MySQL を使う v1.3

Seiichi Nukayama

2021-10-09

目次

1　データベースを設計する

1.1　扱うデータ

以下のようなデータを扱うこととする。

菅原文太	千葉真一	北大路欣也	梶芽衣子
40 歳	34 歳	30 歳	26 歳
1933 年生まれ	1939 年生まれ	1943 年生まれ	1947 年生まれ
総務部	営業部	経理部	営業部

あなたがプログラマで、上のような社員名簿アプリを作成することになったとする。PHP か Java でアプリを作成することになる。クライアントの会社の総務部がこのアプリを使うことになる。そのアプリには社員の登録画面、一覧画面、編集画面、削除画面などがあるだろう。そういった画面と処理をあなたは作らなければならない。

そのときに、データを保存するしくみとして、データベースを使うことになる。かりに PHP でプログラミングするならば、PHP という言語を使ってデータベースを操作することになる。

1.2　primary key

データベースにデータを格納する際には、そのデータに primary key (独自キー) が必要となる。primary key とは、そのデータを他と区別するためのデータである。菅原文太というデータは、この 4 つの中では独自であるが、他のデータを追加する際に、同じデータに出会う可能性 (同姓同名) を排除できない。さらに日本語である以上、文字コードの問題を避けることもできない。つまり、同じ菅原文太という文字でも UTF-8 と Shift_JIS では別物と判定されるのである。

となると、この 4 つのデータには primary key となるものがないということになる。

このような場合、データベースの設計者が primary key を追加することになる。ここでは 数字を primary key として追加する。つまり、菅原文太は 1、千葉真一は 2 というふうにする。

1.3　表を分ける

4 つの各データには、総務部、営業部、経理部 という部署名がはいっている。これらは、部署データとして、別の表で管理するのが自然である。そして、各人のデータから部署データを参照しているというふうにするのがよい。

XX 会社 部署一覧

総務部
営業部
経理部
開発部

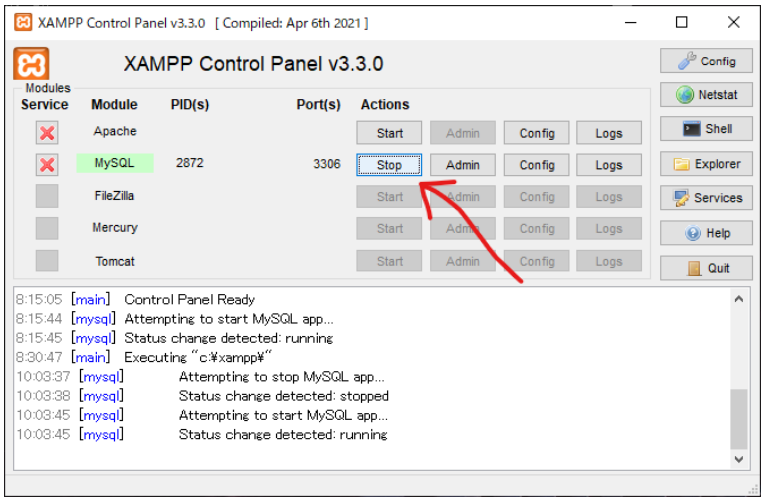
2　データベースを作成する

2.1　ユーザーを作成して、データベースを作成する

2.1.1　MySQL の起動

まず、MySQL を起動しなくてはならない。

1. XAMPP コントロールパネルを管理者として起動する。
2. MySQL の行の Start ボタンをクリックして MySQL を起動する。



2.1.2　root ユーザーでログインする

データベースを作成するために、まずそのデータベースを扱うことのできるユーザーを作成する。

ユーザーを作成するために、まず管理者 (root) でログインする。MariaDB の場合、以下の手順でログインできる。

コマンドプロンプトを起動して、以下のコマンドを入力する。

```
＞mysql -u root -p (Enter キー)
＞Enter password: (何も入力せず、Enter キー)
```

これで、4 行ほどのメッセージと、次のプロンプトが表示される。

```
MariaDB [(none)]＞
```

2.1.3 **ユーザーの作成と権限の付与**

以下のコマンドで sampleuser を作成し、sampleuser というパスワードを設定し、sample データベースへの権限を与えることができる。[*1] ここでは、ユーザー名を sampleuser、パスワードを sampleuser としている。

```
MariaDB [(note)]> GRANT ALL ON sample.* TO 'sampleuser'@'localhost'
    -> IDENTIFIED BY 'sampleuser';
```

GRANT ALL ON データベース名.* TO ' ユーザー名'@'localhost' IDENTIFIED BY ' パスワード';

sample というデータベースを作成すると、いくつかファイルを作成することになるので、それら全部に権限を与えるため、sample.* としている。

※ sample.(**ドット**)*(**アスタリスク**)

これで root としての仕事は終了である。exit あるいは quit でログアウトする。

```
MariaDB [(none)]> exit
```

2.1.4 **作成したユーザーでログインし、データベースを作成する**

作成したユーザー sampleuser でログインする。

```
> mysql -u sampleuser -p (Enter キー)
> Enter password: ********** (sampleuser と入力)
(... 省略 ...)
MariaDB [(none)]>
```

mysql -u ユーザー名 -p
Enter password: パスワード

データベース sample を作成する。

```
MariaDB [(none)]> create database sample;
```

CREATE DATABASE データベース名 ;

これで、この作成したデータベース sample は、sampleuser ユーザーでアクセスできる。(もちろん root ユーザーもアクセスできる)

[*1] ユーザーの作成と権限付与を別々にすることもできる。
sampleuser というユーザーの作成と sampleuser というパスワードの設定
MariaDB [(note)]> CREATE USER 'sampleuser'@'localhost' IDENTIFIED BY 'sampleuser';
sampleuser に sample データベー Y スへの権限を付与する。
MariaDB [(none)]> GRANT ALL [PRIVILEGES] ON sample.* TO 'sampleuser'@'localhost';

3　テーブル (表) を作成する

3.1　作成する表のイメージ

以下のような表を作成することとする。

表 1　emp

ID	名前	年齢	誕生年	部署 ID
1	菅原文太	40	1933	001
2	千葉真一	34	1939	002
3	北大路欣也	30	1943	003
4	梶芽衣子	26	1947	002

