

# MySQL を使う (XAMPP 版)

## v1.8

Seiichi Nukayama

2023 年 9 月 22 日

### 目次

1	データベースを設計する	1
1.1	扱うデータ . . . . .	1
2	データベースを作成する	2
2.1	root ユーザーでログインする . . . . .	2
2.2	データベースを作成する . . . . .	2
2.3	sample 専用ユーザーを作成する . . . . .	2
3	テーブル (表) を作成する	4
3.1	作成する表のイメージ . . . . .	4
3.2	テーブルの定義 . . . . .	5
3.3	テーブルの作成 . . . . .	5
3.4	データの登録 . . . . .	7
3.5	ファイル読み込みによるデータの登録 . . . . .	8
3.6	テーブル作成からデータの登録までを自動化する . . . . .	11
4	アプリケーションを作るときの初期データの投入	14
5	データベースをバックアップする	15
5.1	バックアップ . . . . .	15
5.2	データのリストア (復元) . . . . .	15
6	2 つのテーブルを結合する	16
6.1	内部結合 (JOIN 句) . . . . .	16
6.2	表示項目を絞る . . . . .	16
6.3	テーブルの指定を簡略化する . . . . .	18
6.4	外部結合 (LEFT OUTER JOIN / RIGHT OUTER JOIN) . . . . .	20

7	制約	23
7.1	外部キー制約 . . . . .	23
8	MySQL その他	31
8.1	文字コードあるいは文字セット . . . . .	31
8.2	照合順序 (collation) . . . . .	33
8.3	ストレージエンジン . . . . .	33
9	さまざまなコマンド	34
9.1	表定義関連 . . . . .	34
9.2	表のデータの更新 . . . . .	35
9.3	表のデータの削除 . . . . .	35
9.4	表のデータの表示・検索 . . . . .	36
9.5	MySQL の管理 . . . . .	36
9.6	ユーザー一覧 . . . . .	37
9.7	パスワードの再設定 . . . . .	38

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

また、その会社には以下のような部署があるとする。

総務部
経理部
営業部
開発部
人事部
情報システム部

## 2 データベースを作成する

### 2.1 root ユーザーでログインする

作業をするために、root ユーザーでログインする。

```
> mysql -u root -p
> Enter password:
```

### 2.2 データベースを作成する

データベース名を sample とする。

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

### 2.3 sample 専用ユーザーを作成する

アプリケーションがこのデータベースにアクセスするためには、アプリケーションがこのデータベースにログインできなければならない。

ふつう、アプリケーションには root ではログインさせない。そのアプリケーション専用のユーザーアカウントを作成し、そのアカウントでログインさせるようにする。

ユーザー名	sampleuser
パスワード	sampleuser

ユーザーの作成は以下のコマンドでできる。

```
MariaDB [(none)]> CREATE USER 'sampleuser'@'localhost'
-> IDENTIFIED BY 'sampleuser';
```

そのユーザーに、sample データベースへの権限を与える。

```
MariaDB [(none)]> GRANT ALL ON sample.* TO 'sampleuser'@'localhost';
```

“sample.\*” というのは、“sample というデータベースの全てのテーブル” に対してという意味である。

ユーザー名は文字列であることを明示するために 'sampleuser' とする。

次に “@” を記述し、そのあとに ホスト名を 文字列で指定する。そのユーザーの属しているホストの名前である。'%' と指定すると、全てのホストから接続可能となるが、通常はそのような指定はしない。'localhost' というのは、このデータベースに接続しようとしているアプリの所属するホストである。つまり、このデータベースとアプリとは同じホストに属しているということになる。

ユーザーの作成と権限の設定は、以下のコマンドで一度にできる。

```
MariaDB [(none)]> GRANT ALL ON sample.* TO 'sampleuser'@'localhost'  
-> IDENTIFIED BY 'sampleuser';
```

