SQL 問い合わせ

URL: http://www.db.is.kyushu-u.ac.jp/rinkou/addb/3.html

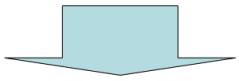
データベース内の1つのテーブルに対する SQL 問い合わせの例. 問い合わせの結果として，新しいテーブルが 1つ生成される.

table 'products'
(in a database)

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

SQL query

```
SELECT *
FROM products
WHERE type = 'orange';
```



id	product_name	type	cost
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20

Query result is one new table

---

table 'products'
(in a database)

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

SQL query

```
SELECT *
FROM products
WHERE cost > 25;
```

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30

Query result is one new table

---

table 'products'
(in a database)

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

SQL query

```
SELECT product_name, cost
FROM products;
```

product_name	cost
Fukuoka apple	50
Kumamoto orange L	30
Kumamoto orange M	20
Fukuoka melon	NULL

Query result is one new table

---

table 'products'
(in a database)

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

SQL query

```
SELECT product_name, cost
FROM products
WHERE cost > 25;
```

product_name	cost
Fukuoka apple	50
Kumamoto orange L	30

Query result is one new table

結合問い合わせの例

R

A	B
a	b
d	a
a	d

S

B	C	D
b	c	f
d	e	a
d	e	c

図. 2つのテーブル R と S

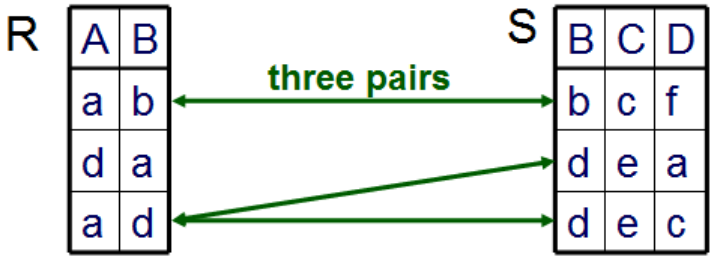


図. 条件 (condition) R.B = S.B

```
SELECT *
FROM R, S
WHERE R.B = S.B;
```

A	R.B	S.B	C	D
a	b	b	c	f
a	d	d	e	a
a	d	d	e	c

Join Query Example　　Query Result is One Table

図. 結合問い合わせの例

```
SELECT *
FROM R, S
WHERE R.B = S.B
    AND C = 'e';
```

A	R.B	S.B	C	D
a	d	d	e	a
a	d	d	e	c

Join Query Example　　Query Result is One Table

図. 結合問い合わせの例(2)

■ **If you want to get only the two attributes A and D**

```
SELECT A, D
FROM R, S
WHERE R.B = S.B;
```

A	D
a	f
a	a
a	c

Join Query Example　　Query Result is One Table

図. 結合問い合わせの例(3)

- **SQL の比較演算子**

  比較演算子は2項述語である．SQL で，数値や文字列の比較演算子には次のようなものがある

  - <
  - <=
  - =
  - >=

- >
- <>

- **SQL の論理演算子**

  AND, OR, NOT

- **IS NULL**

  NULL 値を持つような行を得るために使うことができる.

- **¥'**

  「¥'」は,「'」を含むような文字列を扱うための貴方

- **LIKEと%と_**

  SQLでの文字列のマッチングに使うキーワード

- **SQL 問い合わせ (SQL query)**

  SQL 問い合わせは, リレーショナルデータベースに格納された1つまたは複数のテーブルを使う. 問い合わせの評価結果として, リレーショナルデータベース内のテーブルをそのままの形で得るということはめったになく, **新しいテーブルが 1つ作られる**.

  SQL 問い合わせのプログラムは, SELECT, FROM, WHERE を使うのが基本である. 文法は次のようになる. ※ SQL の文法は多彩なので, ここでは, 基本的な場合に説明を絞る.

```
SELECT <list-of(result-column)>
FROM <list-of(table-name)>
```

  あるいは

```
SELECT <list-of(result-column)>
FROM <list-of(table-name)>
WHERE <expression>
```

  例えば,

```
SELDCT A1, A2, ..., An
FROM   T1, T2, ..., Tm
WHERE  <expression>
```

  のように書くと, m 個のテーブル T1, T2, ..., Tm の直積集合から <expression> を満足する行のみを選び, そうして出来たテーブルの属性 A1, A2, ..., An を出力するという意味になる.

  WHERE の部分が無い場合, 例えば,

```
SELDCT A1, A2, ..., An
FROM   T1, T2, ..., Tm
```

  のように書くと, m 個のテーブル T1, T2, ..., Tm の直積集合から属性 A1, A2, ..., An を出力するという意味になる.

  - **SELECT <list-of(result-column)>**

    SELECT の後には, 問い合わせの結果として得たい**列名あるいは列名を含む式の, 1個以上の並び**を書く. 2個以上ある場合には**半角のカンマ「,」で区切る**. 問い合わせ結果として得られるテーブルの中の列の並びは, SQL の SELECT に書いた列名あるいは列名を含む式の並びの順序通りになる.

    SELECT の後に列名を書くとき,「<テーブル名>.<列名>」のように,「<テーブル名>.」を前に付けねばならない場合がある. これは, **FROM に指定するテーブルが2個以上**あり, しかも, これらのテーブルが**同じ属性名の属性**を持つ場合である. 例えば,「科目」と「履修」という2つのテーブルが, **同じ属性名「科目番号」を持つ**場合, 下記のように「科目番号」の前に「科目.」を付けるなどして, あいまいさを排除する必要がある.

```
SELECT 科目.科目番号, 科目名
FROM 科目, 履修
WHERE 科目.科目番号 = 履修.科目番号 AND 履修.学生番号 = 00001;
```

    列名として「*」あるいは「<テーブル名>.*」のような書き方ができ, これは、**全ての属性名を並べて書いたのと同じ意味**を持つ.

  - **FROM <list-of(table-name)>**

    FROM の後には, テーブル名の1個以上の並びを書く. 2個以上ある場合には**半角のカンマ「,」で区切る**.

    テーブル参照リストの中で, タップル変数を定義することもできる. 例えば, 下記の SQL 文では X と Y の 2つのタップル変数が定義されている.

```
SELECT X.社員番号, Y.社員番号
FROM 社員 X, 社員 Y
WHERE X.部長 = Y.社員番号 AND X.給与 > Y.給与
```

  - **WHERE <expression>**

    WHERE の後には**探索条件**を書く. 複数の探索条件を組み合わせたい場合には, (半角のカンマではなく)論理演算の AND, OR を用いる. 探索条件は, FROM で指定したテーブルの属性名や, FROM で指

定したタップル変数の変数名を含む式である. 探索条件に使用できるキーワードとしては次のものがある.

- 比較演算(<, <=, =, >=, >, <>)
- 論理演算(AND, OR, NOT)
- 各種の述語(BETWEEN, IN, LIKE, NULL, EXIST など)

など

- **θ-結合演算**

  R(A1, A2, …, An) とS(B1, B2, …, Bm)をリレーションとする. Rの属性 Ai と S の属性 Bj 上の**θ-結合演算**を「**R[AiθBj]S**」と書く. 定義は次の通りである. (θ は比較演算子である).

  R[AiθBj]S は, **R と S の直積集合 (R×S) の中から「R.AiθS.Bj」を満足する要素を選んだもの**

  θ-結合演算「R[AiθBj]S」を SQL を用いて書くと次の通りである.

```
SELECT *
FROM R, S
WHERE <AiθBj>
```

  ※ (参考)リレーショナル代数式を使って次のように書くことができる.

$$R[A_i \theta B_j]S = \{ (t, u) \mid t \in R \wedge u \in S \wedge t[A_i] \theta u[B_j] \}$$

  但し, 「(t, u)」と書いているのは, t = (a1, a2, …, an), u = (b1, b2, …, bm) とするときに, (t, u) = (a1, a2, …, an, b1, b2, …, bm) なる n + m 項のタップルである.

- **結合問い合わせ**(結合質問ともいう)

  結合問い合わせとは, 問い合わせの中にθ-結合演算を含むような問い合わせのことである. 結合問い合わせでは, **SQL のテーブル参照リストに, 2つ以上のテーブル名が現れる**. 一方で, 結合問い合わせの評価結果は, 一般の問い合わせと同様に1つのテーブルであることに注意して欲しい.

- **テキスト・エンコーディング** (text encoding)

  データベースには, 文字データはエンコード (encoding) されて格納されている. エンコードの手法にはいくつかの種類がある. SQLite バージョン 3 では, エンコードとして次のいずれかを指定できる.

  - UTF-8
  - big-endian UTF-16
  - little-endian UTF-16

演習

演習で行うこと

- SQLiteman の起動と終了 (Start and end SQLiteman)
- SQLiteman で新しいデータベースを作成する (Create a new database)
- SQLiteman で既存のデータベースを開く (Open an existing database using SQLiteman)
- SQL を用いたテーブル定義と一貫性制約の記述 (Table defintion and integrity constraint specification using SQL)

  リレーショナル・スキーマ (relational schema): **products(id, product_name, type, cost, created_at)**

  SQL プログラム:

```
CREATE TABLE products (
    id           INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    product_name TEXT     UNIQUE NOT NULL,
    type         TEXT     NOT NULL,
    cost         REAL,
    created_at   DATETIME NOT NULL );
```

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

```
BEGIN TRANSACTION;
INSERT INTO products VALUES( 1, 'Fukuoka apple',     'apple',  50, datetime('now') );
INSERT INTO products VALUES( 2, 'Kumamoto orange L', 'orange', 30, datetime('now') );
INSERT INTO products VALUES( 3, 'Kumamoto orange M', 'orange', 20, datetime('now') );
INSERT INTO products VALUES( 4, 'Fukuoka melon',     'melon',  NULL, datetime('now') );
COMMIT;
```

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

```
SELECT * FROM products;
SELECT * FROM products WHERE type = 'orange';
SELECT * FROM products WHERE cost > 25;
```

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

  リレーショナル・スキーマ (relational schema):

requests(id, year, month, day, product_id, qty, created_at)

SQL プログラム:

```
CREATE TABLE requests (
    id         INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    year       INTEGER  NOT NULL CHECK ( year > 2008 ),
    month      INTEGER  NOT NULL CHECK ( month >= 1 AND month <= 12 ),
    day        INTEGER  NOT NULL CHECK ( day >= 1 AND day <= 31 ),
    product_id INTEGER  NOT NULL REFERENCES products(id),
    qty        INTEGER  NOT NULL,
    created_at DATETIME NOT NULL );
```

- SQL を用いたテーブルへの行の挿入 (Insert rows into a table using SQL)
- SQLiteman を用いたデータのブラウズ (Browse Data using SQLiteman)
- 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 を用いたテーブル定義と一貫性制約の記述 (Table defintion and integrity constraint specification using SQL)

  リレーショナル・スキーマ (relational schema, created_at):

  bundles(id, request_id, qty, shipping_id)

  shippings(id, year, month, day, created_at)

  SQL プログラム:

```
CREATE TABLE bundles (
    id          INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    request_id  INTEGER  NOT NULL REFERENCES requests(id),
    qty         INTEGER  NOT NULL,
    shipping_id INTEGER  NOT NULL REFERENCES shippings(id),
    created_at DATETIME NOT NULL );

CREATE TABLE shippings (
    id    INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    year  INTEGER  NOT NULL CHECK ( year > 2008 ),
    month INTEGER  NOT NULL CHECK ( month >= 1 AND month <= 12 ),
    day   INTEGER  NOT NULL CHECK ( day >= 1 AND day <= 31 ),
    created_at DATETIME NOT NULL );
```

- SQL を用いたテーブルへの行の挿入 (Insert rows into a table using SQL)
- SQLiteman を用いたデータのブラウズ (Browse Data using SQLiteman)
- SQL 問い合わせの発行と評価結果の確認 (Issue SQL queries and inspect the results)

演習を行うために必要になる機能や文法

SQLite の SQL の説明は http://www.hwaci.com/sw/sqlite/lang.html (English Web Page) にある.

- SQL 問い合わせ (SQL query)
  - SELECT <**list-of(result-column)**> FROM <**list-of(table-name)**>

    table-name で指定したテーブルの直積集合から, **list-of(result-column) で指定した属性を抽出したり, 式を評価して**新しいテーブルを 1つ作り出力する.
  - SELECT <**list-of(result-column)**> FROM <**list-of(table-name)**> WHERE <**expression**>

    table-name で指定したテーブルの直積集合から, **<expression> を満足する行のみを選び**, そうして出来たテーブルから, list-of(result-column) で指定した属性を抽出したり, 式を評価して新しいテーブルを1つ作り出力する.

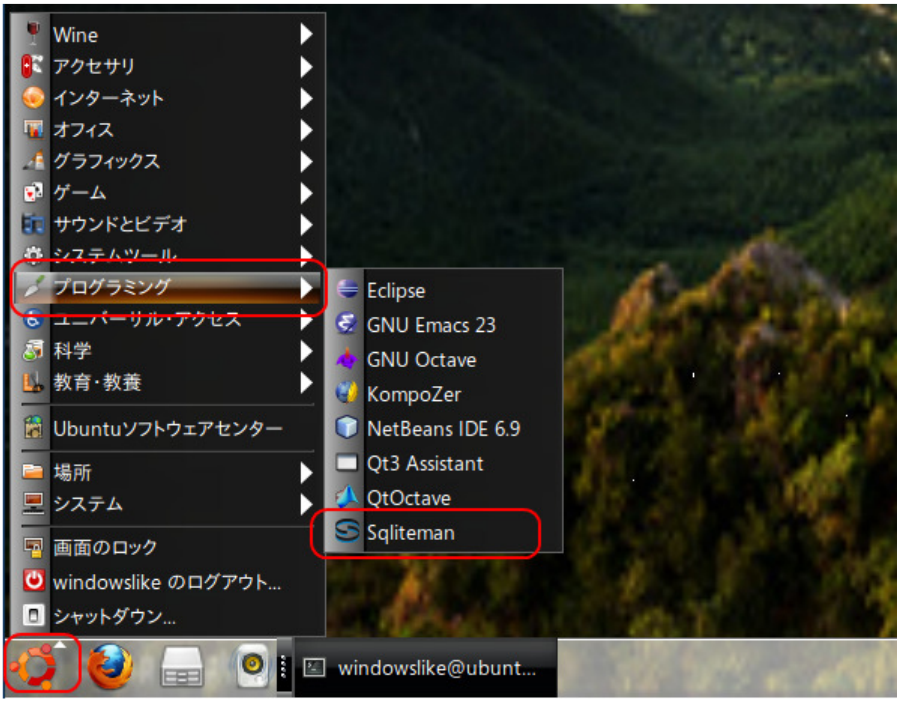
SQLiteman の起動と終了 (Start and end SQLiteman)

CREATE TABLE などの SQL コマンドを編集できるエディタの機能等をもったソフトウエアとshちえ SQLiteman を使うことにします. (We use the SQLiteman as SQL editor, database manager interface, ...)

