データベースシステム（9）・（10）

SQL演習

（問い合わせ）



（1） Windowsにログインしてください．

（2） コマンドプロンプトを起動し，
作業フォルダに移動してください．

（3） データベースに接続してください．
（データベース名は「<自分のログイン名>.db」）

➔ sqlite3 ueto.db

SQLite version 3.6.1
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>

これが接続状態のプロンプト
接続状態で「SQL文」，「SQLiteのコマンド」を実行できます．



Windowsコマンド（「コマンドプロンプト」で使用）

| | |
|---|---|
| dir | 現在のフォルダ内のファイルのリスト<br>を表示する |
| cd | 現在のどのフォルダにいるかを表示 |
| cd <d> | <d>で示されるフォルダへの移動 例：cd db<br>（<d>が「..」のときは一つ上のフォルダ<br>への移動となる 例：cd ..） |
| <d>: | <d>ドライブへ移動 例： z: |
| del <f> | <f>で示されるファイルを削除 例：del foo.sql |
| mkdir <d> | <d>で示される名前のフォルダを作成 |
| rmdir <d> | <d>で示される名前のフォルダを削除 |



SQLiteコマンド（データベースに接続した状態で使用）

| | |
|---|---|
| .read <file> | <file>に書かれたSQL文を実行 |
| .header ON | 問い合わせ結果表の列名を表示 |
| .mode tab | 問い合わせ結果表をTSV形式で表示 |
| .mode csv | 問い合わせ結果表をCSV形式で表示 |
| .tables | データベース内の全テーブル名のリスト |
| .schema <table> | テーブルのスキーマ |
| .output <file> | 問い合わせ結果表をファイルに出力 |
| .output stdout | 問い合わせ結果表をコマンドプロンプトに表示 |
| .import <file> <table> | ファイルのデータをテーブルに読み込む |
| .quit | SQLiteを終了 |



リレーション（テーブル）作成

```
CREATE TABLE Students
        (sid CHAR(20),
         name CHAR(20),
         login CHAR(10),
         age INTEGER,
         gpa REAL)
```

| | |
|---|---|
| CHAR(n) | n文字からなる固定長文字列 |
| VARCHAR(n) | 最大n文字の可変長文字列 |
| INTEGER | 整数 |
| REAL | 実数 |

リレーションの削除

```
DROP TABLE  Students
```

「学生」リレーションのスキーマ情報と
タップル（のすべて）が削除される



SQLによる問い合わせの基本的な記述方法

```
SELECT   <SELECTリスト>
FROM     <FROMリスト>
WHERE    <選択条件>
```

<FROMリスト>
問い合わせの対象たるテーブル名のリスト

<選択条件>
（AND, OR, NOTで組み合わされた）真偽を返す演算

<SELECTリスト>
FROMリストのなかにあるテーブルの属性のリスト

Studentsテーブル

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53688 | Smith | smith@ee | 18 | 3.2 |
| 53650 | Smith | smith@ee | 19 | 3.8 |

**18歳である学生の名前とログイン名は？**

```
SELECT  name, login
FROM    Students
WHERE  age = 18
```

| name | login |
|---|---|
| Jones | jones@cs |
| Smith | smith@ee |

**WHEREが「選択（σ）」演算**
**SELECTが「射影（π）」演算**



## 行（タップル）の追加

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53650 | Smith | smith@math | 19 | 3.8 |

```
INSERT INTO  Students (sid, name, login, age, gpa)
VALUES  (53688, 'Smith', 'smith@ee', 18, 3.2)
```

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53650 | Smith | smith@math | 19 | 3.8 |
| 53688 | Smith | smith@ee | 18 | 3.2 |



## 行（タップル）の削除

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53650 | Smith | smith@math | 19 | 3.8 |
| 53688 | Smith | smith@ee | 18 | 3.2 |

```
DELETE
FROM   Students
WHERE   name = 'Smith'
```

where**節に該当するデータをすべて削除**
（**この場合**，Smith**という名前の行をすべて削除**）

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |

where**節は省略できる．その場合，テーブルの**
**すべての行が削除される**



## 行（タップル）の変更

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53650 | Smith | smith@math | 19 | 3.8 |
| 53688 | Smith | smith@ee | 18 | 3.2 |

```
UPDATE Students
SET age = age + 1
WHERE name = 'Smith'
```

