КАФЕДРА: КИИ

ТЕМА: ОСНОВЫ ЯЗЫКА ЗАПРОСОВ SQL

ДИСЦИПЛИНА: ВВЕДЕНИЕ В БАЗЫ ДАННЫХ

ДЛЯ СТУДЕНТОВ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ
6B06105 «DATA SCIENCE»

СТАРШИЙ ПРЕПОДАВАТЕЛЬ КЛЮЕВА Е.Г.

ПЛАН ЛЕКЦИИ

- Язык структурированных запросов (SQL);
- Создание базы данных, таблиц и индексов;
- Определение ограничений для таблиц БД;
- Предложения модификации данных;
- Предложения выборки данных;
- Выборка с упорядочением и агрегированием данных;
- Запросы с использованием нескольких таблиц.

ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ (SQL)

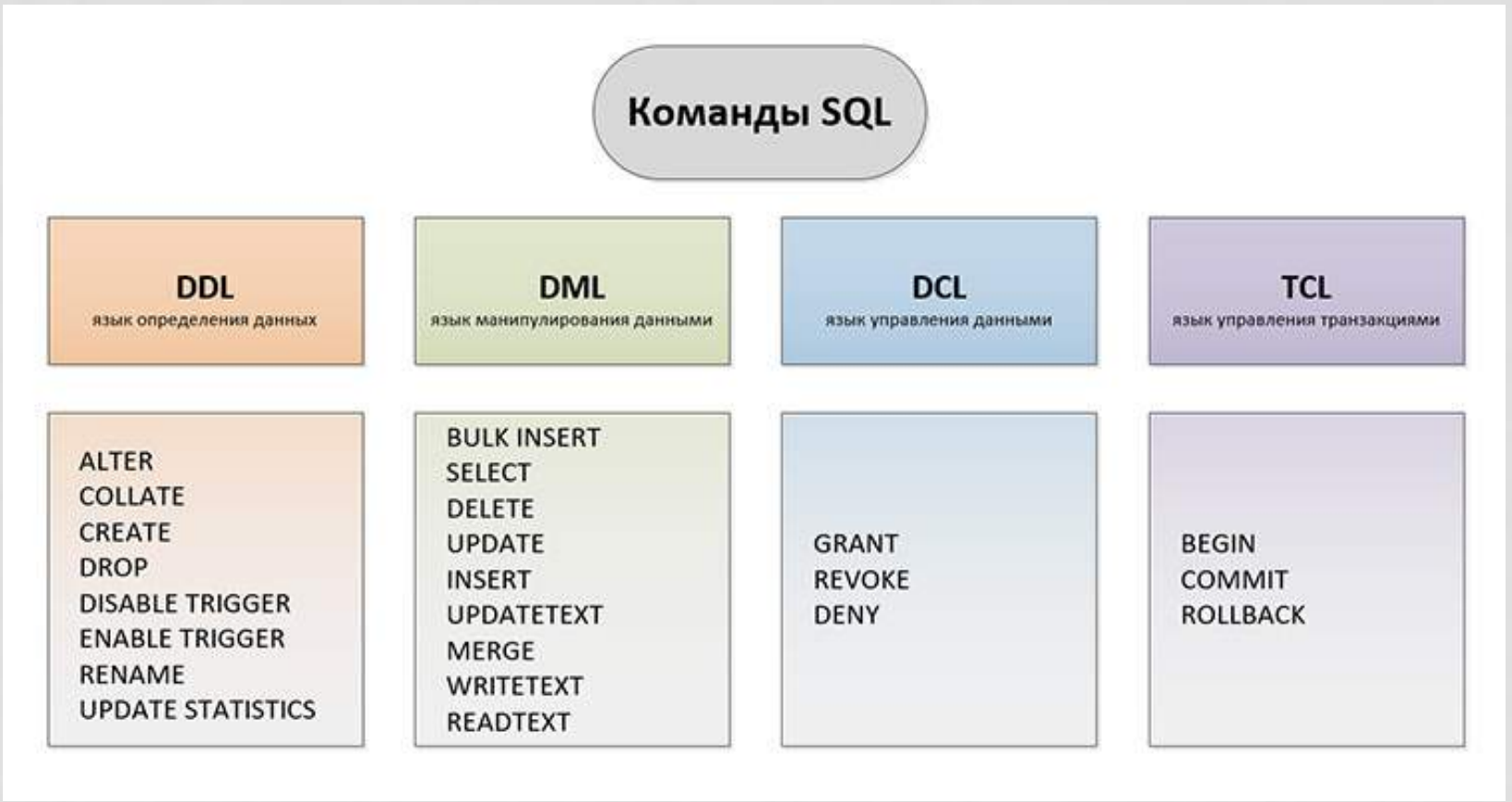
Язык SQL (Structured Query Language) в настоящее время стал фактически стандартным языком доступа к "реляционным" базам данных.

