СУРБД для обработки SQL-предложений и предназначены для работы в среде клиент-сервер. Помимо обработки ANSI SQL, они также могут поддерживать и собственную грамматику конкретной СУРБД, поскольку ODBC может без трансляции передавать SQL-предложения источникам данных (механизм "passthrough").

Драйверы ODBC для баз данных типа клиент-сервер реализованы для Oracle, Informix, Microsoft и Sybase SQL Server, Rdb, DB2, Ingres, HP/Image и Any SQL.

Существует 4 важных этапа (шага) процедуры запроса данных через API.

**Шаг 1** - установление соединения. Первый шаг состоит в размещении указателей (handle) среды ODBC, которые выделяют оперативную память под ODBC драйверы и библиотеки. Затем происходит выделение памяти для указателей соединения, и соединение устанавливается.

**Шаг 2** - выполнение предложения SQL. Выделяется указатель предложения, локальные переменные связываются со столбцами в SQL-выражении (это необязате~ьное действие), и выражение представляется на разбор главному ODBC драйверу для обработки.

**Шаг 3** - извлечение данных. Перед извлечением данных возвращается информация о результирующем наборе, такая как число столбцов в наборе. Исходя из этого числа, результирующий набор помещается в буфер записей, выполняется цикл по нему и извлекается по одному столбцу в локальные переменные. Этот шаг необязателен, если используется связывание столбцов.

**Шаг 4** - освобождение ресурсов. После того, как данные получены, освобождаются ресурсы вызовом функций освобождения указателей предложения, соединения и среды. Указатели предложения и соединения могут быть использованы в процессе обработки.

Технология ODBC разрабатывалась как общий, независимый от источников данных, способ доступа к данным. Также ее применение должно было обеспечить переносимость приложений на различные базы данных без переработки самих приложений. В этом смысле технология ODBC уже стала промышленным стандартом, ее поддерживают практически все производители СУБД и средств разработки.

Однако универсальность стоит дорого. Если при разработке приложений одним из основных критериев является переносимость на различные СУБД, то использование ODBC является оправданным. Для увеличения производительности и эффективности приложения активно применяют специфические для данной СУБД расширения языка SQL, используют хранимые на сервере процедуры и функции. В этом случае теряется роль ODBC как общего метода доступа к данным. Тем более, что для разных СУБД драйверы ODBC поддерживают разные уровни совместимости. Поэтому многие производители средств разработки помимо поддержки ODBC поставляют "прямые" драйверы к основным СУБД.

***1.2. Описание предметной области***

В управленческой, экономической, финансовой, правовой сферах широко используется информация, представляющая собой неструктурированную информацию (помимо структурированной информации, организованной в БД, находящихся под управлением СУБД). Информационные ресурсы представляют собой отдельные документы и отдельные массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, других видах информационных систем). К ним относятся рукописные, печатные и электронные издания, содержащие нормативную, распорядительную, фактографическую, справочную, аналитическую и др. информацию по различным направлениям общественной деятельности (законодательство, политика, демография, социальная сфера, наука, техника, технология и т.д.).

Для однопользовательских АС характерно использование следующих баз данных:

локальные реляционные базы данных, находящиеся под управлением одной или нескольких СУБД (Microsoft Access, FoxPro и т.п.) и предназначенные для решения пользователем прикладных задач с использованием собственного или покупного специального программного обеспечения на его АРМе;

локальные базы неструктурированной информации (текстовых и табличных документов, созданных пользователем средствами Microsoft Word и Microsoft Excel, полученных по электронной почте,на машинных носителях, а также документов, полученных в результате решения пользователем прикладных задач с использованием информации реляционных баз данных), организованные и хранящиеся в виде каталогов и подкаталогов на его АРМе;

базы данных, размещенные на удаленных ПК в федеральных и международных сетях, к которым организован доступ самим пользователем со своего АРМ (если АРМ подключен к федеральным и международным сетям передачи данных).

Современные автоматизированные информационные системы представляют собой, как правило, ЛВС, подключенные к федеральным и международным сетям передачи данных. Пользователь ЛВС использует не только вышеперечисленные локальные базы данных, но и распределенные:

реляционные базы данных на сервере ЛВС, находящиеся под управлением одной или нескольких СУБД;

базы неструктурированной информации (документов, созданных и полученных разными пользователями ЛВС), организованные и хранящиеся в виде каталогов и подкаталогов на сервере ЛВС;

базы данных различных приобретенных АС, установленные в ЛВС и доступные всем пользователям сети;

базы данных, размещенные на удаленных ПК в федеральных и международных сетях, к которым организован доступ для всех пользователей ЛВС.