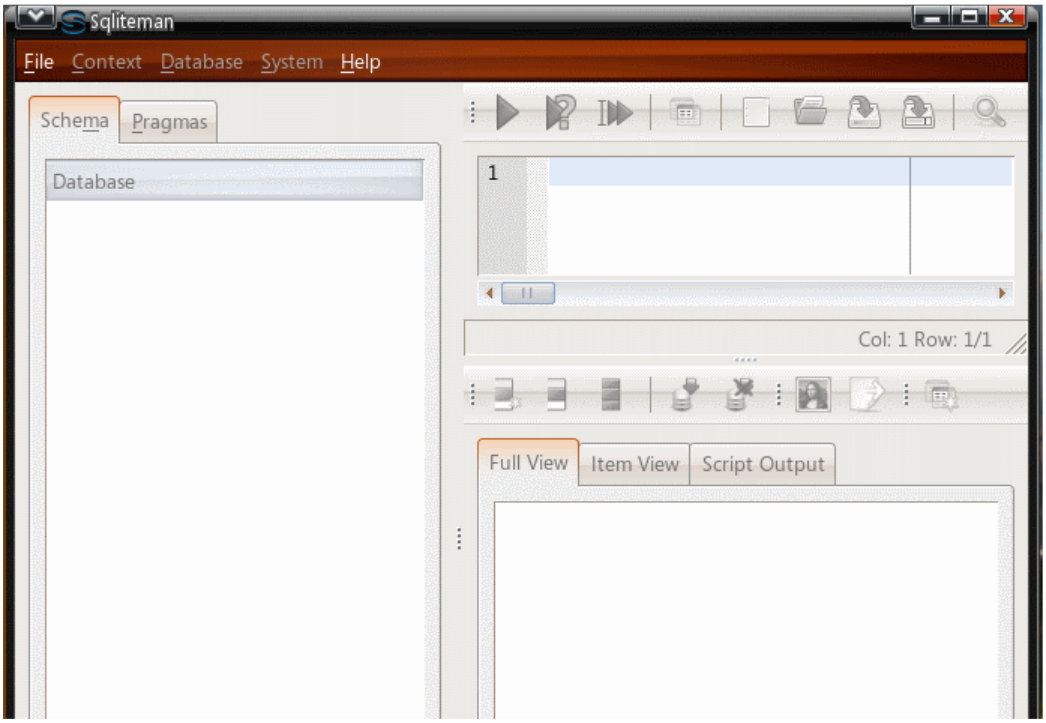
1. **SQLiteman の起動** (Start SQLiteman)

   ■ **Ubuntu での SQLiteman の起動例**

   「**プログラミング**」→「**Sqliteman**」と操作する.



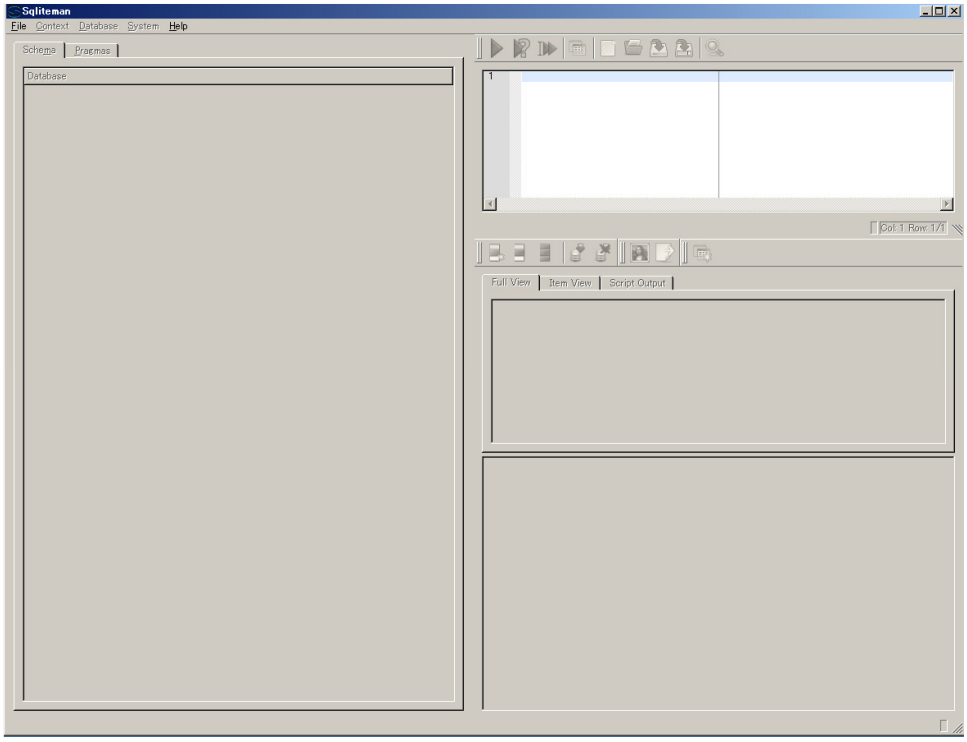
SQLiteman の新しいウインドウが開く. (A New window appears)



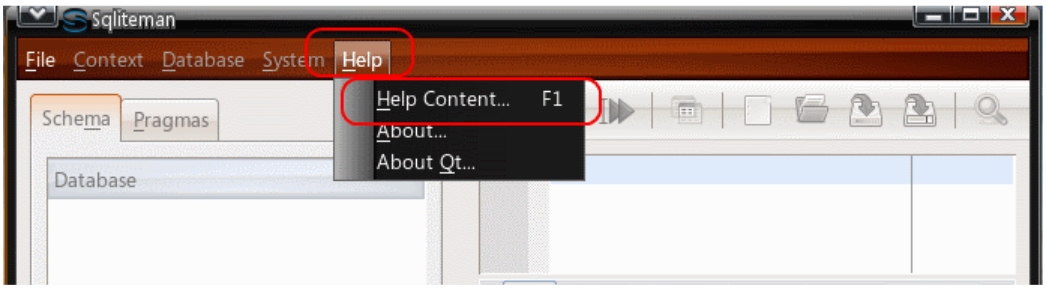
■ **Windows での SQLiteman の起動例**

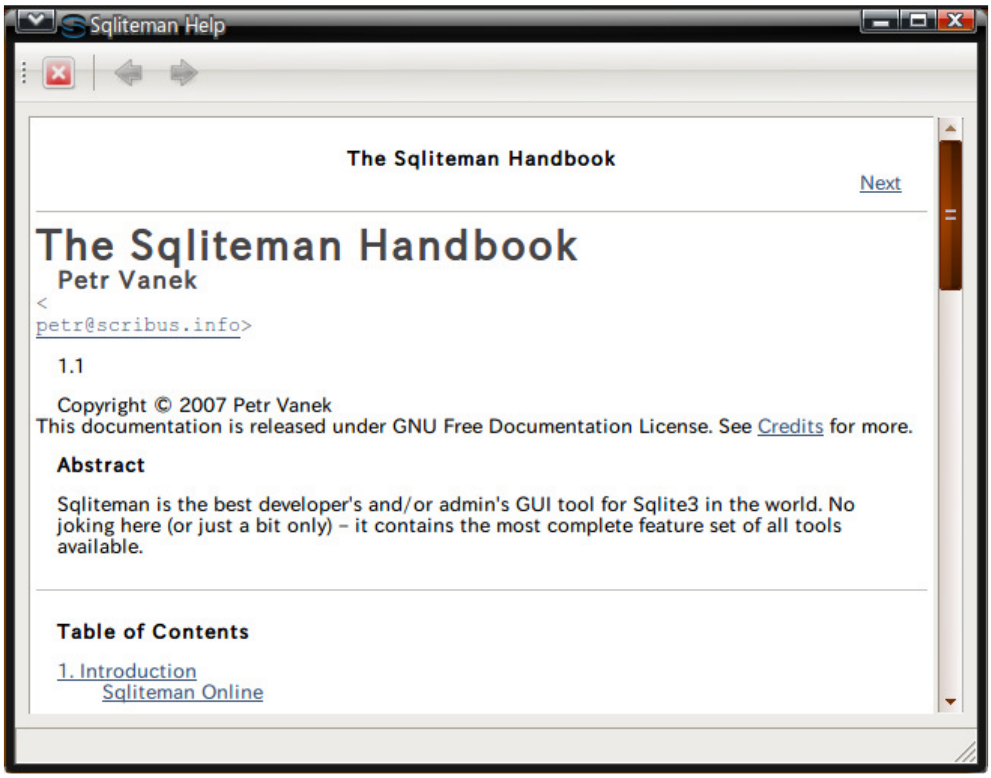
「SQLiteman」のアイコンをダブルクリック (double click "SQLiteman.exe")

SQLiteman の新しいウインドウが開く. (A New window appears)

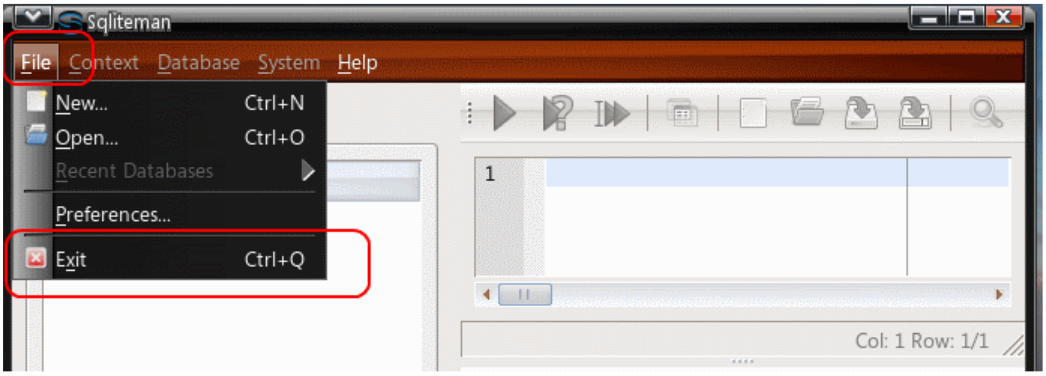


2. **ヘルプの表示** (Help Content)
   「**Help**」→「**Help Content**」





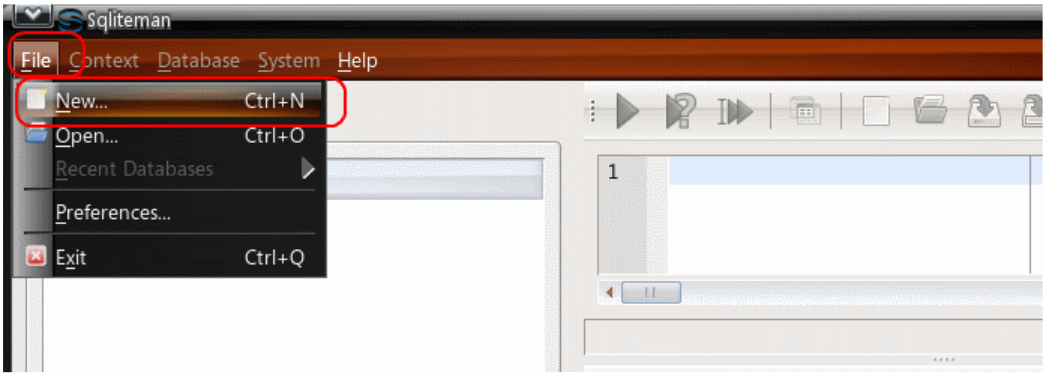
3. **終了** (End SQLiteman)
   「**File**」→「**Exit**」で終了.



SQLiteman で新しいデータベースを作成する (Create a new database)

以下の手順で，**新しいデータベースを作成する**．その結果，データベースファイルができる．(Create a new database)

1. 「**File**」→「**New**」

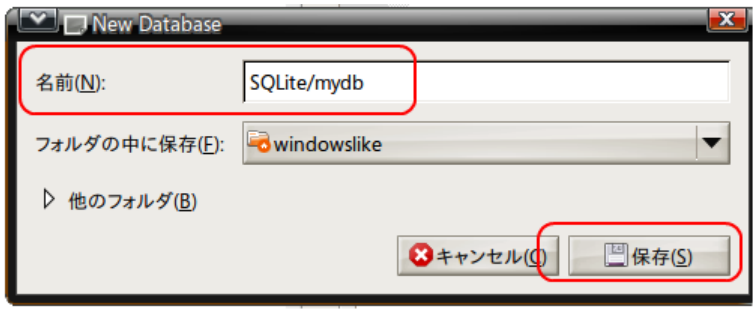


2. **データベースの新規作成を開始する** (Start a new databae creation)

   ■ **Ubuntu での実行例**(データベースファイル名を「**SQLite/mydb**」にする場合)

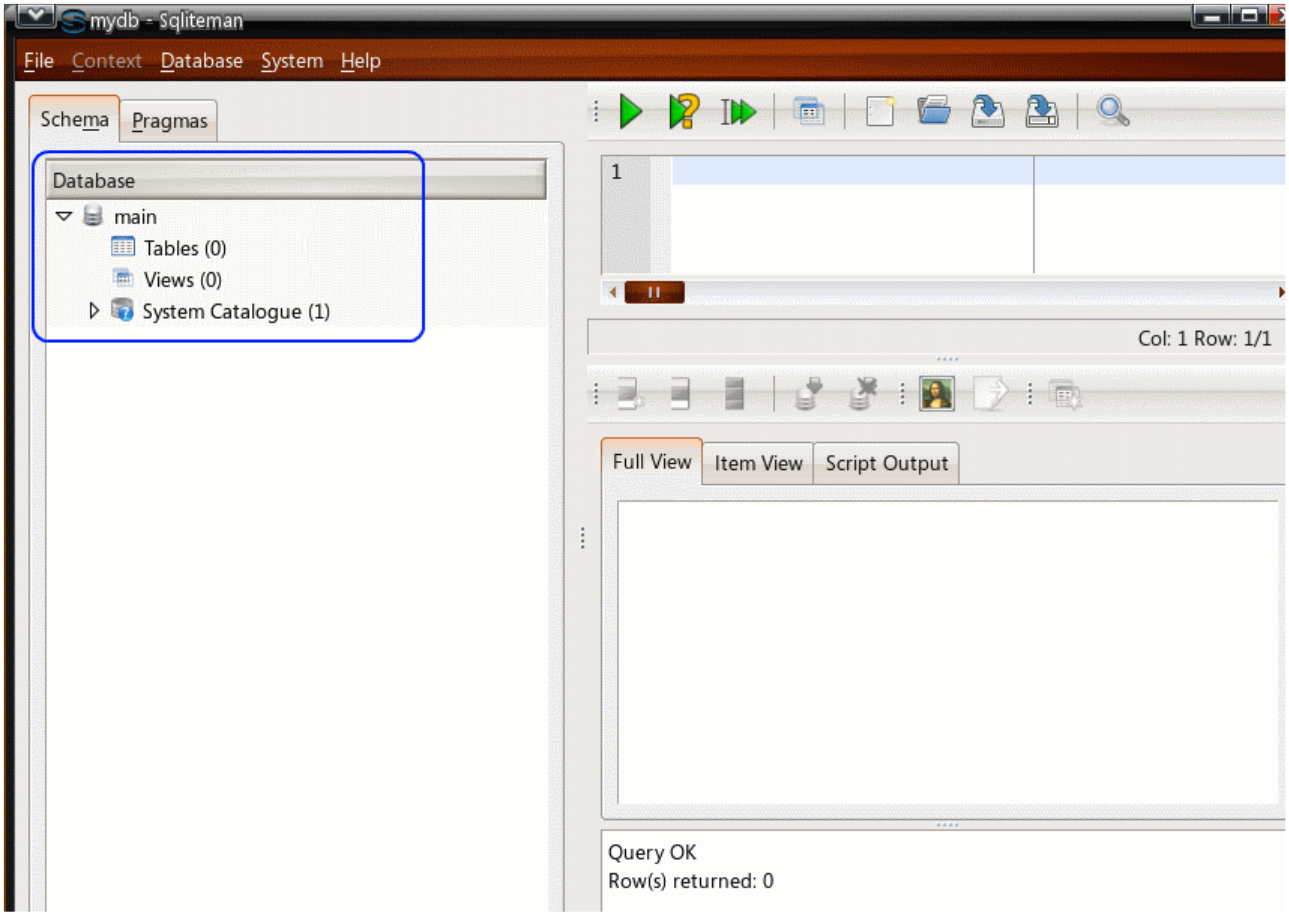
   データベースファイル名**SQLite/mydb** を指定し，「保存」をクリック (Click '保存' after writing a database file name)

