データベースシステム（3）

SQL問い合わせ

データベースに必須の機能

- データ定義機能：スキーマの定義
- データ操作機能：問合せ・追加・削除・変更

関係データベースは表形式（関係モデル）で記録されたデータを管理するソフトウエアパッケージ

関係データベースは表形式（関係モデル）で記録されたデータに対する（関係代数に基づく）強力な問い合わせ言語をもっている

関係データベースでは，
・データ操作言語（DML，Data Manipulation Language）
・データ記述言語（DDL，Data Description Language）
は標準化されている．
「SQL-99」
（どんな関係データベースでもSQLを使った問い合わせ・操作・記述が使えます）

「スキーマ」は
データ記述言語（Data Description Language, DDL）で定義される

```
CREATE TABLE  Students
(sid char(5), sname char(20), login char(40),
                    age integer, gpa real)
```

リレーションの中身であるデータは
データ操作言語（Data Manipulation Language, DML）で操作（問い合わせ，追加，削除，変更）する

DBの内容は変わらない

```
SELECT  sname, gpa
FROM    Students
WHERE  sid = 25322
```

DBの内容が変化する

関係モデルに対するDDL，DMLはSQLとして標準化されている

DDL

リレーション（テーブル）作成

学生（Students）テーブル作成例

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

sid	name	login	age	gpa

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

リレーションの変更

```
ALTER TABLE  Students
      ADD COLUMN  firstYear  integer
```

「学生」リレーションに新たなカラム（属性）を追加

（すでにデータがある場合，追加された列の値は空値 null valueとなる）

DML

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！）

ID	name	salary	dept	manager
0650	ichiro	50	K55	0650
1508	ayu	40	K41	1508
0231	hikaru	60	K41	1508
2034	beckham	40	K55	0650

Employee
リレーション

上司より給与の高い社員の社員番号とその上司の社員番号をあげよ

```
SELECT  E1.id, E2.id
FROM    Employee E1, Employee E2
WHERE   E1.manager=E2.id AND E1.salary > E2.salary
```

答え

0231	1508

このような演算を
自己結合 (Self-Join) **と呼ぶ**

Employee**である社員を**E1**に**
その上司のEmployee**である社員を**E2**に**
<**選択条件**>**は行に対する真偽を判定する**
E1,E2**が**1**行にまとまっていないと判定できない**

19

E1.ID	E1.name	E1.salary	E1.dept	E.1manager	E2.ID	E2.name	E2.salary	...
0650	ichiro	50	K55	0650	0650	ichiro	50	...
1508	ayu	40	K41	1508	1508	ayu	40	...
0231	hikaru	60	K41	1508	1508	ayu	40	...
2034	beckham	40	K55	0650	0650	ichiro	50	...

上司より給与の高い社員の社員番号とその上司の社員番号をあげよ

```
SELECT  E1.id, E2.id
FROM    Employee E1, Employee E2
WHERE   E1.manager=E2.id AND E1.salary > E2.salary
```

答え

0231	1508

20

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
```

(例のように) 別名が必要でない場合も
別名を使うほうが読みやすいので
(玄人はとくに) いつも別名を使う人が多い！

<**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」**

23