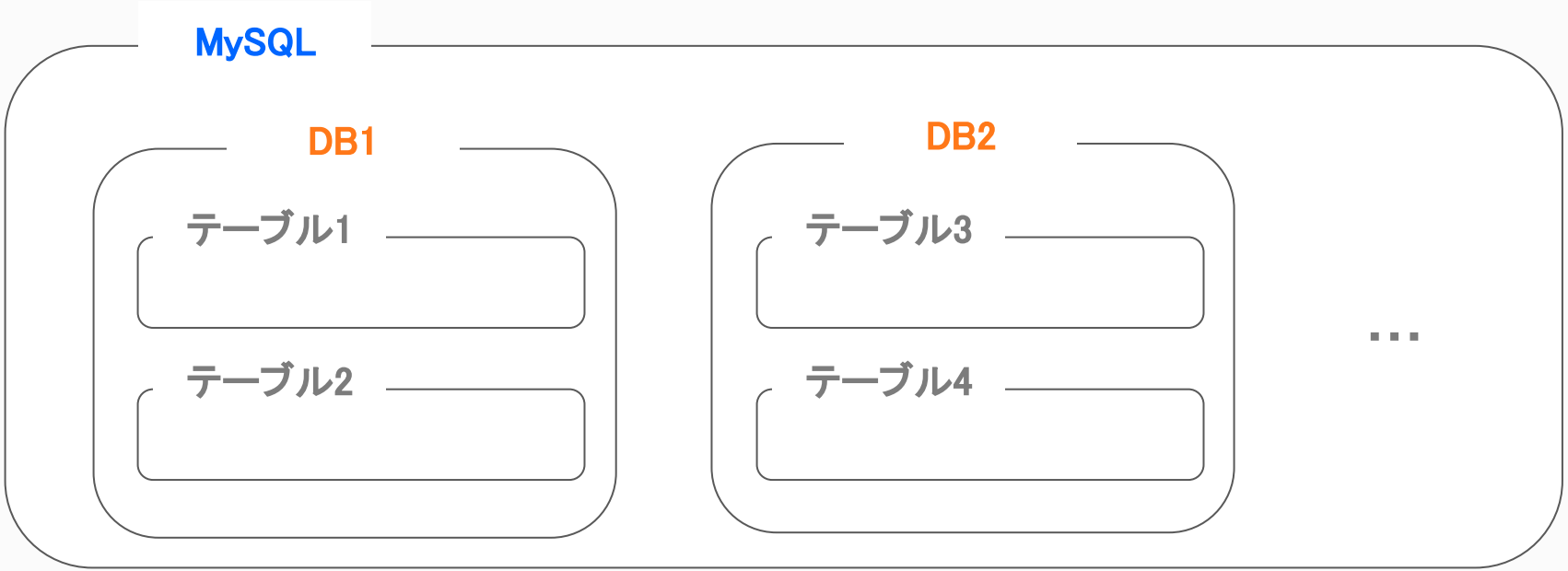
DBの作成、テーブルの定義

1日目	2日目	3日目	4日目	5日目	6日目	7日目
8日目	9日目	10日目	11日目	12日目	13日目	14日目
15日目	16日目	17日目	18日目	19日目	20日目	21日目

DBの作成

データベース(DB)とテーブル

MySQLではデータベースを複数持ち、各データベースの中にテーブルが格納されている



データベース(DB)を作成する

MySQLは複数のDBを所持している。CREATE DATABASEを実行すると、DBが作成され、SHOW DATABASESでデータベース一覧が表示される。

```
mysql> SHOW DATABASES; # データベース一覧を表示する
Database                |
------------------+
information_schema | # 他のデータベースに関する情報が格納されている
mysq                    | # MySQL全般の設定が格納されている
performance_schema | # パフォーマンスに関する情報が格納されている
sys                        | # performance_schema, information_schemaを確認する機能
```

```
mysql> CREATE DATABASE my_db; # データベースを作成する
mysql> USE my_db; # my_dbデータベースを利用する
```

データベース(DB)を削除する

DROP DATABASE: DBを削除する
USE DATABASE名: DBを利用する
SELECT DATABASE(): 利用中のDB名を表示する

```
mysql> DROP DATABASE my_db; # データベースを削除する
```

```
mysql> USE my_db; # my_dbデータベースを利用する
mysql> SELECT DATABASE(); # 利用しているDB名を表示
DATABASE()|
----------+
my_db     |
```

コメント文

SQLの意味を説明するための文をコメント文と言うが、MySQLでは以下3通りの書き方をする

① #を使った行末までコメント化
```
mysql> SELECT * FROM USER; # USERテーブルからレコードを取り出す
```

② --を使った行末までコメント化
```
mysql> SELECT * FROM USER; -- USERテーブルからレコードを取り出す
```

③ /*　*/を使って複数行コメント化
```
mysql> SELECT * FROM USER; /*
 USERテーブルからレコードを取り出します
*/
```

