1 курс

ПЛАН – КОНСПЕКТ
проведения лекционных, практических занятий № 15–16 по дисциплине «Информатика»

Раздел 3. «Моделирование формализация»

Тема 3.2:
«Информационные модели и их моделирование в СУБД.»

Подготовил: преподаватель
В.Н. Борисов

Рязань 2023

Вопросы занятий:

1. Понятие информационной модели. Структурные информационные модели.
2. Введение в базы данных (БД) и системы управления базами данных (СУБД). Основные понятия и определения.
3. Принципы построения баз данных и управления ими.
4. Основные этапы проектирования баз данных (проектирование и создание базы данных (практическое занятие № 15, теоретическая часть).
5. Основные элементы базы данных. Составные части инфологической модели. Реляционная модель данных.
6. Классификация баз данных. Базы данных (табличные, иерархические, сетевые).
7. Системы управления базами данных. Характеристики СУБД.
8. Работа с базами данных. Режим работы базы данных.
9. Создание и редактирование таблиц.
10. Запросы. Виды запросов. Запросы на выборку к единственной таблице. Создание запросов на выборку (Организация работы с данными в БД. Формирование запросов. (практическое занятие № 16, теоретическая часть)).
11. Знакомство с программой ЭТРАН – автоматизированной системой подготовки и оформления перевозочных документов (практическое занятие № 16, теоретическая часть).
12. Определение результатов выполнения запросов с применением аппарата алгебры логики.

Время проведения лекционного, практических занятий – 6 часов.

Первый вопрос: Понятие информационной модели. Структурные информационные модели.

Понятие информационной модели.

Компьютерное моделирование, включает в себя процесс реализации информационной модели при помощи компьютерных средств. Информационная модель представляет собой целый перечень информации о каком-либо объекте.

Что данная модель описывает, и какую полезную информацию несет: свойства моделируемого объекта; его состояние; связи с окружающим миром; отношения с внешними объектами. Что может служить информационной моделью: словесное описание; текст; рисунок; таблица; схема; чертеж; формула и так далее.

Отличительная особенность информационной модели заключается в том, что ее нельзя потрогать, попробовать на вкус и так далее. Она не несет материального воплощения, так как представлена в виде информации.

Структурные информационные модели.

Структурные информационные модели определяют построение таких важных средств, как *базы данных* (структурированные хранилища информации) и, соответственно, системы управления базами данных (СУБД).

В тех случаях, когда необходимо переработать большой объем информации, ее нужно структурировать, т.е. выделить в ней элементарные составляющие и их взаимосвязи.

Информационная структура представляет собой упорядоченную систему данных. Наиболее простыми информационными структурами являются списки, таблицы, схемы, графы.

Пример табличного структурирования информации — школьное расписание уроков.

Основными структурными моделями являются иерархическая, сетевая и табличная.

Иерархическая модель представляется в виде дерева, где отдельные элементы объекта являются узлами, а стрелочки показывают связи между этими элементами (рис. 1).

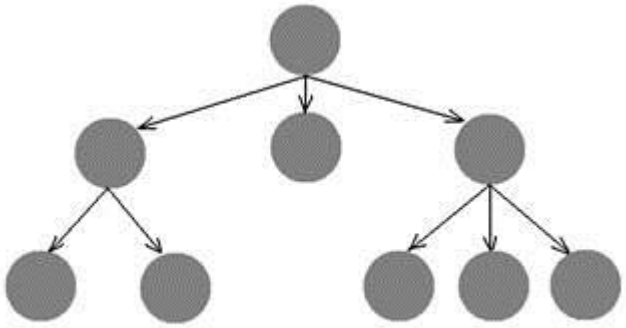


Рис. 1. Структура иерархической модели данных

Такая модель обладает следующими свойствами:

1. Иерархия начинается с верхнего узла. Каждый узел имеет характеристики (атрибуты), которые описывают моделируемый объект в данном узле.

2. Чем ниже уровень, тем выше зависимость» узла.

3. Каждый узел имеет только одну связь с более высоким уровнем. Каждый узел может иметь несколько связей с «зависимыми» (более низкими) уровнями.

4. Доступ к любому элементу структуры осуществляется только черед верхний узел по принципу «сверху-вниз».

5. Количество узлов не имеет ограничений.

Например, в биологии весь животный мир рассматривается как иерархическая система (тип, класс, отряд, семейство, род, вид). В информатике используется иерархическая файловая система.

В сетевой модели каждый узел может иметь любое количество связей с другими узлами без соблюдения какой бы то ни было иерархии (рис. 2).

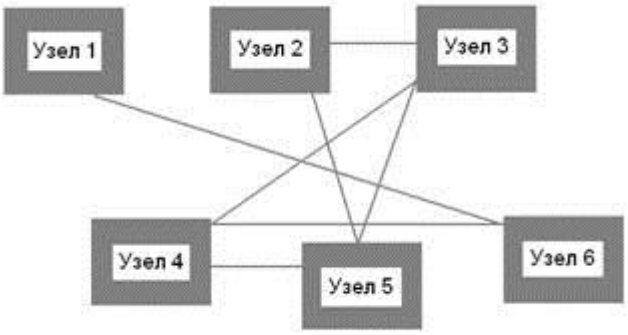


Рис. 2. Структура сетевой модели данных

Сетевые информационные модели применяются для отражения таких систем, в которых связь между элементами имеет сложную структуру. Например, различные части сети Интернет (американская, европейская, российская и т.д.) связаны между собой высокоскоростными линиями связи. При этом какие-то части (американская) имеют прямые связи со всеми региональными частями, в то время как другие могут обмениваться информацией между собой только через американскую часть (например, российская и японская).

В ***табличной модели*** каждой объект моделируемой системы описывается в виде таблицы с набором атрибутов. *Атрибуты*, или поля, — это построчные ячейки таблицы. Взаимосвязь между таблицами описывается отношениям между полями.

Таблица 1.

Код производителя	Название производителя	Адрес
...	...	...

Таблица 2.

Код производителя	Код товара	Цена товара за единицу
...	...	...

Таблица 3.

Код товара	Наименование товара
...	...

Взаимосвязь между полями разных таблиц может иметь *три вида*:

1. «Один к одному». Одному элементу первого объекта соответствует только один элемент второго объекта. Например, конкретному человеку может соответствовать не более одного номера паспорта, а одному номеру паспорта — не более одного конкретною человека.

2. «Один ко многим». Одному элементу первого объекта может соответствовать несколько элементов второго объекта, а одному элементу второго объекта может

соответствовать только один элемент первого объекта. Например, в 11 «А» классе школы № 5 может учиться несколько учеников, а конкретный ученик школы № 5 может учиться не более чем в одном классе.

3. «Многие ко многим». Каждому элементу первого объекта может соответствовать множество элементов второго, и каждому элементу второго - множество элементов первого. Например, один предмет школьной программы могут изучать многие ученики, и один ученик может изучать многие предметы школьной программы.

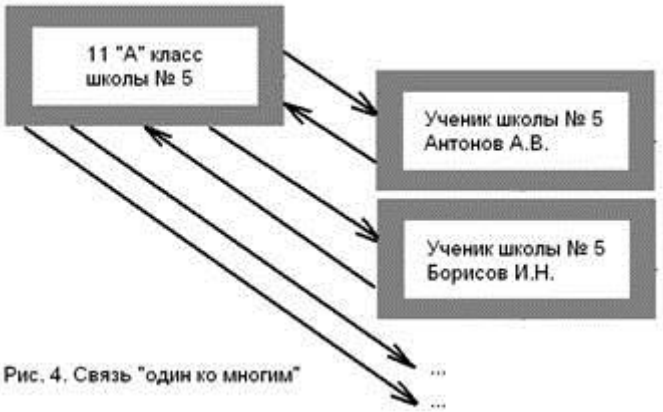


Рис. 4. Связь "один ко многим"

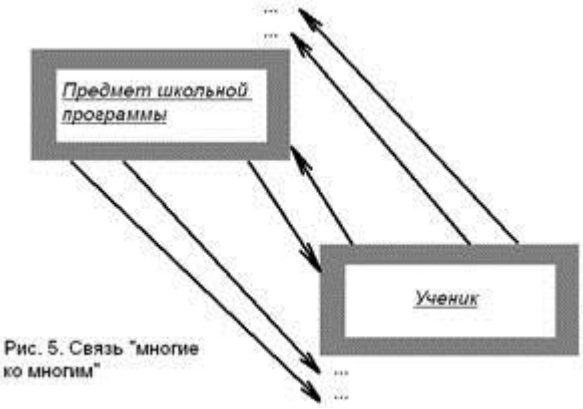


Рис. 5. Связь "многие ко многим"

Второй вопрос: <u>Введение в базы данных (БД) и системы управления базами данных (СУБД). Основные понятия и определения.</u>

Создание, использование, ведение, сопровождение баз и банков данных, информационных систем в настоящее время являются неотъемлемой частью хозяйственной, административной, других видов деятельности предприятий, организаций, юридических, физических лиц в целях накопления, хранения, оперативного доступа к необходимой информации, регламентированного информационного обмена между заинтересованными субъектами.

Восприятие реального мира можно соотнести с последовательностью разных, хотя иногда и взаимосвязанных, явлений. С давних времен люди пытались описать эти явления (даже тогда, когда не могли их понять). Такое описание называют данными.
Традиционно фиксация данных осуществляется с помощью конкретного средства общения, например, с помощью естественного языка на конкретном носителе.

В настоящее время успешное функционирование различных фирм, организаций и предприятий просто не возможно без развитой информационной системы, которая позволяет автоматизировать сбор и обработку данных. Обычно для хранения и доступа к данным, содержащим сведения о некоторой предметной области, создается база данных.

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

Под предметной областью принято понимать некоторую область человеческой деятельности или область реального мира, подлежащих изучению для организации управления и автоматизации, например, предприятие, вуз и т.д.

Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, наполнения, обновления и удаления баз данных.

Программы, с помощью которых пользователи работают с БД, называются приложениями.

Третий вопрос: <u>Принципы построения баз данных и управления ими.</u>

К современным базам данных, а, следовательно, и к СУБД, на которых они строятся, предъявляются следующие основные требования.

1. Высокое быстродействие (малое время отклика на запрос).
   Время отклика – промежуток времени от момента запроса к БД до фактического получения данных. Похожим является термин время доступа – промежуток времени между выдачей команды записи (считывания) и фактическим получением данных. Под доступом понимается операция поиска, чтения данных или записи их. Часто операции записи, удаления и модификации данных называют операцией обновления.

2. Простота обновления данных.
3. Независимость данных.
4. Совместное использование данных многими пользователями.
5. Безопасность данных − защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения.
6. Стандартизация построения и эксплуатации БД (фактически СУБД).
7. Адекватность отображения данных соответствующей предметной области.
8. Дружелюбный интерфейс пользователя.

Важнейшими являются первые два противоречивых требования: повышение быстродействия требует упрощения структуры БД, что, в свою очередь, затрудняет процедуру обновления данных, увеличивает их избыточность.

Независимость данных − возможность изменения логической и физической структуры БД без изменения представлений пользователей. Независимость данных предполагает инвариантность к характеру хранения данных, программному обеспечению и техническим средствам. Она обеспечивает минимальные изменения структуры БД при изменениях стратегии доступа к данным и структуры самих исходных данных. Это достигается «смещением» всех изменений на этапы концептуального и логического проектирования с минимальными изменениями на этапе физического проектирования.

Безопасность данных включает их целостность и защиту.

Целостность данных − устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей.

Она предполагает:

1. отсутствие неточно введенных данных или двух одинаковых записей об одном и том же факте;
2. защиту от ошибок при обновлении БД;
3. невозможность удаления (или каскадное удаление) связанных данных разных таблиц;
4. не искажение данных при работе в многопользовательском режиме и в распределенных базах данных;
5. сохранность данных при сбоях техники (восстановление данных).

Целостность обеспечивается триггерами целостности – специальными приложениями-программами, работающими при определенных условиях. Защита данных от несанкционированного доступа предполагает ограничение доступа к конфиденциальным данным и может достигаться:

1. введением системы паролей;
2. получением разрешений от администратора базы данных (АБД);
3. запретом от АБД на доступ к данным;
4. формирование видов − таблиц, производных от исходных и предназначенных конкретным пользователям.

Три последние процедуры легко выполняются в рамках языка структуризованных запросов Structured Query Language − SQL.

Стандартизация обеспечивает преемственность поколений СУБД, упрощает взаимодействие БД одного поколения СУБД с одинаковыми и различными моделями данных. Стандартизация (ANSI/SPARC) осуществлена в значительной степени в части интерфейса пользователя СУБД и языка SQL. Это позволило успешно решить задачу взаимодействия различных реляционных СУБД как с помощью языка SQL, так и с

применением приложения Open DataBase Connection (ODBC). При этом может быть осуществлен как локальный, так и удаленный доступ к данным (технология клиент/сервер или сетевой вариант).

Четвертый вопрос: Основные этапы проектирования баз данных (проектирование и создание базы данных (практическое занятие № 15, теоретическая часть).

Процесс разработки базы данных обычно начинается с построения информационной модели предметной области (ПО). На приведенном ниже рисунке 1 показаны основные этапы создания и эксплуатации БД.

Жизненный цикл базы данных можно условно разбить на 2 фазы.

1. Анализа и проектирования;
2. Реализации и функционирования.

На этапе анализа и проектирования осуществляется:

1. Формирование и анализ требований к информации о предметной области.
   Здесь осуществляется сбор требований к содержанию и процессу обработки данных от всех пользователей, обеспечивается согласованность данных.

2. Концептуальное проектирование.
   Построение независимой от СУБД информационной структуры путем объединения требований пользователей. Концептуальная схема не зависит от конкретной СУБД и технических решений.

3. Проектирование реализации.
   На этом этапе осуществляется реализация информационной модели в рамках конкретной СУБД. Производится описание структуры данных, разработка программ обработки данных.

   3.1. Логическое проектирование.
   Высокоуровневое представление данных преобразуется в структуру используемой СУБД. Основной целью этапа является устранение избыточности данных с использованием специальных правил нормализации. Цель нормализации – минимизировать повторения данных и возможные структурные изменения БД при процедурах обновления. Это достигается разделением (декомпозицией) одной таблицы в две или несколько с последующим использованием при запросах операции навигации. Навигационный поиск снижает быстродействие БД, т.е. увеличивает время отклика на запрос.

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

3.2. Физическое проектирование.

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

На этапе реализации и функционирования БД осуществляется:

1. Реализация БД.
2. Анализ функционирование и поддержка.
3. Модификация и адаптация.

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

К количественным критериям относятся время ответа на запрос, стоимость модификации, стоимость памяти, время на создание, стоимость на реорганизацию.

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

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

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

Основными причинами низкой эффективности проектируемых БД могут быть:

1. недостаточно глубокий анализ требований (начальные этапы проектирования), включая их семантику и взаимосвязь данных;

2. большая длительность процесса структурирования, делающая этот процесс утомительным и трудно выполняемым при ручной обработке.

В этих условиях важное значение приобретают вопросы автоматизации разработки.

