データベース演習 （Advanced Database Exercise）

３．SQLによる結合

URL: http://www.kkaneko.jp/cc/dbenshu/index.html

**概要 Abstract**

リレーショナルデータベースは、複数のテーブルから構成される。この演習では、複数のテーブルを結合し1つのテーブルを生成する操作について説明する。この演習では、SQL言語とPython 言語を使う。

　Relational database contains relational tables. Today, more-than-one relaional tables are joined, and the result is a new relational table. We use SQL and Python today.

**■　結合演算 join operation**

*R*(*A*1, *A*2, …, *A*n) と*S*(*B*1, *B*2, …, *B*m)をテーブルとする．*R*の属性 *A*i と *S* の属性 *B*j 上の結合演算「*R*[*A*i*θB*j]*S*」の定義は次の通りである．（*θ*は比較演算子である）．

*R*[*A*i*θB*j]*S* は，*R* と *S* の直積集合 (*R*×*S*) の中から「*R*.*A*i*θS*.*B*j」を満足する要素を選んだもの

結合演算「*R*[*A*i*θB*j]*S*」は，SQL を用いて次のように書き下すことができる．

A Join operation " *R*[*A*i*θB*j]*S* " cab be written in SQL language as follows.

SELECT \*

FROM **R**, **S**

WHERE **<AiθBj>**

Let *R*(*A*1, *A*2, …, *A*n) and *S*(*B*1, *B*2, …, *B*m) be relational tables, and *θ* be a comparison operator. Join operation on *A*i in *R* and *B*j in *S* (denoted as *R*[*A*i*θB*j]*S*) is defined as follows.

*R*[*A*i*θB*j]*S* is a set of elements that satisfy *R*.*A*i*θS*.*B*j in the Cartesian product of R and S.

A Join operation " *R*[*A*i*θB*j]*S* " cab be written in SQL language as follows.

SELECT \*

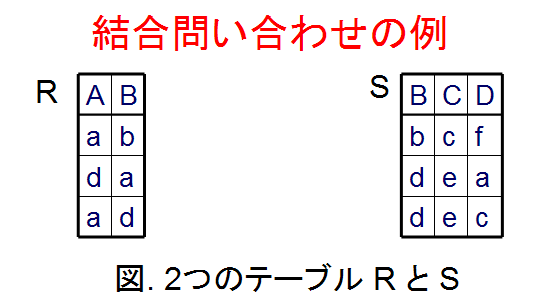
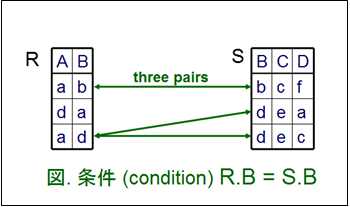
FROM **R**, **S**

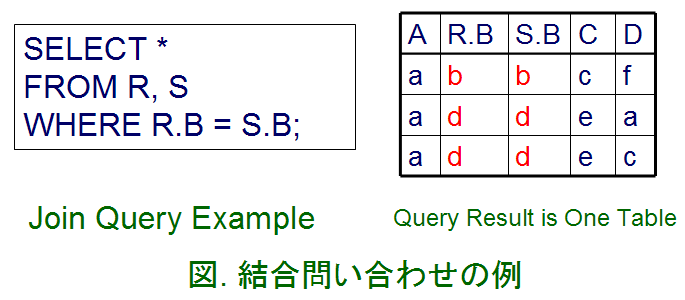
WHERE **<AiθBj>**

**■ 結合問い合わせ join query**

結合問い合わせとは，SQL問い合わせの中の「WHERE <expression>」のところの「<expression>」の中に，結合演算を含むような問い合わせのことである．

A join query is a SQL query that contains join operation in " WHERE <expression>"





■ **外部キー foreign key**

テーブル*R*(…, *A*i, …), *S*(…, *B*j, …)について，属性*B*jがテーブル*S*の主キーであるとする．以下の条件が成り立つとき，属性*A*iをリレーション*R*の外部キーであるという．

　・Rの任意の行の *A*i の値が，必ず，*B*jのいずれかの値になっているか 空値(NULL)である．

Let *R*(…, *A*i, …) and *S*(…, *B*j, …) be relational tables, and *B*j be the primary key of *S*. *A*i is a foreign key if *A*i satisfy the following condition

・Any value of *A*i in *R* is always equal to a value of *B*j in S or NULL.