   データベースを新規作成したいときは，データベースファイル名として「新しい」ものを指定すること

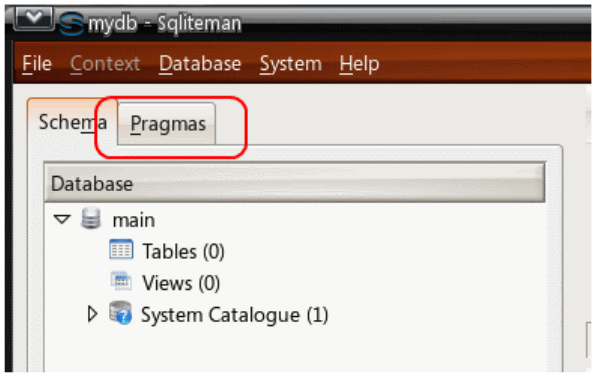


3. **データベースの中身が表示されるので確認する** (Database appears)

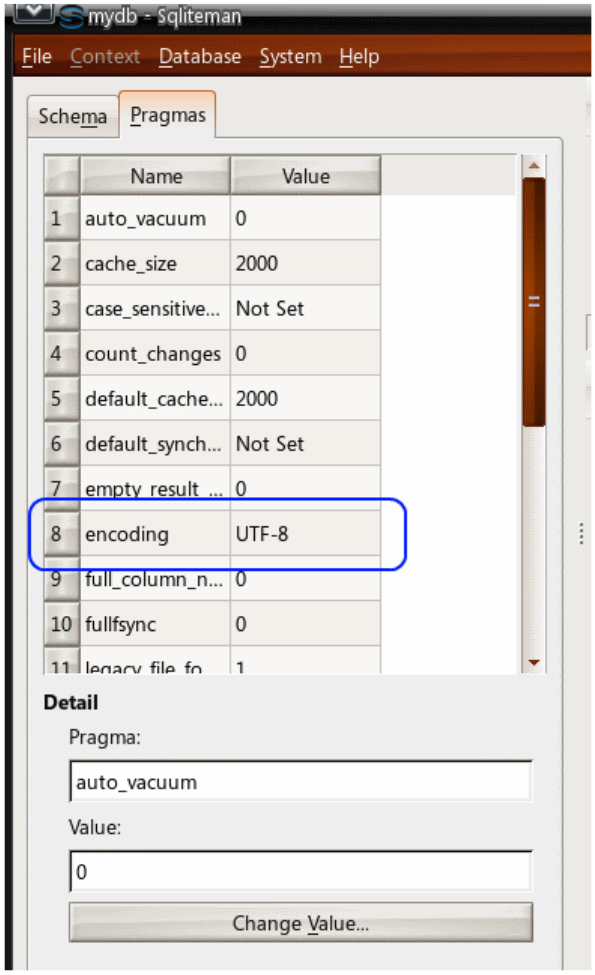
   このときテーブル (Tables) 数も，ビュー (Views) の数も 0 である．



4. **テキスト･エンコーディングの設定を確認**する (text encoding)
   まず,「**Pragmas**」をクリック. (Click 'Pragmas')



encodingの行に「**UTF-8**」のように表示されている.

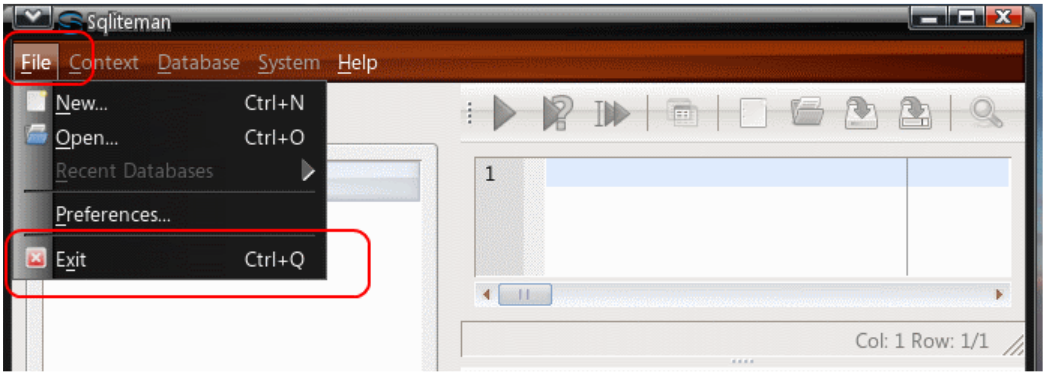


※ もし，**データベースの文字のエンコーディングを変えたい**ときは，SQLiteman のようなグラフィカルなツールを使うのではなく，sqlite.exe を起動し「**PRAGMA encoding=...;**」で変える方がずっと簡単でしょう．例えば「UTF-16le」などに変えたいなど．

5. **終了** (End SQLiteman)

あとの混乱を防ぎたいので、1度、SQLiteman を終了する

「**File**」→「**Exit**」で終了．

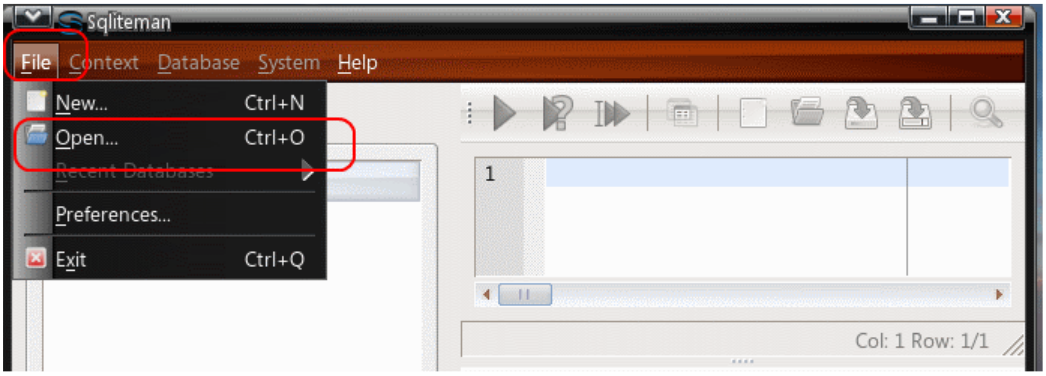


SQLiteman で既存のデータベースを開く (Open an existing database using SQLiteman)

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)

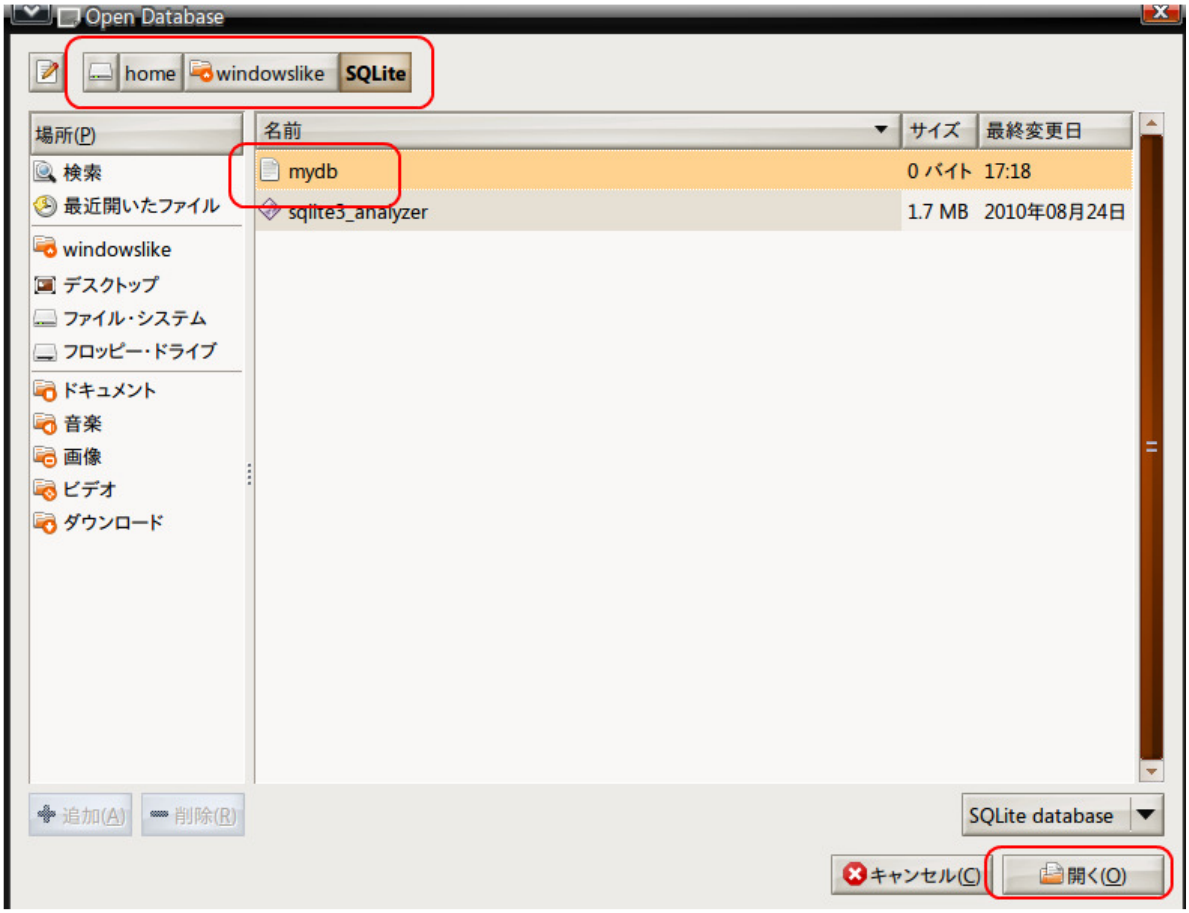
1. 「**File**」→「**Open**」



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)

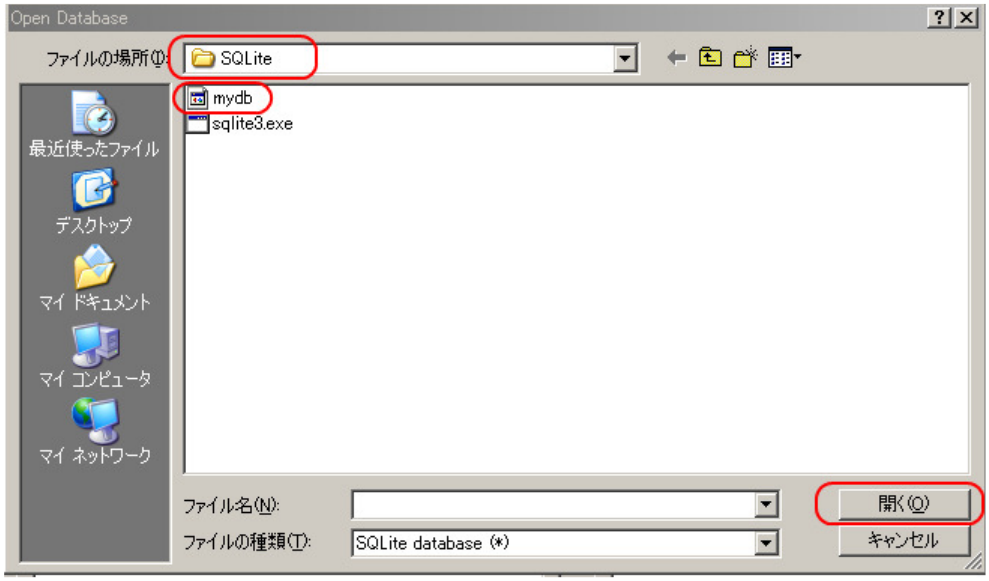
データベースファイル **SQLite/mydb** を選び,「開く」をクリック (Click '開く' after choosing the database file "SQLite/mydb")



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

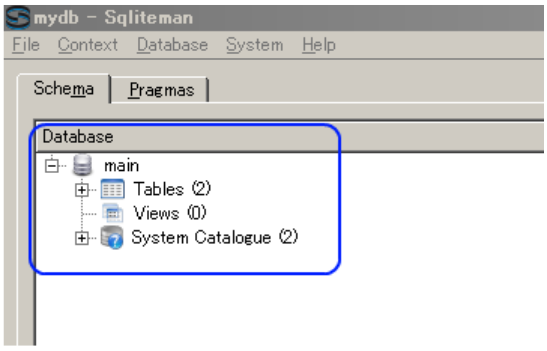
データベースファイル **C:¥SQLite¥mydb** を選び,「開く」をクリック (Click '開く' after choosing the database file "C:¥SQLite¥mydb")

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)

◆ 表示例(データベースの中身によって表示が変わる)

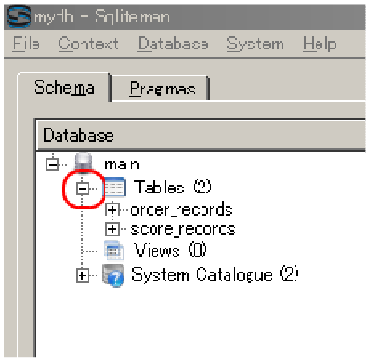


4. **「Tables」の数字が1以上の場合には展開できる**

「Tables」の数字が1以上のとき,「Tables」を展開すると, テーブルの一覧 (List of Tables) が表示される

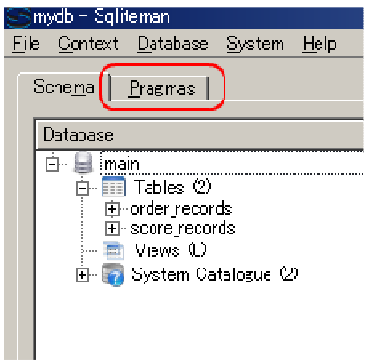
ので確認する (List of tables appears by clicking 'Tables')

◆ 展開の例

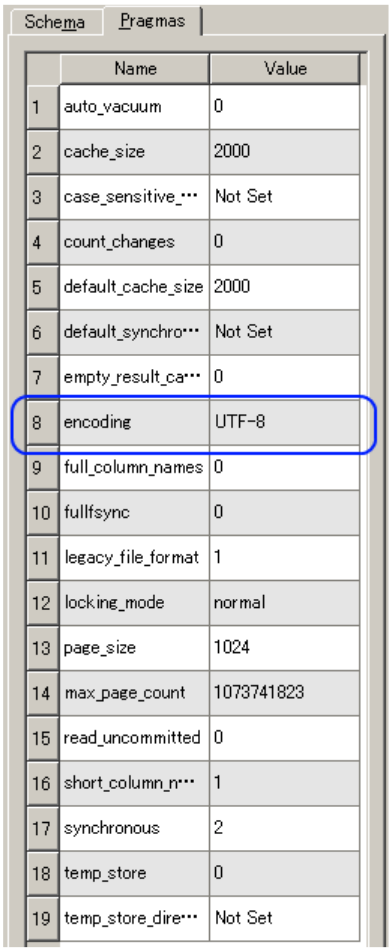


5. **テキスト・エンコーディングの設定を確認**する (text encoding)

まず,「**Pragmas**」をクリック. (Click 'Pragmas')



encodingの行に「**UTF-8**」のように表示されている.



