Манипулирование данными

CREATE TABLE

```
CREATE TABLE STUDENT1
        (STUDENT_ID INTEGER,
        SURNAME VARCHAR(60),
        NAME VARCHAR(60),
        STIPEND DOUBLE,
        KURS INTEGER,
        CITY VARCHAR(60),
        BIRTHDAY DATE,
        UNIV_ID INTEGER);
```

Создали пустую таблицу без строк.
Определили имя таблицы и множество именованных столбцов.
Для каждого столбца определили тип и размер.
В создаваемой таблице должен быть по крайней мере один столбец.

INSERT

Добавляем строки с новыми данными в только что созданную или уже существующую таблицы

INSERT INTO STUDENT
VALUES (11000,'Иванов','Александр', 200, 3,'Москва','1979-10-06', 15);

INSERT INTO STUDENT1
VALUES (11000,'Иванов',NULL, 200, 3,'Москва','1979-10-06', 15);

INSERT INTO STUDENT1 (STUDENT_ID, CITY, SURNAME, NAME)
VALUES (101, 'Москва', 'Иванов', 'Саша');
Что случилось с теми столбцами, куда мы ничего не добавили?

INSERT

Извлекаем строки из одной таблицы и добавляем в другую

```
INSERT INTO STUDENT1
SELECT *
FROM STUDENT
WHERE CITY = 'Воронеж';
```

Важно, чтобы таблица STUDENT1 была уже создана и имела структуру аналогичную таблице STUDENT!

Задание 1

Введите запись для нового студента, которого зовут Орлов Николай, обучающегося на первом курсе ВГУ, живущего в Воронеже, сведения о дате рождения и размере стипендии неизвестны

Подзапросы с INSERT

Добавить в таблицу STUDENT1 сведения о всех студентах, которые учатся в Воронеже:

```
INSERT INTO STUDENT1
SELECT *
FROM STUDENT
WHERE UNIV_ID IN
        (SELECT UNIV_ID
        FROM UNIVERSITY
        WHERE CITY = 'Воронеж');
```

Подзапросы с INSERT

Заполнить таблицу STUDENT1 данными только о тех студентах, которые учатся там же, где живут:

```
INSERT INTO STUDENT1
SELECT *
FROM STUDENT A
WHERE CITY IN
        (SELECT CITY
        FROM UNIVERSITY B
        WHERE A.UNIV_ID = B.UNIV_ID);
```

DELETE

Удаляем строки из таблицы

Группу строк:

```
DELETE FROM STUDENT1
WHERE CITY = 'Москва';
```

Все строки:

```
DELETE FROM STUDENT1;
```

Если не удаляется:
SET SQL_SAFE_UPDATES = 0;

Задание 2

Заполните таблицу STUDENT1 данными о студентах, обучающихся в городе Воронеж и Москва

Подзапросы с DELETE

Удалить из таблицы STUDENT1 данные о студентах, учащихся в университетах с рейтингом, равным 296:

```
DELETE
FROM STUDENT1
WHERE EXISTS
        (SELECT *
        FROM UNIVERSITY
        WHERE RATING = 296
        AND STUDENT1.UNIV_ID = UNIVERSITY.UNIV_ID);
```

В FROM подзапроса нельзя ссылаться на таблицу, из которой осуществляется удаление

UPDATE

Изменяем значения некоторых или всех полей в существующей строке или строках таблицы

Устанавливаем для столбца STIPEND в таблице STUDENT1 значение 200 во всех строках:
UPDATE STUDENT1
SET STIPEND = 200;

Для студентов 4 курса назначаем стипендию в 300:
UPDATE STUDENT1
SET STIPEND = 300
WHERE KURS = 4;

Для всех студентов 2 курса по фамилии Иванов изменить город проживания на Москву:
UPDATE STUDENT1
SET SURNAME = 'Иванов', CITY = 'Москва‘
WHERE KURS = 2;

UPDATE

Использование скалярных выражений:

Увеличить размер всех стипендий в 2 раза в таблице STUDENT1:

```
UPDATE STUDENT1
SET STIPEND = STIPEND*2;
```

