Уважаемые студенты выполните следующие задания

1. Лекция рассчитана на 4 часа: на 18.01.23 и 20.01.23. Повторить ранее изученный материал, изучить новый теоретический материал, кратко законспектировать.

2. Фотоотчет присылать на электронную почту в трехдневный срок

С уважением, Хвастова Светлана Ивановна

!!! Если возникнут вопросы обращаться по телефону 0721389311.

Электронная почта: xvsviv@rambler.ru

Сортировка результатов запроса. Итоговые функции и средства группировки данных в SQL.

1. SQL – структурированный язык запросов

SQL (Structured Query Language) – структурированный язык запросов – является инструментом, предназначенным для выборки и обработки информации, содержащейся в компьютерной базе данных. SQL является языком программирования, применяемым для организации взаимодействия пользователя с базой данных (см. *рис.* 1). SQL работает только с реляционными базами данных и предоставляет пользователю следующие функциональные возможности:

– изменение структуры представления данных;

– выборка данных из базы данных;

– обработка базы данных, т. е. добавление новых данных, изменение, удаление имеющихся данных;

– управление доступом к базе данных;

– совместное использование базы данных пользователями, работающими параллельно;

– обеспечение целостности базы данных.

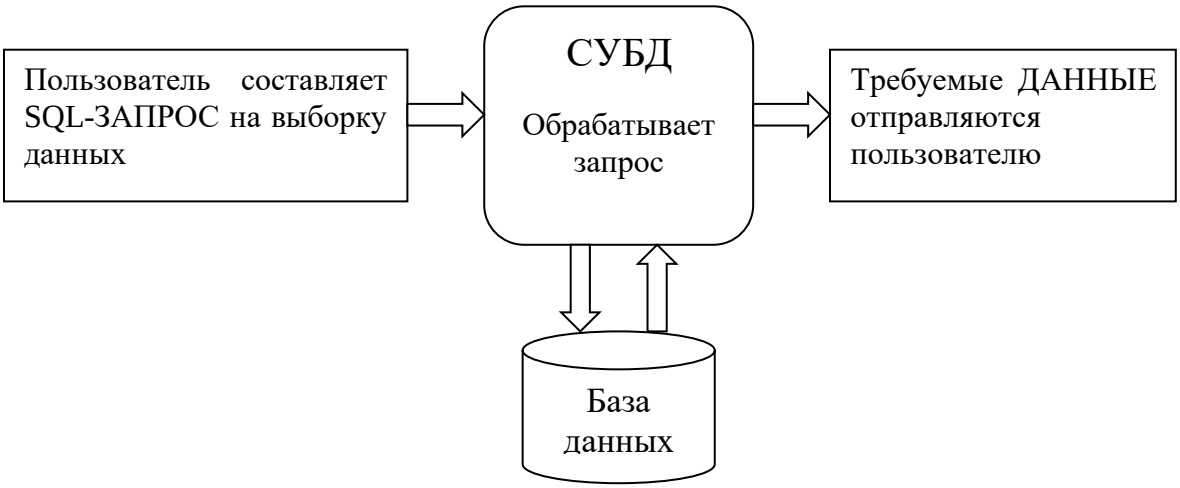

Рис. 44. Доступ к БД с помощью SQL

SQL – это не полноценный компьютерный язык типа PASCAL, C++, JAVA. Еще раз отметим, что SQL, также как и QBE, является непроцедурным языком. С помощью SQL описываются свойства и взаимосвязи сущностей (объектов, переменных и т. п.), но не алгоритмы решения задачи. Он не содержит условных операторов, операторов цикла, организации подпрограмм, ввода-вывода и т. п. В связи с этим SQL автономно не используется. Инструкции SQL встраиваются в программу, написанную на традиционном языке программирования, и дают возможность получить доступ к базам данных (*встроенный SQL*). Кроме того, из таких языков, C, C++, JAVA инструкции SQL можно посылать СУБД в явном виде, используя интерфейс вызовов функций.

