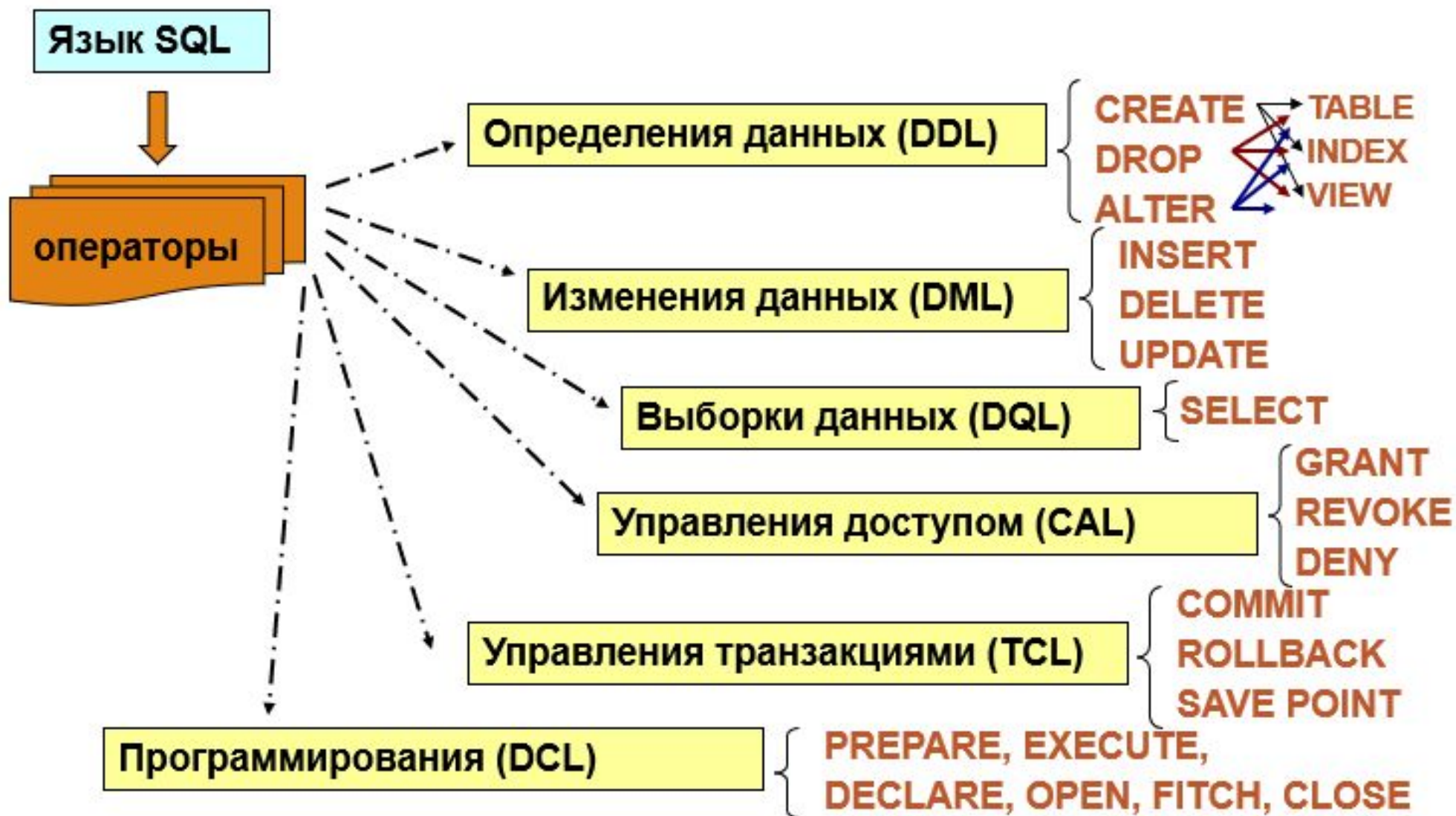


# Язык SQL

## DDL и DML

# Структура языка SQL



## Целостность данных

**Целостность сущностей** - определяет строку таблицы как уникальный экземпляр некоторой сущности.

**Первичный ключ (primary key)** - столбец или группа столбцов уникально идентифицирующий каждую запись.

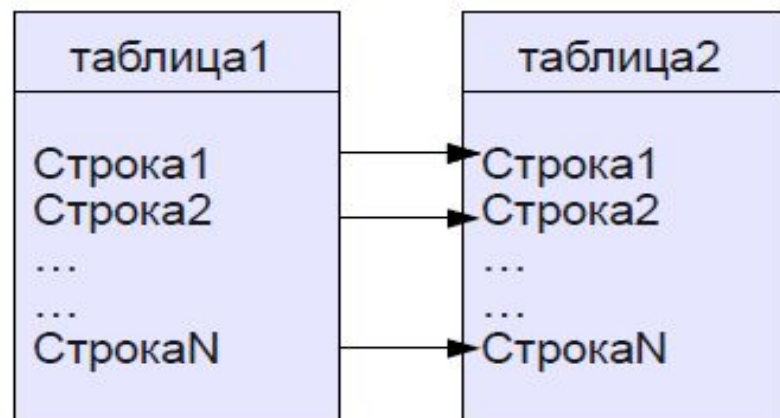
**Внешний ключ (foreign key)** – отражение связей между таблицами. Подчиненная таблица должна иметь идентичный столбец (или группу столбцов) для хранения значений, уникально идентифицирующих главные записи.

**Ссылочная целостность** – в подчиненных таблицах не должно быть записей, ссылающихся на несуществующие записи главных таблиц.

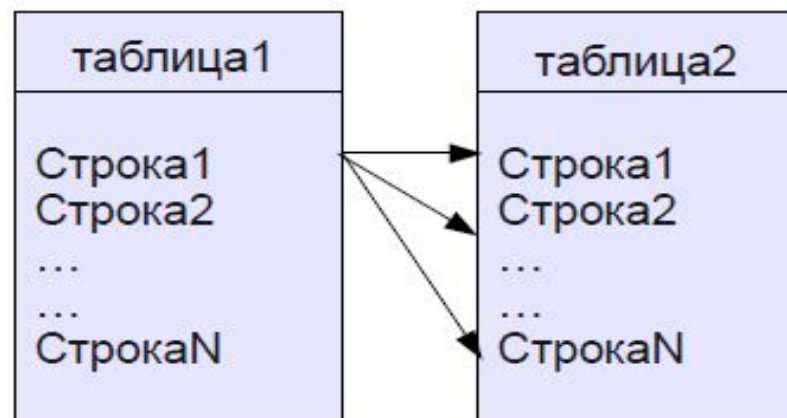
