Structured Query Language

Курс «Базы данных»

Вадим Цесько

Санкт-Петербургский государственный политехнический университет

15 марта 2012 г.

Содержание

Определения

В SQL определены два подмножества языка:

Определения

В SQL определены два подмножества языка:

- **SQL-DDL** (Data Definition Language) — язык определения структур и ограничений целостности БД: создание и удаление БД; создание, изменение и удаление таблиц; управление пользователями и т. д.

Определения

В SQL определены два подмножества языка:

- **SQL-DDL** (Data Definition Language) — язык определения структур и ограничений целостности БД: создание и удаление БД; создание, изменение и удаление таблиц; управление пользователями и т. д.
- **SQL-DML** (Data Manipulation Language) — язык манипулирования данными: добавление, изменение, удаление и извлечение данных; управление транзакциями

История и стандарты

- 1970, IBM, System R, язык SEQUEL
- 1981, IBM, SQL/DS
- Первый стандарт SQL — ANSI и ISO 1986
- SQL dialect 1 — ANSI 1987
- SQL-92
- SQL:1999
- SQL:2003

Типы данных

Целые числа:

- `INTEGER` — обычное целое, 4 байта
- `SMALLINT` — короткое целое, 2 байта

Типы данных

Целые числа:

- `INTEGER` — обычное целое, 4 байта
- `SMALLINT` — короткое целое, 2 байта

Символьные строки:

- `CHAR(n)` — строка длины `n`
- `VARCHAR(n)` — строка переменной длины, но не более `n`

Типы данных

Целые числа:

- `INTEGER` — обычное целое, 4 байта
- `SMALLINT` — короткое целое, 2 байта

Символьные строки:

- `CHAR(n)` — строка длины `n`
- `VARCHAR(n)` — строка переменной длины, но не более `n`

Вещественные числа:

- `FLOAT` — число с плавающей запятой, 4 байта
- `DECIMAL(p, n)` — число с фиксированной запятой (`p` — число знаков до запятой, `n` — после запятой)
- `DOUBLE PRECISION` — число с плавающей запятой, 8 байт

Типы данных

Целые числа:

- `INTEGER` — обычное целое, 4 байта
- `SMALLINT` — короткое целое, 2 байта

Символьные строки:

- `CHAR(n)` — строка длины `n`
- `VARCHAR(n)` — строка переменной длины, но не более `n`

Вещественные числа:

- `FLOAT` — число с плавающей запятой, 4 байта
- `DECIMAL(p, n)` — число с фиксированной запятой (`p` — число знаков до запятой, `n` — после запятой)
- `DOUBLE PRECISION` — число с плавающей запятой, 8 байт

Другие типы данных:

- `BLOB` — двоичные типы данных большого размера
- `DATA` — дата с точностью до дня (единица измерения — день)
- `TIME` — время с точностью до мс (единица измерения — секунда)
- `TIMESTAMP` — дата и время вместе

SQL-DDL

Основные операторы управления метаданными:

- `CREATE` — создать
- `ALTER` — изменить
- `DROP` — удалить

SQL-DDL

Основные операторы управления метаданными:

- `CREATE` — создать
- `ALTER` — изменить
- `DROP` — удалить

Выполняются над сущностями:

- `DOMAIN`
- `TABLE`
- `VIEW`
- `PROCEDURE`
- `TRIGGER`
- `GENERATOR`
- `INDEX`
- ...

Создание БД

Создание БД

```
CREATE DATABASE <путь к БД>
    USER <логин>
    PASSWORD <пароль> [...];
```

Создание БД

Создание БД

```
CREATE DATABASE <путь к БД>
    USER <логин>
    PASSWORD <пароль> [...];
```

Пример создания БД

```
CREATE DATABASE 'localhost:/var/db/stud.fdb'
    USER 'SYSDBA'
    PASSWORD 'masterkey'
    PAGE_SIZE 16384
    DEFAULT CHARACTER SET UTF8;
```

Соединение и удаление БД

Соединение с БД

```
CONNECT DATABASE <путь к БД>
    USER <логин>
    PASSWORD <пароль> [...];
```

Соединение и удаление БД

Соединение с БД

```
CONNECT DATABASE <путь к БД>
    USER <логин>
    PASSWORD <пароль> [...];
```

