

Правила целостности

Тип правила целостности	Описание
NULL/NOT NULL	Разрешение/запрет использования неопределенных значений
UNIQUE	Контроль уникальности значений атрибутов
PRIMARY KEY	Первичный ключ
FOREIGN KEY	Внешний (или ссылочный) ключ
CHECK	Контроль допустимых значений атрибутов

Операторы для создания и удаления правил целостности

```
CREATE TABLE <имя_таблицы> ...
```

```
ALTER TABLE <имя_таблицы> ...
```

```
DROP TABLE <имя_таблицы>
```

Добавление и удаление правил целостности в контексте оператора CREATE TABLE

```
CREATE TABLE <имя_таблицы>  
    ({ {имя столбца}  
      {тип данных}  
      [значение по умолчанию]  
      [список правил целостности]  
    }+)
```

Добавление и удаление правил целостности в контексте оператора **ALTER TABLE**

```
ALTER TABLE <имя_таблицы>  
{ADD|DROP}  
<список правил целостности>
```

Правило целостности NULL/NOT NULL

- Представляет собой частный случай правила целостности CHECK.
- NULL позволяет не задавать значение атрибута.
- NOT NULL проверяет, чтобы значение атрибута задано (не позволяет хранить неопределенные значения).

Пример: NULL/NOT NULL

```
CREATE TABLE Person  
  (PersonId INTEGER,  
   DepartmentId INTEGER NOT NULL,  
   Job VARCHAR(30)) ;
```

```
DESCRIBE Person;
```

Правило целостности Primary Key

- Используется для однозначной идентификации строк таблицы.
- В таблице может быть определен только один Primary Key.
- Может состоять из нескольких полей.
- Столбцы, на основе которых строится Primary Key, не могут содержать неопределенные значения.

Пример: PRIMARY KEY (одно поле)

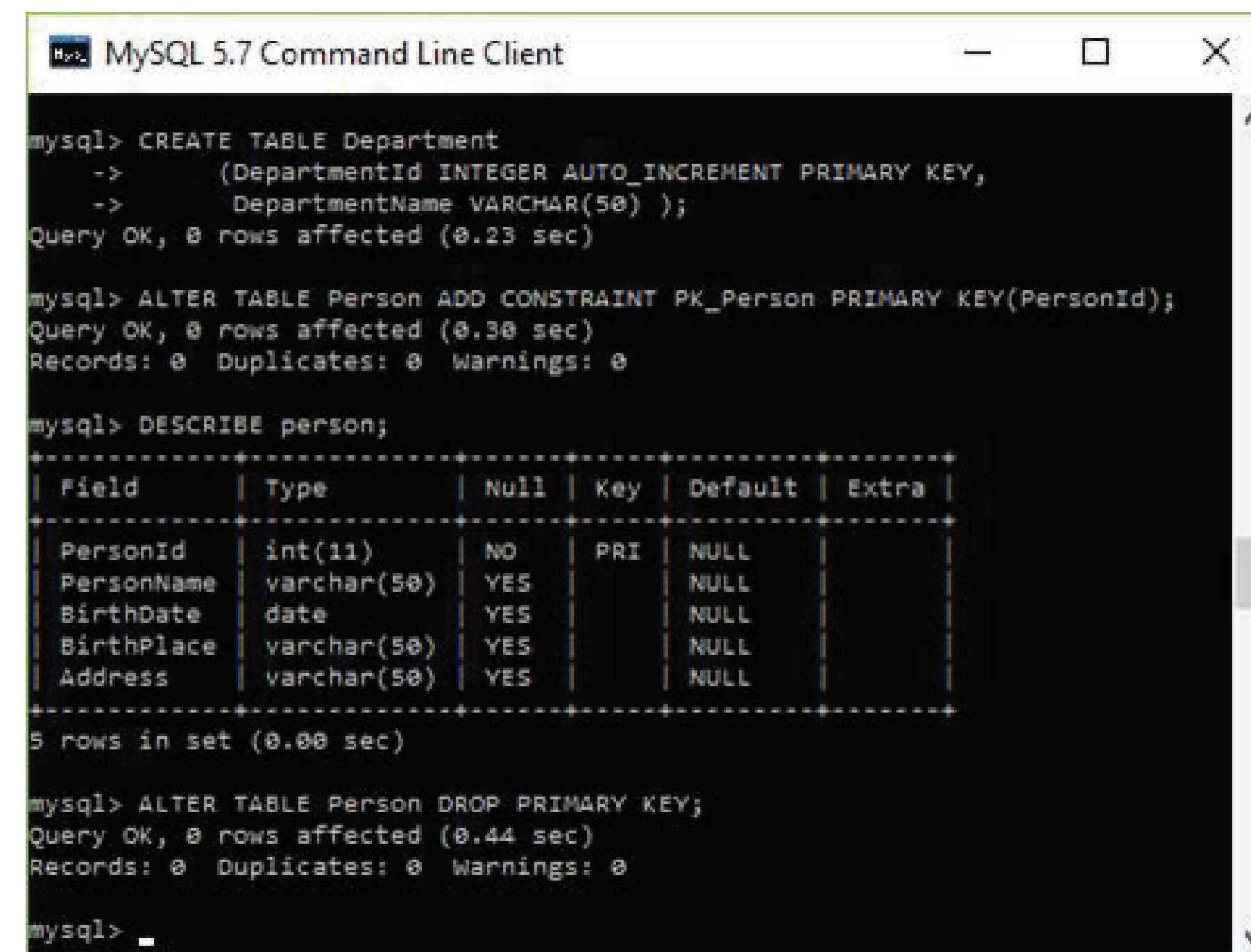
```
CREATE TABLE Department (  
    DepartmentId INTEGER PRIMARY KEY,  
    DepartmentName VARCHAR(50) );
```


