Команды для работы с базами данных

1. Просмотр доступных баз данных

```
SHOW DATABASES;
```

2. Создание новой базы данных

```
CREATE DATABASE;
```

3. Выбор базы данных для использования

```
USE <database_name>;
```

4. Импорт SQL-команд из файла .sql

```
SOURCE <path_of_.sql_file>;
```

5. Удаление базы данных

```
DROP DATABASE <database_name>;
```

Работа с таблицами

6. Просмотр таблиц, доступных в базе данных

```
SHOW TABLES;
```

7. Создание новой таблицы

```
CREATE TABLE <table_name1> (
  <col_name1> <col_type1>,
  <col_name2> <col_type2>,
  <col_name3> <col_type3>
  PRIMARY KEY (<col_name1>),
  FOREIGN KEY (<col_name2>) REFERENCES <table_name2>(<col_name2>)
);
```

Ограничения целостности при использовании *CREATE TABLE*

Может понадобиться создать ограничения для определённых столбцов в таблице. При создании таблицы можно задать следующие ограничения:

- ячейка таблицы не может иметь значение NULL;
- первичный ключ — `PRIMARY KEY (col_name1, col_name2, …)`;

- внешний ключ — `FOREIGN KEY (col_namex1, …, col_namexn) REFERENCES table_name(col_namex1, …, col_namexn)`.

Можно задать больше одного первичного ключа. В этом случае получится составной первичный ключ.

Пример

Создайте таблицу «instructor»:

```
CREATE TABLE instructor (
  ID CHAR(5),
  name VARCHAR(20) NOT NULL,
  dept_name VARCHAR(20),
  salary NUMERIC(8,2),
  PRIMARY KEY (ID),
  FOREIGN KEY (dept_name) REFERENCES department(dept_name)
);
```

8. Сведения о таблице

Можно просмотреть различные сведения (тип значений, является ключом или нет) о столбцах таблицы следующей командой:

```
DESCRIBE <table_name>;
```

9. Добавление данных в таблицу

```
INSERT INTO <table_name> (<col_name1>, <col_name2>, <col_name3>, …)
  VALUES (<value1>, <value2>, <value3>, …);
```

При добавлении данных в каждый столбец таблицы не требуется указывать названия столбцов.

```
INSERT INTO <table_name>
  VALUES (<value1>, <value2>, <value3>, …);
```

10. Обновление данных таблицы

```
UPDATE <table_name>
  SET <col_name1> = <value1>, <col_name2> = <value2>, ...
  WHERE <condition>;
```

11. Удаление всех данных из таблицы

```
DELETE FROM <table_name>;
```

12. Удаление таблицы

```
DROP TABLE <table_name>;
```

Команды для создания запросов

13. SELECT

`SELECT` используется для получения данных из определённой таблицы:

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>;
```

Следующей командой можно вывести все данные из таблицы:

```
SELECT * FROM <table_name>;
```

14. SELECT DISTINCT

В столбцах таблицы могут содержаться повторяющиеся данные. Используйте `SELECT DISTINCT` для получения только неповторяющихся данных.

```
SELECT DISTINCT <col_name1>, <col_name2>, …
  FROM <table_name>;
```

15. WHERE

Можно использовать ключевое слово `WHERE` в `SELECT` для указания условий в запросе:

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>
  WHERE <condition>;
```

В запросе можно задавать следующие условия:

- сравнение текста;
- сравнение численных значений;
- логические операции AND (и), OR (или) и NOT (отрицание).

Пример

Попробуйте выполнить следующие команды. Обратите внимание на условия, заданные в `WHERE`:

```
SELECT * FROM course WHERE dept_name=’Comp. Sci.’;
SELECT * FROM course WHERE credits>3;
SELECT * FROM course WHERE dept_name='Comp. Sci.' AND credits>3;
```

16. GROUP BY

Оператор `GROUP BY` часто используется с агрегатными функциями, такими как `COUNT`, `MAX`, `MIN`, `SUM` и `AVG`, для группировки выходных значений.

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>
  GROUP BY <col_namex>;
```

Пример

Выведем количество курсов для каждого факультета:

```
SELECT COUNT(course_id), dept_name
  FROM course
  GROUP BY dept_name;
```

17. HAVING

Ключевое слово `HAVING` было добавлено в SQL потому, что `WHERE` не может быть использовано для работы с агрегатными функциями.

```
SELECT <col_name1>, <col_name2>, ...
  FROM <table_name>
  GROUP BY <column_namex>
  HAVING <condition>
```

Пример

Выведем список факультетов, у которых более одного курса:

```
SELECT COUNT(course_id), dept_name
  FROM course
  GROUP BY dept_name
  HAVING COUNT(course_id)>1;
```

18. ORDER BY

`ORDER BY` используется для сортировки результатов запроса по убыванию или возрастанию. `ORDER BY` отсортирует по возрастанию, если не будет указан способ сортировки `ASC` или `DESC`.

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>
  ORDER BY <col_name1>, <col_name2>, … ASC|DESC;
```

Пример

Выведем список курсов по возрастанию и убыванию количества кредитов:

```
SELECT * FROM course ORDER BY credits;
SELECT * FROM course ORDER BY credits DESC;
```

19. BETWEEN

`BETWEEN` используется для выбора значений данных из определённого промежутка. Могут быть использованы числовые и текстовые значения, а также даты.