Язык SQL является многофункциональным языком. Во-первых, SQL используется в качестве языка интерактивных запросов пользователей с целью выборки данных и в качестве встроенного языка программирования баз данных. Кроме того, SQL используется в качестве языка администрирования БД для определения структуры базы данных и управления доступом к данным, находящимся на сервере; в качестве языка создания приложений клиент/сервер, доступа к данным в среде Internet, распределенных баз данных.

С помощью SQL можно *динамически* изменять и расширять структуру

базы данных даже в то время, когда пользователи работают с ее содержимым. Таким образом, SQL обеспечивает максимальную гибкость. *Статические* языки определения данных запрещают доступ к БД во время изменения ее структуры

Официальный стандарт языка SQL был опубликован ANSI и ISO в 1986 г. В дальнейшем, он был расширен стандартами SQL-89 (1989 г.) и SQL-92 (1992 г.). Действующая версия стандарта SQL:1999 была принята ANSI и ISO в конце 1999 г. В настоящее время ведется работа над стандартом для SQL3, содержащим объектно-ориентированные расширения.

Кроме перечисленных выше версий языка SQL для универсальных ЭВМ существует множество версий типа "клиент-сервер", а также версий SQL для персональных компьютеров.

2. Основные инструкции языка SQL

Основные задачи, решаемые средствами языка SQL – манипулирование различными объектами базы данных (таблицами, индексами, представлениями и т. д.) и манипулирование данными, хранящимися в таблицах базы данных. В связи с этим, язык SQL принято делить на две части: язык определения данных DDL и язык манипулирования данными DML. Основные инструкции языка SQL представлены в *таблице 1*.

При описании синтаксиса инструкций будем использовать следующие правила:

– каждая инструкция начинается с *команды* – ключевого слова, описывающего действие, выполняемое инструкцией (например, CREATE – создать, DELETE – удалить и т. д.);

– после команды следует одно или несколько *предложений*, описывающих данные, с которыми работает инструкция, или содержащих дополнительную информацию о действии, выполняемом инструкцией. Каждое предложение начинается с ключевого слова, например, WHERE (где),

FROM (откуда), INTO (куда), HAVING (имеющий);

– в квадратные скобки "**[…]**" заключены необязательные элементы;

– вертикальная черта "|" , разделяющая два элемента, указывает на то, что в инструкции используется либо один элемент, либо второй;

– в фигурные скобки "{…}"заключаются элементы, разделенные вертикальной чертой;

– троеточие означает, что далее в инструкции либо следует выражение, либо повторяются элементы, указанные перед тремя точками.

Таблица 1 – *Инструкции языка SQL*

Вид	Название	Назначение
DDL	CREATE TABLE	Добавление новой таблицы в БД
	DROP TABLE	Удаление таблицы
	ALTER TABLE	Изменение структуры таблицы
	CREATE INDEX	Создание индекса для столбца
	DROP INDEX	Удаление индекса столбца
	CREATE VIEW	Создание нового представления
	DROP VIEW	Удаление представления
	GRAND	Назначение привилегий доступа пользователей к БД
	REVOKE	Удаление привилегий
	CREATE SCHEMA	Добавление новой схемы в БД
	DROP SCHEMA	Удаление схемы
DML	SELECT	Выборка данных из таблицы
	UPDATE	Обновление данных в таблице
	INSERT	Вставка новых строк в таблицу
	DELETE	Удаление строк из таблицы

Рассмотрим основные инструкции языка SQL.

Инструкция *создания таблицы* имеет формат вида:

CREATE TABLE <имя таблицы>

(<имя столбца> <тип данных> [NOT NULL]

[,<имя столбца> <тип данных> [NOT NULL]]...)

После выполнения инструкции появляется новая таблица, которой присваивается имя, указанное в инструкции. Имя таблицы (как и имена других

объектов – столбцов и пользователей) согласно стандарту ANSI/ISO должны содержать от 1 до 18 символов, начинаться с буквы и не содержать пробелов или специальных символов пунктуации (на практике поддержка имен в различных СУБД реализована по-разному).

Обязательными операндами данной инструкции являются имя создаваемой таблицы и имя хотя бы одного столбца с указанием типа данных, хранимых в этом столбце. SQL поддерживает различные типы данных: целые числа, числа с плавающей запятой, строки символов, значения даты и времени и др. В общем случае в современных СУБД могут использоваться самые разнообразные дополнительные типы данных, расширяющие базовый набор SQL.