Значительная часть неструктурированной информации в вышеназванных базах является, как правило, гипертекстовыми и гипермедиа-документами, объединенными с помощью гиперссылок в гипертекстовые базы данных.

В последние годы находят все более широкое применение так называемые геоинформационные системы. Геоинформационные системы (ГИС)–это интегрированные в единой информационной среде электронные пространственно-ориентированные изображения (карты, схемы, планы и т.п.) и базы данных (БД). В качестве БД могут использоваться таблицы, паспорта, иллюстрации, расписания и т. п. Такая интеграция значительно расширяет возможности системы и позволяет упростить аналитические работы с координатно-привязанной информацией. Принципиальным отличием ГИС является наличие в них картографических данных местности, региона и т.д., к которым привязывается остальная информация системы. Геоинформационные системы уже широко используются в управлении градостроительством, транспортом, природными ресурсами и т.п.

Для современного этапа развития информационных технологий характерно наличие разнообразных инструментальных средств и покупного специального программного обеспечения, которыми может овладеть любой пользователь, а такженаличие большого количества промышленно функционирующих БД коммерческих организаций, органов государственной власти и местного самоуправления, предприятий и организаций.

Такая ситуация позволяет при создании многих АС отказаться от проектирования и разработки собственных реляционных баз данных и собственного специального программного обеспечения. Использование современных инструментальных средств позволяет пользователю самостоятельно (без помощи системного программиста) организовывать со своего АРМ доступ к различным информационным ресурсам, например, создавать каталоги нормативно-правовых актов, каталоги адресов WWW-серверов Интернета и т.п. Появление ОПО последних версий позволяет пользователю организовывать доступ к различным ресурсам АРМ и ЛВС через гиперссылки (по принципу “паутины”) взамен иерархического принципа доступа (принципа “дерева”).

Распределенная система организации баз данных предполагает наличие соответствующей технологии доступа пользователей к информационным ресурсам, ориентированной, прежде всего, на вычислительные модели типа "клиент-сервер".

Технология "клиент-сервер" предполагает разделение функций обработки данных на три группы: функции ввода/вывода и отображения данных; прикладные функции, характерные для данной предметной области; функции хранения и управления данными. Каждая группа функций выполняется отдельным логическим компонентом.

Различия в реализации приложений в рамках "клиент-сервер" определяются механизмом использования и распределения между компьютерами в сети этих компонент, в соответствии с этим выделяют три подхода, реализованные в моделях:

модель доступа к удаленным данным (Remote Data Access-RDA), в которой компонент представления и прикладной компонент совмещены и выполняются на одном компьютере. Запросы к информационным ресурсам направляются по сети к удаленному компьютеру, который обрабатывает запросы и возвращает блоки данных. Эта модель является самой простой и традиционно используется в локальных вычислительных сетях, где скорость обмена достаточно высока, однако она неприемлема при работе в среде низкоскоростных каналов передачи данных. Поскольку вся логика локализована на одном компьютере, то приложение нуждается в передаче по сети большого, часто избыточного объема данных, что существенно повышает загрузку информационной системы в целом и может привести к длительному блокированию данных от других пользователей;

модель сервера базы данных (DataBase Server-DBS), которая строится в предположении, что процесс, выполняемый на компьютере-клиенте, ограничивается функциями представления, в то время как собственно прикладные функции реализованы в хранимых непосредственно в базе данных процедурах, выполняющихся на компьютере-сервере БД. Преимущества DBS-модели перед RDA заключаются в очевидном снижении сетевого трафика. Однако DBS-модель не обеспечивает требуемой эффективности использования вычислительных ресурсов в случае нескольких серверов;

модель сервера приложений (Application Server-AS), в которой процесс, выполняющийся в компьютере-клиенте, реализует функции первой группы. Прикладные функции выполняются на удаленном компьютере. Доступ к информационным ресурсам, необходимым для решения прикладных задач, обеспечивается тем же способом, что и в RDA модели. AS-модель не требует обеспечения миграции прикладных функций между серверами, что значительно облегчает администрирование системы в целом, однако, для обеспечения достаточной скорости обработки данных сервер приложений и сервер БД должны находится в одной ЛВС или быть соединены по выделенному каналу.

На практике часто для создания более гибких и динамичных систем используются смешанные модели.

Компьютер-клиент и компьютер-сервер могут работать в условиях ЛВС и быть абонентами глобальной компьютерной сети, общаясь между собой по организуемому виртуальному каналу или, используя для этого (при снижении требований на реактивность системы) электронную почту.