### 2.3.1 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

このまま root ユーザーとして作業をしてもいいのであるが、ここでは、今作成したユーザーでログインして作業することにする。

exit あるいは quit でログアウトする。

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

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)
```

### 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
005	人事部
006	情報システム部

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          入力が必要。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,  
-> name VARCHAR(20) NOT NULL,  
-> age INT NOT NULL,  
-> birthday YEAR NOT NULL,  
-> dept_id CHAR(3),  
-> PRIMARY KEY (id)  
-> );
```

同様に dept テーブルも作成する。<sup>\*1</sup>

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

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時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,
```

---

<sup>\*1</sup> 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');

```

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。

TeraPad などのエディタで記述しておいて、コピー&貼り付け 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
-> (' 千葉真一', 34, 1939, '002'),
-> (' 北大路欣也', 30, 1943, '003'),
-> (' 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 ファイル読み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
4   (id, name)
5 VALUES
6   ('001', '総務部'),
7   ('002', '営業部'),
8   ('003', '経理部'),
9   ('004', '開発部'),
10  ('005', '人事部'),
11  ('006', '情報システム部');
```

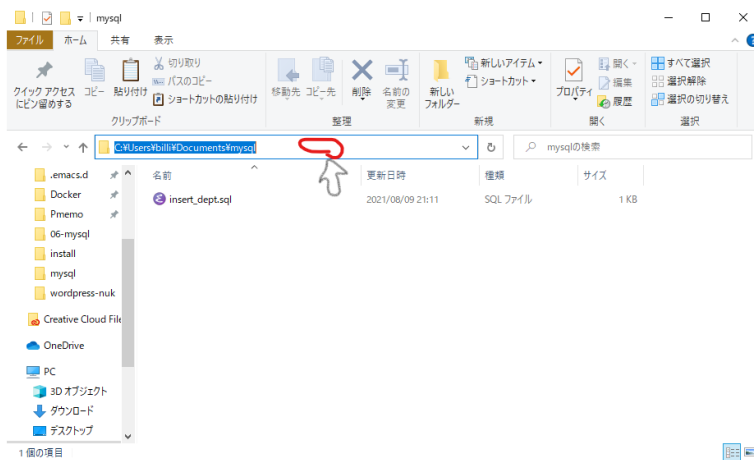
--(半角空白)で始まる行は、コメントである。<sup>\*2</sup>

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。

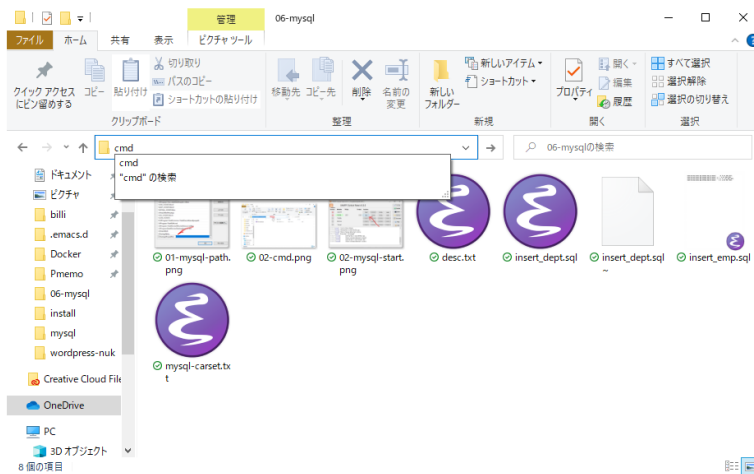
#### 3.5.2 そのフォルダでコマンドプロンプトを起動する。

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。

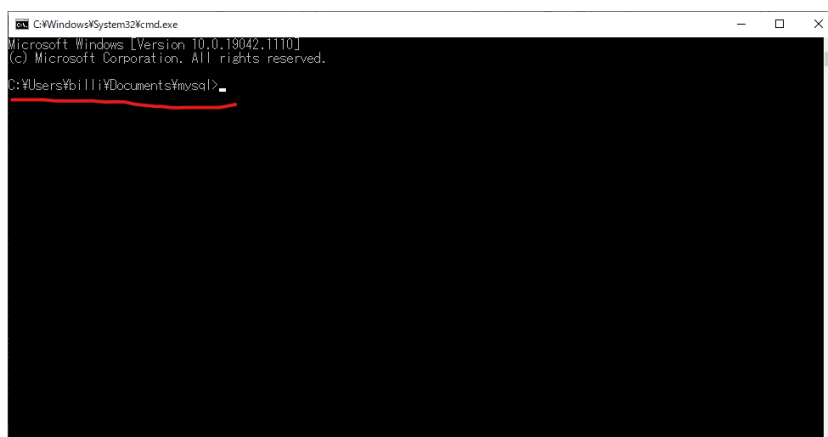
<sup>\*2</sup> ほかに、# で始まる行もコメント。また複数行は、/\* ... \*/ が使える。



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: *****
```

