Кафедра ИВС

ЛЕКЦИЯ

Тема: «Язык реляционных баз данных SQL. Функции и основные возможности»

Дисциплина «Базы данных в ИС»

ОП «Информационные системы»

Авторы: ст.преп.каф.ИВС Саданова Б.М.
ст.преп.каф.ИВС Олейникова А.В.

План лекции:

1. История развития SQL
2. Структура SQL
3. Типы данных SQL
4. Создание базы данных
5. Создание (удаление) таблиц базы данных
6. Создание (удаление) индексов
7. Выборка данных

Цель лекции: дать общую характеристику операторов языка SQL и показать, как записываются основные запросы к базе данных на языке SQL .

История развития SQL

- **SQL (Structured Query Language)** — Структурированный Язык Запросов — стандартный язык запросов по работе с реляционными БД.
- Язык SQL появился после реляционной алгебры, и его прототип был разработан в конце 70-х годов в компании IBM Research.
- В дальнейшем этот язык применялся во многих коммерческих СУБД и в силу своего широкого распространения постепенно стал стандартом "де-факто" для языков манипулирования данными в реляционных СУБД.

Структура SQL

- Многие нереляционные системы также имеют в настоящее время средства доступа к реляционным данным.
- Целью стандартизации является *переносимость приложений* между различными СУБД.
- В отличие от реляционной алгебры, где представлены только операции запросов к БД, SQL является полным языком, в нем присутствуют не только операции запросов, но и операторы, соответствующие **Data Definition Language (DDL) — языку описания данных**. Кроме того, язык содержит операторы, предназначенные для управления (администрирования) БД.

Структура SQL

- SQL может использоваться как *интерактивный* (для выполнения запросов) и как *встроенный* (для построения прикладных программ). В нем существуют:
  - предложения определения данных (определение баз данных, определение и уничтожение таблиц и индексов);
  - предложения на выбор данных;
  - предложения модификации данных (добавление, удаление и изменение данных);
  - предложения управления данными (предоставление и отмена привилегий на доступ к данным, управление транзакциями и др).

Структура SQL

SQL выполняет в этих предложениях:

- арифметические *вычисления*, обработку *текстовых строк* и *сравнения* значений арифметических выражений и текстов;
- *упорядочение* строк и столбцов при выводе на печать или экран;
- создание *представлений* (виртуальных таблиц), позволяющих пользователям иметь свой взгляд на данные без увеличения их объема в БД;
- запоминание выводимого по запросу содержимого таблицы в другой таблице (реляционная операция *присваивания*).
- *агрегирование данных*: группирование данных, применение операций Среднее, Сумма, Максимум, Минимум, Число элементов и т.п.

Структура SQL

Существуют 3 группы операторов SQL:

- Операторы DDL (Data Definition Language) - операторы определения объектов базы данных
- Операторы DML (Data Manipulation Language) - операторы манипулирования данными
- Операторы защиты и управления данными

Операторы DDL (Data Definition Language):

- CREATE DATABASE - создать базу данных
- DROP DATABASE - удалить базы данных
- CREATE TABLE - создать таблицу
- ALTER TABLE - изменить таблицу
- DROP TABLE - удалить таблицу
- CREATE DOMAIN - создать домен
- ALTER DOMAIN - изменить домен
- DROP DOMAIN - удалить домен
- CREATE VIEW - создать представление
- DROP VIEW - удалить представление

Операторы DML (Data Manipulation Language):

- SELECT - отобрать строки из таблиц
- INSERT - добавить строки в таблицу
- UPDATE - изменить строки в таблице
- DELETE - удалить строки в таблице
- COMMIT - зафиксировать внесенные изменения
- ROLLBACK - откатить внесенные изменения

Операторы защиты и управления данными:

- GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами
- REVOKE - отменить привилегии пользователя или приложения

Типы данных SQL

- INTEGER - целое число;
- SMALLINT - "короткое целое";
- DECIMAL(p,q) - десятичное число, имеющее p цифр (0 < p < 16) и знак; q - число цифр справа от десятичной точки;
- FLOAT - вещественное число с 15 значащими цифрами;
- CHAR(n) - символьная строка фиксированной длины из n символов (0 < n < 256);
- VARCHAR(n) - символьная строка переменной длины, не превышающей n символов (n > 0 и разное в разных СУБД, но не меньше 4096);

Типы данных SQL

- DATE - дата в формате, определяемом специальной командой (по умолчанию mm/dd/yy;
- TIME - время в формате, определяемом специальной командой, (по умолчанию hh.mm.ss);
- DATETIME - комбинация даты и времени;
- MONEY - деньги в формате, определяющем символ денежной единицы ($, руб, ...) и его расположение (суффикс или префикс), точность дробной части и условие для показа денежного значения.

