埋め込み SQL

URL: http://www.db.is.kyushu-u.ac.jp/rinkou/addb/9.html

(Windows の場合のみ, オプション) MkEditor のインストール (Windows only. Install an editor. It is optional)

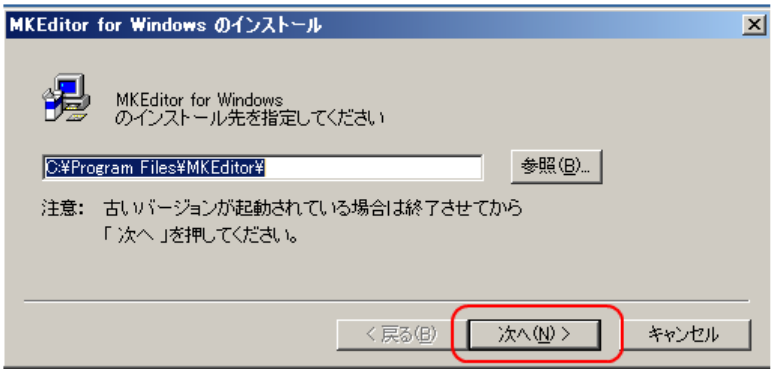
エディタとしては MkEditor または xyzzy を推奨しておきます.

MkEditor は優れたエディタ. Windows で動く.

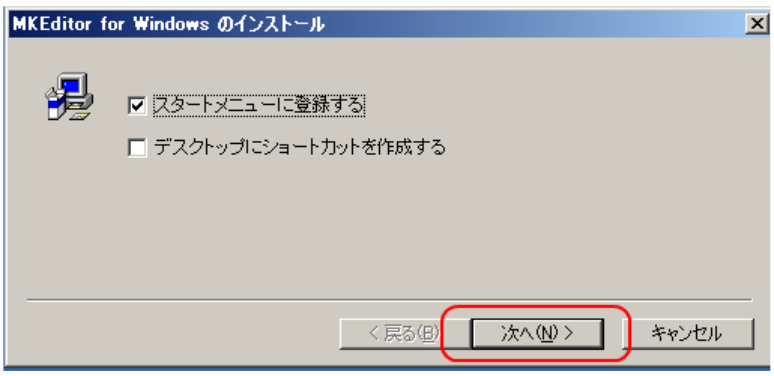
http://www.mk-square.com/

・「はい」をクリック

・「次へ」をクリック

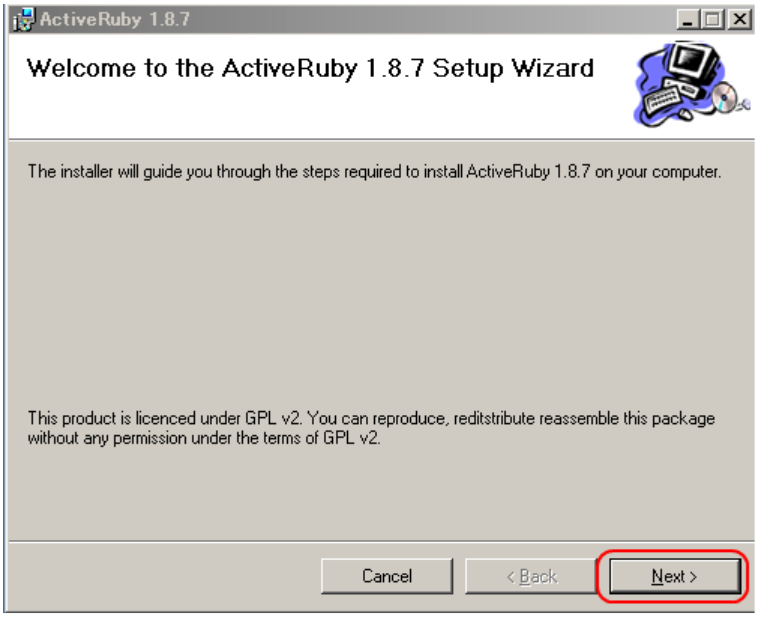


・「次へ」をクリック



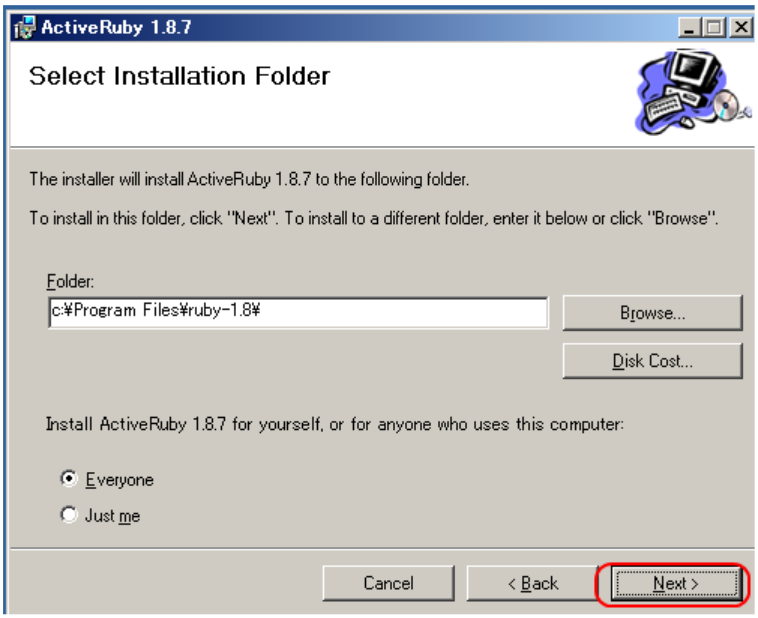
(Windows の場合のみ) ActiveScriptRuby のインストール

