



Information Technology





Первичный ключ.
**INSERT, UPDATE,
DELETE**

Первичный ключ (primary key) представляет собой один из примеров уникальных индексов и применяется для уникальной идентификации записей таблицы. Никакие из двух записей таблицы не могут иметь одинаковых значений первичного ключа. Первичный ключ обычно сокращенно обозначают как РК (primary key).

В реляционных базах данных практически всегда разные таблицы логически связаны друг с другом. Первичные ключи как раз используются для однозначной организации такой связи.



Первичный ключ



Таблица Themes

PK

id_theme	Записи
1	
2	
3	
4	

Таблица Posts

PK

FK

id_post	Записи	id_theme
1		1
2		1
3		1
4		2



Как видно на вышеуказанном рисунке первичным ключом таблицы `themes` является `id_theme`, а таблицы `posts` - `id_post`. Обратите внимание, что поле `id_theme` присутствует и в таблице `posts`. Каждое значение этого поля в таблице `posts` является **внешним ключом** (в данном случае это внешний ключ для первичного ключа таблицы `themes`). Внешний ключ сокращенно обозначают как FK (`foreign key`). Внешний ключ ссылается на первичный ключ таблицы `themes`, устанавливая однозначную логическую связь между записями таблиц `themes` и `posts`



- Отношения один к одному
- Один ко многим и многие к одному
- Многие ко многим
- Связь с самим собой



Вставка данных в таблицу



Все строки в SQL вводятся с использованием команды **INSERT**. В самой простой форме, **INSERT** использует следующий синтаксис:

```
INSERT INTO tbl_name (col_name,...) VALUES (data,...)
```

ИЛИ

```
INSERT INTO tbl_name SET col_name=data
```

где,

tbl_name-имя таблицы, в которую будет вставлена новая строка

col_name-название полей

data – значения, соответствующие полям



Вставка данных в таблицу



Например:

Следующая команда вставит в таблицу *users* новую запись, присвоив полям *name*, *age*, *country*, *city* значения *Evgen*, *26*, *Russia*, *Ryazan* соответственно:

```
1 INSERT INTO
2   `users` (`name`, `age`, `country`, `city`)
3 VALUES
4   ('Evgen', 26, 'Russia', 'Ryazan')
```

Готово



Вставка данных в таблицу



Если для поля или группы полей, присутствующих в таблице, не установить значение, то используется значение, установленное по умолчанию при создании таблицы.



Вставка данных в таблицу



- С помощью одного запроса можно вставить несколько записей в таблицу, например:

```
01 INSERT INTO
02   `users` (
03     `name`, `age`
04   )
05 VALUES
06   ('Миша', 25),
07   ('Ксюша', 15),
08   ('Настя', 12),
09   ('Саша', 26),
10   ('Дима', 30)
```

Таблица users имеет следующие столбцы:



Редактирования записи в таблице



Например, следующий пример производит обновление поля *country* у ВСЕХ записей в таблице *users*:

```
1 UPDATE
2   `users`
3 SET
4   `country`='Russia'
```

На следующем изображении обновление полей *country* и *city* у ВСЕХ записей таблицы *users*:

```
1 UPDATE
2   `users`
3 SET
4   `country`='Russia',
5   `city`='Ryazan'
```



Редактирования записи в таблице



Оператор `LIMIT` задает максимальное число записей для редактирования.

Например, следующий запрос в таблице `users` обновит только 2 записи, у которых поле `area_id = 2`

```
1 UPDATE `users`  
2     SET  
3     `city` = 'Краматорск'  
4     WHERE  
5     `area_id` = 2  
6     LIMIT 3  
7
```



Удаление записи в таблице



Удаление записей осуществляется командой **DELETE FROM**.

Синтаксис оператора DELETE FROM

DELETE FROM table_name [WHERE where_definition]

где,

tbl_name-имя таблицы, в которую будет вставлена
where_definition-условие, по которому будет
удаляться запись



Удаление записи в таблице



Команда **DELETE** удаляет из таблицы **table_name** все записи, удовлетворяющие условию **where_definition**. Если условие **WHERE where_definition** не задано, то из таблицы **table_name** удаляются все записи.

Команда **DELETE** возвращает количество удаленных записей.

Если в команде **DELETE** не задано условие **WHERE where_definition**, то команда возвратит 0, хотя записи были удалены.



Удаление записи в таблице



Нижеуказанная запись удалит все данные из
таблицы *users*

```
1 DELETE FROM  
2 `users`
```

Следующая запись удалит только одну
запись из таблицы *users*, у которой *id=1*

```
1 DELETE FROM  
2 `users`  
3 WHERE  
4 `id` = 1
```



Удаление записи в таблице



С помощью оператора LIMIT можно задать максимальное число записей для удаления, например следующая запись удалит только 5 записей из таблицы users

```
1 DELETE FROM
2   `users`
3 LIMIT 5
```

Оператор LIMIT прописывается в конце запроса. Следующий запрос удалит 3 записи из таблицы users, у которых поле area_id= 2

```
1 DELETE FROM
2   `users`
3 WHERE `area_id` = 2
4 LIMIT 3
```


Удаление записи в таблице



С помощью ORDER BY можно отсортировать записи по определенному полю по возрастанию или убыванию и удалить первые, либо последние элементы, например:

```
1 DELETE FROM
2   `users`
3 ORDER BY `id` DESC
4 LIMIT 5
5 |
```



Поиск записей осуществляется командой **SELECT**, простой синтаксис выглядит следующим образом:

*SELECT * FROM table_name WHERE
(выражение) [order by field_name [desc][asc]]*

```
1 SELECT *  
2     FROM table_name  
3     WHERE (выражение)  
4     [ORDER BY field_name [desc][asc]]  
5     [LIMIT number]
```



где,

tbl_name-имя таблицы, из которой идёт
выборка

where_definition-условие, по которому будут
выбираться запись