В предложениях SQL могут использоваться следующие обозначения:

- звездочка (*) для обозначения "все" -, т.е. "все случаи, удовлетворяющие определению";
- квадратные скобки ([]) –конструкции, заключенные в эти скобки, являются необязательными (т.е. могут быть опущены);
- фигурные скобки ({}) –конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы, т.е. они позволяют уточнить порядок разбора синтаксических конструкций, заменяя обычные скобки, используемые в синтаксисе SQL;

В предложениях SQL могут использоваться следующие обозначения:

- многоточие (...) – указывает на то, что непосредственно предшествующая ему синтаксическая единица факультативно может повторяться один или более раз;
- прямая черта (|) –наличие выбора из двух или более возможностей. Например, ASC|DESC указывает, что можно выбрать один из терминов ASC или DESC; если один из элементов выбора заключен в квадратные скобки, то он выбирается по умолчанию ([ASC]|DESC означает, что отсутствие всей этой конструкции будет восприниматься как выбор ASC);
- точка с запятой (;) – завершающий элемент предложений SQL;

В предложениях SQL могут использоваться следующие обозначения:

- запятая (,) – используется для разделения элементов списков;
- пробелы ( ) – могут вводиться для повышения наглядности между любыми синтаксическими конструкциями предложений SQL;
- прописные жирные латинские буквы и символы – используются для написания конструкций языка SQL;
- строчные буквы – используются для написания конструкций, которые должны заменяться конкретными значениями, выбранными пользователем;

Создание базы данных

Предложение ***CREATE DATABASE*** создает БД и, возможно, журнал транзакций на указанных устройствах и в размере.

- CREATE DATABASE *база_данных_имя*[ON [DEFAULT| *база_данных_устройство*][= *размер*][,*база_данных_устройство*][= *размер*]]...]LOG ON *база_данных_устройство*[=*размер*][,*база_данных_устройство*[=*размер*]...][FOR LOAD]

Пример создания базы данных с именем MyDB, содержащей файл с данными MyDBroot и файл журнала транзакций Logdata1

```
CREATE DATABASE  MyDB
ON
(NAME="MyDBroot",
FILENAME="c:\mssql2k\MSSQL\data\mydbroot.mdf",
SIZE=8MB,
MAXSIZE=9MB,
FILEGROWTH=100KB),
LOG ON
(NAME="Logdata1",
FILENAME="e:\log_files\logdata1.ldf",
SIZE=1000MB,
MAXSIZE=1500MB,
FILEGROWTH=100MB)
```

Описание параметров оператора **CREATE DATABASE**:

- ON – ключевое слово, указывает, что в команде должны быть заданы расположение файлов данных, их имена, объем и величина объема приращения;
- NAME – логическое имя файла, по которому происходит обращение к этому файлу со стороны SQL Server;
- FILENAME – физическое имя файла с указанием полного пути с обязательным указанием расширения файла;
- SIZE – исходный объем в мегабайтах. Необязателен.
- FILEGROWTH – приращение объема файла после его заполнения. Приращение можно указать в мегабайтах или процентах от текущего объема;
- LOG ON – ключевое слово, указывает, что в команде должны быть заданы расположение файлов журнала, их имена, объем и величина объема приращения;

Создание (удаление) таблиц базы данных

Базовые таблицы создаются в SQL с помощью предложения ***CREATE TABLE***.

CREATE TABLE *базовая_таблица_имя*

(столбец тип_данных

[,столбец тип_данных] ...);

где столбец - логическое имя столбца создаваемой таблицы,

тип_данных должен принадлежать к одному из типов данных, поддерживаемых СУБД.

CREAT TABLE - выполняемое предложение. Если его ввести с терминала, система построит таблицу, которая сначала будет пустой: она будет содержать только строку заголовков столбцов, но не будет еще содержать никаких строк с данными.