Многие нереляционные системы также имеют в настоящее время средства доступа к реляционным данным.

Целью стандартизации является переносимость приложений между различными СУБД.

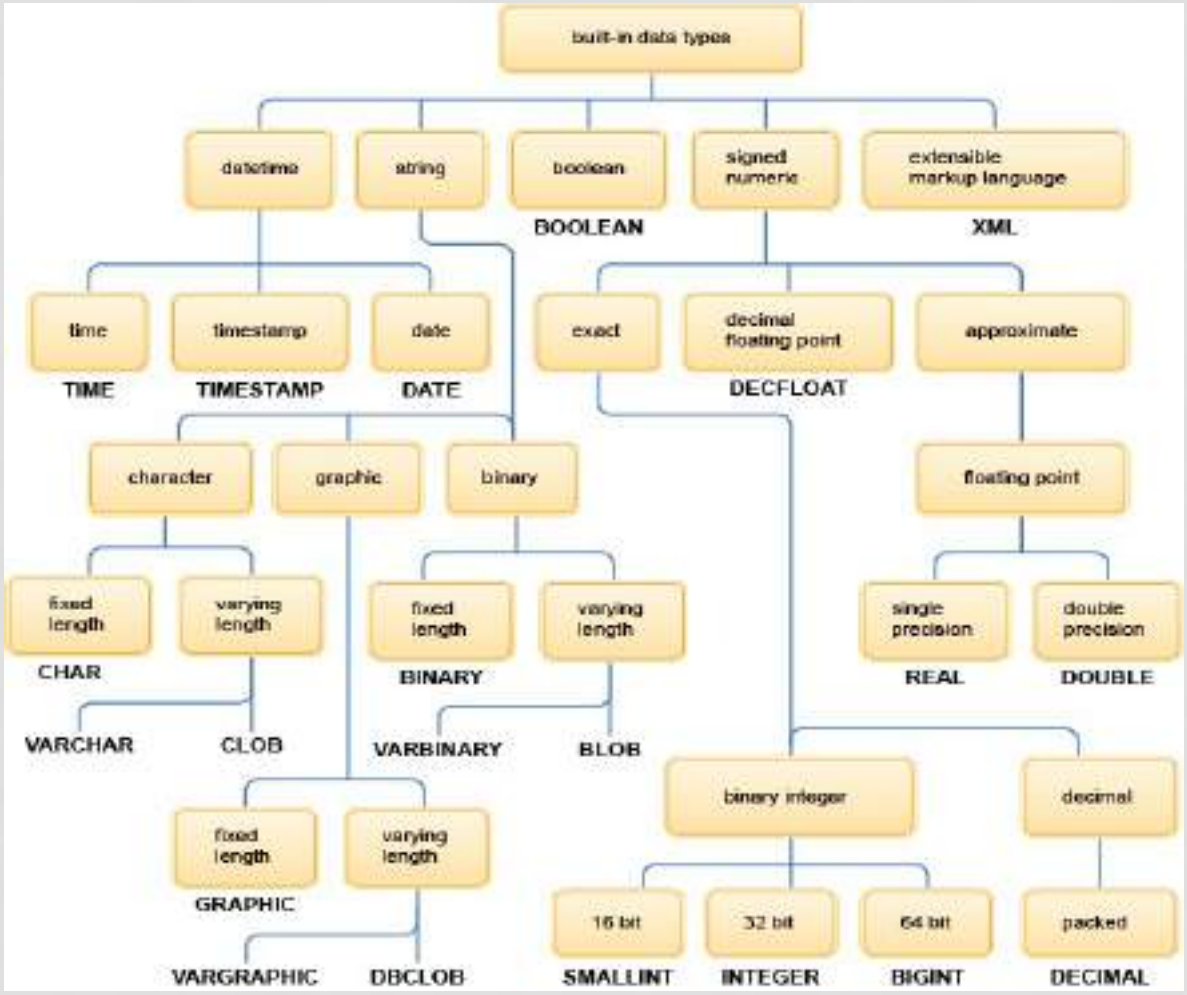
ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ (SQL)

ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ (SQL)



https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcQIT9pzl1i6Na-ffKTLlwaQh8e87LHXAlmccrE1c844R82Z5o3y

ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ (SQL)



https://andreyex.ru/wp-content/uploads/2017/04/SQL-Tipy-dannyh-330x250.gif

ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ (SQL)

В синтаксических конструкциях предложений SQL могут использоваться следующие обозначения:

- звездочка (*);
- квадратные скобки ([]);
- фигурные скобки ({});
- многоточие (...);
- прямая черта (|);
- запятая (,);
- пробелы ( );
- прописные латинские буквы;
- строчные буквы.

