**редактирование таблиц в БД**

**show tables;**  --- покозать существующие таблицы в бд

**DESC название\_таблицы**; ← показывает структуру таблицы т.е., название столбцов, тип используемых значений в каждом столбце, key, default, Extra

**DROP TABLE salespeople;** ← ydaliaet tablicy salespeople

**Создание таблицы (table) / удаление**

Для создания таблицы используется SQL-запрос. В нем указывается какая таблица создается, из каких атрибутов(полей) она состоит и какой тип данных имеет каждое поле

CREATE TABLE IF NOT EXISTS TABLE\_NAME (column\_name datatype, column\_name datatype);

CREATE TABLE nazvanie\_tablici (

nazvanie\_stolbca VARCHAR(20),

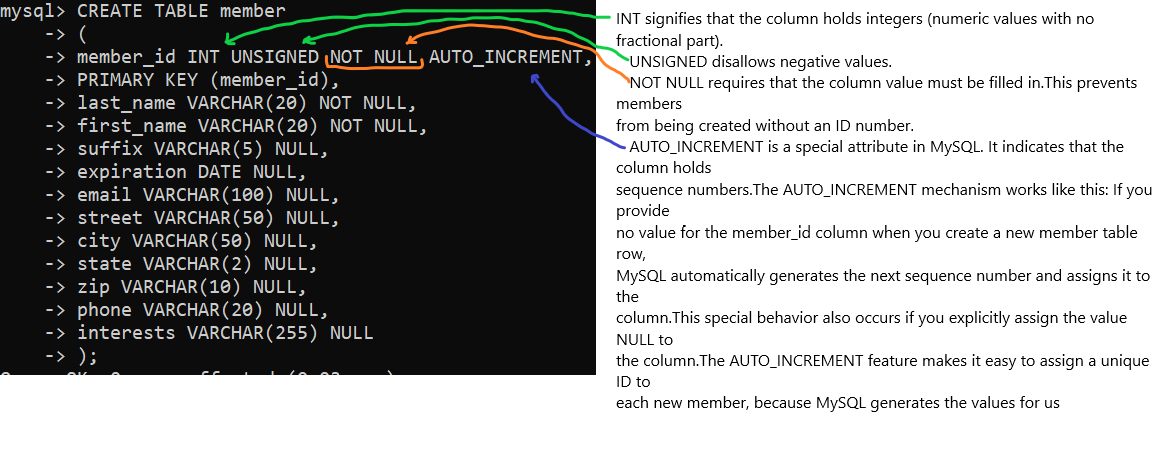
nazvanie\_stolbca VARCHAR(20),

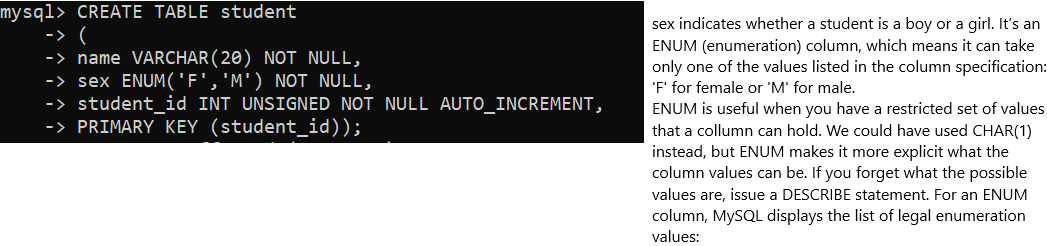
nazvanie\_stolbca VARCHAR(20),

nazvanie\_stolbca CHAR(1),

nazvanie\_stolbca DATE,

nazvanie\_stolbca DATE

)



Ещё пример создания таблицы:

**CREATE TABLE supply (**

**supply\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**title VARCHAR(50),**

**author VARCHAR(30),**

**price DECIMAL(8,2),**

**amount INT)**

The PRIMARY KEY clause indicates that the member\_id column is indexed to allow fast lookups.