where**節に該当する**
**データをすべて変更**

| sid | name | login | age | gpa |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53650 | Smith | smith@math | 20 | 3.8 |
| 53688 | Smith | smith@ee | 19 | 3.2 |

where**節は省略できる．その場合，テーブルの**
**すべての行が変更される**



## SQL問い合わせ



## SQLによる問い合わせの基本的な記述方法

```
SELECT   <SELECTリスト>
FROM     <FROMリスト>
WHERE    <選択条件>
```

<FROMリスト>
**問い合わせの対象たるテーブル名のリスト**

**<選択条件>**
**（AND, OR, NOTで組み合わされた）真偽を返す演算**

<SELECTリスト>
**FROMリストのなかにあるテーブルの属性のリスト**

## SQL問い合わせの（概念的）評価手順

```
SELECT    [DISTINCT]  <SELECTリスト>
FROM      <FROMリスト>
WHERE     <選択条件>
```

(1) <FROMリスト>のテーブルの直積 (cross-product) を計算
(2) <選択条件>を各行に適用し，偽ならばその行を削除
(3) <SELECTリスト>にない属性を削除
(4) もしDISTINCTが指定されているなら重複行を除く



## SQL問い合わせの（概念的）評価手順

「103番のボートを予約した船乗りの名前を挙げよ」

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=103
```

Sailors

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

Reserves

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 22 | 101 | 10/10/96 |
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |



## SQL問い合わせの（概念的）評価手順

```
SELECT    [DISTINCT]  <SELECTリスト>
FROM      <FROMリスト>
WHERE     <選択条件>
```

(1) <FROMリスト>のテーブルの直積 (cross-product) を計算
(2) <選択条件>を各行に適用し，偽ならばその行を削除
(3) <SELECTリスト>にない属性を削除
(4) もしDISTINCTが指定されているなら重複行を除く

この順番で得られる答えがSQL問い合わせの
答えとなるが，一般的にこの手順は効率的ではない！
DBMSの最適化機能は同じ答えを求める
「より効率的な手順で」SQL問い合わせを処理する



## SQL問い合わせ内のリレーションの別名

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=103
```

例にあげたSQLではテーブル名に別名を
つけて記述した．

この場合，別名を使わなくても書ける

```
SELECT sname
FROM   Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid
       AND bid=103
```

<FROMリスト>に同じリレーションが2回でるときには
別名が必要（2回書かないといけない場合もある！）



## SQLの練習問題

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45.0 |
| 29 | Brutus | 1 | 33.0 |
| 31 | Lubber | 8 | 55.5 |
| 32 | Andy | 8 | 25.5 |
| 58 | Rusty | 10 | 35.0 |
| 64 | Horatio | 7 | 35.0 |
| 71 | Zorba | 10 | 16.0 |
| 74 | Horatio | 9 | 35.0 |
| 85 | Art | 3 | 25.5 |
| 95 | Bob | 3 | 63.5 |

Sailors

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/98 |
| 22 | 102 | 10/10/98 |
| 22 | 103 | 10/8/98 |
| 22 | 104 | 10/7/98 |
| 31 | 102 | 11/10/98 |
| 31 | 103 | 11/6/98 |
| 31 | 104 | 11/12/98 |
| 64 | 101 | 9/5/98 |
| 64 | 102 | 9/8/98 |
| 74 | 103 | 9/8/98 |

Reserves

| bid | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

Boats



前頁のテーブルを自分のDBに作ってください．
以下のようなスキーマにしてください．

```
create table Sailors(
sid char(2),
sname char(8),
rating integer,
age real
);
```

```
create table Reserves(
sid char(2),
bid char(3),
day char(10)
);
```

```
create table Boats(
bid char(3),
bname char(10),
color char(6)
);
```

テーブルを作ったら以下のCSVファイルからデータを
インポートしてください．

sailors.csv
reserves.csv
boats.csv

はそれぞれ，Sailors表，Reserves表，Boats表のデータです．

19

SQL問い合わせ

20

「少なくとも1回ボートを予約した船乗りのsnameは」

以下のSQL問い合わせを作成し実行結果を確認せよ

SQLの最後にセミコロンを忘れないように！

```
SELECT  S.sname
FROM  Sailors S, Reserves R
WHERE  S.sid=R.sid
```

| sname |
|---|
| Dustin |
| Dustin |
| Dustin |
| Dustin |
| Lubber |
| Lubber |
| Lubber |
| Horatio |
| Horatio |
| Horatio |