1. ActiveScriptRuby のインストーラプログラム **ActiveRuby.msi を起動**
2. **ようこそ画面**
   「**Next**」をクリック



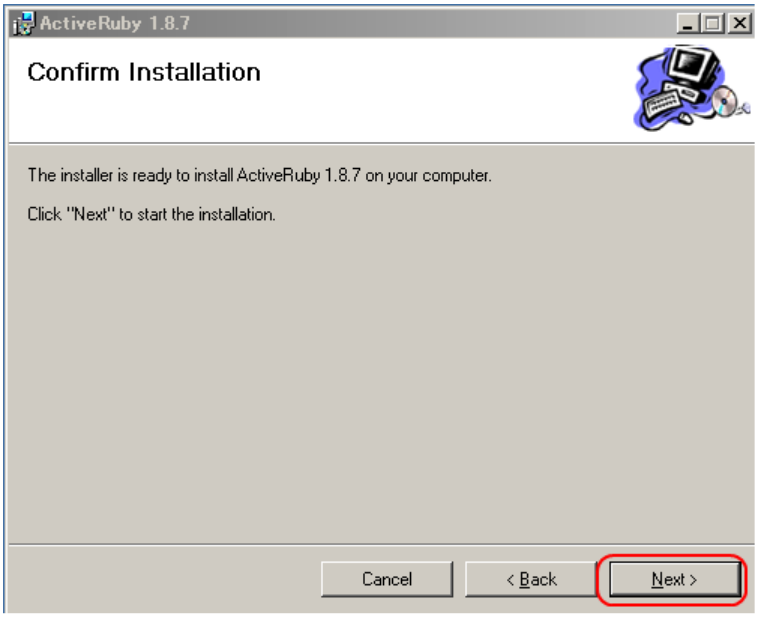
3. **インストールディレクトリの設定**

   デフォルトのままでよい．「**Next**」をクリック

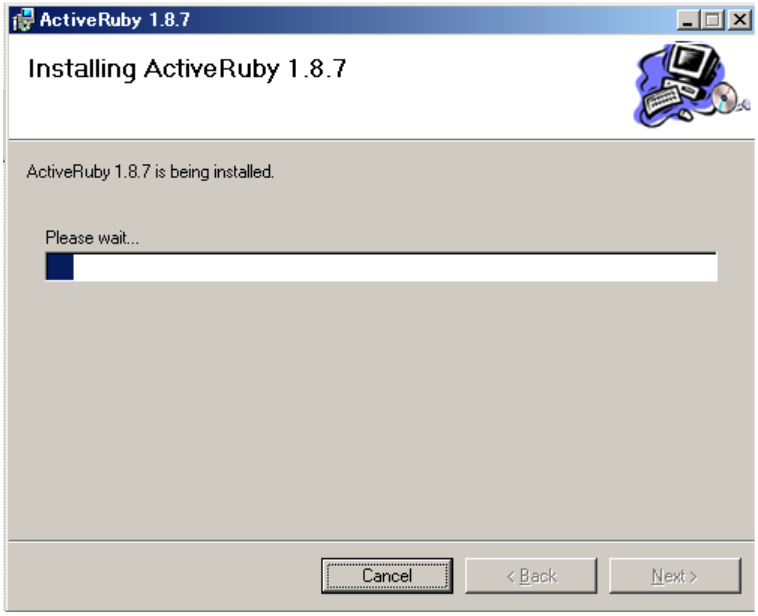


4. **インストール開始の確認**

   「**Next**」をクリック

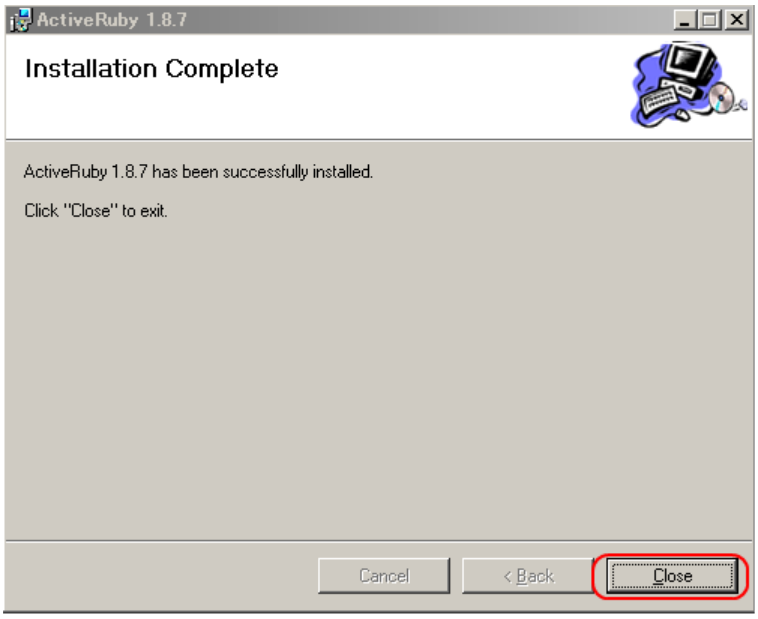


5. インストールが始まる



6. **インストール完了の確認**

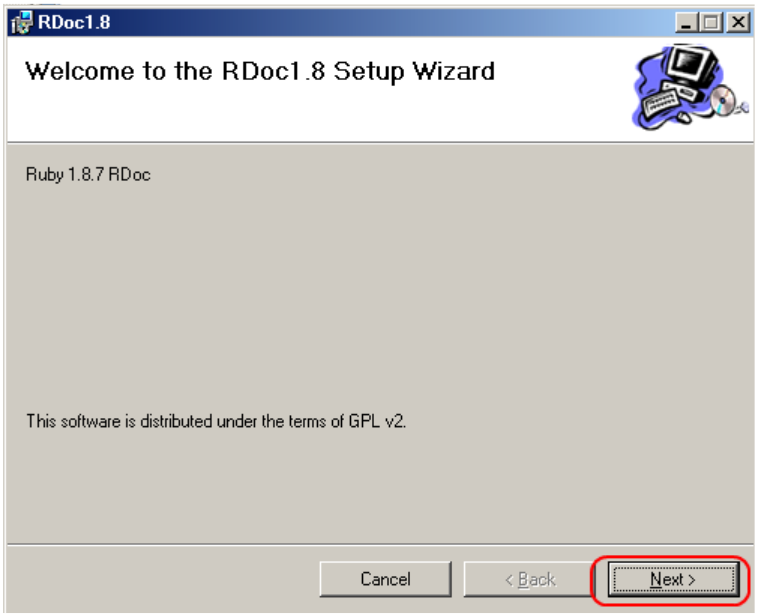
「**Close**」をクリック



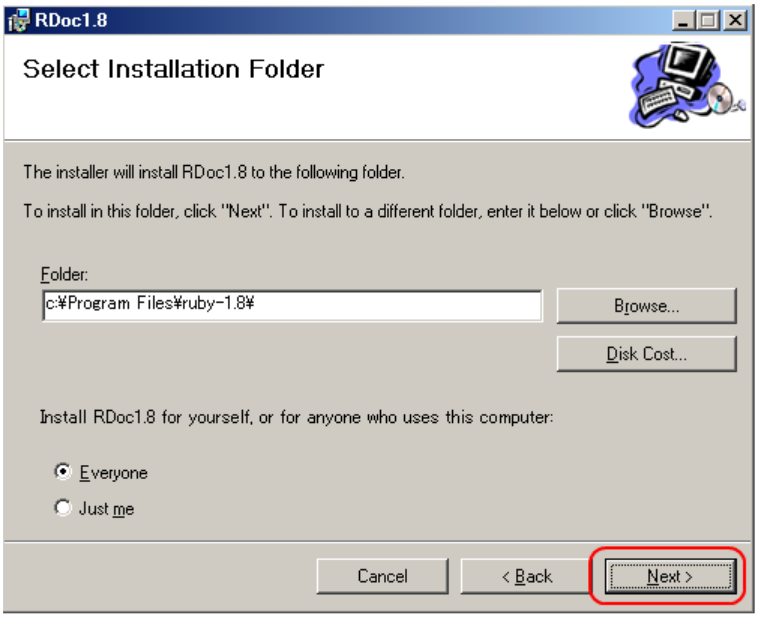
7. RDoc のインストーラプログラム **RDoc.msi を起動**

8. **ようこそ画面**

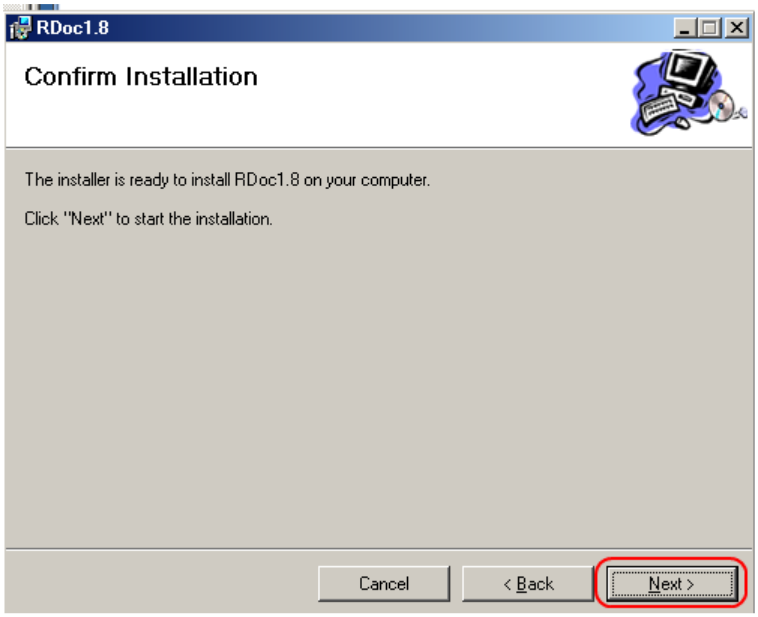
「**Next**」をクリック



9. **インストールディレクトリの設定**
   デフォルトのままでよい．「**Next**」をクリック

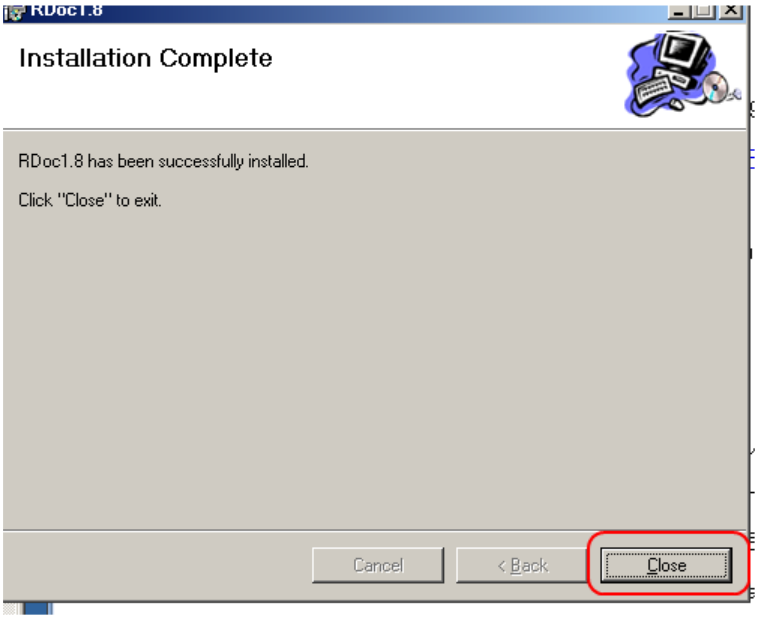


10. **インストール開始の確認**
    「**Next**」をクリック



11. インストールが始まる

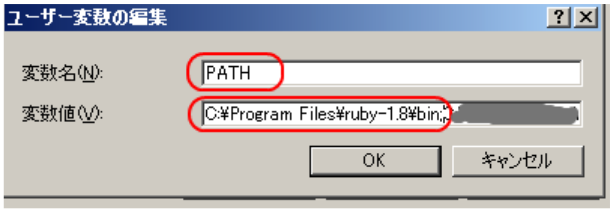
12. **インストール完了の確認**
    「**Close**」をクリック