Удаление текущей БД

```
DROP DATABASE;
```

Создание таблиц

Создание таблицы

```
CREATE TABLE <имя_таблицы> (<поле>, ....)
<поле> = <имя_поля>
    {<тип_поля> | COMPUTED BY (<выражение>) | <домен>}
    [DEFAULT {<литерал> | NULL | USER | ...}]
    [NULL | NOT NULL]
    [<ограничения_поля>]
```

Создание таблиц

Создание таблицы

```
CREATE TABLE <имя_таблицы> (<поле>, ....)
<поле> = <имя_поля>
    {<тип_поля> | COMPUTED BY (<выражение>) | <домен>}
    [DEFAULT {<литерал> | NULL | USER | ...}]
    [NULL | NOT NULL]
    [<ограничения_поля>]
```

Пример создания таблицы

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    yob INTEGER NOT NULL DEFAULT 1984,
    age COMPUTED BY (current_date - yob))
```

Изменение и удаление таблиц

Изменение таблицы

```
ALTER TABLE <имя_таблицы>
    {ADD <поле>| DROP <имя_поля>}, ...
```

Изменение и удаление таблиц

Изменение таблицы

```
ALTER TABLE <имя_таблицы>
    {ADD <поле>| DROP <имя_поля>}, ...
```

Пример изменения таблицы

```
ALTER TABLE students
    ADD hobby VARCHAR(64) NOT NULL,
    DROP group_id;
```

Изменение и удаление таблиц

Изменение таблицы

```
ALTER TABLE <имя_таблицы>
    {ADD <поле>| DROP <имя_поля>}, ...
```

Пример изменения таблицы

```
ALTER TABLE students
    ADD hobby VARCHAR(64) NOT NULL,
    DROP group_id;
```

Удаление таблицы

```
DROP TABLE <имя_таблицы>;
```

Изменение и удаление таблиц

Изменение таблицы

```
ALTER TABLE <имя_таблицы>
    {ADD <поле>| DROP <имя_поля>}, ...
```

Пример изменения таблицы

```
ALTER TABLE students
    ADD hobby VARCHAR(64) NOT NULL,
    DROP group_id;
```

Удаление таблицы

```
DROP TABLE <имя_таблицы>;
```

Пример удаления таблицы

```
DROP TABLE students;
```

Ограничения

Ограничения на значения полей таблиц

```
<ограничение_поля> = [CONSTRAINT <имя_ограничения>]
    { PRIMARY KEY |
      UNIQUE |
      FOREIGN KEY (<имя_поля>)
        REFERENCES <имя_главной_таблицы> (<имя_поля>) |
      CHECK <выражение> }
```

Ограничение первичного ключа

Пример определения ограничения PRIMARY KEY

```
CREATE TABLE students(
    id INTEGER NOT NULL PRIMARY KEY,
    group_id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    age INTEGER NOT NULL);
```

Ограничение первичного ключа

Пример определения ограничения PRIMARY KEY

```
CREATE TABLE students(
    id INTEGER NOT NULL PRIMARY KEY,
    group_id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    age INTEGER NOT NULL);
```

Пример добавления ограничения PRIMARY KEY

```
ALTER TABLE students
    ADD CONSTRAINT students_pk PRIMARY KEY (id);
```

Ограничение уникальности

Пример определения ограничения UNIQUE

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    group_id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    passport VARCHAR(32) NOT NULL UNIQUE,
    age INTEGER NOT NULL);
```

Ограничение уникальности

Пример определения ограничения UNIQUE

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    group_id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    passport VARCHAR(32) NOT NULL UNIQUE,
    age INTEGER NOT NULL);
```

Пример добавления ограничения UNIQUE

```
ALTER TABLE students
    ADD CONSTRAINT students_passport_unique
        UNIQUE (passport);
```

Ограничение на диапазон значений

Пример определения ограничения CHECK

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    group_id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    age INTEGER NOT NULL CHECK (age > 18));
```

Ограничение на диапазон значений

Пример определения ограничения CHECK

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    group_id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    age INTEGER NOT NULL CHECK (age > 18));
```

Пример добавления ограничения CHECK

```
ALTER TABLE students
    ADD CONSTRAINT students_age_check
        CHECK (age > 18);
```

Ограничение внешнего ключа

Пример определения ограничения FOREIGN KEY

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    group_id INTEGER NOT NULL,
    CONSTRAINT students_group_fk
        FOREIGN KEY (group_id)
            REFERENCES groups (id));
```