## Типы связей

Данные хранятся в разных таблицах,  
между таблицами существуют связи (relationships).

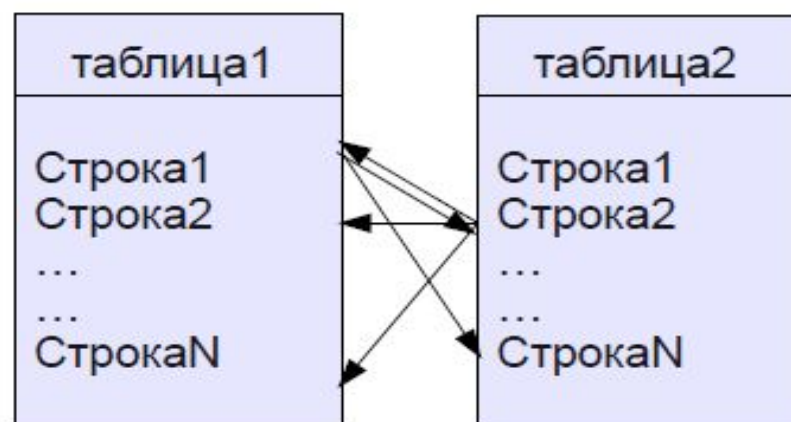
1 : 1



1 : n



m : n



5

**DDL – определение данных**

# 1. CREATE TABLE - создание таблиц

## Общий синтаксис:

```
CREATE TABLE имя_таблицы (  
поле1 Тип поля1,  
поле2 Тип поля2,  
..., полеN Тип поляN);
```

ПРИМЕР. Создание таблицы person\_info (без указания первичного ключа)

```
CREATE TABLE person_info (  
person_id INTEGER NOT NULL,  
first_name VARCHAR(15) NOT NULL,  
last_name VARCHAR(20) NOT NULL,  
gender CHAR(1),  
birthday DATE,  
salary NUMERIC(7,2));
```

