Лабораторная работа № 6

Знакомство с утилитой Sqlite

Цель работы: познакомиться с возможностями утилиты Sqlite по управлению данными в простейшей реляционной базе данных, с основными командами языка SQL, закрепить навык взаимодействия пользователя с операционной системой (*.nix) через командную строку.

Задание

1. Откройте программу Терминал, в домашнем каталоге создайте подкаталог mydb и перейдите в него.
2. Запустите утилиту sqlite3, при запуске укажите имя создаваемой базы данных.
3. Создайте при помощи команды языка SQL таблицу в соответствии с заданием (см. задание к лабораторной работе № 1).
4. Поместите в таблицу 3-5 записей произвольного содержания.
5. Создайте запросы к таблице, позволяющие просмотреть содержимое всех ее полей. Сравните с ожидаемыми результатами.
6. Измените запросы предыдущего пункта таким образом, чтобы вывод данных осуществлялся в отсортированном виде (по возрастанию, по убыванию). Поле, по которому выполняется сортировка, выбрать произвольно.
7. Сформулируйте условие отбора данных и создайте запрос на выборку, соответствующий этом условию.
8. Сформулируйте два условия отбора и создайте запрос на выборку, соответствующий обоим условиям.
9. Добавьте в таблицу еще одну запись.
10. Отредактируйте любую из записей таким образом, чтобы в одном из полей появились повторяющиеся значения.
11. Посчитайте, сколько различных значений находится в поле с повторяющимися значениями. Определите, сколько раз встречается конкретное значение в этом поле.
12. Определите при помощи запроса, сколько всего записей имеется в таблице. Удалите одну запись и посчитайте количество записей еще раз.
13. Завершите работу утилиты sqlite, затем Терминала. Найдите файл базы данных, оцените его размер. Прочитайте содержимое файла базы данных.
14. Покажите результаты преподавателю.

Справочный материал

Утилита Sqlite

Категория программы: система управления базами данных. Назначение: создание локальной реляционной базы данных и управление помещенными в базу данными. Для управления данными используется язык SQL. Существуют реализации для операционных систем: Windows, Linux, Mac OS X.

Запуск утилиты осуществляется из командной строки терминала, в качестве параметра указывается имя открываемой базы данных.

Поиск файла базы данных идет в текущем каталоге. Если файл с заданным именем отсутствует, sqlite создает новую базу данных. Если новую базу надо создать в отдельном каталоге, следует сперва создать этот каталог:

mkdir new_directory <Enter>

а затем в него перейти:

cd new_directory <Enter>

После перехода в нужный каталог запускаем утилиту sqlite:

sqlite3 mydb <Enter>

Если sqlite запустилась, на экране появится приглашение для ввода команд СУБД sqlite и SQL-запросов:

sqlite>_

Команды sqlite начинаются с точки:

.help — получение справочной помощи по командам sqlite.
.databases — сообщает имя и название файла загруженной базы данных.
.tables - сообщает названия таблиц загруженной базы данных.
.quit — осуществляет завершение работы sqlite.
.exit — также осуществляет завершение работы sqlite.

Команды языка SQL также помещаются в командную строку после приглашения, но точка перед командой не ставится.

Язык SQL

Язык SQL предназначен для управления данными в реляционных базах данных. Основные операции в базе данных- создание таблицы, вставка, изменение и удаление данных, чтение (выборка) данных. Язык SQL нечувствителен к регистру.

Типы данных SQL

INTEGER – целый тип,
FLOAT — вещественный тип,
VARCHAR — строка, в скобках задается длина строки: VARCHAR(20).

Создание таблицы базы данных

Используется оператор CREATE TABLE, в котором указываются названия столбцов таблицы, их тип и дополнительные атрибуты. Например, один из столбцов таблицы может быть задан как первичный ключ (PRIMARY KEY): любое значение из этого столбца однозначно определяет запись в базе данных. Соответственно такое поле может хранить только неповторяющиеся значения и не может быть пустым (NOT NULL). Обычно в качестве первичного ключа используются поля целочисленные или счетчики, автоматически получающие значение при создании новой записи. Такие поля часто называют идентификаторами.

```
CREATE TABLE contacts(
        number INTEGER PRIMARY KEY NOT NULL,
        name VARCHAR(20),
        phone VARCHAR(12)
);
```

Добавление записи в таблицу

Используется оператор INSERT INTO, после команды следует имя таблицы, в которую осуществляется вставка данных, затем в скобках перечисляются поля таблицы, в которые следует поместить данные, значения которых указаны после ключевого слова VALUES.

```
INSERT INTO contacts(number,name,phone) VALUES(2,'Nick','+7001');
```