Пример создания таблицы с именем Сотрудники.

```
CREATE TABLE Сотрудники
                (Код          SMALLINT,
                 ФИО          CHAR (70),
                Адрес         CHAR  (15),
                Телефон       CHAR (10),
                Дата рожд.    DATE,
                Город         CHAR (10),
                РНН           CHAR  (10));
```

Существующую таблицу можно уничтожить с помощью предложения ***DROP TABLE*** (уничтожить таблицу):

DROP TABLE базовая_таблица;

DROP TABLE Сотрудники;

Создание (удаление) индексов

- Для построения индекса в SQL существует предложение ***CREATE INDEX*** (создать индекс).
- **Индекс** - это системная таблица, построенная по значениям заданного столбца заданной таблицы.
- В индексе размещается перечень уникальных значений указанного столбца таблицы со ссылками на те ее строки, где встречаются эти значения (структура, похожая на предметный указатель книги).

- Пример: в БД существует таблица *НАЧИСЛЕНИЯ*, в которой хранятся сведения о заработной плате сотрудников предприятия

ФИО сотрудника	Месяц	Сумма
Иванов	1	10000
Петров	1	12000
Сидоров	1	10000
Иванов	2	10000
Сидоров	2	10000
Иванов	3	10000
Петров	3	12000

- *Индекс*, построенный для столбца ФИО сотрудника таблицы НАЧИСЛЕНИЯ будет содержать следующие сведения:

Значения столбца	Строки, в которых встречается такое значение		
Иванов	1	4	6
Петров	2	7	
Сидоров	3	5	

Синтаксис предложения CREATE INDEX

CREATE [UNIQUE][CLUSTERED|NONCLUSTERED]INDEX *имя_индекса* ON [*база_данных*.]*базовая_таблица*(*имя_столбца*[[ASC] | DESC] [,*имя_столбца* [[ASC] | DESC]]]...)[WITH][FILLFACTOR = *x*] [ON *имя_сегмента*]

- CLUSTERED - кластеризованный индекс, т.е. такой индекс, при котором в листьях B-дерева, образующего индекс, находятся не ссылки на данные, а собственно страницы данных.
- FILLFACTOR - позволяет управлять заполнением страниц B-дерева индекса, задается в процентах, 100% - полное заполнение.
- UNIQUE (уникальный) - никаким двум строкам в индексируемой базовой таблице не позволяется принимать одно и то же значение для индексируемого столбца

Пример: в БД существуют таблицы:

1. *БЛЮДА* со столбцами БЛ (код блюда) и Основа.
2. *СОСТАВ* со столбцами БЛ(код блюда), ПР (код продукта), ВЕС

Индексы для столбцов БЛ и Основа таблицы Блюда создаются с помощью предложений:

```
CREATE UNIQUE INDEX Блюда_БЛ ON Блюда (БЛ);
CREATE INDEX Блюда_Основа ON Блюда (Основа);
```

Индекс для первичного ключа (столбцы БЛ и ПР) таблицы Состав – создается с помощью предложения:

```
CREATE UNIQUE INDEX Состав_БЛ_ПР ON Состав (БЛ, ПР);
```

Выборка данных (предложение SELECT)

SELECT [[ALL] | DISTINCT]{ * | элемент_select [,элемент_select] ...}

FROM базовая_таблица | представление [псевдоним]

[,базовая_таблица | представление [псевдоним]] ...

[WHERE фраза]

[GROUP BY фраза [HAVING фраза]];

где элемент_select - это одна из следующих конструкций:

[таблица.]* | [таблица.]столбец | SQL_функция | переменная | (выражение) | системная_переменная

- SELECT - (выбрать) данные из указанных столбцов
- FROM - (из) перечисленных таблиц
- WHERE - (где) строки из указанных таблиц должны удовлетворять условию отбора строк
- GROUP BY - (группируя по) указанному перечню столбцов с тем, чтобы получить для каждой группы единственное агрегированное значение, используя во фразе SELECT SQL-функции SUM (сумма), COUNT (количество), MIN (минимальное значение), MAX (максимальное значение) или AVG (среднее значение)
- ORDER BY - а затем упорядочить результаты выбора данных. При этом упорядочение можно производить в порядке возрастания - ASC (ASCending) или убывания DESC (DESCending), по умолчанию принимается ASC.
- HAVING - (имея) в результате лишь те группы, которые удовлетворяют условию отбора групп

- Кроме операторов сравнения (= | <> | < | <= | > | >=) в WHERE фразе используются условия BETWEEN (между), LIKE (похоже на), IN (принадлежит), IS NULL (не определено) и EXISTS (существует), которые могут предваряться оператором NOT (не).
- Критерий отбора строк формируется из одного или нескольких условий, соединенных логическими операторами:

