

# データベースシステム

## 第6回

---

理工学部情報科学科

松澤 智史

# 本日の内容

- 集約関数
- 集合演算
- ビュー
- データ処理, 加工のためのSQL

# 前置き: SQLをファイルから実行

- 構文

mysql> source ファイル名

```
mysql> source test.sql
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> _
```

名前	更新日時	種類	サイズ
database1.sql	2019/10/20 21:27	SQL Text File	1 KB

WindowsのCommand Line Client  
ならファイルをドラッグアンドドロップで  
ファイルパスを指定することも可能

今回使うサンプルのテーブル作成文は  
createtable.sqlに記述(LETUSにアップロード済)

```
MySQL 8.0 Command Line Client - Unicode
mysql> source C:\tmp\database1.sql
```

# 集約関数

- 集約(集計)を行うための関数
- GROUP BYやHAVINGにも使用可能
- 関数
  - AVG(列名) 指定した列のNULL以外の値の平均値
  - COUNT(列名) 指定した列のNULL以外の値の数
  - COUNT(\*) 表に含まれるレコード(行)の数
  - MAX(列名) 指定した列のNULL以外の値の最大値
  - MIN(列名) 指定した列のNULL以外の値の最小値
  - SUM(列名) 指定した列のNULL以外の値の合計値

# 集約関数の使用例

```
MySQL 8.0 Command Line Client - Unicode
mysql> select * from student group by name;
+----+-----+-----+-----+
| id | name   | dep_code | email      |
+----+-----+-----+-----+
| 1  | Alice  | 63       | alice@is.jp |
| 2  | Bob    | 64       | bob@bs.jp   |
| 3  | Charlie| 63       | char@is.jp  |
| 4  | Dave   | 64       | dave@bs.jp  |
+----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select sum(dep_code) from student;
+-----+
| sum(dep_code) |
+-----+
| 254           |
+-----+
1 row in set (0.00 sec)

mysql>
```

あまり良い例ではない

# 集約関数の使用例

## 実験用のデータ作成

```
create table test(  
    id integer primary key,  
    value integer  
);
```

```
insert into test set id=1,value=80;  
insert into test set id=2,value=70;  
insert into test set id=3,value=75;  
insert into test set id=4,value=60;
```

# 集約関数の使用例 (GROUP BY)

```
MySQL 8.0 Command Line Client - Unicode
mysql> select * from test natural join student;
+-----+-----+-----+-----+-----+
| id | value | name  | dep_code | email  |
+-----+-----+-----+-----+-----+
| 1  | 80    | Alice | 63       | alice@is.jp |
| 2  | 70    | Bob   | 64       | bob@bs.jp   |
| 3  | 75    | Charlie | 63      | char@is.jp  |
| 4  | 60    | Dave  | 64       | dave@bs.jp  |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select avg(value) from test natural join student;
+-----+
| avg(value) |
+-----+
| 71.2500    |
+-----+
1 row in set (0.00 sec)

mysql> select dep_code, avg(value) from test natural join student group by dep_code;
+-----+-----+
| dep_code | avg(value) |
+-----+-----+
| 63       | 77.5000    |
| 64       | 65.0000    |
+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

# 集約関数の使用例 (HAVING)

```
MySQL 8.0 Command Line Client - Unicode

mysql> select dep_code, avg(value) from test natural join student group by dep_code;
+-----+-----+
| dep_code | avg(value) |
+-----+-----+
|      63 |    77.5000 |
|      64 |    65.0000 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> select dep_code, avg(value) from test natural join student group by dep_code having avg(value)>70;
+-----+-----+
| dep_code | avg(value) |
+-----+-----+
|      63 |    77.5000 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

HAVINGはGROUP BYの結果に検索条件を加える



# 集合演算

- UNION
  - 和演算
  - 構文: SQL文1 UNION SQL文2
  - 例 `SELECT a1, a2 FROM A UNION SELECT b1, b2 FROM B`
  - 列数を同じにする必要がある
- EXCEPT
  - 差演算
  - 構文: SQL文1 EXCEPT SQL文2
- INTERSECT
  - 積演算
  - 構文: SQL文1 INTERSECT SQL文2