テーブルの作成

テーブルとは

データを入れるための器をテーブルと言う。RDBMSでは、表形式でデータを挿入するため、各列のデータの形式(型)をテーブル作成時に定義する

USERテーブル

id	name	class_no
1	山田太郎	1
2	佐藤次郎	2
3	田中花子	1

USERテーブルは、id、name、class_noを列に持つ。
そのため、

id: **数値**
name: **文字列**
class_no: **数値**

を要素に持つテーブルを作成する

CREATE文でテーブルを作成

テーブルを作成するには、CREATEを用いる。

```
CREATE TABLE テーブル名(
    column1 column_type,
    column2 column_type,
    :
);
```

```
# 数値(INT)型id, 文字列(VARCHAR)型name, 数値(INT)型class_noを持つテーブルusersを作成
CREATE TABLE users(
 id INT,
 name VARCHAR(255),
 class _ no INT
);
```

代表的なデータ型一覧

MySQLには、以下のようなデータ型が存在する

データタイプ名	データ種別
INT	整数値
DECIMAL	小数
BOOLEAN	真偽値
CHAR	固定長文字列(最大: 255バイト(MySQL8の場合))
VARCHAR	可変長文字列(最大: 16383バイト(MySQL8の場合))
DATE	日付
DATETIME	時刻
TEXT	65536文字までの文字列

主キー(PRIMARY KEY)

重複が存在せず行を一意に識別でき、空の行が存在しない列

例) 社員番号, マイナンバー番号など

USERテーブル

id	name	class_no
1	山田太郎	1
2	佐藤次郎	2
:		

idは一意(重複がない)で、空でない

```
# PRIMARY KEYを付与する
CREATE TABLE users(
  id INT PRIMARY KEY,
  name VARCHAR(255),
  class _ no INT
);
```

テーブルの削除(DROP TABLE)

テーブルを削除するには、DROP TABLEを用いる。

DROP TABLE users; # usersテーブルが削除される

テーブルの一覧表示(SHOW TABLES)

テーブルを一覧表示するには、SHOW TABLESを用いる。
(MySQLの場合)

SHOW TABLES; # テーブル一覧が表示される

```
Tables_in_my_db|
---------------
users          |
```

テーブル定義の確認

テーブル定義を確認するには、DESCRIBE テーブル名を用いる。
（MySQLの場合）

DESCRIBE users; # usersテーブルの定義を確認

```
Field      |Type           |Null|Key|Default|Extra|
--------+-----------+----+---+-------+----
id         |int            |NO  |PRI|       |     |
name       |varchar(255)|YES |       |       |     |
class_no|int            |YES |       |       |     |
```

テーブルの定義変更

テーブル定義の変更

テーブル定義を作成後に変更することがある。この場合、ALTER TABLEを用いる

テーブル名の変更(RENAME TO)

```
# テーブル名をusersからusers_tableに変更
mysql> ALTER TABLE users RENAME TO users_table;
```

カラム(列)の削除(DROP COLUMN)

```
# usersからclass_no列を削除
mysql> ALTER TABLE users DROP COLUMN class_no;
```

カラムの追加(ADD)

```
ALTER TABLE table_name
ADD
    new_column_name column_definition
    [FIRST | AFTER column_name]
```

FIRST: テーブルの一番最初の列にカラムを追加する
AFTER: テーブルの指定した列の後にカラムを追加する
*) デフォルトだと、一番最後の列にカラムは追加される

```
# nameカラムの後に、messageカラム(TEXT型)を追加する
ALTER TABLE users
ADD message TEXT AFTER name;
```

カラム定義の変更(MODIFY)

```
#nameカラムのデータタイプをCHAR(255)にする
ALTER TABLE users
MODIFY name CHAR(255);
```

カラムの名前・場所定義の変更(CHANGE COLUMN)

```
ALTER TABLE table_name
CHANGE COLUMN original_name new_name column_definition
[FIRST | AFTER column_name];

# usersテーブルのidカラムをid_newに名前変更して、messageカラムの後に移動させる
ALTER TABLE users
CHANGE COLUMN id id_new INT
AFTER message;
```

主キーの削除(DROP PRIMARY KEY)

```
#usersテーブルから、主キーを削除する
ALTER TABLE users
DROP PRIMARY KEY
```