- AND - должны выполняться оба условия;
- OR - должно выполняться одно из условий;
- AND NOT - когда должно удовлетворяться первое условие и не должно второе;
- OR NOT - когда или должно удовлетворяться первое условие или не должно удовлетворяться второе, существует приоритет AND над OR (сначала выполняются все операции AND и только после этого операции OR)

Пусть в базе данных имеются три таблицы:

- **Товары**

Код товара	Наименование	Ед_изм	Категория_товара
1	Товар_1	шт	электроника
2	Товар_2	шт	мебель
3	Товар_3	литр	напитки
4	Товар_4	кг	мясные изделия
5	Товар_5	шт	книги

- **Поставщики**

Код поставщика	Наименование	Номер счета	Банк
1	Поставщик_1	1111	Народный
2	Поставщик_2	2222	Альянс
3	Поставщик_3	3333	Народный
4	Поставщик_4	4444	Альянс
5	Поставщик_5	5555	Народный

- **Движение товара**

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
01.01.06	1	1	2	100	10
10.01.06	2	5	3	50	20
20.01.06	3	1	2	100	10
30.01.06	4	2	1	50	15
01.03.06	5	3	4	100	25
10.03.06	6	4	3	200	20
20.03.06	7	1	2	10	12
30.03.06	8	2	1	50	15
01.10.06	9	3	4	10	25
10.10.06	10	5	3	20	20
20.10.06	11	1	2	100	12

Простая выборка

Запрос: выдать название и категории товаров

```
SELECT   Наименование, Категория_товара
FROM     Товары;
```

Результат:

Наименование	Категория_товара
Товар_1	электроника
Товар_2	мебель
Товар_3	напитки
Товар_4	мясные изделия
Товар_5	книги

Запрос для получения полной информации о поставщиках:

```
SELECT Код поставщика, Наименование, Номер счета, Банк
FROM Поставщики;
```

или использовать его более короткую нотацию:

```
SELECT *
FROM Поставщики;
```

«Звездочка» (*) служит кратким обозначением всех имен полей в таблице, указанной во фразе FROM. При этом порядок вывода полей соответствует порядку, в котором эти поля определялись при создании таблицы.

Результат запроса:

Код поставщика	Наименование	Номер счета	Банк
1	Поставщик_1	1111	Народный
2	Поставщик_2	2222	Альянс
3	Поставщик_3	3333	Народный
4	Поставщик_4	4444	Альянс
5	Поставщик_5	5555	Народный

- Выдать список банков, в которых обслуживаются поставщики товаров:

```
SELECT Банк
FROM Поставщики;
```

Банк
Народный
Альянс
Народный
Альянс
Народный

- Для исключения дубликатов и одновременного упорядочения строк необходимо дополнить запрос ключевым словом DISTINCT (различный, различные):

```
SELECT DISTINCT Банк
FROM Поставщики;
```

Банк
Альянс
Народный

Выборка вычисляемых значений

- В фразе SELECT могут содержаться не только перечень столбцов таблицы или символ *, но и выражения.
- Пример запроса для получения стоимости поставленных товаров по накладной:

```
SELECT № накладной, (Количество*Цена)
FROM Движение товара;
```

Результат выборки:

№ накладной	Количество*Цена
1	1000
2	100
3	1000
4	750
5	2500
6	4000
7	120
8	750
9	250
10	400
11	1200
12	75

Выборка вычисляемых значений

- Фраза SELECT может включать не только выражения, но и отдельные числовые или текстовые константы.
- Текстовые константы заключаются в апострофы (').

```
SELECT № накладной, 'Стоимость =',
(Количество*Цена)
FROM Движение товара;
```

Результат выборки:

№ накладной		Количество*Цена
1	Стоимость =	1000
2	Стоимость =	100
3	Стоимость =	1000
4	Стоимость =	750
5	Стоимость =	2500
6	Стоимость =	4000
7	Стоимость =	120
8	Стоимость =	750
9	Стоимость =	250
10	Стоимость =	400
11	Стоимость =	1200
12	Стоимость =	75

Выборка с использованием фразы WHERE

- Во фразе WHERE для отбора нужных строк таблицы можно использовать операторы сравнения = (равно), <> (не равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), которые могут предваряться оператором NOT, создавая, например, отношения "не меньше" и "не больше"
- Запрос для получения перечня накладных, выписанных организаций с кодом 1:

```
SELECT  № накладной, Дата
FROM    Движение товара
WHERE  Код поставщика = 1;
```