**describe nazvanie\_tablici**; <--vivodit dannyy table.(shabky). Также с этой целью можно использовать:

DESCRIBE **nazvanie\_tablici**;

DESC **nazvanie\_tablici**;

EXPLAIN **nazvanie\_tablici**;

SHOW COLUMNS FROM **nazvanie\_tablici**;

SHOW FIELDS FROM **nazvanie\_tablici**;

**CREATE TABLE salespeople** (num int PRIMARY KEY,

name VARCHAR (20) NOT NULL DEFAULT 'sumText',

city VARCHAR (33) NOT NULL DEFAULT 'sumText',

comm INT NOT NULL DEFAULT '0'); ← создать таблицу под названием salespeople (название столбца и его тип значения)

**Новая таблица может быть создана на основе данных из другой таблицы**.

Для этого используется запрос SELECT, результирующая таблица которого и будет новой таблицей базы данных. При этом имена столбцов запроса становятся именами столбцов новой таблицы. Запрос на создание новой таблицы имеет вид:

**CREATE TABLE имя\_таблицы AS**

**SELECT …**

Пример: Создать таблицу заказ (ordering), куда включить авторов и названия тех книг, количество экземпляров которых в таблице book меньше 4. Для всех книг указать одинаковое количество экземпляров 5.

**CREATE TABLE ordering AS**

**SELECT author, title, 5 AS amount ←** всё, что будет выбрано SELECT-ом будет новой таблицей

**FROM book**

**WHERE amount < 4;**

При создании таблицы можно использовать вложенные запросы как после SELECT, так и после WHERE.

Пример: Создать таблицу заказ (ordering), куда включить авторов и названия тех книг, количество экземпляров которых в таблице book меньше 4. Для всех книг указать одинаковое значение - среднее количество экземпляров книг в таблице book.

**CREATE TABLE ordering AS**

**SELECT author, title,**

**( ←Вложенный селект как часть внешнего SELECT-a**

**SELECT ROUND(AVG(amount))**

**FROM book**

**) AS amount**

**FROM book**

**WHERE amount < 4;**

Пример: Создать таблицу заказ (ordering), куда включить авторов и названия тех книг, количество экземпляров которых в таблице book меньше среднего количества экземпляров книг в таблице book. В таблицу включить столбец amount, в котором для всех книг указать одинаковое значение - среднее количество экземпляров книг в таблице book.

**CREATE TABLE ordering AS**

**SELECT author, title,**

**(SELECT AVG(amount)**

**FROM book <**

**AS amount**

**FROM book**

**WHERE amount < (SELECT AVG(amount)**

**FROM book);**

**SELECT \***

**FROM ordering**

**Создание таблицы с присвоением Foreign key**

Пример: создать таблицу book с внешним ключом. Внешним ключом здесь будет primary key из таблицы author.

**CREATE TABLE book (**

**book\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**title VARCHAR(50),**

**author\_id INT NOT NULL,**

**price DECIMAL(8,2),**

**amount INT,**

**FOREIGN KEY (author\_id) REFERENCES author (author\_id)**

**);**

**Также можно создовать в одной таблице несколько Foreign key.**

пример:

CREATE TABLE book (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(50),

author\_id INT NOT NULL,

genre\_id INT,

price DECIMAL(8,2),

amount INT,

FOREIGN KEY (author\_id) REFERENCES author (author\_id),

FOREIGN KEY (genre\_id) REFERENCES genre (genre\_id) ← 2-ой Foreign key

);

**Действия при удалении записи из главной таблицы**

С помощью выражения ON DELETE можно установить действия, которые выполняются для записей подчиненной таблицы при удалении связанной строки из главной таблицы. Т.е. если в главной таблице будут удалены какие то строки то и в подчинённой таблице надо это как-то обработать. При удалении можно установить следующие опции:

**CASCADE**: автоматически удаляет строки из зависимой таблицы при удалении связанных строк в главной таблице.

**SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение NULL. (В этом случае столбец внешнего ключа должен поддерживать установку NULL).

**SET DEFAULT** похоже на SET NULL за тем исключением, что значение внешнего ключа устанавливается не в NULL, а в значение по умолчанию для данного столбца.