При создании таблицы для отдельных столбцов могут указываться некоторые дополнительные правила контроля вводимых в них значений. Конструкция – NOT NULL (не пустое) служит именно таким целям и для столбца таблицы означает, что в этом столбце должно быть непустое (определенное) значение.

Пример 1. Создание таблицы

Пусть требуется определить таблицу ORDERS и ее столбцы. Инструкция для определения таблицы может иметь следующий вид:

CREATE TABLE ORDERS (ORDER_NUM INTEGER NOT NULL, CUST_NUM INTEGER NOT NULL, PROD_ID INTEGER NOT NULL, QTY INTEGER NOT NULL, DATE_ORDER DATE NOT NULL)

где INTEGER обозначает тип данных целое число, а DATE – тип данных, обозначающих значения даты.

В процессе работы у пользователей возникает необходимость добавить в таблицу информацию. Инструкция *изменения структуры таблицы* имеет формат вида:

ALTER TABLE <имя таблицы>

{ADD|ALTER|DROP} <имя столбца> [<тип данных>]
[NOT NULL]
[,{ADD|ALTER|DROP} <имя столбца> [<тип данных>]
[NOT NULL],..]

Изменение структуры таблицы может состоять в добавлении (ADD), изменении (ALTER) или удалении (DROP) одного или нескольких столбцов таблицы.

Пример 2. Добавление столбца таблицы

Добавим в созданную ранее таблицу CUST столбец CUST_PHN, содержащий телефоны клиентов. Для этого следует записать инструкцию вида:

ALTER TABLE CUST ADD CUST_PHN CHAR (10)

Пример 3. Удаление столбца

Удалим из таблицы PROD столбец STORE. (Примеры рассматриваются без учета ограничений целостности).

ALTER TABLE PROD DROP STORE

Инструкция *удаления* таблицы имеет формат вида:

DROP TABLE <имя таблицы>

Например, для удаления таблицы с именем SALE достаточно записать оператор вида:

DROP TABLE SALE

Одним из структурных элементов физической памяти является *индекс*. Индекс – это средство, обеспечивающее быстрый доступ к строкам таблицы

на основе значений одного или нескольких столбцов. В индексе хранятся значения данных и указатели на строки, где эти данные встречаются. Данные в индексе располагаются в убывающем или в возрастающем порядке, чтобы СУБД могла быстро найти требуемое значение. Затем по указателю СУБД сможет быстро определить строку, содержащую требуемое значение. Инструкция *создания индекса* имеет формат вида:

CREATE [UNIQUE] INDEX <имя индекса> ON <имя таблицы> (<имя столбца> [ASC| DESC]

[,<имя столбца> [ASC| DESC],...)

Оператор позволяет создать индекс для одного или нескольких столбцов заданной таблицы с целью ускорения выполнение запросных и поисковых операций с таблицей. Для одной таблицы можно создать несколько индексов.

Необязательная опция UNIQUE обеспечивает запрет задания совпадающих значений для индекса. По существу, создание индекса с указанием признака UNIQUE означает определение ключа в созданной ранее таблице.

При создании индекса можно задать порядок автоматической сортировки значений в столбцах – в порядке возрастания ASC (по умолчанию), или в порядке убывания DESC. Для разных столбцов можно задавать различный порядок сортировки.

Пример 4. Создание индекса

Пусть из таблице CUST часто извлекаются данные по названию фирм-клиентов. Можно создать индекс **main_index** для сортировки названий фирм-клиентов в алфавитном порядке по возрастанию. Оператор создания индекса может иметь вид:

CREATE INDEX main_index ON CUST (CUST_NAME)

Инструкция *удаления индекса* имеет формат вида:

DROP INDEX<имя индекса>

Эта инструкция позволяет удалять созданный ранее индекс с соответствующим именем. Так, например, для уничтожения индекса main_index к таблице CUST достаточно записать инструкцию **DROP INDEX main_index**.

