**Азбука SQLite**

**Основные команды**

**CREATE TABLE**

Новая таблица создаётся с помощью команды **CREATE TABLE** после которой указывается имя таблицы, а затем в круглых скобках указываются имена столбцов с параметрами.

Создать новую таблицу **tbl\_info** с четырьмя столбцами:

CREATE TABLE tbl\_info (

\_id integer PRIMARY KEY AUTOINCREMENT,

name text NOT NULL,

age integer NOT NULL,

city text NOT NULL

);

Ключевое слово **NOT NULL** требует обязательно присваивать значение столбцу при команде **INSERT INTO**.

Существует также ключевое слово **DEFAULT**, позволяющее вставить значение по умолчанию, например, **city text NOT NULL DEFAULT "Москва"**.

**DROP TABLE**

Удаление таблицы происходит с помощью команды **DROP TABLE**, затем указывается имя таблицы. При желании можно поставить проверочное условие **IF EXISTS**.

Удалить таблицу **tbl\_info**:

DROP TABLE IF EXISTS tbl\_info;

**INSERT INTO VALUES**

Для вставки новой записи в таблицу используется команда **INSERT INTO**, затем указывается имя таблицы, а в скобках имена столбцов. После них идёт ключевое слово **VALUES**, после которого в скобках идут вставляемые значения. Важно соблюдать количество столбцов с вставляемыми данными и их очерёдность при перечислении.

Вставить запись в таблицу **tbl\_info**:

INSERT INTO tbl\_info(name, age, city) VALUES ("Кот Васька", 29, "Москва, Россия");

Существует укороченная запись без перечисления столбцов. В этом случае нужно указывать все значения и в том порядке, в котором создавались соответствующие столбцы.

INSERT INTO tbl\_info VALUES (3, "Кот Мурзик", 10, "Рязань, Россия");

Также можно указывать только нужные столбцы для вставки данных. В первом примере мы использовали данный способ, указав только три столбца, хотя таблица содержит четыре столбца.

**SELECT \* FROM**

Для чтения данных используется команда **SELECT** с условием, затем ключевое слово **FROM** с указанием имени таблицы. Чтобы просмотреть все записи, для условия **SELECT**используется звёздочка (\*).

SELECT \* FROM tbl\_info;

**UPDATE**

Обновить запись в таблице **tbl\_info**:

UPDATE tbl\_info SET age=12 WHERE \_id=1;

После **UPDATE** указываете таблицу, после **SET** - в каком столбце нужно внести изменения и указывается новое значение, а затем указывается условие.

Можно обновлять группу столбцов, указывая их через запятую.

UPDATE ваша\_таблица SET первый\_столбец="новое значение", второй\_столбец="новое значение";

Команда **UPDATE** заменяет собой пару команд **INSERT/DELETE**. Обновить данные в нужном столбце:

UPDATE table\_name SET имя\_столбца = новое\_значение WHERE имя\_столбца = старое\_значение

Также можно производить математические действия: прибавлять, отнимать, умножать, делить. Увеличим возраст кота на день рождения.

UPDATE tbl\_info SET age=age+1 WHERE name="Мурзик";

**DELETE**

Удалить запись из таблицы **tbl\_info**:

DELETE FROM tbl\_info WHERE \_id=1;

Вам не надо перечислять все столбцы, достаточно указать в условии нужный столбец. Условие **WHERE** работает аналогично как в команде **SELECT** и позволяет использовать ключевые слова **LIKE**, **BETWEEN** и т.д.

Удалить все записи:

DELETE \* FROM tbl\_info;

**SELECT**

Рассмотрим подробнее команду **SELECT**.

Чтобы не искать все записи, можно ограничить поиск условием **WHERE**, после которого идёт имя столбца и условие равенства. Показать всех котов, чей возраст меньше 15.

SELECT \* FROM tbl\_info WHERE age < 15;

Вместо звёздочки можно указать столбцы, которые вам нужны. Например, нам нужны только имена котов с этим же условием.

SELECT name FROM tbl\_info WHERE age < 15;

Столбцы указываются через запятую. Нам нужны имена и адреса котов с этим же условием.

SELECT name, city FROM tbl\_info WHERE age < 15;

Условие **WHERE** можно объединять с помощью ключевого слова **AND**. Список котов младше 15 лет и проживающих в Москве.

SELECT name, city FROM tbl\_info WHERE age < 15 AND city="Москва";

Также доступны слова **OR** (ИЛИ):

SELECT name, city FROM tbl\_info WHERE age < 15 OR city="Москва";

Слов **AND** и **OR** может быть несколько в одном запросе.

