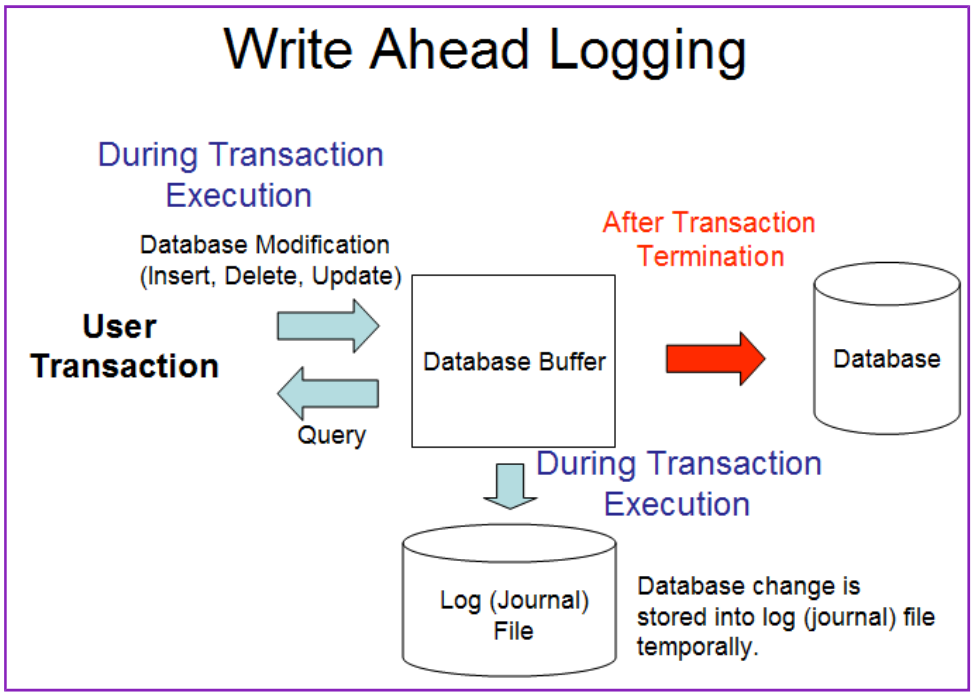
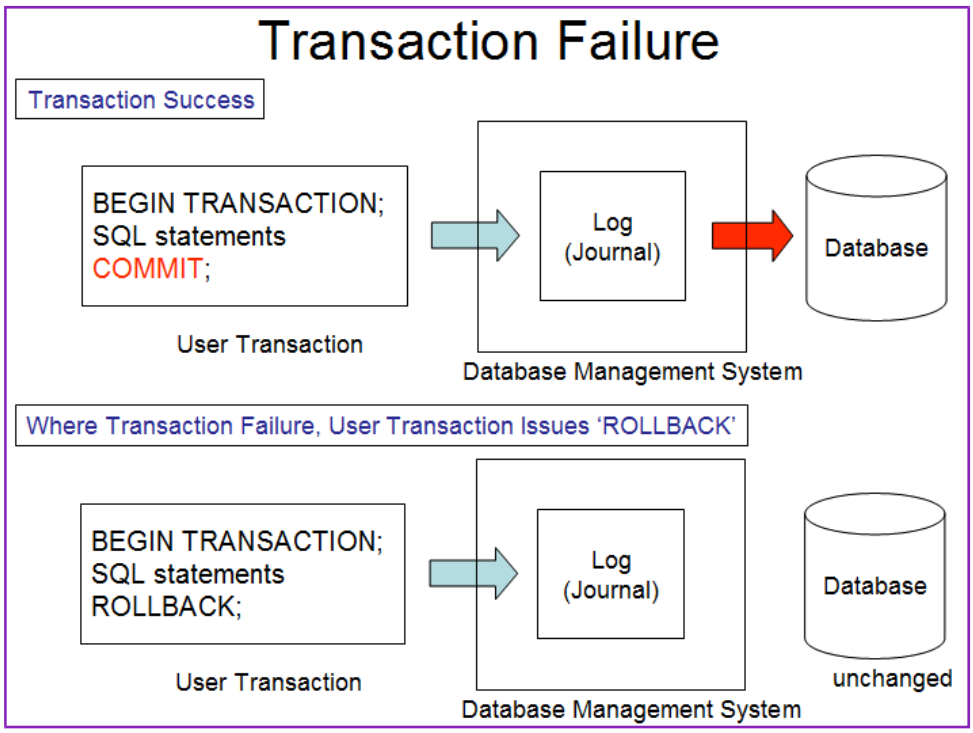
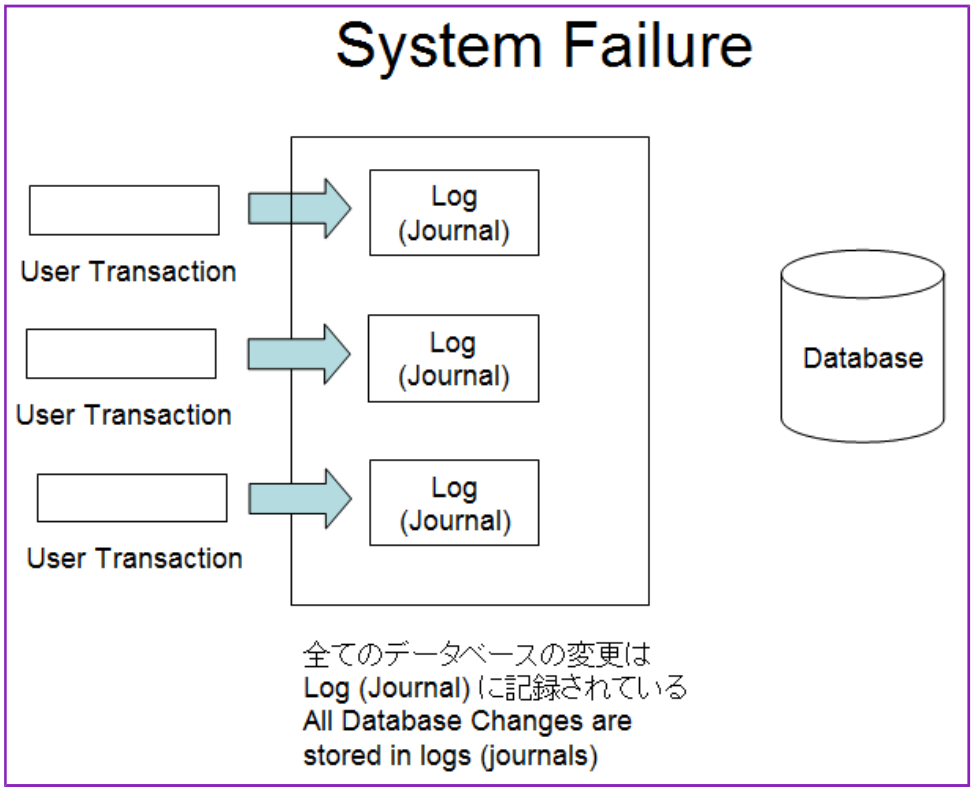
トランザクション