Кроме таблиц и индексов существуют другие объекты базы данных, например, представления. Представление является "виртуальной" таблицей, содержащей набор столбцов одной или нескольких таблиц. Однако, в отличие от таблицы, представление как совокупность значений в базе данных реально не существует. Представление определяется как запрос на выборку данных, которому присвоили имя и сохранили в базе данных. Представление позволяет пользователю увидеть результаты сохраненного запроса.

Инструкция *создания представления* имеет формат вида:

CREATE VIEW<имя представления>
[(<имя столбца> [,<имя столбца>]…)]
AS <оператор SELECT>

При необходимости можно задать имя для каждого столбца создаваемого представления. Список имен столбцов должен содержать столько элементов, сколько столбцов содержится в запросе. Если список имен в инструкции отсутствует, то каждый столбец представления получает имя соответствующего столбца запроса.

Пример 5. Создание представления

Необходимо создать представление с именем CUSTINF таблицы CUST, включающее только названия клиентов и их номера.

CREATE VIEW CUSTINF

AS SELECT CUST_NUM, CUST_NAME

FROM CUST

Создать представление ORDER_CUS, показывающее заказы сделанные клиентом 3105.

CREATE VIEW ORDER_CUS

AS SELECT

FROM ORDERS

WHERE CUST_NUM=3105

Инструкция *удаления представления* имеет формат вида**:**

DROP VIEW <имя представления>

Оператор позволяет удалить созданное ранее представление. Заметим, что при этом таблицы, участвующие в запросе, удалению не подлежат. Удаление представления **ORDER_CUS** производится инструкцией вида:

DROP VIEW ORDER_CUS

Представления широко применяются для ограничения доступа пользователей ко всей информации в таблицах базы данных.

Пример 6. Использование инструкции GRAND и REVOKE

Можно определить права доступа к таблицам базы данных с помощью инструкций GRAND и REVOKE. Например, инструкция

GRAND INSERT

ON CUST

TO PETROV

разрешает сотруднику Петрову ввод данных в таблицу CUST.

Следующая инструкция отменяет привилегии сотрудника Иванова на

изменение данных о клиентах и чтение информации о них

REVOKE UPDATE, SELECT

ON CUST

TO IVANOV

Запросы являются фундаментом SQL. Многие разработчики используют SQL исключительно в качестве инструмента для создания запросов. Поэтому важнейшей инструкцией является инструкция SELECT, которая используется для построения SQL-запросов:

SELECT [ALL | DISTINCT]<список данных>

FROM <список таблиц>

[WHERE <условие отбора>]

[GROUP BY <имя столбца> [,<имя столбца>]...]

[HAVING <условие поиска>]

[ORDER BY <спецификация> [,<спецификация>]...]

Оператор SELECT позволяет производить выборку и вычисления над данными из одной или нескольких таблиц. В предложении SELECT указывается список возвращаемых столбцов, разделенных запятыми. Для каждого элемента из списка в *ответной* таблице будет создан один столбец. Ответная таблица может иметь (ALL), или не иметь (DISTINCT) повторяющиеся строки. По умолчанию в ответную таблицу включаются все строки, в том числе и повторяющиеся. Список данных может содержать выражения над столбцами, показывающие, что наряду с выборкой данных выполняются вычисления, результаты которого попадают в новый (создаваемый) столбец ответной таблицы

В отборе данных участвуют записи одной или нескольких таблиц, перечисленных в списке предложения FROM. Такие таблицы называют *исходными таблицами запроса*.

При использовании в списках данных имен столбцов нескольких таблиц

для указания принадлежности столбца некоторой таблице применяют конструкцию вида: <имя таблицы>.<имя столбца>.

Предложение WHERE задает условия, которым должны удовлетворять строки в результирующей таблице. Вслед за ключевым словом WHERE указывается логическое выражение <условие отбора>. Его элементами могут быть имена столбцов, операции сравнения, арифметические операции, логические операции (И, ИЛИ, НЕТ), скобки, специальные функции LIKE и т.д.

Предложение GROUP BY содержит список столбцов, которые используются для группировки строк. *Группой* называются строки с совпадающими значениями в столбцах, перечисленных за ключевыми словами GROUP BY. Для сгруппированных данных можно использовать статистические функции: AVG (среднее значение в группе), MAX (максимальное значение в группе), MIN (минимальное значение в группе), SUM (сумма значений в группе), COUNT (число значений в группе).