В настоящее время существует целый ряд программных средств, как системных, так и прикладных, реализующих описанные выше модели. Стоит отметить такие пакеты, как Oraclе SQL Server и Sybase SQL Server для платформы NetWare, продукт Microsoft Windows NTSQL Server, Oracle для среды Unix, Lotus Notes. Все эти программные средства работают на различных платформах (на машинах с процессорами Intel, на RISC-серверах и станциях производства HP, DEC и т.д.), в различных операционных средах. СУБД Oracle выделяется среди прочих исключительным быстродействием, мощными сетевыми средствами и средствами межплатформенной связи. Развитые средства электронной почты пакета Oracle позволяют организовать безбумажный документооборот, совместную подготовку и обработку документов. Существует интегрированный программный продукт ORACLE 2000WG, объединяющий достоинства популярной сетевой операционной системы Novell NetWare и СУБД Oracle. В структурах управления федеральных, государственных и местных органов власти все шире применяется пакет Lotus Notes.

# 2. Инфологическое моделирование

## 

## 2.1.Модель «сущность-связь»

Инфологическая модель отображает реальный мир в некоторые понятные человеку концепции, полностью независимые от параметров среды хранения данных. Существует множество подходов к построению таких моделей: графовые модели, семантические сети, модель "сущность-связь" и т.д. Наиболее популярной из них оказалась модель "сущность-связь" или называемая ещё ER-моделью (от англ. Entity-Relationship, т.е. сущность-связь).

Инфологическая модель применяется после словесного описания предметной области.

Проведем анализ предметной области проектируемой БД.

|  |
| --- |
| **Пользователи** |
| ***Код пользователя*** |
| Логин |
| Пароль |
| Примечание |

|  |
| --- |
| **Права пользователя** |
| ***Код доступа*** |
| Права |

|  |
| --- |
| **Сеанс** |
| ***Код сеанса*** |
| Код пользователя |
| Код доступа |
| Номер сеанса |
| Время начала |
| Время окончания |

Как любая модель, модель «сущность-связь» имеет несколько базовых понятий, которые образуют исходные кирпичики, из которых строятся уже более сложные объекты по заранее определенным правилам.

Эта модель в наибольшей степени согласуется с концепцией объектно-ориентированного проектирования, которая в настоящий момент, несомненно, является базовой для разработки сложных программных систем, поэтому многие понятия вам могут показаться знакомыми, и если это действительно так, то тем проще вам будет освоить технологию проектирования баз данных, основанную на ER-модели.

*Сущность*, с помощью которой моделируется класс однотипных объектов. Сущность имеет имя, уникальное в пределах моделируемой системы. Так как сущность соответствует некоторому классу однотипных объектов, то предполагается, что в системе существует множество экземпляров данной сущности. Объект, которому соответствует понятие сущности, имеет свой набор *атрибутов* – характеристик, определяющих свойства данного представителя класса. При этом набор атрибутов должен быть таким, чтобы можно было различать конкретные экземпляры сущности.

Рассмотрим сущности БД на примере исследуемой предметной области.

**Пользователи**

**Код пользователя**

Логин

Пароль

Примечание

**Код сеанса**

Код пользователя

Код доступа

Сеанс

Номер сеанса

Время начала

Время окончания

Доступ

Домашний телефон

Адрес

Год рождения Отчество

Права пользователя

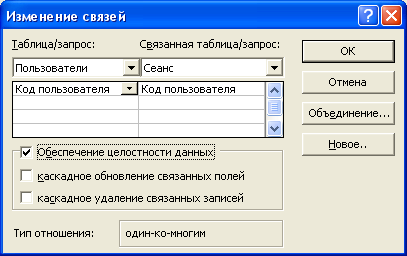
**Код доступа**

## 2.2. Связи между сущностями инфологической модели

Между сущностями могут быть установлены *связи* – бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют между собой. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой (*рекурсивная связь*). Она показывает, как связаны экземпляры сущностей между собой. Если связь устанавливается между двумя сущностями, то она определяет взаимосвязь между экземплярами одной и другой сущности.

Определим связи между выявленными сущностями.

Связь ОДИН-КО-МНОГИМ (1:М): одному представителю сущности А соответствуют 0, 1 или несколько представителей сущности В.