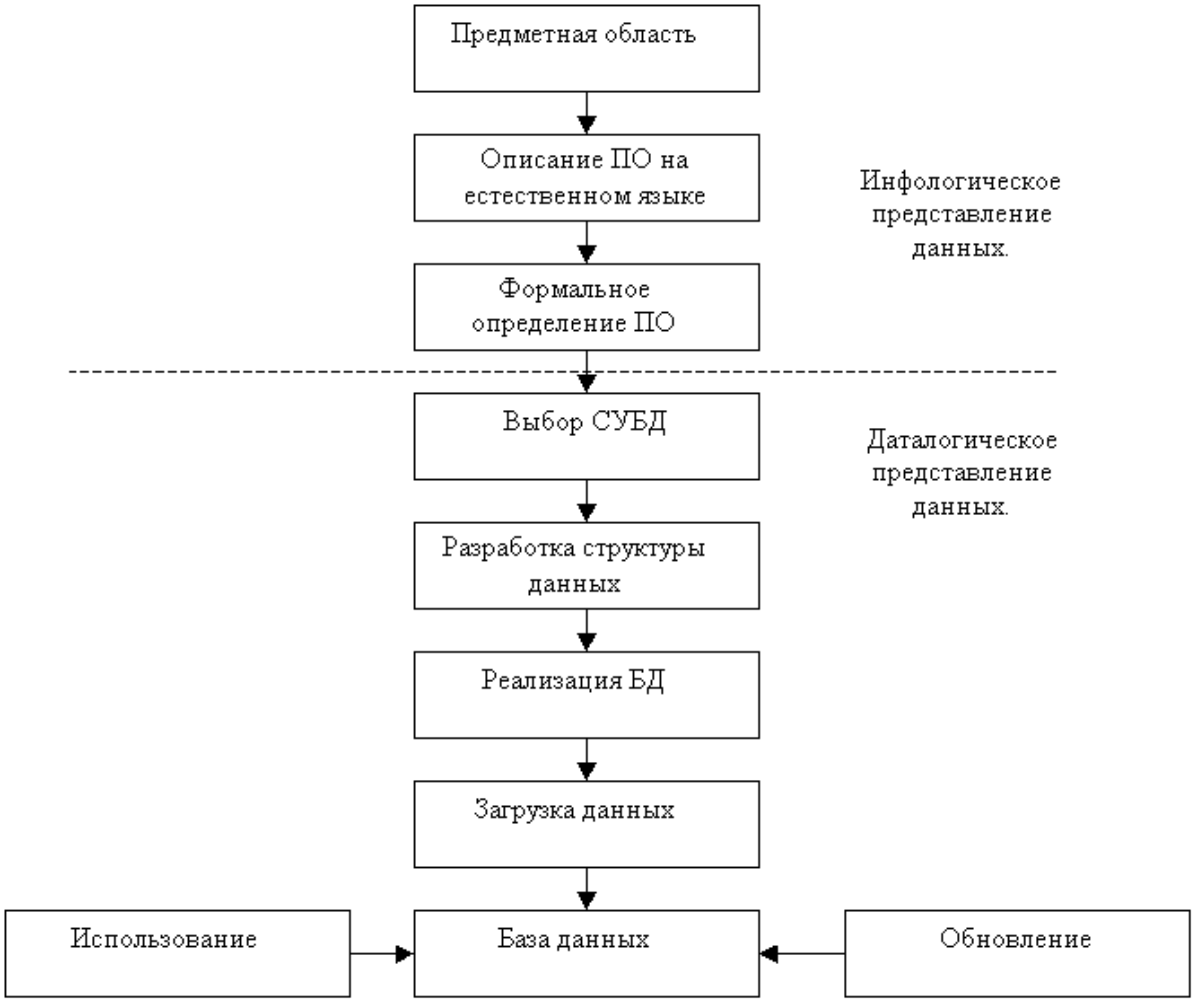


Рисунок 1. Основные этапы создания и эксплуатации БД.

При проектировании БД на внешнем уровне необходимо изучить функционирование объекта управления, для которого проектируется БД, всю первичную и выходную документацию с точки зрения определения того, какие именно данные необходимо сохранять в базе данных. Внешний уровень это, как правило, словесное описание входных и выходных сообщений, а также данных, которые целесообразно сохранять в БД. Описание внешнего уровня не исключает наличия элементов дублирования, избыточности и несогласованности данных. Поэтому для устранения этих аномалий и противоречий внешнего описания данных выполняется инфологическое проектирование. Инфологическая модель является средством структуризации предметной области и понимания концепции семантики данных.

Инфологическую модель можно рассматривать в основном как средство документирования и структурирования формы представления информационных потребностей, которая обеспечивает непротиворечивое общение пользователей и разработчиков системы.

Все внешние представления интегрируются на инфологическом уровне, где формируется инфологическая (каноническая) модель данных, которая не является простой суммой внешних представлений данных.

Инфологический уровень представляет собой информационно-логическую модель (ИЛМ) предметной области, из которой исключена избыточность данных и отображены информационные особенности объекта управление без учета особенностей и специфики конкретной СУБД. То есть инфологическое представление данных ориентированно преимущественно на человека, который проектирует или использует базу данных.

Логический (концептуальный) уровень построен с учетом специфики и особенностей конкретной СУБД. Этот уровень представления данных ориентирован больше на компьютерную обработку и на программистов, которые занимаются ее разработкой. На этом уровне формируется концептуальная модель данных, то есть специальным способом структурированная модель предметной области, которая отвечает особенностям и ограничениям выбранной СУБД. Модель логического уровня, поддерживаемую средствами конкретной СУБД, называют еще даталогической.

Инфологическая и даталогическая модели, которые отображают модель одной предметной области, зависимы между собой. Инфологическая модель может легко трансформироваться в даталогическую модель.

Внутренний уровень связан с физическим размещением данных в памяти ЭВМ. На этом уровне формируется физическая модель БД, которая включает структуры сохранения данных в памяти ЭВМ, в т.ч. описание форматов записей, порядок их логического или физического приведения в порядок, размещение по типам устройств, а также характеристики и пути доступа к данным.

От параметров физической модели зависят такие характеристики функционирования БД: объем памяти и время реакции системы. Физические параметры БД можно изменять в процессе ее эксплуатации с целью повышения эффективности функционирование системы. Изменение физических параметров не предопределяет необходимости изменения инфологической и даталогической моделей.

Схема взаимосвязи уровней представления данных в БД изображена на рисунке 2. В соответствии с этими уровнями проектируется БД. Проектирование БД – это сложный и трудоемкий процесс, который требует привлечения многих высококвалифицированных специалистов. От того, насколько квалифицированно спроектирована БД, зависят производительность информационной системы и полнота обеспечения функциональных потребностей пользователей и прикладных программ. Неудачно спроектированная БД может усложнить процесс разработки прикладного программного обеспечения, обусловить необходимость использования более сложной логики, которая, в свою очередь, увеличит время реакции системы, а в дальнейшем может привести к необходимости перепроектирования логической модели БД. Реструктуризация или внесение изменений в логическую модель БД это очень нежелательный процесс,

поскольку он является причиной необходимости модификации или даже перепрограммирование отдельных задач.

Все работы, которые выполняются на каждом этапе проектирования, должны интегрироваться со словарем данных. Каждый этап проектирования рассматривается как определенная последовательность итеративных процедур, в результате которых формируется определенная модель БД.

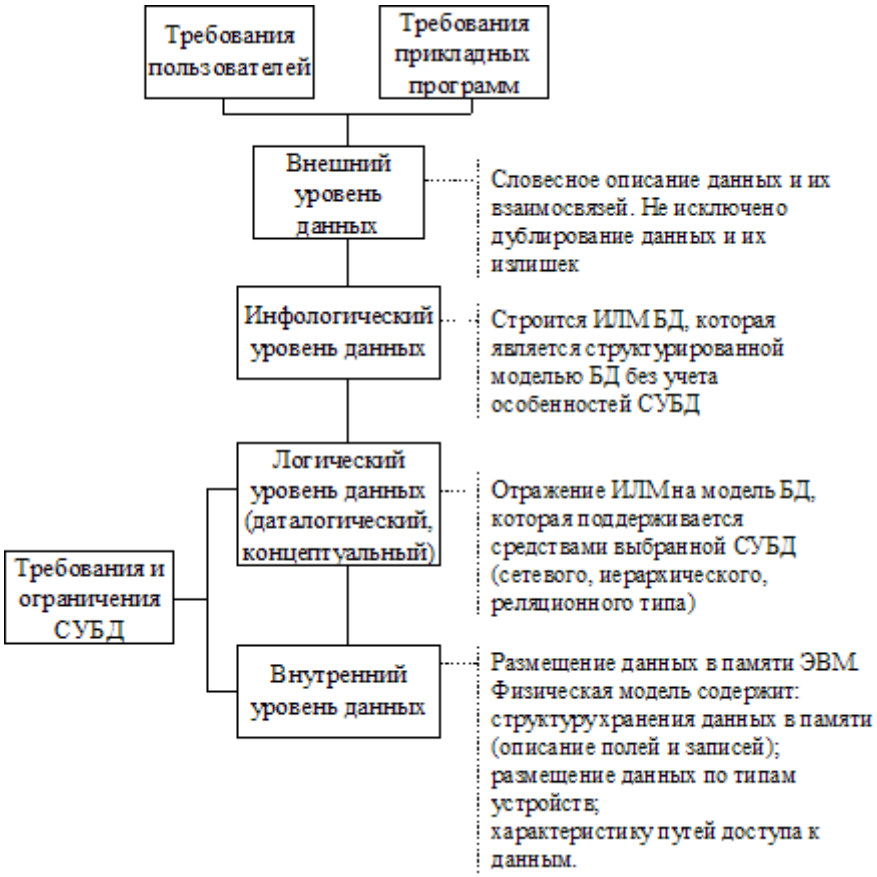


Рисунок 2. *Схема взаимосвязи уровней представление данных в БД*

Пятый вопрос: Основные элементы базы данных. Составные части инфологической модели. Реляционная модель данных.

Модель данных − средство абстракции, позволяющее увидеть информационное содержание данных, а не их конкретные значения. Обеспечивает интерпретацию данных в соответствии с указанными требованиями.

Единица информации может быть представлена в виде:

Единица данных = < имя объекта, свойство объекта, значение свойства>.

Для обработки на компьютере данные должны быть каким-то образом упорядочены. Совокупность именованных объектов, их свойств и связей между ними – схема данных.

Основными составными элементами инфологической модели являются сущности (информационные объекты), связи между ними и их атрибуты (свойства).

Один из основных способов структуризации данных при построении модели предметной области – абстракция. Широко используется обобщение и агрегация.

Обобщение позволяет соотнести множество знаков или типов с одним общим типом.

Агрегация – конструирование объекта из других (базовых) объектов.

На основе обобщения формируются атрибуты, представляющие семантически значимые объекты.

Атрибуты существуют не сами по себе, а как компоненты других объектов. Интерпретация атрибутов и соотношений между ними определяется агрегатами, соответствующими объектам реального мира.

Отношение – агрегат, объединяющий между собой несколько атрибутов. Может описывать какие-то объекты. Отношению можно придать различную семантическую окраску, например, соотнести каждый кортеж с конкретной сущностью.

Сущность – это нечто, принадлежащее объективной реальности. Взаимосвязь различных типов сущностей может быть задана другими отношениями – связями. Тип связи – агрегат атрибутов двух и более типов сущностей. В общем случае принципиальной разницы между типами сущности и связи нет.

Для формализации описание предметной области широкое распространение получила модель «сущность-связь».

Сущность – основное содержание явлений или процессов.

Атрибут – наименование характеристики сущности.

Связи – ассоциации между различными сущностями.

Для построения модели необходимо:

1. определение сущностей,
2. определение атрибутов сущностей,
3. определение ключевых атрибутов сущностей,
4. определение связей между сущностями.

В качестве сущностей в моделях предметной области рассматривают материальные (предприятия, изделия, сотрудники и т. п.) и нематериальные (явления, структуры) объекты реальной действительности. В моделях предметной области типа “сущность – связь” каждая рассматриваемая конкретная сущность является узловой точкой сбора информации об этой сущности. В модели используется также понятие “экземпляр сущности”. Графически модель “сущность – связь” представляется в виде ER-диаграмм (Entity-Relation Diagrams).

Реализация информационных моделей осуществляется в рамках СУБД.

СУБД поддерживает некоторую модель данных, которая обеспечивает логику информационной модели предметной области.

Модель данных должна обеспечивать:

– допустимую структуру данных для описания разнообразных объектов;

– множество допустимых операций над данными;

– ограничение контроля целостности (логические ограничения модели на данные для сохранения непротиворечивости данных и адекватного отображения предметной области в БД).

Модели данных, поддерживаемые различными СУБД, делят на иерархические, сетевые и реляционные. На персональных компьютерах наибольшее распространение получили реляционные модели данных.

В основе реляционной модели лежит понятие отношения, которое удобно представлять в виде двумерной таблицы. Набор отношений может быть использован для хранения данных об объектах и моделирования связи между ними.

Схема отношения – множества именованных столбцов.

Имена столбцов – атрибуты.

Строки – кортежи. Кортежи образуют множества.

Из множества имен атрибутов можно выделить подмножество, по которому можно уникально различить кортежи, отношения. Это подмножество – ключ отношения.

Степень отношения – число атрибутов.

Мощность – количество картежей.

К отношениям можно применять стройную систему операций, позволяющие получать одни отношения из других, вычислять не хранимую информацию.

Для обработки на компьютере таблицы должны удовлетворять определенным требованиям. Отношения должны быть нормализованы. Рассмотрим основные требования к отношениям.

1. Каждый атрибут должен быть простым.
   То есть должен содержать только одно свойство объекта.
2. Не может быть одинаковых первичных ключей (все записи уникальны).
3. Все строки должны иметь одну структуру (одинаковое количество атрибутов с совпадающими именами).
4. Имена столбцов различны , а значения однородны.
5. Ссылочная целостность ключей (каждому внешнему ключу должна соответствовать строка какого-либо объектного отношения). Если

отношение удовлетворяет этим условиям, то оно находиться в 1-ой нормальной форме.

Потенциальный ключ − в реляционной модели данных − подмножество атрибутов отношения, удовлетворяющее требованиям уникальности и минимальности (несократимости).

Уникальность означает, что не существует двух кортежей данного отношения, в которых значения этого подмножества атрибутов совпадают (равны).

Минимальность (несократимость) означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, удовлетворяющее условию уникальности. Иными словами, если из потенциального ключа убрать любой атрибут, он утратит свойство уникальности.

Поскольку все кортежи в отношении по определению уникальны, в нём всегда существует хотя бы один потенциальный ключ (например, включающий все атрибуты отношения).

В отношении может быть одновременно несколько потенциальных ключей. Один из них может быть выбран в качестве первичного ключа отношения, тогда другие потенциальные ключи называют альтернативными ключами

Теоретически, все потенциальные ключи равно пригодны в качестве первичного ключа, на практике в качестве первичного обычно выбирается тот из потенциальных ключей, который имеет меньший размер (физического хранения) и/или включает меньшее количество атрибутов.

Первичный ключ (англ. primary key) − в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию).