# 集合演算の例

```
MySQL 8.0 Command Line Client - Unicode
mysql> select id, name from student union select dep_code, dep_name from dep;
+----+-----+
| id | name  |
+----+-----+
| 1  | Alice |
| 2  | Bob   |
| 3  | Charlie |
| 4  | Dave  |
| 61 | MA    |
| 63 | IS    |
| 64 | BS    |
+----+-----+
7 rows in set (0.00 sec)

mysql> _
```

列の数が同じであれば列名が違っても演算可能

# ビュー

- 仮想的なテーブルである
- データは持たない
- CREATE VIEWで作成する
- 参照されるたびに定義されたSELECT文を実行する

- 構文

```
CREATE VIEW ビュー名[(列のリスト)] AS  
(SELECT文 [WITH [CASCADED | LOCAL ]  
CHECK OPTION])
```

# ビューの例

右の二つのテーブルが  
存在した時

```
MySQL 8.0 Command Line Client - Unicode
mysql> show tables;
+-----+
| Tables_in_test_db |
+-----+
| dep                |
| student            |
+-----+
2 rows in set (0.02 sec)

mysql> select * from dep;
+-----+-----+
| dep_code | dep_name |
+-----+-----+
|        61 | MA      |
|        63 | IS      |
|        64 | BS      |
+-----+-----+
3 rows in set (0.02 sec)

mysql> select * from student;
+-----+-----+-----+-----+
| id | name   | dep_code | email   |
+-----+-----+-----+-----+
|  1 | Alice  |        63 | alice@is.jp |
|  2 | Bob    |        64 | bob@bs.jp   |
|  3 | Charlie |        63 | char@is.jp  |
|  4 | Dave   |        64 | dave@bs.jp  |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

# ビューの例

```
MySQL 8.0 Command Line Client - Unicode

mysql> create view VIEW_TEST as (select * from student natural join dep);
Query OK, 0 rows affected (0.02 sec)

mysql> select * from VIEW_TEST;
+-----+-----+-----+-----+-----+
| dep_code | id | name   | email      | dep_name |
+-----+-----+-----+-----+-----+
|        63 | 1 | Alice  | alice@is.jp | IS       |
|        64 | 2 | Bob    | bob@bs.jp   | BS       |
|        63 | 3 | Charlie| char@is.jp  | IS       |
|        64 | 4 | Dave   | dave@bs.jp  | BS       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

以後, VIEW\_TESTを仮想的なテーブルとして扱うことができる

※show tables;でも存在を確認することができる

# ビューの例

```
MySQL 8.0 Command Line Client - Unicode

mysql> update student set name='Alice2' where name='Alice';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from VIEW_TEST;
+-----+-----+-----+-----+-----+
| dep_code | id | name   | email   | dep_name |
+-----+-----+-----+-----+-----+
|        63 | 1 | Alice2 | alice@is.jp | IS       |
|        64 | 2 | Bob    | bob@bs.jp  | BS       |
|        63 | 3 | Charlie| char@is.jp | IS       |
|        64 | 4 | Dave   | dave@bs.jp | BS       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

studentテーブルのAliceの名前を変更するとビューの値も変更される

# ビュー 使い方等

- 良く結合する表や取り出す列の組み合わせがある場合はビューを作っておくとSQLの文が簡略化できる
- 参照するたびにビューで定義したSQL文(SELECT文)を実行するため、処理効率は(基本的には)変わらない
- ビューを作成して隠したい情報はアクセスできないようにすることでセキュリティを確保することができる
  - studentテーブルは管理者以外アクセス不可にし、電子メール以外の情報のみ列挙したビューを作成し、そちらをオープンアクセスにする など

# 体現ビュー

- ビューは毎回、SELECTを実行して仮想テーブルを作るので、頻繁に参照される場合は効率が悪い



- 実際のデータベースビューのデータを格納したビューを体現ビュー(マテリアライズド・ビュー)という
- 体現ビューは参照が高速になる反面、データが更新された場合などには不具合が生じる可能性がある
- MySQLには実装されていない



# データ処理, 加工のためのSQL

- CASE関数

構文