root ユーザーでログインしてもよい。  
sample データベースの使用を宣言する。

```
MariaDB [(none)]> USE 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        |
| 005 | 人事部        |
| 006 | 情報システム部 |
+-----+-----+
```

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、ファイル読み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 CREATE DATABASE sample;
2
3 -- sample データベースを使用
4 USE sample;
5
6 -- emp テーブルの作成
7
8 CREATE TABLE emp (
9     id          INT          AUTO_INCREMENT,
10    name        VARCHAR(20) NOT NULL,
11    age         INT          NOT NULL,
12    birthday    YEAR         NOT NULL,
13    dept_id     CHAR(3),
14    PRIMARY KEY (id)
15 );
16
17 -- dept テーブルの作成
18
19 CREATE TABLE dept (
20     id  CHAR(3)  PRIMARY KEY,
21     name VARCHAR(20) NOT NULL
22 );
23
24 INSERT INTO emp
25     (name, age, birthday, dept_id)
26 VALUES
27     ('菅原文太', 40, 1933, '001'),
28     ('千葉真一', 34, 1939, '002'),
29     ('北大路欣也', 30, 1943, '003'),
30     ('梶芽衣子', 26, 1947, '002');
31
32 INSERT INTO dept
33     (id, name)
34 VALUES
35     ('001', '総務部'),
36     ('002', '営業部'),
37     ('003', '経理部'),
38     ('004', '開発部'),
39     ('005', '人事部'),
40     ('006', '情報システム部');
41
42
43 SELECT * FROM emp;
44 SELECT * FROM dept;

```

これを実行するには、まず最初に sample データベースを削除しておかねばならない。なぜなら、最初に

```
CREATE DATABASE sample;
```

としているからである。

削除しなくてもエラーを無視することもできる。

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

また、途中で CREATE TABLE emp としているが、すでに emp テーブルが存在していたら、エラーが発生

する。

ただ、そのとき、存在している emp テーブルが、これから作ろうとしているテーブルの定義に合致したものなのかは不明である。たまたま同名の emp テーブルが存在しているのかもしれない。

だから、テーブルを作成するときに、すでに同名のテーブルが存在するときは、そのテーブルを削除してからテーブルを作成したほうがよい。

それらを踏まえて記述したのが、以下である。

### リスト 3 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
4  -- sample データベースを使用
5  USE sample;
6
7  -- emp テーブルの作成
8  -- もし emp テーブルが存在していたら削除する。
9  -- 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。
10
11 DROP TABLE IF EXISTS emp;
12
13 CREATE TABLE emp (
14     id      INT      AUTO_INCREMENT,
15     name    VARCHAR(20) NOT NULL,
16     age     INT      NOT NULL,
17     birthday YEAR    NOT NULL,
18     dept_id CHAR(3),
19     PRIMARY KEY (id)
20 );
21
22 -- dept テーブルの作成
23 -- もし dept テーブルが存在していたら削除する。
24 DROP TABLE IF EXISTS dept;
25
26 CREATE TABLE dept (
27     id  CHAR(3)    PRIMARY KEY,
28     name VARCHAR(20) NOT NULL
29 );
30
31 INSERT INTO emp
32     (name, age, birthday, dept_id)
33 VALUES
34     ('菅原文太', 40, 1933, '001'),
35     ('千葉真一', 34, 1939, '002'),
36     ('北大路欣也', 30, 1943, '003'),
37     ('梶芽衣子', 26, 1947, '002');
38
39 INSERT INTO dept
40     (id, name)
41 VALUES
42     ('001', '総務部'),
43     ('002', '営業部'),
44     ('003', '経理部'),
45     ('004', '開発部');
```

```
46      ('005', '人事部'),
47      ('006', '情報システム部');
48
49
50 SELECT * FROM emp;
51 SELECT * FROM dept;
```