Если в отношении имеется единственный потенциальный ключ, он является и первичным ключом. Если потенциальных ключей несколько, один из них выбирается в качестве первичного, а другие называют «альтернативными».

С точки зрения теории все потенциальные ключи отношения эквивалентны, то есть обладают одинаковыми свойствами уникальности и минимальности. Однако в качестве первичного обычно выбирается тот из потенциальных ключей, который наиболее удобен для тех или иных практических целей, например для создания внешних ключей в других отношениях либо для создания кластерного индекса. Поэтому в качестве первичного ключа как правило выбирают тот, который имеет наименьший размер (физического хранения) и/или включает наименьшее количество атрибутов.

Исторически термин «первичный ключ» появился и стал использоваться существенно ранее термина «потенциальный ключ». Вследствие этого множество определений в реляционной теории были изначально сформулированы с упоминанием первичного (а не потенциального) ключа, например, определения нормальных форм. Так же термин «первичный ключ» вошёл в формулировку 12 правил Кодда как основной способ адресации любого значения отношения (таблицы) наряду с именем отношения (таблицы) и именем атрибута (столбца).

Если первичный ключ состоит из единственного атрибута, его называют **простым ключом.**

Если первичный ключ состоит из двух и более атрибутов, его называют **составным ключом.**

Первичный ключ может состоять из информационных полей таблицы (то есть полей, содержащих полезную информацию об описываемых объектах). Такой первичный ключ называют **естественным ключом.**

Теоретически, естественный ключ всегда можно сформировать, в этом случае мы получим так называемый и**нтеллектуальный ключ.**

Суперключ − в реляционной модели данных — подмножество атрибутов отношения, удовлетворяющее требованию уникальности: не существует двух кортежей данного отношения, в которых значения этого подмножества атрибутов совпадают (равны).

Суперключ отличается от потенциального ключа тем, что на суперключ не накладывается требование минимальности, или несократимости (это требование означает, что в составе ключа отсутствует меньшее подмножество атрибутов, удовлетворяющее условию уникальности). Вследствие этого в состав суперключа может входить другой, более «компактный» по количеству атрибутов суперключ.

Потенциальный ключ может быть определён как суперключ, обладающий свойством минимальности (несократимости).

Поскольку все кортежи в отношении по определению уникальны, в нём всегда существует хотя бы один суперключ (например, включающий все атрибуты отношения).

Таким образом, реляционная модель данных − логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам

обработки данных таких разделов математики как теории множеств и логика первого порядка.

Реляционная модель данных включает следующие компоненты:

- структурный аспект (составляющая) − данные в базе данных представ-ляют собой набор отношений.
- аспект (составляющая) целостности − отношения (таблицы) отвечают определенным условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.
- аспект (составляющая) обработки (манипулирования) − РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

Кроме того, в состав реляционной модели данных включают теорию нормализации. Нормализация отношений – это пошаговый обратимый процесс разложения исходных отношений на более мелкие.

Состав атрибутов должен обеспечивать:

1. отсутствие нежелательных функциональных зависимостей,
2. минимальное дублирование данных.

Существует огромное количество разновидностей баз данных, отличающихся по различным критериям.

Основные классификации баз данных:

1. по модели данных,
2. по среде физического хранения,
3. по содержимому,
4. по степени распределённости.

Шестой вопрос: Классификация баз данных. Базы данных (табличные, иерархические, сетевые).

Классификация БД по модели данных:

- иерархические,
- сетевые,
- реляционные,
- объектные,
- объектно-ориентированные,
- объектно-реляционные,

- иные БД.

Классификация БД по среде физического хранения:

- БД во вторичной памяти (традиционные): средой постоянного хранения является периферийная энергонезависимая память (вторичная память) – как правило жёсткий диск. В оперативную память СУБД помещает лишь кеш и данные для текущей обработки.
- БД в оперативной памяти *(in-memory databases):* все данные находятся в оперативной памяти.
- БД в третичной памяти (*tertiary databases*): средой постоянного хранения является отсоединяемое от сервера устройство массового хранения (третичная память), как правило на основе магнитных лент или оптических дисков. Во вторичной памяти сервера хранится лишь каталог данных третичной памяти, файловый кеш и данные для текущей обработки; загрузка же самих данных требует специальной процедуры.
- иные БД.

Классификация БД по содержимому:

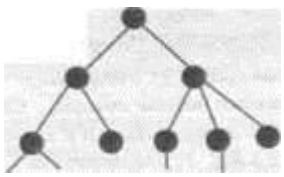
- географические;
- исторические;
- научные;
- мультимедийные;
- иные БД.

Классификация БД по степени распределённости:

- централизованные (сосредоточенные);
- распределённые.

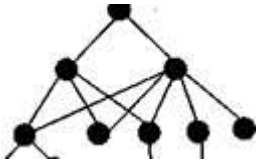
Базы данных (табличные, иерархические, сетевые).

Иерархические базы данных графически могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй — объекты второго уровня и т. д.*



Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Иерархической базой данных является Каталог папок Windows.

Сетевая база данных отличается от иерархической тем, что в ней каждый элемент вышестоящего уровня может быть связан одновременно с любыми элементами следующего уровня.



Вообще, на связи между объектами в сетевых моделях не накладывается никаких ограничений. Сетевой базой данных фактически является Всемирная паутина глобальной компьютерной сети Интернет. Гиперссылки связывают между собой сотни миллионов документов в единую распределенную сетевую базу данных.

Табличная (или реляционная) база данных содержит перечень объектов одного типа, т. е. объектов с одинаковым набором свойств. Такую базу данных удобно представлять в виде двумерной таблицы (а чаще — нескольких связанных между собой таблиц).

Основные понятия (поле, запись, первичный ключ записи) Столбцы таблицы называют полями; каждое поле характеризуется своим именем (названием соответствующего свойства) и типом данных, отражающих значения данного свойства. Каждое поле обладает определенным набором свойств (размер, формат и др.). Поле базы данных — это столбец таблицы, включающий в себя значения определенного свойства.

Строки таблицы являются записями об объекте; эти записи разбиты на поля столбцами таблицы. Запись базы данных — это строка таблицы, которая содержит набор значений различных свойств объекта.

В каждой таблице должно быть, по крайней мере, одно ключевое поле, содержимое которого уникально для любой записи в этой таблице**.** Значения ключевого поля однозначно определяют каждую запись в таблице. В нашей таблице ключевым полем является поле «Номер личного дела». Очень часто в качестве ключевого поля используется поле, содержащее тип данных *Счетчик.*

Типы данных

В реляционных базах данных используются следующие основные типы полей:

• Счетчик — целые числа, которые задаются автоматически при вводе записей и не могут быть изменены пользователем.

• Числовой. Этот тип имеют поля, значения которых могут быть только числами.

• Символьный (или текстовый) — такой тип имеют поля, в которых хранятся символьные последовательности (слова, тексты, коды и пр.), содержащие до 255 символов.

• Дата/время — дата и время.

• Логический — значения *Истина* или *Ложь* (или «да»/ «нет»).

От типа величины зависят те действия, которые можно с ней производить. Например, с числовыми величинами можно выполнять арифметические операции, а с символьными и логическими — нельзя.

Седьмой вопрос: Системы управления базами данных. Характеристики СУБД.

Система управления базами данных (СУБД) − совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Программное обеспечение, которое получило название **системы управления базами данных – СУБД**, позволяет обеспечить доступ к одним и тем же данным со стороны различных прикладных программ. СУБД служат для создания баз данных, их заполнения и корректировки, поиска и выборки необходимой информации и ее представления в наглядном виде.

В состав СУБД входят пакеты программ, библиотеки, а также логическое описание структуры данных и их физическое описание (где и как записаны данные на магнитных носителях), создаваемые СУБД при инициации БД и ее структуры.

Таким образом, СУБД является программным средством, обеспечивающим интерфейс (взаимодействие) прикладных программ и операционной системы, в процессе обработки данных. Современные СУБД содержат развитый пользовательский интерфейс, который позволяет вводить и модифицировать информацию, выполнять поиск и представлять информацию в текстовом или графическом виде, а также средства программирования высокого уровня, с помощью которых можно создавать собственные приложения.

Классификация СУБД по модели данных:

- иерархические,
- сетевые,
- реляционные,
- объектно-ориентированные,
- иные БД.

Классификация СУБД по степени распределённости:

- локальные СУБД (все части локальной СУБД размещаются на одном компьютере),

- распределённые СУБД (части СУБД могут размещаться на двух и более компьютерах).

Классификация СУБД по способу доступа к БД:

- файл-серверные,
- клиент-серверные,
- встраиваемые.

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на ЦП сервера. Недостатки: потенциально высокая загрузка локаль-

ной сети; затруднённость централизованного управления; затрудненность обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность. Применяются чаще всего в локальных приложениях, которые используют функции управления БД.

На данный момент файл-серверная технология считается устаревшей.

Примеры: Microsoft Access, Paradox, dBase, FoxPro, Visual FoxPro.

Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно. Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу. Достоинства: потенциально более низкая загрузка локальной сети; удобство централизованного управления; удобство обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность.

Примеры: Oracle, Firebird, Interbase, IBM DB2, Informix, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MySQL, Caché, ЛИНТЕР.

Встраиваемая СУБД (англ. embedded DBMS) − СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установки. Встраиваемая СУБД предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически встраиваемая СУБД чаще всего реализована в виде подключаемой библиотеки. Доступ к данным со стороны приложения может происходить через SQL либо через специальные программные интерфейсы.

Примеры: OpenEdge, SQLite, BerkeleyDB, Firebird Embedded, MySQL, Sav Zigzag, Microsoft SQL Server Compact, ЛИНТЕР.

Восьмой вопрос: <u>Работа с базами данных. Режим работы базы данных.</u>

Основными категориями персонала, выполняющего работу с базами данных являются:

1. пользователи – лица, в интересах которых создается база данных, и потребляющие информацию в процессе своей работы;
2. прикладные программисты – лица, осуществляющие разработку прикладных программ, то есть средств, позволяющих пользователям в автоматизированном режиме получать и обрабатывать информацию,
   необходимую в процессе работы;

3. администратор базы данных – лицо, несущее ответственность за реализацию требований пользователей по составу, содержанию и качеству информации базы данных, а также за обеспечение защиты информации, в том числе и за разграничение прав доступа.
4. отдельной группой лиц, участвующих в работах с базами данных, можно считать системных программистов и специалистов по техническому обеспечению, выполняющих функции по обеспечению корректной работы средств общего программного обеспечения и технических средств системы.

Хорошо спроектированные системы управления БД (СУБД), используют развитые графические интерфейсы и поддерживают системы отчетов, отвечающие специфике пользователей **указанных четырех** категорий. В этом случае персонал поддержки БД и конечные пользователи могут легко осваивать и использовать СУБД для обеспечения своих потребностей без какой-либо специальной подготовки, т.е. специфика функционирования данных систем скрыта от пользователя. Более того, хорошо спроектированные СУБД предоставляют опытному пользователю средства для создания собственных БД-приложений, не требуя от него специальной программистской подготовки. **Конечным** пользователям для обеспечения доступа к информации БД предоставляется графический интерфейс, как правило, в виде системы окон с функциональными меню, позволяющими легко получать необходимую информацию на экран и/или принтер в виде удобно оформленных отчетов.

Программисты и системные **аналитики** используют СУБД совершенно в ином качестве, обеспечивая разработку новых БД-приложений, поддерживая и модифицируя (при необходимости) уже существующие. Для данной группы пользователей СУБД требуются средства, обеспечивающие указанные функции (создание, откладка, редактирование и т.д.). Пользователи **четвертой** категории нуждаются в интерфейсе, как правило, графическом для обеспечения задач поддержания БД в актуальном состоянии. Эти пользователи состоят в штатах подразделений функциональных и/или обработки информации, обеспечивающих прикладную область, и отвечают за актуальное состояние соответствующей ей БД (контроль текущего состояния, удаление устаревшей информации, добавление новой и т.д.). **Программисты** выполняют своего рода посреднические функции между БД и **конечными** пользователями. И если на

первых этапах развития БД-технологии они составляли весьма многочисленную группу пользователей, то в процессе развития СУБД и, прежде всего, массового использования ПК эта категория сходит на нет. Особую и ответственную роль выполняет **администратор,** отвечающий как за актуальность находящейся в БД информации, так и за корректность функционирования и использования БД и СУБД.

В случае больших БД может быть достаточно много конечных пользователей, ряд программистов и несколько администраторов БД; в случае небольших БД (что особенно характерно для ПК) все эти функции могут обеспечиваться одним человеком. Важные функции выполняет администратор БД, отвечающий за выработку требований к БД, ее проектирование, реализацию, эффективное использование и сопровождение.

Необходимость в таком специалисте вытекает из принципа независимости данных, а также диктуется важностью БД в деятельности организаций и более крупных объединений – поставщиков и потребителей информации БД. Администратор БД взаимодействует с пользователями в определении требований к базе в процессе выработки требований к системе в целом, пользуется языков описания данных для определения БД в процессе проектирования системы, взаимодействует с программистами, которые соз-дают ПС использующее доступ к БД, отвечает за загрузку БД информацией в процессе реализации системы, контролирует работоспособность БД, используя соответствующие программные и аппаратные средства, и определяет, когда следует реорганизовывать данные в базе или начать работы по созданию новой, более совершенной БД. В целом функции **администратора** БД сводятся к поддержанию целостности БД, необходимого уровня защиты ее данных и эффективности. Среди его наиболее важных обязанностей – **согласование** конфликтующих требований, которое требуется достаточно часто, ибо БД обслуживает, как правило, целый ряд различных прикладных процессов.

Соответственно система управления базами данных имеетдва режима работы: проектировочный и пользовательский. Первый режим предназначен для создания или изменения структуры базы и создания ее объектов. Во втором режиме происходит использование ранее подготовленных объектов для наполнения базы или получения данных из нее. Объекты базы данных. Мы уже упомянули о том, что кроме таблиц база данных может содержать и другие типы объектов. Привести полную классификацию возможных объектов баз данных затруднительно, поскольку каждая система управления базами данных может реализовать свои типы объектов.

Объекты базы данных

Мы уже упомянули о том, что кроме таблиц база данных может содержать и другие типы объектов. Привести полную классификацию возможных объектов баз данных затруднительно, поскольку каждая система управления базами данных может

реализовать свои типы объектов. Однако основные типы объектов мы можем рассмотреть на примере СУБД Microsoft Access. Эта СУБД позволяет создавать и использовать объекты семи различных типов.

Таблицы. Как отмечалось, это основные объекты любой базы данных. Во-первых, в таблицах хранятся все данные, имеющиеся в базе, а во-вторых, таблицы хранят и структуру базы (поля, их типы и свойства).

Запросы. Эти объекты служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции, как отбор данных, их сортировку и фильтрацию. С помощью запросов можно выполнять преобразование данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое наполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и многое другое.

В целях безопасности чем меньше доступа к базовым таблицам имеют конечные пользователи, тем лучше. Во-первых, снижается риск того, что неумелыми действиями они повредят данные в таблицах. Во-вторых, предоставив разным пользователям разные запросы, можно эффективно разграничить их доступ к данным в строгом соответствии с кругом персональных обязанностей.

Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе *временную результирующую таблицу.* Если хотят подчеркнуть факт «временности» этой таблицы, то ее еще называют *моментальным снимком.* Когда мы работаем с основными таблицами базы, мы физически имеем дело с жестким диском, то есть с очень медленным устройством (напомним, что это связано с особенностью сохранения данных, описанной выше). Когда же на основании запроса мы получаем результирующую таблицу, то имеем дело с электронной таблицей, не имеющей аналога на жестком диске: это только *образ* отобранных полей и записей. Разумеется, работа с «образом» происходит гораздо быстрее и эффективнее – это еще одно основание для того, чтобы широко использовать запросы.

Формы. Если запросы – это специальные средства для отбора и анализа данных, то формы – это средства для ввода данных. Смысл их тот же – предоставить пользователю средства для заполнения только тех полей, которые ему положено заполнять. Одновременно с этим в форме можно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и прочие) для автоматизации ввода. Преимущества форм раскрываются особенно наглядно, когда происходит ввод данных с заполненных бланков. В этом случае форму делают графическими средствами так, чтобы она повторяла оформление бланка. Это заметно упрощает работу наборщика, снижает утомление и предотвращает появление печатных ошибок.

С помощью форм данные можно не только вводить, но и отображать. Запросы тоже отображают данные, но делают это в виде результирующей таблицы, не имеющей

почти никаких средств оформления. При выводе данных с помощью форм можно применять специальные средства оформления.

Отчеты. По своим свойствам и структуре отчеты во многом похожи на формы, но предназначены только для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В связи с этим отчеты отличаются тем, что в них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, служебная информация о времени создания отчета и т. п.).

Страницы. Это специальные объекты баз данных, реализованные в СУБД Microsoft Access. Правда, более корректно их называть *страницами доступа к данным.* Физически это особый объект, выполненный в коде *HTML,* размещаемый на Web-странице и передаваемый клиенту вместе с ней. Сам по себе этот объект не является базой данных, но содержит компоненты, через которые осуществляется связь переданной Web-страницы с базой данных, остающейся на сервере. Пользуясь этими компонентами, посетитель Web-узла может просматривать записи базы в полях страницы доступа. Таким образом, страницы доступа к данным осуществляют интерфейс между клиентом, сервером и базой данных, размещенной на сервере. Эта база не обязательно должна быть базой данных Microsoft Access. Страницы доступа, созданные средствами Microsoft Access, позволяют работать также с базами данных Microsoft SQL Server.

Макросы и модули. Эти категории объектов предназначены как для автоматизации повторяющихся операций при работе с системой управления базами данных, так и для создания новых функций путем программирования. В СУБД Microsoft Access *макросы* состоят из последовательности внутренних команд СУБД и являются одним из средств автоматизации работы с базой. *Модули* создаются средствами внешнего языка программирования, в данном случае языка Visual Basic for Applications. Это одно из средств, с помощью которых разработчик базы может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, а также уровень ее защищенности.

Девятый вопрос: Создание и редактирование таблиц.

Большинство баз данных имеет табличную структуру. Таблицы являются основой, на которой строится все дальнейшее создание базы данных. В программе Access предусмотрено несколько вариантов построения таблиц, их можно увидеть при выборе в области объектов раздела Таблицы**.**

Таблицы, как и любой другой объект базы данных, имеют два основных режима создания:

1. режим Конструктора;

2. режим Таблицы.

И два дополнительных режима:

1. режим Сводной таблицы;
2. режим Сводной диаграммы.

В режиме Таблицы осуществляется работа с данными, находящимися в таблице – просмотр, редактирование, добавление, сортировка и т. п. В режиме Конструктора создается или модифицируется структура таблицы, т. е. задаются имена полей таблицы и их типы, поля описываются, задаются их свойства. В режимах Сводной таблицы и Сводной диаграммы удобно выполнять анализ данных, динамически изменяя способы их представления.

Существует также дополнительный режим — режим Предварительного просмотра, который позволяет увидеть расположение данных на листе перед осуществлением печати таблицы.

Создание таблицы в режиме Конструктора

В режиме Конструктора таблицы создаются путем задания имен полей, их типов и свойств. Чтобы создать таблицу в режиме Конструктора, необходимо:

1. Дважды щелкнуть левой кнопкой мыши на ярлыке Создание таблицы с помощью конструктора или нажать на кнопку Создать в верхней части окна базы данных, выбрать из списка в окне Новая таблица элемент Конструктор и нажать кнопку

   ОК. В том и в другом случае откроется пустое окно Конструктора таблиц.

2. В окне Конструктора таблиц в столбец Имя поля ввести имена полей создаваемой таблицы.
3. В столбце Тип данных для каждого поля таблицы выбрать из раскрывающегося списка тип данных, которые будут содержаться в этом поле.
4. В столбце Описание можно ввести описание данного поля (не обязательно).
5. В нижней части окна Конструктора таблиц на вкладках Общие и Подстановка ввести свойства каждого поля или оставить значения свойств, установленные по умолчанию.
6. После описания всех полей будущей таблицы нажать кнопку Закрыть (в верхнем правом углу окна таблицы).
7. На вопрос Сохранить изменения макета или структуры таблицы <имя таблицы>? нажать кнопку Да, в поле Имя таблицы ввести имя создаваемой таблицы и нажать кнопку ОК.
8. В ответ на сообщение Ключевые поля не заданы и вопрос Создать ключевое поле сейчас? нажмите кнопку Да если ключевое поле необходимо, или кнопку Нет если такого не требуется.

После указанных действий в списке таблиц в окне базы данных появятся имя и значок новой таблицы. Ввести данные в созданную таблицу можно, открыв таблицу в режиме Таблицы.

Имена полей в окне заполняются согласно разработанной на бумаге таблице. При этом необходимо придерживаться ряда правил.

- Имена полей в таблице не должны повторяться, т. е. должны быть уникальными.
- Имена полей могут содержать не более 64 символов, включая пробелы.
- Желательно избегать употребления имен полей, совпадающих с именами встроенных функций или свойств Microsoft Access (например, Name — имя).
- Имя поля не должно начинаться с пробела или управляющего символа (коды ASCII 00-31).
- Имена полей могут содержать любые символы, включая буквы, цифры, пробелы, специальные символы, за исключением точки (.), восклицательного знака (!), апострофа (') и квадратных скобок ([), (]).

Тип данных выбирается из списка. Раскрывающийся список открывается только при установке кур сора в ячейку поля Тип данных. В этом случае она примет вид раскрывающегося списка, и щелчок по стрелке откроет список типов полей. В Microsoft Access имеются следующие типы данных:

- *Текстовый* — символьные или числовые данные, не требующие вычислений. Поле данного типа может содержать до 255 символов.
- *Поле МЕМО* — поле МЕМО предназначено для ввода текстовой информации, по объему превышающей 255 символов. Такое поле может содержать до 65 535 символов. Этот тип данных отличается от типа Текстовый тем, что в таблице хранятся не сами данные, а ссылки на блоки данных, хранящиеся отдельно. За счет этого ускоряется обработка таблиц (сортировка, поиск и т. п.). Поле типа МЕМО не может быть ключевым или проиндексированным.
- *Числовой* — числовой тип применяется для хранения числовых данных, используемых в математических расчетах. Имеет много подтипов. От выбора подтипа (размера) данных числового типа зависит точность вычислений. Данные этого типа могут содержаться в 1, 2, 4, 8_oили 16 байтах. Обычно по умолчанию используется подтип Длинное целое, который занимает 4 байта и представляет собой число в пределах от -2 147 483 648 до +2 147 483 647. Но, кроме этого типа, можно указать Байт — 1 байт, Целое — 2 байта, Одинарное с плавающей точкой — 4 байта, Двойное с плавающей точкой — 8 байтов, Десятичное — 12 байтов, Код репликации — 16 байтов.
- *Дата/Время* — тип для представления даты и времени. Позволяет вводить даты с 100 по 9999 год. Размер поля — 8 байтов. Даты и время хранятся в специальном фиксированном числовом формате. Дата является целой частью значения поля типа Дата/Время, а время — его дробной частью. Access предоставляет большой выбор форматов отображения даты и времени.

- *Денежный* — тип данных, предназначенный для хранения данных, точность представления которых колеблется от 1 до 4 десятичных знаков. Целая часть данного типа может содержать до 15 десятичных знаков.
- *Счетчик* — поле содержит 4-байтный уникальный номер, определяемый Microsoft Access автоматически для каждой новой записи либо случайным образом, либо путем увеличения предыдущего значения на 1. Значения полей типа счетчика обновлять нельзя. Максимальное число записей в таблице с полем счетчика не должно превышать двух миллиардов.
- *Логический* — логическое поле, которое может содержать только два значения, интерпретируемых как Да/Нет, Истина/Ложь, Включено/Выключено. Поля логического типа не могут быть ключевыми, но их можно индексировать.
- *Поле объекта OLE* — содержит ссылку на OLE-объект (лист Microsoft Excel, документ Microsoft Word, звук, рисунок и т. п.). Объем объекта ограничивается имеющимся в наличии дисковым пространством. Тип объекта OLE не указывается в свойствах поля объекта OLE. Он зависит от приложения, использованного для создания объектов OLE, которые хранятся в этом поле. Упаковщик объектов позволяет внедрять файлы, созданные приложениями, которые не являются серверами объектов OLE.
- *Гиперссылка* — дает возможность хранить в поле ссылку, с помощью которой можно ссылаться на произвольный фрагмент данных внутри файла или Web-страницы на том же компьютере, в интранет или в Интернет. Гиперссылка состоит из четырех частей: отображаемый текст, адрес (путь к файлу или странице), дополнительный адрес (положение внутри файла или страницы) и текст всплывающей подсказки. Каждая часть гиперссылки может содержать до 2048 символов. Поле типа Гиперссылка не может быть ключевым или индексированным.
- В поле типа можно также выбрать значение Мастер подстановок, который запускает Мастера подстановок, создающего поле подстановок. *Поле подстановок* позволяет выбирать значение поля из списка, содержащего набор постоянных значений или значений из другой таблицы.

После установки типа поля автоматически открывается раздел Свойства поля, который соответствует выбранному типу. С правой стороны для каждой ячейки этих свойств даются пояснения.

Свойства полей таблицы зависят от типа поля. Для большинства типов данных характерно свойство Подпись. С помощью этого свойства можно задать названия полей таблицы, которые выводятся в различных режимах (в надписях, присоединенных к элементам управления формы, в заголовке столбца в режиме Таблицы; в строке заголовка в режиме Формы; в заголовке отчета, выводящемся в режиме Предварительного просмотра; текст, который Выводится в элементе управления). Поле может содержать до 2048 символов. Кроме того, для большинства типов данных существует свойство Обязательное поле, которое определяет необходимость ввода данных в это поле. Свойство Формат поля указывает формат отображения данных из

поля в режиме Таблицы. Для определения формата полей текстового типа используются специальные символы форматирования. Для числовых полей значение формата можно выбрать из раскрывающегося списка. Для логических полей можно выбрать из списка следующие варианты: Да/Нет, Истина/Ложь, Вкл/Выкл. С помощью свойства Маска ввода указывается маска, позволяющая автоматизировать проверку ввода символов в поле. Она применяется к таким полям, как номер телефона, дата и т. д. Задавать маску ввода можно вручную или с помощью Мастера. Свойство Индексированное поле определяет, является ли данное поле индексированным, и если является, то в каком режиме. Существуют два режима индексирования: Совпадения допускаются и Совпадения не допускаются. В первом случае поле может содержать повторяющиеся значения, во втором — нет. Для большинства типов полей определено свойство Значение по умолчанию. В этом свойстве указывается значение, автоматически добавляемое в поле для каждой новой записи, если это значение не введено пользователем. Два свойства, которые тоже определены для большинства полей, позволяют выполнять проверку данных, вводимых в поле: 1)Условие на значение — свойство определяет условие (ограничение), накладываемое на вводимые в это поле данные. При несоответствии вводимых данных указанному условию выдается сообщение об ошибке; 2)Сообщение об ошибке — свойство определяет то сообщение, которое будет выдаваться пользователю, если при вводе данных не соблюдается условие, указанное в свойстве Условие на значение.

Работа в режиме Конструктора считается неоконченной, если не определены ключевые поля.

Ключевое поле – это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то Microsoft Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов, форм и отчетов.

В Microsoft Access можно выделить три типа ключевых полей:

- Счетчик;
- Простой ключ;
- Составной ключ.

Для создания ключевого поля типа Счетчик необходимо в режиме Конструктора таблиц:

1. включить в таблицу поле счетчика.
2. задать для него автоматическое увеличение на 1.
3. указать это поле в качестве ключевого путем нажатия на кнопку Ключевое поле на панели инструментов Конструктор таблиц.

Для создания простого ключа достаточно иметь поле, которое содержит уникальные значения (например, коды или номера). Если выбранное поле содержит повторяющиеся

или пустые значения, его нельзя определить как ключевое. Для определения записей, содержащих повторяющиеся данные, можно выполнить запрос на поиск повторяющихся записей. Если устранить повторы путем изменения значений невозможно, следует либо добавить в таблицу поле счетчика и сделать его ключевым, либо определить составной ключ.

Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой комбинацию нескольких полей. Для определения составного ключа необходимо:

1. открыть таблицу в режиме Конструктора.
2. выделить поля, которые определить как ключевые.
3. нажать кнопку Ключевое поле на панели инструментов Конструктор таблиц.

С целью ускорения поиска и сортировки данных в любой СУБД используются индексы. *Индекс* является средством, которое обеспечивает быстрый доступ к данным в таблице на основе значений одного или нескольких столбцов. Индекс представляет собой упорядоченный список значений и ссылок на те записи, в которых хранятся эти значения. Чтобы найти нужные записи, СУБД сначала ищет требуемое значение в индексе, а затем по ссылкам быстро отбирает соответствующие записи. Индексы бывают двух типов: простые и составные. *Простые* индексы представляют собой индексы, созданные по одному столбцу. Индекс, построенный по нескольким столбцам, называется *составным.* Примером составного индекса может быть индекс, построенный по столбцам "Фамилия" и "Имя".

Однако применение индексов приносит не только преимущества, но и недостатки. Главным среди них является тот, что при добавлении и удалении записей или при обновлении значений в индексном столбце требуется обновлять индекс, что при большом количестве индексов в таблице может замедлять работу. Поэтому индексы обычно рекомендуется создавать только для тех столбцов таблицы, по которым наиболее часто выполняется поиск записей. Во многих СУБД (например, FoxPro) индексы хранятся в отдельных файлах и являются предметом заботы разработчиков, т. к. при нарушении индекса поиск данных выполняется некорректно. В Microsoft Access индексы хранятся в том же файле базы данных, что и таблицы и другие объекты Access. Индексировать можно любые поля, кроме MEMO-полей, полей типа Гиперссылка и объектов OLE.

Чтобы создать простой индекс, необходимо:

1. Открыть таблицу в режиме Конструктора.
2. Выбрать поле, для которого требуется создать индекс.

