Лекція 11-2020_2021

- На попередній лекції
- Розрахунок віку людини
- Корегування структури таблиці
- Оператор UPDATE
- Оператори DELETE, DROP
- Створення таблиць

Что было в предыдущей лекции

FOREIGN KEY – внешний ключ, **TEMPORARY –** создание временной таблицы, **LIKE –** создание одной таблицы из другой.

Создание индекса: **CREATE INDEX** или **CREATE TABLE** .

ALTER TABLE - изменение определения существующей таблицы.
RENAME TABLE – переименование, **DESCRIBE –** информация о структуре таблицы, **USE –** установка текущей БД.

UPDATE – обновление таблицы, **DELETE –** удаление строк из таблицы. **DROP –** удаление таблицы. **UNION –** объединение в один результирующий набор результатов нескольких операторов **SELECT**. UNION будет автоматически исключать дубликаты строк из вывода.

Что было в предыдущей лекции

```
CREATE DATABASE [IF NOT EXISTS] db_name
[ [ DEFAULT ] CHARACTER SET character_ name ]
[ [ DEFAULT ] COLLATE collation_ name ]
```

создает базу данных с указанным именем.

CHARACTER SET – кодировка, **COLLATE –** сравнение.

ALTER DATABASE db_name – изменение структуры базы данных

CREATE TABLE – создание таблицы данных, **PRIMARY KEY** определяет поле первичного ключа, ограничение **CHECK** позволяет установить условие, которому должно удовлетворять значение, вводимое в таблицу, **DEFAULT –** значение по умолчанию, **AUTO_INCREMENT PRIMARY KEY –** автоинкрементное значение первичного ключа,

Что было в предыдущей лекции

SELECT ... UNION [ALL] SELECT …

UNION используется, чтобы объединить в один простой результирующий набор результаты нескольких операторов **SELECT**. Объединяемые запросы должны иметь одинаковые заголовки. **UNION** удаляет из объединения дублирующие записи, **UNION ALL** не удаляет.

Расчет полного возраста человека

YYYY-MM-DD MySQL
DD.MM.YYYY Access

Вывести фамилию, дату рождения и ***полный возраст человека***

Пример 1. **MySQL**

```
Select FIO, BIRTH, Current_Date,
(YEAR(Current_Date)- YEAR(BIRTH))-
  (RIGHT(Current_Date,5)< RIGHT(BIRTH,5))) as
  age From Table;
```

От разности лет нужно либо вычесть 1, либо не вычитать. Результат сравнения либо true (1), либо false (0).

Расчет полного возраста человека

Access – функция **Iif** (SQL).

Iif(условие**(true/false),** оператор**1(true),** оператор**2(false))**

Пример 4.

```
SELECT Name, Birth,
Date() AS Текущая_Дата,
Year(Now()) -Year(Birth) -
(Iif(
(DatePart("m",Now())
   < DatePart("m",Birth))
Or ( (DatePart("m",Now())
   = DatePart("m",Birth))
And ( DatePart("d",Birth)
      > DatePart("d",Now())
    ) ) , 1, 0) ) as Age
From Person;
```

→ Результат правильный с точностью до дня.

Расчет полного возраста человека

Access

Пример 2.

```
SELECT Name, Birth,
Int((Date()- Birth)/365) AS Age FROM Doctor;
```

Не верные результаты за счет наличия високосных лет.

Access - функция **DateDiff**

Пример 3.

```
SELECT Name, Birth,
DateDiff("yyyy", Birth, Date()) AS Age
FROM Doctor;
```

Не всегда возвращает правильное количество лет между указанными датами.

Расчет полного возраста человека

Access

Пример 5.

```
SELECT Name, Birth,
Int((Date()- Birth)/365.25) AS Age
FROM Doctor;
```

Область ограничения – должно быть > 4 лет.

Пример 6.

Значения True, False.

Mysql. True =1, False=0.

```
select  if(TRUE > FALSE, 1, 0); Результат  1
```

Access, True = - 1, False=0.

```
select iif(TRUE > FALSE, 1, 0); Результат 0
```

Изменение структуры таблицы

Команда **ALTER TABLE** не часть стандарта ANSI. Используется для изменения определения существующей таблицы.

Возможности **ALTER TABLE** :

- добавить столбцы к таблице,
- удалить столбцы из таблицы,
- изменить размеры столбцов,
- в некоторых СУБД добавлять или удалять ограничения.