表 2　dept

ID	部署名
001	総務部
002	営業部
003	経理部
004	開発部

そして、上の 2 つの表から、以下の結合表を表示することとする。

ID	名前	年齢	部署名
1	菅原文太	40	総務部
2	千葉真一	34	営業部
3	北大路欣也	30	経理部
4	梶芽衣子	26	営業部

3.2 テーブルの定義

テーブルの定義を決める。

表 3　emp テーブルの定義

項目	型	オプション
id	int	primary key, auto_increment
name	varchar(20)	not null
age	int	not null
birthday	year	not null
dept_id	char(3)	

型

int 型	整数。これがよく使われる。
varchar 型	可変長の文字列型。ここでは最大 20 文字としている。(全角文字を使った場合)
year 型	年のみを扱う型。誕生の年だけを入力する。
char 型	固定長の文字型。ここで半角で 3 文字としている。

オプション

primary key	項目 id をデータの識別に使う。重複する値がないことが保証される。
auto_increment	自動連番。自動的に順に番号を振ってくれる機能を使う。
not null	入力が必須。もしも入力しなかったら エラー になる。

最後の dept_id を not null にしなかったのは、部署 ID のない社員もいるかもしれないからである。

dept テーブルは、このような定義になる。

表 4　dept テーブルの定義

項目	型	オプション
id	char(3)	primary key
name	varchar(20)	not null

今回の場合、emp テーブルには dept_id が入っている。これは、dept テーブルの id のことである。このことにより、emp テーブルと dept テーブルを結合させることができる。

このときの emp テーブルの dept_id のことを “**外部キー**” という。

3.3 テーブルの作成

テーブルを作成する前に、データベースの使用を宣言する。

```
MariaDB [(none)]> USE sample;
```
use データベース名 ;

以下のコマンドにより emp テーブルを作成できる。

```
MariaDB [sample]> CREATE TABLE emp (
    -> id INT AUTO_INCREMENT, (カンマ)
    -> name VARCHAR(20) NOT NULL,
    -> age INT NOT NULL,
    -> birthday YEAR NOT NULL,
    -> dept_id CHAR(3),
    -> PRIMARY KEY (id) (カンマなし)
    -> );
```

同様に dept テーブルも作成する。[*2]

```
MariaDB [sample]> CREATE TABLE dept (
    -> id CHAR(3) PRIMARY KEY, (カンマ)
    -> name VARCHAR(20) NOT NULL (カンマなし)
    -> );
```

※　作成したテーブルの構造は以下のコマンドで確認できる。

MariaDB [sample]> desc emp;

```
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(20) | NO   |     | NULL    |                |
| age      | int(11)     | NO   |     | NULL    |                |
| birthday | year(4)     | NO   |     | NULL    |                |
| dept_id  | char(3)     | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
```

また、テーブルを作成したときのコマンドは以下で確認できる。

MariaDB [sample]> show create table emp;

```
...(省略)...
CREATE TABLE `emp` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
```

[*2] primary key の指定は、id の指定のときに書くこともできるし、別項目に分けて書くこともできる。

```
  `age` int(11) NOT NULL,
  `birthday` year(4) NOT NULL,
  `dept_id` char(3) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

3.4 データの登録

データの登録は、以下のコマンドでできる。

```
MariaDB [sample]> INSERT INTO emp (name, age, birthday, dept_id) VALUES (' 菅原文太', 40, 1933, '001');
```

INSERT INTO テーブル名 (カラム名 1, カラム名 2, カラム名 3, ...) VALUES (データ 1, データ 2, データ 3, ...)

※ 各データの区切りは ,(カンマ)

"id" は auto_increment なので、指定しない。

また、dept_id は char(3) なので、'001' シングルクォーテーションを使って入力する。

画面の関係で一行で入力しづらければ、次のように二行で入力することもできる。

```
MariaDB [sample]> INSERT  INTO  emp (name, age, birthday, dept_id)
    -> VALUES  (' 千葉真一', 34, 1939, '002');
```

TeraPad などのエディタで記述しておいて、コピー&貼り付け ですることもできる。

以下のようにすると、一度で入力できてしまう。

```
MariaDB [sample]> INSERT  INTO  emp (name, age, birthday, dept_id) VALUES
    -> (' 北大路欣也', 30, 1943, '003'), (カンマ)
    -> (' 梶芽衣子', 26, 1947, '002');
```

これも、エディタに記述しておいて、コピー&貼り付けでも できる。

データの確認は次のコマンドでできる。

```
MariaDB [sample]> SELECT * FROM emp;
```

SELECT * FROM テーブル名

```
+----+------------+-----+----------+---------+
| id | name       | age | birthday | dept_id |
+----+------------+-----+----------+---------+
|  1 | 菅原文太   |  40 |     1933 | 001     |
|  2 | 千葉真一   |  34 |     1939 | 002     |
|  3 | 北大路欣也 |  30 |     1943 | 003     |
```