Открыть вкладку Общие и выбрать для свойства Индексированное поле значение Да (Допускаются совпадения) или Да (Совпадения не допускаются).

Ключевое поле таблицы автоматически индексируется и свойству Индексированное поле присваивается значение Да (Совпадения не допускаются).

Составной индекс создается в специальном диалоговом окне. Чтобы создать составной индекс, необходимо:

1. Открыть таблицу в режиме Конструктора.
2. На панели инструментов Конструктор таблиц нажать кнопку Индексы.
3. В первой пустой строке поля Индекс ввести имя индекса.

4. В поле поля Имя нажать на стрелку и выбрать первое поле, для которого необходимо создать индекс.
5. В следующей строке поля Имя поля указать второе индексируемое поле. (Для данной строки поле Индекс должно оставаться пустым). Повторите эту операцию для всех полей, которые необходимо включить в индекс. В индексе может быть использовано до 10 полей.

Диалоговое окно Индексы используется также для просмотра, изменения \ удаления существующих индексов. Изменить можно:

- название индекса в поле Индекс;
- поле таблицы, соответствующее данному индексу, выбрав новое поле из списка поле Имя поля ;
- порядок сортировки в поле Порядок сортировки;
- свойства данного индекса в нижней части окна :
    - Ключевое поле определяет, является ли индексированное поле ключевым
    - Уникальный индекс определяет, должно ли быть каждое значение в этом поле уникальным;
    - Пропуск пустых полей определяет, включаются или не включаются в индекс записи с пустым (Null) значением данного поля.

Десятый вопрос: Запросы. Виды запросов. Запросы на выборку к единственной таблице. Создание запросов на выборку (Организация работы с данными в БД. Формирование запросов. (практическое занятие № 16, теоретическая часть)).

Обработка данных, хранящихся в БД, выполняется, в основном, через запросы. Запрос – это требование на получение определенной информации из таблиц БД. С помощью запросов можно просматривать, анализировать и изменять данные, выполнять расчеты и обобщать информацию. Они используются также в качестве источника записей при создании форм и отчетов.

В отличие от фильтров, запросы позволяют отбирать отдельные поля записей из одной или нескольких таблиц, объединяя данные, хранящиеся в разных таблицах. Запрос

можно сохранять в виде объекта БД. Для того чтобы создать или выполнить запрос, нет необходимости открывать таблицы БД.

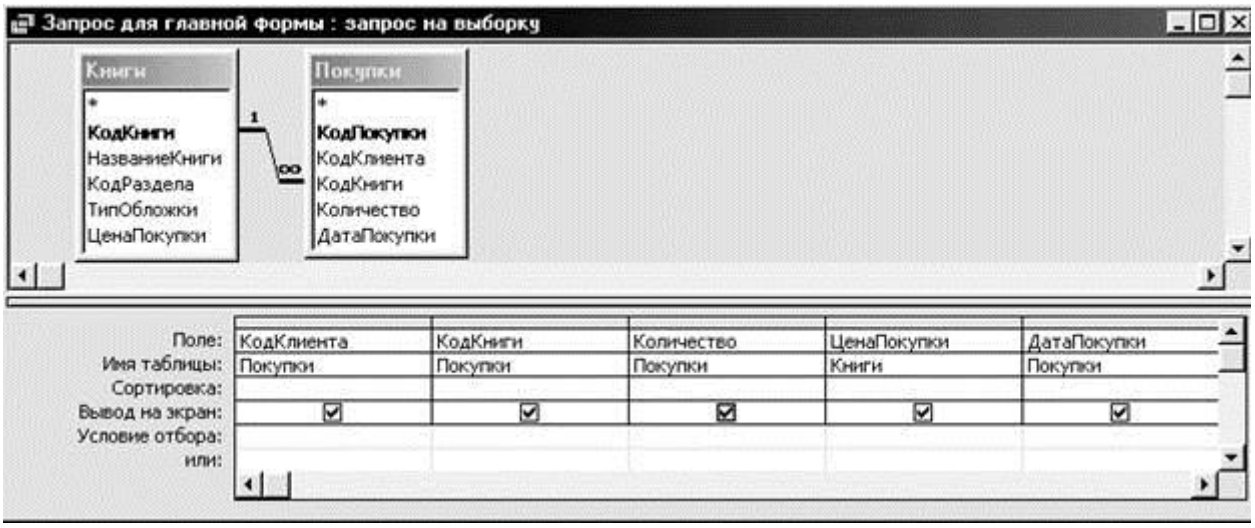
В Microsoft Access можно создавать различные типы запросов: запросы на выборку, запросы с параметрами, перекрестные запросы, запросы на изменение (запросы на создание таблицы, удаление, обновление, добавление записей), запросы SQL (запросы на объединение, запросы к серверу, управляющие запросы, подчиненные запросы).

Наиболее часто используемым запросом является запрос на выборку. После выполнения запроса на выборку Access создает новый набор записей, содержащий отобранные данные. Такой набор физически не существует в БД, но работать с ним можно как с обычной таблицей.

Запросы можно строить с помощью **Мастера запросов** или самостоятельно в режиме Конструктора.

Для создания запроса на выборку без использования Мастера надо в окне БД выбрать объект *Запросы*, нажать кнопку **Создать и** в диалоговом окне **Новый запрос** выбрать команду *Конструктор*. В открывшемся окне Конструктора запросов можно создавать запрос на выборку из одной или нескольких связанных таблиц. В качестве источника данных для запроса можно использовать не только таблицу, но и другой ранее составленный запрос.

Окно Конструктора разделено на две части. В верхней части находятся списки полей таблиц или запросов, на основе которых создается новый запрос. В нижней части располагается бланк запроса, в котором формируется запрос по типу QBE. Каждый столбец бланка соответствует одному полю, используемому в запросе. В первой строке бланка определяются имена полей, которые должны присутствовать в наборе отобранных записей или используются для задания условий отбора. В других строках бланка задаются имена таблиц, порядок сортировки отобранных записей, флажки для вывода полей запроса на экран и условия отбора записей. Поля, для которых снят флажок, будут отсутствовать в отобранном наборе записей. Такие поля (обычно это ключевые поля) по разным причинам часто приходится включать в бланк запроса, но нет необходимости выводить их на экран.



При создании нового запроса в окне Конструктора раскрывается диалоговое окно **Добавление таблицы** с вкладками, позволяющими выбрать объекты, содержащие требуемые данные. Для добавления в запрос каждого из объектов нужно выделить его имя и нажать кнопку **Добавить**. После появления всех необходимых объектов в верхней части окна Конструктора окно **Добавление таблицы** нужно закрыть. При необходимости это окно вновь можно вызвать, щелкнув правой кнопкой мыши в верхней части окна и выбрав из контекстного меню команду *Добавить таблицу*.

Если запрос строится на нескольких таблицах или запросах, то они должны быть связаны. Если таблицы не связаны, то Access рассматривает каждую таблицу как независимый объект и позволяет составлять любую возможную комбинацию из полей в окне Конструктора.

Поля добавляются в бланк запроса перетаскиванием их мышью из списка полей или двойным щелчком мыши. Затем определяются условия отбора, порядок сортировки, создаются вычисляемые поля, устанавливаются необходимые флажки. Созданный запрос необходимо сохранить, определив его имя. Результат выполнения запроса можно просмотреть в режиме Таблицы.

Условие отбора позволяет определить, какие именно записи следует отобрать с помощью запроса. Обычно условие отбора представляет собой конкретное значение того или иного поля. Это значение указывается в строке *Условие отбора* бланка запроса. Символьные значения заключаются в кавычки, Access сам добавляет кавычки к введенному тексту. Календарные даты и время заключаются в символы #. При определении условий отбора можно использовать операторы сравнения =, <, >, <=, >=, <> и логические операторы OR и AND. Значения, введенные в разных строчках бланка запроса, рассматриваются как операнды операции ИЛИ. Значения, введенные в разных столбцах, рассматриваются как операнды логического И.

При определении условия отбора необходимо использовать те значения поля, которые хранятся в таблице, но не подстановочные значения. Пусть, например, необходимо отобрать записи о покупках, сделанных клиентом *РГРТУ*, и в качестве источника

данных используется таблица ПОКУПКИ. Тогда в условии отбора для поля КодКлиента необходимо указать код РГРТА, т. е. цифру 4. Если этот же запрос создать на двух таблицах - ПОКУПКИ и КЛИЕНТЫ, то в качестве условия отбора для поля ИмяКлиента надо ввести слово *РГРТУ*. В том случае, когда условие отбора содержит значки подстановки ? или *, условие (даже число) надо заключать в апострофы.

В запрос можно добавлять ***вычисляемые поля***, содержимое которых является результатом арифметических операций над определенными полями таблиц. Для создания вычисляемого поля надо в свободном столбце бланка запроса в строке *Поле* ввести имена полей, являющихся операндами, и указать оператор. Здесь же нужно задать имя вычисляемого поля. Например, можно составить запрос, формирующий список покупателей, названий купленных ими книг, количество купленных экземпляров каждой из книг, цену книги и стоимость каждой покупки. В бланке такого запроса надо указать поля КодКлиента (таблица ПОКУПКИ), КодКниги (таблица ПОКУПКИ), Количество (таблица ПОКУПКИ), ЦенаПокупки (таблица КНИГИ) и вычисляемое поле

Сумма:[ЦенаПокупки]*[Количество]

Выражение для вычисляемого поля удобнее вводить в *Область ввода*. Для ее вызова надо после щелчка в свободной ячейке нажать **Shift+F2.**

Если в бланк этого запроса добавить поле КодПокупки, установить для него порядок сортировки *По возрастанию* и снять флажок вывода на экран, то отобранные записи будут выводиться в том же порядке, что и в таблице ПОКУПКИ.

В запросе можно предусмотреть подсчет итоговых значений, например, подсчитать общее количество всех книг, купленных каждым из клиентов, и итоговую стоимость всех сделанных им покупок. Такой запрос называется ***Итоговым запросом***. Для составления такого запроса надо нажать кнопку **Групповые операции** в режиме Конструктора запроса. В бланке запроса появится новая строка *Групповая операция,* а в каждом поле - установка *Группировка***.**

В поле, для которого установлена опция *Группировка,* записи с одинаковыми значениями объединятся в группы. При выполнении запроса каждое из значений будет присутствовать на экране единожды, т.е. набор отобранных записей будет содержать по одной строке для каждого уникальногозначения такого поля. В пределах каждой группы над содержимым других полей можно выполнять определенные расчеты с помощью функций. Для этого надо в соответствующих полях заменить установку *Группировка* на конкретную *итоговую функцию*, которую можно выбрать в раскрывающемся списке.

Существует девять функций для выполнения групповых операций. В их числе:

Sum - возвращает сумму всех значений данного поля в каждой группе;

Avg – возвращает среднее арифметическое значений полей в группе;

Min, Max – возвращает наименьшее или наибольшее значение в каждой группе,

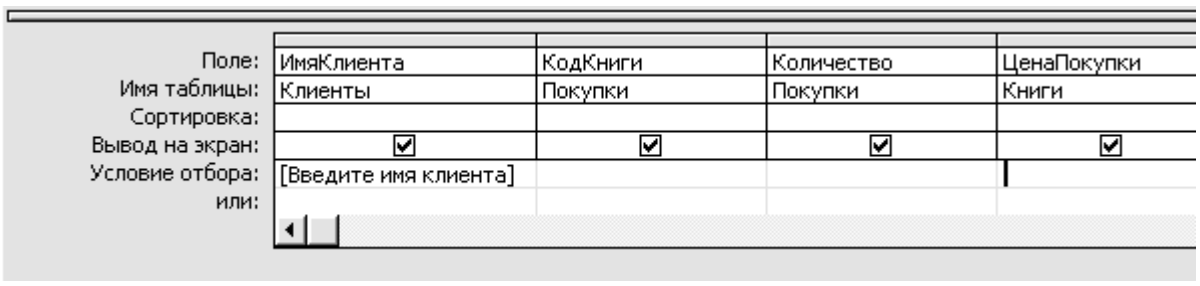
Count - возвращает число записей в группе, для которых значения данного поля отличны от *Null*.

Групповые операции можно выполнять и над вычисляемыми полями.

Так, например, для подведения итогов по объемам покупок, сделанных каждым из клиентов, можно составить итоговый запрос на таблицах ПОКУПКИ и КНИГИ В качестве источника данных для итогового запроса можно использовать также рассмотренный выше запрос с вычисляемым полем. В бланк итогового запроса надо включить поля КодКлиента, Количество из таблицы ПОКУПКИ и создать вычисляемое поле Сумма так, как это описано выше. Для первого из полей надо оставить установку *Группировка*, а для двух других полей эту установку следует заменить итоговой функцией Sum. В результате выполнения запроса сформируется список, содержащий по одной записи для каждого клиента с итоговыми значениями – общим количеством купленных этим клиентом книг и общей стоимостью всех сделанных им покупок.

Полям записей запроса можно задать свойства, отличные от свойств полей таблиц. Например, можно определить новые значения свойства *Подпись*, определяющие имена столбцов выводимой на экран таблицы. Для этого нужно щелкнуть правой кнопкой мыши в любой ячейке соответствующего столбца и из контекстного меню вызвать окно **Свойства поля.** Для вызова этого окна можно воспользоваться также кнопкой **Свойства** на панели инструментов или командой меню **Вид**/ *Свойства*.

В рассмотренных выше запросах условия отбора вводились непосредственно в бланк запроса. Однако удобнее было бы иметь возможность изменять в запросе условия отбора, вводя конкретные значения для поиска перед выполнением запроса. Эту задачу можно решить, создав ***запрос с параметром***.



Чтобы создать запрос с параметром, надо в бланке запроса в строку *Условия отбора* вместо конкретного значения ввести приглашение пользователю определить условия отбора. Строка приглашения должна быть заключена в квадратные скобки. Текст, заключенный в квадратные скобки, Access рассматривает как ***имя параметра***. Перед выполнением запроса выводится диалоговое окно с фразой-приглашением и полем для определения конкретного значения параметра. Вводить значение параметра надо в том виде, в котором это значение хранится в таблице БД. Нельзя вводить подстановочные значения.

В запросе можно задать несколько параметров; при этом имя каждого из них должно быть уникальным и информативным. При выполнении запроса значения параметров вводятся поочередно. Используя имена параметров, связанные операторами отношения и логическими операторами, можно определить сложное условие отбора. В этом случае условие отбора рассматривается как выражение, операндами которого являются имена параметров.

Для параметров запроса можно указать тип данных для проверки введенного значения и предотвращения ошибок ввода. По умолчанию Access назначает параметрам запроса текстовый тип данных. Изменить тип данных параметра можно, выбрав команду **Запрос**/ *Параметры* и определив нужный тип данных в открывшемся окне **Параметры запроса.**

В Microsoft Access можно создавать так называемые ***перекрестные запросы***. В перекрестном запросе отображаются результаты статистических расчетов (суммы, количество записей, средние значения), выполненных по данным из одного поля таблицы. Эти результаты группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй - в верхней строке. Результат выполнения такого запроса напоминает электронную таблицу.