13. **Windows の環境変数 PATH の設定**

```
C:¥Program Files¥ruby-1.8¥bin
```

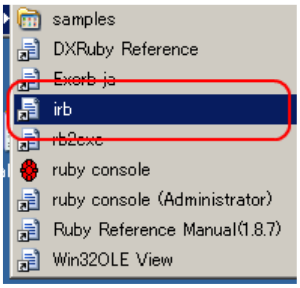
を設定



(Windows の場合のみ) Ruby 処理系の起動(対話型)(Interactive Ruby Interpreter)

対話型の処理系は，コンソールでコマンドを打つたびに評価結果が表示されるもの．手軽に試すときは便利です．

- Windows の「スタート」から「Ruby 1.8」を選び，「irb」を選ぶ
  irb は「Interactive Ruby」



- 足し算

```
1 + 2 + 3
1234567890123456789 + 1234567890123456789
```

```
irb(main):008:0> 1 + 2 + 3
=> 6
irb(main):009:0> 1234567890123456789 + 1234567890123456789
=> 2469135780246913578
```

- sin, sqrt, log

```
include Math
sin(1.57)
sqrt(2)
log(10)
```

```
irb(main):011:0> include Math
=> Object
irb(main):012:0> sin(1.57)
=> 0.999999682931835
irb(main):013:0> sqrt(2)
=> 1.4142135623731
irb(main):014:0> log(10)
=> 2.30258509299405
```

・変数

```
x = 100
y = 200
x + y
```

```