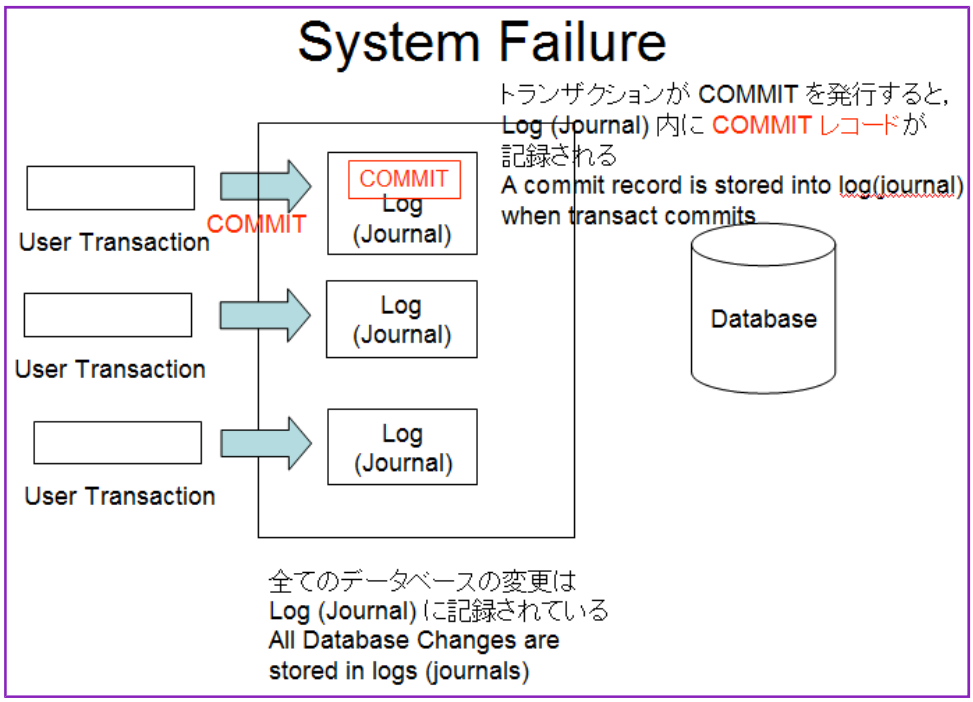


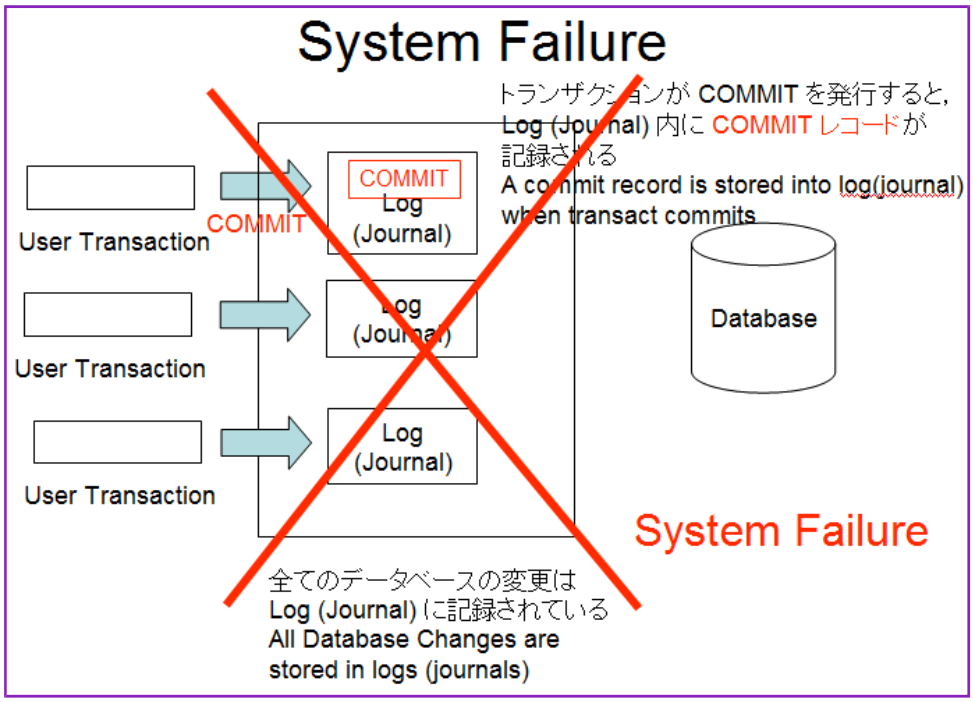
Failures

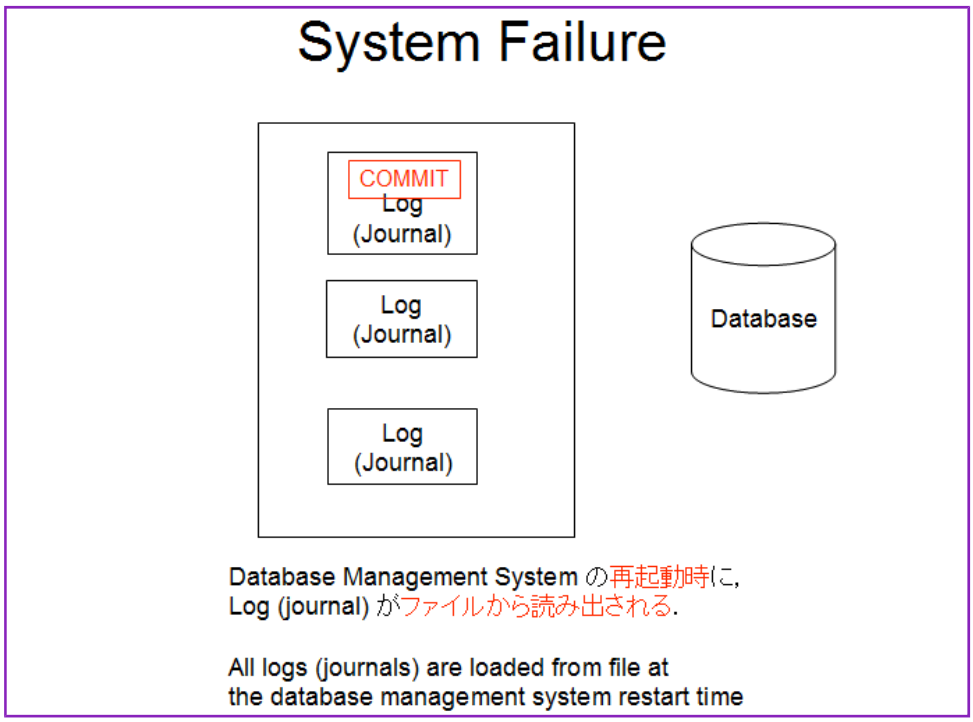
- Transaction Failure
  プログラムのバグ，一貫性制約違反などの理由で，個々のトランザクションが失敗 (fail)
  User transaction fails due to the application program bugs, database constraint violation, etc.
- System Failure
  OS, データベース管理システムが原因の失敗
  OS, Database Management System fails

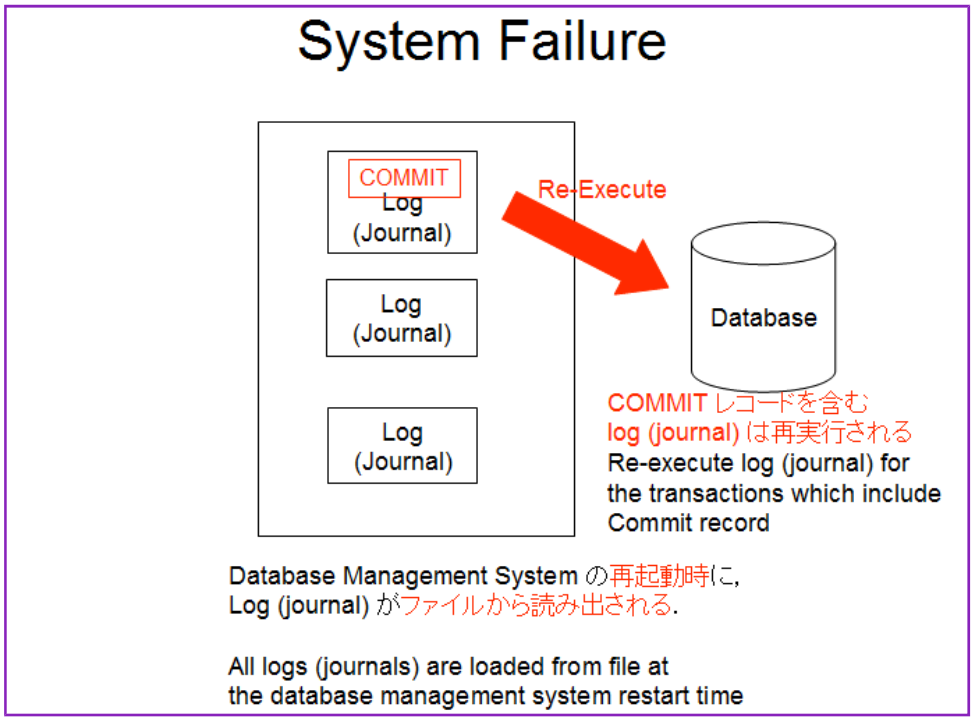

Transaction Failure
Transaction Success
BEGIN TRANSACTION;
SQL statements
COMMIT;
User Transaction
Log
(Journal)
Database
Database Management System
Where Transaction Failure, User Transaction Issues 'ROLLBACK'
BEGIN TRANSACTION;
SQL statements
ROLLBACK;
User Transaction
Log
(Journal)
Database
unchanged
Database Management System


System Failure
User Transaction
User Transaction
User Transaction
Log
(Journal)
Log
(Journal)
Log
(Journal)
Database
全てのデータベースの変更は
Log (Journal) に記録されている
All Database Changes are
stored in logs (journals)


System Failure
トランザクションが COMMIT を発行すると,
Log (Journal) 内に COMMIT レコードが
記録される
A commit record is stored into log(journal)
when transact commits
COMMIT
Log
(Journal)
User Transaction
COMMIT
User Transaction
Log
(Journal)
User Transaction
Log
(Journal)
Database
全てのデータベースの変更は
Log (Journal) に記録されている
All Database Changes are
stored in logs (journals)


System Failure
トランザクションが COMMIT を発行すると,
Log (Journal) 内に COMMIT レコードが
記録される
A commit record is stored into log(journal)
when transact commits
COMMIT
Log
(Journal)
User Transaction
COMMIT
User Transaction
Log
(Journal)
User Transaction
Log
(Journal)
Database
System Failure
全てのデータベースの変更は
Log (Journal) に記録されている
All Database Changes are
stored in logs (journals)





演習

演習を行うために必要になる機能や文法

- **BEGIN TRANSACTION, COMMIT, ROLLBACK**
  SQL 挿入文 (insert statement) などでのデータベース更新を行うときは, 最初に「BEGIN TRANSACTION;」を実行する. データベース更新が終わったら「COMMIT;」または「ROLLBACK;」を実行する.
  - COMMIT: ・・・「BEGIN TRANSACTION;」以降の全てのデータベース更新操作を**確定**したいとき
  - ROLLBACK: ・・・「BEGIN TRANSACTION;」以降の全てのデータベース更新操作を**破棄**したいとき
- **DELETE FROM <table-name> WHERE <expression>;**
  条件に合致する行の削除
- **UPDATE <table-name> SET <attribute-name>=<expression> WHERE <expression>**
  条件に合致する行に関するデータの更新
- **now**
  現在の日時を取得する SQLite3 の関数

A SQLite3 function to get the current datetime.

- **datetime**
  日時のデータを「YYYY-MM-DD HH:MM:SS」形式の文字列に変換する SQLite3 の関数

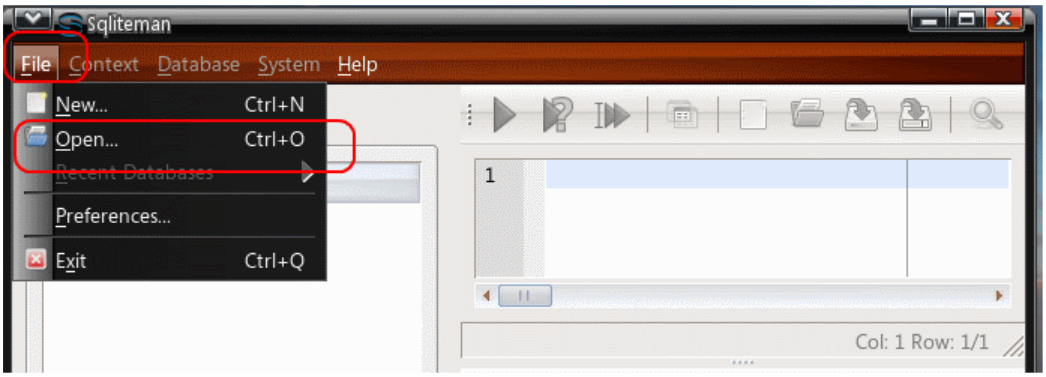
  A SQLite3 function to convert a datetime data into the string like "YYYY-MM-DD HH:MM:SS".

Sqliteman で既存のデータベースを開く(Open an existing database using Sqliteman)

すでに作成済みのデータベースを, 下記の手順で開くことができる.

以下の手順で, **既存のデータベースファイルを開く**. (Open an existing database file)