**RESTRICT**: отклоняет удаление строк в главной таблице при наличии связанных строк в зависимой таблице.

Пример: Создать таблицу book Будем считать, что при удалении автора из таблицы author, должны удаляться все записи о книгах из таблицы book, написанные этим автором. А при удалении жанра из таблицы genre для соответствующей записи book установить значение Null в столбце genre\_id.

**CREATE TABLE book (**

**book\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**title VARCHAR(50),**

**author\_id INT NOT NULL,**

**genre\_id INT,**

**price DECIMAL(8,2),**

**amount INT,**

**FOREIGN KEY (author\_id) REFERENCES author (author\_id) ON DELETE CASCADE,**

**FOREIGN KEY (genre\_id) REFERENCES genre (genre\_id) ON DELETE SET NULL**

**);**

**УДАЛЕНИЕ ДАННЫХ ИЗ ТАБЛИЦЫ**

TRUNCATE nazvanie\_tablici <--stiraet vse dannie is tablici

delete from nazvani\_tablici <--stiraet vse dannie iz tablici. **(DELETE FROM page\_information WHERE name='intputForAnAnswer';)**

DROP TABLE nazvanie\_tablici; ← УДАЛЕНИЕ ТАБЛИЦЫ. ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

vnesenie izmenenii v tablicy

INSERT INTO salespeople VALUES(1001,'Peel','London',12), --- T.E. SNACHALA MI YKAZIVALI (INT, varchar, varchar, INT)

-> (1002,'Serres','San Jose',13), --- И ТАК С КАЖДОЙ ЗАПИСЬЮ (Т.Е. С СТРОЧКОЙ)

-> (1004,'Motika','London',11),

-> (1007,'Rifkin','Barcelona',15),

-> (1003,'Axelrod','New York', 10);

INT VARCHAR VARCHAR INT

mysql> SELECT \* FROM salespeople; <---ЭТО УЖЕ "ВЫБОРКА" С ПОМОЩЬЮ SELECT ВСЕГО, ЧТО ЕСТЬ В ТАБЛИЦЕ ПОД НАЗВАНИЕМ salespeople;

+------+---------+-----------+------+

| snum | sname | city | comm |

+------+---------+-----------+------+

| 1001 | Peel | London | 12 |

| 1002 | Serres | San Jose | 13 | <--- ПОЛУЧАЕМ РЕЗУЛЬТАТ

| 1003 | Axelrod | New York | 10 |

| 1004 | Motika | London | 11 |

| 1007 | Rifkin | Barcelona | 15 |

+------+---------+-----------+------+

CREATE TABLE Customers (cnum int NOT NULL,

cname VARCHAR (20) NOT NULL DEFAULT ' ',

city VARCHAR (20) NOT NULL DEFAULT ' ',

raiting INT NOT NULL DEFAULT 0,

snum int NOT NULL DEFAULT 0);

INSERT INTO Customers VALUES(2001,'Hoffman','London',100,1001),

(2002, 'Giovanni','Rome',200,1003),

(2003, 'Liu','San Jose',200,1002),

(2004, 'Grass','Berlin',300,1002),

(2006, 'Clements','London',100,1001),

(2008, 'Cisneros','San Jose', 300,1007),

(2007, 'Perejra','Rome',100,1004);

CREATE TABLE Orders (onum int,AMT FLOAT,ODATE DATE,CNUM INT, SNUM INT);

INSERT INTO Orders VALUES(3001,18.69,'1990-03-10',2008,1007),

(3003,767.19,'1990-03-10',2001,1001),

(3002,1900.10,'1990-03-10',2007,1004),

(3005,5160.45,'1990-03-10',2003,1002),

(3006,1098.16,'1990-03-10',2008,1007),

(3009,1713.23,'1990-04-19',2002,1003),

(3007,75.75,'1990-04-10',2004,1002),

(3008,4723.00,'1990-05-10',2006,1001),

(3010,1309.95,'1990-06-10',2004,1002),

(3011,9891.88,'1990-06-10',2006,1001);