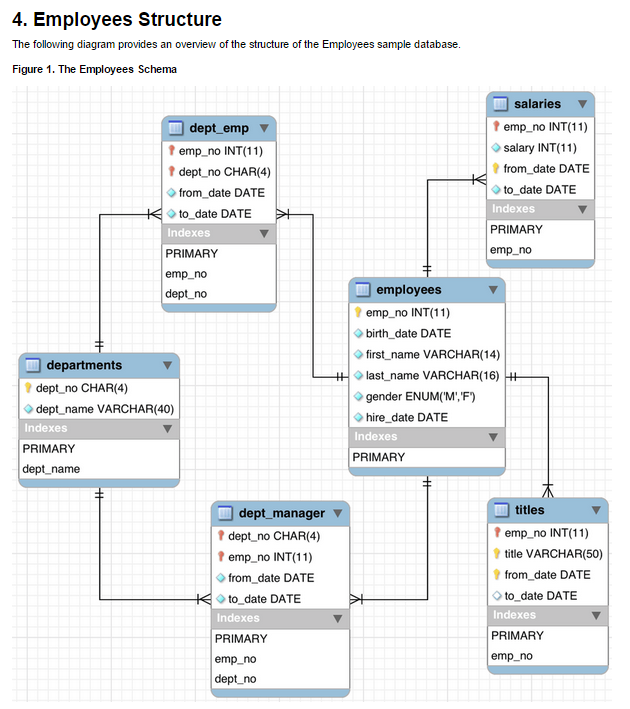
**■ SQLite3 のデータベースファイル SQLite3 database file**

この授業ではリレーショナルデータベース管理システムSQLite3を使う．リレーショナルデータベースはファイルに格納されている．SQLite3では、リレーショナルデータベースのファイルのコピーが簡単にできる．

　We use a relational database management system 'SQLite3'. Relation database of SQLite3 is stored into a single file. We can copy and distribute SQLite3 database file easily.

**■ UML 図 UML diagram**

UML 図は、リレーショナルデータベースのテーブル名、各テーブルの属性名、各属性のデータ型、各テーブルの主キーなどを記述する．今日の授業で使用するデータベースの UML 図を下に示す



出典: MySQL Employees Sample Database https://dev.mysql.com/doc/employee/en/sakila-structure.html

より引用

**演習 3 (Exercises 3)**

* **ステップ1 (Step 1)**

**以下の手順で、MySQL Employees Sampleデータベースを準備しなさい**

(Prepare a MySQL Employee Sample database)

**hoge3.sqlite** の「オリジナル」については，下に説明している．便利のため

https://www.kunihikokaneko.com/free/dbenshu/hoge3.sqlite

として同じもの（改変していないもの）を置いている．

A database file named 'hoge3.sqlite' will be distributed using a USB memory. Everyone is expected to copy the database file to '**C:\hoge3.sqlite** '.



今日の授業で使用するMySQL employees sample データベースの出典と著作権表示

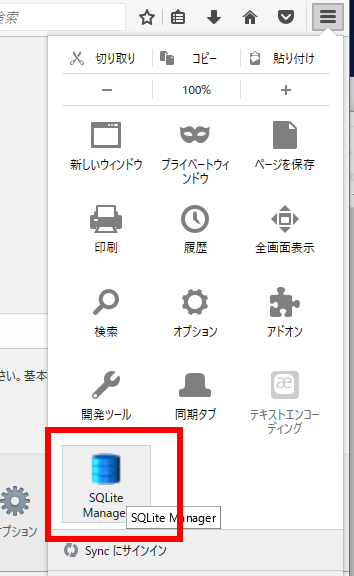
出典：MySQL employees sample database https://dev.mysql.com/doc/employee/en/   
===  
This work is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.  
Please contact <http://www.mysql.com/about/contact/> for more information.