```
|  4 | 梶芽衣子     |  26 |      1947 | 002       |
+----+-------------+-----+-----------+-----------+
```

3.5 ファイル読込みによるデータの登録

同様に、dept テーブルについてもデータを登録する。

ただ、今度は 登録のための SQL 文を外部ファイルに記述しておいて、それを読み込むという方法で登録してみる。

3.5.1 作業のためのフォルダを用意して、そこでファイルをつくる。

ファイルを置くためのフォルダを用意する。ここでは仮に、ドキュメントフォルダに mysql というフォルダを作成したとする。

そこに、以下の内容のファイル "insert_dept.sql" を作成する。

リスト 1　insert_dept.sql

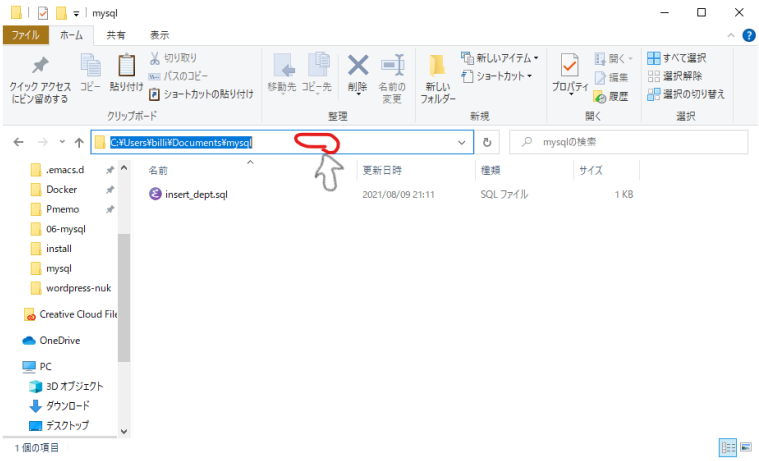
```
1 -- dept テーブル
2
3 INSERT INTO dept (id, name) VALUES
4 ('001', '総務部'),
5 ('002', '営業部'),
6 ('003', '経理部'),
7 ('004', '開発部');
```

-- で始まる行は、コメントである。[3]

また、SQL のコマンドは大文字で記述したほうがよい。コマンドプロンプトでは、小文字でかまわないが、このようにファイルとして記述する場合は、SQL のコマンド文字列は大文字で記述し、ユーザーが用意した変数などは小文字で記述しておく。あとで見なおしたりする場合にわかりやすい。

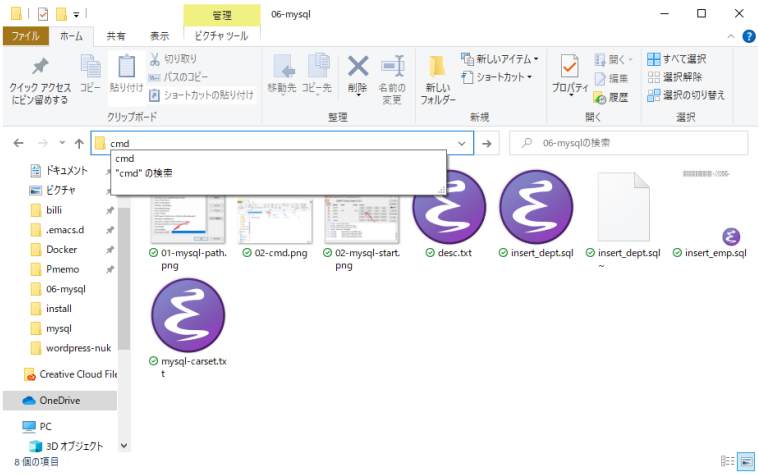
3.5.2 そのフォルダでコマンドプロンプトを起動する。

そのフォルダでコマンドプロンプトを起動する。次の図のようにする。

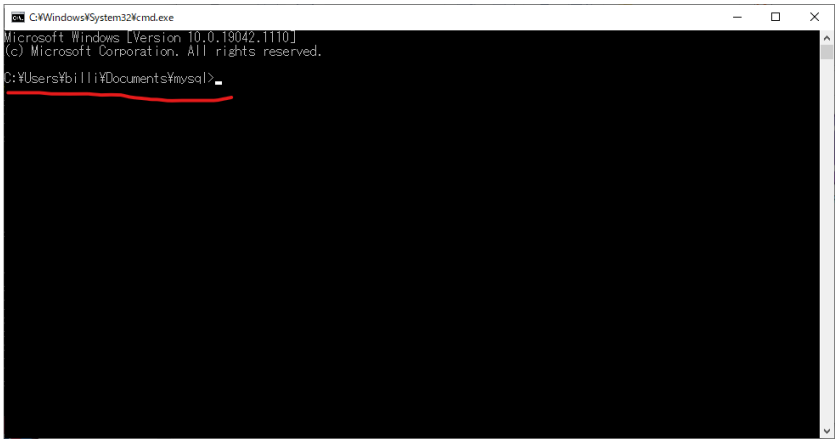


[3] ほかに、# で始まる行もコメント。また複数行は、/* ... */ が使える。

上の図のように、エクスプローラのアドレス欄の余白部分をクリックする。すると、現在のフォルダをあらわす文字列が青く反転する。



上の図のように、そこに "cmd" と入力して Enter キーを押下する。すると、そのフォルダでコマンドプロンプトが起動する。



上の図の赤い線の部分が、現在のフォルダになっている。

```
C:¥Users¥(ユーザー名)¥Documents¥mysql> dir
```

dir というコマンドを実行すると、現フォルダのファイルが一覧できる。insert_dept.sql があることがわかる。

```
2021/08/09  22:05    <DIR>          .                （このフォルダ）
2021/08/09  22:05    <DIR>          ..               （ひとつ上の階層）
2021/08/09  22:05               222 dept.sql
```

3.5.3 ファイルを読み込んで、SQL 文を実行する

ここで、mysql を起動する。

```
C:¥Users¥(ユーザー名)¥Documents¥mysql> mysql -u sampleuser -p
Enter password: **********
```

sample データベースの使用を宣言する。

```
MariaDB [(none)]> USE sample;
MariaDB [sample]>
```

次に、以下のコマンドで insert_dept.sql を実行できる。

```
MariaDB [sample]> source insert_dept.sql
```

あるいは、以下のような省略形もある。

```
MariaDB [sample]> ¥. insert_dept.sql
```

確認する。

```
MariaDB [sample]> SELECT * FROM dept;
```

```
+-----+--------+
| id  | name   |
+-----+--------+
| 001 | 総務部 |
| 002 | 営業部 |
| 003 | 経理部 |
| 004 | 開発部 |
+-----+--------+
```

読み込めているのがわかる。

3.6 テーブル作成からデータの登録までを自動化する

このことを応用して、テーブルの作成からデータの登録までを、ファイル読込みによって自動化することができる。

以下のような記述が考えられる。

リスト 2 init_data.sql

```
1 -- もし存在していなかったら sample データベースを作成する
2 CREATE DATABASE IF NOT EXISTS sample;
3
```

```
-- sample データベースを使用
USE sample;

-- emp テーブルの作成
-- もし empテーブルが存在していたら削除する。
-- その empテーブルの定義がこれから作成するのと同じという保証がないからである。

DROP TABLE IF EXISTS emp;

CREATE TABLE emp (
  id INT AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  age INT NOT NULL,
  birthday YEAR NOT NULL,
  dept_id CHAR(3),
  PRIMARY KEY (id)
);

-- dept テーブルの作成
-- もし deptテーブルが存在していたら削除する。
DROP TABLE IF EXISTS dept;

CREATE TABLE dept (
  id CHAR(3) PRIMARY KEY,
  name VARCHAR(20) NOT NULL
);

-- 自動連番を初期化する。
ALTER TABLE emp AUTO_INCREMENT = 1;

INSERT INTO emp (name, age, birthday, dept_id) VALUES
('菅原文太', 40, 1933, '001'),
('千葉真一', 34, 1939, '002') ,
('北大路欣也', 30, 1943, '003'),
('梶芽衣子', 26, 1947, '002');