このSQLにDISTINCTを加えるとどうなるか確認せよ

```
SELECT  DISTINCT S.sname
FROM  Sailors S, Reserves R
WHERE  S.sid=R.sid
```

| sname |
|---|
| Dustin |
| Lubber |
| Horatio |

データベースシステム（担当：森本康彦@広島大 2009/2/12-14）

21

「赤のボートを予約した船乗りのsidを挙げよ」

以下のSQL問い合わせを作成し実行結果を確認せよ

```
SELECT DISTINCT R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid  AND  B.color='red'
```

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/98 |
| 22 | 102 | 10/10/98 |
| 22 | 103 | 10/8/98 |
| 22 | 104 | 10/7/98 |
| 31 | 102 | 11/10/98 |
| 31 | 103 | 11/6/98 |
| 31 | 104 | 11/12/98 |
| 64 | 101 | 9/5/98 |
| 64 | 102 | 9/8/98 |
| 74 | 103 | 9/8/98 |

Reserves

| bid | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

Boats

| sid |
|---|
| 22 |
| 31 |
| 64 |

22

「赤または緑のボートを予約した船乗りのsidを挙げよ」

以下のSQL問い合わせを作成し実行結果を確認せよ

```
SELECT DISTINCT R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
  AND (B.color='red' OR B.color='green')
```

| sid |
|---|
| 22 |
| 31 |
| 64 |
| 74 |

この問い合わせのOR（赤文字の部分）を
ANDにすると答えはどうなる？

→ 空集合となる

「赤と緑の両方のボートを予約した船乗りの
sidを挙げよ」という場合はどうするのか考えよ！

23

「赤または緑のボートを予約した船乗りのsidを挙げよ」

集合演算の例

その属性の数と型が同じであるような
2つの表に対しては
集合演算（和：UNION，共通：INTERSECTION，差：EXCEPT）
を計算することができる

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
          AND B.color='red'
UNION
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
          AND B.color='green'
```

| sid |
|---|
| 22 |
| 31 |
| 64 |
| 74 |

| sid |
|---|
| 64 |

この問い合わせのUNION（赤文字の部分）を
EXCEPT（差集合）にすると答えはどうなる？

24

「赤と緑のボートを予約した船乗りのsidを挙げよ」

```
SELECT DISTINCT R1.sid
FROM  Boats B1, Reserves R1,
        Boats B2, Reserves R2
WHERE  R1.sid=R2.sid
  AND R1.bid=B1.bid AND R2.bid=B2.bid
  AND (B1.color='red' AND B2.color='green')
```

| sid |
|---|
| 22 |
| 31 |

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color='red'
INTERSECT
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color='green'
```

共通(INTERSECTION)で計算した例



「赤と緑のボートを予約した船乗りのsidを挙げよ」

SELECT DISTINCT R1.sid
FROM Boats B1, Reserves R1,
Boats B2, Reserves R2
WHERE R1.sid=R2.sid
AND R1.bid=B1.bid AND R2.bid=B2.bid
AND (B1.color='red' AND B2.color='green')

| sid |
|---|
| 22 |
| 31 |

R1×B1部分

| R1.sid | bid | … | B1.color |
|---|---|---|---|
| 22 | 102 | … | red |
| 22 | 104 | | red |
| 31 | 102 | | red |
| 31 | 104 | | red |
| 64 | 102 | | red |

R2×B2部分

| R2.sid | bid | … | B2.color |
|---|---|---|---|
| 22 | 103 | … | green |
| 31 | 102 | | green |
| 74 | 102 | | green |



<選択条件>の書き方

例　name = '広末'　　age > 18

Students.sid = Enrolled.sid

属性 op 定数　属性1 op 属性2

opには　=, <>, <, >, >=, <=　などの比較演算子が使える

AND,OR,NOTで組み合わせることもできる

name = 'さとみ' OR name = 'まお' AND age > 20

NOT (name = 'さとみ') AND age > 20

他に「集合比較演算子」「文字列マッチング演算子」が使える
(これらの解説は後)