Вслед за предложением HAVING указывается логическое выражение <условия поиска>, определяющее, какие из отобранных и сгруппированных строк будут отображаться в результирующем наборе данных. Правила записи аналогичны правилам формирования <условия отбора> предложения WHERE.

Предложение ORDER BY задает порядок сортировки результирующего множества строк. Обычно каждая <спецификация> аналогична соответствующей конструкции оператора CREATE INDEX и представляет собой конструкцию вида: <имя столбца> [ASC | DESC].

Пример 7. Выбор строк

Пусть требуется вывести названия товаров и их цену. Инструкцию выбора можно записать следующим образом:

```
SELECT PROD_NAME, PRICE
FROM PROD
```

Пример 8. Выбор с условием

Вывести названия товаров, цена которых больше 100$. Инструкцию SELECT для этого запроса можно записать так:

```
SELECT PROD_NAME
FROM PROD
WHERE PRICE>100
```

Пример 9. Выбор с сортировкой

Строки результатов запроса, как и строки таблицы базы данных, не имеют определенного порядка. Включив в инструкцию SELECT предложение ORDER BY, можно отсортировать результаты запроса.

Пусть требуется вывести название клиентов и годовой объем заказов. Последний столбец отсортировать по возрастанию. По умолчанию данные сортируются по возрастанию.

```
SELECT CUST_NAME, CUST_SUM
FROM CUST
ORDER BY CUST_SUM
```

Пример 10. Получение итоговых данных

Каков средний объем заказов?

Этот запрос обеспечивает вычисление среднего объема заказов, используя данные из таблицы CUST

```
SELECT AVG (CUST_SUM)
FROM CUST
```

Пример 11. Вычисляемые столбцы

Например, требуется вычислить стоимость остатков товара, хранящихся на складе. Данные вывести по каждому товару. Значения вычисляемых столбцов определяются на основе выражения, указанного в списке возвращаемых столбцов.

```
SELECT PROD_NAME, PRICE, STORE, (PRICE* STORE)
```

FROM PROD

Пусть требуется увеличить цену каждого товара на 5%. Запрос можно сформулировать следующим образом:

SELECT PROD_NAME, PRICE, (PRICE*1.05)

FROM PROD

Во многих СУБД реализованы дополнительные арифметические операции, операции над строками, встроенные функции для работы со значениями даты и времени.

Например, требуется вывести номер заказа, месяц и год его поставки. Запрос выглядит следующим образом:

SELECT ORDER_NUM, MONTH (DATE_ORDER), YEAR (DATE_ORDER)

FROM ORDERS

Пример 12. Выбор всех столбцов

Иногда требуется получить содержимое всех столбцов таблицы. С учетом этого в SQL разрешается использовать вместо списка возвращаемых столбцов символ "*".

SELECT *

FROM ORDERS

Пример 13. Повторяющиеся строки

Результаты запроса могут содержать повторяющиеся строки. Например, требуется вывести номера клиентов, сделавших заказы, из таблицы ORDERS. Клиенты 3101, 3105, 3103 сделали более, чем по одному заказу, поэтому строки с их номерами будут повторяться. Для исключения повторяющихся строк используется предикат DISTINCT.

SELECT DISTINCT CUST_NUM

FROM ORDERS

Пример 14. Выбор с группированием

Пусть требуется найти минимальное и максимальное заказанное количество для каждого из видов товаров. Оператор SELECT для этого запроса имеет вид:

```
SELECT PROD_ID, MIN (QTY ), MAX (QTY )
FROM ORDERS
GROUP BY PROD_ID
```

Запрос выполняется следующим образом: заказы делятся на группы, по одной группе на каждый товар. Затем для каждой группы вычисляется максимальное и минимальное значения столбца QTY по всем строкам, входящим в группу, и генерируется одна итоговая строка результатов.