Типичный синтаксис команды :

```
ALTER TABLE <table name>
   {ADD|DROP|ALTER} <column name>
                    <data type> <size>;
```


Изменение структуры таблицы

1. Пример 7. К таблице **Stuff** добавить столбец **Comments** для примечаний.

```
ALTER TABLE Stuff ADD COLUMN Comments TEXT(25);
```

Столбец будет добавлен со значением NULL для всех строк таблицы. Новый столбец станет последним по порядку столбцом таблицы.

2. Пример 14 . Изменить тип данных «Целое» столбца **Ind** на тип данных «Текстовый(10)»

```
ALTER TABLE Stuff ALTER COLUMN Ind TEXT(10);
```

3. Пример 15. Удалить столбец с примечаниями

```
ALTER TABLE Stuff DROP COLUMN Comments;
```


Добавление индекса к таблице

```
 CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX
index_name [index_type]
     ON tbl_name (index_col_name,...)
```

Обычно все индексы создают в момент создания таблицы оператором **CREATE TABLE**.

Оператор **CREATE INDEX** добавляет индексы к существующей таблице. Работает аналогично **ALTER TABLE.**

Переименование, описание, использование

MySQL. **RENAME, DESCRIBE, USE**

RENAME TABLE tbl_name TO new_tbl_name

DESCRIBE tbl_name Информация о структуре таблицы

USE db_name

USE – установка текущей БД для последующих операторов. Указанная в **USE** БД – текущая до конца сессии или до следующего **USE**

Пример 8.

```
USE db1;
SELECT COUNT(*) FROM mytable; # из db1.mytable

USE db2;
SELECT COUNT(*) FROM mytable; # из db2.mytable
```

Обновление таблицы

Синтаксис команды **UPDATE**:

```
UPDATE <табл>  SET col1= expr [,… col n= expr]
[where условие]
```

UPDATE обновляет столбцы в соответствии с их новыми значениями в строках существующей таблицы.

В выражении **SET** указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены.

В выражении **WHERE**, если оно присутствует, задается, какие строки подлежат обновлению.

Если нет условия – обновляются все строки.

Если задано **ORDER BY** – обновление в указанном порядке.

Обновление таблицы

Пример 9. **UPDATE person SET age=age+1;**

Значения команда **UPDATE** присваивает **слева направо**.

Пример 10. Удвоить значение столбца **age**, затем инкрементировать:

```
UPDATE person SET age=age*2, age=age+1;
```

Команда **UPDATE** возвращает количество измененных строк.

В **SET** можно вводить пустые NULL значения так же, как вводятся другие значения.

Пример 11. Установить рейтинги заказчиков в Харькове в NULL.

```
UPDATE customers SET rating = NULL
      WHERE city = 'Харьков';
```

Запрос обнулит все рейтинги заказчиков в Харькове.

Обновление таблицы

Пример 12. Таблицы: список должностей **Dolg {CodeD, NameD}** и **л**ичные карточки сотрудников

```
Card {CodeC, Inn, Name, Salary, CodeD}
```

1.Изменить всем должность “инженер” на “программист”.

```
UPDATE Dolg SET NameD="программист"
        Where NameD="инженер";
```

2. Всем программистам увеличить оклад на 50% .

```
UPDATE Card INNER JOIN Dolg
         On Card.CodeD=Dolg.CodeD
SET Card.Salary= Card.Salary*1.5
 WHERE Dolg.NameD= "программист";
```


Команда DELETE

Синтаксис: **DELETE [LOW_PRIORITY] таблица [.*]**

FROM таблица [WHERE условие] – удалить одну или несколько строк из таблицы или представления (запроса).

MySQL: [**LOW_PRIORITY**] – низкий приоритет выполнения (ожидание до тех пор, пока другие клиенты не завершат чтение этой таблицы). [**DELAYED**] – задержанное выполнение (подтверждение сервера сразу же, операции с таблицей блоком, когда эту таблицу перестанет использовать другой поток).

При использовании инструкции DELETE, **удаляются только данные,** структура таблицы и ее свойства не меняются.

Если нет WHERE – удаляются все строки .

Если в FROM только одна таблица – список выбора не нужен.

Команда DELETE

Пример 13. Удалить все строки из таблицы **Tab**

```
Delete Tab.*  From Tab;
```
Access

```
Delete   From Tab;
```
MySQL

Пример 14. Удалить все записи из таблицы **Врачи**, для тех врачей, у которых стаж > 30 лет

```
Delete Врачи.* From Врачи Where Стаж>30;
```

Команда DELETE – возвращает количество удаленных записей.

Команда DROP

Удаляет таблицу, процедуру или представление из базы данных либо индекс из таблицы.

```
Drop Table;
```

Удаление таблицы приводит к потере ее структуры.