## 4 アプリケーションを作るときの初期データの投入

データベースを使ったアプリケーションを作るときは、まず最初にデータベースに何件かのデータを入れておく。なぜなら、これからアプリケーションを作成するから。データベースにデータを挿入したり、更新したり、削除したりする機能をこれから作成する。で、最初に作成する機能が、データー一覧の機能であることが多い。

データベースの CRUD 機能をアプリケーションに組み込んでいる際に、データベースのデータを初期状態に戻したいことがよくある。

また、別の PC を使ってプログラミングするときも、こういうスクリプトを作成しておけば、簡単に初期データを作成することができる。



## 5 データベースをバックアップする

### 5.1 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 というコマンドを使っておこなう。

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 root -p -B sample > sample.dump
```

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

ここでは保存ファイル名を sample.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.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。<sup>\*3</sup>

### 5.2 データのリストア (復元)

以下のコマンドを実行する。<sup>\*4</sup>

```
> mysql -u sampleuser -p < sample.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ます。

<sup>\*3</sup> また、-B 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。

<sup>\*4</sup> -B 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.dump
```

とするか、あるいは、sampleuser でログインしてから

```
MariaDB[sample]> source sample.dump
```

 とすればよい。

## 6 2つのテーブルを結合する

### 6.1 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。

### 6.2 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.name” 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。

### 6.3 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;
```

## 6.4 外部結合 (LEFT OUTER JOIN / RIGHT OUTER JOIN)

### 6.4.1 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
誕生日   : 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？

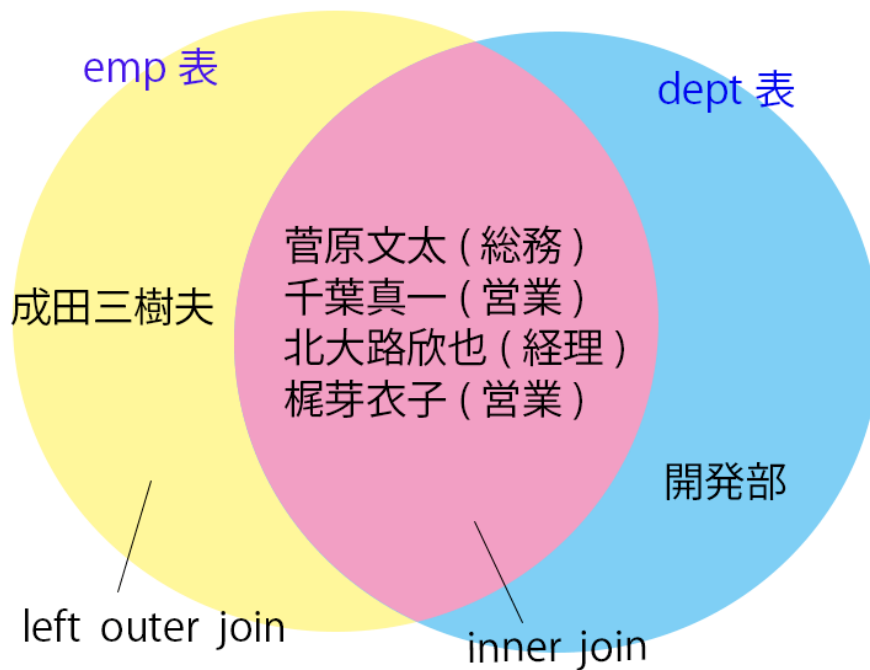
```
MariaDB [sample]> SELECT
-> e.id AS ID,
-> e.name AS 名前,
-> e.age AS 年齢,
-> d.name AS 部署名
-> FROM emp e
-> JOIN dept d
-> ON e.dept_id = d.id
-> order by ID;
```

```
+----+-----+-----+-----+
| 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

-----+

成田三樹夫は、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

#### 6.4.2 右外部結合 RIGHT [OUTER] JOIN

また、dept 表をみると、id: '004' が開発部、id: '005' が人事部、id: '006' が情報システム部であるが、emp 表には dept\_id が '004'、'005'、'006' である人はいない。

この状態で結合表をつくり、開発部などの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
NULL	NULL	NULL	人事部
NULL	NULL	NULL	情報システム部
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



## 7 制約

### 7.1 外部キー制約

#### 7.1.1 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
005	人事部
006	情報システム部

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、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。

ここでは、emp 表と dept 表を作成したあとに、emp 表の定義に外部キー制約をつけ加える形でおこなうことにする。

dept 表の定義 (再掲)

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

emp 表の定義 (再掲)