```
CASE WHEN 条件式  
      THEN 条件が真の場合の値  
      [ELSE 条件が真でなかった場合の値]  
END
```

- 日付関数

- CURRENT\_DATE
- CURRENT\_TIMESTAMP

- COALESCE関数

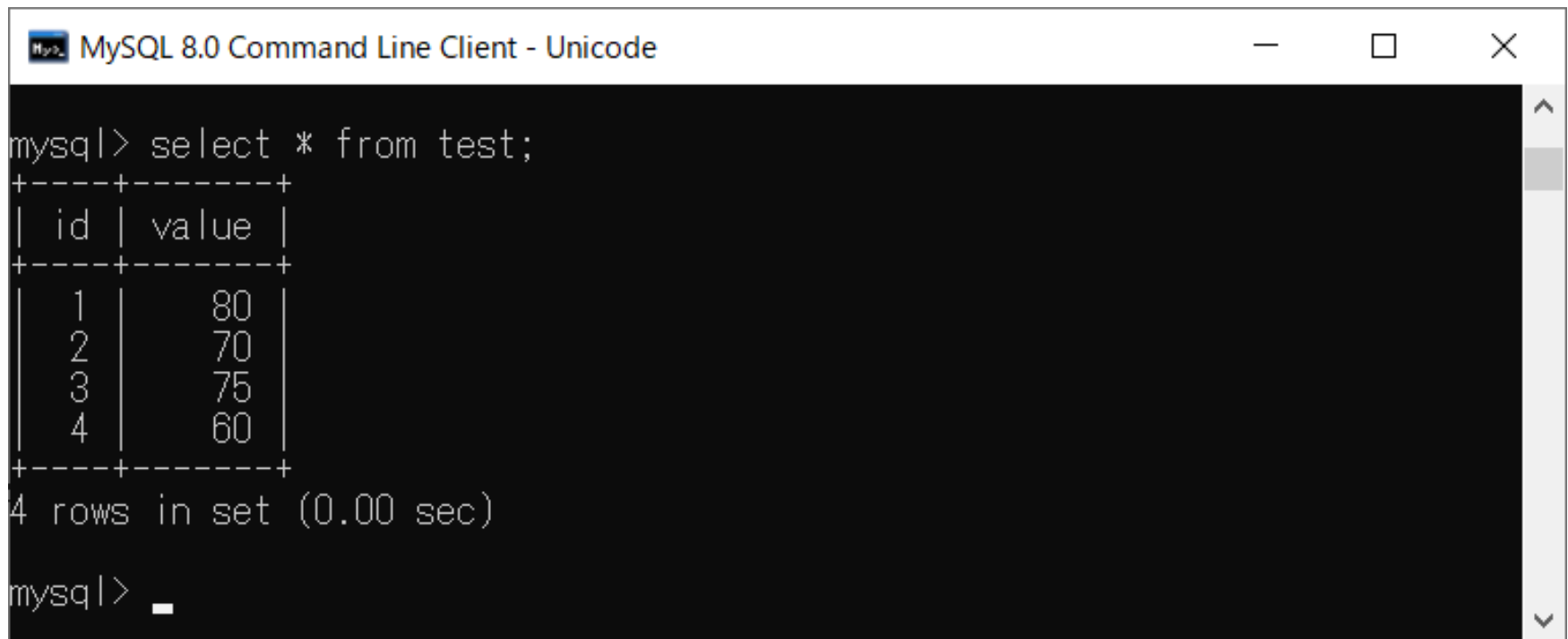
- 与えられた引数のうち, NULLでない最初の引数を返す

- CONCAT関数

- 文字列を連結

# CASEの例

- 以下の表を用意する



A screenshot of the MySQL 8.0 Command Line Client window. The title bar reads "MySQL 8.0 Command Line Client - Unicode". The terminal shows the command `mysql> select * from test;` and its output, which is a table with two columns: `id` and `value`. The table contains four rows of data. Below the table, it says "4 rows in set (0.00 sec)". The prompt `mysql>` is followed by a cursor.