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

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

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]

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

```
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)
```

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

Базовые таблицы создаются в SQL с помощью предложения *CREATE TABLE*, синтаксис которого имеет небольшие различия в различных СУБД. Однако все они поддерживают следующую минимальную форму:

CREATE TABLE *базовая_таблица_имя* (столбец тип_данных [,столбец тип_данных] ...);

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

```
CREATE TABLE Сотрудники
                (Код            SMALLINT,
                 ФИО            CHAR (70),
                 Адрес                  CHAR  (15),
                 Телефон        CHAR (10),
                 Дата рожд.     DATE,
                 Город          CHAR (10),
                 ИИН            CHAR  (12));
```

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

Существующую базовую таблицу можно в любой момент уничтожить с помощью предложения *DROP TABLE* (уничтожить таблицу):

DROP TABLE базовая_таблица;

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

Для построения индекса в SQL существует предложение *CREATE INDEX* (создать индекс).

Индекс - это системная таблица, построенная по значениям заданного столбца заданной таблицы.

Уничтожить индексы, построенные по столбцам таблицы можно с помощью предложения DROP INDEX (уничтожить индекс), имеющего следующий формат:

DROP INDEX имя_индекса;

СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ИНДЕКСОВ

Синтаксис предложения CREATE INDEX (создать индекс):

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*]

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

Ограничения - это часть определений таблицы, которое ограничивает значения, которые можно вводить в столбцы. Когда вы создаете таблицу (или, когда вы ее изменяете), вы можете помещать ограничения на значения, которые могут быть введены в поля.

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

https://i.pinimg.com/736x/fe/73/1a/fe731ac001f42b09cf294efd12778f95.jpg

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Сотрудники
        (Код            SMALLINT NOT NULL,
        ФИО             CHAR (70) NOT NULL,
        Адрес           CHAR  (15) NOT NULL,
        Телефон         CHAR (10),
        Город           CHAR (10),
        Дата рождения   DATE NOT NULL,
        ИИН             CHAR  (12) NOT NULL);
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Сотрудники
        (Код            SMALLINT NOT NULL,
        ФИО             CHAR (70) NOT NULL,
        Адрес           CHAR  (15) NOT NULL,
        Телефон         CHAR (10),
        Город           CHAR (10) DEFAULT 'Караганда',
        Дата рождения   DATE NOT NULL,
        ИИН             CHAR  (12) NOT NULL);
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Заказчики (
    Код_заказчика integer CHECK
    (Код_заказчика >=0),
    Наименование_заказчика char (10) NOT
    NULL,
    Город char (10),
    Код_продавца integer CHECK
    (Код_продавца >=0));
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Сотрудники
        (Код            SMALLINT NOT NULL,
        ФИО             CHAR (70) NOT NULL,
        Адрес           CHAR  (15) NOT NULL,
        Телефон         CHAR (10),
        Город           CHAR (10) DEFAULT 'Караганда',
        Дата рождения   DATE NOT NULL,
        ИИН             CHAR  (12) NOT NULL UNIQUE);
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Заказчики          (
        Код_заказчика      integer NOT NULL,
        Наименование_заказчика                  char (10) NOT
        NULL,
        Город                       char (10),
        Код_продавца                        integer NOT NULL,
        UNIQUE  (Код_заказчика, Код_продавца));
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Сотрудники
        (Код            SMALLINT NOT NULL PRIMARY KEY,
         ФИО            CHAR (70) NOT NULL,
        Адрес           CHAR  (15) NOT NULL,
        Телефон         CHAR (10),
        Город           CHAR (10) DEFAULT 'Караганда',
        Дата рождения   DATE NOT NULL,
        ИИН             CHAR  (12) NOT NULL UNIQUE);
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

```
CREATE TABLE Телефонный справочник
    (Номер_телефона     integer NOT NULL
    PRIMARY KEY
    ФИО                 char(20),
    Код_улицы           integer,
    FOREIGN KEY (Код_улицы) REFERENCES
    Справочник_улиц (Код));
```

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

CREATE TABLE Телефонный справочник
(Номер_телефона integer NOT NULL
PRIMARY KEY
ФИО char(20),
Код_улицы integer **REFERENCES**
Справочник_улиц (Код));

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

Внешний ключ может содержать только те значения которые фактически представлены в родительском ключе или пустые (NULL).

