Sqlite3 Commands

1. Shell Mode
  - Windows 의 명령 프롬프트나 리눅스의 쉘과 같이 SQLite3 엔진과 대화식으로 명령어를 전달하고 그에 대한 결과를 받을 수 있다.
  - 이 모드는 개발 과정에서 SQL 구문을 작성하거나 디버깅 과정에서 데이터베이스의 내용을 확인할 때 가장 많이 사용된다.
  1) 데이터베이스 생성
    - SQLite3는 기본적으로 데이터베이스 파일 하나에 테이블, 레코드, 인덱스, 설정 정보 등 모든 데이터를 저장한다.
    - 아래와 같이 데이터베이스를 커맨드 라인 인자로 전달하여 생성한다.
    sqlite3 [데이터베이스 파일명]

```
D:\temp>sqlite3 test.db
SQLite version 3.8.6 2014-08-15 11:46:33
Enter ".help" for usage hints.
sqlite>
```

    - 셀을 종료하려면 .exit를 입력한다.

```
sqlite> .exit
```

    ※ SQLite3 의 셀 모드에서 사용되는 명령어는 모두 마침표(.)로 시작하고 명령어 뒤에 세미콜론(;)을 붙이지 않는다.

    - SQLite3 셀을 종료하고 데이터베이스 파일인 'test.db' 파일을 찾아보면 파일이 생성되어 있지 않는 것을 발견한다. SQLite3에서는 기본적으로 실제 데이터베이스에 테이블 생성과 같이 어떤 데이터가 입력되기 전까지는 인자로 전달된 데이터베이스 파일을 생성하지 않는다. 이는 페이지 크기(Page Size)나 인코딩(Encoding)과 같은 값을 실제 데이터베이스 파일이 생성되기 전에 설정해야 하기 때문이다.
    - 인자 없이 sqlite3 명령어만 실행하면 데이터베이스의 내용이 파일에 저장되지 않고 메모리 내에서만 존재한다. 이 경우 sqlite3 셀이 실행되는 동안에만 정보가 유지되고, 셀을 종료하면 모든 내용이 사라진다.
    - 아래와 같이 .databases 명령어를 통해 test.db를 생성한다.

```
sqlite> .databases
seq  name             file
---  ---------------  ----------------
0    main             D:\temp\test.db
sqlite>
```

  2) SQL 문 수행
    - sqlite3 셀에서 직접 SQL 문을 입력해서 수행할 수 있다.
    - 만약 질의가 복잡하다면 줄 바꿈 문자를 입력해서 SQL 문을 여러 줄에 걸쳐 작성할 수 있다.

```
sqlite> CREATE TABLE EMP
   ...> (
   ...> EMPNO   INTEGER PRIMARY KEY,
   ..> ENAME   TEXT NOT NULL,
   ...> JOB     TEXT,
   ...> SAL     INTEGER
```

```
     ...> );
sqlite> CREATE INDEX IDX_EMPNO ON EMP(EMPNO);
sqlite> INSERT INTO EMP
     ...> VALUES (7369, 'SMITH', 'CLERK', 800);
sqlite> INSERT INTO EMP
     ...> VALUES (7499, 'ALLEN', 'SALESMAN', 1600);
sqlite> INSERT INTO EMP
     ...> VALUES (7521, 'WARD', 'SALESMAN', 1250);
sqlite> INSERT INTO EMP
     ...> VALUES (7566, 'JONES', 'MANAGER', 2975);
```

- SQL 문을 작성할 때 추후 유지보수를 위해 구문 중간에 주석을 입력해야 할 때가 있다. 한 줄 주석인 경우에는 '--' 이후에 주석 내용을 입력하고, 여러 줄에 걸친 주석의 경우 C언어와 동일하게 '/*' 과 '*/'사이에 주석 내용을 작성한다.

