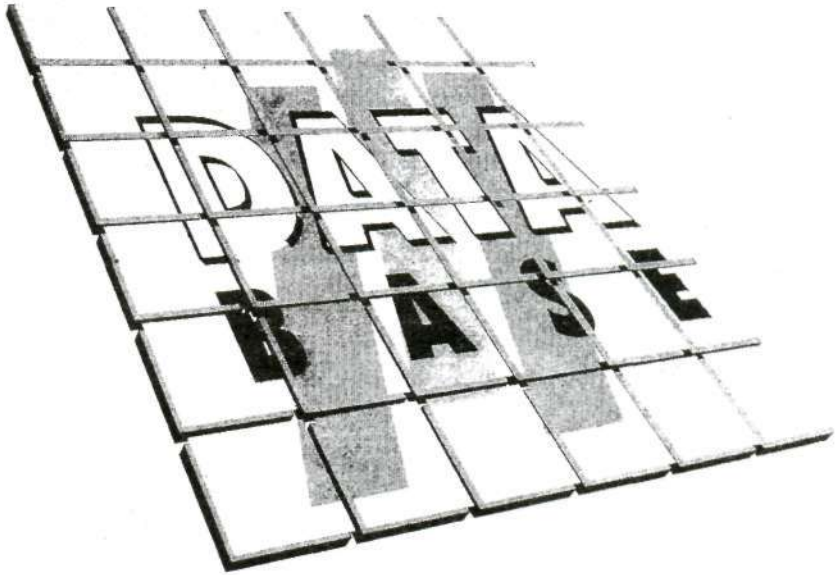
МЕЖДУНАРОДНАЯ АКАДЕМИЯ БИЗНЕСА

Е.Ж. Айтхожаева

СТАНДАРТНЫЙ ЯЗЫК БАЗ ДАННЫХ SQL

Учебное пособие



ББК 32.973я73
С 76

Рассмотрено на заседании кафедры «Информационных технологий» и рекомендовано к применению в учебном процессе.

Составитель: Айтхожаева Е.Ж.
Рецензенты:
Зав. кафедрой Информатики университета КАЙНАР
доктор техн. наук, профессор **УТЕПБЕРГЕНОВ И.Т.**
Канд. техн. наук, доцент **АЗАНОВ Н.П.**

С76 Стандартный язык баз данных SQL – Учебное пособие/Сост.Е.Ж.Айтхожаева.-Алматы,2004-48с.

ISBN 9965-9500-1-6

В учебном пособии основное внимание уделено стандартному языку взаимодействия с реляционными базами данных SQL и его реализации, применению визуальных средств проектирования запросов SQL. Кроме того, в учебном пособии рассматриваются основные модели данных в теории баз данных, классификация СБД, основы современных реляционных систем управления базами данных (СУБД), технология клиент-сервер в приложениях баз данных. Учебное пособие содержит контрольные вопросы и задания для самостоятельной работы по языку баз данных SQL. В качестве приложения приведено задание на выполнение лабораторной работы по изучению SQL.

Учебное пособие предназначено для студентов вузов при изучении дисциплин, посвященных теоретическим и практическим вопросам технологии баз данных Оно может быть использовано студентами различных специальностей.

С 2404010000
00(05)-04

ББК 32.973я73

ISBN 9965-9500-1-6

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В 60-е годы появился новый подход к организации процессов переработки информации, который основывается на понятии **систем баз данных** (СБД). **Базой данных** (БД) называют специальным образом организованные данные, хранимые в вычислительной системе (ВС). База данных создается для определенной предметной области (банк, биржа, магазин, склад, библиотека и т.д.). Сегодня базы данных можно встретить практически везде. Их используют в медицине, на транспорте, в правоохранительных органах, в городских справочных службах, на производстве и в учебных заведениях. Базы данных могут содержать в себе различную информацию, получить которую можно в считанные секунды, нажав для этого всего лишь несколько клавиш на клавиатуре компьютера.

Для создания и использования БД служат **системы управления базами данных** (СУБД), которые занимают особое место в мире программного обеспечения и нашей повседневной жизни. Системы управления базами данных обеспечивают реализацию новых концепций в организации информационных служб через создание информационных систем на основе технологии баз данных. В настоящее время широко применяются муниципальные, банковские, биржевые информационные системы, информационные системы оптовой и розничной торговли, торговых домов, служб управления трудом и занятостью, базы данных рынка товаров и услуг, справочной и аналитико-прогнозной котировочной информации и др. Как правило, работа этих систем осуществляется в локальных вычислительных сетях различной архитектуры или их объединениях, получивших название корпоративных сетей, дальнейшая интеграция которых возможна с помощью глобальной сети Интернет.

Подавляющее большинство современных СБД представляют собой системы реляционного типа, т.е.

использующие реляционную модель данных. Доктор Е.Ф.Кодд (E.F.Codd) разработал модель реляционной базы данных в 1970 г., используя математическую теорию отношений (реляционную теорию). Данные в реляционных базах данных хранятся в таблицах – **отношениях** (relation). Реляционные СБД (РСБД) - это компьютеризованные системы хранения записей в табличном виде. Под БД в различных РСБД понимается табличное хранение данных, но название "база данных" может объединять не только таблицы, но и производные этих таблиц (в виде отчетов, форм, виртуальных таблиц - представлений), формы запросов, программные модули и т.д. СУБД, поддерживающие реляционную модель данных, называются реляционными СУБД (РСУБД). Стандартным языком взаимодействия с реляционными базами данных является **язык запросов SQL**, который реализуется в РСУБД на основе операций реляционной алгебры и реляционного исчисления.

1. РЕЛЯЦИОННЫЕ СИСТЕМЫ

История реляционных СБД ведет свое начало с конца 60-х годов. Первой была экспериментальная система управления базами данных System R с реализацией в ней языка SEQUEL, который можно считать непосредственным предшественником языка запросов SQL. В реляционной модели базы данных обращение к данным происходит таким образом, как будто они сохранены в двумерной таблице (отношении).

Модель двумерной таблицы позволяет обращаться к данным как к строкам и столбцам таблицы. По этой причине данные воспринимаются значительно легче. Это связано с тем, что в каждодневной жизни часто встречаются такие таблицы с хранящимися в них данными.

Таблицу можно представить как двумерный массив или как набор записей одинаковой (для данной таблицы)

структуры. Записи еще называют строками, рядами. Столбцы таблицы называются полями или атрибутами. Другими словами, таблица состоит из записей и полей. Число полей (столбцов) и записей (строк) для каждой таблицы теоретически неограниченно, хотя на практике ограничения, естественно, существуют.

Каждая таблица описывает тип объекта, существующего в предметной области. Каждая строка в таблице описывает экземпляр объекта, поля (столбцы) таблицы содержат характеристики - значения неких свойств (атрибутов) этих объектов. Таблица является набором записей и содержит записи, объединенные по какому-то признаку. Например, в таблице 1 представлены данные о студентах, в таблице 2 – данные о служащих.

Таблица 1 - Данные о студентах

ФИО	Номер зачетной книжки	Группа	Курс	Стипендия
Иванов А.С.	99001	ЗБИ-1	4	1500
Ахметов С.Б.	99002	ЗБИ-1	4	1200
Лян Ю.В.	98123	ПОС-1	5	0
Петров С.М.	98001	ЗБИ-1	5	1200
Величко А.Р.	99120	ЗБИ-2	4	0

Таблица 2 - Данные о служащих

ФИО	Лаборатория	Табельный номер	Зарплата	Телефон
Саянов М.А.	ВТ	1	15000	55-55-55
Сидоров С.В.	ВТ	2	12000	33-33-33
Шульц Ю.В.	ИТ	3	15500	44-44-44
Петков С.М.	ИТ	4	14600	22-22-22
Исин А.Р.	ВТ	5	17000	11-11-11

Таблица должна удовлетворять следующим требованиям:

- каждая строка представляет собой кортеж из k-значений, принадлежащих k-столбцам (атрибутам);
- порядок столбцов фиксирован, порядок строк - безразличен;
- любые две строки различаются хотя бы одним элементом;
- строки и столбцы таблицы могут обрабатываться в любой последовательности;
- значение каждого атрибута в каждом кортеже является неделимым (атомарным), т.е. значение элемента не может быть множеством (условие нормализации).База данных в современных РСБД – это набор таблиц. Связь между таблицами существует на мысленном, логическом уровне и определяется предметной областью. Практически связь между таблицами устанавливается за счет логически связанных данных в разных таблицах, для чего используется, так называемый, внешний ключ.

В модели реляционной базы данных атрибут (столбец) или набор атрибутов (столбцов), значение которого однозначно определяет строки таблицы, называется **ключом** (key). Ключ, который уникально определяет строку в таблице, называется **первичным ключом** (primary key). Он используется для идентификации записи в таблице. Если добавляется столбец, который является первичным ключом одной таблицы, ко второй таблице, то там этот столбец будет называться **внешним ключом** (foreign key). Столбец называется внешним ключом потому, что он используется только для реализации операции поиска строк с одинаковыми значениями ключа в логически связанных между собой таблицах.

Все современные СУБД могут работать в вычислительных сетях. Для этого в версиях систем имеются дополнительные возможности для совместного использования данных, такие как команды блокировки и

разблокировки файлов и записей; команды защиты файлов, записей, полей данных паролями пользователя или паролем группы пользователей. Многие СУБД существуют в нескольких версиях, работающих в различных операционных системах.