## 2. ALTER TABLE - изменение таблиц

- Добавление колонок
- Удаление колонок
- Модификация колонок
- Изменения имени таблицы
- Изменения кодировки таблицы
- Добавление и удаление ограничений

**ALTER TABLE** имя\_таблицы {**ADD** <имя столбца>  
<определение столбца>}| {**MODIFY** <имя столбца> <Определение  
столбца>}| {**DROP COLUMN** <имя столбца>}

## 2.1. Определение первичного ключа таблицы

```
ALTER TABLE имя_таблицы  
ADD PRIMARY KEY (имя_столбца);
```

### ПРИМЕР.

```
ALTER TABLE person_info  
ADD PRIMARY KEY(person_id);
```

Значения первичного ключа подразумевают уникальную идентификацию записи, соответственно, значения не могут повторяться.



## 2.2. Определение внешнего ключа таблицы

```
ALTER TABLE имя_подчиненной_таблицы  
ADD {CONSTRAINT имя_ограничения}  
FOREIGN KEY (имя_поля_подчиненной_таблицы)  
REFERENCES имя_главной_таблицы (поле_главной_таблицы);  
{ON DELETE действие}  
{ON UPDATE действие}
```

**ON DELETE CASCADE** (каскадное удаление) – позволяет при удалении строки из главной таблицы автоматически удалить все связанные строки из зависимой таблицы

## 2.2. Определение внешнего ключа таблицы



### ПРИМЕР.

```
CREATE TABLE person_address (  
address_id INTEGER PRIMARY KEY,  
person_id INTEGER,  
address VARCHAR(200));
```

```
ALTER TABLE person_address  
ADD FOREIGN KEY (person_id)  
REFERENCES person_info (person_id) ON DELETE CASCADE;
```

Первичный (*PRIMARY KEY*) и внешний (*FOREIGN KEY*) ключи можно добавлять при создании таблицы:

**AUTO\_INCREMENT** — при добавлении новых записей значение этого поля устанавливается автоматически, причём значение равно на единицу больше предыдущего.