```
mysql> select * from test;
+-----+-----+
| id | value |
+-----+-----+
| 1  | 80    |
| 2  | 70    |
| 3  | 75    |
| 4  | 60    |
+-----+-----+
4 rows in set (0.00 sec)

mysql> _
```

id	value
1	80
2	70
3	75
4	60

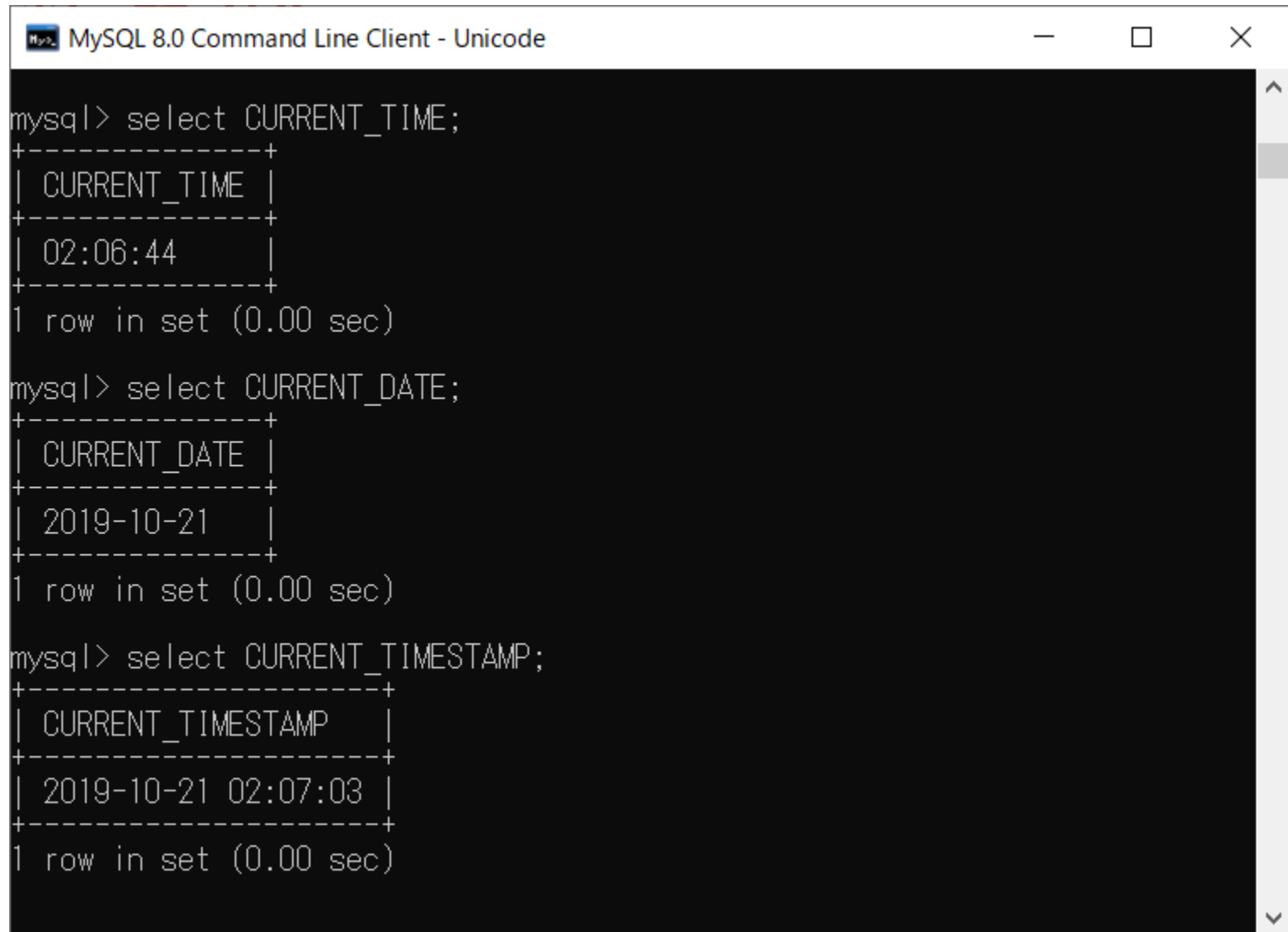
# CASEの例