№ накладной	Дата
1	01.01.06
3	20.01.06
7	20.03.06
11	20.10.06

Выборка с использованием фразы WHERE

- Возможность использования нескольких условий, соединенных логическими операторами AND, OR, AND NOT и OR NOT, позволяет осуществить более детальный отбор строк.
- Запрос для получения перечня накладных, содержащих товары стоимостью менее 20 тенге и количеством>=50 :

```
SELECT № накладной, Дата. Количество, Цена
FROM   Движение товара
WHERE  Цена<20 AND Количество>=50;
```

Результат :

№ накладной	Дата	Количество	Цена
1	01.01.06	100	10
3	20.01.06	100	10
4	30.01.06	50	15
8	30.03.06	50	15
11	20.10.06	100	12

Использование BETWEEN

- С помощью BETWEEN ... AND ... можно отобрать строки, в которых значение какого-либо столбца находится в заданном диапазоне.
- Например, выдать перечень накладных, в которых значение цены товаров находится в диапазоне от 10 до 15:

```
SELECT № накладной, Дата, Количество, Цена
FROM Движение товара
WHERE Цена BETWEEN 10 AND 15;
```

№ накладной	Дата	Количество	Цена
1	01.01.06	100	10
3	20.01.06	100	10
4	30.01.06	50	15
7	20.03.06	10	12
8	30.03.06	50	15
11	20.10.06	100	12
12	30.10.06	5	15

Использование IN

- Оператор IN определяет набор значений, в которые данное значение может или не может быть включено.
- Выдать сведения о накладных, по которым были поставлены товары с кодами 1 и 2:

```
SELECT *
FROM   Движение товара
WHERE  Код товара IN (1, 2);
```

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
01.01.06	1	1	2	100	10
20.01.06	3	1	2	100	10
30.01.06	4	2	1	50	15
20.03.06	7	1	2	10	12
30.03.06	8	2	1	50	15
20.10.06	11	1	2	100	12
30.10.06	12	2	1	5	15

Использование LIKE

- LIKE применим только к полям типа CHAR или VARCHAR, с которыми он используется чтобы находить подстроки.
- LIKE ищет поле символа чтобы видеть, совпадает ли с условием часть его строки. В качестве условия он использует групповые символы - специальные символы которые могут соответствовать чему-нибудь.
- Имеются два типа групповых символов в LIKE:

✓ _ (подчеркивание) – заменяет любой одиночный символ,

✓ % (процент) – заменяет любую последовательность из N символов (где N может быть нулем),

Использование LIKE

- Синтаксис "имя_столбца LIKE текстовая_константа" для столбца текстового типа позволяет отыскать все значения указанного столбца, соответствующие образцу, заданному "текстовой_константой".
- Выдать список организаций, в наименовании которых присутствует сочетание символов «оставщик»

```
SELECT Наименование
FROM   Поставщики
WHERE      Наименование LIKE '%оставщик%';
```

Наименование
Поставщик_1
Поставщик_2
Поставщик_3
Поставщик_4
Поставщик_5

Выборка с упорядочением

- Таблицы - это неупорядоченные наборы данных, и данные, которые выходят из них, не обязательно появляются в какой-то определенной последовательности.
- SQL использует команду ORDER BY для упорядочивания списка вывода.
- Многочисленные столбцы упорядочиваются один внутри другого и можно определять возрастание (ASC) или убывание (DESC) для каждого столбца. По умолчанию установлено - возрастание.

Выборка с упорядочением

- Выдать перечень накладных в порядке убывания количества поставленных по ним товаров

```
SELECT *
FROM Движение товаров
ORDER BY Количество DESC;
```

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
10.03.06	6	4	3	200	20
01.01.06	1	1	2	100	10
20.01.06	3	1	2	100	10
01.03.06	5	3	4	100	25
20.10.06	11	1	2	100	12
10.01.06	2	5	3	50	20
30.01.06	4	2	1	50	15
30.03.06	8	2	1	50	15
10.10.06	10	5	3	20	20
20.03.06	7	1	2	10	12
01.10.06	9	3	4	10	25
30.10.06	12	2	1	5	15

Агрегирование данных

- Запросы могут производить обобщенное групповое значение полей с помощью *агрегатных функций*:

  COUNT - количество значений в столбце,

  SUM - сумма значений в столбце,

  AVG - среднее значение в столбце,

  MAX - самое большое значение в столбце,

  MIN - самое малое значение в столбце.

- Для функций SUM и AVG рассматриваемый столбец должен содержать числовые значения.
- Аргументу всех функций, кроме COUNT(*), может предшествовать ключевое слово DISTINCT (различный), указывающее, что дублирующие значения должны быть исключены перед тем, как будет применяться функция.
- Функция COUNT(*) служит для подсчета всех без исключения строк в таблице (включая дубликаты).