MySQL. Удаление базы данных

```
DROP DATABASE [IF EXISTS] db_name
```

Оператор **DROP DATABASE** удаляет все таблицы в указанной базе данных и саму базу.

Команда вставки INSERT

INSERT применяется для добавления записей в таблицу.

Формат оператора:

```
INSERT INTO <имя_таблицы>
 [(  имя_столбца [,...n]]
  { VALUES (значение[,...n]
  ) | <SELECT_оператор>}
```

Команда **INSERT** во всех СУБД возвращает количество вставленных строк.

INSERT …VALUES

INSERT с параметром **VALUES** - вставка единственной строки в таблицу. Список столбцов указывает столбцы, которым будут присвоены значения в добавляемых записях. Список может быть опущен, тогда подразумеваются все столбцы таблицы.

Пример 15. Таблица Department {Otd, Name, Shef, Secretar}. Ввести новый отдел.

```
INSERT INTO Department VALUES
(24, 'Отдел радиолокации', 'Иванов И.И.',
'Петров П.П.');
```

INSERT …VALUES

1. Если указан конкретный список имен полей , любые пропущенные в нем столбцы должны быть объявлены при создании таблицы как допускающие значение NULL.
2. Список значений должен соответствовать списку столбцов:

- количество элементов в обоих списках одинаково;
- позиционирование одинаково;
- типы данных элементов в списке совместимы с типами данных соответствующих столбцов таблицы.

Access: INSERT …VALUES

Если нужно, чтобы при добавлении записей в таблицу СУБД автоматически изменяла значение счетчика - не включать поле счетчика в запрос.

Пример 16. Таблица **Tabl** (**id** – счетчик, **text** – текст)
Вставить новую запись в таблицу, счетчик отрабатывает автоматически

```
INSERT INTO Tabl (text) VALUES ("Петя");
```

Если до вставки значение счетчика =5,
после вставки будет = 6.

Access: INSERT …VALUES

Поле счетчика необходимо включить в запрос, чтобы не допустить автоматическое изменение счетчика.

Пример 17. В таблицу **Tabl** после вставки Пети вставляем запись как

```
INSERT INTO Tabl  VALUES (10, "Вася");
```

Добавится запись с номером счетчика = 10, значения 7,8,9 – не будут использоваться, если в дальнейшем вставка будет осуществляться с автоматическим формированием значения счетчика.

Access: INSERT …VALUES

Но!!

Пример 18.

```
INSERT INTO Tabl  VALUES (7, "Ваня");
```

Добавляем вручную запись со значением счетчика = 7.
Если значение 7 уже есть – инициируется ошибка.

Пример 19. Удалить ошибочно размещенного Ваню с номером = 7 и под этим номером разместить Вову.

```
Delete * From  Tabl where id =7;
INSERT INTO Tabl  VALUES (7, "Вова");
```

удаляет запись со значением id=7,
на ее место вставляется другая запись

MySQL: INSERT …VALUES

Правило

Если в столбец, объявленный как **Not Null,** вводится значение **Null** , в этом столбце устанавливается значение, принятое по умолчанию.

Для первого значения автоинкрементного столбика – это 0, для последующих это MAX+1.

Пример 20.

```
INSERT INTO Tabl  VALUES (Null, "Коля");
```

– вставка записи с автоматическим наращиванием автоинкрементного ключа. Первичный ключ объявлен с атрибутом **Not Null**. Можно писать 0, но рекомендуется **Null**.

MySQL: множественная вставка

MySQL допускает использование **INSERT** с параметром **VALUES** для вставки нескольких строк, при этом каждый набор значений отделяется запятой.

Пример 21. Правильная запись

```
INSERT INTO Tabl
VALUES (Null, 'Таня'),(Null, 'Маша');
```

Не правильная запись

```
INSERT INTO Tabl
VALUES (Null, 'Таня', Null, 'Маша');
```


Особенности INSERT …VALUES

В варианте вставки **VALUES** – можно писать выражения. Выражение может относиться к любому столбцу, который был ранее внесен в список значений.

Пример 22. Правильная запись

```
Insert Into Tab ( col1, col2)
     Values (15, col1*10);
```

Не правильная запись

```
Insert Into Tab ( col1, col2)
    Values (col2*10, 15);
```


INSERT ... VALUES

Пример 23. Если необходимо вставить поле даты в MySQL, это необходимо делать в формате (YYYY-MM-DD). Если данные берутся из приложения, в котором формат даты (DD-MM-YYYY), необходимо выделять подстроку в SQL операторе.

```
INSERT INTO Orders (idContact, Contractno,
                    Description, Startdate )
 VALUES ( '2', '2040906', '',
   CONCAT(SUBSTRING('07-09-2014',7,4),
          SUBSTRING('07-09-2014',4,2),
          SUBSTRING('07-09-2014',1,2) )
);
```

Сопутствующие задачи

Пример 24. Посчитать количество строк в таблице.

```
Select Count(*) From Tab;
```

Запрос работает везде.