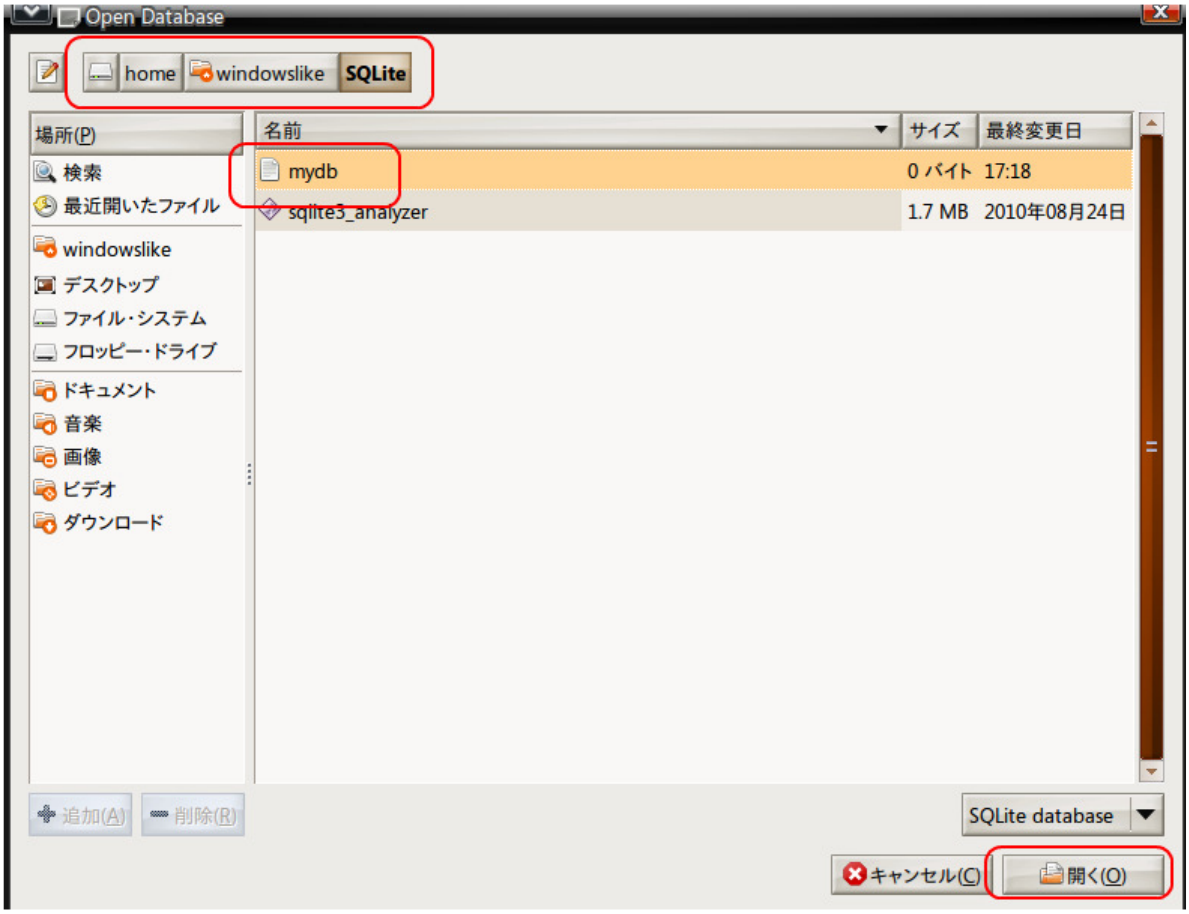
1. 「**File**」→「**Open**」



2. **データベースファイルを開く** (Open Database File)

■ **Ubuntu での実行例**(「SQLite/mydb」を開く場合)

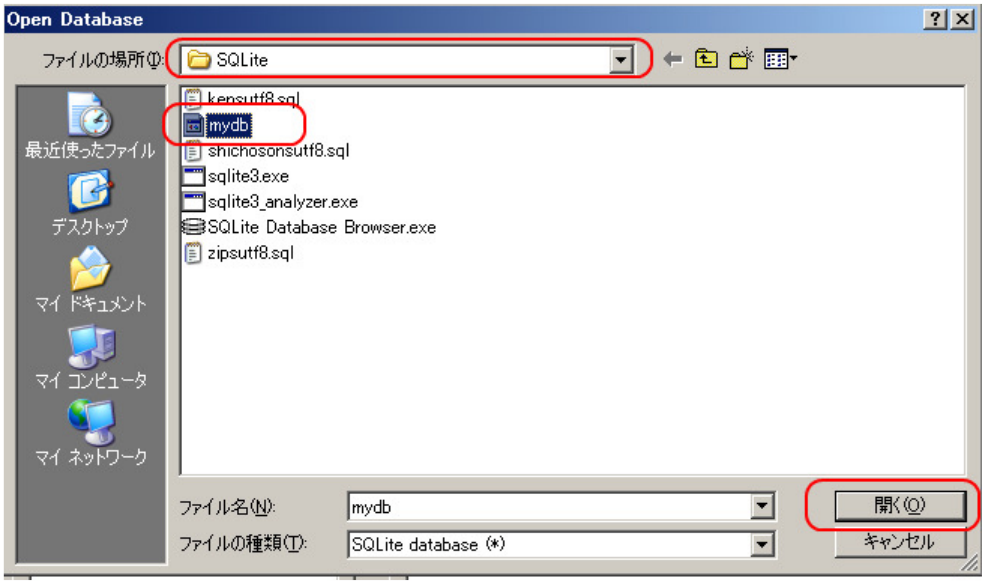
データベースファイル **SQLite/mydb** を選び,「開く」をクリック (Click '開く' after choosing the database file "SQLite/mydb")



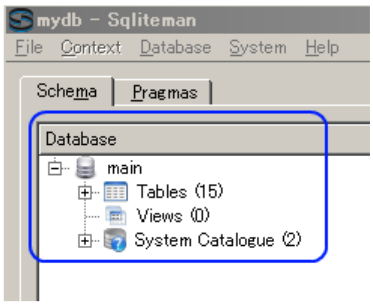
■ **Windows での実行例**(「C:¥SQLite¥mydb」を開く場合)

データベースファイル **C:¥SQLite¥mydb** を選び,「開く」をクリック (Click '開く' after choosing the database file "C:¥SQLite¥mydb")

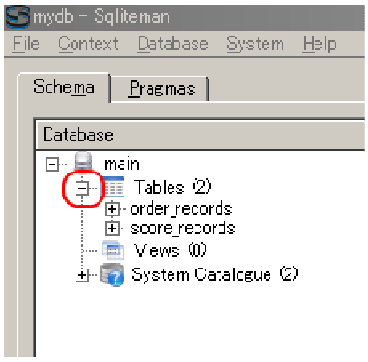
要するに, /home/<ユーザ名>/SQLite の**下**の mydb を選ぶ.



3. データベースの中身が表示されるので確認する (Database appears)



- **「Tables」を展開すると，テーブルの一覧 (List of Tables) が表示される**ので確認する (List of tables appears by clicking 'Tables')



SQL を用いたテーブル定義と一貫性制約の記述 (Table definition and integrity constraint specification using SQL)

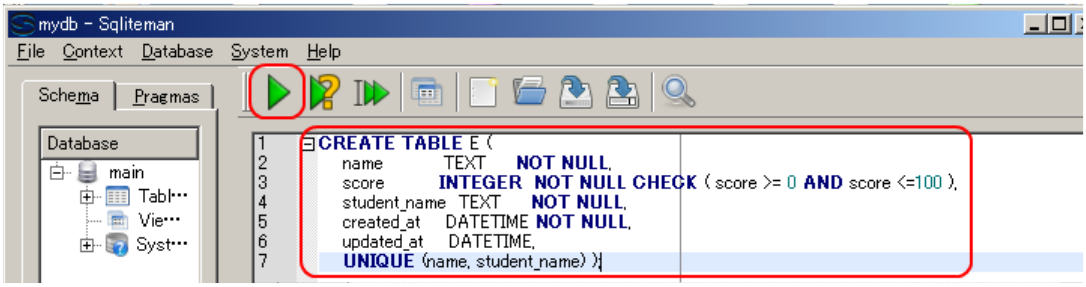
SQL を用いて，**E テーブルを定義し，一貫性制約を記述**する．(Define table 'E' and specify integrity constrants of the table using SQL)

リレーショナル・スキーマ (relational schema): E(name, score, student_name)

1. **E テーブルの定義** (Define a table)

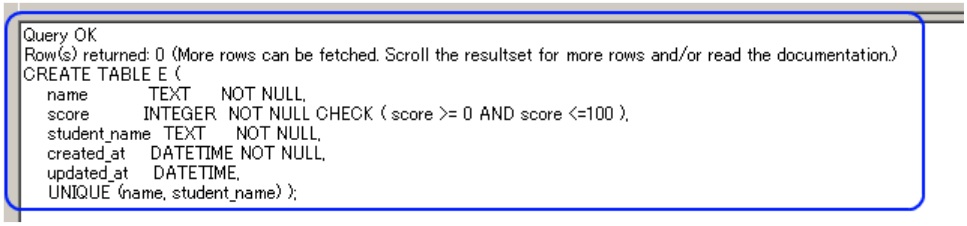
   次の SQL を入力し，「Run SQL」のアイコンをクリック (Write the following SQL, and click "Run SQL" icon).

```
CREATE TABLE E (
    name         TEXT     NOT NULL,
    score        INTEGER  NOT NULL CHECK ( score >= 0 AND score <=100 ),
    student_name TEXT     NOT NULL,
    created_at   DATETIME NOT NULL,
    updated_at   DATETIME,
    UNIQUE (name, student_name) );
```



2. **コンソールの確認** (Inspect console)

   エラーメッセージが出ていないことを確認



SQL を用いたテーブルへの行の挿入 (Insert rows into a table using SQL)

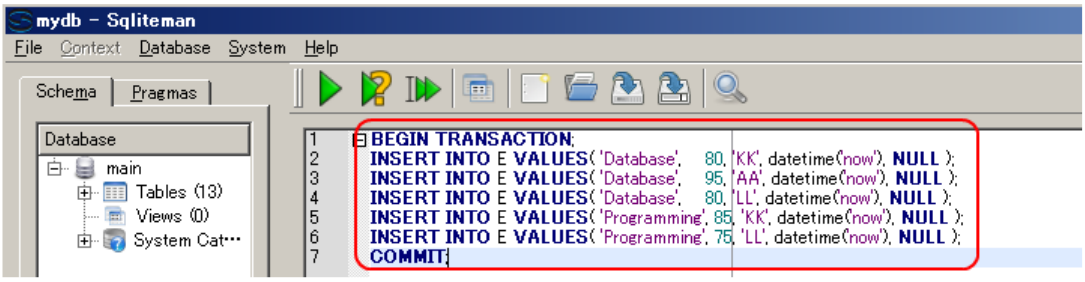
テーブルの 'created_at' 属性には，行を挿入する日時を記録する .

以下の手順で，SQL を用いて **E テーブルへの行の挿入**を行う (Insert rows into table 'E' using SQL)from

1. **SQL プログラムの記述**

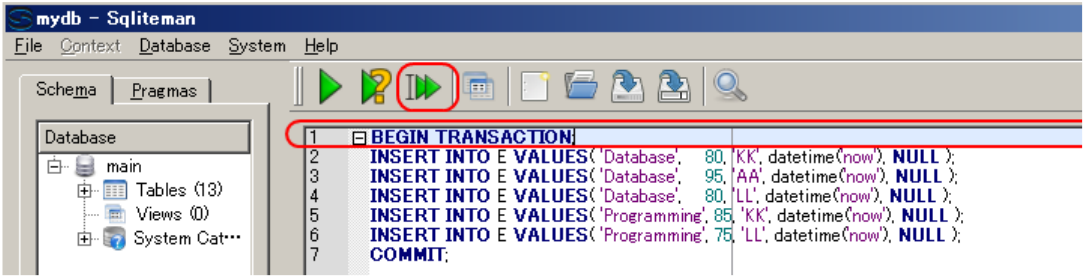
   「INSERT INTO ...」は行の挿入．ここには **4つの SQL 文**を書き，「BEGIN TRANSACTION」と「COMMIT」で囲む．
   ("INSERT INTO ..." means inserting a row into a table. Four SQL statements are wrote).

```
BEGIN TRANSACTION;
INSERT INTO E VALUES( 'Database',    80, 'KK', datetime('now'), NULL );
INSERT INTO E VALUES( 'Database',    95, 'AA', datetime('now'), NULL );
INSERT INTO E VALUES( 'Database',    80, 'LL', datetime('now'), NULL );
INSERT INTO E VALUES( 'Programming', 85, 'KK', datetime('now'), NULL );
INSERT INTO E VALUES( 'Programming', 75, 'LL', datetime('now'), NULL );
COMMIT;
```