INSERT INTO dept (id, name) VALUES
('001', '総務部'),
('002', '営業部'),
('003', '経理部'),
('004', '開発部');

SELECT * FROM emp;
SELECT * FROM dept;
```

これを実行する。

```
MariaDB [sample]> ¥. init_data.sql
```

これにより、いつでもデータを初期状態に戻すことができるようになった。

4　データベースをバックアップする

4.1　バックアップ

データの入力が終わったら、データベースをバックアップする。
ここでは、mysqldump を使っておこなう。
まず、MySQL をログアウトする。

```
MariaDB [sample]> exit (もしくは quit)
C:¥Users¥XXXXXX¥Documents¥mysql>
```

※ XXXXXX は、各自のユーザー名

コマンドプロンプトに戻る。
ここで以下のコマンドを実行する。

```
> mysqldump -u sampleuser -p --databases sample > sample_db.dump
```

`mysqldump -u ユーザー名 -p --databases データベース名 >保存ファイル名`

※ --databases は -(ハイフン)2 つ

ここでは保存ファイル名を sample_db.dump としたが、好みのファイル名を指定すればよい。
dir というコマンドを実行すると、ファイル一覧が見れる。
ファイル群の中に sample_db.dump があるはず。
これは テキストファイルなので、TeraPad などのエディタで内容を見ることができる。[*4]

4.2　データのリストア (復元)

以下のコマンドを実行する。[*5]

```
> mysql -u sampleuser -p < sample_db.dump
```

`mysql -u ユーザー名 -p < 保存したファイル名`

これでリストアができている。

[*4] また、*–databases* オプションをつけずにバックアップを取ることもできる。
`> mysqldump -u sampleuser -p sample > sample_db.dump`
この場合は、バックアップファイルの記述の中に、CREATE DATABASE 文がない。

[*5] –databases オプションをつけずにバックアップファイルを作成した場合は、まず sample データベースを作成してから
`> mysql -u sampleuser -p < sample_db.dump`
とするか、あるいは、sampleuser でログインしてから
`MariaDB[sample] > source sample_db.dump` とすればよい。

5　2 つのテーブルを結合する

5.1　内部結合 (JOIN 句)

emp テーブルの dept_id は、dept テーブルの id である。
だから、dept_id をキーにして、二つのテーブルを結合できる。
結合するには JOIN 句を使う。
実行例

```
MariaDB [sample]> SELECT * FROM emp JOIN dept ON emp.dept_id = dept.id;
```

```
SELECT * FROM メインの表 [INNER] JOIN サブの表
ON メインの表. キーカラム名 = サブの表. キーカラム名
```

- *(アスタリスク) — メインの表、サブの表のすべての項目を表示する。
- emp.dept_id — emp 表の dept_id カラム名。ピリオドで区切って指定。ピリオドの後に空白を入れてはいけない。
- dept.id — dept 表の id カラム名。

これで二つのテーブルが結合される。

```
+----+------------+-----+----------+---------+-----+--------+
| id | name       | age | birthday | dept_id | id  | name   |
+----+------------+-----+----------+---------+-----+--------+
|  1 | 菅原文太   |  40 |     1933 | 001     | 001 | 総務部 |
|  2 | 千葉真一   |  34 |     1939 | 002     | 002 | 営業部 |
|  4 | 梶芽衣子   |  26 |     1947 | 002     | 002 | 営業部 |
|  3 | 北大路欣也 |  30 |     1943 | 003     | 003 | 経理部 |
+----+------------+-----+----------+---------+-----+--------+
```

JOIN は INNER JOIN と記述できる。“内部結合” と呼ばれている。
ON emp.dept_id = dept.id は emp の dept_id と dept の id が等しければ、そのレコードを抜き出す。
※ emp.dept_id は emp テーブルの dept_id という意味になる。

5.2　表示項目を絞る

現在は項目を全て表示しているが、これを変更する。
emp 表の id, name, age と dept 表の name だけを表示させる。

```
MariaDB [sample]> SELECT emp.id, emp.name, age, dept.name FROM emp JOIN dept
    -> ON emp.dept_id = dept.id;
```

```
+----+------------+-----+--------+
| id | name       | age | name   |
```

```
+----+------------+-----+--------+
|  1 | 菅原文太   |  40 | 総務部 |
|  2 | 千葉真一   |  34 | 営業部 |
|  4 | 梶芽衣子   |  26 | 営業部 |
|  3 | 北大路欣也 |  30 | 経理部 |
+----+------------+-----+--------+
```

このように必要な項目のみ表示させることができる。ただ、name という項目が二つあったり、英語であったりするので、これを適切な日本語に変えることにする。

それには、AS **句** というのが使える。

たとえば、"emp.name AS 名前" とすると、"emp.neme" は "名前" と表示される。

```
MariaDB [sample]> SELECT emp.id AS ID, emp.name AS 名前, age AS 年齢,
    -> dept.name AS 部署名
    -> FROM emp JOIN dept
    -> ON emp.dept_id = dept.id;
```

```
+----+------------+------+--------+
| ID | 名前       | 年齢 | 部署名 |
+----+------------+------+--------+
|  1 | 菅原文太   |   40 | 総務部 |
|  2 | 千葉真一   |   34 | 営業部 |
|  4 | 梶芽衣子   |   26 | 営業部 |
|  3 | 北大路欣也 |   30 | 経理部 |
+----+------------+------+--------+
```

さらによく見てみると、この表は部署名の順に並んでいる。これを ID 順に並びかえる。

そのためには ORDER **句** というのが使える。

たとえば、今回の場合だと、ORDER BY emp.id [ASC] とすることで、ID 順になる。

ASC というのは" 昇順" という意味で、省略すると ASC と指定したことになる。

また、DESC と指定すると " 降順" で並びかえできる。

```
MariaDB [sample]> SELECT emp.id AS ID, emp.name AS 名前, age AS 年齢,
    -> dept.name AS 部署名
    -> FROM emp JOIN dept
    -> ON emp.dept_id = dept.id
    -> ORDER BY ID;
```

```
+----+------------+------+--------+
| ID | 名前       | 年齢 | 部署名 |
+----+------------+------+--------+
|  1 | 菅原文太   |   40 | 総務部 |
|  2 | 千葉真一   |   34 | 営業部 |
|  3 | 北大路欣也 |   30 | 経理部 |
|  4 | 梶芽衣子   |   26 | 営業部 |
+----+------------+------+--------+
```

ORDER BY emp.id とするところを ORDER BY ID としている。
これは、1 行目で emp.id AS ID としているので、ID を使うことができるのである。

5.3 テーブルの指定を簡略化する

emp とか dept とかのテーブルの指定も別名を使うことで簡略化できる。