3) 테이블 조회

- 레코드를 조회하는 SELECT 문도 다른 SQL 문과 같이 셀에서 수행한다.
- 아래와 같이 각 레코드마다 한 줄씩 표시되며, '|'를 구분자로 써서 각 필드를 구분하고 있다.
- 만일 NULL 값일 경우에는 출력할 때 아무것도 표시하지 않으므로 '||' 와 같이 구분자가 연이어 화면에 나타난다.

```
sqlite> SELECT * FROM EMP;
7369|SMITH|CLERK|800
7499|ALLEN|SALESMAN|1600
7521|WARD|SALESMAN|1250
7566|JONES|MANAGER|2975
7654|MARTIN|SALESMAN|1250
7698|BLAKE||2850
7782|CLARK|MANAGER|
sqlite>
```

- 이번에는 출력되는 결과를 테이블과 같이 보기 쉽게끔 필드의 이름을 표시(.header ON) 하고, 각 필드는 공백으로 구분(.mode column)하도록 설정을 변경하겠다. NULL 값에 대해서도 공백으로 출력하는 대신 NULL 이라고 명시적으로 출력(.nullvalue NULL)하자.

```
sqlite> .header ON
sqlite> .mode column
sqlite> .nullvalue NULL
sqlite> SELECT * FROM EMP;
EMPNO       ENAME       JOB         SAL
----------  ----------  ----------  ----------
7369        SMITH       CLERK       800
7499        ALLEN       SALESMAN    1600
7521        WARD        SALESMAN    1250
7566        JONES       MANAGER     2975
7654        MARTIN      SALESMAN    1250
```

```
7698        BLAKE       NULL        2850
7782        CLARK       MANAGER     NULL
sqlite>

    - 입력한 데이터가 길어서 잘리는 경우에는 '.width' 명령어로 출력될 필드의 길이를 조정할 수 있다.
    - 이때 '.width' 명령어 다음에 출력할 필드의 갯수만큼 길이를 나열하면 된다.

sqlite> .width  6  10   15   6
sqlite> SELECT * FROM EMP;
EMPNO   ENAME       JOB              SAL
------  ----------  ---------------  ------
7369    SMITH       CLERK            800
7499    ALLEN       SALESMAN         1600
7521    WARD        SALESMAN         1250
7566    JONES       MANAGER          2975
7654    MARTIN      SALESMAN         1250
7698    BLAKE       NULL             2850
7782    CLARK       MANAGER          NULL
sqlite>

4) 데이터베이스 및 테이블 스키마 정보
  - .tables 명령어를 사용하면 한 데이터베이스가 포함하고 있는 테이블의 이름을 조회할 수 있다.

sqlite> .tables
EMP
sqlite>
  - .indices 명령어는 특정 테이블에 포함된 인덱스의 이름을 모두 출력한다. 이 때 테이블의 이름을 지정하면 그 테이블의 인덱스를 출력한다.
sqlite> .indices
IDX_EMPNO
sqlite>

  - 다른 데이터베이스의 DESC 명령어 처럼 테이블의 스키마, 인덱스와 같은 테이블의 정보를 SQL 형태로 조회하려면  아래와 같이 .schema table_name 으로 할 수 있다.
sqlite> .schema EMP
CREATE TABLE EMP
(
EMPNO   INTEGER PRIMARY KEY,
ENAME   TEXT NOT NULL,
JOB     TEXT,
SAL     INTEGER
);
CREATE INDEX IDX_EMPNO ON EMP(EMPNO);
sqlite>

```

- DDL 구문으로 생성되는 모든 정보는 sqlite_master 라는 시스템 테이블에 저장된다. 이 테이블에는 타입(type), 이름(name), 생성한 테이블 이름(tbl_name), 루트 페이지 번호(rootpage), SQL 구문(sql)이 저장되며, 다음과 같이 개발자가 생성한 테이블과 마찬가지로 SELECT 문으로 조회할 수 있다.

```
sqlite> SELECT * FROM sqlite_master;
type    name        tbl_name         rootpa  sql
------  ----------  ---------------     ------
-----------------------------------
table   EMP         EMP                2          CREATE TABLE EMP
                    (
                    EMPNO   INTEGER PRIMARY KEY,
                    ENAME   TEXT NOT NULL,
                    JOB     TEXT,
                    SAL     INTEGER
                    )
index   IDX_EMPNO   EMP              3       CREATE INDEX IDX_EMPNO ON
EMP(EMPNO)
sqlite>
```