В СБД в настоящее время широко используется **клиент–серверная** технология (архитектура) обработки удаленных данных. В архитектуре клиент-сервер выделяется специальное ядро – так называемый **сервер баз данных**, который принимает на себя функции обработки запросов пользователей, именуемых теперь **клиентами.**

Сервер баз данных представляет собой программу, выполняющуюся, как правило, на мощном компьютере. Приложения-клиенты посылают с рабочих станций запросы на выборку (вставку, обновление, удаление) данных. При этом сервер выполняет всю работу по отбору данных, отправляя клиенту только требуемые данные. Сервер БД возвращает клиентскому приложению только требуемые данные, которые, в общем случае, составляют малую часть от общего объема БД. Поэтому в сети не наблюдается резкого увеличения нагрузки при увеличении числа клиентов. Клиентские же приложения могут выполняться на менее мощных (по сравнению с сервером) компьютерах.

Но наиболее важным результатом перехода в архитектуру клиент-сервер является гарантированное сохранение **логической целостности** базы данных, т.е. система становится более устойчивой и более защищенной. Достигается это благодаря возможности переложить заботу о сохранении целостности базы данных на сервер. Для этого серверы обладают большим набором встроенных механизмов, защищающих систему от неверных действий клиентов. Среди этих механизмов можно назвать такие, как ограничение целостности, декларативная ссылочная целостность, триггеры, виртуальные таблицы (представления), авторизация пользователей и др.

Таким образом, клиент-серверная архитектура обеспечивает решение трёх важных задач: уменьшение нагрузки на сеть, уменьшение требований к компьютерам клиента, повышение надёжности и сохранение логической целостности базы данных.

Реляционные СУБД условно можно разделить на три класса, но все современные СУБД любого класса поддерживают реализацию SQL.

Первый класс - это персональные (настольные) СУБД (Visual dBase, Visual FoxPro, Paradox, Access и др.), способные работать в качестве клиента в клиент-серверной архитектуре.

Второй класс - SQL-серверы (MS SQL Server, MySQL, InterBase и др.), специализированные для работы в многопользовательском режиме в качестве серверов баз данных, применяющие язык SQL в качестве стандартного интерфейса с фронтальными программами на рабочих станциях. При этом в качестве фронтальных программ могут выступать как программы в среде СУБД - клиентов, так и любые программы, поддерживающие SQL-интерфейс (например, программы на языке С, электронные таблицы, издательские системы и т.д.).

Третий класс - это полнофункциональные СУБД (Oracle, Sybase, Informix, Ingres и др.), включающие в свой состав SQL-серверы и способные работать как на рабочей станции в однопользовательском режиме, так и на сервере в многопользовательском режиме. В таких СУБД есть клиентская и серверная части, которые устанавливаются соответственно на компьютер - клиент и компьютер - сервер.

Использование серверов БД и соответствующих СУБД позволяет обеспечить коллективный доступ к совокупности взаимосвязанных данных, интеграцию и централизацию управления данными, устранение излишней избыточности данных при минимальных нагрузках на сеть, защиту данных от несанкционированных пользователей, сохранение целостности (непротиворечивости) данных.

2. ЯЗЫК ВЗАИМОДЕЙСТВИЯ С РЕЛЯЦИОННЫМИ БАЗАМИ ДАННЫХ SQL

2.1. Основные сведения

Структурированный язык запросов SQL (Structured Query Language) – это язык, разработанный корпорацией IBM в 1970 г. Он фактически стал стандартом в качестве языка реляционных баз данных. Служащий IBM доктор Е.Ф.Кодд предложил язык SQL (называвший тогда SEQUEL – Structured English Query Language, структурированный английский язык запросов) как средство извлечения информации из реляционной базы данных.

В настоящее время SQL представляет собой не просто язык запросов, а наиболее распространенный язык взаимодействия с реляционными базами данных типа клиент-сервер. Основное достоинство SQL заключается в том, что он унифицирован: стандартный набор инструкций SQL можно использовать в любой системе управления базами данных, которая поддерживает SQL. Первый американский стандарт SQL был зарегистрирован в 1986 г. как ANSI X3.135-1986. ANSI (Американский национальный институт стандартизации) – это организация, которая занимается созданием и обновлением научных и инженерных стандартов. ANSI-стандарт SQL был принят в качестве всемирного стандарта отделом ООН Международной организацией стандартизации (ISO) в 1989 г. - SQL/89 (SQL/1). В настоящее время в РСУБД поддерживается стандарт ANSI X3.135-1992, широко известный как SQL/92 (SQL/2). В стадии разработки и утверждения находится стандарт SQL/3.

SQL является языком управления реляционными базами данных, а не языком программирования. Таким образом, ANSI SQL не включает ни средств управления выполнением программы (ветвлений и циклов), ни средств для создания форм или отчетов. А функции управления реализуются в языках программирования (например, xBase, C++, COBOL и

др.). Однако в некоторые версии (диалекты) SQL, например, в Transact-SQL, используемый в серверах БД Sybase и Microsoft SQL Server, добавлены два оператора (IF ELSE и WHILE).

При работе с языком SQL в клиент-серверной архитектуре все строится на понятии транзакции. **Транзакция** (transaction) – логический блок операций, содержащих одну или несколько инструкций SQL и рассматриваемых как единое целое, которые необходимо выполнить на одном или нескольких серверах (в последнем случае это распределенные транзакции).

Основное назначение транзакций – это преобразование одного согласованного состояния базы данных в другое, причем в промежуточных точках база данных находится в несогласованном состоянии. Система, поддерживающая процесс транзакции, гарантирует, что если во время некоторых обновлений базы данных произошла ошибка (по любой причине), то все эти обновления будут аннулированы.

Транзакции обладают четырьмя важными свойствами: атомарность, согласованность, изоляция и долговечность.

Атомарность. Транзакции атомарны - выполняется все из транзакции или ничего.

Согласованность. Транзакции защищают базу данных согласованно. Это означает, что транзакции переводят одно согласованное состояние базы данных в другое без обязательной поддержки согласованности во всех промежуточных точках.

Изоляция. Транзакции отдельны одна от другой. Это означает, что, если даже будет запущено множество конкурирующих друг с другом транзакций, любое обновление определенной транзакции будет скрыто от остальных до тех пор, пока эта транзакция выполняется.

Долговечность. Когда транзакция выполнена, ее обновления сохраняются, даже если в следующий момент произойдет сбой системы.

Отсюда вытекает необходимость управления

транзакциями, что предусмотрено в стандарте языка SQL. ANSI SQL включает набор стандартных команд, сгруппированных по шести категориям: описание данных, выполнение запросов, манипулирование данными, управление курсором, управление транзакциями, а также администрирование или управление. В 1989 г. в исходный стандарт ANSI были добавлены инструкции для обеспечения целостности данных.

Существует три способа реализации SQL:

1. Непосредственный вызов. Инструкции SQL передаются в систему управления реляционными базами данных, которая их выполняет.

2. Язык модулей. Позволяет записывать в текстовый файл инструкции SQL, которые позднее выполняются приложением.

3. Встроенный SQL. Это наиболее распространенный метод реализации: инструкции SQL генерируются системой управления или включаются как текст в команды языка этой системы.

В настоящее время именно язык SQL является стандартом для работы с реляционными СУБД. SQL стал единственным языком баз данных клиент-сервер. Все реляционные серверы БД являются SQL-серверами. Сервер баз данных (нижний уровень) отвечает за хранение данных. Приложения-клиенты (верхний уровень) добавляют или обновляют данные. Кроме того, приложение генерирует инструкции SQL. При регулярной работе с базами данных знание SQL обязательно. Это также относится к разработчикам, которым требуется создавать приложения с определяемыми пользователем запросами.

Если приходится работать с несколькими реляционными базами данных разных форматов или необходимо конвертировать файлы из одной базы данных в другую, использование синтаксиса SQL является самым общим и легким решением. Хранимые наборы инструкций SQL, если

они имеют общий синтаксис, могут быть легко конвертированы или использованы различными реляционными базами данных.

2.2. Стандарт SQL

SQL - очень динамичный, быстро развивающийся язык. Поэтому за достаточно короткое время были приняты уже два стандарта (SQL/89 и SQL/92) и в настоящее время имеется проект третьего стандарта (SQL/3).

В SQL/89 были стандартизованы следующие типы данных: символьные (**CHARACTER**), числовые (**NUMERIC**), целые (**INTEGER**), укороченные целые (**SMALLINT**), вещественные (**REAL**), с удвоенной точностью (**DOUBLE PRECISION**), с плавающей запятой (**FLOAT**), десятичные (**DECIMAL**).

В SQL/92 были добавлены следующие стандартные типы данных: даты (**DATE**), время (**TIME**), интервалы (**INTERVAL**). В стандарте SQL/92 появилась стандартная версия динамического SQL.

Динамический SQL включает специальные динамические операторы, поддерживающие динамическую компиляцию базовых операторов SQL, то есть компиляцию во время выполнения прикладной программы.

В языке SQL часть операторов выражается как в реляционной алгебре, часть - в стиле реляционного исчисления. В командах SQL могут использоваться арифметические операторы сравнения (>, <, >=, <=, =, <>) и логические операторы (**AND**, **OR**, **NOT**). Для обозначения всех кортежей в отношении используется символ звездочка (*). В шаблонах используются символ подчеркивание (_) для обозначения любого одиночного символа и символ процент (**%**) для обозначения любой последовательности символов. Круглые скобки используются для определения порядка выполнения действий, в фигурных скобках указываются константы типа даты, символьные константы указываются в

апострофах или двойных кавычках. Ниже рассматриваются основные синтаксические конструкции языка.