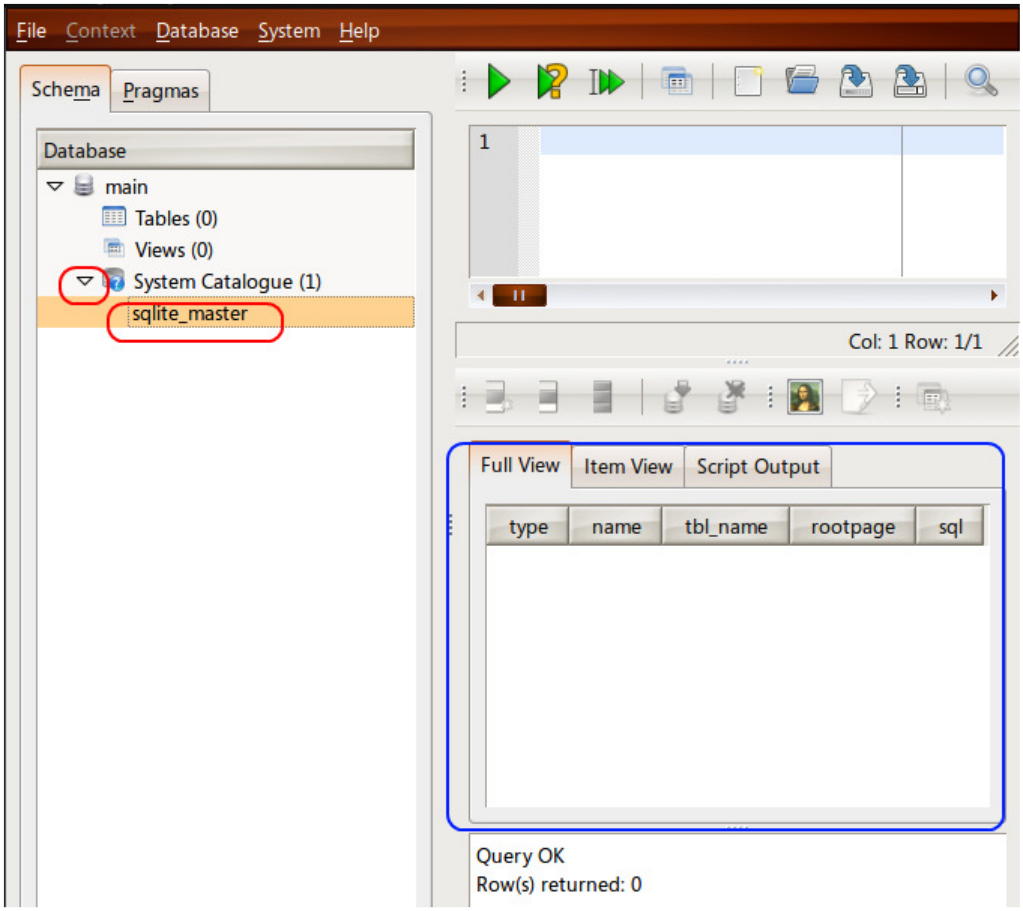
	Name	Value
1	auto_vacuum	0
2	cache_size	2000
3	case_sensitive_…	Not Set
4	count_changes	0
5	default_cache_size	2000
6	default_synchro…	Not Set
7	empty_result_ca…	0
8	encoding	UTF-8
9	full_column_names	0
10	fullfsync	0
11	legacy_file_format	1
12	locking_mode	normal
13	page_size	1024
14	max_page_count	1073741823
15	read_uncommitted	0
16	short_column_n…	1
17	synchronous	2
18	temp_store	0
19	temp_store_dire…	Not Set

※ もし, **データベースの文字のエンコーディングを変えたい**ときは, SQLiteman のようなグラフィカルなツールを使うのではなく, sqlite.exe を起動し「PRAGMA encoding=...;」で変える方がずっと簡単でしょう. 例えば「UTF-16le」などに変えたいなど.

6. **「System Catalogue」を展開し, 「sqlite_master」をクリックすると, データベース・スキーマ (database schema) が表示される**ので確認する (Database schema appears by clicking 'sqlite_master')

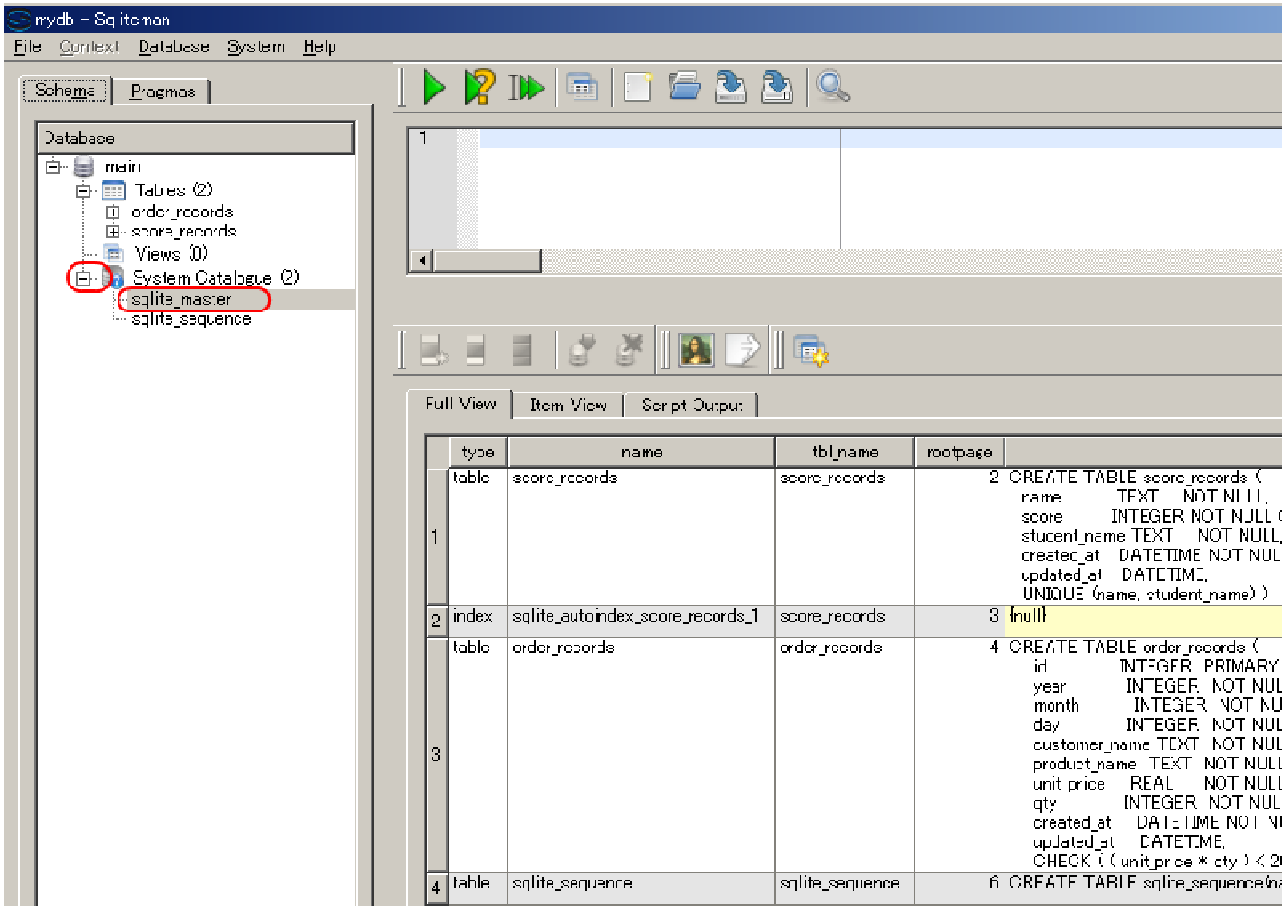
◆ 表示の例(1)

データベースが空の場合, 表示も空.



◆ 表示の例(2)

score_records, order_records などのテーブルを定義すみの場合



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

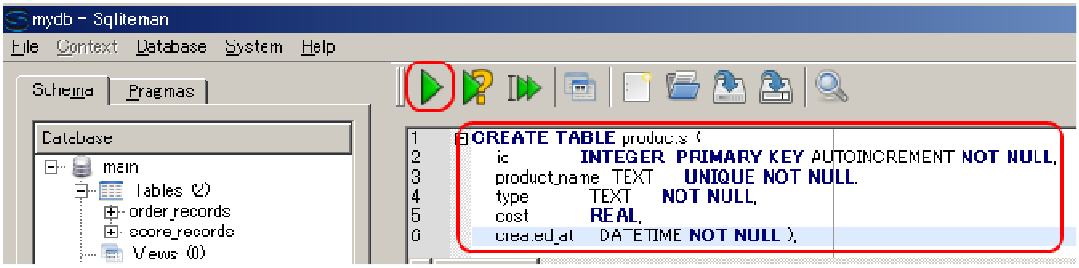
リレーショナル・スキーマ (relational schema): **products(id, product_name, type, cost, created_at)**

1. **products 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).

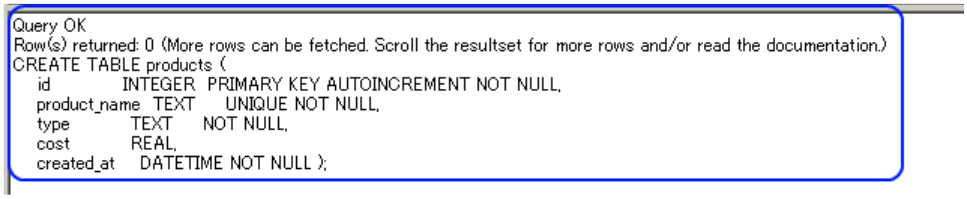
```
CREATE TABLE products (
    id           INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    product_name TEXT     UNIQUE NOT NULL,
    type         TEXT     NOT NULL,
    cost         REAL,
    created_at   DATETIME NOT NULL );
```

   ※「SQL Editor」のウインドウには, SQL プログラムを書くことができる. In the '**SQL Editor**' window, you can write down **SQL program(s)**.



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)

次のような **products** テーブルを作る. (Construct table 'products')

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

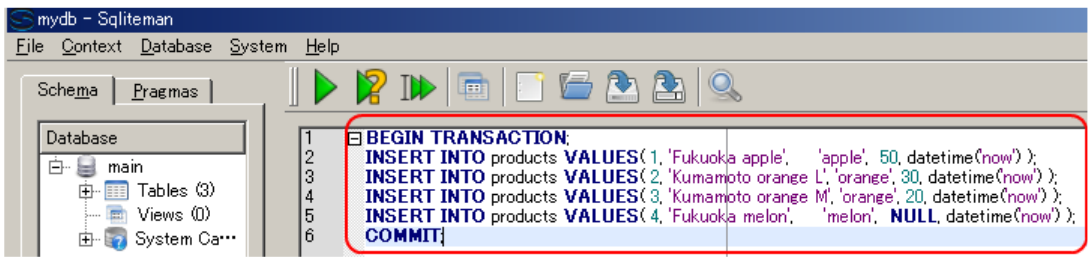
以下の手順で, SQL を用いて **products テーブルへの行の挿入**を行う (Insert rows into table 'products' using SQL)

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).

   ※ つまり, 挿入の前に **BEGIN TRANSACTION;** を実行し, 一連の挿入が終わったら **COMMIT;** を実行する. (Issue "BEGIN TRANSACTION" before database update and "COMMIT" after database update).

```
BEGIN TRANSACTION;
INSERT INTO products VALUES( 1, 'Fukuoka apple',      'apple',  50, datetime('now') );
INSERT INTO products VALUES( 2, 'Kumamoto orange L', 'orange', 30, datetime('now') );
INSERT INTO products VALUES( 3, 'Kumamoto orange M', 'orange', 20, datetime('now') );
INSERT INTO products VALUES( 4, 'Fukuoka melon',      'melon',  NULL, datetime('now') );
COMMIT;
```



   INSERT INTO には 2つの方法がある. (Two styles of "INSERT INTO")

   ■ 属性の値を, テーブル定義の順に全て並べる (List all attribute values. The order is the same as its table definition)

```
INSERT INTO products VALUES( 1, 'Fukuoka apple',      'apple',  50, datetime('now') );
```

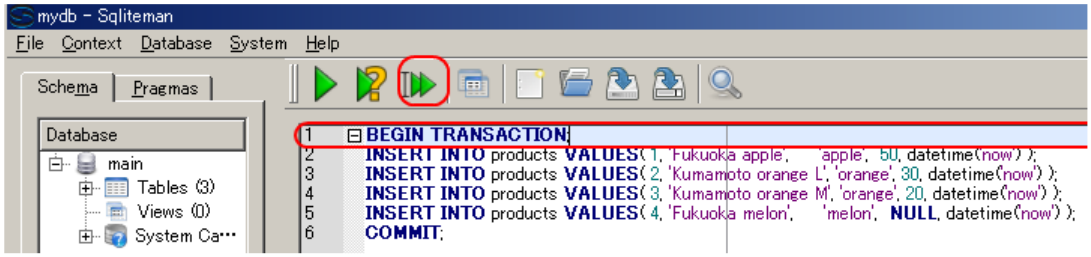
■ 属性の値の並び方を，属性名を使って明示的に指定する (Specify the order of attribute values using attribute name list)

このとき，属性値を省略すると，テーブル定義のときに指定されたデフォルト値が使われる (defaults values are used)

```
INSERT INTO products VALUES( 2, 'Kumamoto orange L', 'orange', 30, datetime('now') );
```

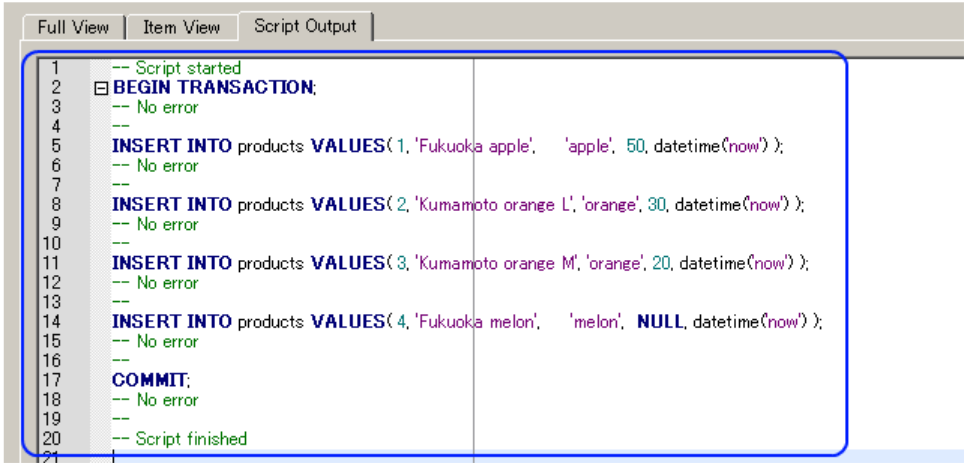
2. **複数の SQL 文の一括実行** (Execute multiple SQL statements)