5) 데이터 추출

- dump 명령어를 이용하면 특정 테이블의 내용을 추출할 수 있다. 이 때 특정 테이블과 관련된 스키마, 트리거, 인덱스 정보를 비롯해 테이블에 저장된 레코드까지 추출할 수 있다.
- 또한 바이너리 포맷이 아닌 SQL 구분 형태로 추출하므로 SQLite3뿐만 아니라 다른 DBMS에서 추출한 데이터를 가지고 동일한 데이터베이스를 구축할 수 있다.
- 이 때 추출되는 SQL 구문은 명시적으로 트랙잭션을 사용해서 해당 데이터베이스에 온전히 작용하게 된다
- .dump 명령어에 인자가 없으면 데이터베이스의 모든 내용을 추출한다.

```
sqlite> .dump EMP
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE EMP
(
EMPNO   INTEGER PRIMARY KEY,
ENAME   TEXT NOT NULL,
JOB     TEXT,
SAL     INTEGER
);
INSERT INTO "EMP" VALUES(7369,'SMITH','CLERK',800);
INSERT INTO "EMP" VALUES(7499,'ALLEN','SALESMAN',1600);
INSERT INTO "EMP" VALUES(7521,'WARD','SALESMAN',1250);
INSERT INTO "EMP" VALUES(7566,'JONES','MANAGER',2975);
INSERT INTO "EMP" VALUES(7654,'MARTIN','SALESMAN',1250);
INSERT INTO "EMP" VALUES(7698,'BLAKE',NULL,2850);
INSERT INTO "EMP" VALUES(7782,'CLARK','MANAGER',NULL);
CREATE INDEX IDX_EMPNO ON EMP(EMPNO);
```

```
COMMIT;
sqlite>
```

- 기본적으로 화면에 추출되는 내용을 특정 파일에 저장하려면 .output 명령어를 사용하면 된다.
- 저장할 파일의 이름을 인자로 전달하면 화면에 내용이 해당 파일에 저장된다.

```
sqlite> .output EMP.sql
sqlite> .dump EMP
sqlite>
```

- .output 명령어를 사용해서 출력 파일을 지정하고 난 이후, sqlite3 셀에서 수행해서 나온 모든 결과는 화면이 아니라 파일에 저장된다.
- 원래대로 화면에 출력하려면 다음과 같은 명령어를 사용한다.

```
sqlite>
sqlite> .output stdout
sqlite>
```

6) 파일에 저장된 SQL 구문의 실행
- .dump 명령어로 파일에 저장된 SQL 구문을 읽어서 수행해 보자.
- .read 명령어를 이용하면 파일에 저장된 SQL 구문을 수행할 수 있다.
- 수행할 파일 이름을 인자로 전달하면 된다.

```
sqlite> .exit

D:\temp>sqlite3 test2.db
SQLite version 3.8.6 2014-08-15 11:46:33
Enter ".help" for usage hints.
sqlite> .read EMP.sql

sqlite> .schema
CREATE TABLE EMP
(
EMPNO   INTEGER PRIMARY KEY,
ENAME   TEXT NOT NULL,
JOB     TEXT,
SAL     INTEGER
);
CREATE INDEX IDX_EMPNO ON EMP(EMPNO);
sqlite>
```

- .read 명령어를 수행한 후, .schema 명령어로 생성된 스키마를 확인하고, SELECT 문으로 데이터를 조회하면 정상적으로 데이터가 복원된 것을 확인할 수 있다.

```
sqlite> SELECT * FROM EMP;
7369|SMITH|CLERK|800
```

```
7499|ALLEN|SALESMAN|1600
7521|WARD|SALESMAN|1250
7566|JONES|MANAGER|2975
7654|MARTIN|SALESMAN|1250
7698|BLAKE||2850
7782|CLARK|MANAGER|
sqlite>
```

7) 출력 형식 변경

- .mode 명령어를 이용하면 HTML 형식으로 결과를 출력할 수 있다.

```
sqlite> .mode html
sqlite> SELECT * FROM EMP;
<TR><TD>7369</TD>
<TD>SMITH</TD>
<TD>CLERK</TD>
<TD>800</TD>
</TR>
<TR><TD>7499</TD>
<TD>ALLEN</TD>
<TD>SALESMAN</TD>
<TD>1600</TD>
</TR>
<TR><TD>7521</TD>
<TD>WARD</TD>
<TD>SALESMAN</TD>
<TD>1250</TD>
</TR>
<TR><TD>7566</TD>
<TD>JONES</TD>
<TD>MANAGER</TD>
<TD>2975</TD>
</TR>
```