```
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),  
    -> PRIMARY KEY (id)  
    -> );
```

emp 表に外部キー制約を追加

```
MariaDB [sample]> ALTER TABLE emp  
    -> ADD  
    -> FOREIGN KEY (dept_id) REFERENCES dept (id) ;
```

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 4 reinit\_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  
4  -- sample データベースを使用  
5  USE sample;  
6  
7  -- もし empテーブルが存在していたら削除する。。  
8  DROP TABLE IF EXISTS emp;  
9  
10 -- もし deptテーブルが存在したら削除する。  
11 -- empテーブルが存在していたら削除できないので、  
12 -- (empテーブルがdeptテーブルを参照しているため)  
13 -- empテーブルを先に削除しなくてはならない。  
14 DROP TABLE IF EXISTS dept;  
15  
16 -- dept テーブルの作成  
17 CREATE TABLE IF NOT EXISTS dept (  
18     id    CHAR(3),  
19     name  VARCHAR(20) NOT NULL,  
20     PRIMARY KEY (id)  
21 );  
22  
23 -- emp テーブルの作成  
24 CREATE TABLE IF NOT EXISTS emp (  
25     id        INT            AUTO_INCREMENT,  
26     name      VARCHAR(20) NOT NULL,  
27     age       INT            NOT NULL,  
28     birthday  YEAR           NOT NULL,  
29     dept_id   CHAR(3),  
30     PRIMARY KEY (id)  
31 );
```

```

32
33 -- emp表に外部キー制約を追加
34 ALTER TABLE emp
35     ADD
36     FOREIGN KEY(dept_id) REFERENCES dept(id);
37
38 -- dept表の初期データ
39 INSERT INTO dept
40     (id, name)
41 VALUES
42     ('001', '総務部'),
43     ('002', '営業部'),
44     ('003', '経理部'),
45     ('004', '開発部');
46
47 -- emp表の初期データ
48 INSERT INTO emp
49     (name, age, birthday, dept_id)
50 VALUES
51     ('菅原文太', 40, 1933, '001'),
52     ('千葉真一', 34, 1939, '002'),
53     ('北大路欣也', 30, 1943, '003'),
54     ('梶芽衣子', 26, 1947, '002');
55
56 SELECT * FROM dept;
57 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。(root ユーザーでもよい)

```

> mysql -u sampleuser -p
Enter password: ****

```

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 |
| 005 | 人事部 |
| 006 | 情報システム部 |
+-----+-----+

```

```

+-----+-----+

```

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, '007');
```

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 7 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。<sup>\*5</sup> 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。

### 7.1.2 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きる。

```
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	人事部
006	情報システム部
007	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	007
3	北大路欣也	30	1943	003
4	梶芽衣子	26	1947	007