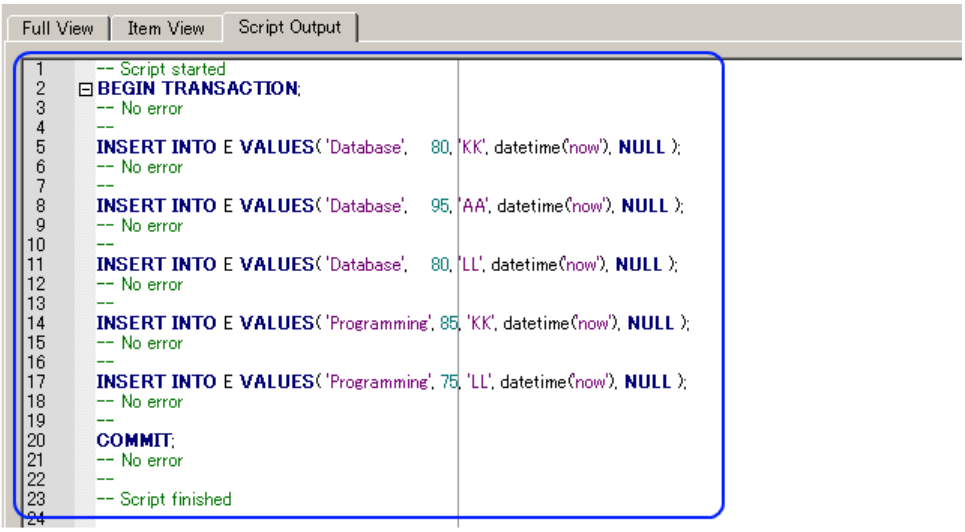
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

   複数の SQL 文を一括実行したいので，**カーソルを先頭行に移動**した後に，「Run multiple SQL statements ...」のボタンをクリックする．「Move the cursor to the top statement. Click "Run multiple SQL statements from current cursor position in one batch" icon)



3. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)

   エラーメッセージが出ていないことを確認

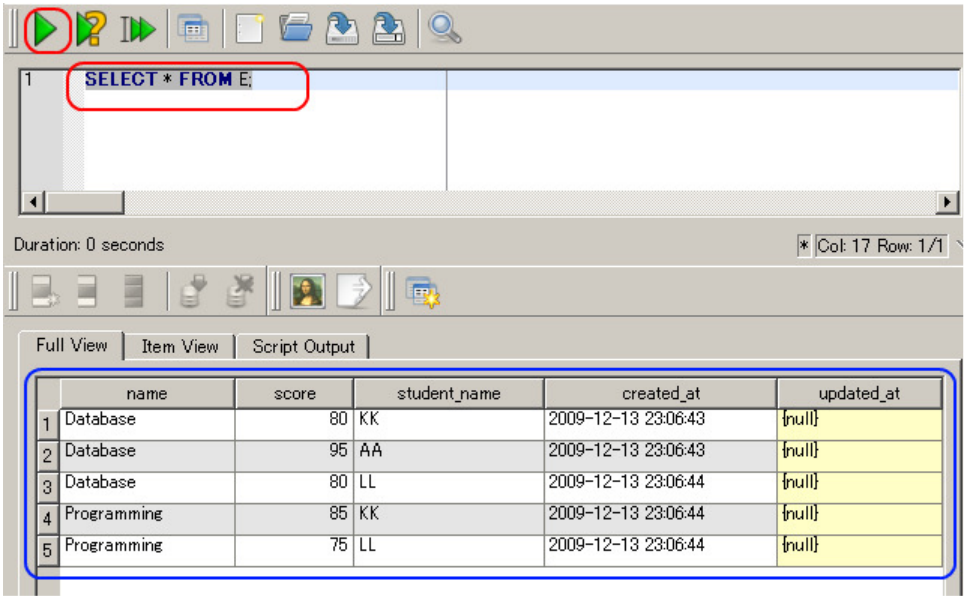


SQL 問い合わせの発行と評価結果の確認 (Issue SQL queries and inspect the results)

SQL を用いた**問い合わせ**の実行例を示す.

テーブルの全ての行の表示 (List all rows of a table)

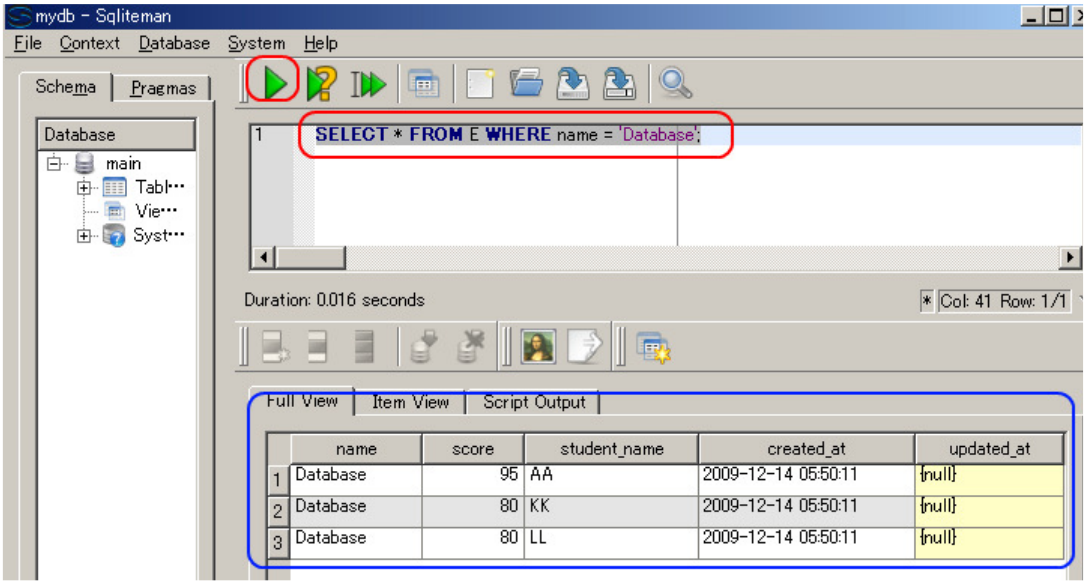
```
SELECT * FROM E;
```



	name	score	student_name	created_at	updated_at
1	Database	80	KK	2009-12-13 23:06:43	{null}
2	Database	95	AA	2009-12-13 23:06:43	{null}
3	Database	80	LL	2009-12-13 23:06:44	{null}
4	Programming	85	KK	2009-12-13 23:06:44	{null}
5	Programming	75	LL	2009-12-13 23:06:44	{null}

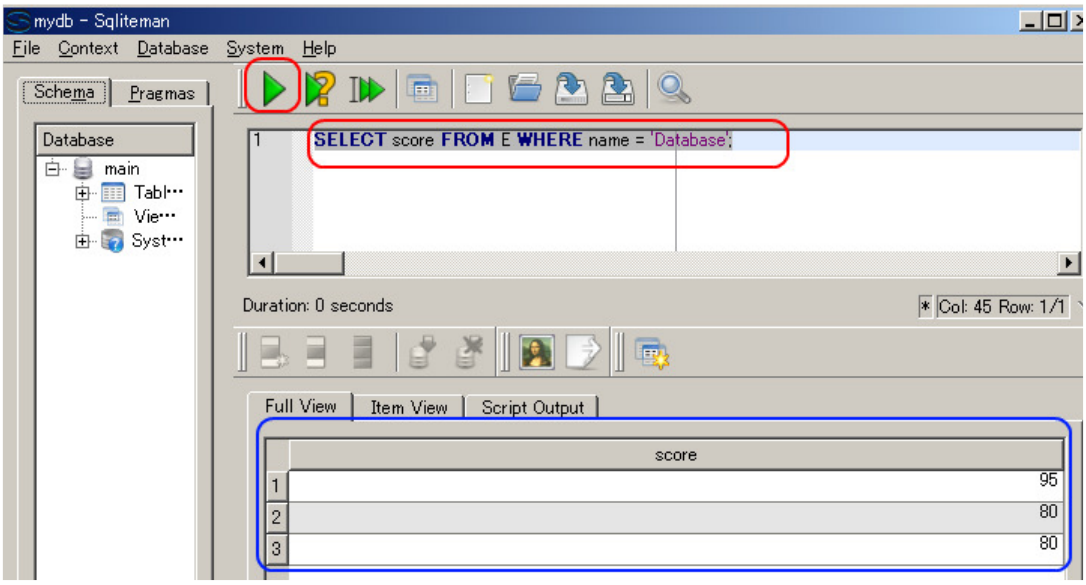
条件を満足する行のみの表示 (List the rows which satisfy a given condition)

```
SELECT * FROM E WHERE name = 'Database';
```



	name	score	student_name	created_at	updated_at
1	Database	95	AA	2009-12-14 05:50:11	{null}
2	Database	80	KK	2009-12-14 05:50:11	{null}
3	Database	80	LL	2009-12-14 05:50:11	{null}

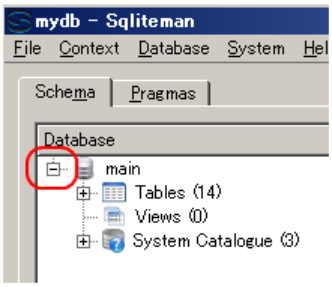
```
SELECT score FROM E WHERE name = 'Database';
```

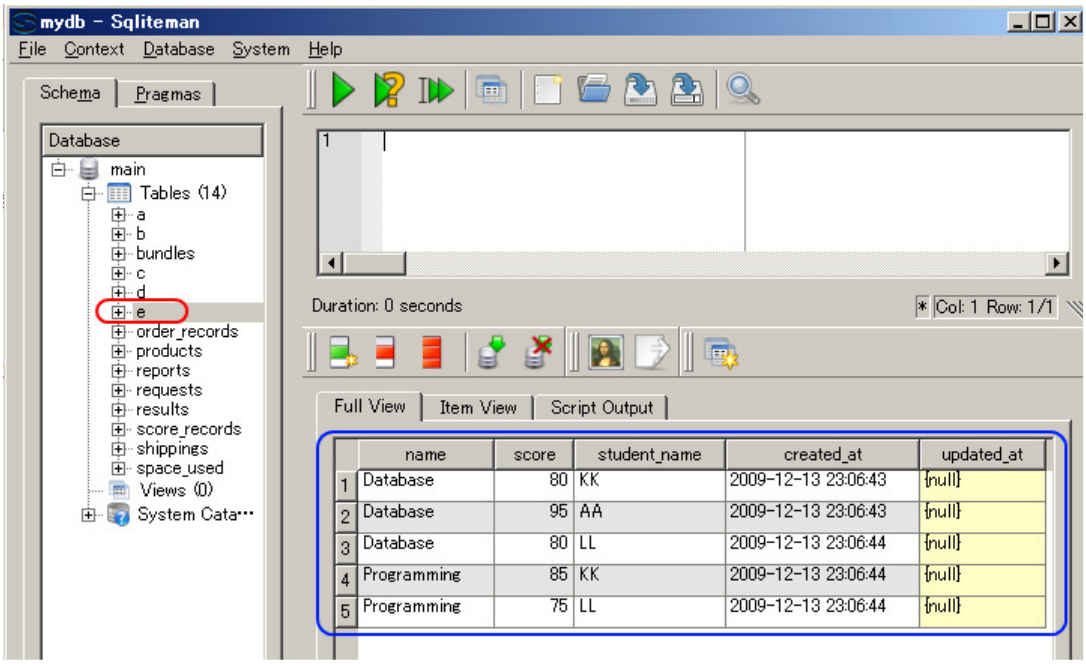


Sqliteman を用いたデータのブラウズ (Browse Data using Sqliteman)