Удаление записи из таблицы

Используется оператор DELETE FROM, затем указывается имя таблицы, в которой находится удаляемая запись. После ключевого слова WHERE записывается условие, которому должна удовлетворять удаляемая запись. Например, удаляем информацию о телефоне, которым пользуется Nick.

```
DELETE FROM contacts WHERE name = 'Nick';
```

Обновление записи из таблицы

Для обновления данных используется оператор UPDATE. В команде указывается имя редактируемой таблицы, название столбца и правило, по которому надо изменить его значение, и, если надо редактировать отдельные записи, условие их отбора. Если условие отбора не задано, будут изменены значения в каждой строке данного столбца. Например, изменяем номер телефона человека с именем Nick.

```
UPDATE contacts SET phone='+7010' WHERE name='Nick';
```

Запрос на выборку

Используется оператор SELECT — одна из наиболее часто используемых команд языка SQL. После названия команды перечисляются через запятую поля, являющиеся результатом запроса, ключевое слово определяет таблицу, из которой берутся данные. Если данные должны быть получены из нескольких таблиц, то они после ключевого слова FROM перечисляются через запятую, а в обращении к полю конкретной таблицы перед именем поля следует указать имя таблицы "contacts.name". Предложение WHERE определяет условие отбора.

Пример запроса на вывод содержимого таблицы contacts без служебного поля number:

SELECT name, phone FROM contacts;

Варианты условий отбора в запросах на выборку

Запрос на вывод всей таблицы:

SELECT * FROM contacts;

Запрос на вывод информации о телефонах, которыми пользуется Nick:

SELECT name, phone FROM contacts WHERE name = 'Nick';

Запрос на вывод информации о телефонах, которыми пользуются Nick или Ann:

SELECT name, phone FROM contacts WHERE name = 'Nick' or name = 'Ann';

Запрос на вывод информации о телефонах, начинающихся на '+7921' (знак % заменяет любое количество символов в строке, знак _ - только один символ; в Windows используется другая пара символов: * и ?):

SELECT name, phone FROM contacts WHERE phone LIKE '+7921%';

Запрос на вывод информации о вывод записей с номерами в диапазоне от 5 до 10:

SELECT name, phone FROM contacts WHERE number>=5 and number <= 10;

Сортировка данных при их выводе

Для указания сортировать данные при выводе результатов выборки используется ключевое слово ORDER BY, после которого следует имя поля, по которому выполняется сортировка, и направление сортировки. Направление сортировки задается ключевыми словами ASC (по возрастанию) или DESC (по убыванию). Если опустить ключевые слова ASC и DESC, будет выполнена сортировка по возрастанию.

SELECT name, phone FROM contacts ORDER BY name;

SELECT name, phone FROM contacts ORDER BY name DESC;

SELECT name, phone FROM contacts WHERE number>=5 and number <= 10 ORDER BY name;

Функции агрегирования

SQL-запрос позволяет вместо вывода данных из таблицы выполнить вычисления статистических величин, таких, как количество записей, сумма или среднее значение по некоторому полю, и т.д. Для этого используются функции агрегирования:

COUNT() - количество записей,
SUM() - сумма,
AVG() - среднее арифметическое,
MAX() - максимальное значение,
MIN() - минимальное значение.

Функции используют для вычислений данные столбца таблицы, указанного как параметр при вызове функции.

```
SELECT COUNT(name) FROM contacts WHERE name = 'Nick';
```

Группировка в запросах

SQL позволяет группировать записи по какому-либо критерию и выполнять статистические вычисления в пределах каждой группы. Группировка задается ключевым словом GROUP BY, после которого указывается поле, по значениям которого группируются строки. Таким образом, количество выходных строк будет равно количеству уникальных значений в поле, используемом для группировки.

```
SELECT COUNT(number) FROM contacts GROUP BY name;
```

Вывод уникальных строк

Иногда в запросах на выборку могут появиться одинаковые строки. Например, если из таблицы contacts выбирать только имена, то можно получить многократное повторение одного и того же имени. Для исключения вывода повторяющихся строк используется ключевое слово DISTINCT.

```
SELECT DISTINCT name FROM contacts;
```

Вопросы к защите

1. Для чего предназначен язык SQL?
2. Что такое система управления базами данных?
3. Приведите примеры СУБД.
4. Как создать в СУБД SQLite новую базу данных?
5. Как проверить, какие таблицы имеются в базе данных?
6. Какое действие выполняет команда INSERTязыка SQL?