Увеличить стипендию студентов по фамилии Иванов в 2 раза:

```
UPDATE STUDENT1
SET STIPEND = STIPEND*2
WHERE SURNAME = 'Иванов';
```

Указать значение NULL в столбце STIPEND для всех студентов по фамилии Иванов:

```
UPDATE STUDENT1
SET STIPEND = NULL
WHERE SURNAME = 'Иванов';
```

Подзапросы с UPDATE

В таблице STUDENT1 увеличить на 20 стипендию всем студентам, сдавшим экзамены на 4 и 5:

```
UPDATE STUDENT1
SET STIPEND = STIPEND + 20
WHERE 4 <=
        (SELECT MIN(MARK)
        FROM EXAM_MARKS
        WHERE EXAM_MARKS.STUDENT_ID =STUDENT1.STUDENT_ID);
```

Задание 3

Напишите запрос, увеличивающий данные о величине стипендии на 20% всем студентам, у которых общая сумма баллов превышает значение 30.

Индексирование

Например, в таблице STUDENT1 необходимо часто осуществлять поиск фамилий студентов по полю STUDENT_ID (т.е. делать запросы с использованием SELECT).
Осуществление таких запросов можно ускорить, если создать индекс по полю STUDENT_ID:

CREATE INDEX STUDENT_ID_1 **ON** STUDENT1 (STUDENT_ID);

Индексы можно создавать по одному и по множеству полей.
Наличие индексов существенно замедляет операции INSERT и DELETE.

Сначала думаем о запросах, потом создаем структуру БД, индексы, дополнительные таблицы, … !

Удаление индекса:

DROP INDEX STUDENT_ID_1 **ON** student1;

Задание 4

Напишите команду, которая позволит быстро выбрать данные о студентах по курсам, на которых они учатся.

ALTER TABLE

Добавим новый столбец STREET в таблицу STUDENT1:

ALTER TABLE STUDENT1 **ADD** STREET character (50);

Посмотреть информацию о полях таблицы:

DESCRIBE STUDENT1;

Изменение описания столбцов (размер, тип данных, ограничений и т.п.):

ALTER TABLE STUDENT1 **MODIFY** STREET integer;

ALTER TABLE MODIFY

При модификации характеристик столбца есть некоторые ограничения:

1. Изменение типа данных возможно, если столбец пуст
2. Для незаполненного столбца можно изменять размер/точность
3. Для заполненного столбца можно увеличить размер/точность, но не понизить
4. Ограничение NOT NULL может быть установлено, если ни одно значение столбца не содержит значения NULL
5. Ограничение NOT NULL всегда можно отменить

Задание 5

Создайте новую таблицу, которая по структуре будет аналогична таблице LECTURER, добавьте столбец ANIMAL, куда будет вводиться информация о домашнем питомце преподавателя

DROP TABLE

Удаление таблицы:

1. Удалить из таблицы все данные
2. Удалить саму таблицу

Удаляем таблицу STUDENT1:

DELETE FROM STUDENT1;
DROP TABLE STUDENT1;

CREATE TABLE

CREATE TABLE STUDENT1 **LIKE** STUDENT;

Сохраняются все типы данных, размеры столбцов, индексы и ограничения.

Ограничения на множество допустимых значений

Ограничения на столбцы

```
CREATE TABLE STUDENT1
        (STUDENT_ID INTEGER NOT NULL,
        SURNAME VARCHAR (60) NOT NULL,
        NAME VARCHAR (60) NOT NULL,
        STIPEND DOUBLE,
        KURS INTEGER,
        CITY VARCHAR (60),
        BIRTHDAY DATE,
        UNIV_ID INTEGER);
```

Ограничения на множество допустимых значений

Ограничения на столбцы

```
CREATE TABLE STUDENT1
        (STUDENT_ID INTEGER NOT NULL UNIQUE,
        SURNAME VARCHAR(60) NOT NULL,
        NAME VARCHAR(60) NOT NULL,
        STIPEND DOUBLE,
        KURS INTEGER,
        CITY VARCHAR(60),
        BIRTHDAY DATE,
        UNIV_ID INTEGER);
```