   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

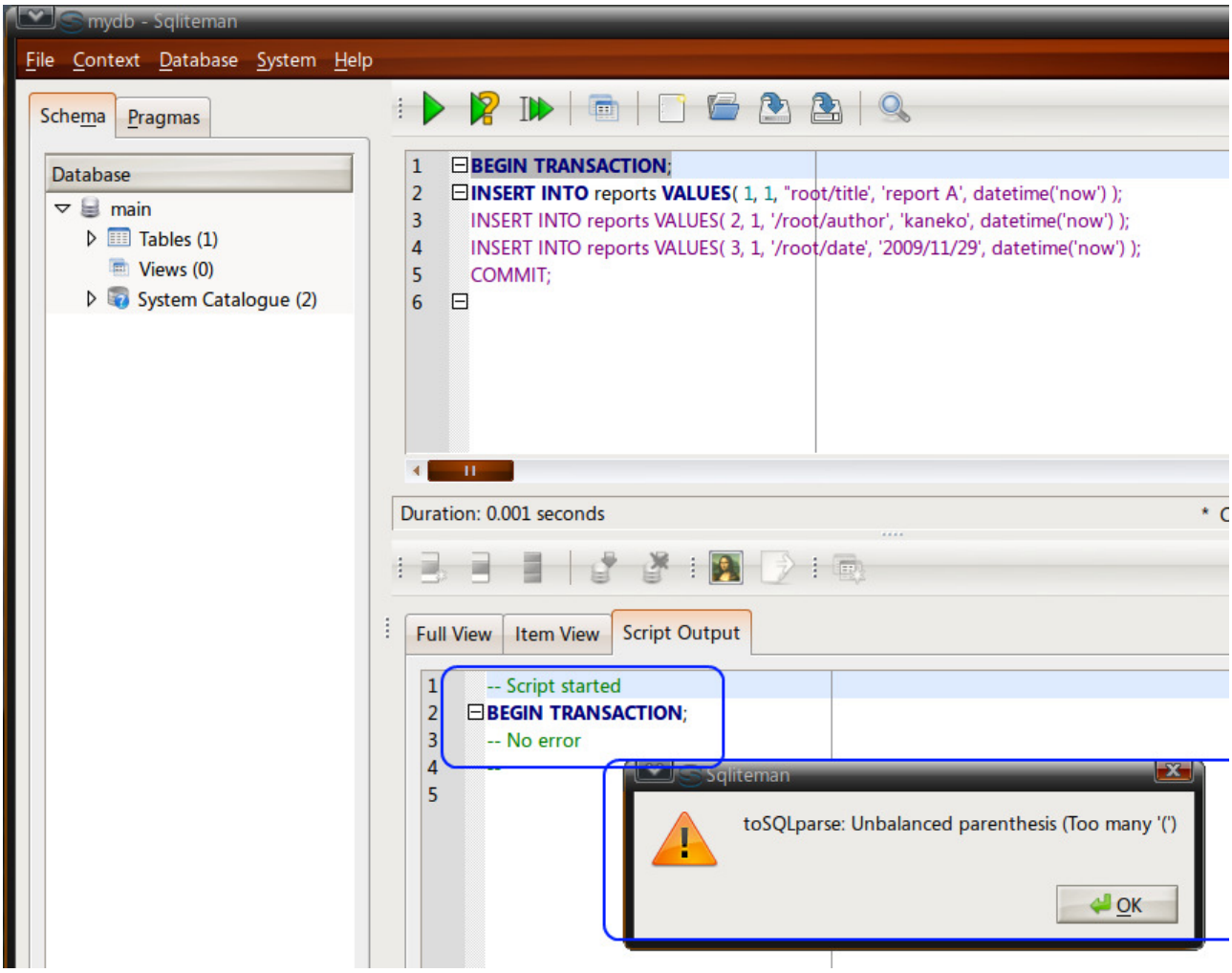


※

エラーメッセージが出ているときは，テーブル定義を書き直して，「Run multiple SQL statements ...」のボタンをクリックする．

例えば，下記のようにエラーが出ていたとする．このときは，

- **「BEGIN TRANSACTION」は終わっている．**
- それ以降は実行されていない



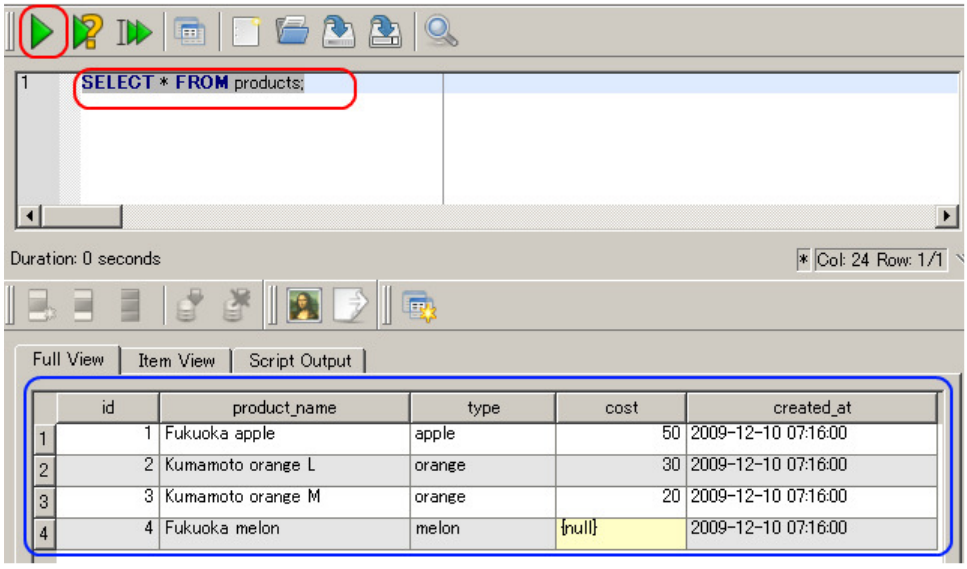
このようなときは，再開したい行にカーソルをあわせて、「Run multiple SQL statements ...」のボタンをクリックする．

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

ここでは，SQL を用いた**問い合わせ**の実行例を示す．SQL 問い合わせの詳細については，別の Web ページで説明する．ここでは，テーブルの中身を確認して欲しい．

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

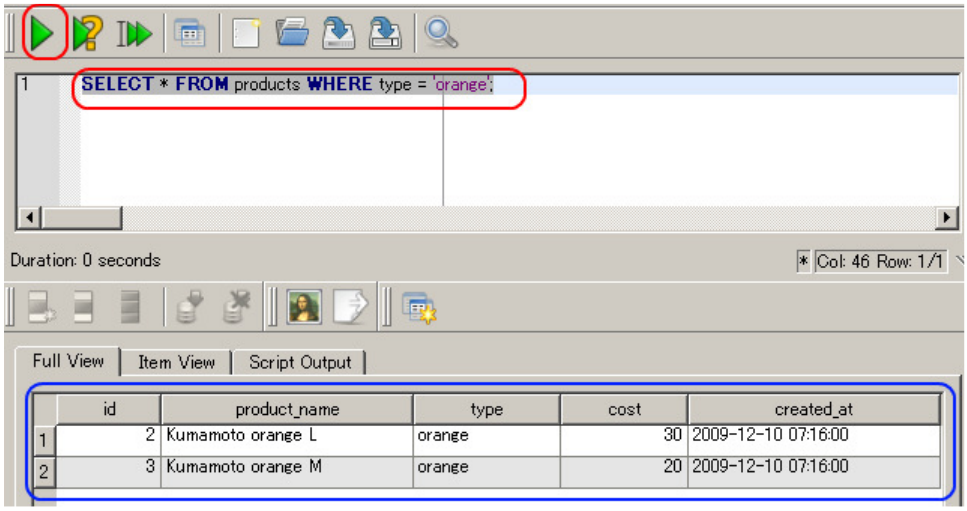
```
SELECT * FROM products;
```



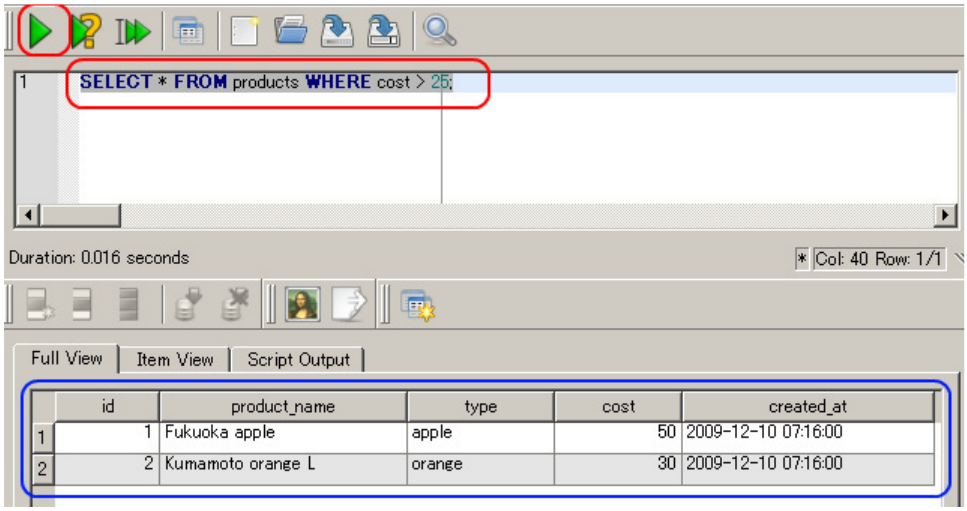
	id	product_name	type	cost	created_at
1	1	Fukuoka apple	apple	50	2009-12-10 07:16:00
2	2	Kumamoto orange L	orange	30	2009-12-10 07:16:00
3	3	Kumamoto orange M	orange	20	2009-12-10 07:16:00
4	4	Fukuoka melon	melon	[null]	2009-12-10 07:16:00

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

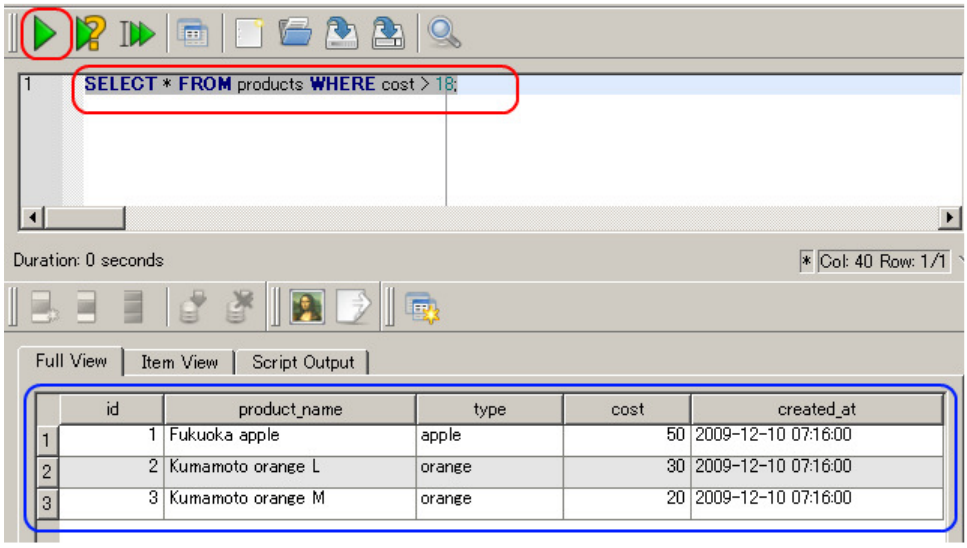
```
SELECT * FROM products WHERE type = 'orange';
```



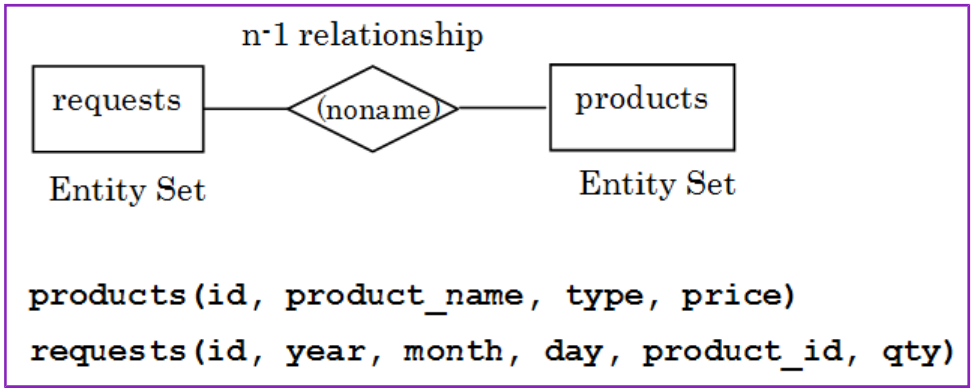

```
SELECT * FROM products WHERE cost > 25;
```




```
SELECT * FROM products WHERE cost > 18;
```



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



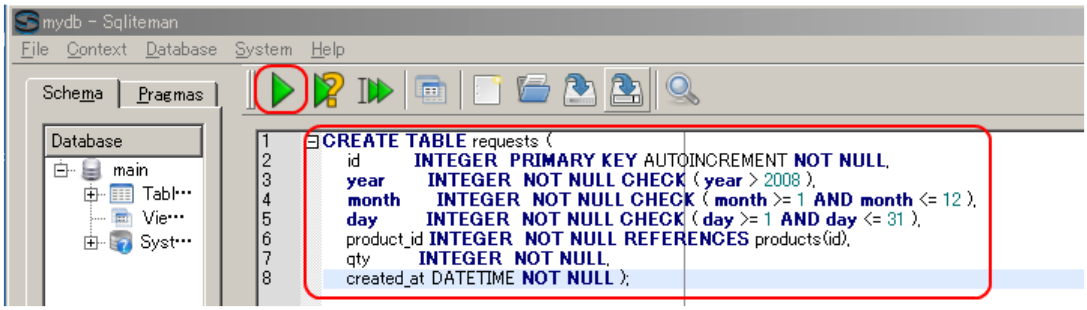
実体関連図 (Entity Relationship Diagram)

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

1. **products テーブルの定義** (Define a table)