Первичный ключ с автоинкрементацией

- В СУБД MySQL для генерации значений ключевых атрибутов (и только для них!) может использоваться опция `AUTO_INCREMENT`.
- По умолчанию идентификация начинается с 1 и шагом 1, однако стартовое значение может быть задано и явным образом (например, `AUTO_INCREMENT=100`).

Пример

```
CREATE TABLE Department (  
    DepartmentId INTEGER  
    AUTO_INCREMENT PRIMARY KEY,  
    DepartmentName VARCHAR(50)  
);  
ALTER TABLE Person ADD  
    CONSTRAINT PK_Person  
    PRIMARY KEY(PersonId);  
DESCRIBE Person;  
ALTER TABLE Person  
    DROP PRIMARY KEY
```



```
mysql> CREATE TABLE Department  
-> (DepartmentId INTEGER AUTO_INCREMENT PRIMARY KEY,  
-> DepartmentName VARCHAR(50) );  
Query OK, 0 rows affected (0.23 sec)  
  
mysql> ALTER TABLE Person ADD CONSTRAINT PK_Person PRIMARY KEY(PersonId);  
Query OK, 0 rows affected (0.30 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> DESCRIBE person;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| PersonId | int(11) | NO | PRI | NULL | |  
| PersonName | varchar(50) | YES | | NULL | |  
| BirthDate | date | YES | | NULL | |  
| BirthPlace | varchar(50) | YES | | NULL | |  
| Address | varchar(50) | YES | | NULL | |  
+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql> ALTER TABLE Person DROP PRIMARY KEY;  
Query OK, 0 rows affected (0.44 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> _
```

Пример: PRIMARY KEY (несколько полей)

```
CREATE TABLE Person  
  (PersonId INTEGER,  
   DepartmentId INTEGER,  
   Name VARCHAR(30) ,  
   PRIMARY KEY (DepartmentId,  
                PersonId) ) ;
```

Правило целостности **UNIQUE**

- Обеспечивает в таблице уникальность значений, заданных в правиле **UNIQUE** атрибутов.
- Допускает неопределенные значения (NULL).

