Запрос

Каждая СУБД должна предоставлять возможность чтения, добавления, изменения и удаления записей из БД со стороны пользователя СУБД (чаще всего пользователем СУБД выступают информационные системы), если, конечно, у него есть **соответствующие права**. Для достижения этой цели в СУБД предусмотрен **язык запросов**, представляющий набор основных правил, которым должны следовать все запросы к СУБД.

Вид запросов, главным образом, зависит от модели данных, которая используется в СУБД. Если модель данных поддерживает определенный порядок элементов (например “модель инвертированных таблиц”, “иерархическая модель”), то все запросы к СУБД начинаются с операции позиционирования (необходимо явно переместиться на определенный объект БД). Если же модель данных состоит из объектов, не поддерживающих порядок, как, например, **“реляционная модель данных”**, в которой все таблицы представляют собой множество кортежей, то операция позиционирования не требуется и можно писать запросы в декларативном виде.

Язык SQL

Обычно языки БД разделяются на два вида:

1. DDL (Data definition language) - язык определения схемы базы данных.
2. DML (Data manipulation language) - язык манипулирования данными.

Язык SQL объединяет в себе обе функции, т.е. язык SQL используется как для определения схемы БД, так и для написания запросов.

Определения БД

Основным элементом определения схемы БД в языке SQL является оператор CREATE TABLE, который создаёт новую таблицу. При создании новой таблицы можно указать имена столбцов, типы значений, которые будут храниться в этих столбцах (

целочисленный тип, тип с плавающей точкой, тип строки, тип даты и т.д.). Также для каждого столбца можно указать значение по умолчанию, обозначить соответствующий столбец как первичный ключ, или как внешний [1]. Общий вид операции создания новой таблицы представлен ниже:

```
CREATE TABLE имяТаблицы (
имяСтолбца1 тип1 [дополнительныеУсловия1],
имяСтолбца2 тип2 [дополнительныеУсловия2],
имяСтолбца3 тип3 [дополнительныеУсловия3],
 ....
);
```

В качестве примера демонстрируется создания таблицы служащих, в которой содержится номер паспорта (целочисленное, первичный ключ), имя (строка) и номер отдела, в котором работает служащий (целочисленное, внешний ключ к столбцу **Номер** таблицы **Отделы.**

```
CREATE TABLE Служащие (
Номер_паспорта int PRIMARY KEY,
Имя varchar(255),
Номер_отдела int FOREIGN KEY REFERENCES Отделы(Номер)
);
```

Манипулирования данными

SELECT

Основным элементом манипулирования данными языка SQL является оператор выборки, общая форма которого представлена ниже [2]:

```
SELECT [DISTINCT] списокСтолбцов1
FROM списокСсылокНаТаблицы
[WHERE условноеВыражение1]
[GROUP BY списокСтолбцов2]
[HAVING условноеВыражение2]
[ORDER BY списокСтолбцов3]
```

На данном изображение разделы, помещенные в квадратные скобки являются необязательными и их можно пропустить.

Выборка данных производиться из одной или нескольких таблиц, представленных в разделе **FROM.**

```
SELECT attribute
FROM table;
```

Если таблиц больше одной, то, перед выполнением выборки, они объединяются в одну путём операции расширенного **Декартового умножения**.

```
SELECT attribute1, attribute2
FROM table1, table2;
```

Также таблицы в разделе FROM могут быть полученные в результате применения другого оператора SELECT.

На следующем шаге происходит фильтрация на основе условия, определённого в разделе **WHERE**. При пропуске раздела WHERE фильтрации не происходит и все выбранные строки переходят дальше. Ниже показано использование WHERE в запросе, получающий имена всех служащих из таблицы “Служащие”, которые имеют зарплату более 40000 рублей.

```
SELECT Имя
FROM Служащие
WHERE Зарплата > 40000;
```

Раздел **GROUP BY** позволяет группировать результаты по указанным атрибутам. При его использовании в разделе SELECT разрешается использовать только имена тех столбцов, по которым осуществляется группировка, а также агрегатные функции:

- COUNT(*) - подсчитать число элементов внутри группы
- MIN(столбец) - минимальное значение столбца внутри группы
- MAX(столбец) - максимальное значение столбца внутри группы
- и другие

Раздел **HAVING** действует аналогично разделу WHERE, только условия применяются не к отдельным строкам, а к всей группе. Если раздел HAVING присутствует, а раздел GROUP BY нет, то вся таблица воспринимается как одна группа.

В качестве примера использования разделов GROUP BY и HAVING, демонстрируется запрос, получающий список отделов фирмы , в которых работают не менее 10 служащих:

```
SELECT Номер_отдела
FROM Отделы JOIN Служащие ON Отделы.Номер_отдела =
Служащие.Номер_отдела
GROUP BY Номер_отдела
HAVING count(*) >= 10;
```

Раздел **ORDER BY** позволяет указать порядок сортировки значений в результирующей таблице. Можно выполнять сортировку по нескольким столбцам с указанием порядка сортировки(от большего к меньшему / от меньшего к большему).

INSERT

Операция вставки нового значения в таблицу осуществляется с помощью оператора **INSERT INTO**, для которого необходимо указать имя таблицы, список столбцов, в которые будет происходить вставка значений, а также сами значения вставки [3]. Общий вид операции вставки представлен ниже:

```
INSERT INTO имяТаблицы (столбец1, столбец2, столбец2, ...)
VALUES (значение1, значение2, значение3, ...);
```

В качестве примера приводится операция вставки нового служащего в таблицу Служащие:

```
INSERT INTO Служащие (Номер_паспорта, Имя, Номер_отдела)
VALUES (314159, “Иванов Иван Иванович”, 2718);
```

DELETE

Для удаления данных из таблицы необходимо использовать оператор **DELETE FROM**, общий вид которого представлен ниже [4]:

```
DELETE FROM имяТаблицы WHERE условие;
```

После выполнения данного запроса, из таблицы **“имяТаблицы”** будут удалены все записи, для которые было выполнено условие в разделе WHERE.

QBE

Теоретическую основу языка QBE составляет реляционное исчисление доменов. С помощью языка QBE можно создавать сложные запросы к базе данных, заполняя предлагаемую СУБД форму запроса. Данный способ создания запросов позволяет получить высокую наглядность и не требует указывать алгоритм выполнения операции. Каждая современная реляционная СУБД содержит свой вариант QBE.

В качестве использования QBE, демонстрируется запрос на получение имён всех служащих из таблицы **Служащие**, которые работают в отделе **8128**:

Номер_паспорта	Имя	Номер_отдела
	P.	8128

В данном примере демонстрируется следующие:

1. Для того, чтобы включить столбец в результирующую таблицу, необходимо вписать в соответствующие поле строку **“P.”** (первая буква слова Print)
2. Есть необходимо отфильтровать результаты по значению в определённом столбце, необходимо просто указать в соответствующие поле необходимое значение.
3. Есть столбец не используется в фильтрации и не включается в результирующую таблицу, то соответствующие ему поле просто пропускается.

Источники

1. SQL CREATE TABLE Statement: https://www.w3schools.com/sql/sql_create_table.asp
2. SQL SELECT Statement: https://www.w3schools.com/sql/sql_select.asp
3. SQL DELETE Statement: https://www.w3schools.com/sql/sql_delete.asp
4. SQL UPDATE Statement: https://www.w3schools.com/sql/sql_update.asp