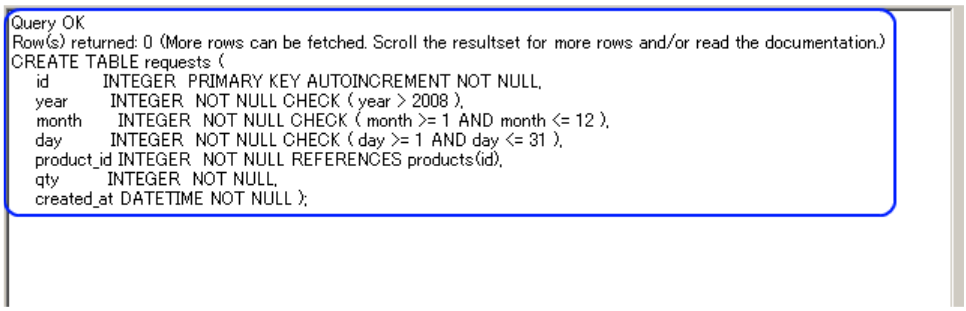
   次の SQL を入力し，「Run SQL」のアイコンをクリック (Write the following SQL, and click "Run SQL" icon).

```
CREATE TABLE requests (
    id          INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    year        INTEGER  NOT NULL CHECK ( year > 2008 ),
    month       INTEGER  NOT NULL CHECK ( month >= 1 AND month <= 12 ),
    day         INTEGER  NOT NULL CHECK ( day >= 1 AND day <= 31 ),
    product_id INTEGER  NOT NULL REFERENCES products(id),
    qty         INTEGER  NOT NULL,
    created_at DATETIME NOT NULL );
```



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)

次のような **requests** テーブルを作る．(Construct table 'requests')

requests

id	year	month	day	product id	qty
1001	2009	10	28	1	3
1002	2009	11	1	2	1
1003	2009	11	2	1	2
1004	2009	11	2	3	4

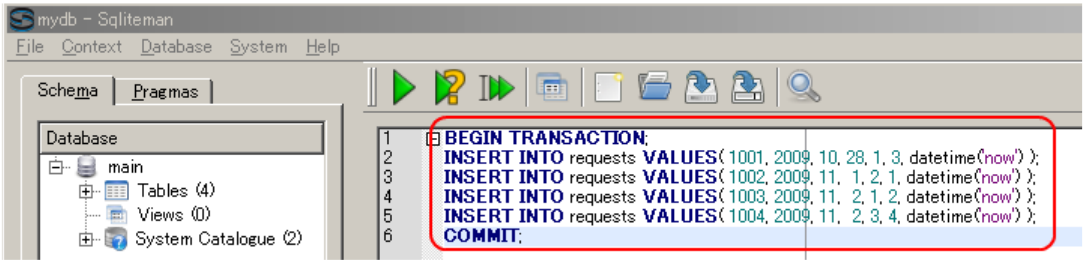
以下の手順で，SQL を用いて **requests テーブルへの行の挿入**を行う (Insert rows into table 'requests' using SQL)

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).

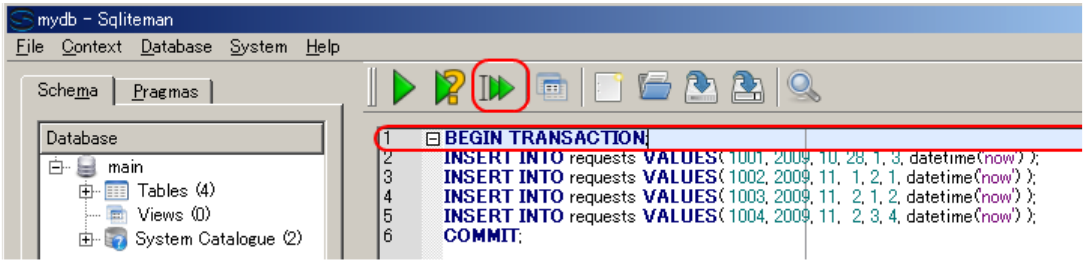
```
BEGIN TRANSACTION;
INSERT INTO requests VALUES( 1001, 2009, 10, 28, 1, 3, datetime('now') );
INSERT INTO requests VALUES( 1002, 2009, 11,  1, 2, 1, datetime('now') );
```

```
INSERT INTO requests VALUES( 1003, 2009, 11,  2, 1, 2, datetime('now') );
INSERT INTO requests VALUES( 1004, 2009, 11,  2, 3, 4, datetime('now') );
COMMIT;
```



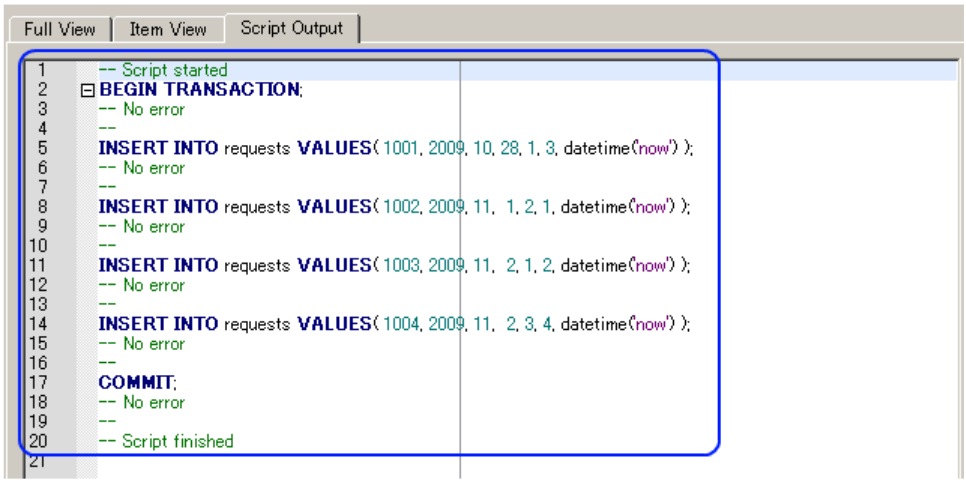
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

   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)

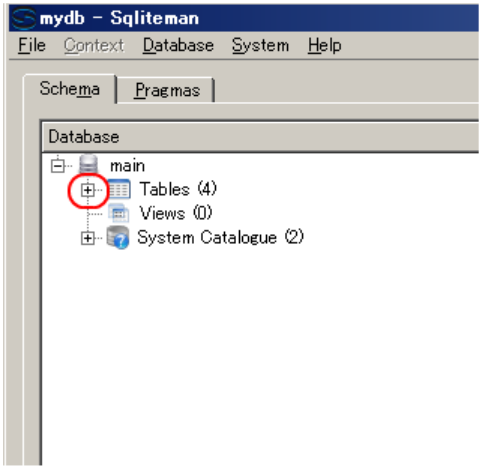
   エラーメッセージが出ていないことを確認



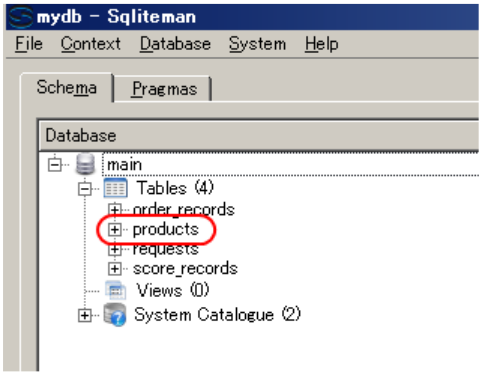
SQLiteman を用いたデータのブラウズ (Browse Data using SQLiteman)

- **products 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')



次に，テーブル **products**を選ぶ (Select table 'products')



テーブル **products**が表示される (table 'products' appears)

Full View | Item View | Script Output

	id	product_name	type	cost	created_at
1	1	Fukuoka apple	apple	50	2009-12-10 07:16:00
2	2	Kumamoto orange L	orange	30	2009-12-10 07:16:00
3	3	Kumamoto orange M	orange	20	2009-12-10 07:16:00
4	4	Fukuoka melon	melon	{null}	2009-12-10 07:16:00

※ 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).

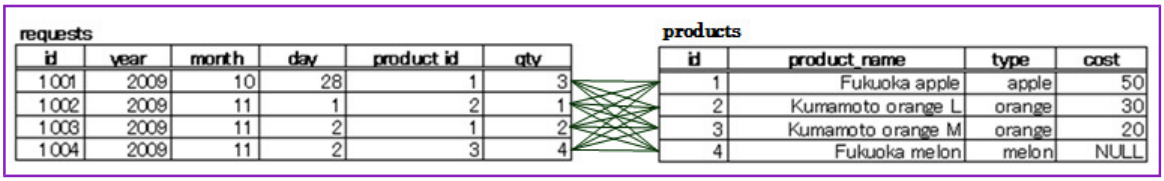
- 今度は, **requests テーブル** を表示

Full View | Item View | Script Output

	id	year	month	day	product_id	qty	created_at
1	1001	2009	10	28	1	3	2009-12-10 08:47:21
2	1002	2009	11	1	2	1	2009-12-10 08:47:21
3	1003	2009	11	2	1	2	2009-12-10 08:47:21
4	1004	2009	11	2	3	4	2009-12-10 08:47:21

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

直積 (Cartesian product)



requests

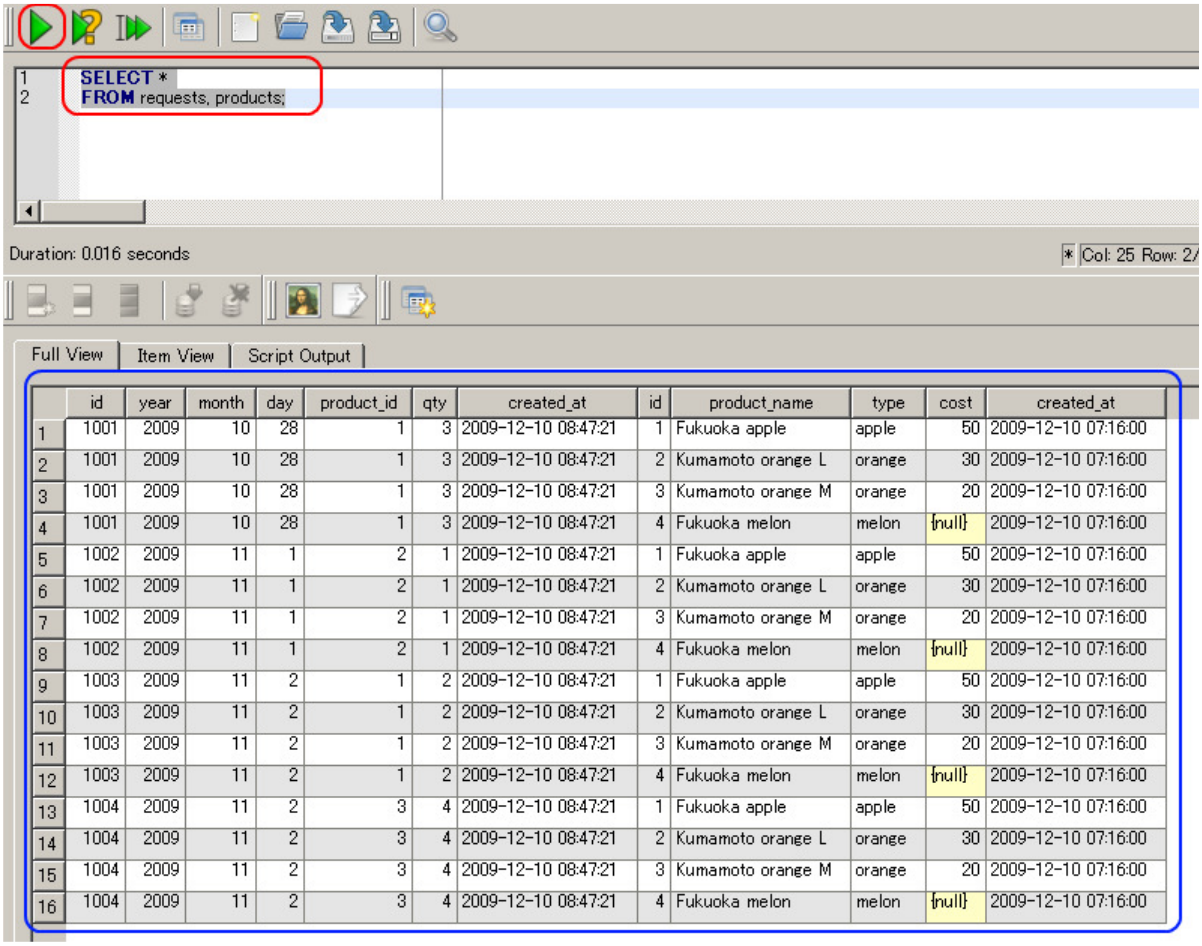
id	year	month	day	product id	qty
1001	2009	10	28	1	3
1002	2009	11	1	2	1
1003	2009	11	2	1	2
1004	2009	11	2	3	4

products

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

SQL を使い, 複数のテーブルの**直積**を簡単に得ることができる.

```
SELECT *
FROM requests, products;
```



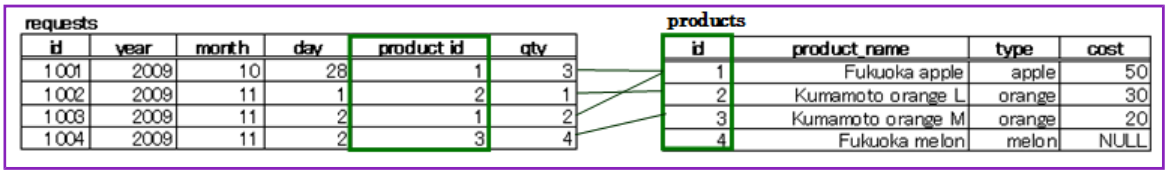
```
SELECT *
FROM requests, products;
```

Duration: 0.016 seconds

Full View | Item View | Script Output

	id	year	month	day	product_id	qty	created_at	id	product_name	type	cost	created_at
1	1001	2009	10	28	1	3	2009-12-10 08:47:21	1	Fukuoka apple	apple	50	2009-12-10 07:16:00
2	1001	2009	10	28	1	3	2009-12-10 08:47:21	2	Kumamoto orange L	orange	30	2009-12-10 07:16:00
3	1001	2009	10	28	1	3	2009-12-10 08:47:21	3	Kumamoto orange M	orange	20	2009-12-10 07:16:00
4	1001	2009	10	28	1	3	2009-12-10 08:47:21	4	Fukuoka melon	melon	{null}	2009-12-10 07:16:00
5	1002	2009	11	1	2	1	2009-12-10 08:47:21	1	Fukuoka apple	apple	50	2009-12-10 07:16:00
6	1002	2009	11	1	2	1	2009-12-10 08:47:21	2	Kumamoto orange L	orange	30	2009-12-10 07:16:00
7	1002	2009	11	1	2	1	2009-12-10 08:47:21	3	Kumamoto orange M	orange	20	2009-12-10 07:16:00
8	1002	2009	11	1	2	1	2009-12-10 08:47:21	4	Fukuoka melon	melon	{null}	2009-12-10 07:16:00
9	1003	2009	11	2	1	2	2009-12-10 08:47:21	1	Fukuoka apple	apple	50	2009-12-10 07:16:00
10	1003	2009	11	2	1	2	2009-12-10 08:47:21	2	Kumamoto orange L	orange	30	2009-12-10 07:16:00
11	1003	2009	11	2	1	2	2009-12-10 08:47:21	3	Kumamoto orange M	orange	20	2009-12-10 07:16:00
12	1003	2009	11	2	1	2	2009-12-10 08:47:21	4	Fukuoka melon	melon	{null}	2009-12-10 07:16:00
13	1004	2009	11	2	3	4	2009-12-10 08:47:21	1	Fukuoka apple	apple	50	2009-12-10 07:16:00
14	1004	2009	11	2	3	4	2009-12-10 08:47:21	2	Kumamoto orange L	orange	30	2009-12-10 07:16:00
15	1004	2009	11	2	3	4	2009-12-10 08:47:21	3	Kumamoto orange M	orange	20	2009-12-10 07:16:00
16	1004	2009	11	2	3	4	2009-12-10 08:47:21	4	Fukuoka melon	melon	{null}	2009-12-10 07:16:00