Ограничение внешнего ключа

Пример определения ограничения FOREIGN KEY

```
CREATE TABLE students(
    id INTEGER NOT NULL,
    fname VARCHAR(32) NOT NULL,
    lname VARCHAR(32) NOT NULL,
    group_id INTEGER NOT NULL,
    CONSTRAINT students_group_fk
        FOREIGN KEY (group_id)
            REFERENCES groups (id));
```

Пример добавления ограничения FOREIGN KEY

```
ALTER TABLE students
    ADD CONSTRAINT students_group_fk
        FOREIGN KEY (group_id)
            REFERENCES groups (id);
```

SQL-DML

Основные операторы управления данными:

- `SELECT` — извлечь
- `INSERT` — добавить
- `UPDATE` — изменить
- `DELETE` — удалить

Извлечение данных

Оператор `SELECT` — оператор извлечения данных из таблиц в SQL. Соответствует операциям реляционной алгебры: проекция, выборка, соединение.

Извлечение данных

Оператор SELECT — оператор извлечения данных из таблиц в SQL. Соответствует операциям реляционной алгебры: проекция, выборка, соединение.

Синтаксис оператора SELECT

```
SELECT [ALL | DISTINCT] <список_выбора>
    FROM <имя_таблицы>, ...
    [ WHERE <условие> ]
    [ GROUP BY <имя_столбца>, ... ]
    [ HAVING <условие> ]
    [ ORDER BY <имя_столбца> [ASC | DESC], ... ]
```

Извлечение данных

Оператор SELECT — оператор извлечения данных из таблиц в SQL. Соответствует операциям реляционной алгебры: проекция, выборка, соединение.

Синтаксис оператора SELECT

```
SELECT [ALL | DISTINCT] <список_выбора>
    FROM <имя_таблицы>, ...
    [ WHERE <условие> ]
    [ GROUP BY <имя_столбца>, ... ]
    [ HAVING <условие> ]
    [ ORDER BY <имя_столбца> [ASC | DESC], ... ]
```

Внимание

Порядок предложений должен строго соблюдаться.

Схема БД для примеров

На доске.

Псевдонимы и условия

Операторы

=, <>, >, <, AND, OR, NOT, LIKE, BETWEEN ... AND ..., IN (...).

Псевдонимы и условия

Операторы

=, <>, >, <, AND, OR, NOT, LIKE, BETWEEN ... AND ..., IN (...).

Примеры запросов SELECT

```
SELECT * FROM students
SELECT id, fname, lname as "Фамилия", age FROM students
SELECT fname, lname FROM students
    WHERE (fname like '%A') and (age > 20)
SELECT fname, lname FROM students
    WHERE not (id between 100 and 200)
SELECT fname, lname FROM students
    WHERE lname in ('Ivanov', 'Petrov', 'Sidorov')
```

Соединение таблиц и вычисляемые поля

Соединение нескольких таблиц

```
SELECT fname, lname, number FROM students, groups
SELECT fname, lname, number FROM students, groups
    WHERE students.group_id = groups.id
SELECT fname, lname FROM students, groups
    WHERE (students.group_id = groups.id) and
          (groups.number = '4081/2')
SELECT fname, lname, specs.name FROM students, groups, specs
    WHERE (students.group_id = groups.id) and
          (groups.spec_id = specs.id)
```

Соединение таблиц и вычисляемые поля

Соединение нескольких таблиц

```
SELECT fname, lname, number FROM students, groups
SELECT fname, lname, number FROM students, groups
    WHERE students.group_id = groups.id
SELECT fname, lname FROM students, groups
    WHERE (students.group_id = groups.id) and
          (groups.number = '4081/2')
SELECT fname, lname, specs.name FROM students, groups, specs
    WHERE (students.group_id = groups.id) and
          (groups.spec_id = specs.id)
```

Вычисляемые поля

```
SELECT fname, lname, (2008 - age) AS YoB FROM students
```

Агрегатные функции

Агрегатные функции

`MAX`, `MIN`, `COUNT`, `AVG`, `SUM`.

Агрегатные функции

Агрегатные функции

MAX, MIN, COUNT, AVG, SUM.