Пример 15. Условия отбора групп

Предложение HAVING можно использовать для отбора групп строк, участвующих в запросе. Пусть требуется найти максимальное заказанное количество каждого товара. Общее количество заказанного товара не должно превышать 20.

```
SELECT PROD_ID, MAX (QTY )
FROM ORDERS
GROUP BY PROD_ID
HAVING SUM (QTY )<=20
```

При реализации данного запроса, вначале заказы разделяются на группы по видам товаров. Затем исключаются те группы, в которых общее количество заказанного товара превышает 20. И после этого в оставшихся группах определяется максимальное заказанное количество каждого товара.

Пример 16. Многотабличные запросы

На практике, многие запросы считывают информацию сразу из

нескольких таблиц базы данных. Например, необходимо вывести список всех заказов, а также название клиента, сделавшего заказ. Инструкция SELECT должна содержать условие отбора, которое определяет связь между столбцами таблиц ORDERS и CUST.

```
SELECT ORDER_NUM, CUST_NAME, PROD_ID, QTY, DATE_ORDER
FROM ORDERS, CUST
WHERE CUST.CUST_NUM=ORDERS.CUST_NUM
```

Приведенный запрос отличается от предыдущих, во-первых, тем, что предложение FROM содержит не одну, а две таблицы. Во-вторых, в условии отбора WHERE CUST.CUST_NUM=ORDERS.CUST_NUM сравниваются столбцы из двух различных таблиц. Эти столбцы называются *связанными*.

В данном примере, столбцы, используемые из разных таблиц, имеют одинаковые имена. В этом случае необходимо указывать полные имена столбцов, которые однозначно определяют их местонахождение.

Инструкция на *изменения строк* имеет формат вида:

```
UPDATE <имя таблицы>
SET <имя столбца> = {<выражение> | NULL}
[,SET <имя столбца> = {<выражение> | NULL}... ]
[WHERE <условие>]
```

Инструкция UPDATE обновляет значения в определенных предложением SET столбцах таблицы для тех строк, которые удовлетворяют условию, заданному предложением WHERE.

Новые значения столбцов могут быть пустыми (NULL), либо вычисляться в соответствии с арифметическим выражением.

Пример 17. Изменение строк

Пусть необходимо увеличить на 15% цену только тех товаров, которые

стоят меньше 100$. Запрос, сформулированный с помощью оператора UPDATE, может выглядеть так:

UPDATE PROD

SET PRICE=(PRICE*1.15)

WHERE PRICE <=100

Инструкция для *вставки новых строк* имеет форматы двух видов:

INSERT INTO <имя таблицы> [(<список столбцов>)] VALUES (<список значений>)

и

INSERT INTO <имя таблицы> [(<список столбцов>)]

<предложение SELECT>

В первом формате оператор INSERT предназначен для одной строки с заданными значениями в столбцах. Порядок перечисления имен столбцов должен соответствовать порядку значений, перечисленных в списке предложения VALUES. Если <список столбцов> опущен, то в <списке значений> должны быть перечислены все значения в порядке столбцов структуры таблицы.

Во втором формате оператор INSERT предназначен для ввода в заданную таблицу новых строк, отобранных из другой таблицы с помощью предложения SELECT.

Пример 18.

Ввести в таблицу CUST строку, содержащую сведения о новом клиенте. Для этого можно записать инструкцию такого вида:

INSERT INTO CUST

VALUES ("3110", "ЧП Иванов П.Т.", NULL)

Пример 19.

Требуется скопировать в новую таблицу OLDORDERS сведения о старых заказах.

INSERT INTO OLDORDERS (ORDER_NUM, CUST_NUM, PROD_ID, QTY, DATE_ORDER)

SELECT ORDER_NUM, CUST_NUM, PROD_ID, QTY, DATE_ORDER

FROM ORDERS

WHERE DATE_ORDER< '17.01.03'

Инструкция *удаления строк* имеет формат вида:

DELETE FROM <имя таблицы> [WHERE <условие>]

Результатом выполнения оператора DELETE является удаление из указанной таблицы строк, которые удовлетворяют условию, определенному предложением WHERE. Если необязательный операнд WHERE опущен, т. е. условие отбора удаляемых записей отсутствует, удалению подлежат все записи таблицы.

Пример 20. Удаление строк

Удалить из таблицы ORDERS сведения о старых заказах.

DELETE FROM ORDERS

WHERE DATE_ORDER<'17.01.03'