Попытка ввести другие значения в этот ключ при выполнении операций вставки (INSERT) или модификации (UPDATE) будет отклонена.

Можно удалять (DELETE) любые строки с внешними ключами не используя родительские ключи вообще.

Вы можете объявить внешний ключ как NOT NULL, но это необязательно, и в большинстве случаев, нежелательно.

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

Каскадная ссылочная целостность:

- ограничить, или запретить, изменение (удаление), обозначив, что изменения в родительском ключе – ***ограниченные изменения*** (NO ACTION);

- сделать изменение (удаление) в родительском ключе и тем самым сделать изменения во внешнем ключе автоматическим, что называется - ***каскадным изменением*** (CASCADE);

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

Каскадная ссылочная целостность:

- сделать изменение (удаление) в родительском ключе, и установить внешний ключ в NULL, автоматически (полагая, что NULL разрешен во внешнем ключе), что называется - ***пустым изменением внешнего ключа*** (SET NULL);

- сделать изменение (удаление) в родительском ключе, и установить внешний ключ ***в значение по умолчанию***, автоматически, полагая, что значение по умолчанию установлено во внешнем ключе (SET DEFAULT).

ОПРЕДЕЛЕНИЕ ОГРАНИЧЕНИЙ ДЛЯ ТАБЛИЦ БД

CREATE TABLE Телефонный_справочник
(Номер_телефона integer NOT NULL PRIMARY KEY,
ФИО char(20),
Код_улицы integer REFERENCES Справочник_улиц,
ON UPDATE CASCADE,
ON DELETE NO ACTION);

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

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

DELETE
FROM    Поставщики
WHERE   ПС = 7;

DELETE
FROM Поставки;
```

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

```
DROP TABLE Поставки

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

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

INSERT
INTO {базовая таблица | представление} [(столбец [,столбец] ...)]
VALUES ({константа | переменная} [,{константа | переменная}] ...);

Или

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

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

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

INSERT
INTO   Блюда (Основа, В, Блюдо, БЛ, Выход)
VALUES       ('Мясо', 'Г', 'Шашлык', 34, 150);

INSERT
INTO   Блюда
VALUES       (34, 'Шашлык', 'Г', 'Мясо', 150, 5);
```

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

```
INSERT INTO    К_меню
SELECT         Вид, Блюдо,
          INT(SUM(((Белки+Углев)*4.1+Жиры*9.3) * Вес/1000)),
          (SUM(Стоимость/К_во*Вес/1000) + MIN(Труд/100))
FROM   Блюда, Вид_блюд, Состав, Продукты, Наличие
WHEREБлюда.БЛ          = Состав.БЛ
AND    Состав.ПР       = Продукты.ПР
AND    Состав.ПР       = Наличие.ПР
AND    Блюда.В         = Вид_блюд.В
AND    БЛ NOT IN
          (         SELECT         БЛ
                    FROM   Состав
                    WHEREПР IN
                              (         SELECT         ПР
                                        FROM   Наличие
                                        WHEREК_во = 0))
GROUP BY Вид, Блюдо
ORDER BY Вид, 3;
```

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

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

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

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

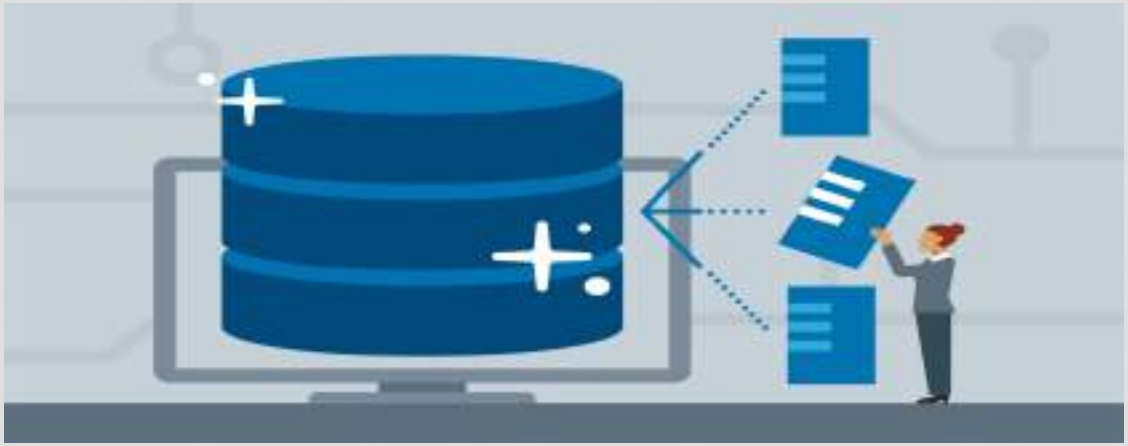
```
UPDATE Блюда
SET    Блюдо = 'Форшмак', Выход = (Выход+30), Труд
= NULL
WHERE  БЛ = 5;
```

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

freepik.com

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

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



freepik.com

ПРЕДЛОЖЕНИЯ МОДИФИКАЦИИ ДАННЫХ