Пример: UNIQUE (при создании таблицы)

```
CREATE TABLE Person  
(PersonId INTEGER UNIQUE NOT NULL,  
  DepartmentId INTEGER,  
  Name VARCHAR(30),  
  UNIQUE (DepartmentId, Name)  
);
```

Пример: добавление ограничения **UNIQUE** к существующей таблице

```
DROP TABLE Person;
```

```
CREATE TABLE Person
```

```
(PersonId INTEGER PRIMARY KEY,  
  DepartmentId INTEGER NOT NULL,  
  Job VARCHAR(30) CHECK (Job IN  
    ('программист', 'аналитик', 'менеджер')) ,  
  Salary REAL,  
  Phone VARCHAR(11));
```

```
ALTER TABLE Person ADD CONSTRAINT  
U_Phone_Person UNIQUE(PersonId, Phone);
```

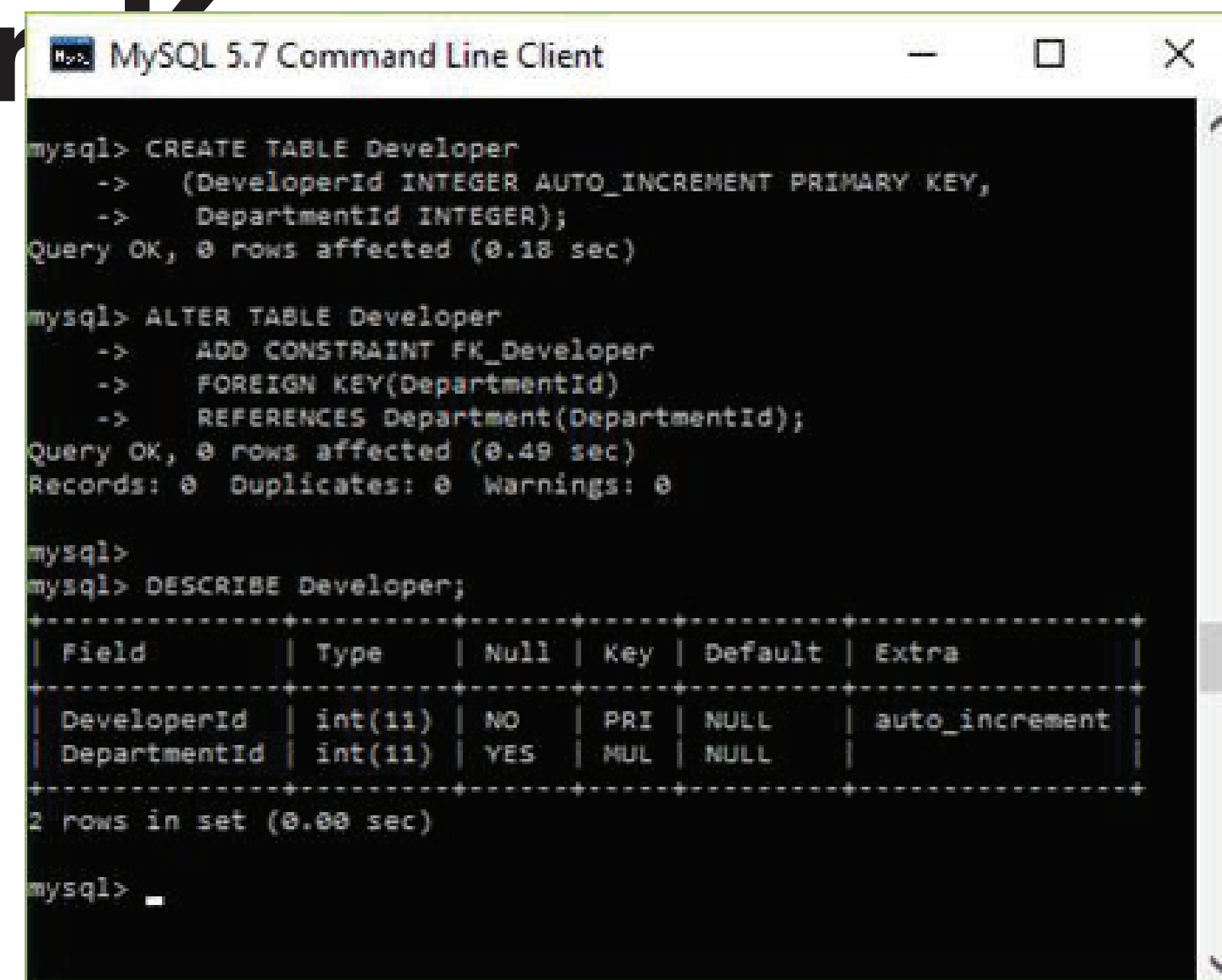
```
DESCRIBE Person;
```

Правило целостности Foreign Key

- Связывает по заданным атрибутам родительскую и подчиненную таблицы.
- Гарантирует, что подчиненная таблица не будет использовать значения, не упомянутые в родительской таблице.
- По атрибутам родительской таблицы должно быть построено правило целостности Primary Key.
- Атрибуты должны быть строго одного типа.

Пример: создание правила целостности Foreign Key

```
CREATE TABLE Developer  
  
    (DeveloperId INTEGER  
    PRIMARY KEY,  
    DepartmentId INTEGER) ;  
  
ALTER TABLE Developer  
    ADD CONSTRAINT FK_Developer  
    FOREIGN KEY (DepartmentId)  
  
    REFERENCES  
    Department (DepartmentId) ;  
  
DESCRIBE Developer;
```



```
mysql> CREATE TABLE Developer  
-> (DeveloperId INTEGER AUTO_INCREMENT PRIMARY KEY,  
-> DepartmentId INTEGER);  
Query OK, 0 rows affected (0.18 sec)  
  
mysql> ALTER TABLE Developer  
-> ADD CONSTRAINT FK_Developer  