Так как SQL начинал создаваться как язык запросов, то и большинство команд предназначено для **выборки** (SELECT) результатов из таблиц баз данных, то есть отношений. Основная конструкция операции выборки: SELECT - FROM - WHERE (выбрать-из-где). Для исключения дублирующих результатов при необходимости используется опция DISTINCT (различные): SELECT DISTINCT (что) FROM (откуда) WHERE (условие). Например, для выборки из отношения R1 значений атрибутов A1 и A3 со значением атрибута A1 равного S4 и значением атрибута A2 равного 1000 с исключением дублирующих значений используется команда: SELECT DISTINCT A1, A3 FROM R1 WHERE A1="S4" AND A2=1000.

Для удаления кортежей (строк, записей) из отношений используется аналогичная конструкция команды DELETE (удалить): DELETE - FROM - WHERE. Например, для удаления из отношения (таблицы) R1 записей со значением атрибута A2, равного или больше 200, используется команда: DELETE FROM R1 WHERE A2>=200. Для удаления всех записей из отношения используется команда: DELETE <имя отношения>.

Для изменения значений атрибутов используется команда UPDATE (обновить): UPDATE - SET- WHERE. Обновление осуществляется в записях, удовлетворяющих условию, указанному после опции WHERE (или для всех записей, если отсутствует опция WHERE). Имя и значение обновляемого атрибута указывается после опции SET. Например, если в отношении R1 необходимо заменить значение атрибута A1 на S2 для записей со значением атрибута A2, равного 500 или 1000, то используется команда: UPDATE R2 SET A1='S2" WHERE A2=500 OR A2=1000.

Для вставки строк в отношения используется команда: INSERT INTO (включить в). Можно вставлять одну строку

(кортеж) или произвольное количество строк. Значения вставляемых строк определяются списком кортежей или получают в результате выполнения внутреннего запроса (подзапроса). Например, для вставки в отношение R1 записей из отношения R2 со значением атрибута A5, равного "Алматы", используется команда: INSERT INTO R1: SELECT * FROM R2 WHERE A5="Алматы". Для вставки в отношение R1 строки со значением атрибута A1, равного S5, и значением атрибута A3, равного 2000, используется команда: INSERT INTO R1 (A1, A3) VALUES ("S5", 2000).

Стандарт SQL предусматривает и другие команды, в том числе команды для работы с отношениями (таблицами) целиком:

CREATE TABLE - создание новой таблицы базы данных (структура таблицы и параметры полей задаются в команде);

ALTER TABLE - изменение структуры таблицы (изменение структуры таблицы и параметров полей задается в команде);

DROP TABLE - удаление (уничтожение) таблицы.

В командах SQL возможно использование следующих операторов:

MINUS (DIFFERENCE) - вычитание множеств, полученных в результате нескольких запросов;

UNION - объединение множеств, полученных в результате нескольких запросов;

INTERSECT - пересечение множеств, полученных в результате нескольких запросов;

IS IN - указание принадлежности множеству;

IS NOT IN - указание непринадлежности множеству;

EXISTS - квантор существования, используется как альтернатива оператора IS IN;

CONTAINS - указание содержания каких-либо значений;

DOES NOT CONTAINS - указание несодержания каких-либо значений;

BETWEEN - задание диапазона для чисел, букв и т.д.;

= - указание эквивалентности множеств;

┐ - указание неэквивалентности множеств;

LIKE - указание использования шаблона, содержащего символы _ и %;

GROUP BY - задание разделения на группы (группировка данных), группы указываются через запятую);

HAVING - задание условия отбора в группы (используется только с оператором GROUP BY);

ORDER BY ASCENDING - задание упорядочения по какому-либо столбцу (атрибуту) по возрастанию (по умолчанию) или убыванию (DESCENDING) ASCII-кодов.

При формировании запросов в языке SQL могут использоваться собственные функции работы с полями языка SQL (**агрегатные функции**), которые представлены ниже. Функции работы с полями не могут быть вложенными, работают по вертикали, их нельзя использовать для формирования условий.

COUNT (<выражение>) - подсчет числа строк.

MIN (<выражение>) - определяет наименьшее значение искомого элемента в колонке.

MAX (<выражение>) - определяет наибольшее значение искомого элемента в колонке.

SUM (<выражение>) - определяет суммарный итог по колонке числовых данных.

AVG (<выражение>) - среднее значение по колонке числовых данных.

Команда SELECT и вообще команды SQL мало зависят от текущего состояния среды исполнения, они сами открывают нужные им БД.

Стандартный язык SQL, особенно SQL/3, включает в себя большое количество команд, ниже рассматриваются наиболее часто употребляющиеся команды. Следует отметить, что реализация стандартного языка баз данных SQL несколько отличается в различных СУБД. В дальнейшем все примеры команд SQL приводятся с ориентацией на реализацию в среде

СУБД Visual FoxPro, за исключением команд управления доступом, которые реализованы только в SQL-серверах.

3. ФОРМИРОВАНИЕ ЗАПРОСОВ

Команда **выборки данных** SELECT является мощным средством обработки запросов. С ее помощью из таблицы-источника выделяются необходимые данные и пересылаются на экран, в файл, массив, таблицу, курсор, принтер. Можно с помощью команды SELECT получать данные из нескольких таблиц, логически связанных между собой. Команда имеет массу опций - возможностей. Ввиду этого ниже приведен сначала ее предварительный синтаксис с основными операторами (см. стандарт SQL в 2.2):

SELECT <что выводится> FROM <откуда (источник)>
INTO <куда (получатель)> WHERE <каким условиям должны отвечать выбираемые данные> GROUP BY <колонки, по которым выполняется группирование>
HAVING <условие отбора группируемых записей>
ORDER BY <в каком порядке выводить данные>.

Термин "колонка" здесь очень близок к понятию "поле" таблицы данных, но может быть и выражением. Кроме того, вследствие выборки можно получить как новую таблицу данных, так и текстовый файл или только отображение на экране, т.е. колонки. Команда SELECT допускает включение в себя других внутренних команд SELECT (формирование подзапросов).

Более полный формат команды SELECT включает дополнительные опции:

SELECT [DISTINCT] [<псевдоним>.]<выражение> [AS <колонка>][, [<псевдоним>]<выражение> [AS <колонка>]..]
FROM<имя_таблицы1>[<псевдоним1>][,<имя_таблицы2> [<псевдоним2>].]
[[INTO <получатель>]/[TO FILE <имя_файла> [ADDITIVE]/TO PRINTER]]

[NOCONSOLE] [PLAIN] [NOWAIT]
[WHERE <условие_связи> [AND <условие_связи>]
[AND/OR <условие_связи >]]
[GROUP BY <колонка> [, <колонка>...]] [HAVING
<условие_отбора>]
[ORDER BY <колонка> [ASC/DESC] [,<колонка>
[ASC/DESC]...]]

Включение опции DISTINCT исключает возможность вывода одинаковых строк в выборке. Перед словом FROM перечисляются отбираемые <выражения>, а после - имена таблиц, из которых берутся данные. <Выражение> может быть полем записи из таблицы, константой (выводимой в каждой строке выборки), функцией от переменных (полей) и т.п. Если <выражение> является именем поля, то оно может быть составным (с включением имени таблицы данных или псевдонима), в особенности если выборка делается из нескольких таблиц, где имена полей совпадают. Псевдонимом может быть не только официальный псевдоним (ALIAS) таблицы данных, но и любое другое имя, которое присваивается в команде SELECT. Это задаваемое временное имя указывается в опции <псевдоним> после слова FROM за именем таблицы и используется только в данной команде SELECT в других ее опциях (локальный псевдоним). Если необходимо построить выборку из всех полей таблицы данных, вместо их перечня можно указать символ *. В результате выполненной выборки получается совокупность колонок, заголовками которых могут быть имена исходных полей.

Например, для вывода всех записей из таблицы STUD используется команда:

```
SELECT * FROM STUD
```

Для вывода названий всех лабораторий организации (поле LAB) из таблицы данных KADR с предотвращением вывода дублирующих значений, используется команда:

```
SELECT DISTINCT LAB FROM KADR
```


Команду SELECT можно использовать для одновременной выборки данных из нескольких таблиц. Если имена полей, выбираемых из разных таблиц совпадают, то такие колонки получают совпадающие имена, к которым присоединяется одна из букв (по алфавиту), например, FAM_A, FAM_B и т.д. Аналогичным образом даются имена колонкам, полученным в результате вычисления выражений. Их имена состоят из слова EXP и последовательных чисел (EXP_1, EXP_2 и т.д.). Исключения составляют выражения, использующие собственные функции SQL: AVG, MIN, MAX, SUM, COUNT. Последняя функция может иметь в качестве аргумента здездочку (COUNT(*)), что означает подсчет всех записей, попавших в выборку. Имена колонок в этом случае будут включать имена функций. Вместо имен, формируемых по умолчанию, можно назначить колонкам другие имена, указав их после слова AS в виде <выражение> AS <новое_имя_колонки>.

Например, для вывода минимального, максимального и среднего значений зарплаты (поле ZARP) из таблицы KADR необходимо использовать команду:

```
SELECT MIN(ZARP), MAX(ZARP), AVG(ZARP)
FROM KADR
```

Для вывода из таблицы KADR фамилий (поле FAM) и табельных номеров (поле TAB) с другими именами колонок (Фамилии и Таб_номер) используется команда:

```
SELECT FAM AS Фамилии, TAB AS Таб_номер
FROM KADR
```

Для вывода всех фамилий из двух таблиц данных KADR и STUD с использованием локальных псевдонимов P и T, а также с заменой при выводе имен колонок FAM_A и FAM_B на KFIO и STFIO соответственно, используется команда:

```
SELECT P.FAM AS KFIO, T.FAM AS STFIO
FROM KADR P, STUD T
```

В команде SELECT можно задать достаточно сложные условия для выборки данных в запрос. Условие связи

применяется в случае, если выборка делается более, чем из одной таблицы данных, и определяет критерий объединения данных из разных таблиц. В условии связи указываются поля из разных таблиц с псевдонимами и используются знаки отношений =, #, = =, >, >=, <, <=. Допускается задание нескольких критериев, соединенных знаком AND. Условие выборки строится аналогично, но из выражений только для одной таблицы, и допускается использование логических операторов OR, AND и NOT.