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>
```

```
  WHERE <col_namex> BETWEEN <value1> AND <value2>;
```

Пример

Выведем список инструкторов, чья зарплата больше 50 000, но меньше 100 000:

```
SELECT * FROM instructor
  WHERE salary BETWEEN 50000 AND 100000;
```

20. LIKE

Оператор `LIKE` используется в `WHERE`, чтобы задать шаблон поиска похожего значения.

Есть два свободных оператора, которые используются в `LIKE`:

- `%` (ни одного, один или несколько символов);
- `_` (один символ).

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>
  WHERE <col_namex> LIKE <pattern>;
```

Пример

Выведем список курсов, в имени которых содержится «to», и список курсов, название которых начинается с «CS-»:

```
SELECT * FROM course WHERE title LIKE ‘%to%’;
SELECT * FROM course WHERE course_id LIKE 'CS-___';
```

21. IN

С помощью `IN` можно указать несколько значений для оператора `WHERE`:

```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name>
  WHERE <col_namen> IN (<value1>, <value2>, …);
```

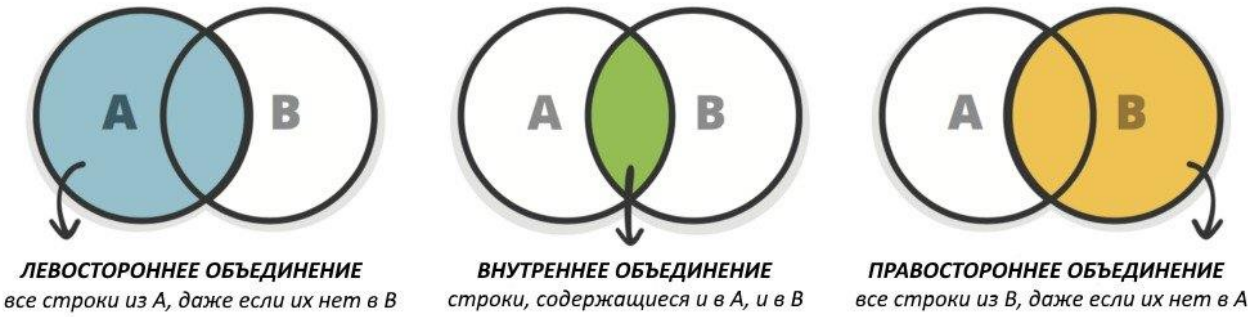
Пример

Выведем список студентов с направлений Comp. Sci., Physics и Elec. Eng.:

```
SELECT * FROM student
  WHERE dept_name IN (‘Comp. Sci.’, ‘Physics’, ‘Elec. Eng.’);
```

22. JOIN

`JOIN` используется для связи двух или более таблиц с помощью общих атрибутов внутри них. На изображении ниже показаны различные способы объединения в SQL. Обратите внимание на разницу между левым внешним объединением и правым внешним объединением:




```
SELECT <col_name1>, <col_name2>, …
  FROM <table_name1>
  JOIN <table_name2>
  ON <table_name1.col_namex> = <table2.col_namex>;
```

Пример 1

Выведем список всех курсов и соответствующую информацию о факультетах:

```
SELECT * FROM course
    JOIN department
    ON course.dept_name=department.dept_name;
```

Пример 2

Выведем список всех обязательных курсов и детали о них:

```
SELECT prereq.course_id, title, dept_name, credits, prereq_id
  FROM prereq
  LEFT OUTER JOIN course
  ON prereq.course_id=course.course_id;
```

Пример 3

Выведем список всех курсов вне зависимости от того, обязательны они или нет:

```
SELECT course.course_id, title, dept_name, credits, prereq_id
  FROM prereq
  RIGHT OUTER JOIN course
  ON prereq.course_id=course.course_id;
```

23. View

`View` — это виртуальная таблица SQL, созданная в результате выполнения выражения. Она содержит строки и столбцы и очень похожа на обычную SQL-таблицу. `View` всегда показывает самую свежую информацию из базы данных.

Создание

```
CREATE VIEW <view_name> AS
```

```
SELECT <col_name1>, <col_name2>, …
FROM <table_name>
WHERE <condition>;
```

Удаление

```
DROP VIEW <view_name>;
```

24. Агрегатные функции

Эти функции используются для получения совокупного результата, относящегося к рассматриваемым данным. Ниже приведены общеупотребительные агрегированные функции:

- `COUNT (col_name)` — возвращает количество строк;
- `SUM (col_name)` — возвращает сумму значений в данном столбце;
- `AVG (col_name)` — возвращает среднее значение данного столбца;
- `MIN (col_name)` — возвращает наименьшее значение данного столбца;
- `MAX (col_name)` — возвращает наибольшее значение данного столбца.

25. Вложенные подзапросы

Вложенные подзапросы — это SQL-запросы, которые включают выражения `SELECT`, `FROM` и `WHERE`, вложенные в другой запрос.

Пример

Найдём курсы, которые преподавались осенью 2009 и весной 2010 годов:

```
SELECT DISTINCT course_id
  FROM section
  WHERE semester = 'Fall' AND year= 2009 AND course_id IN (
    SELECT course_id
    FROM section
    WHERE semester = 'Spring' AND year= 2010
  );
```