irb(main):007:0> x = 100
=> 100
irb(main):008:0> y = 200
=> 200
irb(main):009:0> x + y
=> 300
```

(Windows の場合のみ) Ruby の SQLite3 パッケージ

参考 Web ページ: http://rubyforge.org/projects/sqlite-ruby

sqlite3.dll の設定

1. **sqlite3.dll を C:¥Windows¥System32 にコピーする**

Ruby の SQLite3 パッケージのインストール

1. **Windows のコマンドプロンプト**を使う
2. Ruby の SQLite3 パッケージのインストーラプログラム **sqlite3-ruby-1.2.5-x86-mingw32.gem** を使う

```
>gem install --local sqlite3-ruby-1.2.5-x86-mingw32.gem
```

エラーが出たときは，環境変数 PATH の設定を間違っている可能性がある．

3. インストールできたかは，「**gem list**」の実行で確認できる．実行例は次の通り．

```
R:¥>gem list

*** LOCAL GEMS ***

sqlite3-ruby (1.2.5)

R:¥>
```

Ruby プログラムの中に SQL プログラムを埋め込み SQLite3 上で動かす (Embedded SQL in a Ruby Program)

・Ruby の中に埋め込まれた SQL を用いた問い合わせ
・Ruby の中に埋め込まれた SQL を用いたテーブルへの行の挿入
・テーブルの一覧表示

SQLite の SQL に関する詳しい説明は:

・http://www.hwaci.com/sw/sqlite/lang.html にあります．

Ruby から SQLite を使う方法については:

・http://sqlite-ruby.rubyforge.org/sqlite3/faq.html

SQLiteが認識できる SQL について，日本語でのすばらしい説明が http://net-newbie.com/sqlite/lang.html にあります．

SQLite のデータ型 (SQL Data Types)

SQLite では，**INTEGER, REAL, TEXT, BLOB** などの**データ型**を扱うことができます． SQLite のデータ型の詳しい説明は http://www.sqlite.org/datatype3.html にあります．

- SQL の **INTEGER**：符号付きの整数 (signed integer)
- SQL の **REAL**：浮動小数点数 (floating point value)
- SQL の **TEXT**：文字列 (text string)
- SQL の **BLOB**)：バイナリ・ラージ・オブジェクト (Binary Large Object). 入力がそのままの形で格納される (stored exactly as it was input).

演習では E テーブルを用いる. **すでに E テーブルを定義済みの場合には，次の SQL を実行する必要は無い**.

If you already define table 'E', you don't have to execute the following SQL again.

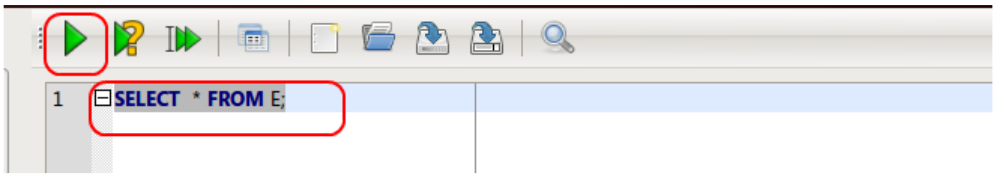
```
CREATE TABLE E (
    name         TEXT     NOT NULL,
    score        INTEGER  NOT NULL CHECK ( score >= 0 AND score <=100 ),
    student_name TEXT     NOT NULL,
    created_at   DATETIME NOT NULL,
    updated_at   DATETIME,
    UNIQUE (name, student_name) );
```

	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]

Ruby の中に埋め込まれた SQL による問い合わせ (SQL query embedded in a Ruby program)

- **SQL を用いてテーブルの全ての行の表示** (All rows in a table)

```
SELECT * FROM E;
```



- 今度は Ruby のプログラムを作成してみる.