結合問い合わせ (join query)



requests

id	year	month	day	product id	qty
1001	2009	10	28	1	3
1002	2009	11	1	2	1
1003	2009	11	2	1	2
1004	2009	11	2	3	4

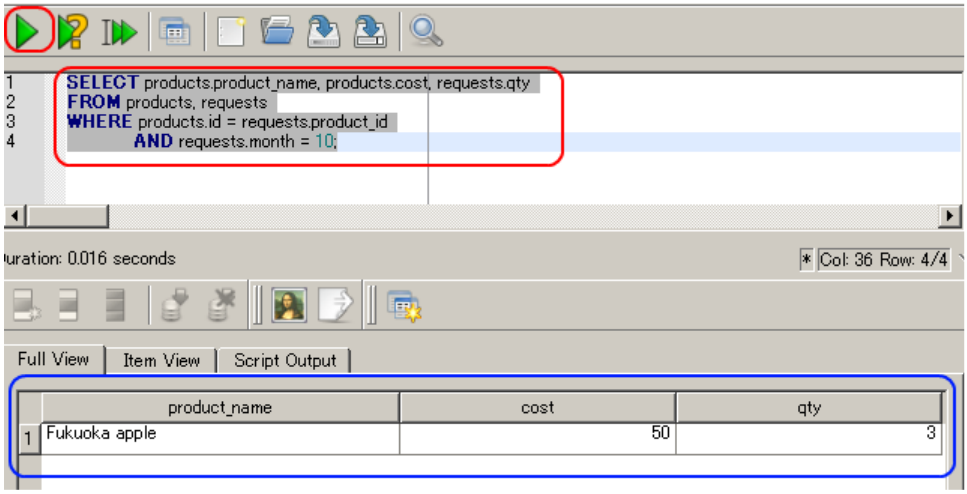
products

id	product_name	type	cost
1	Fukuoka apple	apple	50
2	Kumamoto orange L	orange	30
3	Kumamoto orange M	orange	20
4	Fukuoka melon	melon	NULL

結合問い合わせは，直積から，条件を満足する行を選んだものになる．

List all 'name', 'price' and 'requests.qty' that satisfy "requests.month = 10"

```
SELECT products.product_name, products.cost, requests.qty
FROM products, requests
WHERE products.id = requests.product_id
          AND requests.month = 10;
```



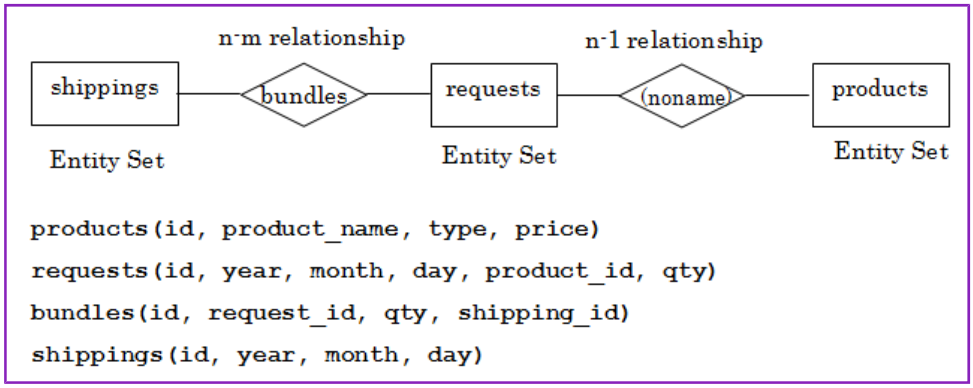
```
SELECT products.product_name, products.cost, requests.qty
FROM products, requests
WHERE products.id = requests.product_id
        AND requests.month = 10;
```

uration: 0.016 seconds

Full View | Item View | Script Output

	product_name	cost	qty
1	Fukuoka apple	50	3

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



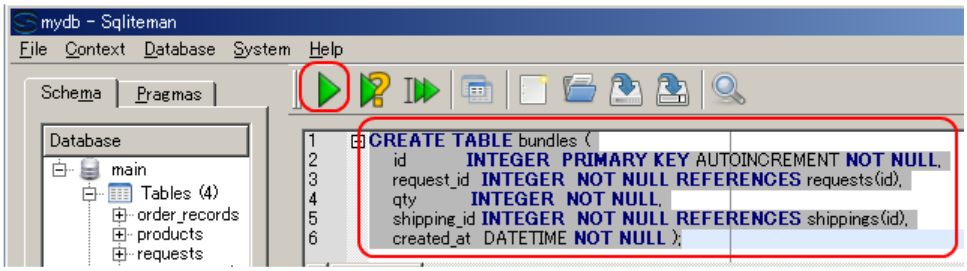
実体関連図 (Entity Relationship Diagram)

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

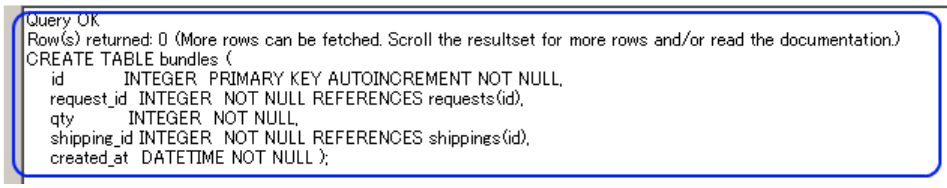
1. **bundles テーブルの定義**

   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 bundles (
    id          INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    request_id  INTEGER  NOT NULL REFERENCES requests(id),
    qty         INTEGER  NOT NULL,
    shipping_id INTEGER  NOT NULL REFERENCES shippings(id),
    created_at  DATETIME NOT NULL );
```



2. **コンソールの確認** (Inspect console)

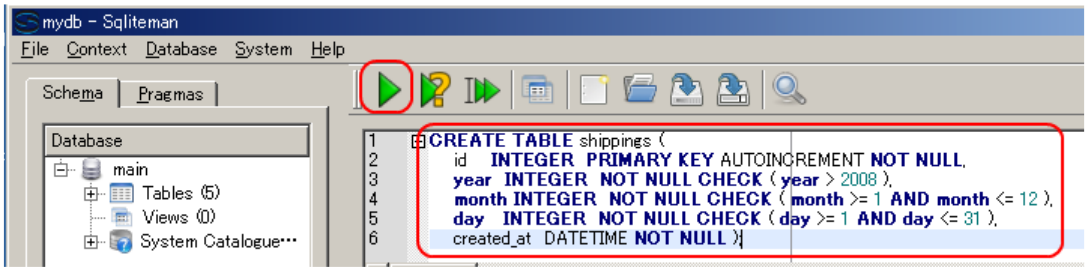


   エラーメッセージが出ていないことを確認

3. **shippings テーブルの定義**

   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 shippings (
    id    INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    year  INTEGER  NOT NULL CHECK ( year > 2008 ),
    month INTEGER  NOT NULL CHECK ( month >= 1 AND month <= 12 ),
    day   INTEGER  NOT NULL CHECK ( day >= 1 AND day <= 31 ),
    created_at  DATETIME NOT NULL );
```



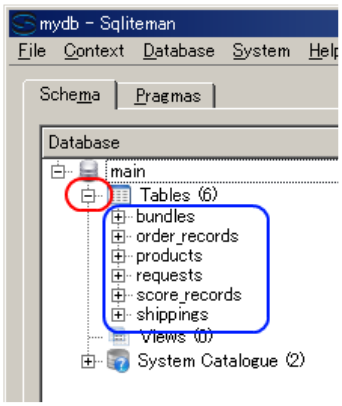
4. **コンソールの確認** (Inspect console)

```
Query OK
Row(s) returned: 0 (More rows can be fetched. Scroll the resultset for more rows and/or read the documentation.)
CREATE TABLE shippings (
    id    INTEGER  PRIMARY KEY AUTOINCREMENT NOT NULL,
    year  INTEGER  NOT NULL CHECK ( year > 2008 ),
    month INTEGER  NOT NULL CHECK ( month >= 1 AND month <= 12 ),
    day   INTEGER  NOT NULL CHECK ( day >= 1 AND day <= 31 ),
    created_at  DATETIME NOT NULL );
```

エラーメッセージが出ていないことを確認

5. **テーブル一覧の表示** (List of tables)

オブジェクト・ブラウザ (Object Browser) の中の「**Tables**」を展開 (Click 'Tables'). テーブル一覧が表示される.



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

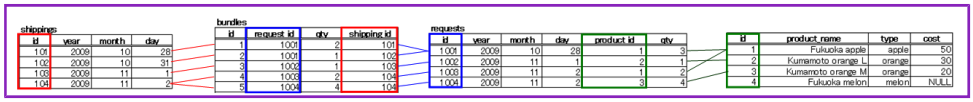
bundles

id	request id	qty	shipping id
1	1001	2	101
2	1001	1	102
3	1002	1	103
4	1003	2	104
5	1004	4	104

shippings

id	year	month	day
101	2009	10	28
102	2009	10	31
103	2009	11	1
104	2009	11	2

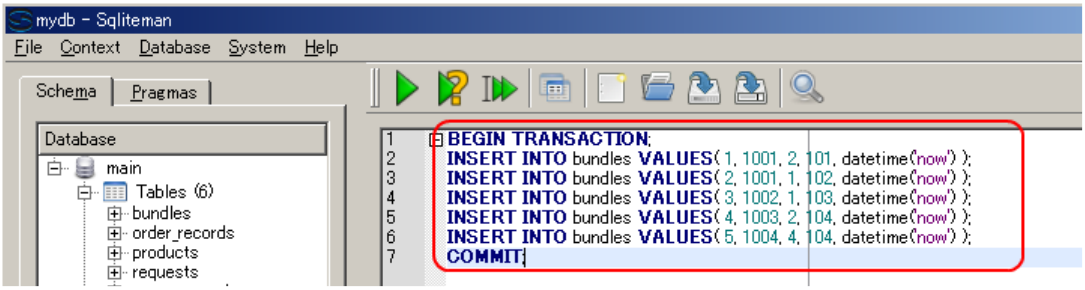
テーブル間の関係



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

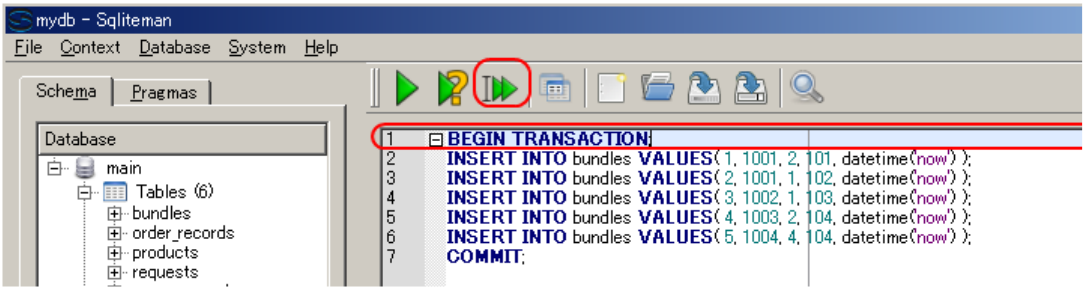
1. **SQL プログラムの記述**

```
BEGIN TRANSACTION;
INSERT INTO bundles VALUES( 1, 1001, 2, 101, datetime('now') );
INSERT INTO bundles VALUES( 2, 1001, 1, 102, datetime('now') );
INSERT INTO bundles VALUES( 3, 1002, 1, 103, datetime('now') );
INSERT INTO bundles VALUES( 4, 1003, 2, 104, datetime('now') );
INSERT INTO bundles VALUES( 5, 1004, 4, 104, datetime('now') );
COMMIT;
```



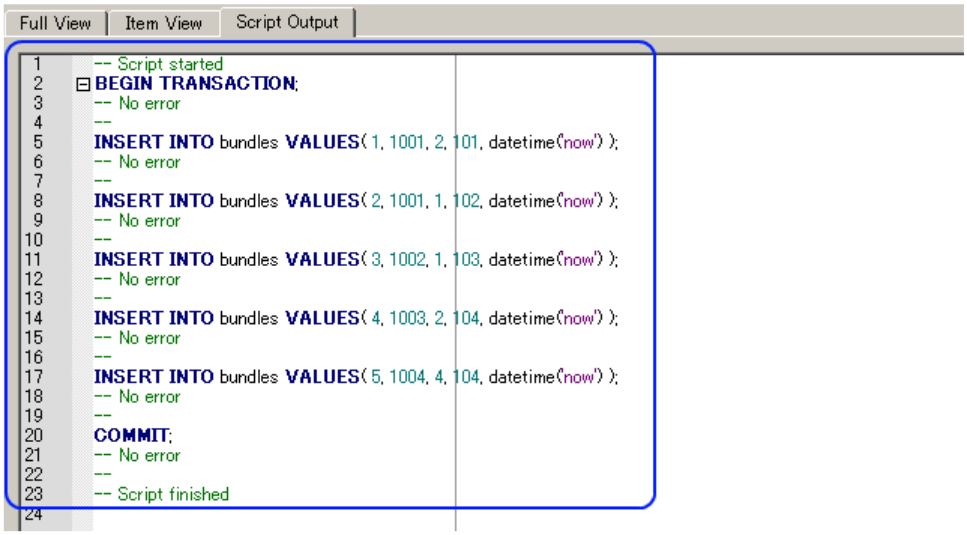
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

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)

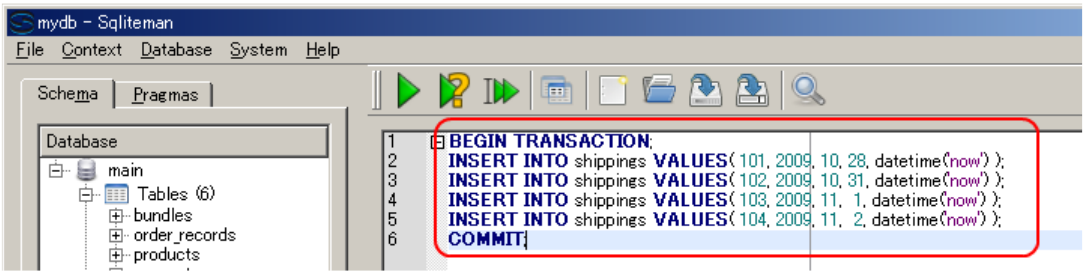
エラーメッセージが出ていないことを確認



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

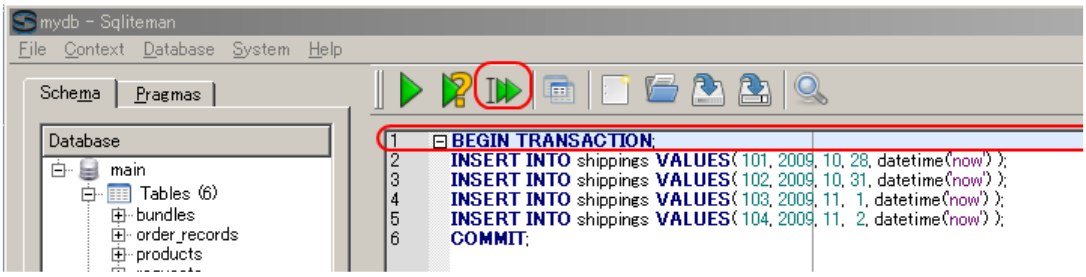
1. **SQL プログラムの記述**

```
BEGIN TRANSACTION;
INSERT INTO shippings VALUES( 101, 2009, 10, 28, datetime('now') );
INSERT INTO shippings VALUES( 102, 2009, 10, 31, datetime('now') );
INSERT INTO shippings VALUES( 103, 2009, 11,  1, datetime('now') );
INSERT INTO shippings VALUES( 104, 2009, 11,  2, datetime('now') );
COMMIT;
```