Вычисление совокупных характеристик

```
SELECT MAX(age) FROM students
SELECT AVG(mark)
    FROM students, stud_results
    WHERE (students.id = stud_results.student_id) and
          (lname = 'Ivanov')
```

Группировка данных

Группировка данных при помощи GROUP BY

```
SELECT groups.id FROM students, groups
    WHERE students.group_id = groups.id
SELECT groups.id, COUNT(students.id) FROM students, groups
    WHERE students.group_id = groups.id
    GROUP BY groups.id
SELECT fname, lname, AVG(mark) FROM students, stud_results
    WHERE students.id = stud_results.student_id
    GROUP BY lname, fname
```

Группировка данных

Группировка данных при помощи GROUP BY

```
SELECT groups.id FROM students, groups
    WHERE students.group_id = groups.id
SELECT groups.id, COUNT(students.id) FROM students, groups
    WHERE students.group_id = groups.id
    GROUP BY groups.id
SELECT fname, lname, AVG(mark) FROM students, stud_results
    WHERE students.id = stud_results.student_id
    GROUP BY lname, fname
```

Внимание

При группировке в выбираемом списке могут быть только поля, по которым делается группировка, и агрегатные функции.

Ограничения на результаты группировки

Ограничения на результаты группировки при помощи HAVING

```
SELECT groups.id, COUNT(students.id) as s_count
FROM students, groups
WHERE students.group_id = groups.id
GROUP BY groups.id
HAVING COUNT(students.id) > 10
```

Ограничения на результаты группировки

Ограничения на результаты группировки при помощи HAVING

```
SELECT groups.id, COUNT(students.id) as s_count
FROM students, groups
WHERE students.group_id = groups.id
GROUP BY groups.id
HAVING COUNT(students.id) > 10
```

Внимание

Условия могут содержать только выбираемые поля.

Сортировка и исключение повторений

Сортировка при помощи ORDER BY

```
SELECT fname, lname, age FROM students
    ORDER BY age
SELECT fname, lname, age FROM students
    ORDER BY fname DESC, lname
```

Сортировка и исключение повторений

Сортировка при помощи ORDER BY

```
SELECT fname, lname, age FROM students
    ORDER BY age
SELECT fname, lname, age FROM students
    ORDER BY fname DESC, lname
```

Исключение повторений при помощи DISTINCT

```
SELECT DISTINCT fname, lname FROM students
```

Вложенные запросы

В конструкции WHERE

```
SELECT fname, lname FROM students
    WHERE age = (SELECT MAX(age) FROM students)
SELECT fname, lname FROM students
    WHERE (SELECT AVG(mark) FROM stud_results
           WHERE stud_results.student_id = students.id) > 3.5
```

Вложенные запросы

В конструкции WHERE

```
SELECT fname, lname FROM students
    WHERE age = (SELECT MAX(age) FROM students)
SELECT fname, lname FROM students
    WHERE (SELECT AVG(mark) FROM stud_results
           WHERE stud_results.student_id = students.id) > 3.5
```

В конструкции FROM (виртуальные таблицы)

```
SELECT fname, lname, avg_mark FROM students,
    (SELECT students.id as s_id, AVG(mark) as avg_mark
     FROM students, stud_results
     WHERE stud_results.student_id = students.id
     GROUP BY students.id) as avg_marks
WHERE (students.id = avg_marks.s_id) and
      (avg_marks.avg_mark > 3.5)
```

Ограничение набора записей

Операторы

- FIRST <число>
- SKIP <число>
- ROWS <число> [TO <число>]

Ограничение набора записей

Операторы

- FIRST <число>
- SKIP <число>
- ROWS <число> [TO <число>]

Выбор диапазона результата

```
SELECT FIRST 1 fname, lname
    FROM students WHERE age = 21
SELECT FIRST 2 SKIP 1 fname, lname
    FROM students WHERE age = 21
SELECT fname, lname
    FROM students WHERE age = 21
    ROWS 2 TO 3
```

Преобразование типов

Синтаксис

```
CAST (<поле> AS <требуемый_тип>).
```

Преобразование типов

Синтаксис

```
CAST (<поле> AS <требуемый_тип>).
```

Преобразование типов

```
SELECT fname, lname, AVG(CAST(mark AS float))
FROM students, stud_results
WHERE students.id = stud_results.student_id
GROUP BY fname, lname
```

Работа с датой и временем

Операторы

- CURRENT_DATE
- CURRENT_TIME
- EXTRACT (YEAR | MONTH | DAY FROM <поле>)
- CAST('NOW' AS DATE | TIME | TIMESTAMP)

Работа с датой и временем

Операторы

- CURRENT_DATE
- CURRENT_TIME
- EXTRACT (YEAR | MONTH | DAY FROM <поле>)
- CAST('NOW' AS DATE | TIME | TIMESTAMP)

Дата и время

```
SELECT fname, lname, CURRENT_DATE - CAST(birthdate AS DATE)
    FROM students
    WHERE EXTRACT(month FROM birthdate) = 3

CAST ('22.03.2010' AS DATE)
```

Значение NULL

Трёхзначная логика: TRUE, FALSE и NULL.