- **E テーブル**

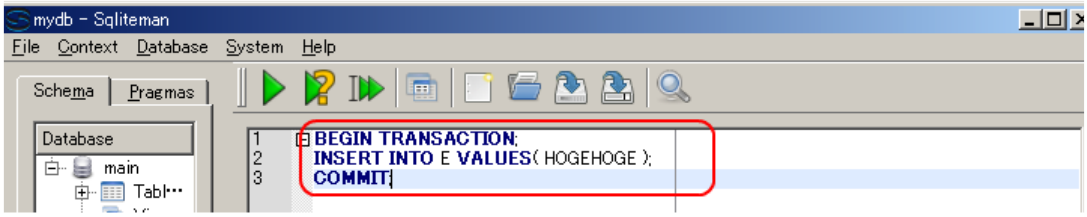
まず, オブジェクト・ブラウザ (Object Browser) の中の「**Tables**」を展開 (Click 'Tables')

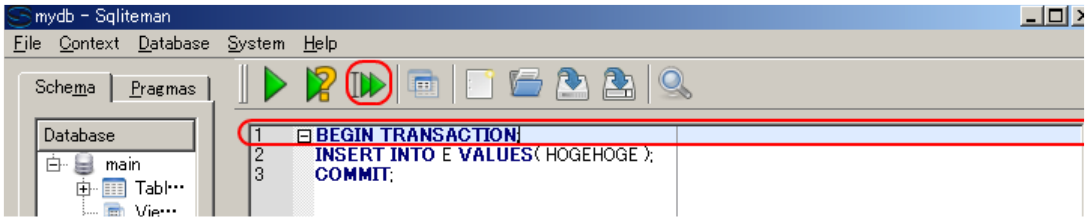


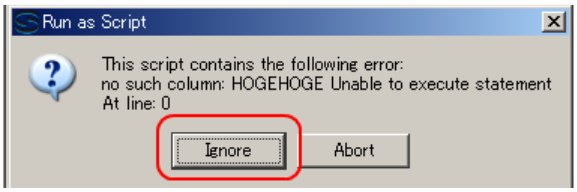


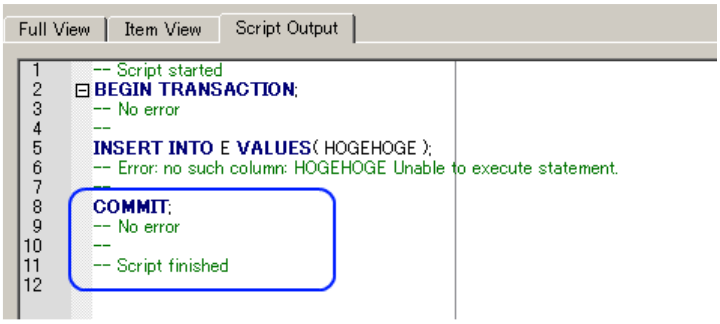
※ もし, データに間違いがあれば, このウインドウで修正できる (If you find any mistakes, you can modify the data using this window).

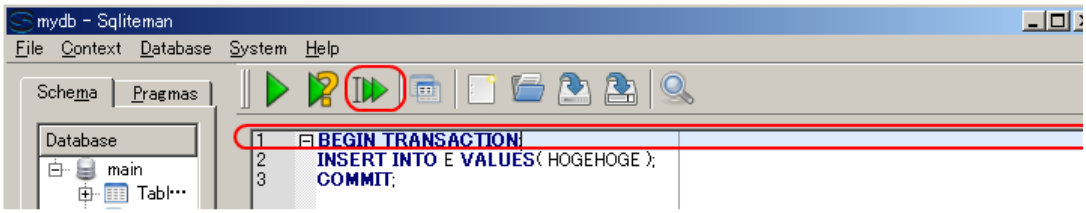
トランザクション開始後に、トランザクション開始できない (Can not start a transaction within a transaction


mydb - Sqliteman
File Context Database System Help
Schema Pragmas
Database
main
1 BEGIN TRANSACTION;
2 INSERT INTO E VALUES(HOGEHOGE);
3 COMMIT;

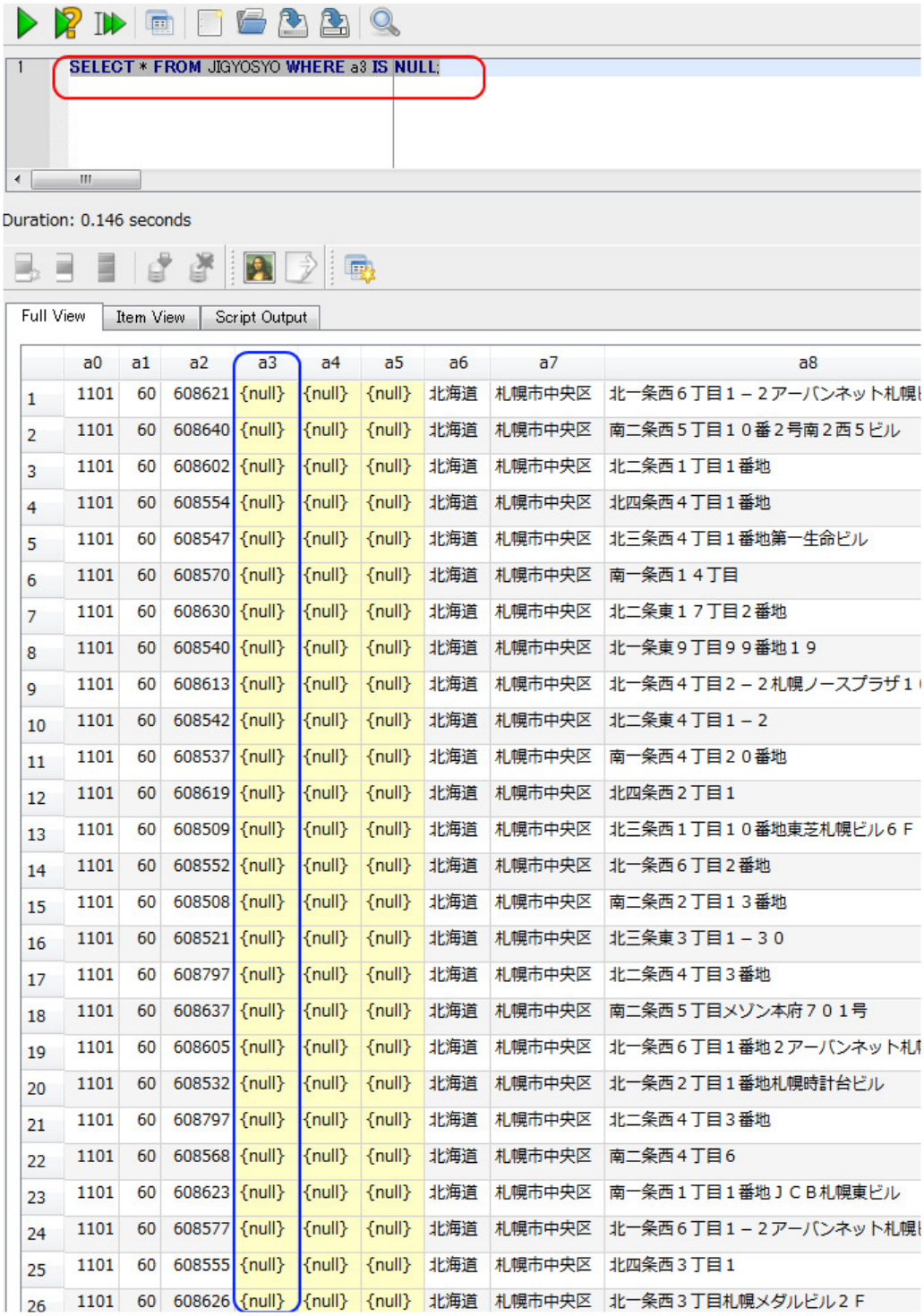

mydb - Sqliteman
File Context Database System Help
Schema Pragmas
Database
main
1 BEGIN TRANSACTION;
2 INSERT INTO E VALUES(HOGEHOGE);
3 COMMIT;


Run as Script
This script contains the following error:
no such column: HOGEHOGE Unable to execute statement
At line: 0
Ignore
Abort


Full View
Item View
Script Output
1 -- Script started
2 BEGIN TRANSACTION;
3 -- No error
4 --
5 INSERT INTO E VALUES(HOGEHOGE);
6 -- Error: no such column: HOGEHOGE Unable to execute statement.
7 --
8 COMMIT;
9 -- No error
10 --
11 -- Script finished
12


mydb - Sqliteman
File Context Database System Help
Schema Pragmas
Database
main
1 BEGIN TRANSACTION;
2 INSERT INTO E VALUES(HOGEHOGE);
3 COMMIT;


Run as Script
This script contains the following error:
no such column: HOGEHOGE Unable to execute statement
At line: 0
Ignore
Abort



```
1  SELECT * FROM JIGYOSYO WHERE a3 IS NULL;
```

Duration: 0.146 seconds

Full View | Item View | Script Output

	a0	a1	a2	a3	a4	a5	a6	a7	a8
1	1101	60	608621	{null}	{null}	{null}	北海道	札幌市中央区	北一条西６丁目１－２アーバンネット札幌
2	1101	60	608640	{null}	{null}	{null}	北海道	札幌市中央区	南二条西５丁目１０番２号南２西５ビル
3	1101	60	608602	{null}	{null}	{null}	北海道	札幌市中央区	北二条西１丁目１番地
4	1101	60	608554	{null}	{null}	{null}	北海道	札幌市中央区	北四条西４丁目１番地
5	1101	60	608547	{null}	{null}	{null}	北海道	札幌市中央区	北三条西４丁目１番地第一生命ビル
6	1101	60	608570	{null}	{null}	{null}	北海道	札幌市中央区	南一条西１４丁目
7	1101	60	608630	{null}	{null}	{null}	北海道	札幌市中央区	北二条東１７丁目２番地
8	1101	60	608540	{null}	{null}	{null}	北海道	札幌市中央区	北一条東９丁目９９番地１９
9	1101	60	608613	{null}	{null}	{null}	北海道	札幌市中央区	北一条西４丁目２－２札幌ノースプラザ１
10	1101	60	608542	{null}	{null}	{null}	北海道	札幌市中央区	北二条東４丁目１－２
11	1101	60	608537	{null}	{null}	{null}	北海道	札幌市中央区	南一条西４丁目２０番地
12	1101	60	608619	{null}	{null}	{null}	北海道	札幌市中央区	北四条西２丁目１
13	1101	60	608509	{null}	{null}	{null}	北海道	札幌市中央区	北三条西１丁目１０番地東芝札幌ビル６Ｆ
14	1101	60	608552	{null}	{null}	{null}	北海道	札幌市中央区	北一条西６丁目２番地
15	1101	60	608508	{null}	{null}	{null}	北海道	札幌市中央区	南二条西２丁目１３番地
16	1101	60	608521	{null}	{null}	{null}	北海道	札幌市中央区	北三条東３丁目１－３０
17	1101	60	608797	{null}	{null}	{null}	北海道	札幌市中央区	北二条西４丁目３番地
18	1101	60	608637	{null}	{null}	{null}	北海道	札幌市中央区	南二条西５丁目メゾン本府７０１号
19	1101	60	608605	{null}	{null}	{null}	北海道	札幌市中央区	北一条西６丁目１番地２アーバンネット札
20	1101	60	608532	{null}	{null}	{null}	北海道	札幌市中央区	北一条西２丁目１番地札幌時計台ビル
21	1101	60	608797	{null}	{null}	{null}	北海道	札幌市中央区	北二条西４丁目３番地
22	1101	60	608568	{null}	{null}	{null}	北海道	札幌市中央区	南二条西４丁目６
23	1101	60	608623	{null}	{null}	{null}	北海道	札幌市中央区	南一条西１丁目１番地ＪＣＢ札幌東ビル
24	1101	60	608577	{null}	{null}	{null}	北海道	札幌市中央区	北一条西６丁目１－２アーバンネット札幌
25	1101	60	608555	{null}	{null}	{null}	北海道	札幌市中央区	北四条西３丁目１
26	1101	60	608626	{null}	{null}	{null}	北海道	札幌市中央区	北一条西３丁目札幌メダルビル２Ｆ

