<u>トップページ</u> -> 研究道具箱と教材 -> リレーショナルデータベース入門(実践で学ぶ) -> 埋め込み SQL [サイトマップへ] [全文検索へ] [統計情報へ]

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



(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



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

🔂 ActiveRuby 1.8.7	
Select Installation Folder	
The installer will install ActiveRuby 1.8.7 to the following folder.	
To install in this folder, click "Next". To install to a different folder,	enter it below or click "Browse".
<u>F</u> older:	
c:¥Program Files¥ruby-1.8¥	Browse
	<u>D</u> isk Cost
Install ActiveRuby 1.8.7 for yourself, or for anyone who us	es this computer:
• Everyone	
◯ Just <u>m</u> e	
Cancel	< <u>B</u> ack

4. インストール開始の確認



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

🙀 ActiveRuby 1.8.7			_ 🗆 🗙
Installation Complete			
ActiveRuby 1.8.7 has been successfully	installed.		
Click "Close" to exit.			
		1	
	Cancel	< <u>B</u> ack	

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



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

🔂 RDoc1.8	
Select Installation Folder	
The installer will install RDoc1.8 to the following folder.	
To install in this folder, click "Next". To install to a different folder, enter it be	low or click "Browse".
<u>F</u> older:	
c:¥Program Files¥ruby-1.8¥	B <u>r</u> owse
	<u>D</u> isk Cost
Install RDoc1.8 for yourself, or for anyone who uses this computer:	
C Just <u>m</u> e	
Cancel <u>Cancel</u>	Next >

10. インストール開始の確認



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

🔯 RDoc1.8			
Installation Complete			
RDoc1.8 has been successfully installed	ł.		
Click "Close" to exit.			
			ľ
			-
		1	
	Cancel	< <u>B</u> ack	

13. Windows の環境変数 PATH の設定

を設定				
	ユーザー変数の編	集	<u>? ×</u>	
	<u>変数名(N)</u> :	PATH		
	変数値(⊻):	C:¥Program Files¥ruby-1.8¥bin;)		
		ОК	キャンセル	

(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」





(Windows の場合のみ) Ruby の SQLite3 パッケージ

参考 Web ページ: <u>http://rubvforge.org/projects/sqlite-rubv</u>

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:

<u>http://www.hwaci.com/sw/sqlite/lang.html</u> 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 <u>http://net-newbie.com/sqlite/lang.html</u> 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 <u>http://www.sqlite.org/datatype3.html</u> 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));
```

Г	name	score	student_name	created_at	updated_at
1	Database	80	кк	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	кк	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)



· 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 )
|sa| = \langle \langle SQ \rangle
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

ファイル(E) 編集(E) 表示(<u>V</u>) 検索(<u>S</u>) ツール(I) ドキュメント(<u>D</u>) ヘルプ(<u>H</u>)
👔 🍰 開く 🔻 🛄 保存 🚔 🥱 元に戻す 🎺 💥 🗐 💼 🔯 🔀
🐼 *hoge.rb 🗶
<pre># -*- coding: cp932 -*- require 'pp' require 'rubygems' require 'sqlite3'</pre>
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. 「¥羊」で区切る # DBNAME = "C:¥羊SQLite¥¥mydb" # Linux の場合 DBNAME = "/home/windowslike/SQLite/mydb" DBDIR = File.dirname(DBNAME) DBBASENAME = File.basename(DBNAME)
データペースオープン Dir.chdir(File.dirname(File.expand_path(DBNAME))) db = SQLite3::Database.new(DBNAME)
<pre>sql = <<sql SELECT * FROM e; SQL db.execute(sql) do row </sql </pre>
db.close

実行してみる

次のように実行する



/sqlite3.dll from C:/Program

Files/ruby-1.8/lib/ruby/vendor_ruby/1.8/rubygems/custom_require.rb:31:in

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)

'require

```
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、ほほそのまま使える.「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)
sq| = <<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、ほほそのまま使える.「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 )
|sa| = \langle \langle SQ \rangle
SELECT * FROM E WHERE score > 80;
SQL
db.execute(sql) do |row|
    p row
end
db.close
```

埋め込み SQL

【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)>



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"
DBJR = 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