Условия могут содержать следующие операторы SQL: LIKE, BETWEEN, IN. Эти операторы можно комбинировать с помощью связок OR, AND, NOT и скобок.

Оператор LIKE позволяет построить условие сравнения по шаблону, где символ "_" указывает единичный неопределенный символ в строке, а символ "%" - любое их количество. Формат оператора LIKE:

<выражение> LIKE <шаблон>.

Оператор BETWEEN задает начальное и конечное значение диапазона и проверяет, находится ли выражение, стоящее слева от оператора, в указанном диапазоне. Формат оператора BETWEEN:

<выражение> BETWEEN <нижнее значение> AND <верхнее значение>.

Оператор IN проверяет, находится ли выражение, стоящее слева от слова IN, среди перечисленных справа от него. Формат оператора IN:

<выражение> IN (<выражение1>, <выражение2>,...)

Например, для выборки фамилий (поле FAM), начинающихся на букву "А", всех мужчин (поле POL="M") из таблицы KADR. DBF используется команда:

SELECT FAM FROM KADR WHERE POL="M" AND
FAM LIKE "A"

Для вывода из таблицы STUD фамилий студентов (поле FAM), получающих стипендии (поле STIP) от 1500 до 2000 тенге, используется команда:

SELECT FAM FROM STUD WHERE STIP
BETWEEN 1500 AND 2000

Для вывода на экран из таблицы KADR фамилий сотрудников (поле FAM), а из таблицы TABEL - соответствующего количества рабочих дней (поле WD) для записей, у которых совпадают табельные номера, необходимо использовать команду с условием связи (в качестве псевдонимов таблиц указаны имена таблиц):

SELECT KADR.FAM, TABEL.WD FROM KADR, TABEL
WHERE KADR.TAB=TABEL.TAB

Для упорядочения по заданной колонке или колонкам используется опция ORDER BY. По умолчанию сортировка выполняется по возрастанию (ASC), но может быть задана и по убыванию (DESC).

Сокращенный вариант команды SELECT с использованием опции ORDER BY выглядит следующим образом:

SELECT <выражение> FROM <имя_таблицы>
ORDER BY <колонка1> [ASC/DESC] [,<колонка2>..]

Указание колонки упорядочения может выполняться именем или номером колонки. Например, для вывода фамилий (поле FAM) и зарплаты (поле ZARP) из таблицы данных KADR в порядке убывания размера заработной платы используется команда:

SELECT FAM, ZARP FROM KADR ORDER BY ZARP DESC

Для вывода фамилий (поле FAM) и группы (поле GRUPPA) из таблицы данных STUD в порядке возрастания групп и фамилий необходимо использовать команду:

SELECT FAM, GRUPPA FROM STUD
ORDER BY GRUPPA, FAM

В опции ORDER BY обычно нельзя использовать вычисляемые выражения. В случае необходимости упорядочения по колонке с вычисляемыми значениями указывается номер колонки. Например, для вывода фамилий (поле FAM) и премии, равной половине зарплаты (величина

премии вычисляется по значению поля ZARP) из таблицы данных KADR в порядке возрастания премии используется команда:

```
SELECT FAM AS Фамилия, ZARP AS Премия
FROM KADR ORDER BY 2,1
```

В этой команде фамилии сотрудников, имеющих одинаковую премию выводятся в алфавитном порядке, так как упорядочение данных выполняется сначала по второй колонке (премия), затем по первой колонке (фамилия).

Опция GROUP BY команды SELECT позволяет сгруппировать записи с одинаковым значением указанной колонки (или колонок):

```
SELECT <выражение> FROM <имя_таблицы>
GROUP BY <колонка1> [,<колонка2>...] [HAVING
<условие_отбора>]
```

Опция GROUP BY задает колонки, по которым производится группирование выходных данных. Все записи таблицы, для которых значения колонок совпадают, отображаются в выборке единственной строкой. Группирование удобно для получения некоторых сводных характеристик группы (суммы, среднего значения, количества записей в группе и т.д.).

Опция HAVING <условие_отбора> задает критерий отбора данных в каждую сформированную в процессе выборки группу, т.е. выполняет роль опции WHERE, но для группируемых данных.

Например, для вывода названий всех групп (поле GRUPPA), количества студентов и суммы стипендий (поле STIP) для каждой группы из таблицы данных STUD используется команда:

```
SELECT GRUPPA, COUNT(FAM), SUM(STIP)
FROM STUD GROUP BY GRUPPA
```

Внутри групп можно создавать подгруппы. При группировании данных в опции HAVING можно использовать агрегатные функции SQL. Например, для

вывода названий всех лабораторий (поле LAB), количества сотрудников и значения суммарной зарплаты (поле ZARP) из таблицы данных KADR используется следующая команда (информация выводится только для лабораторий, где количество сотрудников больше 5):

```
SELECT LAB, SUM(ZARP), COUNT(*) FROM KADR
GROUP BY LAB HAVING COUNT(*)>5
```

Для указания объекта получателя данных выборки используется опция INTO или TO.

Ниже приведен сокращенный вариант команды SELECT с опцией INTO/TO, используемый в Visual FoxPro:

```
SELECT <выражение> FROM <имя_таблицы>
[INTO TABLE <имя_таблицы>] / [INTO CURSOR
<имя_курсора>] / [INTO ARRAY <имя_массива>] /
[TO FILE <имя_файла> [ADDITIVE]] /
[TO PRINTER] [NOCONSOLE] [PLAIN] [NOWAIT]
```

Типы возможных получателей данных выборки в Visual FoxPro описаны ниже.

TABLE <имя_таблицы> - получателем является новая таблица с указанным именем.

CURSOR <имя_курсора> - результат запроса помещается в **курсор** с указанным именем. Курсор - это временный набор данных, который может быть областью памяти или временным файлом и имеет режим "только чтение". Данные курсора могут быть, например, предъявлены в команде BROWSE, напечатаны, из них может быть образовано меню и т.д. Курсор может быть обработан другой командой SELECT. К колонкам курсора надо обращаться по имени этих колонок с префиксом - именем курсора (через точку).

ARRAY <имя_массива> - в качестве получателя результата запроса будет использован новый двумерный массив с указанным именем.

Кроме того, данные выборки можно переслать в файл или на принтер. Для этого в команде указывается получатель TO FILE <имя_файла> [ADDITIVE] / TO PRINTER и выборка

посылается в текстовый файл с указанным именем или на принтер. Если используется слово ADDITIVE, то выборка будет добавлена в конец существующего файла без его перезаписи.

Следующие опции имеют смысл только при выдаче на экран (команда используется без опций INTO или TO):

NOCONSOLE - выборка не выдается на экран,

PLAIN - заголовки колонок не выдаются,

NOWAIT - не делается пауза при заполнении экрана.

Например, для вывода фамилий (поле FAM) и зарплаты (поле ZARP) из таблицы KADR в таблицу FAMSZ необходимо использовать команду:

```
SELECT FAM, ZARP FROM KADR
INTO TABLE FAMSZ
```

Для вывода фамилий (поле FAM) и стипендий (поле STIP) из таблицы STUDENT в курсор CURSTUD необходимо использовать команду:

```
SELECT FAM, STIP FROM STUDENT
INTO CURSOR CURSTUD
```

Для занесения в текстовый файл FIO.TXT всех фамилий из двух таблиц данных KADR и STUD с использованием локальных псевдонимов S и T и заменой при выводе имен колонок FAM_A и FAM_B на KADRFIO и STUDFIO, соответственно, используется команда:

```
SELECT  S.FAM AS KADRFIO, T.FAM AS STUDFIO
FROM KADR S, STUD T  TO  FILE FIO.TXT
```

Для вывода на принтер названий лабораторий (поле LAB), сумм зарплат и количества сотрудников, получающих зарплату (поле ZARP) больше 12000, сгруппированных по каждой лаборатории необходимо использовать следующую команду (вывод данных в порядке убывания сумм зарплат):

```
SELECT LAB, SUM(ZARP), COUNT(*) FROM KADR
TO PRINTER WHERE ZARP>12000 GROUP BY LAB
ORDER BY ZARP DESC
```


Использование подзапросов в условиях команды SELECT позволяет создавать сложные запросы. Подзапрос заключается в круглые скобки и представляет собой вложенную команду SELECT, вложение подзапросов неограничено. Причем, основной запрос и используемый в нем подзапрос могут обращаться как к одним и тем же таблицам, так и к разным таблицам баз данных. В подзапросе нельзя использовать опции ORDER BY и INTO.