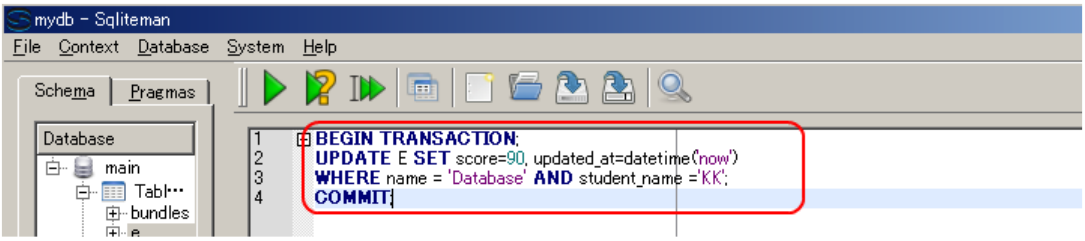
SQL を用いた更新 (Update using SQL)

SQL を用いた**データの更新 (update)**の実行例を示す．「**UPDATE <table-name> SET <attribute-name>=<expression> WHERE <expression>**」の形をした SQL は，データの更新である．この SQL 文を「BEGIN TRANSACTION」と「COMMIT」で囲む．
("UPDATE ... SET ..." means database update).

1. **SQL プログラムの記述**

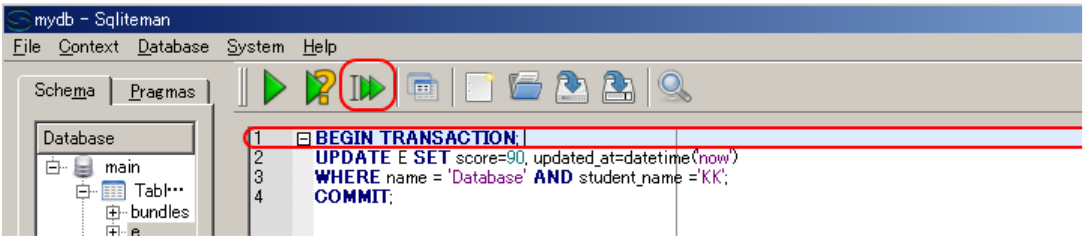
   「UPDATE ... SET ...」は更新．ここには **1つの SQL 文**を書き，「BEGIN TRANSACTION」と「COMMIT」で囲む．
   ("UPDATE ... SET ..." means database update.

```
BEGIN TRANSACTION;
UPDATE E SET score=90, updated_at=datetime('now')
WHERE name = 'Database' AND student_name ='KK';
COMMIT;
```



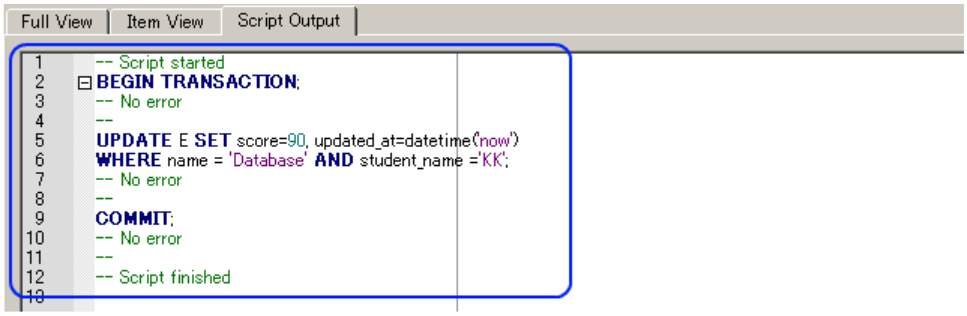
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

   複数の SQL 文を一括実行したいので，**カーソルを先頭行に移動**した後に，「Run multiple SQL statements …」のボタンをクリックする．「Move the cursor to the top statement. Click "Run multiple SQL statements from current cursor position in one batch" icon)

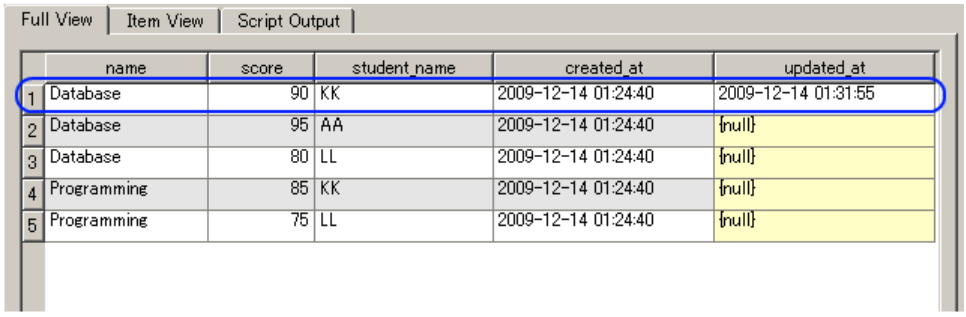


3. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)

   エラーメッセージが出ていないことを確認



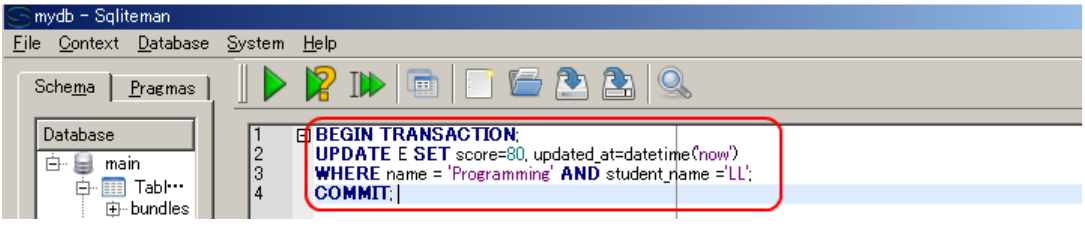
4. **E テーブル** のブラウズ (Browse table 'E')



5. **SQL プログラムの記述**

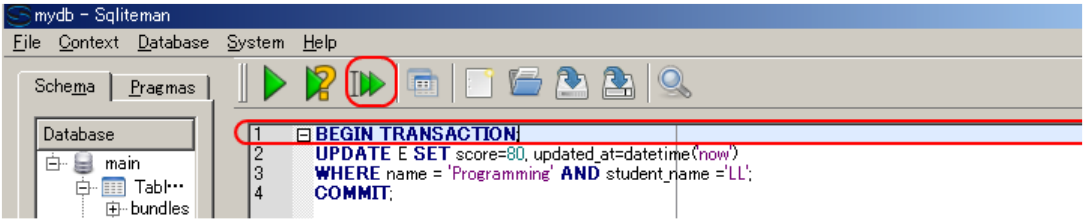
   「UPDATE … SET …」は更新．ここには **1つの SQL 文**を書き，「BEGIN TRANSACTION」と「COMMIT」で囲む．("UPDATE … SET …" means database update.

```
BEGIN TRANSACTION;
UPDATE E SET score=80, updated_at=datetime('now')
WHERE name = 'Programming' AND student_name ='LL';
COMMIT;
```

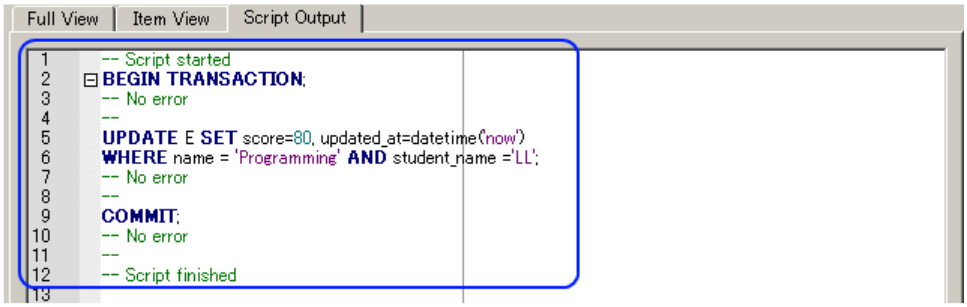


6. **複数の SQL 文の一括実行** (Run multiple SQL statements)

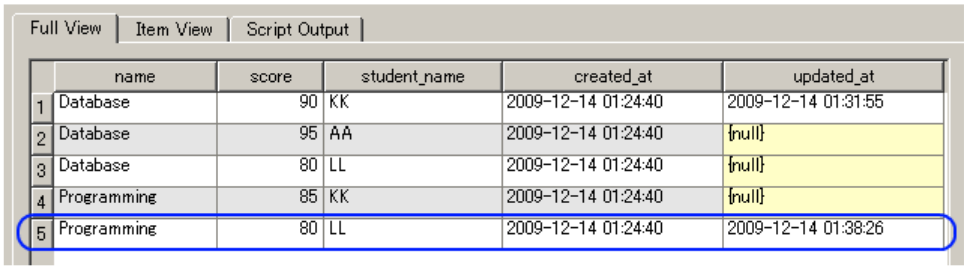
   複数の SQL 文を一括実行したいので，**カーソルを先頭行に移動**した後に，「Run multiple SQL statements …」のボタンをクリックする．「Move the cursor to the top statement. Click "Run multiple SQL statements from current cursor position in one batch" icon)



7. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)
   エラーメッセージが出ていないことを確認



8. **E テーブル** のブラウズ (Browse table 'E')



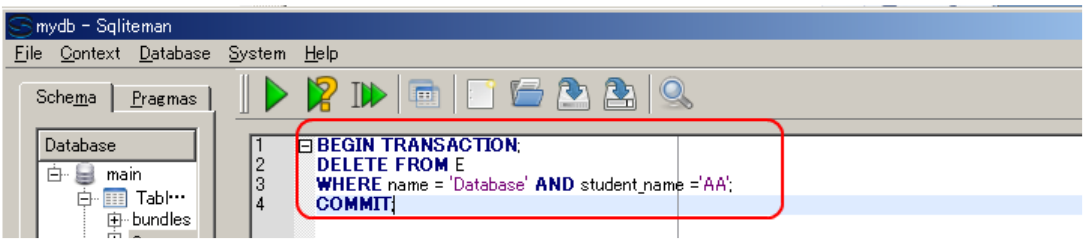
SQL を用いた行の削除 (Delete row(s) using SQL)

SQL を用いた**行の削除 (delete row(s))**の実行例を示す.