Можно, например, создать перекрестный запрос, показывающий, сколько книг по каждому из разделов куплено каждым клиентом, а также общее количество книг, приобретенных каждым клиентом. В итоговой таблице, сформированной

Перекрестный запрос : перекрестный запрос

Код клиента	Итоги	Здоровье	Романы	Техническая	Художественная	Экономика
ООО Спрут	9		2		5	2
Кашкин и Ко	6		6			
ЧП Витязь	5				3	2
РГРТА	30	4	12	6	8	
НПЗ	7				7	

в ответ на запрос, имена клиентов будут располагаться слева сверху вниз (по строкам), а названия разделов - вверху слева направо (по столбцам). Кроме того, в таблице появится столбец, содержащий суммарное количество покупок для каждого клиента. Для создания такого запроса придется объединить данные из нескольких таблиц, поэтому вначале придется создать запрос, объединяющий данные, а потом создать перекрестный запрос, основанный на объединяющем запросе. Объединяющий запрос должен сформировать таблицу с полями КодКлиента (таблица ПОКУПКИ), Раздел (таблица РАЗДЕЛЫ), Количество (таблица ПОКУПКИ).

Перекрестный запрос проще всего создать при помощи Мастера запросов. Для вызова Мастера надо в окне БД для объекта *Запросы* нажать кнопку **Создать** и выбрать из списка строку *Перекрестный запрос*. На первом шаге работы Мастера надо установить флажок *Запросы* и выбрать имя объединяющего запроса. На следующих шагах следует выбрать в качестве заголовков строк поле КодКлиента, для заголовков столбцов поле

Разделы, в списке функций для поля Количество выбрать *Сумма* и определить имя запроса. Готовый запрос можно просмотреть в режиме таблицы.

Одиннадцатый вопрос: Знакомство с программой ЭТРАН – автоматизированной системой подготовки и оформления перевозочных документов (практическое занятие № 16, теоретическая часть).

ТРАН (электронная транспортная накладная) - это автоматизированная система централизованной подготовки и оформления перевозочных документов.

Целью системы ЭТРАН является переход на использование электронного документооборота для взаимодействия с пользователями услуг железнодорожного транспорта при организации перевозок грузов.

В рамках системы ЭТРАН решены вопросы подготовки и оформления перевозочных документов в системе электронного документооборота. При разработке системы были учтены требования по защите конфиденциальной информации.

Система разработана для автоматизации подготовки документов в изменившихся условиях работы железнодорожного транспорта с ориентацией на современные технические средства информатизации. Внедрение в сферу информатизации железнодорожного транспорта современных банковских технологий и программных средств ERP-систем потребовало замены ранее разработанных и эксплуатируемых до настоящего времени автоматизированных систем в сфере станционных, дорожных и сетевых «бумажных» информационных технологий.

Назначение системы.

При разработке системы ЭТРАН предусматривалось:

- внедрение электронного документооборота с использованием внутренней и внешних сетей передачи данных;
- обеспечение защиты конфиденциальной информации сертифицированными средствами;
- обеспечение режима реального времени: не более 3-5 с на одну технологическую операцию;
- обеспечение однократности ввода информации и минимальное число корректировок;
- обеспечение решения новых задач в системе;
- обеспечение решения действующих задач в новой постановке;
- обеспечение своевременности (по моменту осуществления операций погрузки, выдачи или приема (сдачи) груза на границах) и точности расчетов (качество первичной информации, ведение электронной базы тарифов и централизованные расчеты провозных платежей) за перевозки грузов;
- более глубокая интеграция с автоматизированными системами оперативного управления перевозками (АСУП) и автоматизированной системой управления финансами и ресурсами (ЕК АСУФР);
- включение грузоотправителей и экспедиторов в технологический процесс на основе решений в области электронного документооборота;

- сохранение и развитие на переходный период существующих комплексов по обработке перевозочных документов ЕК ИОДВ, АС ТехПД, АРМ ТВК, и интерфейсов с АСУП, ЕКАСУФР.

Взаимодействие с *другими системами.*

Для осуществления электронизации документооборота железные дороги должны иметь информационную взаимосвязь с грузоотправителем (грузополучателем) и экспедитором на уровне, обеспечивающем ведение электронного документооборота (с возможностью установки разработанного в рамках системы ЭТРАН программного обеспечения АРМ ППД).

Система ЭТРАН взаимодействует с системой финансовых расчетов (ЕК АСУФР). Между системами существует соглашение о разделении зон ответственности по функциям, в частности: функции ЕК АСУФР - открытие и ведение лицевого счета клиентам, экспедиторам, функции ЭТРАН - использование присвоенных кодов лицевых счетов, получение и использование сальдо лицевого счета и дебетование лицевого счета.

Реализовано взаимодействие системы ЭТРАН с системами управления перевозочным процессом - это АСОУП и система линейного района АСУ станций.

Виды услуг клиентам.

Возможности, предоставляемые системой ЭТРАН клиенту, позволяют минимизировать бумажный документооборот, время и сроки оформления перевозок на следующих этапах:

- подача заявки на перевозку;
- получение результата согласования заявки;
- оформление накладной на основе заявки;
- оформление результатов погрузки;
- получение в электронном виде квитанции о приеме груза к перевозке;
- получение информации о нештатных ситуациях.

ЭТРАН предоставляет *смежным системам* следующую информацию:

- результаты погрузки/выгрузки;
- переоформление документов;
- результаты расчетов по перевозкам (АСОУП, ЕК ИОДВ, ЕК ЛСУФР).

Также сохраняется возможность обеспечить информацией организации-экспедиторы и организовывать расчеты с ними.

В систему ЭТРАН попадает вся информация по импортным и транзитным перевозкам с момента пересечения границы, что позволяет взять под контроль эти перевозки и организовать необходимую работу с экспедиторскими организациями:

- рассчитать платежи на предстоящие перевозки, принимаемые из-за границы;
- проверить платежеспособность экспедиторов.

Исходная информация.

Исходная информация (с корешков дорожных ведомостей, дорожных ведомостей, копий вагонных листов и дополнительных финансовых документов о кредитовой и дебетовой составляющих работы дороги) поступает: с железнодорожных станций (АРМ ТВК), АС ТехПД, с пограничных станций стран СНГ (АСУ СПВ), с железнодорожных станций на границах с третьими странами, из банков и финансовой службы дороги. После обработки информация передается на сетевой уровень.

Количество обработанной информации в сутки составляет: 120 тыс. перевозочных документов со средним объемом 12 Мбайт, 20 тыс. дополнительных документов, 15 тыс. запросов к базам данных.

Принципы работы системы ЭТРАН.

Система ЭТРАН создана на новых принципах, реализуется на современной программно-технической платформе в едином информационном пространстве со смежными АСУ.

ЭТРАН представляет собой трехуровневую иерархическую корпоративную систему, состоящую из Центра обработки информации (ЦОИ), вспомогательных (или технологических) центров обработки информации (ВЦОИ) и АРМ грузоотправителей (грузополучателей), работников железнодорожного транспорта различных уровней управления (от линейного до сетевого). ЦОИ реализует технологические информационные процессы (включая электронный оборот перевозочных и других документов) на закрепленных территориях: обслуживание клиентуры и обеспечение функционирования низового уровня сквозной вертикали системы ЭТРАН.

Центры обработки информации предназначены для: обеспечения доступа к информационным и вычислительным ресурсам отрасли со стороны клиентуры и персонала железных дорог; регистрации первичных операций в процессе оформления перевозок; формирования электронных документов и их обработки в масштабе реального времени. Эти функции связаны с необходимостью обеспечить:

- непрерывность выполнения технологических операций и взаимодействия с внешними АСУ;
- последовательность выполнения операций обработки информации в соответствии со сбором заявок;
- осуществление процессов планирования;
- контроль за подготовкой процесса перевозок;
- подготовку перевозочных документов;
- расчет провозных платежей;
- контроль оплаты перевозок и осуществления самой перевозки в соответствии с договорными условиями;
- своевременные расчеты с клиентурой.

Другие составные части системы ЭТРАН могут функционировать также в псевдореальном времени (включая интерактивные режимы работы) или по установленным циклам и периодам (режим регламента).

Автоматизированная система централизованной подготовки и оформления перевозочных документов ЭТРАН

ЭТРАН - автоматизированная система централизованной подготовки и оформления перевозочных документов. Система впервые включает клиента (грузоотправителя, грузополучателя, экспедитора) в технологический цикл приема заявок и оформления перевозок, обеспечивая ему возможность оформления заявки на перевозку, подготовки электронной накладной, получения итоговых документов, получения результатов расчетов провозной платы по перевозкам и отслеживания хода перевозок грузов со своего рабочего места. Также, клиенту предоставляется возможность получения информации обо всех грузах, отправленных в его адрес.

В основу решения положены требования, установленные Уставом железнодорожного транспорта Российской Федерации и Правилами приема заявок на перевозку грузов железнодорожным транспортом.

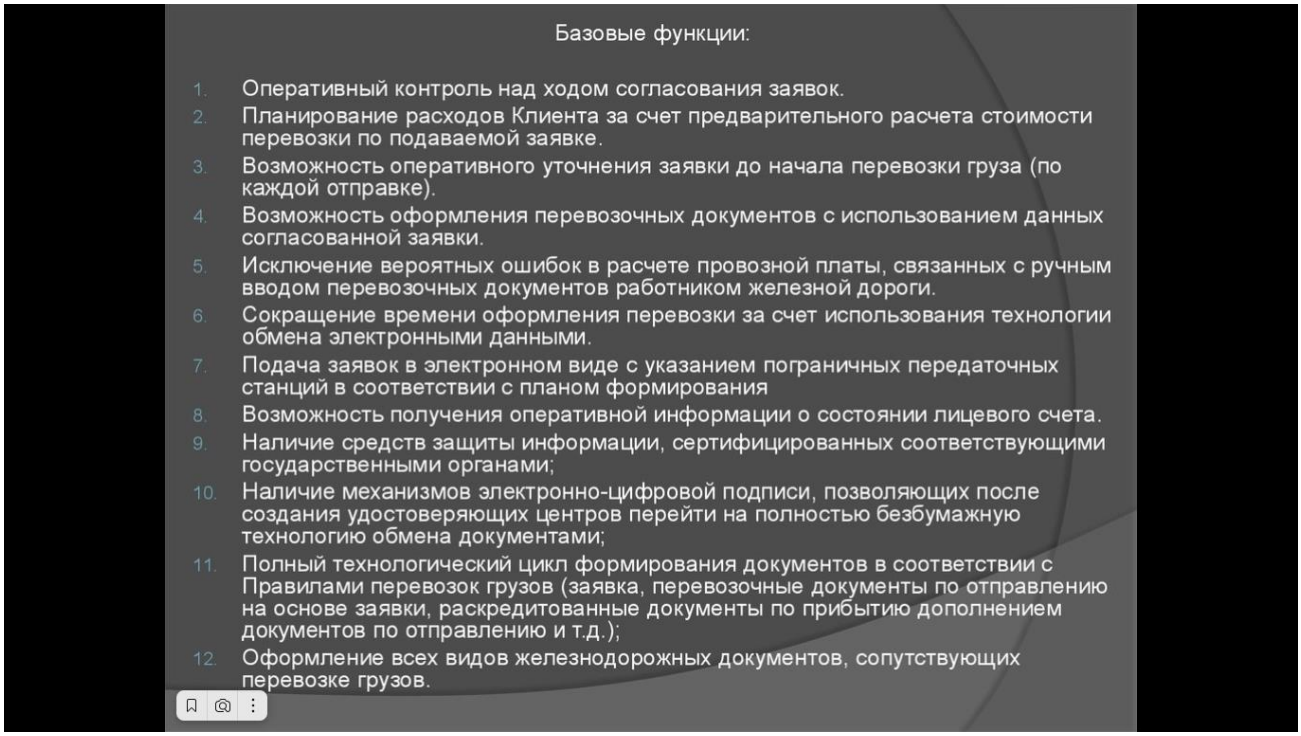
Изначально созданная как прикладной инструмент, система ЭТРАН на седьмом году своего существования перешла к функционированию в качестве полноценной учетной системы, способной контролировать бизнес-процессы грузоперевозок.

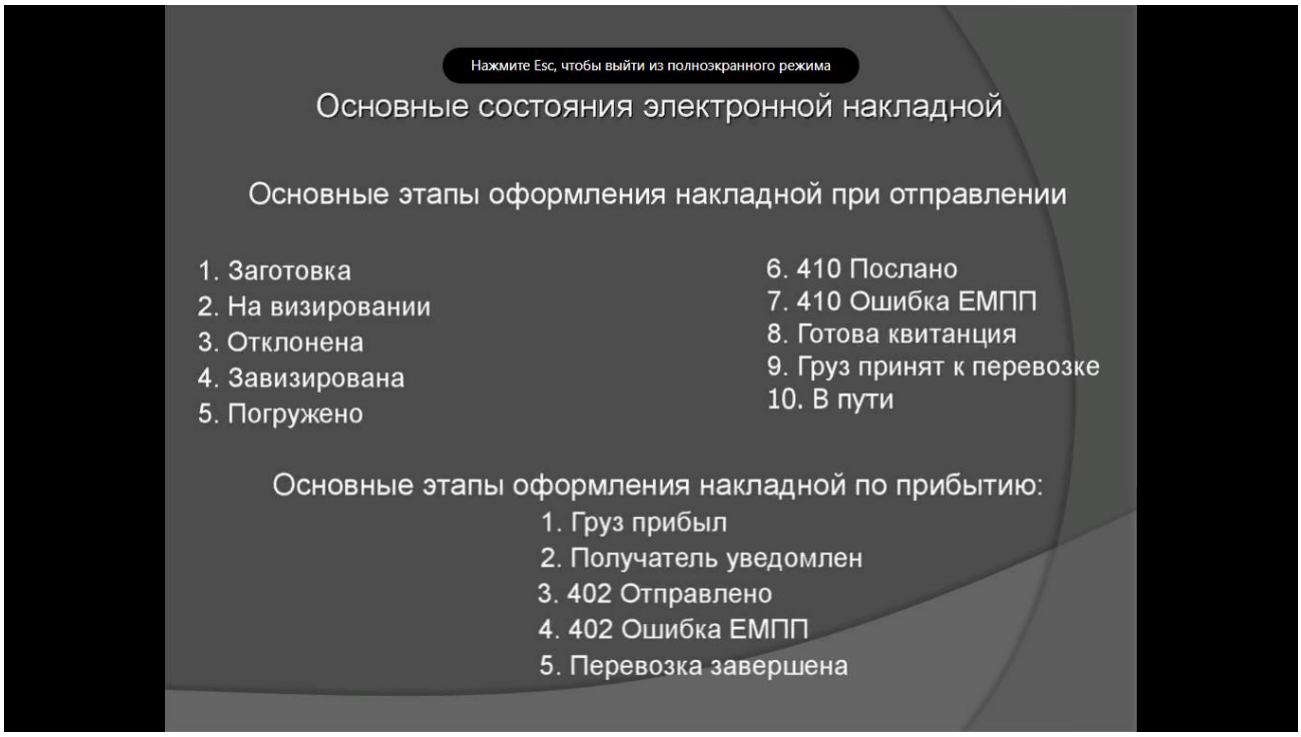
В настоящее время ЭТРАН охватывает 100% железнодорожных грузоперевозок на территории Российской Федерации.