Агрегирование данных

- Выдать данные о количестве поставленного товара и его средней цене на товар с кодом=1:

```
SELECT  SUM(Количество),AVG(Цена)
FROM    Движение товара
WHERE   Код товара=1;
```

SUM(Количество)	AVG(Цена)
105	15

```
SELECT  MAX (Количество*Цена)
FROM    Движение товара;
```

Количество*Цена
4000

Фраза GROUP BY

- Предложение GROUP BY позволяет определять подмножество значений в особом поле в терминах другого поля, и применять функцию агрегата к подмножеству. Это дает возможность объединять поля и агрегатные функции в едином предложении SELECT.
- Пусть требуется вычислить общую массу каждого из поставленных товаров.

```
SELECT  Код товара, SUM(Количество)
FROM    Движение товара
GROUP  BY Код товара;
```

Код товара	SUM(Количество)
2	310
3	270
1	105
4	110

Фраза GROUP BY

- Фраза GROUP BY не предполагает ORDER BY. Чтобы упорядочить результат следует дать запрос

```
SELECT  Код товара, SUM(Количество)
FROM    Движение товара
GROUP   BY Код товара;
ORDER   BY Код товара;
```

Код товара	SUM(Количество)
1	105
2	310
3	270
4	110

Фраза HAVING

- Фраза HAVING играет такую же роль для групп, что и фраза WHERE для строк: она используется для исключения групп, так же, как WHERE используется для исключения строк.
- Эта фраза включается в предложение лишь при наличии фразы GROUP BY, а выражение в HAVING должно принимать единственное значение для группы.
- Выдать список кодов поставщиков, которые поставили товары более чем по двум накладным:

```
SELECT Код поставщика, COUNT(№ накладной)
FROM Движение товара
GROUP      BY Код поставщика
HAVING     COUNT(№ накладной)>2;
```

Код поставщика	COUNT(№ накладной)
1	4
2	3

Запросы с использованием нескольких таблиц

- SQL обладает механизмом для одновременной или последовательной обработки данных из нескольких взаимосвязанных таблиц.
- Получить наименование товаров и наименование категории товаров, поставляемых поставщиком с кодом=1:

```
SELECT Наименование, Категория товара, Количество, Цена
FROM Товары, Движение товара
WHERE Товары.Код товара = Движение товара. Код товара
AND Код поставщика = 1;
```

Запросы с использованием нескольких таблиц

Результат:

Наименование	Категория товара	Количество	Цена
Товар_2	мебель	100	10
Товар_2	мебель	100	10
Товар_2	мебель	10	12
Товар_2	мебель	100	12

- Выборка получена следующим образом: СУБД последовательно формирует строки декартова произведения таблиц, перечисленных во фразе FROM, проверяет, удовлетворяют ли данные сформированной строки условиям фразы WHERE, и если удовлетворяют, то включает в ответ на запрос те ее поля, которые перечислены во фразе SELECT.

Вложенные подзапросы

- *Вложенный подзапрос* - это подзапрос, заключенный в круглые скобки и вложенный в WHERE (HAVING) фразу предложения SELECT или других предложений, использующих WHERE фразу.
- Вложенный подзапрос может содержать в своей WHERE (HAVING) фразе другой вложенный подзапрос и т.д.
- Существуют *простые* и *коррелированные* вложенные подзапросы. Они включаются в WHERE (HAVING) фразу с помощью условий IN, EXISTS или одного из условий сравнения (= | <> | < | <= | > | >=).

Вложенные подзапросы

- Простые вложенные подзапросы обрабатываются *"снизу вверх".* Первым обрабатывается вложенный подзапрос самого нижнего уровня. Множество значений, полученное в результате его выполнения, используется при реализации подзапроса более высокого уровня и т.д.
- Запросы с коррелированными вложенными подзапросами обрабатываются в *обратном порядке*. Сначала выбирается первая строка рабочей таблицы, сформированной основным запросом, и из нее выбираются значения тех столбцов, которые используются во вложенном подзапросе. Если эти значения удовлетворяют условиям вложенного подзапроса, то выбранная строка включается в результат. Затем выбирается вторая строка и т.д., пока в результат не будут включены все строки, удовлетворяющие вложенному подзапросу.