-> FOREIGN KEY (DepartmentId)  
-> REFERENCES Department (DepartmentId);  
Query OK, 0 rows affected (0.49 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql>  
mysql> DESCRIBE Developer;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type   | Null | Key | Default | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| DeveloperId | int(11) | NO   | PRI | NULL    | auto_increment |  
| DepartmentId | int(11) | YES  | MUL | NULL    |                |  
+-----+-----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql> _
```


Пример: внешний ключ на столбец той же таблицы

```
CREATE TABLE Developer
  (DeveloperId INTEGER PRIMARY KEY,
   DepartmentId INTEGER,
   Boss INTEGER,
  FOREIGN KEY (Boss)
   REFERENCES Developer (DeveloperId) ;

DESCRIBE Developer;
```

Каскадные правила целостности

Foreign Key

Задание стратегии при изменении или удалении родительской записи

```
[CONSTRAINT [symbol]] FOREIGN KEY  
    [index_name] (index_col_name, ...)  
REFERENCES tbl_name (index_col_name, ...)  
[ON DELETE reference_option]  
[ON UPDATE reference_option]
```

reference_option:

RESTRICT | CASCADE | SET NULL | NO ACTION

Правило целостности CHECK

- Проверяет значение атрибута добавляемой или изменяемой записи с помощью заданного в правиле CHECK логического выражения или конструкции принадлежности множеству.
- Логическое выражение должно выдавать значение TRUE (только в этом случае запись принимается в таблицу).
- Принадлежность множеству (IN) — значение атрибута проверяется на вхождение в заданный список.