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。

### 7.1.3 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。

<sup>\*5</sup> 参考: <https://qiita.com/SLEAZOIDS/items/d6fb9c2d131c3fdd1387>

```
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	人事部
006	情報システム部
007	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。

#### 7.1.4 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。

#### 7.1.5 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 ('sample'. 'emp', CONSTRAINT 'fk_dept_id' FOREIGN KEY ('dept_id') REFERENCES 'dept' ('id'))
```



## 8 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。

### 8.1 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	<a href="#">ASCII</a>		米国規約。半角英数記号文字を定義したもの。7ビット。
	<a href="#">ISO/IEC 646</a>		国際規格。ASCIIを各国語に拡張したもの。7ビット。
	<a href="#">JIS X 0201</a>		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。
	<a href="#">ISO-8859</a>		欧州系の文字を定めたもの。8ビット。
制御文字	<a href="#">ISO/IEC 6429</a>		制御文字を定義。
	<a href="#">JIS X 0211</a>		ISO/IEC 6429 の日本版。旧称 JIS C 6223。
JIS系	<a href="#">JIS X 0208</a>		平仮名、片仮名、漢字などの日本語を定義。
	<a href="#">ISO-2022-JP</a>		主に電子メールで利用される。俗にいう JISコード。
	<a href="#">EUC-JP</a>		主に Linux 系システムで使用される。
	<a href="#">Shift_JIS</a>		主に Windows 系システムで使用される。
	<a href="#">JIS X 0212</a>		通称「JIS補助漢字」。あまり使用されていない。
	<a href="#">JIS X 0213</a>		通称「JIS2000」「JIS2004」。第三水準・第四水準漢字を定義。
Unicode系	<a href="#">Unicode</a>	<a href="#">UTF-8</a>	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。
		<a href="#">UTF-16</a>	Unicode を 16ビットで表現。
		<a href="#">UTF-32</a>	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>

#### 8.1.1 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、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

### 8.1.2 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。

### 8.1.3 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。

## 8.2 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。

## 8.3 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。

## 9 さまざまなコマンド

### 9.1 表定義関連

#### 9.1.1 表定義の変更

以下のような表定義があったとする。

```
CREATE TABLE student (  
    id char(3),  
    name varchar(20)  
)
```

この name カラムに NOT NULL 制約を追加したいとする。以下のようにできる。

```
MariaDB> ALTER TABLE student  
    -> MODIFY name varchar(20) NOT NULL;
```

```
ALTER TABLE テーブル名  
MODIFY カラム名 カラム定義
```

#### 9.1.2 カラム名の変更

name カラムの名称を st.name に変更したいときは、以下のようにする。

```
MariaDB> ALTER TABLE student  
    -> RENAME COLUMN name TO st_name;
```

```
ALTER TABLE テーブル名  
RENAME COLUMN カラム名 TO 新しいカラム名
```

#### 9.1.3 表定義にカラムを追加

また、age (年齢) という項目を追加したいとする。以下のようにできる。

```
MariaDB> ALTER TABLE student  
    -> ADD age int NOT NULL;
```

```
ALTER TABLE テーブル名  
ADD カラム名 カラム定義
```

#### 9.1.4 表定義にプライマリーキーを追加

あるいは、id に PRIMARY KEY を設定したいとする。以下のようにできる。

```
MariaDB> ALTER TABLE student
-> ADD PRIMARY KEY (id);
```

```
ALTER TABLE テーブル名
ADD PRIMARY KEY (カラム名)
```

### 9.1.5 表定義に外部キー制約を追加

外部キー制約をあとから設定するには、以下のようにする。

```
ALTER TABLE emp          -- emp テーブル再定義
ADD                        -- 追加
CONSTRAINT fk_dept        -- 制約名 fk_dept
FOREIGN KEY (dept_id)     -- 外部キー dept_id
REFERENCES dept(id);      -- 参照 dept 表の (id)
```