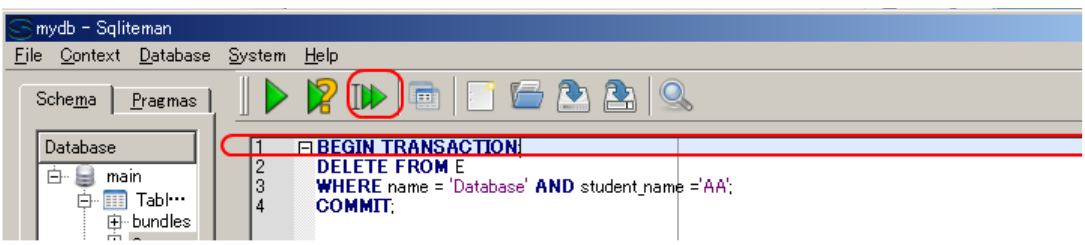
「**DELETE FROM <table-name> WHERE <expression>;**」の形をした SQL は行の削除である. この SQL 文を「BEGIN TRANSACTION」と「COMMIT」で囲む. ("DELETE FROM ... WHERE ..." means deletion of rows).

1. **SQL プログラムの記述**

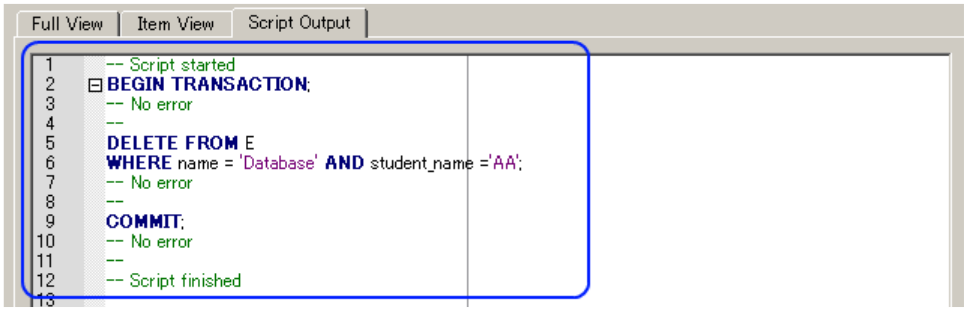
```
BEGIN TRANSACTION;
DELETE FROM E
WHERE name = 'Database' AND student_name = 'AA' ;
COMMIT;
```



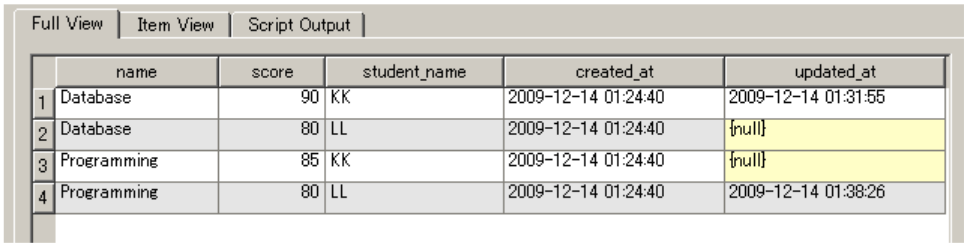
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)
   複数の SQL 文を一括実行したいので, **カーソルを先頭行に移動**した後に,「Run multiple SQL statements ...」のボタンをクリックする.「Move the cursor to the top statement. Click "Run multiple SQL statements from current cursor position in one batch" icon)



3. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)
   エラーメッセージが出ていないことを確認



4. **E テーブル** のブラウズ (Browse table 'E')

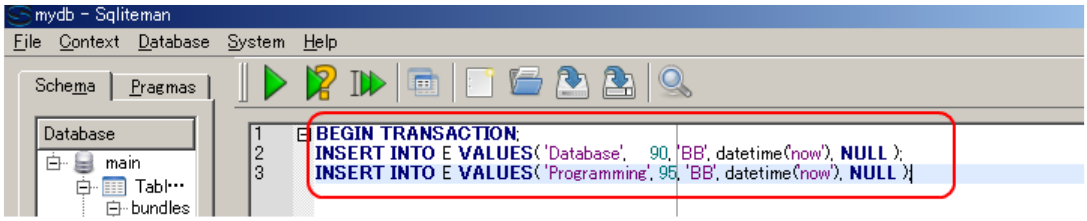


ジャーナルファイルの確認 (Examine a journal file)

1. **SQL プログラムの記述**

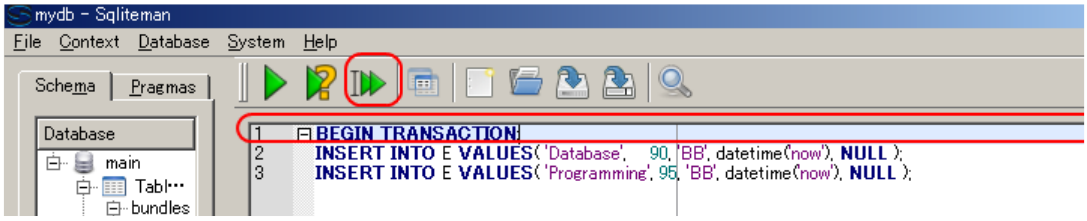
   今度は COMMIT を書かない (Without COMMIT)

```
BEGIN TRANSACTION;
INSERT INTO E VALUES( 'Database',    90, 'BB', datetime('now'), NULL );
INSERT INTO E VALUES( 'Programming', 95, 'BB', datetime('now'), NULL );
```



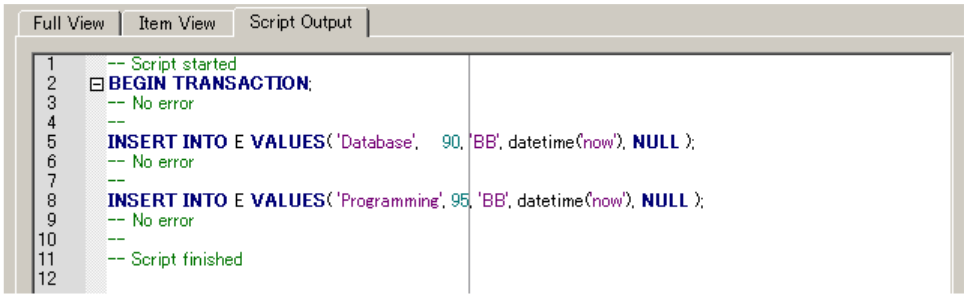
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

   複数の SQL 文を一括実行したいので，**カーソルを先頭行に移動**した後に，「Run multiple SQL statements ...」のボタンをクリックする．「Move the cursor to the top statement. Click "Run multiple SQL statements from current cursor position in one batch" icon)



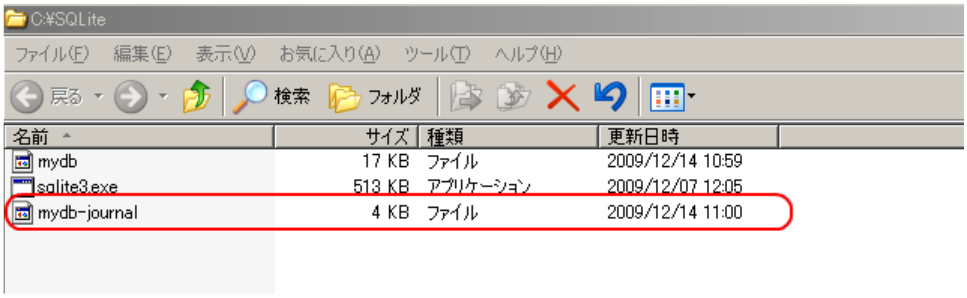
3. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)

   エラーメッセージが出ていないことを確認



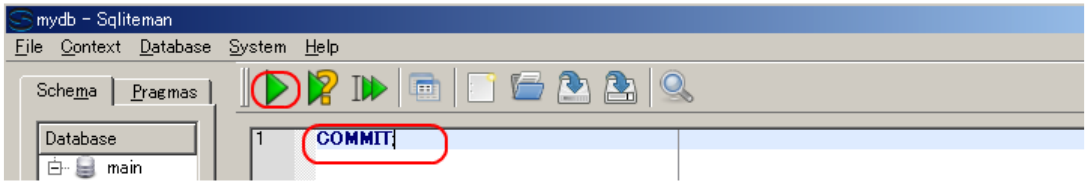
4. **ジャーナル・ファイルの確認** (Journal File)

   C:¥SQLite を開く．「mydb-journal」がジャーナル・ファイルである．



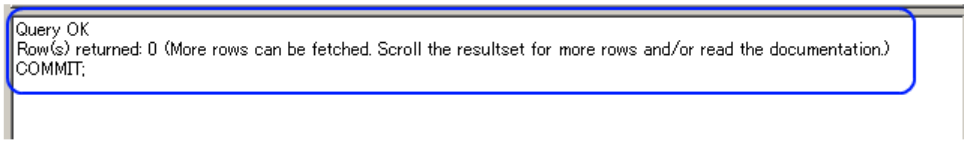
5. Sqliteman で **COMMIT の発行** (COMMIT)

```
COMMIT;
```



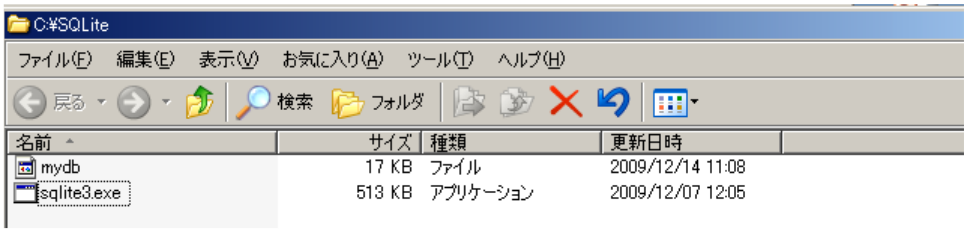
6. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)

   エラーメッセージが出ていないことを確認



7. **ジャーナル-ファイルの確認** (Journal File)

   C:¥SQLite を開く. COMMIT を発行した. データベースの修正は終わっている. そのためジャーナルファイルが消えている.