Запрос с подзапросом может быть некоррелирован или коррелирован. Некоррелированный подзапрос выполняется в первую очередь, затем полученный результат подставляется в условие и выполняется внешний запрос. Например, для выборки фамилий сотрудников (поле FAM) из таблицы KADR с заработной платой (поле ZARP) выше средней можно использовать запрос с подзапросом (для таблицы KADR используется локальный псевдоним S1):

```
SELECT S1.FAM FROM KADR S1 WHERE ZARP >
(SELECT AVG(ZARP) FROM KADR)
```

В коррелированном подзапросе внутренний запрос ссылается на внешний запрос и выполняется поочередно для каждой строки внешнего запроса (многократно). Выбирается первая строка во внешнем запросе, для нее выполняется внутренний запрос, затем вторая строка и так далее. Например, для выборки фамилий сотрудников (поле FAM) и зарплаты (поле ZARP) из таблицы KADR с заработной платой выше средней по каждой лаборатории можно использовать запрос с подзапросом (для таблицы KADR используются локальные псевдонимы S1 во внешнем запросе и S2 в подзапросе для обеспечения сравнения):

```
SELECT FAM, ZARP FROM KADR S1 WHERE ZARP >=
(SELECT AVG(ZARP) FROM KADR S2
WHERE S2.LAB=S1.LAB)
```

Результаты нескольких выборок можно объединить в одном запросе, используя оператор объединения UNION. Результатом будет множество, состоящее из всех строк

входящих в какую-либо выборку или в несколько выборок. Но при этом результаты исходных выборок должны иметь одинаковое число полей (столбцов), тип и ширина i-го поля одной выборки должны совпадать с типом и шириной i-го поля любой другой выборки. При использовании опции UNION часто оказывается полезным включение константы в получаемый результат выборки. Заголовки колонок в выборке определяются первым запросом. Например, текстовую константу можно использовать в качестве поясняющего текста при выборе из таблицы STUD фамилий студентов (поле FAM), получающих стипендию (поле STIP) больше 2000 или проживающих з городе Алматы (город проживания указывается в поле ADRESS):

```
SELECT FAM AS Фамилии, "стипендия больше 2000" AS
Признак_выборки FROM STUD WHERE STIP> 2000
      UNION
SELECT FAM, "город Алматы" FROM STUD WHERE
ADRESS LIKE "%Алматы"
```

Оператором UNION можно соединить любое число команд SELECT, но опция ORDER BY в запросе с использованием оператора UNION может входить только в последнее предложение SELECT. При указании критерия упорядочивания указываются номера полей в получаемой выборке. Например, при выборке из таблицы KADR фамилий сотрудников (поле FAM), имеющих заработную плату (поле ZARP) меньше 10000 или проживающих в городе Алматы (город проживания указывается в поле ADRESS), можно сначала упорядочить данные по второй колонке (признак выборки), а затем по первой колонке (фамилии в алфавитном порядке):

```
SELECT FAM , "заработная плата меньше 1000"
FROM KADR WHERE ZARP<10000
UNION
SELECT FAM , " город Алматы " FROM KADR
WHERE ADRESS LIKE "%Алматы" ORDER BY 2,1
```


По умолчанию оператор UNION устраняет из результата повторяющиеся строки. Чтобы отобразить все строки, необходимо использовать оператор UNION с опцией ALL (UNION ALL).

4. ОПРЕДЕЛЕНИЕ ДАННЫХ

База данных во многих СУБД представляет собой контейнер, содержащий таблицы, представления (виртуальные таблицы), триггеры (для отслеживания ограничений целостности – непротиворечивости данных), хранимые процедуры и другие объекты.

Для **создания** базы данных используется команда:

```
CREATE DATABASE <имя_БД>
[необязательные_параметры].
```

Состав необязательных параметров зависит от используемой СУБД, при их отсутствии в команде используются значения по умолчанию.

Например, для создания БД в Visual FoxPro с именем BASA1 используется команда:

```
CREATE DATABASE BASA1
```

Для **определения** структуры таблицы данных используются команды создания (CREATE TABLE) и изменения (ALTER TABLE) структуры таблицы. Таблица создается в активной БД. Если нет активной БД, то в СУБД, допускающих наличие свободных таблиц, создается свободная таблица (не входящая в состав БД).

Команда создания структуры таблицы данных имеет следующий формат:

```
CREATE TABLE <имя_таблицы> (<определение_поля1>
[,<определение_поля2>...][определение_ограничений])
```

Команда создает новую таблицу данных с указанным именем. Для каждого поля задаются его имя и тип. Для типов полей, которые могут иметь разную ширину, дополнительно необходимо указать требуемую ширину. Для числовых полей

с плавающей запятой указывается также точность (число десятичных разрядов). Если создаваемое поле будет являться ключевым (индексом), то необходимо указать тип ключа (индекса). Кроме того, в определении поля можно указать признак (NOT NULL) обязательного заполнения поля при вводе данных в таблицу, значения полей по умолчанию и ограничения на ввод значений. Определение ограничений таблицы используется для задания ограничений на уровне таблицы, могут указываться также определения первичного и внешнего ключей.

В Visual FoxPro команда CREATE TABLE создает новую таблицу данных (DBF-файл) с указанным именем. Для каждого поля задаются его имя и тип (одной из букв: C - символьный; N - числовой; F - с плавающей запятой; D - дата; L - логический; M - примечание и т.д.). Например, для создания структуры таблицы KADR, предназначенной для хранения следующей информации: фамилий (поле FAM, символьного типа, длиной 25 символов); табельных номеров (поле TAB, целого типа, является первичным ключом - индексом); дат рождения (поле DTR, типа даты); размера зарплаты (поле ZARP, числового типа, общей длиной 8 символов, из которых 2 символа под дробную часть числа) - необходимо использовать следующую команду:

CREATE TABLE KADR (FAM C(25),
TAB I PRIMARY KEY, DTR D, ZARP N(8,2))

Можно создать временную таблицу, с которой можно работать как с обычной таблицей данных, называемую **курсором** (CURSOR). Курсор располагается в оперативной памяти в свободной рабочей области и доступен сразу после создания до тех пор, пока он не будет закрыт. При закрытии курсора временная таблица удаляется из памяти. Команда создания курсора CREATE CURSOR аналогична команде CREATE TABLE.

Кроме того, с помощью команды CREATE можно создать индекс (CREATE INDEX), триггер (CREATE TRIGGER),

представление (CREATE VIEW), значение по умолчанию (CREATE DEFAULT), ограничение на данные (CREATE RULE), хранимую процедуру (CREATE PROCEDURE), пользовательскую функцию (CREATE FUNCTION), схему данных (CREATE SCHEME).

При **изменении структуры** таблицы используется команда ALTER TABLE с ключевыми словами, которые позволяют добавить, удалить, переименовать поле, изменить параметры поля. Эта команда имеет формат:

ALTER TABLE <имя_таблицы>
[ADD <определение_добавляемого_поля>]
[DROP <имя_удаляемого_поля>]
[RENAME <старое_имя_поля> TO <новое_имя_поля>]
[ALTER <имя_изменяемого_поля>
<определение_изменяемых_параметров_поля>]

Например, для добавления символьного поля FAX шириной 12 позиций и удаления поля TELEFON в таблице KADR можно использовать команду:

ALTER TABLE KADR ADD FAX C (12) DROP TELEFON

Кроме того, с помощью команды ALTER можно изменить триггер (ALTER TRIGGER), представление (ALTER VIEW), хранимую процедуру (ALTER PROCEDURE), пользовательскую функцию (ALTER FUNCTION).

С помощью команды DROP можно **удалить** любые объекты, созданные командой CREATE.

Можно удалить всю таблицу целиком командой:

DROP TABLE <имя_таблицы>

Например, следующая команда удалит таблицу BD1:

DROP TABLE BD1

Для удаления базы данных со всеми таблицами используется команда:

DROP DATABASE <имя_базы_данных>

Например, следующая команда удалит базу данных BASA1:

DROP DATABASE BASA1

5. МАНИПУЛИРОВАНИЕ ДАННЫМИ

Дополнять записями базу данных можно используя в качестве источника данных выражения, массивы, переменные, константы, результат запроса.

Для дополнения записей в таблицу БД используется команда INSERT, которая предварительно открывает закрытую таблицу БД. Формат команды INSERT:

INSERT INTO <имя_таблицы> [(<имя_поля1>[, <имя_поля2> …])] VALUES (<выражение1> [, <выражение2> …])

Команда добавляет записи в конец существующей таблицы, используя выражения, перечисленные после слова VALUES. Выражения заносятся в указанные поля. Если опущены имена полей, выражения будут записываться в последовательные поля таблицы данных в соответствии с ее структурой.

Например, результат следующих двух команд будет одинаковым:

INSERT INTO KADR (FAM, ZARP) VALUES ('Киров', 18200)
INSERT INTO KADR (ZARP, FAM) VALUES (18200, 'Киров')

В качестве выражений могут использоваться значения переменных или элементов массивов переменных. Например, в СУБД Visual FoxPro для дополнения записей в таблицу данных с использованием массивов используется следующая команда SQL:

INSERT INTO <имя_таблицы>
FROM ARRAY <имя_массива>

Команда добавляет записи в конец таблицы данных, используя данные, содержащиеся в указанном массиве. Данные из массива заносятся последовательно в поля, начиная с первого. Типы соответствующих полей и элементов массива должны совпадать.