```
MariaDB [sample]> SELECT e.id AS ID, e.name AS 名前, age AS 年齢,
    -> d.name AS 部署名
    -> FROM emp AS e JOIN dept AS d
    -> ON e.dept_id = d.id
    -> ORDER BY ID;
```

さらに、FROM や JOIN の後では、すぐ後ろに 別名 (ここでは e や d のこと) がくる場合、*AS* は省略できる。

```
MariaDB [sample]> SELECT e.id AS ID, e.name AS 名前, age AS 年齢,
    -> d.name AS 部署名
    -> FROM emp e JOIN dept d
    -> ON e.dept_id = d.id
    -> ORDER BY ID;
```

5.4 **外部結合** (LEFT OUTER JOIN / RIGHT OUTER JOIN)

5.4.1 **左外部結合** LEFT OUTER JOIN

この emp 表に次のデータを追加する。

ID : 5
名前 : 成田三樹夫
年齢 : 38
誕生年 : 1935
部署 ID : (なし)

```
MariaDB [sample]> INSERT INTO emp (name, age, birthday) VALUES
    -> ('成田三樹夫', 38, 1935);
```

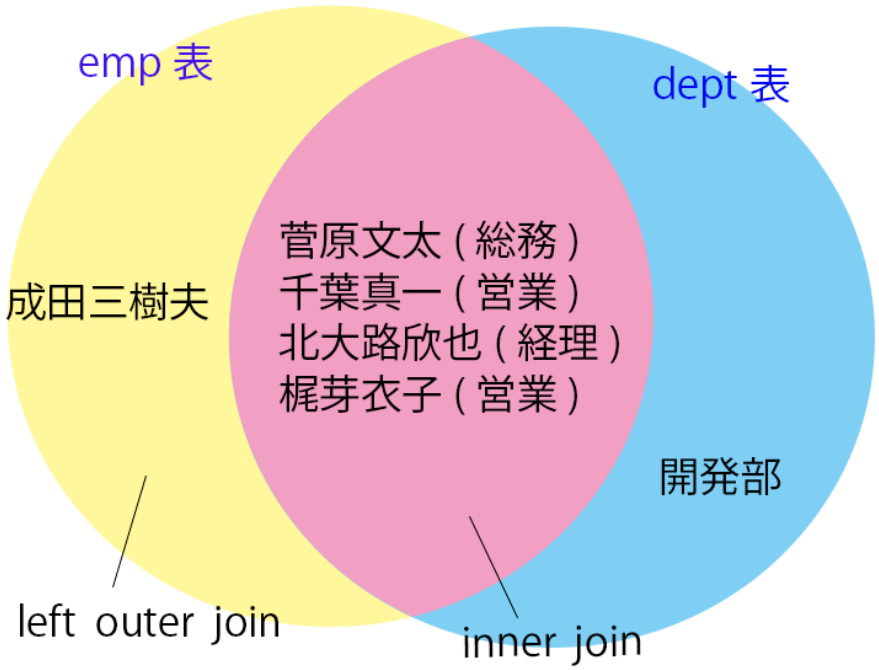
```
MariaDB [sample]> SELECT * FROM emp;
+----+------------+-----+----------+---------+
| id | name       | age | birthday | dept_id |
+----+------------+-----+----------+---------+
|  1 | 菅原文太   |  40 |     1933 | 001     |
|  2 | 千葉真一   |  34 |     1939 | 002     |
|  3 | 北大路欣也 |  30 |     1943 | 003     |
|  4 | 梶芽衣子   |  26 |     1947 | 002     |
|  5 | 成田三樹夫 |  38 |     1935 | NULL    |
+----+------------+-----+----------+---------+
```

このデータには dept_id、つまり部署 ID がない。たとえば社長とかの場合である。
この状態で 内部結合 をすると、どうなるか？

```
MariaDB [sample]> SELECT e.id AS ID, e.name AS 名前, e.age AS 年齢,
    -> d.name AS 部署名
    -> FROM emp e JOIN dept d
    -> ON e.dept_id = d.id
    -> order by ID;
```

```
+----+------------+------+--------+
| ID | 名前       | 年齢 | 部署名 |
+----+------------+------+--------+
|  1 | 菅原文太   |   40 | 総務部 |
|  2 | 千葉真一   |   34 | 営業部 |
|  3 | 北大路欣也 |   30 | 経理部 |
|  4 | 梶芽衣子   |   26 | 営業部 |
+----+------------+------+--------+
```

結合表には出てこない。
これを図であらわすと、このようになる。



成田三樹夫は部署 ID がないので結合の対象ではない。

こんなときは **左外部結合** *(LEFT [OUTER] JOIN)* を使う。

```
MariaDB [sample]> SELECT e.id AS ID, e.name AS 名前, e.age AS 年齢, d.name as 部署名
    -> FROM emp e LEFT JOIN dept d
    -> ON e.dept_id = d.id
    -> ORDER BY ID;
```

※ LEFT OUTER JOIN と記述することもできる

```
+----+------------+------+--------+
| ID | 名前       | 年齢 | 部署名 |
+----+------------+------+--------+
|  1 | 菅原文太   |   40 | 総務部 |
|  2 | 千葉真一   |   34 | 営業部 |
|  3 | 北大路欣也 |   30 | 経理部 |
|  4 | 梶芽衣子   |   26 | 営業部 |
|  6 | 成田三樹夫 |   38 | NULL   |
+----+------------+------+--------+
```

5.4.2 **右外部結合** RIGHT [OUTER] JOIN

また、dept 表をみてみると、id : '004' が **開発部** であるが、emp 表には dept_id が '004' である人はいない。

この状態で結合表をつくり、開発部という項目も表示させるには、次のようにする。