文字列(String)マッチング

文字列マッチングの使い方の例

```
SELECT  S.sname, S.age
FROM  Sailors S
WHERE  S.sname LIKE 'B%'
```

| sname | age |
|---|---|
| Brutus | 33 |
| Bob | 63.5 |

「最初がBであるような名前をもつ船乗りの年齢を挙げよ」

「LIKE」は文字列マッチングの演算子
- 「_」は任意の1文字にマッチ
- 「%」は任意の(0字あるいはそれ以上の)文字列にマッチ

例えば S.sname LIKE 'B_b' の場合

「最初がBで始まり，最後がbで終わる3文字の名前」



「関係演算」は入力も出力もテーブル

「関係演算」を組み合わせることができる

「SQL問い合わせ」は入力も出力もテーブル

「SQL問い合わせ」を組み合わせることができる

入れ子問い合わせ(Nested Query)



「SQL問い合わせ」を組み合わせることができる

```
SELECT S.sid, S.sname, R.bid, B.color
FROM Sailors S, Reserves R, Boats B
WHERE S.sid=R.sid and R.bid=B.bid;
```

| sid | sname | bid | color |
|---|---|---|---|
| 22 | Dustin | 101 | blue |
| 22 | Dustin | 102 | red |
| 22 | Dustin | 103 | green |
| 22 | Dustin | 104 | red |
| 31 | Lubber | 102 | red |
| 31 | Lubber | 103 | green |
| 31 | Lubber | 104 | red |
| 64 | Horatio | 101 | blue |
| 64 | Horatio | 102 | red |
| 74 | Horatio | 103 | green |

この問い合わせ結果に対する問い合わせを書ける.

## 入れ子問い合わせ (Nested Query)

```
SELECT SUB.sid
FROM (
      SELECT S.sid as sid, S.sname as name, R.bid as bid, B.color as col
      FROM Sailors S, Reserves R, Boats B
      WHERE S.sid=R.sid and R.bid=B.bid
        ) SUB
WHERE SUB.col='red';
```

| sid | name | bid | col |
|---|---|---|---|
| 22 | Dustin | 101 | blue |
| 22 | Dustin | 102 | red |
| 22 | Dustin | 103 | green |
| 22 | Dustin | 104 | red |
| 31 | Lubber | 102 | red |
| 31 | Lubber | 103 | green |
| 31 | Lubber | 104 | red |
| 64 | Horatio | 101 | blue |
| 64 | Horatio | 102 | red |
| 74 | Horatio | 103 | green |

| sid |
|---|
| 22 |
| 22 |
| 31 |
| 31 |
| 64 |

31

## 集合比較演算子について

y IN x　　値yがxの部分集合なら真
EXISTS x　　xが空集合でなければ真
UNIQUE x　　xに重複がなければ真

xは副問い合わせ，あるいは(値の)集合

NOT IN, NOT EXISTS, NOT UNIQUEは
それぞれの演算結果の否定



## 集合比較演算子は
## 入れ子構造の問い合わせと組み合わせて使うことが多い

「103番のボートを予約した船乗りの名前を挙げよ」

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN  (SELECT  R.sid
                  FROM  Reserves R
                  WHERE  R.bid=103)
```

| sid |
|---|
| 22 |
| 31 |
| 74 |

→

| sname |
|---|
| Dustin |
| Lubber |
| Horatio |

WHERE節 (<選択条件>部分) のなかにも
SQL問い合わせそのものを書くことができる！

入れ子構造の問い合わせは内側の問い合わせから処理



「103番のボートを予約した船乗りの名前を挙げよ」

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN  (SELECT  R.sid
                  FROM  Reserves R
                  WHERE  R.bid=103)
```

内側の結果 →

| sid |
|---|
| 22 |
| 31 |
| 74 |

「103番のボートを予約していない船乗りの名前を挙げよ」

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid NOT IN  (SELECT  R.sid
                      FROM  Reserves R
                      WHERE  R.bid=103)
```

| sname |
|---|
| Brutus |
| Andy |
| Rusty |
| Horatio |
| Zorba |
| Art |
| Bob |



「赤と緑のボートを両方予約した船乗りのsnameを挙げよ」

SELECT節のS.sidをS.snameに置き換えればいいのか？

```
SELECT  S.sname
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
INTERSECT
SELECT  S.sname
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='green'
```

×

64のHoratioは赤
74のHoratioは緑

```
SELECT  DISTINCT S.sname
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
          AND S.sid IN  (SELECT  R2.sid
                         FROM  Boats B2, Reserves R2
                         WHERE  R2.bid=B2.bid
                                   AND  B2.color='green')
```

snameのように同値がある可能性がある属性の扱いには注意

35