Проверку на NULL можно сделать с помощью ключевого слова **IS NULL** (если столбец таблицы создавался без **NOT NULL**)

SELECT \* FROM tbl\_info WHERE age IS NULL;

Ключевое слово **LIKE** позволяет сократить множество операторов **OR**. Например, мы хотим узнать имена котов, которые заканчиваются на "ик":

SELECT name FROM tbl\_info WHERE name LIKE '%ик'

Символ **%** в строке указывает на любое слово с нужным окончанием (представляет любое количество неизвестных символов).

Также можно использовать спецсимвол **\_** для одного символа. Ищем Ваську.

select name from tbl\_info where name like '\_аська'

С помощью ключевого слова **BETWEEN** можно быстро и удобно задать диапазон.

SELECT name FROM tbl\_info WHERE age BETWEEN 10 and 20;

Сравните с более длинной записью

SELECT name FROM tbl\_info WHERE age >= 10 and age <=20;

С помощью условия **IN** за которыми в скобках идут нужные значения, можно задать нужные параметры.

SELECT name FROM tbl\_info WHERE age IN (10, 29);

Это короче, чем

SELECT name FROM tbl\_info WHERE age = 10 OR age = 29;

Ключевое слово **NOT IN** выполняет обратную задачу и позволяет получить записи, которые не входят в данное условие.

SELECT name FROM tbl\_info WHERE age NOT IN (10);

Ключевое слово **NOT** можно использовать не только с **IN**, но и с **BETWEEN**, **LIKE**.

Узнать число записей можно через функцию **COUNT**. Если запись содержит **NULL**, то она не учитывается.

SELECT COUNT(name) FROM table\_info;

Для показа минимального или максимального значения используются функции **MIN** или **MAX**:

SELECT name, MAX(age) FROM table\_info;

Если нам нужно вывести только определённое количество записей, то используйте ключевое слово **LIMIT** с указанием значения.

SELECT \* FROM table\_info LIMIT 3;

Существует расширенная версия, когда можно указать два значения через запятую. В первой указывается номер записи (отсчёт от 0), а вторая - число записей. Например, показать вторую запись из таблицы.

SELECT \* FROM table\_info LIMIT 1,1;

**Первичный ключ**

Первичный ключ - столбец таблицы, имеющий уникальное значение для каждой записи. Назначается при создании таблицы. Ключ не может содержать NULL, потому что теряется уникальность, ведь в других записях тоже может оказаться NULL. Значения первичного ключа должны оставаться неизменными.

Во многих случаях для этой цели создают новый столбец, который будет содержать уникальный номер. В Android столбец называют **\_id**.

Для указания первичного ключа используется ключевое слово **PRIMARY KEY**.

Чтобы база данных сама заботилась об уникальности первичного ключа, можно добавить к нему ключевое слово **AUTOINCREMENT**, которое будет автоматически увеличивать значение на единицу при вставке новой записи.

**ALTER**

Добавить новый столбец в таблицу можно с помощью необязательного ключевого слова **ALTER**, за которым идёт название столбца в таблице.

ALTER TABLE tbl\_info ADD COLUMN weight INTEGER;

Чтобы указать, после какого столбца нужно добавить новый столбец, используйте ключевое слово **AFTER**.

Другие ключевые слова: **FIRST**, **BEFORE**, **LAST**, **SECOND**, **THIRD**.

Кроме **ADD**, также можно изменить имя и тип данных столбцов (**CHANGE**), изменить тип данных или позиции столбцов (), удалить столбец из таблицы (). Не все эти команды поддерживаются в SQLite, хотя часто используются в обычных SQL.

Переименовать саму таблицу (**RENAME TO**).

ALTER TABLE tbl\_info RENAME TO table\_info;

**UPPER**

Преобразовать текст из указанного столбца в верхний регистр.

UPDATE table\_info SET name = UPPER(name);

На следующих примерах команд SQL вы быстро научитесь ориентироваться в базе данных SQLite:

// установить заголовки столбцов, которые будут отображаться в программе

sqlite> .headers on

// выделить все строки таблицы

select \* from table1;

// вычислить количество строк в таблице

select count(\*) from table1;

// выбрать определенный набор столбцов

select coll, соl2 from table1;

// выделить различные значения в столбце

select distinct col1 from table1;

// вычислить различные значения

select count(col1) from (select distinct col1 from table1);

// сгруппировать по

select count(\*), col1 from table1 group by col1;

// регулярное внутреннее объединение

select \* from table1 tl, table2 t2

where t1.col1 = t2.col1;

// оставшееся внешнее соединение

// дать всю информацию в t1, даже если в t2 нет строк

select \* from table t1 left outer join table t2

on t1.col1 = t2.col1

where ....