```
MariaDB [sample]> SELECT e.id AS ID, e.name AS 名前, e.age AS 年齢, d.name AS 部署名
    -> FROM emp e RIGHT JOIN dept d
    -> ON e.dept_id = d.id
    -> ORDER BY ID;
```

ID	名前	年齢	部署名
NULL	NULL	NULL	**開発部**
1	**菅原文太**	40	**総務部**
2	**千葉真一**	34	**営業部**
3	**北大路欣也**	30	**経理部**
4	**梶芽衣子**	26	**営業部**

6　制約

6.1　外部キー制約

6.1.1　表定義に制約をつけてみる

最初の表を入力するところに戻る。以下の表を作成して入力するのだった。

表 5　emp

ID	名前	年齢	誕生年	部署 ID
1	菅原文太	40	1933	001
2	千葉真一	34	1939	002
3	北大路欣也	30	1943	003
4	梶芽衣子	26	1947	002

表 6　dept

ID	部署名
001	総務部
002	営業部
003	経理部
004	開発部

emp 表のデータの " 部署 ID" を入力するとき、dept 表にない番号を入力するとまずいことになる。

そこで、emp 表の " 部署 ID" を入力するときに、dept 表にある番号だけを入力するように 制限 をかけることができる。

これを " 外部キー制約" という。

emp 表の dept_id に入力する値は dept 表にある値に制限するのであるから、emp 表を定義する前に dept 表が定義されていなくてはならない。

dept **表の定義 (再掲)**

```
MariaDB [sample]> CREATE TABLE dept (
    -> id CHAR(3), (カンマ)
    -> name VARCHAR(20) NOT NULL , (カンマ)
    -> PRIMARY KEY (id) (カンマなし)
    -> );
```

emp **表の定義 (外部キー制約)**

```
MariaDB [sample]> CREATE TABLE emp (
    -> id INT auto_increment,
    -> name VARCHAR(20) NOT NULL,
    -> age INT NOT NULL ,
    -> birthday YEAR NOT NULL ,
    -> dept_id CHAR(3), (カンマをつける)
    -> PRIMARY KEY (id) , (カンマをつける)
    -> FOREIGN KEY (dept_id) REFERENCES dept (id) (カンマなし)
    -> );
```

現在の emp テーブル、dept テーブルを削除して、再定義、初期データを入力する。そのためのスクリプトは、以下である。

リスト 3　reinit_data.sql

```
 -- もし存在していなかったら sample データベースを作成する
CREATE DATABASE IF NOT EXISTS sample;

-- sample データベースを使用
USE sample;

-- もし empテーブルが存在していたら削除する。。
DROP TABLE IF EXISTS emp;

-- もし deptテーブルが存在したら削除する。
-- empテーブルが存在していたら削除できないので、
-- （empテーブルがdeptテーブルを参照しているため）
-- empテーブルを先に削除しなくてはならない。
DROP TABLE IF EXISTS dept;

-- dept テーブルの作成
CREATE TABLE IF NOT EXISTS dept (
  id   CHAR(3),
  name VARCHAR(20) NOT NULL,
  PRIMARY KEY (id)
);

-- emp テーブルの作成
CREATE TABLE IF NOT EXISTS emp (
  id       INT         AUTO_INCREMENT,
  name     VARCHAR(20) NOT NULL,
  age      INT         NOT NULL,
  birthday YEAR        NOT NULL,
  dept_id  CHAR(3),
  PRIMARY KEY (id),
  FOREIGN KEY(dept_id) REFERENCES dept(id)
);

-- 自動連番を初期化する。
ALTER TABLE emp AUTO_INCREMENT = 1;

-- dept表の初期データ
```

```
38 INSERT INTO dept (id, name) VALUES
39 ('001', '総務部'),
40 ('002', '営業部'),
41 ('003', '経理部'),
42 ('004', '開発部');
43
44 -- emp表の初期データ
45 INSERT INTO emp (name, age, birthday, dept_id) VALUES
46 ('菅原文太',   40, 1933, '001'),
47 ('千葉真一',   34, 1939, '002') ,
48 ('北大路欣也', 30, 1943, '003'),
49 ('梶芽衣子',   26, 1947, '002');
50
51 SELECT * FROM dept;
52 SELECT * FROM emp;
```

このファイルを C:¥Users¥XXXXX¥Documents¥mysql に作成する。

そのフォルダで コマンドプロンプトを起動し、sampleuser ユーザーで mysql にログインする。

```
> mysql -u sampleuser -p
Enter password: ********
MariaDB [(none)]>
```

今作成したファイルを 読み込む。

```
MariaDB [(none)]> source reinit_data.sql
```

```
+-----+--------+
| id  | name   |
+-----+--------+
| 001 | 総務部 |
| 002 | 営業部 |
| 003 | 経理部 |
| 004 | 開発部 |
+-----+--------+
```

```
+----+------------+-----+----------+---------+
| id | name       | age | birthday | dept_id |
+----+------------+-----+----------+---------+
|  1 | 菅原文太   |  40 |     1933 | 001     |
|  2 | 千葉真一   |  34 |     1939 | 002     |
|  3 | 北大路欣也 |  30 |     1943 | 003     |
|  4 | 梶芽衣子   |  26 |     1947 | 002     |
+----+------------+-----+----------+---------+
```

さて、この emp 表に 以下のように dept_id の項目に dept 表にない値を指定してデータを入力してみる。

```
MariaDB [sample]> INSERT INTO emp (name, age, birthday, dept_id) VALUES
    -> (' 成田三樹夫', 38, 1935, '005');
```

すると、次のようなエラーメッセージが出て、入力に失敗する。

```
ERROR 1452 (23000): Cannot add or update a child row:
a foreign key constraint fails (`sample`.`emp`, CONSTRAINT `emp_ibfk_1`
FOREIGN KEY (`dept_id`) REFERENCES `dept` (`id`))
```

dept 表にある値にして入力する。

```
MariaDB [sample]> INSERT INTO emp (name, age, birthday, dept_id) VALUES
    -> (' 成田三樹夫', 38, 1935, '004');
```

するとうまく入力できることがわかる。

※ もし dept 表の id が、たとえば 営業部が 2 から 5 に変更になったとするとどうなるか？
emp 表の dept_id も修正しなくてはならなくなる。
こんなときのために、外部キー制約のところに以下のような記述をすることができる。

```
CREATE TABLE IF NOT EXISTS emp (
  id INT AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  age INT NOT NULL,
  birthday YEAR NOT NULL,
  dept_id CHAR(3),
  PRIMARY KEY (id),
  FOREIGN KEY(dept_id) REFERENCES dept(id)
  ON DELETE SET NULL ON UPDATE CASCADE
);
```

ON DELETE SET NULL — 参照先を delete すると、参照元が null になる。
ON UPDATE CASCADE — 参照先を update すると、参照元も update される。[*6] しかし、dept 表が頻繁に修正されるというのはあってほしくないことである。その表を参照している表に大きな影響を与えることになるからである。

6.1.2 参照している表を変更してみる

この状態で dept 表を変更してみる。
表の変更 (更新) は、以下の構文を使うことでできる。

[*6] 参考: https://qiita.com/SLEAZOIDS/items/d6fb9c2d131c3fdd1387