Простые вложенные подзапросы

- Простые вложенные подзапросы используются для представления множества значений, исследование которых должно осуществляться в каком-либо предикате IN.
- Выдать название поставщиков и банков, в которых они обслуживаются, поставивших товары по цене 20 тенге.

```
SELECT  Наименование, Банк
FROM    Поставщики
WHERE   Код поставщика IN
                (SELECT Код поставщика
                 FROM   Движение товара
                 WHERE  Цена = 20 );
```

Простые вложенные подзапросы

Наименование	Банк
Поставщик_5	Народный
Поставщик_4	Альянс
Поставщик_5	Народный

- При обработке полного запроса система выполняет прежде всего вложенный подзапрос. Этот подзапрос выдает множество номеров поставщиков, которые поставляют товары по цене 20 тенге, а именно множество (5,4,5). Поэтому первоначальный запрос эквивалентен такому простому запросу:

```
SELECT  Название, банк
FROM    Поставщики
WHERE   Код поставщика IN (4,5);
```

Коррелированные вложенные подзапросы

- Выдать название поставщиков и банков, в которых они обслуживаются, поставивших товары по цене 20 тенге.

```
SELECT      Наименование, Банк
FROM Поставщики
WHERE       20 IN
                (SELECT Код поставщика
                 FROM       Движение товара
                 WHERE  Код  поставщика  =  Движение товара.Код поставщика
                 AND        Движение товара.Цена = 20  );
```

Предложения модификации данных SQL

Модификация данных может выполняться с помощью предложений

- DELETE (удалить)
- INSERT (вставить)
- UPDATE (обновить).

Они могут оперировать как базовыми таблицами, так и представлениями.

Предложение DELETE

Предложение DELETE имеет формат:

```
DELETE
FROM   базовая таблица | представление
[WHERE фраза];
```

и позволяет удалить содержимое всех строк указанной таблицы (при отсутствии WHERE фразы) или тех ее строк, которые выделяются WHERE фразой.

Предложение DELETE

Запрос «Удалить все мясные блюда»

```
DELETE FROM Блюда
WHERE  Основа = 'Мясо';
```

Удаление с вложенным подзапросом «Удалить все поставки для поставщика из Алматы»

```
DELETE
FROM Поставки
WHERE Код_Поставщика IN
    (SELECT Код_Поставщика     FROM    Поставщики
     WHERE   Город = ‘Алматы');
```

Предложение INSERT

Предложение INSERT имеет один из форматов:

1. INSERT

INTO {базовая таблица | представление} [(столбец [,столбец] ...)]

VALUES ({константа | переменная} [,{константа | переменная}] ...);

или

2. INSERT

INTO {базовая таблица | представление} [(столбец [,столбец] ...)]

подзапрос;

Предложение INSERT

1. В первом формате в таблицу вставляется строка со значениями полей, указанными в перечне фразы VALUES (значения), причем i-е значение соответствует i-му столбцу в списке столбцов.
2. Во втором формате сначала выполняется подзапрос, т.е. по предложению SELECT в памяти формируется рабочая таблица, а потом строки рабочей таблицы загружаются в модифицируемую таблицу. При этом i-й столбец рабочей таблицы (i-й элемент списка SELECT) соответствует i-му столбцу в списке столбцов модифицируемой таблицы.

Предложение INSERT

Вставка единственной записи в таблицу

Добавить в таблицу Блюда блюдо Шашлык (Код_Блюда - 34, Блюдо - Шашлык, Основа - Мясо, Выход - 150)

```
INSERT
INTO  Блюда (Код_Блюда, Блюдо, Основа, Выход)
VALUES       (34, 'Шашлык', 'Мясо', 150);
```

Создается новая запись для блюда с номером 34.

Предложение UPDATE

Предложение UPDATE имеет один из форматов:

1. UPDATE (базовая таблица | представление}
   SET столбец = значение [, столбец = значение] ...
   [WHERE фраза]

где значение - это

столбец | выражение | константа | переменная

и может включать столбцы лишь из обновляемой таблицы, т.е. значение одного из столбцов модифицируемой таблицы может заменяться на значение ее другого столбца или выражения, содержащего значения нескольких ее столбцов, включая изменяемый.

При отсутствии WHERE фразы обновляются значения указанных столбцов во всех строках модифицируемой таблицы. WHERE фраза позволяет сократить число обновляемых строк, указывая условия их отбора.