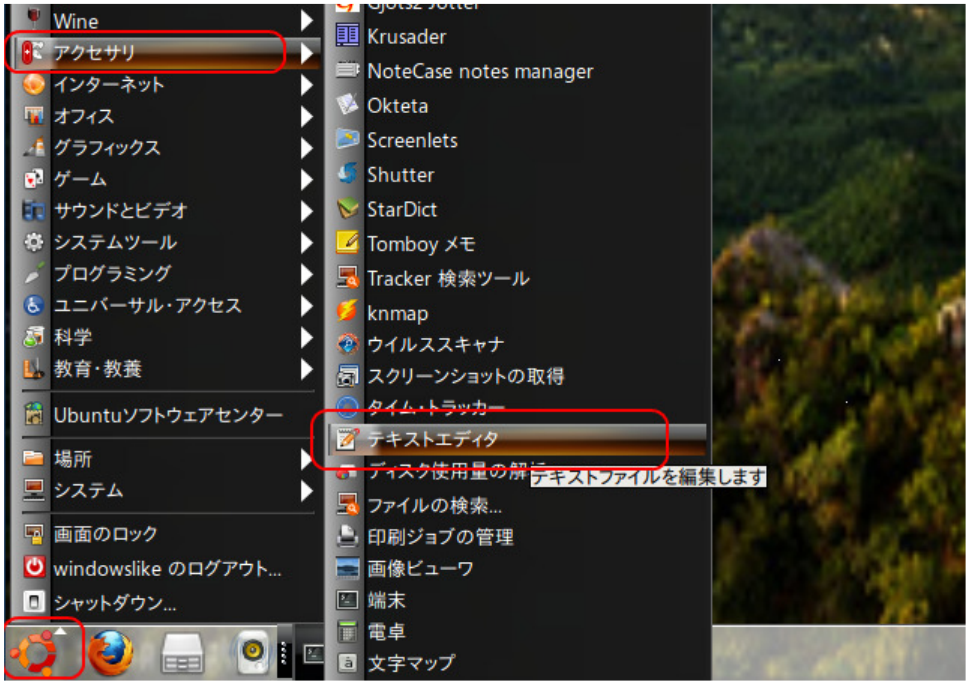
  次の Ruby のプログラムをエディタで入力し，名前をつけて保存する

  ○「#」で始まる行はコメントなので，入力する必要は無い

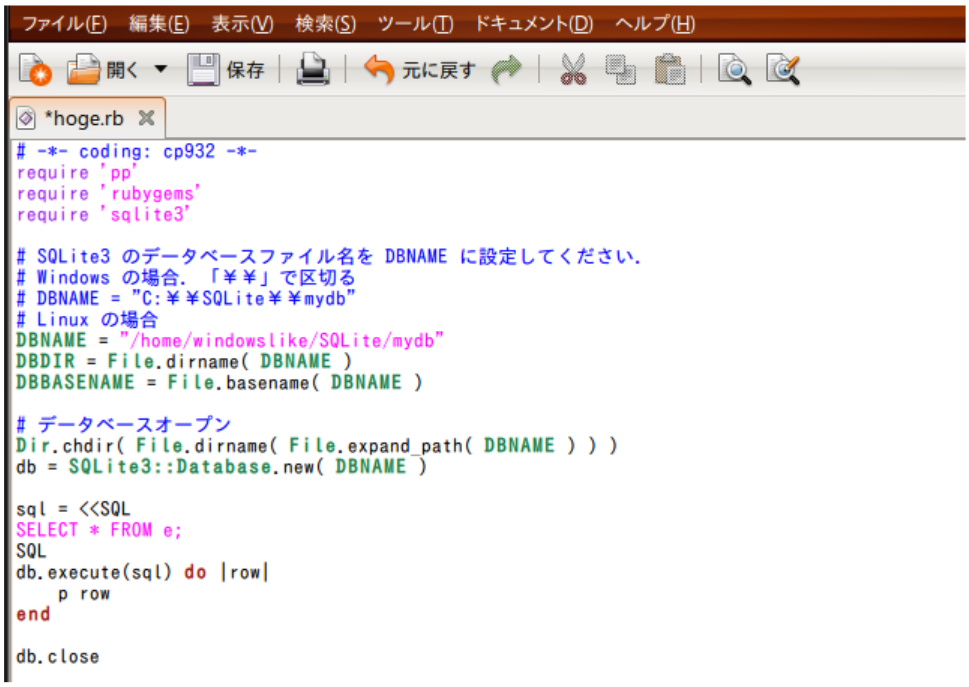
```
require 'pp'
require 'rubygems'
require 'sqlite3'

# SQLite3 のデータベースファイル名を DBNAME に設定してください.
# Windows の場合，「¥¥」で区切る
# DBNAME = "C:¥¥SQLite¥¥mydb"
# Linux の場合
DBNAME = "/home/windowslike/SQLite/mydb"
DBDIR = File.dirname( DBNAME )
DBBASENAME = File.basename( DBNAME )

# データベースオープン
Dir.chdir( File.dirname( File.expand_path( DBNAME ) ) )
db = SQLite3::Database.new( DBNAME )

sql = <<SQL
SELECT * FROM E;
SQL
db.execute(sql) do |row|
    p row
end

db.close
```

■ Ubuntu の場合の例(gedit エディタを使っている)

「アクセサリ」→「テキストエディタ」と操作すると，gedit エディタが起動する.



ファイルを保存したディレクトリとファイル名を覚えておくこと.

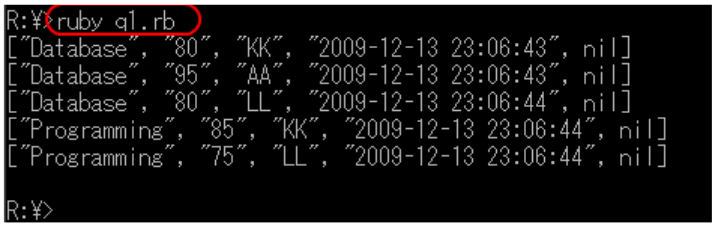



```
# -*- coding: cp932 -*-
require 'pp'
require 'rubygems'
require 'sqlite3'

# SQLite3 のデータベースファイル名を DBNAME に設定してください.
# Windows の場合. 「￥￥」で区切る
# DBNAME = "C:￥￥SQLite￥￥mydb"
# Linux の場合
DBNAME = "/home/windowslike/SQLite/mydb"
DBDIR = File.dirname( DBNAME )
DBBASENAME = File.basename( DBNAME )

# データベースオープン
Dir.chdir( File.dirname( File.expand_path( DBNAME ) ) )
db = SQLite3::Database.new( DBNAME )

sql = <<SQL
SELECT * FROM e;
SQL
db.execute(sql) do |row|
    p row
end

db.close
```

- **実行してみる**

  次のように実行する.

```
cd <保存した Ruby プログラムのディレクトリ名 >
ruby <保存した Ruby プログラムのファイル名 >
```



Windows で, 次のような**エラー**が発生することがあります.

```
./sqlite3.dll: 127: 指定されたプロシージャが見つかりません。 - Init_sqlite3 (LoadError)
./sqlite3.dll
            from C:/Program
Files/ruby-1.8/lib/ruby/vendor_ruby/1.8/rubygems/custom_require.rb:31:in
```

```
        'require'
                from hoge.rb:2
```

回避法ですが，**sqlite3.dll というファイル**を、http://www.sqlite.org/download.html から**再度ダウンロード**し，C:¥Windows¥System32 に置く(昔のファイルを**上書き**)と直る場合があります.

- **条件に合致する行のみの表示** (その 1)

```
SELECT * FROM E WHERE student_name = 'KK';
```

Ruby のプログラムは次のようになります.