17 сентября 2012 года система ЭТРАН ОАО «РЖД», разработанная специалистами ЗАО «ИнтэлЛекс», отметила свой первый юбилей. В 2012 году система развивалась в нескольких направлениях: от простого учета услуг к большому количеству управляющих решений; от внутреннего обмена данными к трансграничному.

За счет реализации проекта трансграничного обмена данными система ЭТРАН перешагнула границы России: уже налажен двухсторонний обмен электронными накладными на порожние вагоны между Россией и Финляндией, приходят вагоны с электронными накладными из Латвии. С начала 2012 года ведутся подготовительные работы по организации электронного документооборота с республикой Беларусь. В ближайшей перспективе эксперимент по обмену электронными накладными может быть распространен на перевозки в направлении остальных стран Балтии, а также Казахстана и Киргизии.

Создание функций, направленных на повышение качества планирования перевозок в целом и перемещения порожних вагонов в частности, также стали важным направлением деятельности разработчиков системы в этом году. В настоящий момент для решения маркетинговых задач реализованы контроли различного уровня, например, касающиеся планирования перевозок, обработки уведомлений о перемещении порожних вагонов, работы с доверенностями на право отправления порожнего вагона.





Двенадцатый вопрос: Определение результатов выполнения запросов с применением аппарата алгебры логики.

Язык SQL, предназначенный для взаимодействия с базами данных, появился в середине 1970-х гг. (первые публикации датируются 1974 г.) и был разработан в компании IBM в рамках проекта экспериментальной реляционной системы управления базами данных. Исходное название языка – SEQUEL (Structured English Query Language) – только частично отражало суть этого языка. Первоначально, сразу после его изобретения и в первичный период эксплуатации языка SQL, его название являлось аббревиатурой от словосочетания Structured Query Language, что переводится как «Язык

структурированных запросов». Конечно, язык был ориентирован главным образом на удобную и понятную пользователям формулировку запросов к реляционным базам данных. Но, в действительности, он почти с самого начала являлся полным языком баз данных, обеспечивающим, помимо средств формулирования запросов и манипулирования базами данных, следующие возможности:

1) средства определения и манипулирования схемой базы данных;

2) средства определения ограничений целостности и триггеров (о которых будет упомянуто позднее);

3) средства определения представлений баз данных;

4) средства определения структур физического уровня, поддерживающих эффективное выполнение запросов;

5) средства авторизации доступа к отношениям и их полям.

В языке отсутствовали средства явной синхронизации доступа к объектам баз данных со стороны параллельно выполняемых транзакций: с самого начала предполагалось, что необходимую синхронизацию неявно выполняет система управления базами данных.

В настоящее время SQL – это уже не аббревиатура, а название самостоятельного языка.

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

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

С течением времени и работы над языком структурированных запросов удалось достигнуть стандарта четкой стандартизации синтаксиса и семантики операторов выборки данных, манипулирования данными и фиксации средств ограничения целостности баз данных. Были специфицированы средства определения первичного и внешних ключей отношений и так называемых проверочных ограничений целостности,

которые представляют собой подмножество немедленно проверяемых ограничений целостности SQL. Средства определения внешних ключей позволяют легко формулировать требования так называемой ссылочной целостности баз данных (о которой мы поговорим позднее). Это распространенное в реляционных базах данных требование можно было сформулировать и на основе общего механизма ограничений целостности SQL, но формулировка на основе понятия внешнего ключа более проста и понятна.

Итак, с учетом всего этого в настоящее время язык структурированных запросов – это название не просто одного языка, а название целого класса языков, поскольку, несмотря на имеющиеся стандарты, в различных системах управления базами данных реализуются различные диалекты языка структурированных запросов, имеющие, разумеется, одну общую основу.

1. Оператор Select – базовый оператор языка структурированных запросов

Центральное место в языке структурированных запросов SQL занимает оператор Select, с помощью которого реализуется самая востребованная операция при работе с базами данных – запросы.

Оператор Select осуществляет вычисление выражений как реляционной, так и псевдореляционной алгебры. В данном курсе мы рассмотрим реализацию только уже пройденных нами унарных и бинарных операций реляционной алгебры, а также осуществление запросов, с помощью так называемых подзапросов.

Кстати, необходимо заметить, что в случае работы с операциями реляционной алгебры в результирующих отношениях могут появляться дублирующие кортежи. В правилах языка структурированных запросов нет строгого запрещения на присутствие повторяющихся строк в отношениях (в отличие от обычной реляционной алгебры), поэтому исключать дубликаты из результата не обязательно.

Итак, рассмотрим базовую структуру оператора Select. Она достаточно проста и включает в себя следующие стандартные обязательные фразы:

Select …

From …

Where … ;

На месте многоточия в каждой строчке должны стоять отношения, атрибуты и условия конкретной базы данных и задания к ней. В самом общем случае базовая структура Select должна выглядеть следующим образом:

Select *выбрать такие-то атрибуты*

From *из таких-то отношений*

Where *с такими-то условиями выборки кортежей*

Таким образом, выбираем мы атрибуты из схемы отношений (заголовки некоторых столбцов), при этом указывая, из каких отношений (а их, как видно, может быть несколько) мы производим нашу выборку и, наконец, на основании каких условий мы останавливаем свой выбор на тех или иных кортежах.

Важно заметить, что ссылки на атрибуты происходят с помощью их имен.

Таким образом, получается следующий **алгоритм работы** этого базового оператора Select:

1) запоминаются условия выборки кортежей из отношения;

2) проверяется, какие кортежи удовлетворяют указанным свойствам. Такие кортежи запоминаются;

3) на выход выводятся перечисленные в первой строчке базовой структуры оператора Select атрибуты со своими значениями. (Если говорить о табличной форме записи отношения, то выведутся те столбцы таблицы, заголовки которых были перечислены как необходимые атрибуты; разумеется, столбцы выведутся не полностью, в каждом из них останутся только те кортежи, которые удовлетворили названным условиям.)

Рассмотрим пример.

Пусть нам дано следующее отношение r_1, как фрагмент некой базы данных книжного магазина:

Код книги	Название книги	Автор книги	Цена книги
1258963	Холодные берега	С. Лукьяненко	186,9
1236954	Мобильник	С. Кинг	201,4

Пусть также нам дано следующее выражение с оператором Select:

Select *Название книги, Автор книги*

From r_1

Where *Цена книги > 200;*

Результатом этого оператора будет следующий фрагмент кортежа:

(Мобильник, С. Кинг).

(В дальнейшем мы подвергнем рассмотрению множество примеров реализации запросов с использованием этой базовой структуры и ее применение изучим очень подробно.)

2. Унарные операции на языке структурированных запросов

В этом параграфе мы рассмотрим, как реализуются на языке структурированных запросов с помощью оператора Select уже знакомые нам унарные операции выборки, проекции и переименования.

Важно заметить, что если раньше мы могли работать только с отдельными операциями, то даже один оператор Select в общем случае позволяет определить целое выражение реляционной алгебры, а не какую-то одну операцию.

Итак, перейдем непосредственно к анализу представления унарных операций на языке структурированных запросов.

1. Операция выборки.

Операция выборки на языке SQL реализуется оператором Select следующего вида:

Select *все атрибуты*

From *имя отношения*

Where *условие выборки*;

Здесь вместо того, чтобы писать «все атрибуты», можно использовать значок «*». В теории языка структурированных запросов этот значок означает выбор всех атрибутов из схемы отношения.

Условие выборки здесь (и во всех остальных реализациях операций) записывается в виде логического выражения со стандартными связками not (не), and (и), or (или). На атрибуты отношения ссылаемся посредством их имен.

Рассмотрим пример. Определим следующую схему отношения:

Успеваемость (***№ зачетной книжки, Семестр, Код предмета***, Оценка, Дата);

Здесь, как уже упоминалось ранее, подчеркнутые атрибуты образуют ключ отношения.

Составим оператор Select следующего вида, реализующий унарную операцию выборки:

Select *

From *Успеваемость*

Where *№ зачетной книжки = 100 and Семестр = 6*;

Понятно, что в результат этого оператора машина выведет успеваемость студента с номером зачетки сто за шестой семестр.

2. Операция проекции.

Операция проекции на языке структурированных запросов реализуется даже проще, чем операция выборки. Напомним, что при применении операции проекции выбираются не строки (как при применении операции выборки), а столбцы. Поэтому достаточно перечислить заголовки нужных столбцов (т. е. имена атрибутов), без указания каких-либо посторонних условий. Итого, получаем оператор следующего вида:

Select *список имен атрибутов*

From *имя отношения*;

После применения этого оператора машина выдаст те столбцы таблицы-отношения, имена которых были указаны в первой строчке этого оператора Select.

Как мы уже упоминали ранее, повторяющиеся строки и столбцы исключать из результирующего отношения не обязательно. Но если в заказе или в задании требуется обязательно элиминировать дубликаты, следует использовать специальную опцию языка структурированных запросов – **distinct**. Эта опция задает автоматическое исключение дубликатов кортежей из отношения. С применением этой опции оператор Select будет выглядеть следующим образом:

Select *distinct список имен атрибутов*

From *имя отношения*;

В языке SQL существует специальное обозначение для необязательных элементов выражений – квадратные скобки […]. Поэтому в самом общем виде операция проекции будет выглядеть следующим образом:

Select [*distinct*] *список имен атрибутов*

From *имя отношения*;

Однако если результат применения операции гарантированно не содержит дубликатов или же дубликаты все-таки допустимы, то опцию **distinct** лучше не указывать, чтобы не загромождать запись, т. е. из соображений производительности работы оператора.

Рассмотрим пример, иллюстрирующий возможность стопроцентной уверенности в отсутствии дубликатов. Пусть дана уже известная нам схема отношений:

Успеваемость (***№ зачетной книжки***, ***Семестр***, ***Код предмета***, Оценка, Дата).

Пусть дан следующий оператор Select:

Select *№ зачетной книжки, Семестр, Код предмета*

From *Успеваемость*;

Здесь, как легко видеть, три возвращающихся оператором атрибута образуют ключ отношения. Именно поэтому опция **distinct** становится излишней, ведь дубликатов гарантированно не будет. Это следует из требования, накладываемого на ключи, называемого ограничением уникальности. Подробнее это свойство мы рассмотрим дальше, но если атрибут ключевой, то дубликатов в нем нет.

3. Операция переименования.

Операция переименования атрибутов на языке структурированных запросов осуществляется довольно просто. А именно воплощается в действительность следующим алгоритмом:

1) в списке имен атрибутов фразы Select перечисляются те атрибуты, которые необходимо переименовать;

2) к каждому указанному атрибуту добавляется специальное ключевое слово as;

3) после каждого вхождения слова as указывается то имя соответствующего атрибута, на которое необходимо поменять имя исходное.

Таким образом, с учетом всего вышесказанного, оператор, соответствующий операции переименования атрибутов, будет выглядеть следующим образом:

Select *имя атрибута 1 as новое имя атрибута 1, …*

From *имя отношения;*

Покажем работу этого оператора на примере. Пусть дана уже знакомая нам схема отношения:

Успеваемость (***№ зачетной книжки***, ***Семестр***, ***Код предмета***,Оценка, Дата);

Пусть у нас имеется заказ поменять имена некоторых атрибутов, а именно вместо «№ зачетной книжки» должно стоять «№ зачетки» и вместо «Оценка» – «Балл».

Запишем, как будет выглядеть оператор Select, реализующий эту операцию переименования:

Select *зачетной книжки as № зачетки, Семестр, Код предмета, Оценка as Балл, Дата*

From *Успеваемость;*

Таким образом, результатом применения этого оператора будет новая схема отношения, отличающаяся от исходной схемы отношения «Успеваемость» именами двух атрибутов.

3. Бинарные операции на языке структурированных запросов

Как и унарные операции, операции бинарные также имеют свою реализацию на языке структурированных запросов или SQL. Итак, рассмотрим осуществление на этом языке уже пройденных нами бинарных операций, а именно – операций объединения, пересечения, разности, декартового произведения, естественного соединения, внутреннего и левого, правого, полного внешнего соединения.

1. Операция объединения.

Для того чтобы реализовать операцию объединения двух отношений приходится использовать одновременно два оператора Select, каждый из которых соответствует какому-то одному из исходных отношений-операндов. И к этим двум базовым операторам Select необходимо применить специальную операцию **Union**. Учитывая все вышесказанное, запишем, как же операция объединения будет выглядеть с использованием семантики языка структурированных запросов:

Select *список имен атрибутов отношения 1*

From *имя отношения 1*

Union

Select *список имен атрибутов отношения 2*

From *имя отношения 2;*

Важно заметить, что списки имен атрибутов двух объединяемых отношений должны ссылаться на атрибуты совместимых типов и быть перечислены в согласованном порядке. Если это требование не соблюдать, ваш запрос не сможет быть выполнен, и компьютер выдаст сообщение об ошибке.

Но, что интересно отметить, сами имена атрибутов в этих отношениях могут быть различными. В таком случае результирующему отношению приписываются имена атрибутов, указанные в первом операторе Select.

Также необходимо знать, что использование операции Union предполагает автоматическое исключение из результирующего отношения всех дубликатов кортежей. Поэтому, если вам нужно, чтобы все повторяющиеся строки в конечном результате сохранились, вместо операции Union следует применять модификацию этой операции – операцию **Union All**. В таком случае операция объединения двух отношений будет выглядеть следующим образом:

Select *список имен атрибутов отношения 1*

From *имя отношения 1*

Union All

Select *список имен атрибутов отношения 2*

From *имя отношения 2*;

В этом случае из результирующего отношения дубликаты кортежей удаляться не будут.

Используя уже упоминавшееся ранее обозначение для необязательных элементов и опций в операторах Select, запишем самый общий вид операции объединения двух отношений на языке структурированных запросов:

Select *список имен атрибутов отношения 1*

From *имя отношения 1*

Union [All]

Select *список имен атрибутов отношения 2*

From *имя отношения 2*;

2. Операция пересечения.

Операция пересечения и операция разности двух отношений на языке структурированных запросов реализуются похожим образом (мы рассматриваем наиболее простой способ представления, так как, чем проще метод, тем он экономичнее, актуальнее и, следовательно, наиболее востребован). Итак, мы подвергнем разбору способ реализации операции пересечения с использованием **ключей.**

Этот способ предполагает участие двух конструкций Select, но они не равноправны (как в представлении операции объединения), одна из них является как бы «подконструкцией», «подциклом». Такой оператор обычно называют **подзапросом.**

Итак, пусть у нас имеются две схемы отношений (R_1 и R_2), приблизительно определенные следующим образом:

R_1 (ключ, …) и

R_2 (ключ, …);

Воспользуемся также при записи этой операции специальной опцией **in**, что буквально означает «в» или (как в данном конкретном случае) «содержится в».