* **ステップ2 (Step 2)**

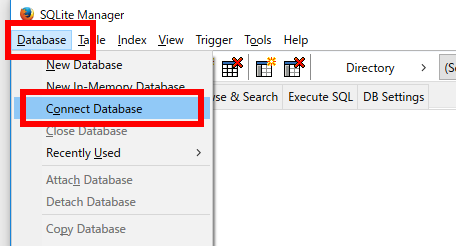
**Firefoxを使って、C:\hoge3.sqliteの確認を行う．**

1. **Firefoxを起動．**Firefoxのメニューで「**SQLite Manager**」をクリック

Click "SQLite Manager" in firefox menu.

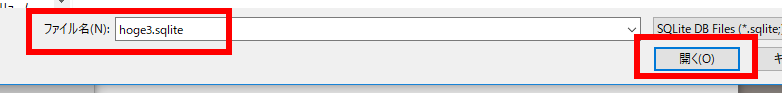


1. **データベースに接続**. Connect to database. And, specify the database file name

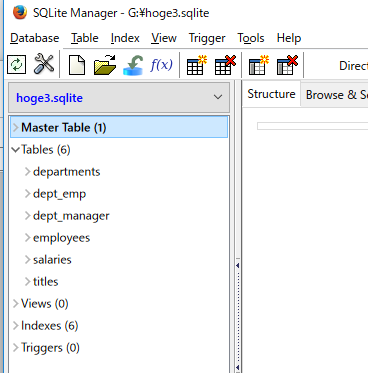


1. データベースファイル名を指定

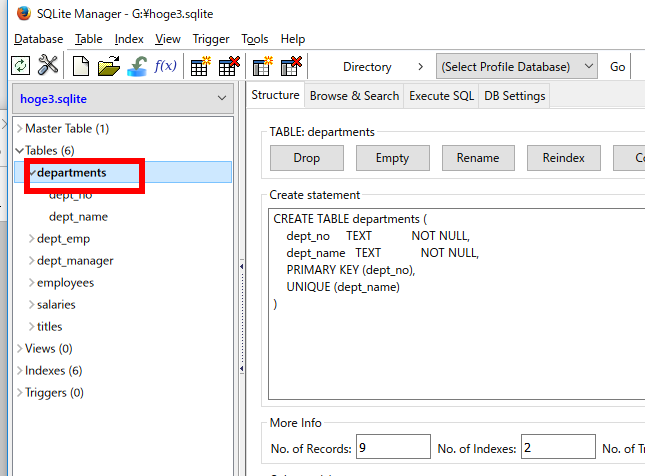
データベースファイル **C:\hoge3.sqlite** を指定する．