В разных нотациях мощность связи изображается по-разному. Между двумя сущностями может быть задано сколько угодно связей с разными смысловыми нагрузками. Связь любого из этих типов может быть *обязательной*, если в данной связи должен участвовать каждый экземпляр сущности, *необязательной* – если не каждый экземпляр сущности должен участвовать в данной связи. При этом связь может быть *обязательной с одной стороны* и *необязательной с другой стороны.* Обязательность связи тоже по-разному обозначается в разных нотациях. Мы снова используем нотацию POWER DESIGNER. Здесь необязательность связи обозначается пустым кружочком на конце связи, а обязательность перпендикулярной линией, перечеркивающей связь. И эта нотация имеет простую интерпретацию. Кружочек означает, что ни один экземпляр не может участвовать в этой связи. А перпендикуляр интерпретируется как то, что, по крайней мере, один экземпляр сущности участвует в этой связи.

Сущность имеет имя, уникальное в пределах модели. При этом имя сущности – это имя типа, а не конкретного экземпляра.

Сущности подразделяются на сильные и слабые. Сущность является слабой, если ее существование зависит от другой сущности – сильной по отношению к ней.

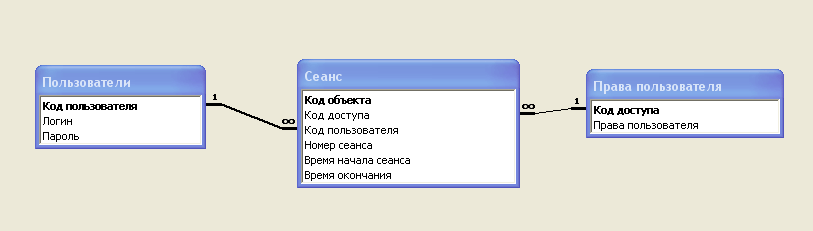
Сущность может быть расщеплена на два или более взаимоисключающих *подтипов,* каждый из которых включает общие атрибуты и/или связи. Эти общие атрибуты и/или связи явно определяются один раз на более высоком уровне. В подтипах могут определяться собственные атрибуты и/или связи. В принципе выделение подтипов может продолжаться на более низких уровнях, но в большинстве случаев оказывается достаточно двух-трех уровней.

Связи делятся на три типа по множественности: один-ко-одному (1:1), один-ко-многим (1:М), многие-ко-многим (М:М).

Связь один-ко-одному означает, что экземпляр одной сущности связан только с одним экземпляром другой сущности.

Связь один-ко-многим (1:М) означает, что один экземпляр сущности, расположенный слева по связи, может быть связан с несколькими экземплярами сущности, расположенными справа по связи.

Связь «многие-ко-многим (М:М) означает, что несколько экземпляров первой сущности могут быть связаны с несколькими экземплярами второй сущности, и наоборот. Между двумя сущностями может быть задано сколько угодно связей с разными смысловыми нагрузками.



# Заключение

Процесс проектирования БД на основе принципов нормализации представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели.

Инфологическая модель применяется на втором этапе проектирования БД, то есть после словесного описания предметной области. Процесс проектирования длительный и требует обсуждений с заказчиком и со специалистами в предметной области. Наконец, при разработке серьезных корпоративных информационных систем проект базы данных является тем фундаментом, на котором строится вся система в целом, и вопрос о возможном кредитовании часто решается экспертами банка на основании именно грамотно сделанного инфологического проекта БД. Следовательно, инфологическая модель должна включать такое формализованное описание предметной области, которое легко будет «читаться» не только специалистами по базам данных. И это описание должно быть настолько емким, чтобы можно было оценить глубину и корректность проработки проекта БД, и конечно, оно не должно быть привязано к конкретной СУБД. Выбор СУБД – это отдельная задача, для корректного ее решения необходимо иметь проект, который не привязан ни к какой конкретной СУБД.

Инфологическое проектирование прежде всего связано с попыткой представления семантики предметной области в модели БД. Реляционная модель данных в силу своей простоты и лаконичности не позволяет отобразить семантику, то есть смысл предметной области.

# Список литературы

1. Бекаревич Ю.Б., Пушкина Н.В. Самоучитель Microsoft Access 2002. – СПб.: БХВ-СПб., 2003. – 720 с.
2. Виноградова И.А., Грибова Е.А., Зубков В.Г. Практикум на ЭВМ. MS Access: Учебное пособие для студентов заочной (дистанционной) формы обучения. – М.: ГИНФО, 2000. – 124 с.
3. Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: Учебное пособие. – М.: ФОРУМ: ИНФРА-М, 2003. – 352 с.
4. Информатика. Базовый курс. /Под ред. С.В.Симоновича. – СПб.: Питер, 1999. – 640 с.
5. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2002. – 304 с.
6. Петров В.Н. Информационные системы. – СПб.: Питер, 2003. – 688 с.
7. Тихомиров Ю.В. MS SQL Server 2000: разработка приложений. – СПб.: БХВ-Петербург, 2000. – 368 с.