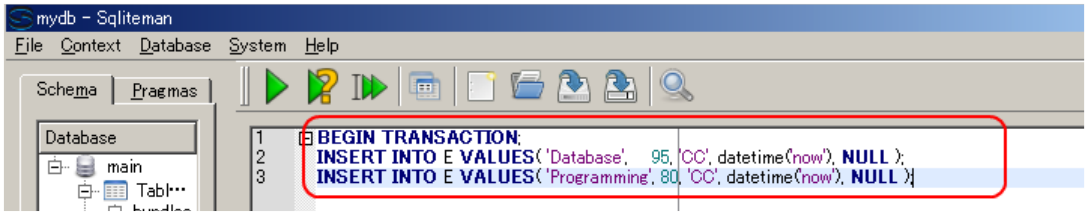


ROLLBACK の確認 (Example a journal file)

1. **SQL プログラムの記述**

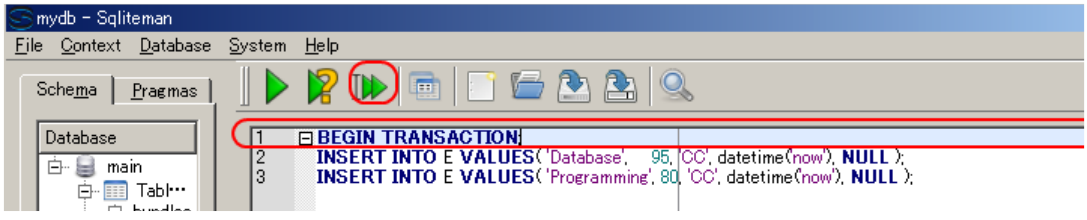
   今度も COMMIT を書かない (Without COMMIT)

```
BEGIN TRANSACTION;
INSERT INTO E VALUES( 'Database',    95, 'CC', datetime('now'), NULL );
INSERT INTO E VALUES( 'Programming', 80, 'CC', datetime('now'), NULL );
```

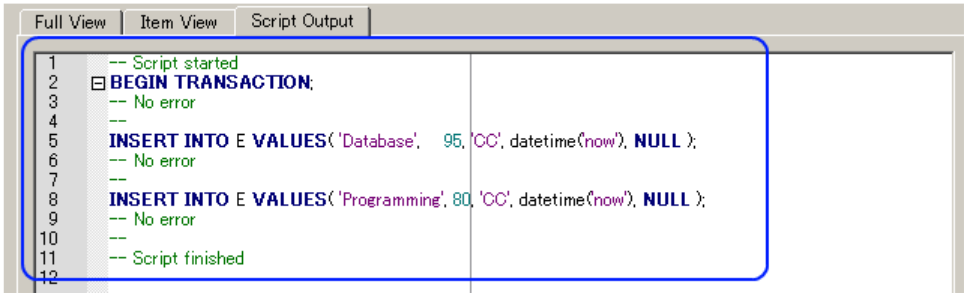


2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

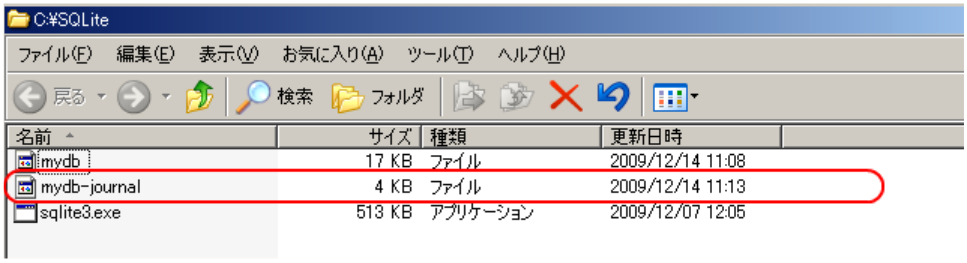
   複数の SQL 文を一括実行したいので, **カーソルを先頭行に移動**した後に,「Run multiple SQL statements ...」のボタンをクリックする.「Move the cursor to the top statement. Click "Run multiple SQL statements from current cursor position in one batch" icon)



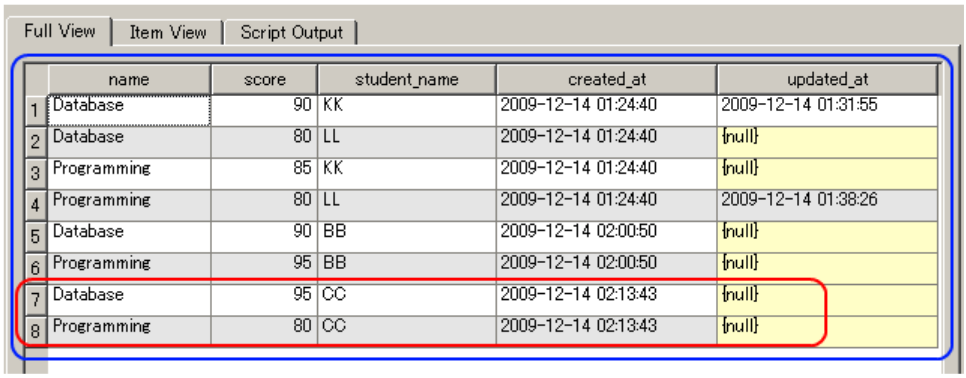
3. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)
   エラーメッセージが出ていないことを確認



4. **ジャーナル・ファイルの確認** (Journal File disappears)
   C:¥SQLite を開く.「mydb-journal」がジャーナル・ファイルである.

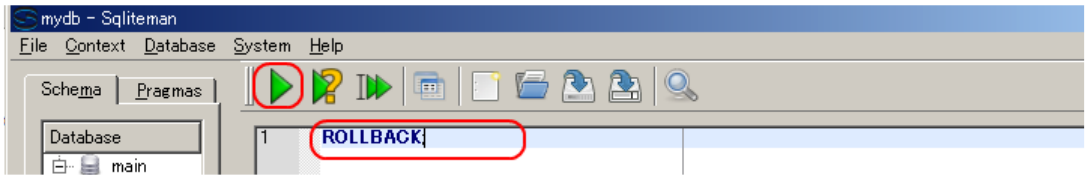


5. **E テーブル** のブラウズ (Browse table 'E')
   「INSERT INTO **E** VALUES('Database', 95, 'CC', datetime('now'), NULL);」,「INSERT INTO **E** VALUES ('Programming', 80, 'CC', datetime('now'), NULL);」の結果がデータベースに反映されているように見える. (It seems that database update is finished)

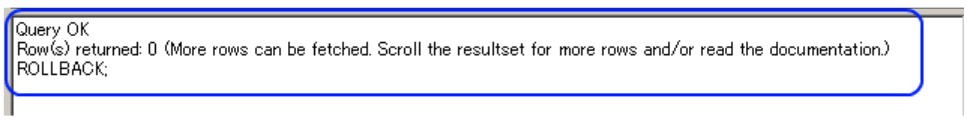


6. Sqliteman で **ROLLBACK の発行** (ROLLBACK)
   今度は ROLLBACK を発行する. (Issue ROLLBACK)

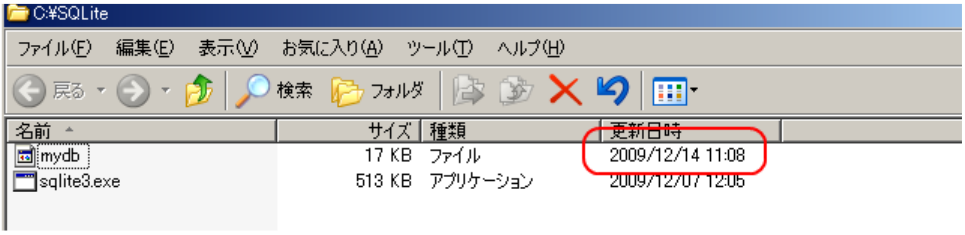
```
ROLLBACK;
```



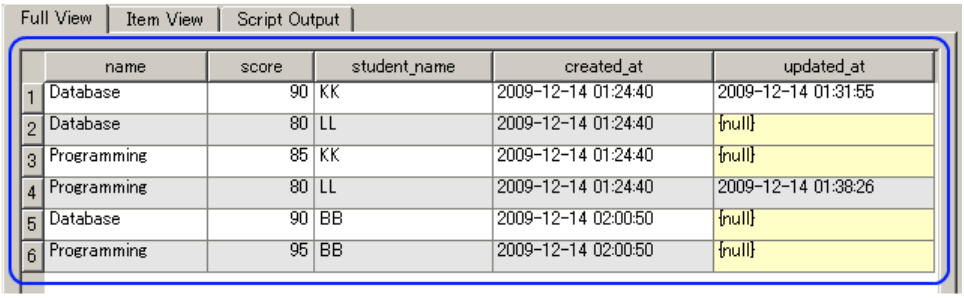
7. **「Script Output」ウインドウの確認** (Inspect "Script Output" window)
   エラーメッセージが出ていないことを確認



8. **データベース・ファイルとジャーナル・ファイルの確認** (Journal File disappears)
   実はデータベースファイルは一切修正されていない (Database file is unchaned. It is observerd by using the file timestamp).



9. **E テーブル** のブラウズ (Browse table 'E')



	name	score	student_name	created_at	updated_at
1	Database	90	KK	2009-12-14 01:24:40	2009-12-14 01:31:55
2	Database	80	LL	2009-12-14 01:24:40	{null}
3	Programming	85	KK	2009-12-14 01:24:40	{null}
4	Programming	80	LL	2009-12-14 01:24:40	2009-12-14 01:38:26
5	Database	90	BB	2009-12-14 02:00:50	{null}
6	Programming	95	BB	2009-12-14 02:00:50	{null}

演習問題と解答例

次の問いに答えよ. その後, 下記の解答例を確認せよ. Answer the following questions. Then, inspect answers described below.

問い (Questions)

次の PTABLE テーブルに関する問題 (About the following 'PTABLE' table)

```
name    | type   | color
-----------------------------
apple   | fruit  | red
apple   | fruit  | blue
rose    | flower | white
rose    | flower | red
rose    | flower | yellow
```

1. PTABLE を次のように更新する SQL を書きなさい (Write a SQL which updates the table).

```
name    | type   | color
-----------------------------
apple   | fruit  | red
apple   | fruit  | blue
rose    | flower | white
rose    | flower | red
rose    | flower | blue
```

2. さらに, PTABLE から行を削除し, 次のテーブルに変更する SQL を書きなさい (Write a SQL which delete a row).

```
name    | type   | color
-----------------------------
apple   | fruit  | red
rose    | flower | white
rose    | flower | red
rose    | flower | blue
```

3. SQLite を使い, 下記の SQL を評価させなさい (Evaluate the following SQL)

```
(1)
        CREATE TABLE GG (
                id INTEGER PRIMARY KEY,
                name TEXT );
(2)
        BEGIN TRANSACTION;
        INSERT INTO GG values ( 1, 'X' );
        COMMIT;
        select * from GG;
(3)
        BEGIN TRANSACTION;
        INSERT INTO GG values ( 2, 'Y' );
        ROLLBACK;
        select * from GG;
```

解答例 (Answers)

1. `UPDATE PTABLE SET color='blue' WHERE name='rose' AND type='flower' AND color='yellow';`

2. `DELETE FROM PTABLE WHERE name='apple' AND type='fruit' AND color='blue';`