Например, для определения значения элементов массива C1 и добавления этих данных из массива в поля таблицы

данных KADR в Visual FoxPro необходимо использовать следующую последовательность команд:

C1(1)="Ахметов"
C1(2)=235
C1(3)={12/20/74}
C1(4)=8000
INSERT INTO KADR FROM ARRAY C1

Во многих СУБД (но не во всех) SQL-команду INSERT можно использовать с подзапросом - вложенной командой SELECT, если множество дополняемых данных является результатом запроса:

INSERT INTO <имя_таблицы> [(<имя_поля1> [, <имя_поля2> ...])] <подзапрос>

Например, дополнение в таблицу STUD1 фамилий студентов (поле FAM) из таблицы STUD2 осуществляется следующей командой:

INSERT INTO STUD1 (FAM) VALUES
SELECT DISTINCT FAM FROM STUD2

Для **модификации** данных в таблице БД используется команда UPDATE, которая имеет следующий формат:

UPDATE <имя_таблицы> SET <имя_поля1>= <выражение1> [,<имя_поля2> = <выражение2>..] [WHERE <условие>]

Замена значений происходит в записях, удовлетворяющих условию, указанному после опции WHERE. Если эта опция отсутствует, то замена значений происходит во всех записях таблицы. Например, для замены фамилии (поле FAM) 'Ахметов' на 'Ахмет' в таблице STUD необходимо использовать следующую команду:

UPDATE STUD SET FAM ='Ахмет' WHERE FAM='Ахметов'

При модификации данных можно использовать подзапрос - вложенную команду SELECT для формирования операнда условия:

UPDATE <имя_таблицы> SET <имя поля>=<выражение> [WHERE <операнд_условия> <оператор_условия> (<подзапрос>)]

Например, установка надбавки к стипендии студентов (поле STIP) на 25% в таблице STUD1, если фамилии этих студентов (поле FAM) имеются в таблице STUD2, реализуется следующей командой:

```
UPDATE STUD1 SET STIP=1.25*STIP
WHERE FAM IN (SELECT FAM FROM STUD2)
```

Для **удаления** записей в SQL используется команда DELETE, имеющая следующий формат:

DELETE FROM <имя_таблицы> [WHERE <условия>]

Удаляются записи, удовлетворяющие условию, указанному после опции WHERE. Если эта опция отсутствует, то удаляются все записи из таблицы.

При удалении можно использовать подзапрос - вложенную команду SELECT для формирования операнда условия:

```
DELETE FROM <имя_таблицы>
[WHERE <операнд_условия> <оператор_условия>
(<подзапрос>)]
```

В Visual FoxPro поддерживается команда SQL - DELETE, которая **помечает** записи на удаление без физического удаления записей. Физическое удаление записей можно реализовать командой PACK. Например, для пометки на удаление данных о сотруднике с табельным номером 65 (поле TAB) из таблицы KADR используется команда:

```
DELETE FROM KADR WHERE TAB=65
```

Для пометки на удаление всех записей из таблицы KADR необходима команда:

```
DELETE FROM KADR
```

Пометка на удаление из таблицы STUD1 данных о студентах, если фамилии этих студентов (поле FAM) имеются в таблице STUD2 реализуется следующей командой:

```
DELETE FROM STUD1 WHERE FAM IN
(SELECT FAM FROM STUD2)
```


6. УПРАВЛЕНИЕ ТРАНЗАКЦИЯМИ

В технологии баз данных **транзакция** (см.2.1) представляет собой логическую единицу работы с данными в базе данных. В общем случае транзакция представляет собой последовательность нескольких операций, которая преобразует некоторое целостное состояние базы данных в другое целостное состояние, однако не гарантирует сохранения целостности во все промежуточные моменты времени.

В SQL имеются команды управления транзакциями COMMIT (**фиксация**) и ROLLBACK (**откат**, аннулирование). ROLLBACK прерывает выполнение транзакции.

Команда COMMIT сигнализирует о успешном завершении транзакции. Она сообщает СУБД, что логическая единица работы была успешно завершена, база данных снова находится в целостном состоянии, все произведенные этой единицей работы обновления могут быть зафиксированы (приняты за совершившиеся).

Команда COMMIT устанавливает, так называемую, точку фиксации (синхронизации). Точка фиксации соответствует концу логической единицы работы. И выполнение команды ROLLBACK (откат) возвращает БД в состояние предыдущей точки фиксации.

Команда ROLLBACK сигнализирует о неудачном завершении транзакции. Она сообщает СУБД, что в результате аварийной ситуации, возможно, было нарушено целостное состояние БД. Поэтому для всех обновлений, произведенных транзакцией, необходимо выполнить откат, т.е. аннулировать их. Под обновлением в этом случае понимаются результаты выполнения команд INSERT, UPDATE, DELETE.

Наличие команд COMMIT и ROLLBACK является гарантией того, что транзакция будет выполнена до конца

или, в противном случае, будут аннулированы все обновления, произведенные незавершившейся транзакцией.

Во многих СУБД начало и конец транзакций можно объявить явно командами, соответственно, BEGIN TRANSACTION и END TRANSACTION.

Для восстановления баз данных при возникновении ошибок, требующих отката или наоборот обновления информации, используется журнал транзакций. Журнал транзакций - это то средство, с помощью которого СУБД обеспечивает целостность баз данных, причем как в отношении правил, которыми управляет сервер, так и в отношении обработки, которую реализуют приложения.

7. УПРАВЛЕНИЕ ДОСТУПОМ

В современных системах управления базами данных поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных, а именно **избирательный** подход или **мандатный** (обязательный) подход. В обоих подходах единицей данных или объектом данных, для которых должна быть создана система безопасности, может быть как вся база данных целиком или какой-либо набор отношений, так и некоторое значение данных для заданного атрибута внутри некоторого кортежа в определенном отношении. В системе управления базами данных поддерживается либо избирательное, либо мандатное, либо оба типа управления доступом.

В случае избирательного управления некий пользователь обладает различными правами (привилегиями или полномочиями) при работе с разными объектами. Более того, разные пользователи обычно обладают и разными правами доступа к одному и тому же объекту. Поэтому избирательные схемы характеризуются значительной гибкостью.

Право доступа предоставляется пользователю или группе пользователей для выполнения таких функций, как выборка

данных, добавление новых строк или обновление данных.

Командами SQL реализуется избирательное управление доступом, для чего используются команды управления доступом GRANT (**предоставление прав**) и REVOKE (**аннулирование прав**).

Формат команды выдачи прав доступа GRANT:

GRANT <привилегии> ON <объект> TO <идентификатор_пользователя> [WITH GRANT OPTION]

Привилегии указывают права доступа, которые будут выданы пользователю на указанный объект:

SELECT - позволяет извлекать данные из таблиц и представлений;

INSERT - позволяет вставлять новые строки в таблицу или представление;

DELETE - позволяет удалять строки из таблицы или представления;

UPDATE - позволяет модифицировать данные в таблице или представлении;

REFERENCES - данная привилегия разрешает ссылаться на таблицу (при создании внешних ключей), представление или хранимую процедуру.

Привилегии SELECT, INSERT, UPDATE, могут быть предоставлены и на отдельные столбцы (поля) таблицы или представления.

Использование параметра WITH GRANT OPTION позволяет пользователям, указанным своими идентификаторами, предоставлять права выдавать привилегии другим пользователям, аналогичные выданные им самим.

Пример применения команды GRANT для предоставления прав доступа пользователям CLERK1 и CLERK2 на выборку данных из всей таблицы STUD и модификацию данных столбца STIP таблицы STUD:

```
GRANT SELECT, UPDATE (STIP) ON STUD
TO CLERK1, CLERK2
```

Формат команды отмены прав доступа REVOKE:

REVOKE [GRANT OPTION FOR] <привилегии> ON <объект>FROM <идентификатор_пользователя> [CASCADE]

Параметры команды REVOKE аналогичны параметрам команды GRANT. Если указан параметр CASCADE, то отмена прав доступа данного пользователя распространяется на всех пользователей, которые получили аналогичные права от данного пользователя. Использование в команде параметра GRANT OPTION FOR позволяет лишать пользователя возможности выдавать права доступа другим пользователям.

Пример применения команды REVOKE для отмены прав пользователя CLERK2 на модификацию данных столбца STIP таблицы STUD:

REVOKE UPDATE (STIP) ON STUD FROM CLERK2

8. ВИЗУАЛЬНОЕ ПРОЕКТИРОВАНИЕ ЗАПРОСОВ В СУБД VISUAL FOXPRO

Из всех команд SQL наиболее часто используется команда SELECT (запрос на выборку). Во многих СУБД существуют **визуальные** средства проектирования, которые обеспечивают визуальное проектирование запросов на языке SQL и работу с ними. Для создания запросов в СУБД Visial FoxPro могут использоваться **мастер запросов** и **конструктор запросов**.

Мастер запросов (Query Wizard) - позволяет создать стандартные запросы из одной или нескольких таблиц, указав условия для выбора данных и условия для связи между таблицами, при этом постоянные связи между таблицами БД игнорируются.

Конструкторы имеют набор специальных средств, которые позволяют с учетом специфики каждого объекта максимально облегчить и сделать наглядным процесс разработки компонента. Для создания запроса с помощью Конструктора запросов (Query Designer) Visual FoxPro в меню

File (Файл) выбирается команда New (Новый). В открывшемся диалоговом окне New необходимо установить опцию Query (Запрос) и нажать кнопку New file (Новый файл). Откроется окно Query Designer - Конструктор запросов (рисунок 1).