Ограничения на множество допустимых значений

Ограничения на таблицу

Каждый студент в один день может сдавать только один экзамен:

```
CREATE TABLE EXAM_MARKS1
        (EXAM_ID INTEGER NOT NULL,
        STUDENT_ID INTEGER NOT NULL,
        SUBJ_ID INTEGER NOT NULL,
        MARK CHAR (1),
        EXAM_DATE DATE NOT NULL,
        UNIQUE (STUDENT_ID, EXAM_DATE));
```

Ограничения на множество допустимых значений

Ограничения на таблицу

Каждый студент в один день может сдавать только один экзамен:

```
CREATE TABLE EXAM_MARKS1
        (EXAM_ID INTEGER NOT NULL,
        STUDENT_ID INTEGER NOT NULL,
        SUBJ_ID INTEGER NOT NULL,
        MARK CHAR (1),
        EXAM_DATE DATE NOT NULL,
        CONSTRAINT STUD_SUBJ_CONSTR – теперь у ограничения есть имя
        UNIQUE (STUDENT_ID, EXAM_DATE));
```

Первичные ключи

Таблица может содержать только один первичный ключ.
Внешние ключи по умолчанию ссылаются на первичный ключ таблицы.
Первичный ключ является идентификатором строк таблицы.

```
CREATE TABLE STUDENT2
        (STUDENT_ID INTEGER PRIMARY KEY,
        SURNAME CHAR (25) NOT NULL,
        NAME CHAR (10) NOT NULL,
        STIPEND INTEGER,
        KURS INTEGER,
        CITY CHAR (15),
        BIRTHDAY DATE,
        UNIV_ID INTEGER);
```

Составные первичные ключи

Несколько полей составляют уникальную для идентификации строк комбинацию значений

```
CREATE TABLE NEW_EXAM_MARKS
        (EXAM_ID INTEGER NOT NULL,
        STUDENT_ID INTEGER NOT NULL,
        SUBJ_ID INTEGER NOT NULL,
        MARK INTEGER,
        DATA DATE,
CONSTRAINT EX_PR_KEY PRIMARY KEY (EXAM_ID, STUDENT_ID));
```

Проверка значений полей

Размер стипендии не может быть меньше 200

```
CREATE TABLE STUDENT3
        (STUDENT_ID INTEGER PRIMARY KEY,
        SURNAME CHAR (25) NOT NULL,
        NAME CHAR (10) NOT NULL,
        STIPEND INTEGER CHECK (STIPEND < 200),
        KURS INTEGER,
        CITY CHAR (15),
        BIRTHDAY DATE,
        UNIV_ID INTEGER);
```

Проверка ограничивающих условий

Размер стипендии не может быть меньше 200 только для студентов из Воронежа:

```
CREATE TABLE STUDENT3
        (STUDENT_ID INTEGER PRIMARY KEY,
        SURNAME CHAR (25) NOT NULL,
        NAME CHAR (10) NOT NULL,
        STIPEND INTEGER,
        KURS INTEGER,
        CITY CHAR (15),
        BIRTHDAY DATE,
        UNIV_ID INTEGER)
        CHECK(STIPEND < 200 AND CITY = 'Воронеж'));
```

Установка значений по умолчанию

DEFAULT не ограничивает значения, вводимые в поле, а конкретизирует значение поля, если оно не было задано

```
CREATE TABLE STUDENT3
        (STUDENT_ID INTEGER PRIMARY KEY,
        SURNAME CHAR (25) NOT NULL,
        NAME CHAR (10) NOT NULL,
        STIPEND INTEGER CHECK (STIPEND < 200),
        KURS INTEGER,
        CITY CHAR (15) DEFAULT 'Воронеж',
        BIRTHDAY DATE,
        UNIV_ID INTEGER);
```

По умолчанию отсутствующие данные заполняются NULL, что не всегда удобно и нужно

Задание 6

Создайте таблицу EXAM_MARKS2 так, чтобы не допускался ввод в таблицу двух записей об оценках одного студента по конкретным экзамену и предмету обучения, а также, чтобы экзамен по предмету с индексом 3 не проводили в апреле.