○ 先ほどのプログラムを、ほぼそのまま使える.「SELECT * FROM E **WHERE student_name = 'KK';**」の**1行を書き換える**だけ.

```
require 'pp'
require 'rubygems'
require 'sqlite3'

# SQLite3 のデータベースファイル名を DBNAME に設定してください.
# Windows の場合. 「¥¥」で区切る
# DBNAME = "C:¥¥SQLite¥¥mydb"
# Linux の場合
DBNAME = "/home/windowslike/SQLite/mydb"
DBDIR = File.dirname( DBNAME )
DBBASENAME = File.basename( DBNAME )

# データベースオープン
Dir.chdir( File.dirname( File.expand_path( DBNAME ) ) )
db = SQLite3::Database.new( DBNAME )

sql = <<SQL
SELECT * FROM E WHERE student_name = 'KK';
SQL
db.execute(sql) do |row|
    p row
end

db.close
```

【実行結果の例】

ruby ＜保存した Ruby プログラムファイル名 (Ruby program file name)＞

- **条件に合致する行のみの表示** (その 2)

```
SELECT * FROM E WHERE score > 80;
```

Ruby のプログラムは次のようになります.

○ 先ほどのプログラムを、ほぼそのまま使える.「**SELECT * FROM E WHERE score > 80;**」の**1行を書き換える**だけ.

```
require 'pp'
require 'rubygems'
require 'sqlite3'

# SQLite3 のデータベースファイル名を DBNAME に設定してください.
# Windows の場合. 「¥¥」で区切る
# DBNAME = "C:¥¥SQLite¥¥mydb"
# Linux の場合
DBNAME = "/home/windowslike/SQLite/mydb"
DBDIR = File.dirname( DBNAME )
DBBASENAME = File.basename( DBNAME )

# データベースオープン
Dir.chdir( File.dirname( File.expand_path( DBNAME ) ) )
db = SQLite3::Database.new( DBNAME )

sql = <<SQL
SELECT * FROM E WHERE score > 80;
SQL
db.execute(sql) do |row|
    p row
end

db.close
```

【実行結果の例】

ruby ＜保存した Ruby プログラムファイル名 (Ruby program file name)＞

```
R:¥>ruby q1.rb
["Database", "95", "AA", "2009-12-13 23:06:43", nil]
["Programming", "85", "KK", "2009-12-13 23:06:44", nil]

R:¥>
```

SQL を用いたテーブルへの行の挿入

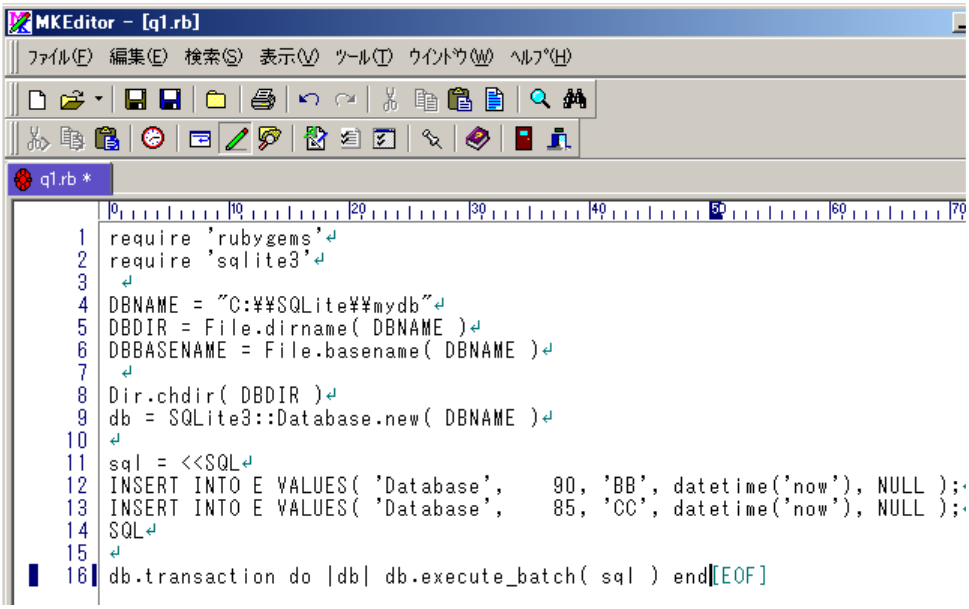
```
INSERT INTO E VALUES( 'Database',    90, 'BB', datetime('now'), NULL );
INSERT INTO E VALUES( 'Database',    85, 'CC', datetime('now'), NULL );
```

Ruby でプログラムを作るときの要点は:

- **db.execute_batch(...)** は, Ruby で**複数の SQL を1度に実行**するためのもの
- **db.transaction do ... end** は**トランザクション**開始のRuby の決まり文句. 正常終了すれば自動的にコミットし, 異常終了の場合には自動的にロールバックされる.

Ruby のプログラムは次のようになります.

```
require 'pp'
require 'rubygems'
require 'sqlite3'

# SQLite3 のデータベースファイル名を DBNAME に設定してください.
# Windows の場合, 「¥¥」で区切る
# DBNAME = "C:¥¥SQLite¥¥mydb"
# Linux の場合
DBNAME = "/home/windowslike/SQLite/mydb"
DBDIR = File.dirname( DBNAME )
DBBASENAME = File.basename( DBNAME )

# データベースオープン
Dir.chdir( File.dirname( File.expand_path( DBNAME ) ) )
db = SQLite3::Database.new( DBNAME )

sql = <<SQL
INSERT INTO E VALUES( 'Database',    90, 'BB', datetime('now'), NULL );
INSERT INTO E VALUES( 'Database',    85, 'CC', datetime('now'), NULL );
SQL

db.transaction do |db| db.execute_batch( sql ) end

db.close
```