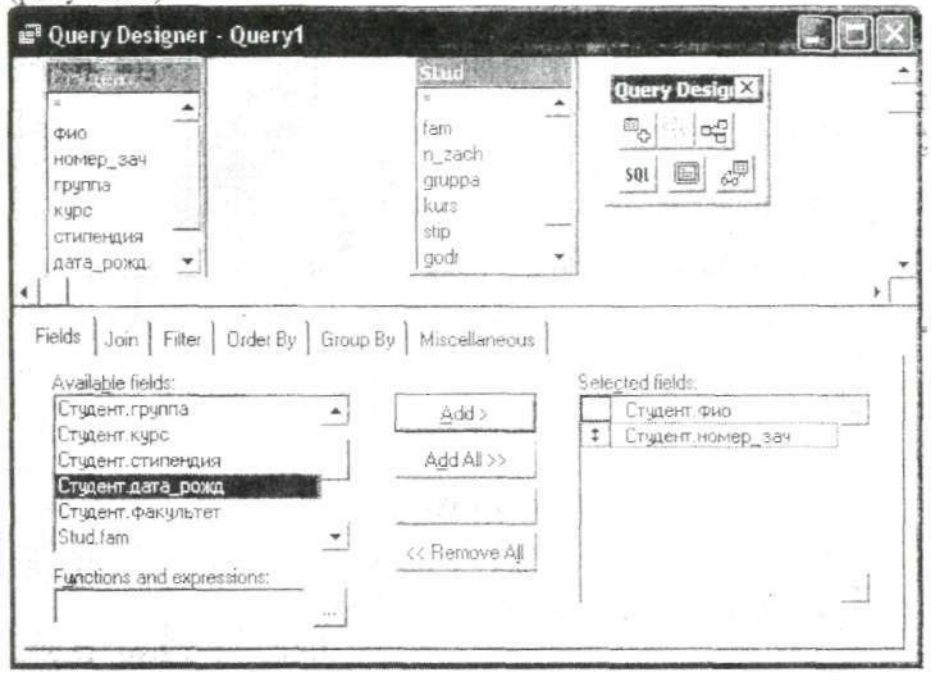
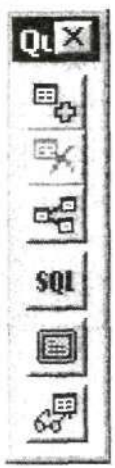


Рисунок 1. Окно Конструктора запросов

В окне Конструктора запросов первоначально отсутствуют источники данных для запроса. После открытия Конструктора запросов появляется окно Add Table or View (добавить таблицу или представление), где можно выбрать базу данных, свободную таблицу или представление, из которых будут выбираться данные в запрос. После выбора таблиц, представлений или базы данных они появляются в окне Query Designer. На рисунке 1 представлено окно Конструктора запросов с таблицами данных.

Окно Query Designer имеет своё меню и панель инструментов Query Designer, с помощью которых можно добавить таблицы в окно Конструктора запроса, создать запрос, сохранить его в виде программного файла с расширением .QPR, включить в запрос различные выражения и функции, указать все опции команды SELECT. На рисунке 2 представлена панель инструментов Query Designer и назначение кнопок панели. Используя последнюю кнопку (Query Destination) панели инструментов Query Designer, можно через диалоговое окно Query Destination (рисунок 3) выбрать получателя данных выборки: виртуальную таблицу (Cursor), физическую таблицу (Table), экран монитора (Screen), отчёт (Report), ярлык (Label). По умолчанию результаты выполнения запроса выводятся в режиме Browse.



- Add table добавляет таблицу или представление.
- Remove table возвращает (убирает) таблицу или представление.
- Add join (добавить связь) для описания связей между таблицами.
- Show the SQL window показывает окно с командой SQL.
- Maximize the table view убирает вкладки с окна Конструктора запросов.
- Query Destination позволяет выбрать получателя результатов запроса.

Рисунок 2. Панель инструментов Query Designer

Используя последнюю кнопку (Query Destination) панели инструментов Query Designer можно через диалоговое окно Query Destination (рисунок 3) выбрать получателя данных выборки: виртуальную таблицу (Cursor), физическую таблицу (Table), экран монитора (Screen), отчёт (Report), ярлык

(Label). По умолчанию результаты выполнения запроса выводятся в режиме Browse.

Окно Конструктора Запросов имеет несколько вкладок, которые служат для визуального задания опций команды SELECT. Ниже рассматривается назначение этих вкладок и работа с ними.

Вкладка **Fields** (рисунок 1) позволяет задать имена полей, данные из которых выбираются в запрос.В левом списке доступных полей таблицы (Available fields) выделяется необходимое поле, а кнопка Add (добавить) позволяет это поле перенести в правый список выбранных полей (Selected fields). Кнопка Add All (добавить все) используется для выбора всех полей таблицы. Кнопка Remove (вернуть) - для удаления столбца из списка выбранных полей. Кнопка Remove All (вернуть все) возвращает все выбранные столбцы из списка выбранных полей.

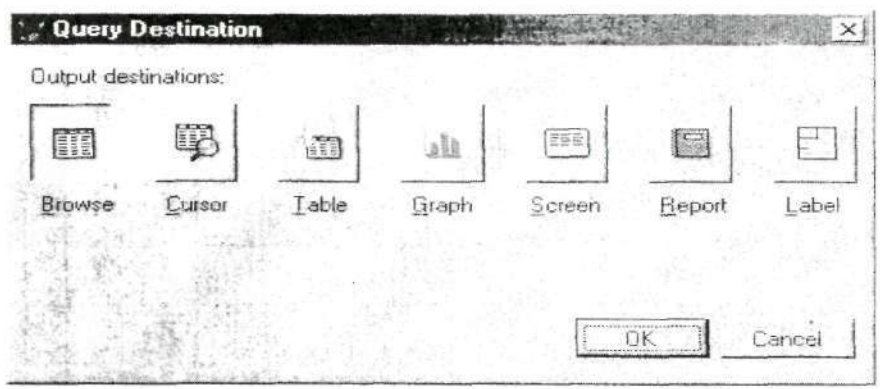


Рисунок 3. Окно Query Destination

В поле **Functions and expressions** формируются вычисляемые выражения как вручную, так и с помощью Построителя Выражений (Expression Builder). При этом можно использовать различные функции и выражения. В выражении можно указать все стандартные функции Visual FoxPro, являющиеся функциями языка программирования

xBase, а также собственные функции SQL для работы со столбцами. Кнопка Add (добавить) позволяет это выражение добавить в правый список выбранных полей.

Вкладка **Join** (рисунок 4) позволяет описать связи между таблицами по какому-либо критерию (условию).

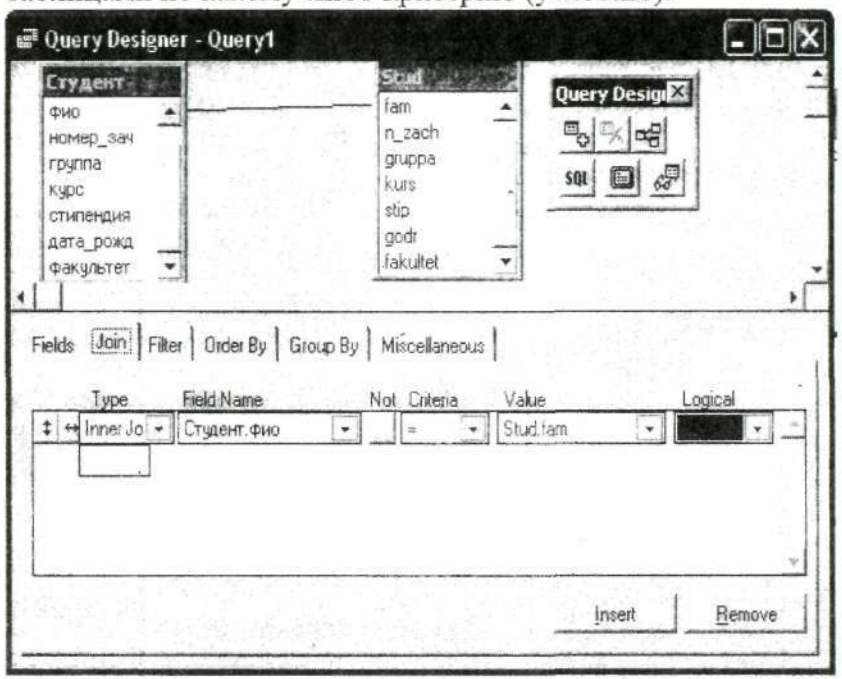


Рисунок 4. Вкладка Join окна Конструктора запросов

Возможны следующие типы связей (Type):

- Inner Join – эта связь позволяет включить в выборку только те значения из первой и второй таблицы, которые удовлетворяют указанному условию;
- Left Outer Join – эта связь позволяет включить в выборку все записи из первой таблицы и записи из второй таблицы, удовлетворяющие указанному условию;
- Right Outer Join – эта связь позволяет включить в выборку все записи из второй таблицы и записи из первой

таблицы, удовлетворяющие указанному условию;

- Full Outer Join – эта связь позволяет включить в выборку все записи из первой и второй таблицы.

Также во вкладке Join указывается имя поля (Field Name), по которому будет происходить связь, и условие - критерий связи (Criteria): =, ==, LIKE (похожий), >, <, >=, <=, BETWEEN (между), IN (в) и др. Указывается также значение критерия (Value) и логическая операция (Logical) для объединения простых условий в сложные: AND (И), OR (ИЛИ).

Вкладка **Filter** позволяет установить условия выборки - критерий отбора данных в запрос, поле или выражение, по которому будет происходить выборка: =, ==, LIKE (похожий), >, <, >=, <=, BETWEEN (между), IN (в) и др. Указывается также значение критерия (Value) и логическая операция (Logical) для объединения простых условий в сложные: AND (И), OR (ИЛИ). В выражении можно указать все стандартные функции Visual FoxPro, являющиеся функциями языка программирования xBase, а также собственные функции SQL для работы со столбцами.

Вкладка **Order By** (упорядочить по) позволяет указать поля, по которым необходимо выполнить упорядочение данных, а также тип упорядочивания: Ascending (по возрастанию), Descending (по убыванию).

Вкладка **Group By** (группировать по) используется для задания поля, по которому необходимо выполнить группировку. Кнопка Having этой вкладки используется для выбора способа группировки, указания имени поля, из которого будут выбираться данные для группировки и различных условий группировки.