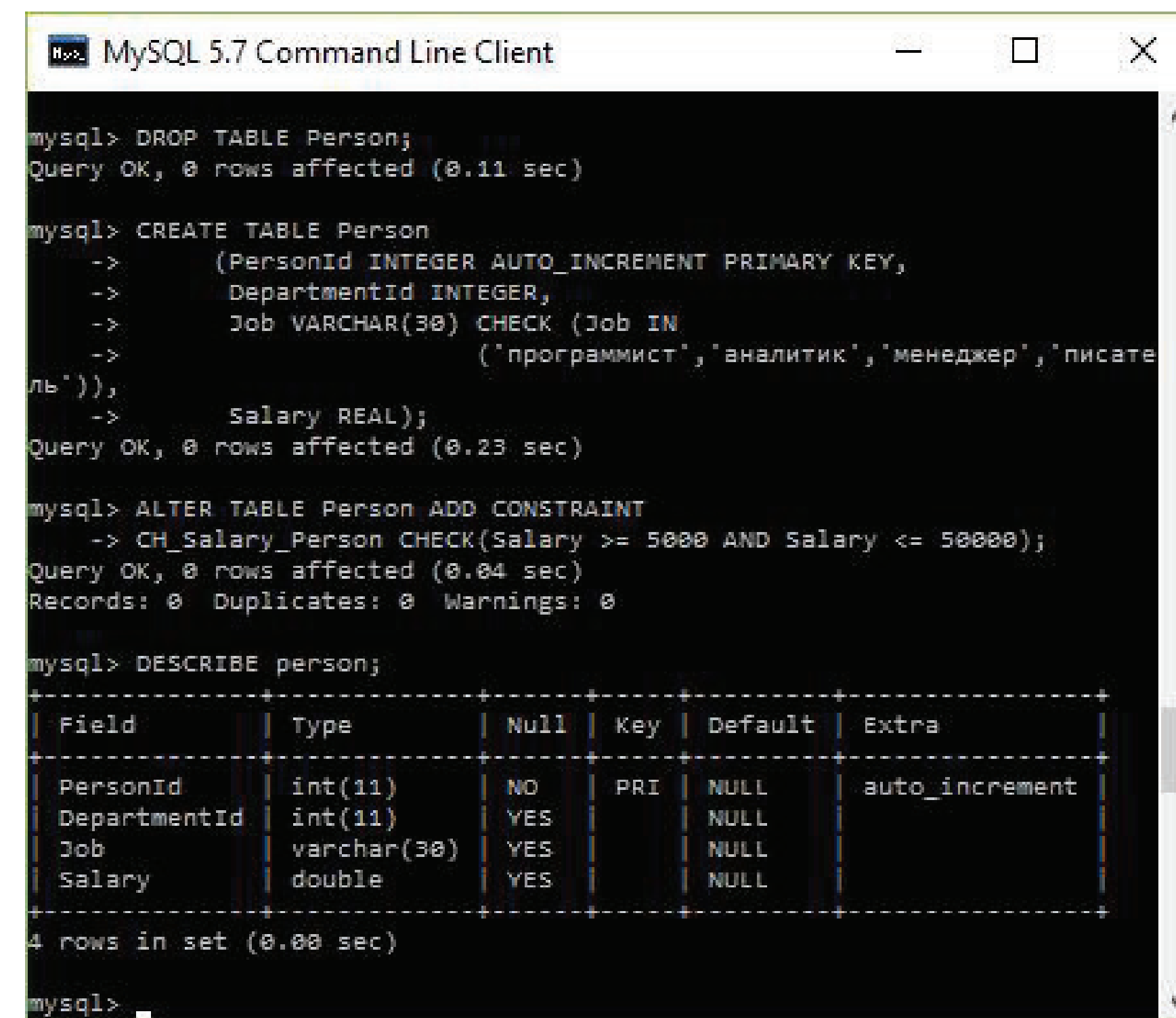
Пример: создание правила целостности CHECK

```
DROP TABLE Person;

CREATE TABLE Person
(PersonId INTEGER AUTO_INCREMENT
PRIMARY KEY,
DepartmentId INTEGER,
Job VARCHAR(30) CHECK (Job IN
('программист', 'аналитик', 'менеджер')),
Salary REAL);

ALTER TABLE Person
ADD CONSTRAINT CH_Salary_Person
CHECK(Salary >= 5000 AND Salary <= 50000);

DESCRIBE person;
```



```
mysql> DROP TABLE Person;
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE Person
-> (PersonId INTEGER AUTO_INCREMENT PRIMARY KEY,
-> DepartmentId INTEGER,
-> Job VARCHAR(30) CHECK (Job IN
-> ('программист', 'аналитик', 'менеджер', 'писатель')),
-> Salary REAL);
Query OK, 0 rows affected (0.23 sec)

mysql> ALTER TABLE Person ADD CONSTRAINT
-> CH_Salary_Person CHECK(Salary >= 5000 AND Salary <= 50000);
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DESCRIBE person;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| PersonId | int(11) | NO | PRI | NULL | auto_increment |
| DepartmentId | int(11) | YES | | NULL | |
| Job | varchar(30) | YES | | NULL | |
| Salary | double | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```