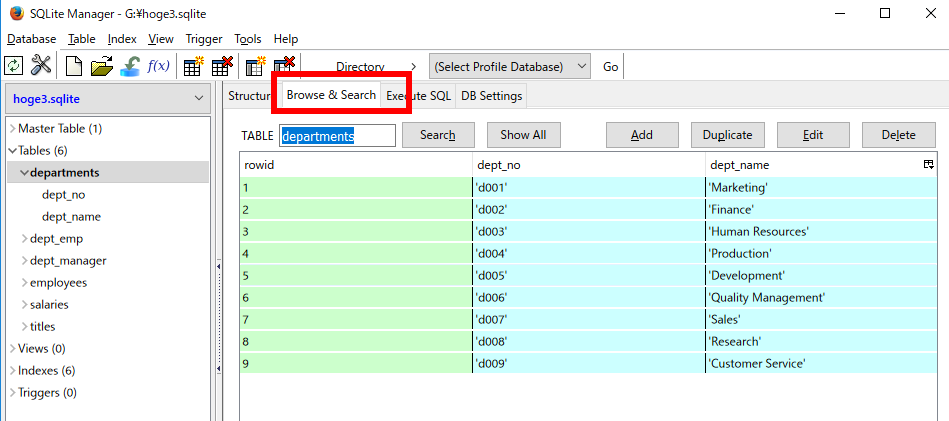
④ **テーブル一覧**が表示される．



⑤ 「departments」をクリックすると，departmentsテーブルが表示される．



⑥ 「Browse & search」をクリックすると，テーブルの中身が表示される．



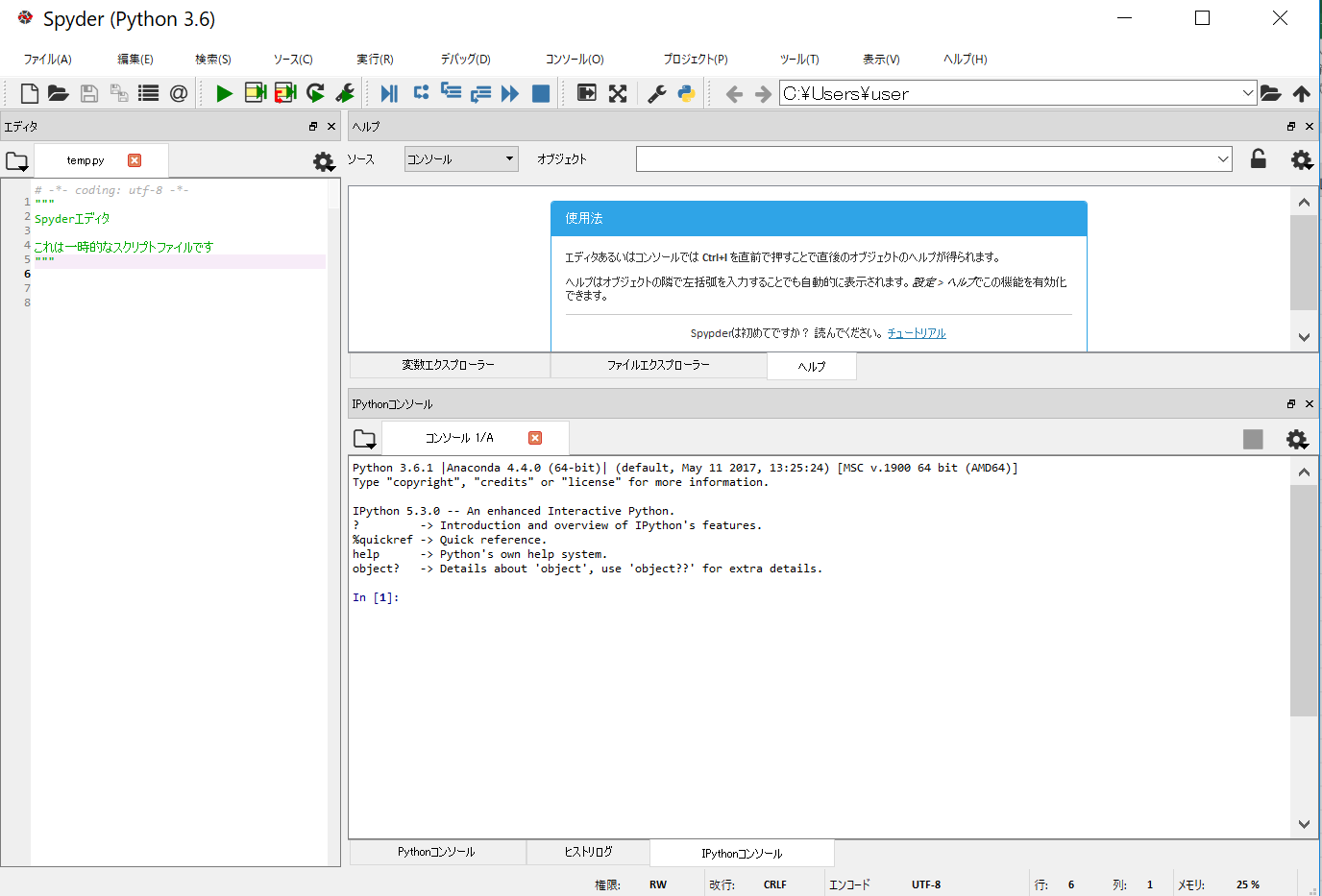
* **ステップ3 (Step 3)**

SQL演習

(SQL exercise)

①　スタートメニューの「**Anaconda3 (64bit)**」の下の「**spyder**」を起動する．右下に IPython コンソール・ウインドウがある．

　　Launch “spyder” in the “Anaconda3 (64bit)” in a start menu. There is an IPython console windows in right lower.