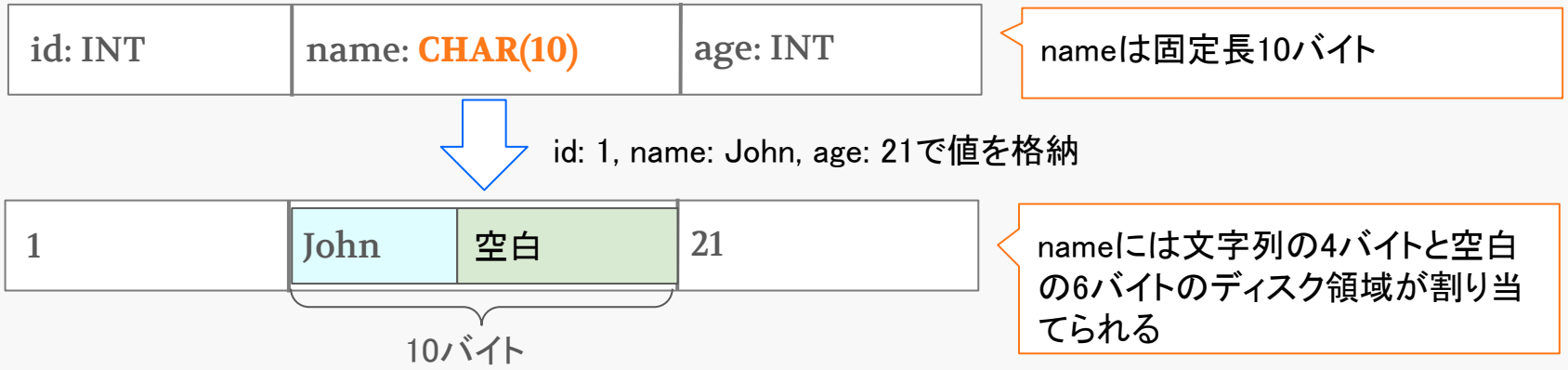
固定長文字列(CHAR)と可変長文字列(VARCHAR)の違い

固定長文字列(CHAR)と可変長文字列(VARCHAR)の違い

テーブルで文字列カラムを定義するときには、固定長の文字列(CHAR)か可変長の文字列(VARCHAR)を利用することが多い。
実際にテーブル定義する際に迷わないようにするため、この2つにどのような違いがあるのか、以下に記載した。

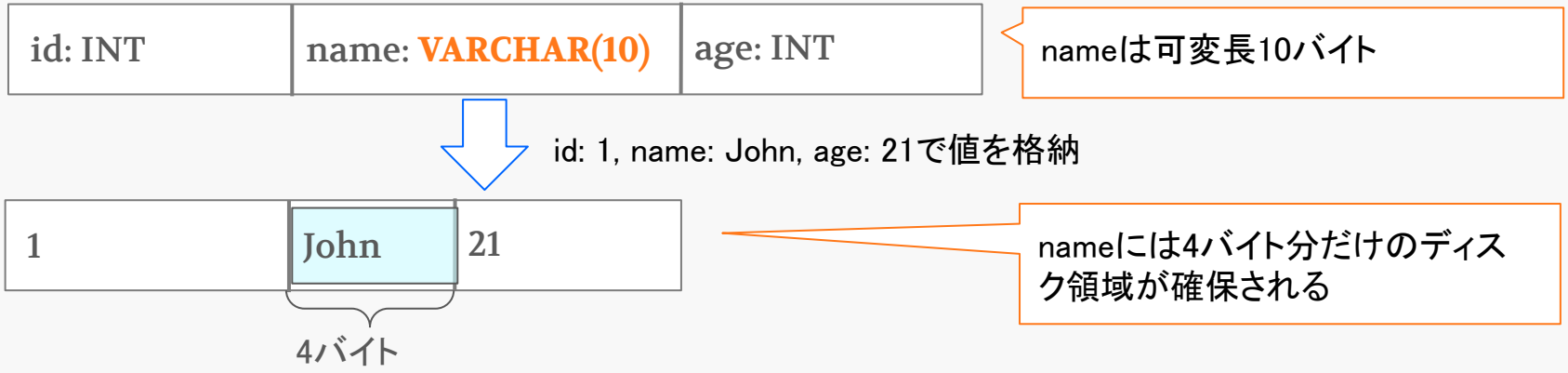
CHAR型

- DBにデータを格納する際に、あらかじめ格納するデータの領域が確保される
- 格納する値が指定した長さよりも短くても、空白の領域もディスクが確保される



VARCHAR型

- DBにデータを格納する際に、データに応じて確保されるディスク領域を調整する



VARCHARは格納時にディスク領域がデータに合わせて縮小されるため、 **ディスク領域を節約できる**

CHAR VS VARCHAR

CHAR	VARCHAR
指定した領域分ディスク領域が確保される	挿入するデータが指定したサイズよりも小さい場合は、ディスク領域が縮小される
ディスク使用量は多くなる	ディスク使用量は少なくなる
データベース操作のパフォーマンスが良い	データベース操作のパフォーマンスが悪い
末尾にスペースをつけて、データを格納した場合、そのスペースは削除される	末尾にスペースをつけて、データを格納した場合、そのスペースは削除されない
データの長さが、ある程度固定されていて、変化が少ない場合に用いられる *) 電話番号、銀行コード	データの長さが、格納するデータに応じて大きく変わる場合に用いられる 例) 住所、テキストメッセージ