```
🔀 MKEditor – [q1.rb]
  ファイル(E) 編集(E) 検索(S) 表示(V) ツール(T) ウインドウ(W) ヘルプ(H)
🗋 D 😅 📲 🔚 🖸 🕼 🗠 🗠 🐰 🖻 🖺 🔍 🛤
|| 🚲 📭 🚯 🥝 📼 🖉 🦻 🚷 🗐 🛣 🗐 🕲 🔶 📑 🋕
🐣 q1.rb *
        DBNAME = "C:¥¥SQLite¥¥mydb"4
DBDIR = File.dirname( DBNAME )4
DBBASENAME = File.basename( DBNAME )4
      4
      5
      6
      8
         Dir.chdir( DBDIR )↓
      9
         db = SQLite3::Database.new( DBNAME )4
     10
         sql = <<SQL+
INSERT INTO E VALUES( 'Database',
INSERT INTO E VALUES( 'Database',
     11
                                               90, 'BB', datetime('now'), NULL );
85, 'CC', datetime('now'), NULL );
     12
     13
     14
         SQL+
     15
     16 db.transaction do |db| db.execute_batch( sql ) end[EOF]
```

【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

	•		🎽 📗 💽	2 🛛 🗟		
	Ful	IView Item View	Script Output	1		
	Γ	name	score	student_name	created_at	updated_at
	1	Database	80	КК	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	КК	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}
l	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ー覧を表示するには, 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.

KEditor - [q1.rb]
ファイル(E) 編集(E) 検索(G) 表示(M) ツール(E) ウインドウ(M) ヘルレ゙/E)
1 😅 - 🖶 🚍 🕾 🗠 茶 珀 🎼 🖺 🔍 🛤
🚱 a1rb
الالالالالالالالالالالالالالالالالالا

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)>

INTEGER CHECK (info14 >= 0 AND info14 <= 1),¥n INTEGE info15 CHECK (info15 >= 0 AND info15 <= 5), ¥n¥n TEXT NOT NUL org_choiki_kana L,¥n org_choiki_kanji TEXT NOT NULL, ¥n comment_kanji TEXT,¥n omment_kana TEXT,¥n kaisyamei_kana TEXT NOT NULL, kanji TEXT NOT NULL)"] ["table", "kens", "kens", "55735", "CREATE TABLE kens (¥n id RIMARY KEY AUTOINCREMENT NOT NULL,¥n ken_kanji TEXT UNIQUE NO TEXT NOT NULL, ¥n kaisyamei. INTEGER P ken_kanji TEXT UNIQUE NOT NULL, ¥n ke RIMARY KEY AUTOINCREMENT NOT NOLL, #N Kerkanji TEXT ONTOOL NOT NOLL, #N Ke n_kana TEXT UNIQUE NOT NULL)"] ["index", "sqlite_autoindex_kens_1", "kens", "65850", nil] ["index", "sqlite_autoindex_kens_2", "kens", "65850", nil] ["table", "shichosons", "shichosons", "65950", "CREATE TABLE shichosons (¥n j iscode INTEGER PRIMARY KEY NOT NULL CHECK (jiscode >= 1000 AND jiscode < = 50000),¥n ken_kanji TEXT NOT NULL,¥n shichoson_kanji TEXT NOT NULL ,¥n shichoson_kana TEXT NOT NULL)"] ["table", "zips", "zips", "66052", "CREATE TABLE zips (¥n id INTEGE R PRIMARY KEY AUTOINCREMENT NOT NULL,¥n zipcode INTEGER NOT NULL, ¥n zip_old INTEGER NOT NULL,¥n jiscode INTEGER NOT NULL REFERENCES shi chosons(jiscode),¥n choiki_kanji TEXT NOT NULL,¥n choiki_kana TEXT NOT NU LN Xa flacilo TEXT NOT NULL Ya flacili INTEGER NOT NULL REFERENCES shi chosons()iscode), #n choiki_kanji iExt NUL NULL, #n choiki_kana iExt NUL NU LL, ¥n flag10 TEXT NUT NULL, ¥n flag11 INTEGER NUT NULL CHECK (flag11 >= 0 AND flag11 <= 1), ¥n flag12 INTEGER NUT NULL CHECK (fla g12 >= 0 AND flag12 <= 3), ¥n flag13 INTEGER NUT NULL CHECK (flag13 >= 0 AND flag13 <= 1), ¥n info14 INTEGER CHECK (info14 >= 0 AND info14 < = 1), ¥n info15 INTEGER CHECK (info15 >= 0 AND info15 <= 5))"] R:¥>