上の例は、emp 表の中にある dept\_id という項目が、dept 表の id という項目を参照しているという設定である。

## 9.2 表のデータの更新

たとえば、emp 表の id=1 の人の氏名を変更するには、以下のようにする。

```
MariaDB> UPDATE emp SET name = '菅原文夫' WHERE id = 1;
```

```
UPDATE テーブル名 SET カラム名 = 訂正データ WHERE 条件
```

## 9.3 表のデータの削除

### 9.3.1 1件のデータを削除する

id が 1 のデータを削除する。

```
MariaDB> DELETE FROM emp WHERE id = 1;
```

```
DELETE FROM テーブル名 WHERE 条件
```

id が 1 のデータを削除すると、1 が欠番になる。

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

この場合、id の数字を振り直すことはしない。なぜなら、id は順番を表す数字ではなく、そのデータを特定

するための数字だからである。それがプライマリー・キーの役割である。順番になっているのは、データを作成するときに、AUTO\_INCREMENT を使ったからである。

### 9.3.2 全データを削除する

条件をつけなければ、全データが削除される。

```
MariaDB> DELETE FROM emp;
```

```
DELETE FROM テーブル名
```

## 9.4 表のデータの表示・検索

SELECT で表のデータを表示するとき、WHERE で条件を指定する際に、“LIKE” で条件を指定できる。

```
MariaDB> SELECT * FROM emp WHERE name LIKE '%葉%';
```

```
+---+-----+---+-----+-----+
| id | name      | age | birthday | dept_id |
+---+-----+---+-----+-----+
|  2 | 千葉真一  |  34 |    1939  |    002   |
+---+-----+---+-----+-----+
```

```
| SELECT [カラム名, ...] FROM テーブル名 WHERE カラム名 LIKE '% 語句 %'; |
```

% は 0 文字以上の文字列に合致する。

\_ は 1 文字以上の文字列に合致する。

## 9.5 MySQL の管理

### 9.5.1 MySQL の文字コードの確認

MySQL で文字コードを確認するには、以下のコマンドを使う。

```
MariaDB> SHOW VARIABLES LIKE '%char%';
```

Windows のコマンドプロンプトで実行すると、以下のようになる。

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	/usr/share/mysql/charsets/

Mac のターミナルや、同じ Windows でも、コマンドプロンプトが UTF-8(cp65001) になっていると、以下のようになる。

Variable_name	Value
character_set_client	utf8mb4
character_set_connection	utf8mb4
character_set_database	utf8mb4
character_set_filesystem	binary
character_set_results	utf8mb4
character_set_server	utf8mb4
character_set_system	utf8
character_sets_dir	/usr/share/mysql/charsets/

*mysql* というのはクライアント・アプリで、*mysqld* というのがサーバー・アプリなのだが、*mysql* クライアントが *mysqld* サーバーに接続するときに、自身の環境の文字コードを *mysqld* サーバーに送るのである。*mysqld* サーバーは、SQL コマンドの結果を *mysql* クライアントに送るときにそのクライアントの文字コードにして送るのである。

ただ、*mysqldump* コマンドの場合、指定しなければ UTF-8 の文字コードで送られる。だから、*mysqldump* で取得できたダンプファイルの文字コードは UTF-8 である。

これは、リストアする時に特に意識する必要はない。

## 9.6 ユーザー一覧

MySQL で今までに作成したユーザーの一覧を見たい場合、以下のコマンドでできる。ただし、ルート・ユーザー (root) でないと権限がないと怒られる。

```
MariaDB> SELECT user, host, db FROM mysql.db;
```

パスワードは暗号化されているので、見ても仕方がないが、以下のコマンドでできる。

```
MariaDB> SELECT user, password FROM mysql.user;
```

## 9.7 パスワードの再設定

パスワードを忘れたなどで、パスワードを再設定したい場合は、以下のコマンドでできる。(root ユーザーで実行)

```
MariaDB> SET PASSWORD FOR 'ユーザー名'@'localhost' = password('パスワード');
```