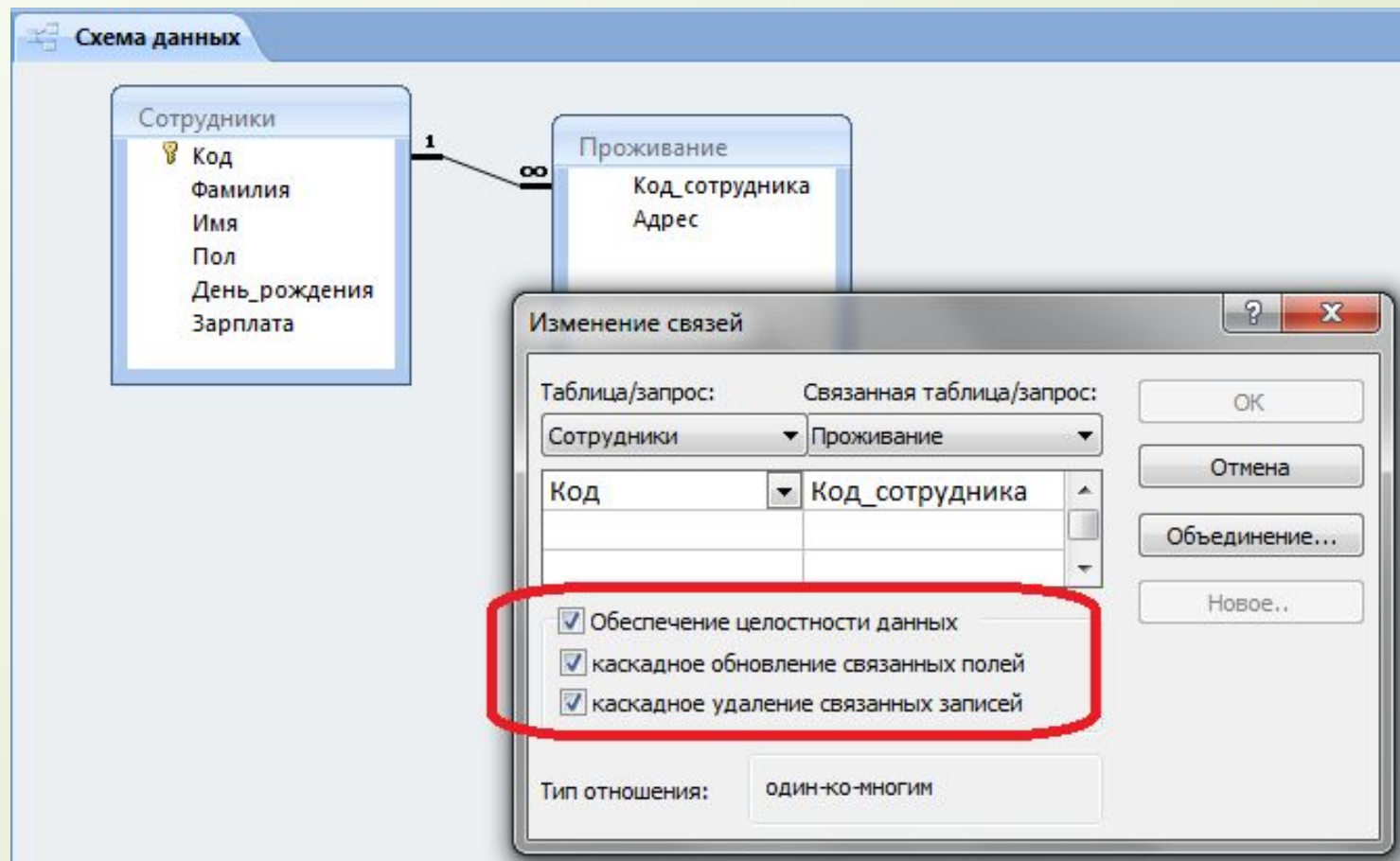
## Второй способ

- 1. CREATE TABLE person\_info (**  
person\_id **INTEGER NOT NULL PRIMARY KEY AUTO\_INCREMENT,**  
first\_name **VARCHAR(15) NOT NULL,**  
last\_name **VARCHAR(20) NOT NULL,**  
gender **CHAR(1),**  
birthday **DATE,**  
salary **NUMERIC(7,2));**
- 2. CREATE TABLE person\_address (**  
address\_id **INTEGER PRIMARY KEY AUTO\_INCREMENT,**  
person\_id **INTEGER,**  
address **VARCHAR(200),**  
**FOREIGN KEY (person\_id)**  
**REFERENCES person\_info (person\_id) ON DELETE**  
**CASCADE);**

# В СУБД MS Access:

После создания связи с помощью запроса:

- Двойным щелчком мыши по связи открывается окно *Изменение связей*.
- Необходимо установить галочки для каскадного обновления связанных полей и каскадного удаления связанных записей.



# Значение по умолчанию

## 1. При создании таблицы:

```
CREATE TABLE something  
(  
  name varchar(20),  
  rank int default 1  
);
```

Задаёт значение по умолчанию

```
CREATE TABLE something  
(  
  name varchar(20),  
  married varchar(20) default  
  'да'  
);
```

значение по умолчанию:  
1

# Значение по умолчанию

2. Если таблица уже существует:

**ALTER TABLE** something

**ALTER COLUMN** rank **SET DEFAULT** 1;



## 2.3. Добавление поля в таблицу

### ПРИМЕР.

```
ALTER TABLE person_info  
ADD o_r int;
```

Каждая новая колонка добавляется в конец таблицы.  
Если вы хотите добавить новую колонку после определенной колонки, то используйте команду **AFTER**.

Добавим колонку `shelf_position` сразу после колонки `price`.

```
ALTER TABLE books ADD shelf_position VARCHAR(20) AFTER Price;
```

### ПРИМЕР.

```
ALTER TABLE person_info  
ADD o_r2 int AFTER person_id;
```



## 2.4. Удаление поля из таблицы

### ПРИМЕР.

```
ALTER TABLE person_info  
DROP o_r2;
```

## 2.5. Перестановка полей в таблице

Чтобы переставить колонку используйте команду **AFTER**, также понадобится повторно определить тип данных.