Вкладка **Miscellaneous** позволяет задать дополнительные условия отбора (не допускать повторения) и установить количество записей попадающих в выборку.

Если щёлкнуть правой кнопкой мыши в окне Конструктора запросов, то появится контекстно зависимое

меню, где можно:

-выбрав команду Run Query, запустить на выполнение подготовленный запрос;

- командой View SQL просмотреть созданный запрос на языке SQL;

- командой Remove Table вернуть назад выбранную таблицу;

- командой Add Table добавить таблицу в окно Конструктора запросов;

- командой Output Settings установить получателя результатов запроса;

- командой Help просмотреть справочную информацию.

Запрос можно сохранить в файле с расширением qpr, используя в меню File (Файл) команду Save as (Сохранить как). Сохранённый запрос можно запустить командой: Do <имя запроса>.qpr из командного окна или из программного файла.

С помощью запросов можно просматривать, анализировать и изменять данные из нескольких таблиц. Они также используются в качестве источника данных для форм и отчетов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего используется язык управления данными SQL, его основное назначение?
2. Какие категории операций выделяют в стандартном SQL?
3. Какие существуют способы реализации SQL?
4. Какие типы данных и операции над данными стандартизованы в SQL?
5. Какие собственные функции в операциях над данными стандартизованы в SQL?
6. В чем заключаются основные функции сервера баз данных?
7. Что собой представляет SQL-сервер и какие СУБД относятся к SQL-серверам?

8. Какие команды используются для определения данных в SQL?
9. Какие команды используются для манипулирования данными в SQL?
10. Какие команды используются для формирования запросов в SQL?
11. Что собой представляют транзакции, в чем состоит их основное назначение, какие требования предъявляются к транзакциям в технологии баз данных?
12. Что собой представляют журналы транзакций и для чего они используются в СУБД?
13. Какие команды используются для управления транзакциями в SQL?
14. В чем состоит особенность избирательного управления обеспечения безопасности данных?
15. Что собой представляет право доступа и для чего оно используется в СБД, какие существуют виды прав доступа?
16. Какие команды используются для предоставления прав доступа к данным и для аннулирования прав доступа к данным в SQL?

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Задание для самостоятельной работы ориентировано на реализацию SQL в СУБД Visual FoxPro.

1. Составьте SQL - команду, позволяющую создать структуру файла таблицы данных с именем KADR1, содержащую следующие поля: FAM (фамилия) – символьного типа (20 позиций); SEM – (семейное положение) – логического типа; DET (количество детей) – числового типа (2 позиции).

2. Составьте команду, позволяющую создать структуру файла таблицы данных с именем KADR2, содержащую следующие поля: FAM (фамилия) – символьного типа (15 позиций); TAB (табельный номер) – числового типа (4

позиции); PER (перемещение по службе) – символьного типа (40 позиций), LAB (лаборатория) – символьного типа (15 позиций).

3. Составьте SQL команду, добавляющую в конец существующего файла таблицы данных KADR1 следующую запись: "Юн" (поле FAM (фамилия) – символьного типа), 'T' (true - истина), (поле SEM – семейное положение, логического типа), 2 (поле DET – количество детей, числового типа)

4. Составьте команду, добавляющую в конец существующего файла таблицы данных KADR2 следующую запись: 'Исин' (поле FAM – фамилия, символьного типа), 5 (поле TAB – табельный номер, числового типа), '15.04.97 переведен в лабораторию ТО' (поле PER - примечания).

5. Составьте SQL-команду добавления записи с данными о студенте Ахметове, получающего стипендию 1400 тенге в конец файла таблицы данных с именем STUD, содержащего следующие поля: поле FAM (фамилия) - символьного типа; поле GODR (дата рождения) - типа даты; поле STIP (стипендия) - числового типа.

6. Составьте команду вывода фамилий студентов (поле FAM), стипендии (поле STIP) из таблицы данных STUD в массив FS.

7. Составьте команду вывода фамилии (поле FAM) и группы (поле GRUPPA) из таблицы данных STUD в порядке возрастания групп и фамилий в курсор CUR1.

8. Составьте команду вывода минимального, максимального и среднего значения стипендии студентов (поле STIP) из таблицы данных STUD на экран.

9. Составьте команду вывода из таблицы данных KADR2 фамилий (поле FAM) и названий лабораторий (поле LAB) в новую таблицу базы данных KDR.

10. Составьте команду вывода из таблицы данных KADR1 фамилий (поле FAM) и количество детей (поле DET) с другими именами полей (Фамилии и Дети).

11. Составьте команду вывода из таблицы данных

KADR2 фамилий сотрудников (поле FAM), работающих в лаборатории "ТО" (LAB = "ТО"), в текстовый файл.

12. Составьте команду вывода из таблицы данных KADR2 фамилий сотрудников (поле FAM), работавших или работающих в лаборатории "ИТ". Поиск ведется в поле PER (сведения о перемещениях по службе).

13. Составьте команду вывода из таблицы данных KADR2 названий лабораторий (поле LAB) и количества сотрудников, сгруппированных по лабораториям, для лабораторий с количеством сотрудников больше 5.

14. Составьте команду вывода из таблицы данных KADR2 и KADR1 названий лабораторий (поле LAB) и суммарного количества детей сотрудников (поле DET) по каждой лаборатории.

15. Составьте команду вывода из таблицы данных KADR1 (псевдоним A) фамилий сотрудников (поле FAM), а из таблицы данных KADR2 (псевдоним B) – названий лабораторий, где они работают (поле LAB), для семейных сотрудников.

16. Составьте команду вывода из таблицы данных KADR1 фамилий сотрудников с указанием их семейного положения (поля FAM и SEM, в качестве псевдонима взять имя таблицы данных), а из таблицы данных KADR2 – их табельных номеров и названий лабораторий (поля TAB и LAB, в качестве псевдонима взять имя таблицы данных) для записей, у которых совпадают фамилии.

17. Составьте команду пометки на удаление данных о сотрудниках из таблицы данных KADR1, если эти сотрудники работают в лаборатории 'ИТ' (поле LAB таблицы KADR2).

18. Составьте команду вывода из таблиц базы данных KADR1 и KADR2 фамилий тех сотрудников (поле FAM), кто работает в лаборатории 'ВТ' (поле LAB таблицы KADR2) или имеет более трех детей (поле DET таблицы KADR1).

ЛИТЕРАТУРА

1. Дейт К. Введение в системы баз данных. / Пер.с англ. - М.: Вильямс, 2001.
2. Каратыгин С.А., Тихонов А.Ф., Тихонова Л.И. Visual FoxPro 7. Руководство пользователя с примерами. - М.: Бином, 2002.
3. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. /Пер. с англ.- М.: Вильямс, 2003.
4. Сичкаренко В.А. SQL-99. Руководство разработчика баз данных. - СПб: ДиаСофт, 2002.
5. Омельченко Л. Самоучитель Visual FoxPro 7.0. - СПб.: БХВ - Санкт-Петербург, 2002.
6. Мамаев Е.В. Microsoft SQL Server. - СПб.: БХВ-Санкт-Петербург, 2001..
7. Базиян М. И др. Использование Visual FoxPro 6. Пер. с англ. / Учебное пособие. - М.: Вильямс, 2000.
8. Ю.Тихомиров. Microsoft SQL Server 7.0. Разработка приложений: - СПб.: БХВ-Санкт-Петербург, 1998
9. Джудит С. Боуман и др. Практическое руководство по SQL; пер. с англ. Диалектика, 1997.

ЗАКЛЮЧЕНИЕ

Широкое внедрение информационных систем обусловило необходимость знания и практического применения технологии БД. А применение клиент-серверной технологии и использование SQL-серверов для хранения многопользовательских данных требует знания языка SQL - стандартного языка взаимодействия с реляционными БД.

Данное учебное пособие предназначено для изучения языка SQL и ориентировано на повышение эффективности самостоятельной работы, может использоваться как студентами, так и всеми, желающими самостоятельно овладеть языком SQL.

ПРИЛОЖЕНИЕ

Задание на лабораторную работу

Лабораторная работа выполняется в среде любой реляционной СУБД, поэтому для выполнения лабораторной работы необходимо повторить основной материал по языку управления данными SQL, запустить выбранную СУБД, выполнить следующее задание:

1.Создать структуру таблицы, модифицировать структуру таблицы (добавить столбец, изменить имя столбца, изменить параметры столбца), используя команды SQL.

2.Используя SQL-команды дополнения и модификации данных внести данные в созданную таблицу (не менее 10 записей). После проверки содержимого таблицы, SQL-командой изменить значения нескольких полей (не менее 2-х) в нескольких записях.

3.Получить из таблицы справочную и статистическую информацию (не менее 10 типов), используя простые запросы, запросы с условиями, запросы с группировкой данных, запросы с подзапросом, объединенные запросы, отправляя результат выборки в таблицу базы данных, текстовый файл, в массив, на принтер и т.д.

Например, получить количество записей в таблице, отвечающих простым и сложным условиям; записи из таблицы, отвечающие определенным условиям с записью их в массив; минимальное, максимальное, среднее арифметическое значение и сумму значений по числовому столбцу с группировкой данных по категориям, удовлетворяющие заданным условиям, присутствующие в другой таблице (запрос с подзапросом); объединить записи из двух таблиц в одну новую таблицу по условию и т.д.

4.Удалите командой SQL часть записей таблицы, удовлетворяющих определенным условиям.

5.Удалите командой SQL часть столбцов таблицы.

6.Удалите командой SQL таблицу данных.