- 엑셀에서 자주 사용되는 CSV 형식으로도 출력가능하다.

```
sqlite> .mode csv
sqlite> SELECT * FROM EMP;
7369,SMITH,CLERK,800
7499,ALLEN,SALESMAN,1600
7521,WARD,SALESMAN,1250
7566,JONES,MANAGER,2975
7654,MARTIN,SALESMAN,1250
7698,BLAKE,,2850
7782,CLARK,MANAGER,
sqlite>
```

- 또한 다음과 같이 탭(tab)으로도 각 필드를 구분할 수 있다.

```
sqlite> .mode tabs
sqlite> SELECT * FROM EMP;
7369    SMITH   CLERK   800
7499    ALLEN   SALESMAN        1600
7521    WARD    SALESMAN        1250
7566    JONES   MANAGER 2975
7654    MARTIN  SALESMAN        1250
7698    BLAKE           2850
7782    CLARK   MANAGER
sqlite>
```

8) 기타 설정

- .echo 명령어를 사용하면 SQL 문을 수행할 때 명령어를 다시 화면에 출력하는 기능을 활성화하거나 (ON), 비활성화(OFF)할 수 있다.
- 이 기능이 활성화되면 SQL 구문을 수행하기 전에 해당 구문을 다시 한번 출력한다.
- 따라서 SELECT 문의 결과를 파일에 따로 남기는 경우 SELECT 문과 함께 저장되기 때문에 나중에 참고하는데 도움이 된다.

```
sqlite> .echo ON
sqlite> SELECT * FROM EMP;
SELECT * FROM EMP;
7369    SMITH   CLERK   800
7499    ALLEN   SALESMAN        1600
7521    WARD    SALESMAN        1250
7566    JONES   MANAGER 2975
7654    MARTIN  SALESMAN        1250
7698    BLAKE           2850
7782    CLARK   MANAGER
sqlite>
```

- .separator 명령어는 SELECT 문에서 구분자를 변경할 때 사용한다.
- 명령어의 인자로 새로 사용할 구분자를 전달하면 된다.

```
sqlite> .separator %
sqlite> SELECT * FROM EMP;
7369%SMITH%CLERK%800
7499%ALLEN%SALESMAN%1600
7521%WARD%SALESMAN%1250
7566%JONES%MANAGER%2975
7654%MARTIN%SALESMAN%1250
7698%BLAKE%%2850
7782%CLARK%MANAGER%
sqlite>
```

- .prompt 명령어는 현재 출력되는 sqlite3 셸 모드의 프롬프트를 변경할 때 사용한다.
- 명령어의 인자로 변경할 프롬프트를 입력하면 현재 연결에 한해 변경할 프롬프트가 유지된다.

```
sqlite> .prompt SQLITE3>
```

```
SQLITE3>
SQLITE3>

  - .timer 명령어를 이용해 SQL 문의 수행 시간 측정을 활성화(ON)하거나 비활성화(OFF)할
  수 있다.
sqlite> .timer ON
sqlite> SELECT * FROM EMP;
7369|SMITH|CLERK|800
7499|ALLEN|SALESMAN|1600
7521|WARD|SALESMAN|1250
7566|JONES|MANAGER|2975
7654|MARTIN|SALESMAN|1250
7698|BLAKE||2850
7782|CLARK|MANAGER|
Run Time: real 0.016 user 0.000000 sys 0.000000
sqlite>

9) 현재 설정된 상태 확인
  - .show 명령어는 현재 설정된 상태값을 확인하는 데 사용된다.
sqlite> .header ON
sqlite> .mode column
sqlite> .width 8  15   20   8
sqlite> .nullvalue NULL
sqlite> .show
     echo: off
      eqp: off
  explain: off
  headers: on
     mode: column
nullvalue: "NULL"
   output: stdout
separator: "|" "\r\n"
    stats: off
    width: 8 15 20 8
sqlite>

10) 도움말 보기
  - .help 명령어를 이용하면 sqlite3에서 수행할 수 있는 명령어와 해당 명령어의 옵션을 확인
  할 수 있다.
sqlite> .help
```