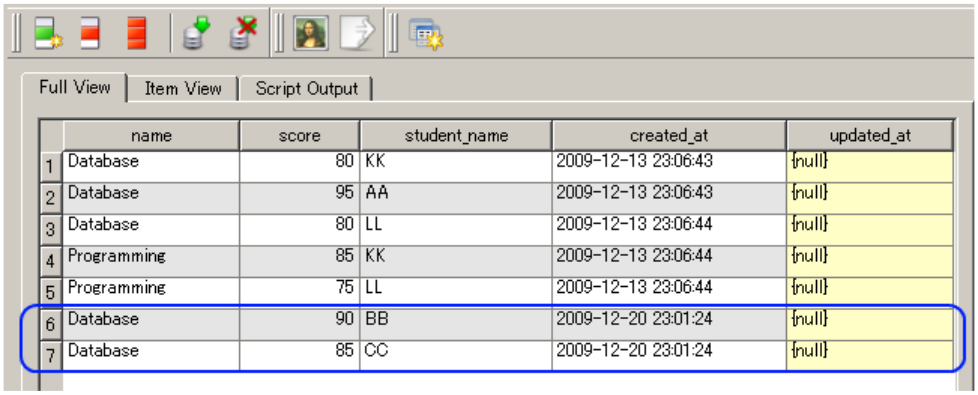


【実行結果の例】

ruby ＜保存した Ruby プログラムファイル名 (Ruby program file name)＞

```
R:¥>ruby q1.rb

R:¥>
```

SQLiteman などを使って, テーブル E を確認しておく



	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]
6	Database	90	BB	2009-12-20 23:01:24	[null]
7	Database	85	CC	2009-12-20 23:01:24	[null]

テーブルの一覧表示

SQLite3 でデータベース内のテーブル一覧を表示するには，**sqlite_maste**, **sqlite_temp_maste**という名前が付いた特別なテーブルを使います．

- **sqlite_master**: TEMPORARY テーブル以外のデータベーススキーマが格納されている．
- **sqlite_temp_master**: TEMPORARY テーブルのデータベーススキーマが格納されている．

データベーススキーマを見たいときは，次のような SQL を実行します．

```
select * from sqlite_master;
```

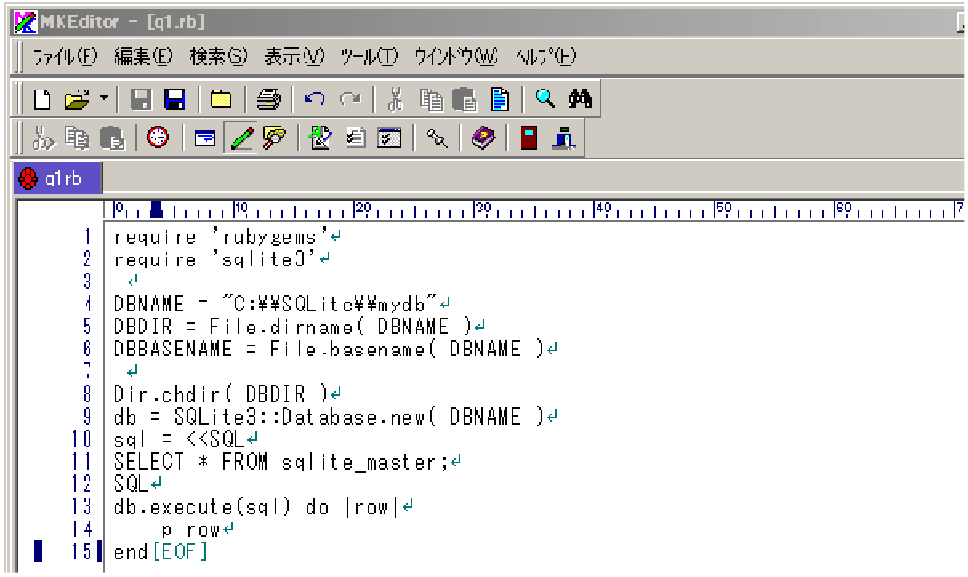
※ sqlite_master, sqlite_temp_master に，DROP TABLE, UPDATE, INSERT, DELETE 操作を行うことは**許されていません**

Ruby のプログラムは次のようになります．

```
require 'pp'
require 'rubygems'
require 'sqlite3'

# SQLite3 のデータベースファイル名を DBNAME に設定してください．
# Windows の場合，「¥¥」で区切る
# DBNAME = "C:¥¥SQLite¥¥mydb"
# Linux の場合
DBNAME = "/home/windowslike/SQLite/mydb"
DBDIR = File.dirname( DBNAME )
DBBASENAME = File.basename( DBNAME )

# データベースオープン
Dir.chdir( File.dirname( File.expand_path( DBNAME ) ) )
db = SQLite3::Database.new( DBNAME )

sql = <<SQL
SELECT * FROM sqlite_master;
SQL
db.execute(sql) do |row|
    p row
end

db.close
```

■ Windows の場合の例（先ほど紹介した Mk エディタを使っている）

ファイルを保存したディレクトリを覚えておくこと．



次に,【実行結果の例】

ruby ＜保存した Ruby プログラムファイル名 (Ruby program file name)＞