```
UPDATE <テーブル名> SET <変更するカラム名> = <新しい値> WHERE <条件となるカラム> = <値>
```

たとえば、" 営業部" の "002" を "005" に変更してみる。

```
MariaDB [sample]> UPDATE dept SET id = '005' WHERE id = '002';
```

```
MariaDB [sample]> SELECT * FROM dept;
```

```
+-----+--------+
| id  | name   |
+-----+--------+
| 001 | 総務部 |
| 003 | 経理部 |
| 004 | 開発部 |
| 005 | 営業部 |
+-----+--------+
```

```
MariaDB [sample]> SELECT * FROM emp;
```

```
+----+------------+-----+----------+---------+
| id | name       | age | birthday | dept_id |
+----+------------+-----+----------+---------+
|  1 | 菅原文太   |  40 |     1933 | 001     |
|  2 | 千葉真一   |  34 |     1939 | 005     |
|  3 | 北大路欣也 |  30 |     1943 | 003     |
|  4 | 梶芽衣子   |  26 |     1947 | 005     |
+----+------------+-----+----------+---------+
```

dept 表の id が変更されたら、emp 表の dept_id も更新されているのがわかる。
これは、emp 表を定義したときの `ON UPDATE CASCADE` の働きによる。

6.1.3 参照している表のデータを削除してみる

今度は、参照している表のデータを削除してみる。削除は、以下の構文を使う。

```
DELETE FROM <テーブル名> WHERE <削除カラム名> = <値>;
```

dept 表の id:'003' name:' 経理部' を削除してみる。

```
MariaDB [sample]> DELETE FROM dept WHERE id = '003';
```

```
MariaDB [sample]> SELECT * FROM dept;
```

```
+-----+--------+
| id  | name   |
+-----+--------+
| 001 | 総務部 |
| 004 | 開発部 |
| 005 | 営業部 |
+-----+--------+
```

```
MariaDB [sample]> SELECT * FROM emp;
```

```
+----+------------+-----+----------+---------+
| id | name       | age | birthday | dept_id |
+----+------------+-----+----------+---------+
|  1 | 菅原文太   |  40 |     1933 | 001     |
|  2 | 千葉真一   |  34 |     1939 | 005     |
|  3 | 北大路欣也 |  30 |     1943 | NULL    |
|  4 | 梶芽衣子   |  26 |     1947 | 005     |
+----+------------+-----+----------+---------+
```

このように、dept 表で削除されたデータを参照していた emp 表の項目は "NULL" になっていることが確認できる。

これは emp 表の定義の中の `ON DELETE SET NULL` の働きによる。

6.1.4 もっと厳しく制限をかける

今までの制限は、緩い制限で、本来変更してはいけないデータの変更を許すものであった。

そこで、もっと厳しい制限をかけたほうがいい場合もある。

```
CREATE TABLE IF NOT EXISTS emp (
  id INT AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  age INT NOT NULL,
  birthday YEAR NOT NULL,
  dept_id CHAR(3),
  PRIMARY KEY (id),
  FOREIGN KEY (dept_id) REFERENCES dept(id)
  ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

NO DELETE RESTRICT — 参照している表 (dept) のデータを削除するときエラーにする。
NO UPDATE RESTRICT — 参照している表 (dept) のデータを変更するときエラーにする。

ファイル "reinit_data.sql" の emp 表の定義部分を上記のように書き変えたのち、"source reinit_data.sql" でファイル reinit_data.sql を読み込む。

その後、以下のように dept 表のデータを変更してみる。

```
MariaDB [sample]> UPDATE dept SET id = '005' WHERE id = '003';
```

このようにエラーが出て、dept 表のデータは変更できない。

```
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`sample`.`emp`, CONSTRAINT `emp_ibfk_1` FOREIGN KEY (`dept_id`) REFERENCES `dept` (`id`))
```

今度は dept 表のデータを削除してみる。

```
MariaDB [sample]> DELETE FROM dept WHERE id = '002';
```

このように dept 表のデータは削除もできなくなっている。

ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (‘sample‘.‘emp‘, CONSTRAINT ‘emp_ibfk_1‘ FOREIGN KEY (‘dept_id‘) REFERENCES ‘dept‘ (‘id‘))

このように、参照している表は、削除したり変更したりできないほうが保守しやすい。しかし、その時々で適切に対応するしかない。

なお、'ON DELETE ……ON UPDATE ……を指定しなかった場合は、この

ON DELETE RESTRICT ON UPDATE RESTRICT

が自動的に指定される。

6.1.5 制約名

ところで、エラーメッセージ中に CONSTRAINT ‘emp_ibfk_1‘ という部分があるが、'emp_ibfk_1' は、MySQL が勝手につけたこの制約の名前である。

制約には名前をつけることができる。名前をつけておくと、エラーが出たときに、どの部分の制約か判別しやすい。

今回の emp 表定義中の制約に名前をつけてみる。ファイル "reinit_data.sql" の emp 表定義の部分を以下のように修正する。

```
CREATE TABLE IF NOT EXISTS emp (
  id INT AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  age INT NOT NULL,
  birthday YEAR NOT NULL,
  dept_id CHAR(3),
  PRIMARY KEY (id),
  CONSTRAINT fk_dept_id
  FOREIGN KEY(dept_id) REFERENCES dept(id)
  ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

※ CONSTRAINT は " 制約" という意味。

修正したら、`> source reinit_data.sql` とする。

そののち、dept 表のひとつのデータを削除してみる。