Значение NULL

Трёхзначная логика: TRUE, FALSE и NULL.

Свойства NULL

- Значение NULL применимо к атрибутам любого типа
- Значение NULL недопустимо для PK и допустимо для FK
- Любая операция с полем, содержащим NULL, даёт NULL

Значение NULL

Трёхзначная логика: TRUE, FALSE и NULL.

Свойства NULL

- Значение NULL применимо к атрибутам любого типа
- Значение NULL недопустимо для PK и допустимо для FK
- Любая операция с полем, содержащим NULL, даёт NULL

Примеры

Пусть A = 1, B = NULL. Тогда:

- A + B = NULL
- A = B = NULL
- A || B = NULL
- A <> B = NULL
- B = NULL = NULL
- NOT (B) = NULL

Операции с NULL-операндами

AND и OR с NULL

- NULL OR FALSE = NULL
- NULL OR TRUE = TRUE
- NULL OR NULL = NULL
- NULL AND FALSE = FALSE
- NULL AND TRUE = NULL
- NULL AND NULL = NULL

Операции с NULL-операндами

AND и OR с NULL

- NULL OR FALSE = NULL
- NULL OR TRUE = TRUE
- NULL OR NULL = NULL
- NULL AND FALSE = FALSE
- NULL AND TRUE = NULL
- NULL AND NULL = NULL

Внимание

Нужно учитывать возможность появления NULL для полей, у которых явно не указано NOT NULL. NULL может появляться при операциях и при вычислении условий в конструкциях WHERE, HAVING и т. д.

Операции с NULL-операндами

AND и OR с NULL

- NULL OR FALSE = NULL
- NULL OR TRUE = TRUE
- NULL OR NULL = NULL
- NULL AND FALSE = FALSE
- NULL AND TRUE = NULL
- NULL AND NULL = NULL

Внимание

Нужно учитывать возможность появления NULL для полей, у которых явно не указано NOT NULL. NULL может появляться при операциях и при вычислении условий в конструкциях WHERE, HAVING и т. д.

Агрегатные функции

NULL игнорируется. За исключением COUNT(*).

Корректная обработка NULL

Примеры запросов, не учитывающих появление NULL

```
SELECT fname || lname FROM students;
IF (A <> B) THEN result = 'NOT EQUAL'
ELSE result = 'EQUAL';
```

Корректная обработка NULL

Примеры запросов, не учитывающих появление NULL

```
SELECT fname || lname FROM students;
IF (A <> B) THEN result = 'NOT EQUAL'
ELSE result = 'EQUAL';
```

Обработка NULL

DISTINCT (<>) и NOT DISTINCT (=) — результат только TRUE или FALSE.

Корректная обработка NULL

Примеры запросов, не учитывающих появление NULL

```
SELECT fname || lname FROM students;
IF (A <> B) THEN result = 'NOT EQUAL'
ELSE result = 'EQUAL';
```

Обработка NULL

DISTINCT (<>) и NOT DISTINCT (=) — результат только TRUE или FALSE.

Примеры запросов, не учитывающих появление NULL

```
IF (A IS DISTINCT B) THEN result = 'NOT EQUAL'
ELSE result = 'EQUAL';
```

Корректная обработка NULL

Примеры запросов, не учитывающих появление NULL

```
SELECT fname || lname FROM students;
IF (A <> B) THEN result = 'NOT EQUAL'
ELSE result = 'EQUAL';
```

Обработка NULL

DISTINCT (<>) и NOT DISTINCT (=) — результат только TRUE или FALSE.

Примеры запросов, не учитывающих появление NULL

```
IF (A IS DISTINCT B) THEN result = 'NOT EQUAL'
ELSE result = 'EQUAL';
```

Явная проверка на NULL

IS NULL, IS NOT NULL, NULLIF(<выражение1>, <выражение2>).