Предложение UPDATE

2. Второй формат описывает предложение, позволяющее производить обновление значений модифицируемой таблицы по значениям столбцов из других таблиц.

```
UPDATE  {базовая таблица | представление}
SET    столбец = значение [, столбец = значение] ...
FROM    {базовая таблица | представление} [псевдоним],
        {базовая таблица | представление} [псевдоним]
      [,{базовая таблица | представление} [псевдоним]] ...
[WHERE  фраза]
```

Предложение UPDATE

Пример обновления множества записей

«Утроить цену всех продуктов таблицы поставки (кроме цены кофе – Код_Продукта = 17)»

```
UPDATE Поставки
SET    Цена = Цена * 3
WHERE   Код_Продукта  <> 17;
```

Пример обновления с подзапросом

«Установить равной нулю цену и К_во продуктов для поставщиков из Алматы и Тараз.

```
UPDATE Поставки
SET    Цена = 0, К_во = 0
WHERE  Код_Поставщика IN
        (SELECT Код_Поставщика  FROM Поставщики
                WHERE  Город IN (‘Алматы', ‘Тараз'));
```

Безопасность и санкционирование доступа

- В контексте баз данных термин *безопасность* означает защиту данных от несанкционированного раскрытия, изменения или уничтожения.
- SQL позволяет индивидуально защищать как целые таблицы, так и отдельные их поля.
- Для этого имеются две возможности:

✓ **механизм представлений**, используемый для скрытия засекреченных данных от пользователей, не обладающих правом доступа;

✓ **подсистема санкционирования доступа**, позволяющая предоставить указанным пользователям привилегии на доступ к данным и дать им возможность передавать часть привилегий другим пользователям, отменяя впоследствии эти привилегии, если потребуется.

Безопасность и санкционирование доступа

- Обычно при установке СУБД в нее вводится какой-то идентификатор, который должен далее рассматриваться как идентификатор наиболее привилегированного пользователя - **системного администратора**.
- Системный администратор может создавать базы данных и имеет все привилегии на их использование. Эти привилегии или их часть могут предоставляться другим пользователям.
- В свою очередь, пользователи, получившие привилегии от системного администратора, могут передать их (или их часть) другим пользователям, которые могут их передать следующим и т.д.

Безопасность и санкционирование доступа

Привилегии предоставляются с помощью предложения GRANT:

GRANT привилегии ON объект TO пользователи;

- где "привилегии" – это список, состоящий из одной или нескольких привилегий, разделенных запятыми, либо фраза ALL PRIVILEGES (все привилегии);
- "объект" - имя и, если надо, тип объекта (база данных, таблица, представление, индекс и т.п.);
- "пользователи" - список, включающий один или более идентификаторов санкционирования, разделенных запятыми, либо специальное ключевое слово PUBLIC (общедоступный).

Безопасность и санкционирование доступа

- К таблицам и представлениям относятся привилегии SELECT, DELETE, INSERT и UPDATE [(столбцы)], позволяющие считывать (выполнять любые операции, в которых используется SELECT), удалять, добавлять или изменять строки указанной таблицы
- Изменение можно ограничить конкретными столбцами. Например, предложение

  GRANT SELECT, UPDATE (Труд) ON Блюда TO cook;

позволяет пользователю, который представился системе идентификатором cook, использовать информацию из таблицы Блюда, но изменять в ней он может только значения столбца Труд.

Безопасность и санкционирование доступа

- Если пользователь USER_1 предоставил какие-либо привилегии другому пользователю USER_2, то он может впоследствии отменить все или некоторые из этих привилегий.
- Отмена осуществляется с помощью предложения REVOKE:

  REVOKE привилегии ON объект FROM пользователи;

Например, можно отобрать у пользователя cook право изменения значений столбца Труд:

REVOKE UPDATE (Труд) ON Блюда FROM cook;

Контрольные вопросы:

1. Какие из операторов относятся к языку определения данных (DDL)?
2. Какие из операторов относятся к языку манипулирования данными (DML)?
3. Какие операторы и операнды могут использоваться при формировании условия выборки записей?
4. После какого служебного слова в операторе SELECT указывается выбор столбцов?
5. После какого служебного слова в операторе SELECT указывается выбор строк?
6. С помощью какого служебного слова можно задать отбор диапазона значений?
7. С помощью какого служебного слова можно найти подстроки?

Список литературы:

1. Т. Конноли. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.: Пер.с англ. – М.: Изд.дом «Вильямс», 2012. – 1440 с.
2. К.Дейт. Введение в системы БД. изд.6-е. :Пер.с англ. - М.: Изд.дом "Вильямс", 2014. – 1200с.
3. Д.Ульман. Введение в системы баз данных. – М.: Издательство «Лори», 2015. – 853с.