```
MariaDB [samle]> DELETE FROM dept WHERE id = '002';
```

以下のように 制約名 "fk_dept_id" が出力されている。

ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (‘sam-

ple'.'emp', CONSTRAINT 'fk_dept_id' FOREIGN KEY ('dept_id') REFERENCES 'dept' ('id'))

7　MySQL その他

以下は、MySQL がどんな設定で動作しているかということで、特に指定しなくても困らない。

7.1　文字コードあるいは文字セット

単に文字コードといった場合、文字セットを指すことが多い。

主な文字集合と符号化方式

大分類	文字集合	符号化方式	説明
半角系	ASCII		米国規約。半角英数記号文字を定義したもの。7ビット。
	ISO/IEC 646		国際規格。ASCIIを各国語に拡張したもの。7ビット。
	JIS X 0201		ISO/IEC 646 の日本カスタマイズ版。英数字・記号・半角カナを定義。旧称 JIS C 6220。
	ISO-8859		欧州系の文字を定めたもの。8ビット。
制御文字	ISO/IEC 6429		制御文字を定義。
	JIS X 0211		ISO/IEC 6429 の日本版。旧称 JIS C 6223。
JIS系	JIS X 0208		平仮名、片仮名、漢字などの日本語を定義。
		ISO-2022-JP	主に電子メールで利用される。俗にいう JISコード。
		EUC-JP	主に Linux 系システムで使用される。
		Shift_JIS	主に Windows 系システムで使用される。
	JIS X 0212		通称「JIS補助漢字」。あまり使用されていない。
	JIS X 0213		通称「JIS2000」「JIS2004」。第三水準・第四水準漢字を定義。
Unicode系	Unicode	UTF-8	Unicode で一番よく利用される形式。ASCIIは1バイト、日本語は3バイトで表現。
		UTF-16	Unicode を 16ビットで表現。
		UTF-32	Unicode を 32ビットで表現。

(出典) 文字コード入門 https://www.tohoho-web.com/ex/charset.html

7.1.1　MySQL の文字コード (文字セット)

MySQL にログインする。

```
> mysql -u sampleuser -p
Enter password: ******** MariaDB [(none)]>
```

ここで以下のコマンドを実行する。

```
MariaDB [(none)]> SHOW VARIABLES LIKE '%char%';
```

※ これは "'char' という文字列を含む変数を表示しなさい" という意味のコマンドである。

```
+--------------------------+---------------------------------+
| Variable_name            | Value                           |
+--------------------------+---------------------------------+
| character_set_client     | cp932                           |
| character_set_connection | cp932                           |
| character_set_database   | utf8mb4                         |
| character_set_filesystem | binary                          |
| character_set_results    | cp932                           |
| character_set_server     | utf8mb4                         |
| character_set_system     | utf8                            |
| character_sets_dir       | C:\xampp\mysql\share\charsets\ |
+--------------------------+---------------------------------+
```

MySQL は、サーバープログラムとクライアントプログラムで動作している。

XAMPP コントロールパネルで “Start” ボタンをクリックしがのは、サーバープログラムを起動しているのである。

コマンドプロンプトで “mysql -u sampleuser -p” としているのは、クライアントプログラムを使って、サーバープログラムに接続し、ログイン処理をおこなっているのである。

普通はサーバーはネットワーク上のどこか離れた場所にあるのだけれど、XAMPP では、各自のパソコン内でサーバープログラムとクライアントプログラムが動いていることになる。

さて、上記の結果の意味は以下である。

character_set_client : cp932
　クライアントの文字セットは cp932(SJIS) である。
character_set_connection : cp932
　クライアントから受け取った文字を cp932(SJIS) に変換する。
character_set_database : utf8mb4
　データベースで使用する文字セットは utf8mb4 である。
character_set_filesystem : binary
character_set_results : cp932
　クライアントへ結果を送信するときの文字セットは cp932(SJIS) である。
character_set_server : utf8mb4
　データベース作成時の既定の文字セット。つまりサーバーの文字セット。
character_set_system : utf8
　サーバーではファイル名をこの文字コードで使う。
character_sets_dir : C:¥xampp¥mysql¥share¥charsets¥
　文字セットを扱う上で必須となるファイルを配置しているディレクトリ

XAMPP の場合、初期状態 (この状態) で Windows に最適な設定になっているはずである。

クライアント — cp932(Shift_JIS)
サーバー — UTF-8

7.1.2 データベース作成時の文字セット

MySQL はデータベースを作成するとき、utf8mb4 という文字セットを使っている。

以下のコマンドを実行することで確認できる。MySQL にログインした状態で実行する。

```
MariaDB [(none)]> SHOW CREATE DATABASE sample;
```

```
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| sample2  | CREATE DATABASE `sample` /*!40100 DEFAULT CHARACTER SET utf8mb4 */ |
+----------+-----------------------------------------------------------------+
```

utf8 には utf8mb3(3 バイト) と utf8mb4(4 バイト) がある。

たとえば “吉” の下が長い文字は utf8mb3 には含まれない。

MySQL で 単に “utf8” と指定した場合は “utf8mb3” が指定されたことになる。これは歴史的な経緯でそうなったらしい。ただ、これは近いうちに “utf8mb4” になるらしい。(MySQL8.0 ではそうなっているという)。

今回は何も指定せずにデータベースを作成したが、“utf8mb4” が暗黙のうちに指定されている。

7.1.3 テーブル作成時の文字セット

テーブル作成時の文字セットは、MySQL にログインし、sample データベースの使用を宣言してのち、以下のコマンドを実行することで確認できる。

```
MariaDB [(none)]> SHOW CREATE TABLE emp;
```

```
CREATE TABLE `emp` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  `age` int(11) NOT NULL,
  `birthday` year(4) NOT NULL,
  `dept_id` char(3) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_dept_id` (`dept_id`),
  CONSTRAINT `fk_dept_id` FOREIGN KEY (`dept_id`) REFERENCES `dept` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
```

DEFAULT CHARSET=utf8mb4 とあるので、“utf8mb4” が使われているのがわかる。

7.2 照合順序 (collation)

照合順序 (collation) というのは、文字データを並び変える場合、どういう順序で並び変えるかということである。たとえば、“は” と” ば” と” ぱ” の場合とかである。

MySQL ではテーブルを作成したときに、何も指定しなければ、“utf8mb4_general_ci” という照合順序が指定される。

これは、> SHOW TABLE STATUS FROM sample; というコマンドで確認できる。

“utf8mb4_general_ci” というのは、大文字と小文字を区別しないという指定である。

“Tom” と “tom” が区別されるとややこしいから、通常はこの区別しないという指定でよい。

7.3 ストレージエンジン

MySQL では、ストレージエンジンとして “InnoDB” というのが使われている。

MySQL5.5 から “InnoDB” がデフォルトのストレージエンジンとなった。それ以前は、MyISAM というのが使われていた。

“InnoDB” の大きな特徴としては “外部キー制約” が使えるようになったことがあげられる。