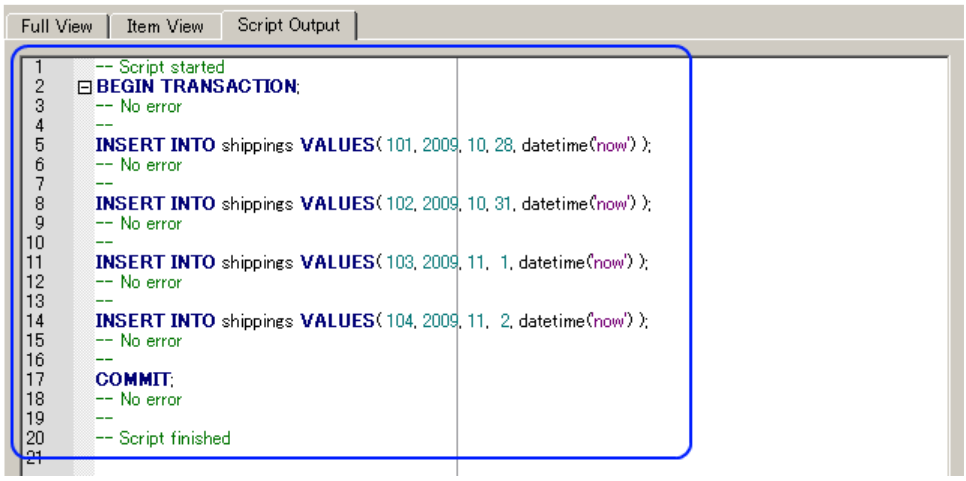
2. **複数の SQL 文の一括実行** (Run multiple SQL statements)

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)

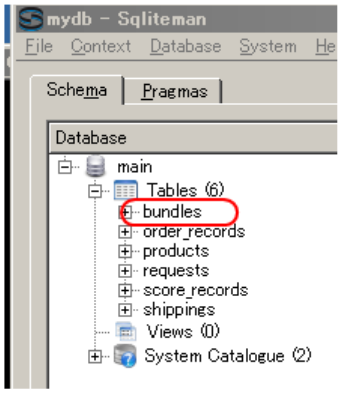
エラーメッセージが出ていないことを確認



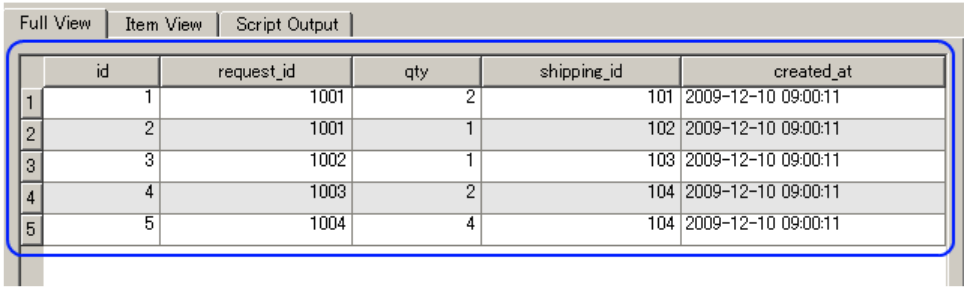
SQLiteman を用いたデータのブラウズ (Browse Data using SQLiteman)

- **bundles テーブル**

テーブル **bundles**を選ぶ (Select table 'products')



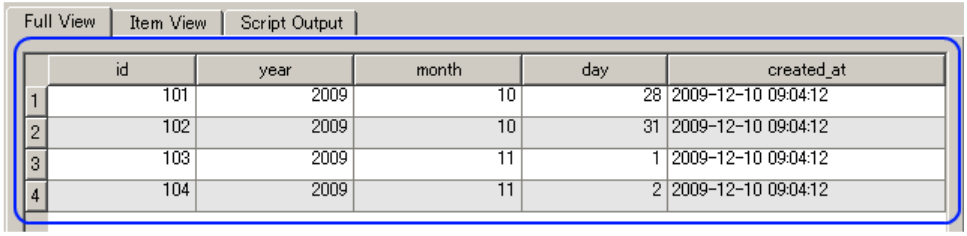
テーブル **bundles**が表示される (table 'products' appears)



	id	request_id	qty	shipping_id	created_at
1	1	1001	2	101	2009-12-10 09:00:11
2	2	1001	1	102	2009-12-10 09:00:11
3	3	1002	1	103	2009-12-10 09:00:11
4	4	1003	2	104	2009-12-10 09:00:11
5	5	1004	4	104	2009-12-10 09:00:11

※ 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).

- **shippings テーブル**



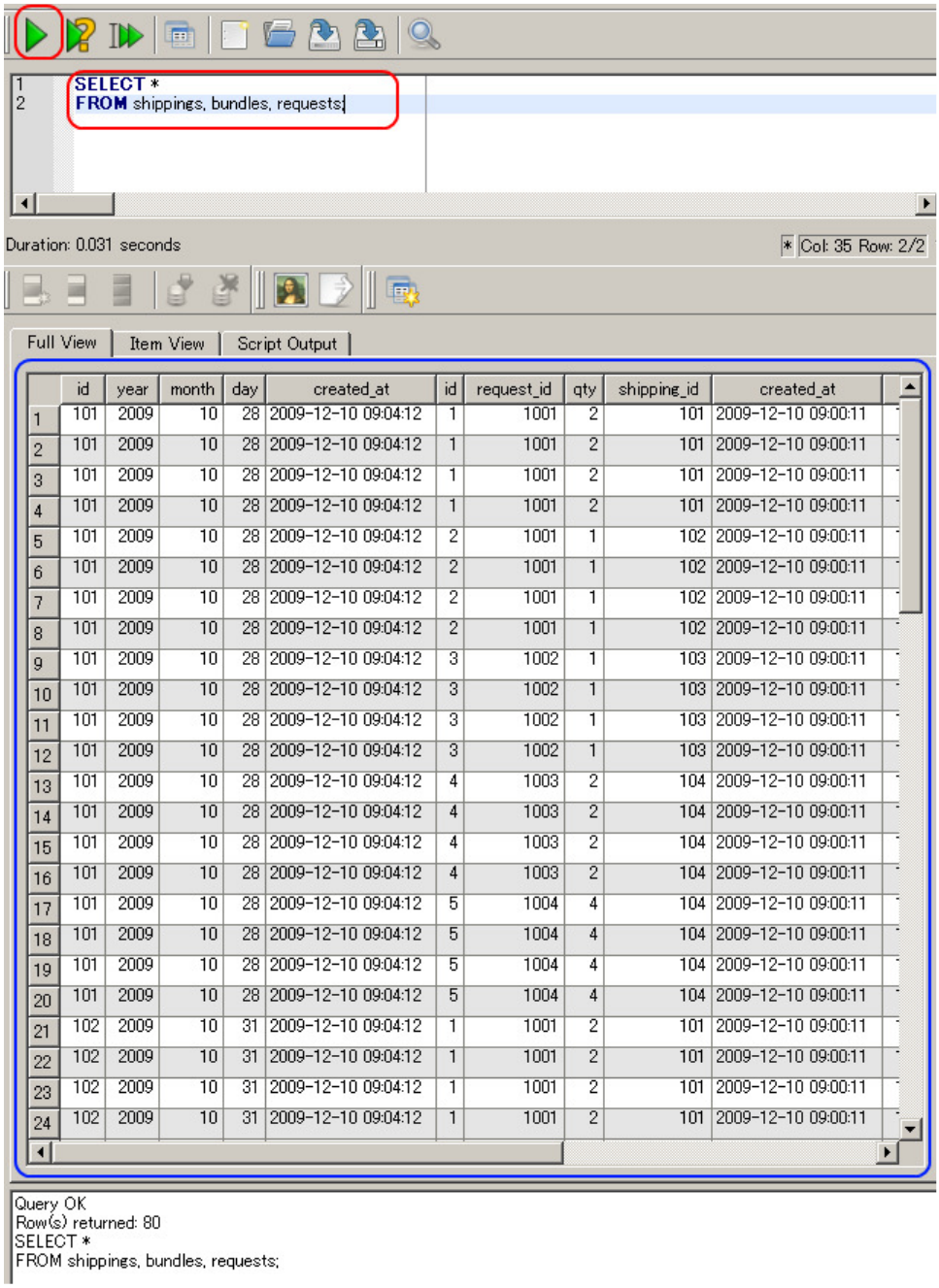
	id	year	month	day	created_at
1	101	2009	10	28	2009-12-10 09:04:12
2	102	2009	10	31	2009-12-10 09:04:12
3	103	2009	11	1	2009-12-10 09:04:12
4	104	2009	11	2	2009-12-10 09:04:12

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

直積集合 (Cartesian product)

SQL を使い, 複数のテーブルからの直積を簡単に得ることができる.

```
SELECT *
FROM shippings, bundles, requests;
```



	id	year	month	day	created_at	id	request_id	qty	shipping_id	created_at
1	101	2009	10	28	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
2	101	2009	10	28	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
3	101	2009	10	28	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
4	101	2009	10	28	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
5	101	2009	10	28	2009-12-10 09:04:12	2	1001	1	102	2009-12-10 09:00:11
6	101	2009	10	28	2009-12-10 09:04:12	2	1001	1	102	2009-12-10 09:00:11
7	101	2009	10	28	2009-12-10 09:04:12	2	1001	1	102	2009-12-10 09:00:11
8	101	2009	10	28	2009-12-10 09:04:12	2	1001	1	102	2009-12-10 09:00:11
9	101	2009	10	28	2009-12-10 09:04:12	3	1002	1	103	2009-12-10 09:00:11
10	101	2009	10	28	2009-12-10 09:04:12	3	1002	1	103	2009-12-10 09:00:11
11	101	2009	10	28	2009-12-10 09:04:12	3	1002	1	103	2009-12-10 09:00:11
12	101	2009	10	28	2009-12-10 09:04:12	3	1002	1	103	2009-12-10 09:00:11
13	101	2009	10	28	2009-12-10 09:04:12	4	1003	2	104	2009-12-10 09:00:11
14	101	2009	10	28	2009-12-10 09:04:12	4	1003	2	104	2009-12-10 09:00:11
15	101	2009	10	28	2009-12-10 09:04:12	4	1003	2	104	2009-12-10 09:00:11
16	101	2009	10	28	2009-12-10 09:04:12	4	1003	2	104	2009-12-10 09:00:11
17	101	2009	10	28	2009-12-10 09:04:12	5	1004	4	104	2009-12-10 09:00:11
18	101	2009	10	28	2009-12-10 09:04:12	5	1004	4	104	2009-12-10 09:00:11
19	101	2009	10	28	2009-12-10 09:04:12	5	1004	4	104	2009-12-10 09:00:11
20	101	2009	10	28	2009-12-10 09:04:12	5	1004	4	104	2009-12-10 09:00:11
21	102	2009	10	31	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
22	102	2009	10	31	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
23	102	2009	10	31	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11
24	102	2009	10	31	2009-12-10 09:04:12	1	1001	2	101	2009-12-10 09:00:11

結合問い合わせ (join query)

結合問い合わせは，直積から，条件を満足する行を選んだものになる．

List all 'shippings.month', 'shiping.day' and 'bundles.qty' that satisfy "requests.month = 11"

```
SELECT shippings.month, shippings.day, bundles.qty
FROM shippings, bundles, requests
WHERE requests.id = bundles.request_id
          AND shippings.id = bundles.shipping_id
          AND requests.month = 11;
```

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)

1. 次の SQL 問い合わせの評価結果は何か？ (What is the evaluation result of the following SQL query).

```
SELECT products.product_name, requests.qty
FROM products, requests
WHERE products.id = requests.product_id
          AND requests.qty > 2;
```

2. 次の SQL 問い合わせの評価結果は何か？ (What is the evaluation result of the following SQL query).

```
SELECT shippings.year, shippings.month, shippings.day
FROM shippings, bundles, requests, products
WHERE products.id = requests.product_id
          AND requests.id = bundles.request_id
          AND shippings.id = bundles.shipping_id
          AND cost > 20;
```

3. まず SQLite を使い，下記のテーブルを定義しなさい (Define a table below using SQL) loan(id, branchname, amount) borrow(id, customername, loanid)
4. 次に SQLite を使い，下記の SQL を評価させなさい (Evaluate the following SQL) BEGIN TRANSACTION; INSERT INTO loan values(1, 'fukuoka', 1000); INSERT INTO loan values(2, 'saga', 2000); INSERT INTO loan values(3, 'saga', 1500); INSERT INTO loan values(4, 'kumamoto', 3000); INSERT INTO loan values(5, 'fukuoka', 2500); COMMIT;
5. 次に SQLite を使い，下記の SQL を評価させなさい (Evaluate the following SQL)

```
(1)
        select * from loan;
(2)
        select branchname from loan;
(3)
        select distinct branchname from loan;
(4)
        select id, branchname, amount * 1000 from loan;
(5)
        select id from loan where branchname = 'fukuoka';
        select id from loan where branchname = 'saga';
        select id from loan where branchname = 'kumamoto';
(6)
        select id from loan where amount < 2000 and amount >= 1000;
(7)
        BEGIN TRANSACTION;
        INSERT INTO borrow values(1001, 'X', 1);
        INSERT INTO borrow values(1002, 'X', 2);
        INSERT INTO borrow values(1003, 'X', 3);
        INSERT INTO borrow values(1004, 'Y', 4);
        INSERT INTO borrow values(1005, 'X', 5);
        COMMIT;
(8)
        select *
        from loan, borrow
(9)
        select customername, amount
        from loan, borrow;
(10)
        select customername, amount
        from loan, borrow
        where loan.id = borrow.loanid;
(11)
        select customername, amount
        from loan, borrow
        where loan.id = borrow.loanid AND borrow.customername = 'X';
```

解答例 (Answers)

※ 問い合わせ結果は1つのテーブルになる．その属性名には，元のテーブル名と属性名をドットでつなげたドット記法を用いている．

1.

products.product_name	requests.qty
Fukuoka apple	3
Kumamoto orange M	4

2.

shippings.year	shippings.month	shippings.day
2009	10	28
2009	10	31
2009	11	1
2009	11	2