### ПРИМЕР.

```
ALTER TABLE person_info  
MODIFY COLUMN birthday DATE AFTER person_id;
```

### ***3. DROP TABLE - удаление таблиц***

**DROP TABLE** имя\_таблицы {**CASCADE CONSTRAINTS**};

- Предложение **CASCADE CONSTRAINTS** позволяет удалять все ссылочные ограничения целостности, которые ссылаются на первичные и уникальные ключи, определенные в удаляемых столбцах.
- Предложение **CASCADE CONSTRAINTS** удаляет также все ограничения, состоящие из нескольких столбцов и определенные в удаляемых столбцах.

### ***4. TRUNCATE TABLE - очистка таблиц***

**TRUNCATE TABLE** имя\_таблицы;

# DML –изменение данных

## 1. *INSERT* – добавление записей в таблицу

**INSERT INTO** имя\_таблицы **VALUES** (значение поля1, значение поля2,..., значение поляN);

**INSERT INTO** имя\_таблицы (поле1, поле3,...) **VALUES** (значение поля1, значение поля2,..., значение поляN);

**INSERT INTO** person\_info  
**VALUES** (1, 'John', 'Smith', 'M', '1973.10.15', 45568.56);

**INSERT INTO** person\_info (person\_id, first\_name, last\_name)  
**VALUES** (5, 'Sarah', 'Connor');

Успешно.

**INSERT INTO** person\_info **VALUES** (NULL, 'Jane', 'Smith', 'F', '8-AUG-1987', NULL);

**Ошибка**, т.к person\_id не может быть NULL.

## *Добавление в таблицу группы записей из другой таблицы*

```
INSERT INTO имя_таблицы
```

```
SELECT ...;
```

```
CREATE TABLE t2 (  
first_1 VARCHAR(15),  
last_1 VARCHAR(20),  
birthday_1 DATE);
```

```
INSERT INTO t2
```

```
SELECT first_name, last_name, birthday
```

```
FROM person_info;
```

```
INSERT INTO person_info VALUES (2, 'Sara', 'Doe', 'F', '1986.10.9',  
29789.56);
```

Успешно.

```
INSERT INTO person_info VALUES (2, 'Rita', 'Blow', 'F', '1975.10.09',  
29789.56);
```

Ошибка

```
INSERT INTO person_info VALUES (3, 'Sara', 'Doe', 'F', '1986.10.9',  
29789.56)
```

```
INSERT INTO person_address VALUES (1, 'Moscow, Arbat street,  
67-14');
```

```
INSERT INTO person_address VALUES (2, 'Moscow, Arbat street,  
67-14');
```

Успешно.

```
INSERT INTO person_address VALUES (4, 'Zelenograd, Green street,  
23');
```

**Ошибка.** Попытка вставить подчиненную запись при отсутствии соответствующей главной записи.

```
INSERT INTO person_address VALUES (3, 'Zelenograd, Green street,  
23');
```

# Заполнение всей таблицы



```
CREATE TABLE `temp`(id int, `price` decimal(18,2));
```

```
INSERT INTO `temp`(`id`, `price`)
```

```
VALUES (1, 176.00), (2, 337.00), (3, 234.00), (4, 180.00),  
       (5, 135.00), (6, 72.00), (7, 72.00), (8, 81.00), (9, 135.00),  
       (10, 113.00), (11, 162.00);
```



## 2. UPDATE - изменение значений полей таблицы

А) Изменение всех значений поля таблицы

```
UPDATE <table_name>  
SET <column_name> = <value>
```

### ПРИМЕР.

```
UPDATE person_address  
SET address = 'Volgograd, First street, 15-20'
```

## 2. UPDATE - изменение значений столбцов таблицы

Б) Изменение конкретных значений полей таблицы

```
UPDATE <table_name>  
SET <column_name> = <value>  
WHERE <column_name> = <value>
```

### ПРИМЕР.

```
UPDATE person_address  
SET address = 'Volgograd, First street, 15-20'  
WHERE person_id = 3;
```

```
UPDATE <table_name>  
SET <column_name> = <value>  
WHERE <column_name> = <column_name> [оператор]
```

### ПРИМЕР.

```
UPDATE person_info SET salary = salary * 2  
WHERE person_id = 3;
```

### 3. *DELETE* - удаление записей таблицы

А) Удаление всех записей таблицы

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

Б) Удаление записей таблицы, удовлетворяющих условию

**ПРИМЕР.** Удаление записи о сотруднике с номером 3.

```
DELETE FROM person_info
```

```
WHERE person_id = 3;
```