```
UPDATE      Продукты
SET     ПР = 20
WHERE       ПР = 13;
UPDATE      Состав
SET     ПР = 20
WHERE       ПР = 13;
UPDATE      Поставки
SET     ПР = 20
WHERE       ПР = 13;
UPDATE      Наличие
SET     ПР = 20
WHERE       ПР = 13;
```

freepik.com

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Кроме традиционных 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).

freepik.com

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Критерий отбора строк формируется из одного или нескольких условий, соединенных логическими операторами:

- AND - когда должны удовлетворяться оба разделяемых с помощью AND условия;
- OR - когда должно удовлетворяться одно из разделяемых с помощью OR 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, причем существует приоритет AND над OR (сначала выполняются все операции AND и только после этого операции OR).

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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	Каспи
3	Поставщик_3	3333	Народный
4	Поставщик_4	4444	Каспи
5	Поставщик_5	5555	Народный

Движение товара

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
01.01.20	1	1	2	100	10
10.01.20	2	5	3	50	20
20.01.20	3	1	2	100	10
30.01.20	4	2	1	50	15
01.03.20	5	3	4	100	25
10.03.20	6	4	3	200	20
20.03.20	7	1	2	10	12
30.03.20	8	2	1	50	15
01.10.20	9	3	4	10	25
10.10.20	10	5	3	20	20
20.10.20	11	1	2	100	12
30.10.20	12	2	1	5	15

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Запрос: выдать название и категории товаров.
SELECT Наименование, Категория_товара
FROM Товары;

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

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

SELECT *
FROM Поставщики;
```

freepik.com

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Выдать список банков, в которых обслуживаются поставщики товаров:

```
SELECT  Банк
FROM  Поставщики;
```

Банк
Народный
Каспи
Народный
Каспи
Народный

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

Банк
Каспи
Народный

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Запрос: получить стоимость поставленных товаров по накладной.

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

№ 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

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

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

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Запрос: получить перечень накладных, выписанных организацией с кодом 1.

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

№ накладной	Дата
1	01.01.20
3	20.01.20
7	20.03.20
11	20.10.20

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

№ накладной	Дата	Количество	Цена
1	01.01.20	100	10
3	20.01.20	100	10
4	30.01.20	50	15
8	30.03.20	50	15
11	20.10.20	100	12

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

№ накладной	Дата	Количество	Цена
1	01.01.20	100	10
3	20.01.20	100	10
4	30.01.20	50	15
7	20.03.20	10	12
8	30.03.20	50	15
11	20.10.20	100	12
12	30.10.20	5	15

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

(Код товара=1 OR Код товара=2;)

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
01.01.20	1	1	2	100	10
20.01.20	3	1	2	100	10
30.01.20	4	2	1	50	15
20.03.20	7	1	2	10	12
30.03.20	8	2	1	50	15
20.10.20	11	1	2	100	12
30.10.20	12	2	1	5	15

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

Имеются два типа групповых символов используемых с LIKE:

- символ _ (подчеркивание) – заменяет любой одиночный символ;

- символ % (процент) – заменяет любую последовательность из N символов (где N может быть нулем).