Итак, с учетом всего вышесказанного, операция пересечения двух отношений с помощью языка структурированных запросов запишется следующим образом:

Select *

From R_1

Where *ключ* in

(**Select** *ключ* **From** $\mathbf{R_2}$);

Таким образом, мы видим, что подзапросом в данном случае будет являться оператор в круглых скобках. Этот подзапрос в нашем случае возвращает список значений ключа отношения R_2. И, как следует из нашей записи операторов, из анализа условия выборки, в результирующее отношение попадут только те кортежи отношения R_1, ключ которых содержится в списке ключей отношения R_2. То есть, в итоговом отношении, если вспомнить определение пересечения двух отношений, останутся лишь те кортежи, которые принадлежат обоим отношениям.

3. Операция разности.

Как уже было сказано ранее, унарная операция разности двух отношений реализуется аналогично операции пересечения. Здесь также, кроме главного запроса с оператором Select, используется второй, вспомогательный запрос – так называемый подзапрос.

Но в отличие от воплощения в жизнь предыдущей операции, при реализации операции разности необходимо использовать другое ключевое слово, а именно **not in**, что в дословном переводе означает «не в» или (как уместно перевести в нашем рассматриваемом случае) – «не содержится в».

Итак, пусть, как и в предыдущем примере, у нас имеются две схемы отношений (R_1 и R_2), приблизительно заданные:

R_1 (ключ, …) и

R_2 (ключ, …);

Как видим, среди атрибутов этих отношений снова заданы ключевые атрибуты.

Таким образом, получаем следующий вид для представления в языке структурированных запросов операции разности:

Select *

From R_1

Where ключ **not in**

(**Select** ключ **From** R_2);

Таким образом, в результирующее отношение выбираются только те кортежи отношения R_1, ключ которых не содержится в списке ключей отношения R_2. Если рассматривать запись буквально, то действительно получается, что из отношения R_1 «вычли» отношение R_2. Отсюда делаем вывод, что условие выборки в этом операторе записано верно (ведь определение разности двух отношений выполняется) и использование ключей, как и в случае реализации операции пересечения, полностью оправдано.

Два случая применения «метода ключей», которые мы рассмотрели, являются самыми распространенными. На этом изучение использования ключей в составлении операторов, представляющих отношения, завершим. Все оставшиеся бинарные операции реляционной алгебры записываются иными способами.

4. Операция декартова произведения.

Как мы помним из предыдущих лекций, декартово произведение двух отношений-операндов составляется как набор всех возможных пар именованных значений кортежей на атрибутах. Поэтому на языке структурированных запросов операция декартова произведения реализовывается при помощи перекрестного соединения, обозначаемого ключевым словом **cross join**, что буквально и переводится «перекрестное объединение» или «перекрестное соединение».

Оператор Select в конструкции, представляющей операцию декартова произведения на языке структурированных запросов, присутствует только один и имеет следующий вид:

Select *

From R_1 **cross join** R_2

Здесь R_1 и R_2 – имена исходных отношений-операндов. Опция **cross join** обеспечивает, что в результирующее отношение запишутся все атрибуты (все, потому что в первой строчке оператора поставлен значок «*»), соответствующие всем парам кортежей отношений R_1 и R_2.

Очень важно помнить одну особенность воплощения в жизнь операции декартова произведения. Эта особенность является следствием определения бинарной операции декартова произведения. Напомним его:

$r_4(S_4) = r_1(S_1) ? r_2(S_2) = \{t(S_1 ? S_2) \mid t[S_1] ? r_1 \,\&\, t(S_2) ? r_2\}, S_1 ? S_2 = ?;$

Как видно из приведенного определения, пары кортежей образуются при обязательно непересекающихся схемах отношений. Поэтому и при работе на языке структурированных запросов SQL непременно оговаривается, что исходные отношения-операнды не должны иметь совпадающих имен атрибутов. Но если эти отношения все же имеют одинаковые имена, сложившуюся ситуацию можно легко

разрешить с помощью операции переименования атрибутов, т. е. в подобных случаях необходимо просто использовать опцию **as**, о которой упоминалось ранее.

Рассмотрим пример, в котором нужно найти декартово произведение двух отношений, имеющих некоторые имена своих атрибутов совпадающими. Итак, пусть даны следующие отношения:

R_1 (A, B),

R_2 (B, C);

Мы видим, что атрибуты R_1.B и R_2.B имеют одинаковые имена. С учетом этого оператор Select, реализующий на языке структурированных запросов эту операцию декартова произведения, будет выглядеть следующим образом:

Select A, R_1.B **as** B1, R_2.B **as** B2, C

From R_1 **cross join** R_2;

Таким образом, с использованием опции переименования as, у машины не возникнет «вопросов», по поводу совпадающих имен двух исходных отношений-операндов.

5. Операции внутреннего соединения.

На первый взгляд может показаться странным, что мы рассматриваем операцию внутреннего соединения раньше операции естественного соединения, ведь, когда мы проходили бинарные операции, все было наоборот. Но анализируя выражение операций на языке структурированных запросов, можно прийти к выводу, что операция естественного соединения является частным случаем операции внутреннего соединения. Именно поэтому рационально рассмотреть эти операции как раз в таком порядке.

Итак, для начала вспомним определение операции внутреннего соединения, которое мы проходили раньше:

$r_1(S_1)$? $_P$ $r_2(S_2)$ = ? <P> (r_1 ? r_2), S_1 **?** S_2 = ?.

Для нас в этом определении особенно важно то, что рассматриваемые схемы отношений-операндов S_1 и S_2 не должны пересекаться.

Для реализации операции внутреннего соединения в языке структурированных запросов существует специальная опция **inner join**, которая и переводится с английского буквально «внутреннее объединения» или «внутреннее соединение».

Оператор Select в случае осуществления операции внутреннего соединения будет выглядеть следующим образом:

Select *

From R_1 **inner join** R_2;

Здесь, как и раньше, R_1 и R_2 – имена исходных отношений-операндов.

При реализации этой операции нельзя допускать пересечения схем отношений-операндов.

6. Операция естественного соединения.

Как мы уже говорили, операция естественного соединения является частным случаем операции внутреннего соединения. Почему? Да потому что при действии естественного соединения кортежи исходных отношений-операндов соединяются по особому условию. А именно по условию равенства кортежей на пересечении отношений-операндов, тогда как при действии операции внутреннего соединения такой ситуации допускать было бы нельзя.

Так как рассматриваемая нами операция естественного соединения является частным случаем операции внутреннего соединения, для ее реализации используется та же опция, что и для предыдущей рассмотренной операции, т. е. опция **inner join**. Но поскольку при составлении оператора Select для операции естественного соединения необходимо еще учесть условие равенства кортежей исходных отношений-операндов на пересечении их схем, то дополнительно к означенной опции применяется ключевое слово **on**. В переводе с английского, это буквально означает «на», а применительно к нашему смыслу, можно перевести как «при условии».

Общий вид оператора Select для выполнения операции естественного соединения следующий:

Select *

From *имя отношения 1* **inner join** *имя отношения 2*

on *условие равенства кортежей*;

Рассмотрим пример.

Пусть даны два отношения:

R_1 (A, B, C),

R_2 (B, C, D);

Операцию естественного соединения этих отношений можно реализовать с помощью следующего оператора:

Select *A*, R_1.*B*, R_1.*C*, *D*

From R_1 **inner join** R_2

on $R_1.B = R_2.B$ and $R_1.C = R_2.C$

В итоге этой операции в результат выведутся атрибуты, указанные в первой строке оператора Select, соответствующие кортежам, равным на указанном пересечении.

Следует заметить, что здесь мы обращаемся к общим атрибутам B и C не просто по именам. Это необходимо делать не по той причине, что и в случае реализации операции декартова произведения, а потому, что в противном случае будет не ясно, к какому отношению они относятся.

Интересно, что использованная формулировка условия соединения (R_1.B = R_2.B and R_1.C = R_2.C) предполагает, что общие атрибуты соединяемых отношений Null-значений не допускают. Это изначально встроено в систему языка структурированных запросов.

7. Операция левого внешнего соединения.

Выражение на языке структурированных запросов SQL операции левого внешнего соединения получается из реализации операции естественного соединения заменой ключевого слова **inner** на ключевое слово **left outer**.

Таким образом, на языке структурированных запросов эта операция запишется следующим образом:

Select *

From *имя отношения* 1 **left outer join** *имя отношения* 2

on *условие равенства кортежей*;

8. Операция правого внешнего соединения.

Выражение для операции правого внешнего соединения на языке структурированных запросов получается из осуществления операции естественного соединения заменой ключевого слова **inner** на ключевое слово **right outer**.

Итак, получаем, что на языке структурированных запросов SQL операция правого внешнего соединения запишется следующим образом:

Select *

From *имя отношения* 1 **right outer join** *имя отношения* 2

on *условие равенства кортежей*;

9. Операция полного внешнего соединения.

Выражение на языке структурированных запросов операции полного внешнего соединения получается, как и в двух предыдущих случаях, из выражения для операции естественного соединения путем замены ключевого слова **inner** на ключевое слово **full outer**.

Таким образом, на языке структурированных запросов эта операция запишется так:

Select *

From *имя отношения 1* **full outer join** *имя отношения 2*

on *условие равенства кортежей*;

Очень удобно, что в семантику языка структурированных запросов SQL изначально встроены эти опции, ведь иначе каждому программисту приходилось бы выводить их самостоятельно и вводить в каждую новую базу данных.

4. Использование подзапросов

Как можно было понять из пройденного материала, понятие «подзапрос» в языке структурированных запросов является понятием базовым и довольно широко применимым (иногда, кстати, их еще называют SQL-запросами. Действительно, практика программирования и работы с базами данных показывает, что составление системы подзапросов для решения различных сопутствующих задач – деятельность гораздо более благодарная по сравнению с какими-то другими приемами работы со структурированной информацией. Поэтому, рассмотрим пример для лучшего понимания действий с подзапросами, их составлением и использованием.

Пусть имеется следующий фрагмент некой базы данных, которая вполне может использоваться в каком-либо учебном заведении:

Предметы (***Код предмета***, Имя предмета);

Студенты (***№ зачетной книжки***, Фамилия, Имя, Отчество);

Сессия (***Код предмета, № зачетной книжки***, Оценка);

Сформулируем SQL-запрос, возвращающий ведомость с указанием номера зачетной книжки, фамилии и инициалов студента и оценки для предмета с наименованием «Базы данных». Такую информацию в университетах необходимо получать всегда и своевременно, поэтому приведенный далее запрос является едва ли не самой востребованной единицей программирования с использованием таких баз данных.

Для удобства работы, дополнительно предположим, что атрибуты «Фамилия», «Имя» и «Отчество» не допускают Null-значений и не являются пустыми. Это требование

вполне объяснимо и закономерно, ведь в базу данных любого учебного заведения первыми из данных на нового ученика вводятся именно данные о его фамилии, имени и отчестве. И само собой разумеется, что не может быть записи в подобной базе данных, в которой присутствуют данные на ученика, но при этом неизвестно его имя.

Заметим, что атрибут «Имя предмета» схемы отношения «Предметы» является ключом, поэтому, как следует из определения (подробнее об этом будет сказано дальше), все наименования предметов являются уникальными. Это тоже понятно и без пояснения представления ключа, ведь все преподающиеся в учебном заведении предметы должны иметь и имеют различные имена.

Теперь, прежде чем мы приступим к составлению текста самого оператора, введем в рассмотрение две функции, которые нам пригодятся по мере нашей деятельности.

Во-первых, нам будет полезна функция **Trim**, записывается Trim (*«строка»*), т. е. аргументом этой функции является строка. Что делает эта функция? Они возвращает сам аргумент без пробелов, стоящих в начале и в конце этой строки, т. е., эту функцию применяют, например, в случаях: Trim («Богучарников») или Trim («Максиме-енко»), когда после или до аргумента стоят по несколько лишних пробелов.

А во-вторых, необходимо также рассмотреть функцию Left, которая записывается Left (строка, число), т. е. функцию от уже двух аргументов, одним из которых является, как и раньше, строка. Второй ее аргумент – число, оно показывает, сколько символов из левой части строки следует вывести в результат.

Например, результатом операции:

Left («Михаил, 1») + «.» + Left («Зиновьевич, 1»)

будут инициалы «М. З.». Именно для выведения инициалов студентов мы и будем использовать эту функцию в нашем запросе.

Итак, приступим к составлению искомого запроса.

Для начала составим небольшой вспомогательный запрос, который потом используем в основном, главном запросе:

Select № *зачетной книжки, Оценка*

From *Сессия*

Where *Код предмета* = (**Select** *Код предмета*

From *Предметы*

Where *Имя предмета = «Базы данных»*)

as *«Оценки „Базы данных“*;

Применение здесь опции as означает, что мы присвоили этому запросу псевдоним *«Оценки „Базы данных“*. Сделали мы это для удобства дальнейшей работы с этим запросом.

Далее, в этом запросе подзапрос:

Select *Код предмета*

From *Предметы*

Where *Имя предмета = «Базы данных»*;

позволяет выделить из отношения «Сессия» те кортежи, которые относятся к рассматриваемому предмету, т. е. к базам данных.

Интересно, что этот внутренний подзапрос может возвращать не более одного значения, так как атрибут «Имя предмета» является ключом отношения «Предметы», т. е. все его значения уникальны.

А весь запрос *«Оценки „Базы данных“* позволяет выделить из отношения «Сессия» данные о тех студентах (их номера зачетных книжек и оценки), которые удовлетворяют условию, оговоренному в подзапросе, т. е. информацию о предмете под названием «База данных».

Теперь составим основной запрос, используя уже полученные результаты.

Select *Студенты. № зачетной книжки*,

Trim (*Фамилия*) + « » + **Left** (*Имя*, 1) + «.» + **Left** (*Отчество*, 1) + «.»**as** *ФИО*, *Оценки «Базы данных»*. *Оценка*

From *Студенты* **inner join**

(

Select *№ зачетной книжки, Оценка*

From *Сессия*

Where *Код предмета* = (**Select** *Код предмета*

From *Предметы*

Where *Имя предмета = «Базы данных»*)

) as *«Оценки „Базы данных“.*

on *Студенты. № зачетной книжки = Оценки «Базы данных». № зачетной книжки.*

Итак, сначала мы перечисляем атрибуты, которые будет необходимо вывести, после окончания работы запроса. Необходимо упомянуть, что атрибут «№ зачетной книжки» из отношения Студенты, оттуда же – атрибуты «Фамилия», «Имя» и «Отчество». Правда, два последних атрибута выводим не полностью, а только первые буквы. Также мы упоминаем атрибут «Оценка» из запроса *Оценки «Базы данных*, которое ввели раньше.

Выбираем мы все эти атрибуты из внутреннего соединения отношения «Студенты» и запроса *«Оценки „Базы данных“*. Это внутреннее соединение, как мы можем видеть, берется нами по условию равенства номеров зачетной книжки. В результате этой операции внутреннего соединения, к отношению «Студенты» добавляются оценки.

Надо заметить, что так как атрибуты «Фамилия», «Имя» и «Отчество» по условию не допускают Null-значений и не являются пустыми, то формула вычисления, возвращающая атрибут «ФИО» (**Trim** (Фамилия) + « » + **Left** (Имя, 1) + «.» + **Left** (Отчество, 1) + «.»**as** ФИО), соответственно не требует дополнительных проверок, упрощается.