**②** IPython コンソール・ウインドウで，**次の Python プログラムを実行させてみなさい**

**SQLite3データベースに接続する**

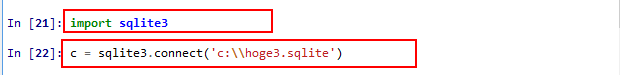
(connect to a SQLite3 database using IPython console)

**◆** 今日の演習ではSQLite3 データベースファイル名は **c:\hoge3.sqlite**

(SQLite3 database file name is '**c:\hoge3.sqlite**')

**import sqlite3**

**c = sqlite3.connect('c:\\hoge3.sqlite')**



**③** IPython コンソール・ウインドウで，**次の Python プログラムを実行させてみなさい**

**テーブルをすべて読みだす．カーソルを使う．**

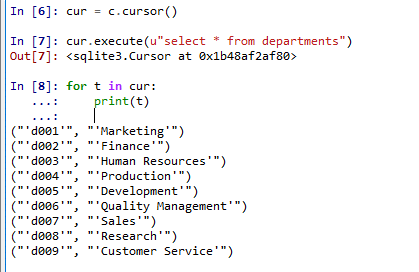
　(read a relational table). (Use a cursor)

**cur = c.cursor()**

**cur.execute(u"select \* from departments")**

**for t in cur:**

**print(t)**

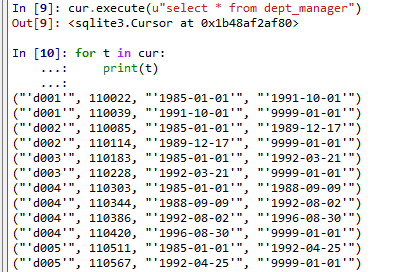


次も行いなさい

**cur.execute(u"select \* from dept\_manager")**

**for t in cur:**

**print(t)**



部門番号 d001 のマネージャは

1991年10月1日に交替したこと

などが記録されている

**④** IPython コンソール・ウインドウで，**次の Python プログラムを実行させてみなさい**

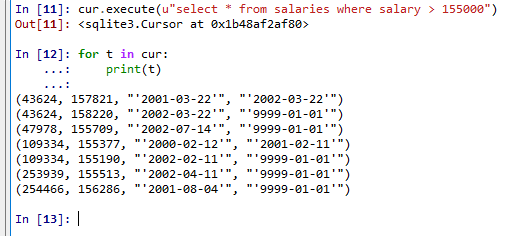
**条件に合致する行を読みだす**

(fetch rows using a condition)

**cur.execute(u"select \* from salaries where salary > 155000")**

**for t in cur:**

**print(t)**



**⑤** IPython コンソール・ウインドウで，**次の Python プログラムを実行させてみなさい**

**結合問い合わせ　(join query)**

　salary > 155000 であるような従業員の氏名などを得たい。SQL のプログラムを**複数行で書きたい**ので「**"""**」を使う．

　We want to get employees name etc. that satisfy the condition "salary > 155000". We use **"""** because SQL program is multiple lines.

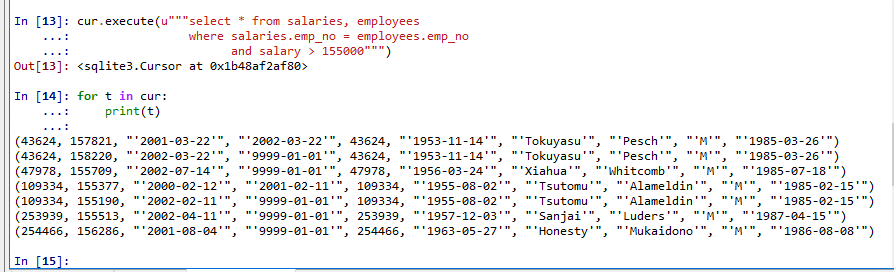
**cur.execute(u"""select \* from salaries, employees**

**where salaries.emp\_no = employees.emp\_no**

**and salary > 155000""")**

**for t in cur:**

**print(t)**



次も行いなさい

**cur.execute(u"""select \* from salaries, titles**

**where salaries.emp\_no = titles.emp\_no**

**and salary > 155000""")**

**for t in cur:**

**print(t)**