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

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

Наименование
Поставщик_1
Поставщик_2
Поставщик_3
Поставщик_4
Поставщик_5

ПРЕДЛОЖЕНИЯ ВЫБОРКИ ДАННЫХ

```
SELECT    № накладной
FROM      Движение товара
WHERE     Цена IS NULL;
```

```
SELECT    № накладной
FROM      Движение товара
WHERE     Цена IS NOT NULL;
```

freepik.com

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

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

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
10.03.20	6	4	3	200	20
01.01.20	1	1	2	100	10
20.01.20	3	1	2	100	10
01.03.20	5	3	4	100	25
20.10.20	11	1	2	100	12
10.01.20	2	5	3	50	20
30.01.20	4	2	1	50	15
30.03.20	8	2	1	50	15
10.10.20	10	5	3	20	20
20.03.20	7	1	2	10	12
01.10.20	9	3	4	10	25
30.10.20	12	2	1	5	15

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

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

Дата	№ накладной	Код поставщика	Код товара	Количество	Цена
10.03.20	6	4	3	200	20
01.03.20	5	3	4	100	25
20.10.20	11	1	2	100	12
01.01.20	1	1	2	100	10
20.01.20	3	1	2	100	10
10.01.20	2	5	3	50	20
30.01.20	4	2	1	50	15
30.03.20	8	2	1	50	15
10.10.20	10	5	3	20	20
01.10.20	9	3	4	10	25
20.03.20	7	1	2	10	12
30.10.20	12	2	1	5	15

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

```
SELECT  № накладной, (Количество*Цена)
FROM    Движение товара;
ORDER   BY 2;
```

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

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

В SQL существует ряд специальных стандартных функций (SQL-функций). Кроме специального случая COUNT(*) каждая из этих функций оперирует совокупностью значений столбца некоторой таблицы и создает единственное значение, определяемое так:

- COUNT - число 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.

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

```
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

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

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

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

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

```
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

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

```
SELECT   Количество, Цена, COUNT(№ накладной)
FROM     Движение товара
GROUP    BY Количество, цена;
```

Количество	Цена	COUNT(№ накладной)
200	20	1
100	10	2
100	25	1
100	12	1
50	20	1
50	15	2
10	12	1
10	25	1
20	20	1
5	15	

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

```
SELECT   Код товара, SUM(Количество)
FROM     Движение товара
WHERE    Код поставщика <> 1
GROUP    BY Код товара;
```

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

ВЫБОРКА С УПОРЯДОЧЕНИЕМ И АГРЕГИРОВАНИЕМ ДАННЫХ

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

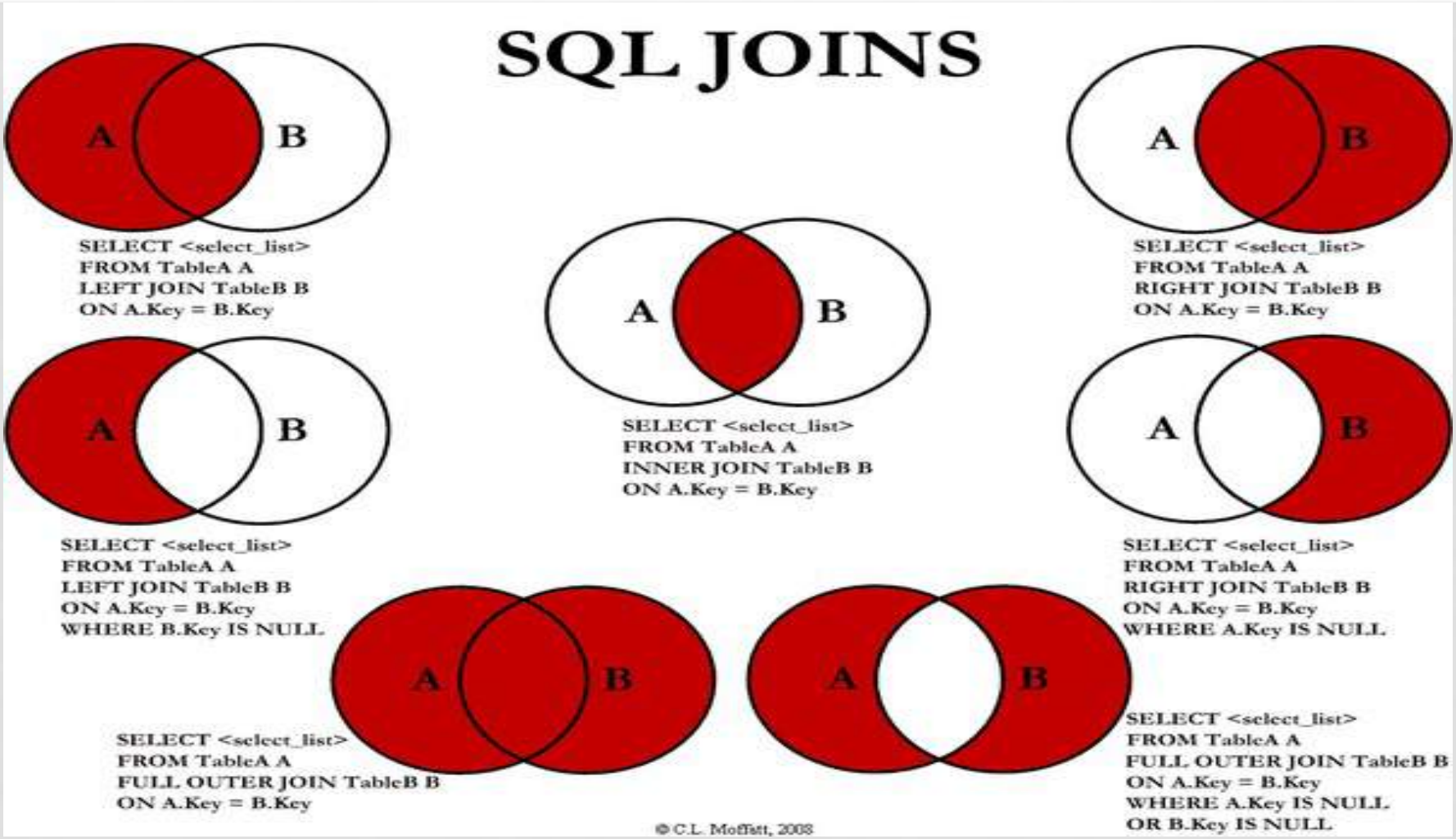
Код поставщика	COUNT(№ накладной)
1	4
2	3