Добавление данных

Добавление одной записи

```
INSERT INTO <имя_таблицы> [(<имя_поля>, ...)]
    VALUES (<значение>, ...)
```

Добавление данных

Добавление одной записи

```
INSERT INTO <имя_таблицы> [(<имя_поля>, ...)]
    VALUES (<значение>, ...)
```

Примеры

```
INSERT INTO students VALUES (1, 'Иван', 'Иванов', 2, 24);
INSERT INTO students (id, fname, group_id)
    VALUES (1, 'Иван', 2);
INSERT INTO students
    VALUES (1, 'Иван', 'Иванов',
            (SELECT groups.id
                FROM groups
                WHERE number = '4081/1'),
            24);
```

Ещё один пример добавления данных

Пример

```
INSERT INTO students
    VALUES (1,
            'Иван',
            'Иванов',
            (SELECT gid FROM
                (SELECT FIRST 1
                    groups.id AS gid,
                    COUNT(students.id) AS s_count
                FROM groups, students
                WHERE students.group_id = groups.id
                GROUP BY groups.id
                ORDER BY s_count)),
            24);
```

Удаление данных

Удаление одной или нескольких записей

```
DELETE FROM <имя_таблицы> [WHERE <условие>]
```

Удаление данных

Удаление одной или нескольких записей

```
DELETE FROM <имя_таблицы> [WHERE <условие>]
```

Примеры

```
DELETE FROM students;
DELETE FROM groups WHERE number = '4081/1';
DELETE FROM students WHERE age =
    (SELECT MAX(age) FROM students);
DELETE FROM students WHERE
    (SELECT AVG(CAST(mark AS float))
     FROM stud_results
     WHERE stud_results.student_id = students.id) < 3.5;
```

Объединение результатов

Объединение результатов из двух выборок

```
<выборка1> UNION <выборка2>
```

Объединение результатов

Объединение результатов из двух выборок

```
<выборка1> UNION <выборка2>
```

Примеры

```
SELECT fname, lname FROM students
UNION
SELECT fname, lname FROM teachers;
```

Внешнее соединение таблиц

Внешнее соединение таблиц по заданному условию

```
<таблица1> [FULL | LEFT | RIGHT] OUTER JOIN <таблица2>
ON <условие>
```

Внешнее соединение таблиц

Внешнее соединение таблиц по заданному условию

```
<таблица1> [FULL | LEFT | RIGHT] OUTER JOIN <таблица2>
ON <условие>
```

Пример

```
SELECT DISTINCT
    students.fname,
    students.lname,
    teachers.fname,
    teachers.lname
FROM students
FULL OUTER JOIN teachers
ON students.fname = teachers.fname AND
   students.lname = teachers.lname;
```

LEFT OUTER JOIN vs WHERE (1)

WHERE

```
SELECT fname, lname, number
FROM students, groups
WHERE students.group_id = groups.id;
```

LEFT OUTER JOIN vs WHERE (1)

WHERE

```
SELECT fname, lname, number
FROM students, groups
WHERE students.group_id = groups.id;
```

LEFT OUTER JOIN

```
SELECT fname, lname, number
FROM students LEFT OUTER JOIN groups
ON students.group_id = groups.id;
```

LEFT OUTER JOIN vs WHERE (2)

WHERE

```
SELECT fname, lname, specs.info
FROM students, groups, specs
WHERE (students.group_id = groups.id) and
      (groups.spec_id = specs.id);
```

LEFT OUTER JOIN vs WHERE (2)

WHERE

```
SELECT fname, lname, specs.info
FROM students, groups, specs
WHERE (students.group_id = groups.id) and
      (groups.spec_id = specs.id);
```

LEFT OUTER JOIN

```
SELECT fname, lname, specs.info
FROM groups LEFT OUTER JOIN specs
ON groups.spec_id = specs.id
LEFT OUTER JOIN students
ON students.group_id = groups.id;
```

Внутреннее соединение таблиц

Внутреннее соединение таблиц по заданному условию

```
<таблица1> INNER JOIN <таблица2> ON <условие>
```

Внутреннее соединение таблиц

Внутреннее соединение таблиц по заданному условию

```
<таблица1> INNER JOIN <таблица2> ON <условие>
```

Примеры для сравнения

```
SELECT fname, lname, number FROM students
FULL OUTER JOIN groups ON students.group_id = groups.id;

SELECT fname, lname, number FROM students
LEFT OUTER JOIN groups ON students.group_id = groups.id;

SELECT fname, lname, number FROM students
RIGHT OUTER JOIN groups ON students.group_id = groups.id;

SELECT fname, lname, number FROM students
INNER JOIN groups ON students.group_id = groups.id;
```