```
MySQL 8.0 Command Line Client - Unicode

mysql> select id, value,
-> CASE WHEN value >=80 THEN 'A'
-> WHEN 80> value AND value>=70 THEN 'B'
-> WHEN 70> value AND value>=60 THEN 'C'
-> ELSE 'D'
-> END AS "EVAL"
-> from test;
+-----+-----+-----+
| id | value | EVAL |
+-----+-----+-----+
| 1 | 80 | A |
| 2 | 70 | B |
| 3 | 75 | B |
| 4 | 60 | C |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

# 日付関数の例



The screenshot shows a MySQL 8.0 Command Line Client window with a black background and white text. It displays three SQL queries and their results. The first query uses CURRENT\_TIME to get the current time. The second query uses CURRENT\_DATE to get the current date. The third query uses CURRENT\_TIMESTAMP to get the current timestamp. Each result is presented in a table format with a header row and a data row, followed by a message indicating the number of rows returned.

```
mysql> select CURRENT_TIME;
+-----+
| CURRENT_TIME |
+-----+
| 02:06:44      |
+-----+
1 row in set (0.00 sec)

mysql> select CURRENT_DATE;
+-----+
| CURRENT_DATE |
+-----+
| 2019-10-21    |
+-----+
1 row in set (0.00 sec)

mysql> select CURRENT_TIMESTAMP;
+-----+
| CURRENT_TIMESTAMP |
+-----+
| 2019-10-21 02:07:03 |
+-----+
1 row in set (0.00 sec)
```

# COALESCEの例(準備)

```
MySQL 8.0 Command Line Client - Unicode
mysql> create table friend(
  ->   id integer primary key,
  ->   name varchar(255),
  ->   phone1 varchar(50),
  ->   phone2 varchar(50)
  -> );
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> insert into friend set id=1, name="Tanaka", phone1="0X0-555-4444";
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> insert into friend set id=2, name="Suzuki", phone2="0X0-333-2222";
Query OK, 1 row affected (0.01 sec)

mysql> select * from friend;
+----+-----+-----+-----+
| id | name  | phone1      | phone2      |
+----+-----+-----+-----+
| 1  | Tanaka | 0X0-555-4444 | NULL        |
| 2  | Suzuki | NULL        | 0X0-333-2222 |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

# COALESCEの例

```
MySQL 8.0 Command Line Client - Unicode
mysql> select * from friend;
+----+-----+-----+-----+
| id | name  | phone1 | phone2 |
+----+-----+-----+-----+
| 1  | Tanaka | 0X0-555-4444 | NULL |
| 2  | Suzuki | NULL | 0X0-333-2222 |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select id, name, coalesce(phone1,phone2) as phone from friend;
+----+-----+-----+
| id | name  | phone |
+----+-----+-----+
| 1  | Tanaka | 0X0-555-4444 |
| 2  | Suzuki | 0X0-333-2222 |
+----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

他の使用例  
mysql> select coalesce(phone1, "AAAA") as phone1 from friend;  
とするとphone1がNULLの場合"AAAA"に置き換えて表示する

# CONCATの例

MySQL 8.0 Command Line Client - Unicode

```
mysql> select * from student;
```

id	name	dep_code	email
1	Alice	63	alice@is.jp
2	Bob	64	bob@bs.jp
3	Charlie	63	char@is.jp
4	Dave	64	dave@bs.jp

4 rows in set (0.00 sec)

```
mysql> select id, concat(name,email) from student;
```

id	concat(name,email)
1	Alicealice@is.jp
2	Bobbob@bs.jp
3	Charliechar@is.jp
4	Davedave@bs.jp

4 rows in set (0.00 sec)

# まとめ

- 集約関数という集計を行う関数がある
- ビューという仮想的なテーブルを作成することで、SQL文の簡略化、セキュリティの強化などを行うことができる
- 様々なデータ処理のための関数がある



質問あればどうぞ