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

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

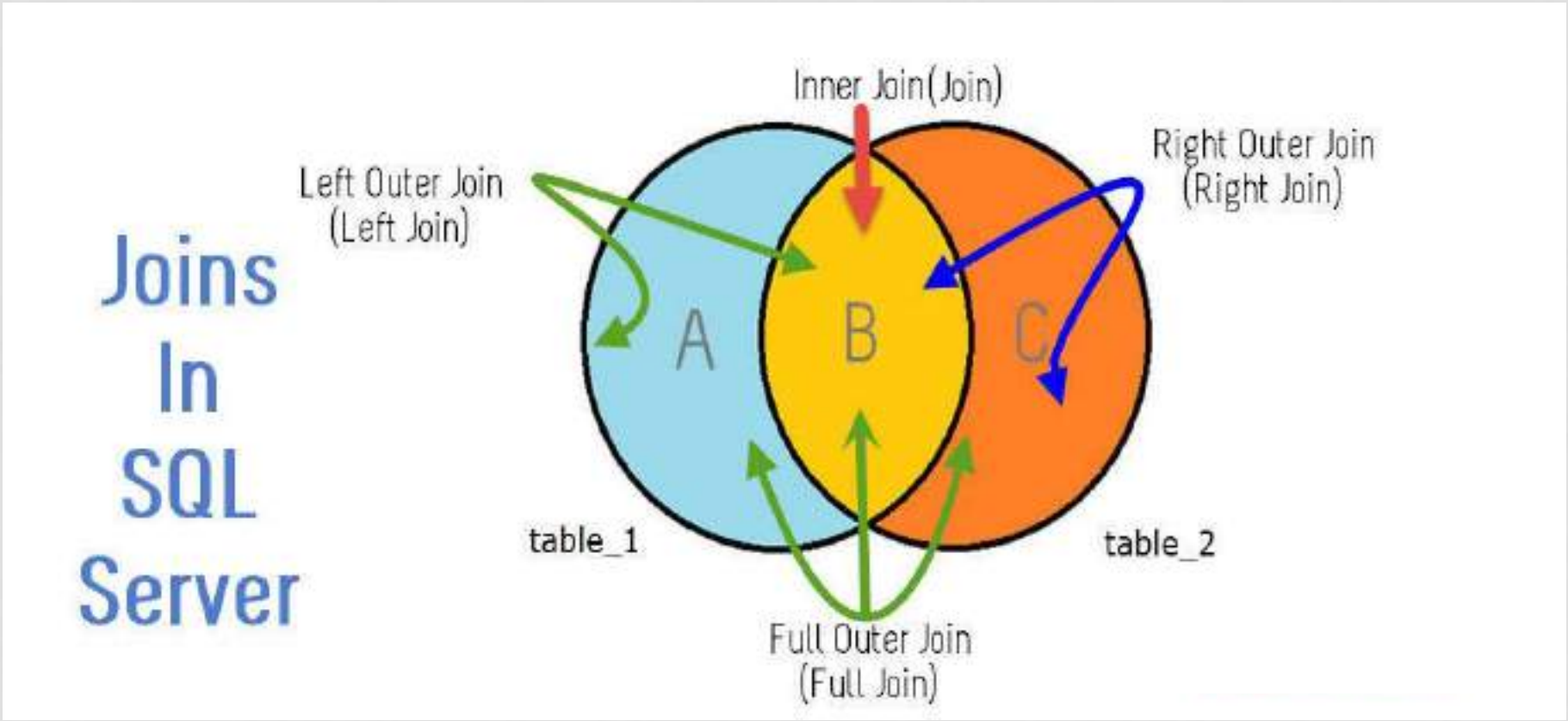
Наименование	Категория товара	Количество	Цена
Товар_2	мебель	100	10
Товар_2	мебель	100	10
Товар_2	мебель	10	12
Товар_2	мебель	100	12