MySQL: `SELECT ROW_COUNT();`

Пример 25. Определить последнее значение автоинкрементного ключа.

```
Select max(id) From Tab;
```

Запрос работает везде.

MySQL: функция `LAST_INSERT_ID()`.

Возвращает последнее вставленное значение автоинкрементного поля. При множественной вставке возвращает значение ключа, первое во вставляемой группе значений.

INSERT ... SELECT

INSERT ... SELECT – внесение большого количества строк в таблицу из одной или более таблиц.

Предложение SELECT задает поля для добавления в указанную конечную таблицу

Условия использования INSERT ... SELECT:

– целевая таблица команды INSERT не должна появляться в утверждении FROM части SELECT, т.к. запрещена выборка из той же таблицы, в которую производится вставка.

– столбцы AUTO_INCREMENT работают, как всегда.

– Предложение SELECT – любой допустимый оператор SELECT.

INSERT ... SELECT

Пример 26. Временная сводная таблица существует в течение месяца **Temp{Дата, Спец, Колич}.** На начало месяца отрабатывает триггер, который очищает содержимое таблицы. В течение месяца в эту таблицу поступают сведения о количестве пациентов, принятых каждым типом специалистов по дням. Используется для отчетности и в кадровой политике.

INSERT ... SELECT

```
INSERT INTO Temp (DateP, Spec, Number)
SELECT Priem.DateP, Spec.Spec,
                    Count(Priem.id_p)
From Spec  Inner Join
(Doctors Inner Join Priem
  On Doctors.id_d = Priem.id_d)
  On Spec.id_s = Doctors. id_s
Where ((Month (Priem.DateP)=
        Month (Now ())) and
       (Year (Priem.DateP)=
        Year (Now ())))
 Group by Priem.DateP, Spec.Spec;
```

INSERT ... SELECT

Если используется оператор INSERT ... VALUES с множественным списком или INSERT.. SELECT, оператор возвращает количество вставленных строк:

```
Records: 100 Duplicates: 0 Warnings: 0
```

MySQL. Порядок, в котором возвращает строки оператор SELECT без предиката ORDER BY не определен. Неоднократное выполнение такого INSERT ... SELECT может давать разный порядок вставки строк. Чтобы этого избежать необходимо использовать INSERT ... SELECT ... ORDER BY *column*..

INSERT ... SELECT

Пример 27. INSERT.. SELECT позволяет скомбинировать в одной таблице информацию, например, из двух таблиц.

```
CREATE TABLE tab1 (user VARCHAR(20) PRIMARY
KEY, age INT);
CREATE TABLE tab2 (user VARCHAR(20) PRIMARY
KEY, os VARCHAR(20));

INSERT INTO tab1 (user,age) VALUES
('fred',20); INSERT INTO tab1 (user,age)
VALUES ('mary',30);
INSERT INTO tab2 (user,os) VALUES
('fred','FreeBSD'); INSERT INTO tab2
(user,os) VALUES ('mary','Linux');
```


INSERT ... SELECT

```
CREATE TABLE newtable
(user VARCHAR(20) PRIMARY KEY, age INT,
      os VARCHAR(20));
INSERT newtable (user, age, os)
 SELECT tab1.user,tab1.age,tab2.os
 FROM tab1,tab2
 WHERE tab1.user = tab2.user;
```


INSERT ... SELECT

Пример 28.

Можно использовать INSERT INTO ... SELECT для копирования данных из одной БД в другую.

Пусть в качестве источника данных будет database1:

```
use database1;
```

Хотим копировать в database2:

```
INSERT INTO database2.table1
          (field1,field3,field9)
  SELECT table2.field3, table2.field1,
       table2.field4
  FROM table2;
```

Оператор Replace

Во всех СУБД при вставке строки со значением первичного ключа, уже имеющимся в таблице, возникает ошибка.

MySQL: оператор **REPLACE**. Действие подобно **INSERT,** в случае, когда значения первичного ключа в старой строке совпадают со значениями первичного ключа в новой строке, **REPLACE** их заменяет, выполняя последовательность из **DELETE** и **INSERT** операторов.

Access: надо явно выполнить удаление, а потом вставку.