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ



https://external-preview.redd.it/M5QHWsp2vgZ-3QDZ4m-qS58lsOUgDNHau8trSFzS8H0.jpg?auto=webp&s=cae9cdc438b71c9025d40dad4650801fdcae1ef8

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ



https://i.sstatic.net/S3WMq.jpg

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

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 (= | <> | < | <= | > | >=).

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

Простые вложенные подзапросы обрабатываются системой "снизу вверх".

Запросы с коррелированными вложенными подзапросами обрабатываются системой в обратном порядке.

freepik.com

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

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

Наименование	Банк
Поставщик_5	Народный
Поставщик_4	Каспи
Поставщик_5	Народный

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

```
SELECT    DISTINCT Код поставщика
FROM              Движение товара
WHERE     Цена  IN
                  (SELECT Цена
                   FROM     Движение товара
                   WHERE    Код поставщика = 20 );
```

Код поставщика
5
4

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

```
SELECT    Наименование, Банк
FROM          Поставщики
WHERE     Банк =
     (SELECT Банк
      FROM Поставщики
     WHERE   Код поставщика = 5 );
```

Код поставщика
1
3

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

```
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  );
```

ЗАПРОСЫ С ИСПОЛЬЗОВАНИЕМ НЕСКОЛЬКИХ ТАБЛИЦ

Квантор EXISTS (существует) - понятие, заимствованное из формальной логики. В языке SQL предикат с квантором существования представляется выражением EXISTS (SELECT * FROM ...).

```
SELECT   Название
FROM     Поставщики
WHERE    EXISTS
   (     SELECT *
         FROM   Поставки
         WHERE  ПС = Поставщики.ПС
         AND    ПР = 11 );
```

Результат:

Название
СЫТНЫЙ
УРОЖАЙ
КОРЮШКА
ЛЕТО

```
SELECT   Название, Статус
FROM     Поставщики
WHERE    NOT EXISTS
   (     SELECT *
         FROM   Поставки
         WHERE  ПС = Поставщики.ПС
         AND    ПР = 11 );
```

Название	Статус
ПОРТОС	кооператив
ШУШАРЫ	совхоз
ТУЛЬСКИЙ	универсам
ОГУРЕЧИК	ферма

ЗАДАНИЯ ДЛЯ СРС

1. Проведите тематическое исследование: в чем разница между T-SQL и PL/SQL.

2. Ответьте на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение языка SQL.
2. Типы предложений языка SQL.
3. Какие группы операторов языка SQL вы знаете?
4. К операторам определения объектов базы данных можно отнести следующие операторы…
5. К операторам манипулирования данными можно отнести следующие операторы…
6. К операторам защиты и управления данными можно отнести следующие операторы…
7. Какие типы данных поддерживает язык SQL?
8. Какие обозначения могут использоваться в конструкциях языка SQL и что они обозначают?

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Базы данных: Проектирование, реализация и сопровождение. Теория и практика [Текст] : учебное пособие / Т. Конноли, К. Бегг. - 3-е изд. - М. ; СПб. ; Киев : Вильямс, 2018. - 1440 с.

2. К.Дж. Дейт. Введение в системы баз данных. М., 2018. - 1328 с.

3. Уидом Д., Гарсиа-Молина Г. Системы баз данных. Полный курс. - М.: Вильяме, 2018. - 1088 с.

freepik.com

ЛЕКЦИЯ ОКОНЧЕНА.
БЛАГОДАРЮ ЗА ВНИМАНИЕ!