

データベースシステム

第8回

理工学部情報科学科

松澤 智史

本日の内容

- ウィンドウ関数
- 副問合せ
- ストアドプロシージャ
- ストアドファンクション

SQLをファイルから実行

- 構文

mysql> source SQL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> _
```

データベースサンプル

<https://dev.mysql.com/doc/index-other.html>

- **Archives:** the documentation archives
- **About:** information about MySQL documentation and the MySQL documentation team

MySQL Server Doxygen Documentation

Title	HTML Online
MySQL Server (latest version)	View

Expert Guides

Language	Title	HTML Online	PDF
English	MySQL Internals	View	
English	MySQL Development Cycle	View	US Ltr A4

Example Databases

Title	Download DB	HTML Setup Guide	PDF Setup Guide
employee data (large dataset, includes data and test/verification suite)	GitHub	View	US Ltr A4
world database	Gzip Zip	View	US Ltr
world_x database	TGZ Zip		
sakila database	TGZ Zip	View	US Ltr A4
menagerie database	TGZ Zip		

```
mysql> use world;
Database changed
mysql> show tables;
+-----+
| Tables_in_world |
+-----+
| city             |
| country          |
| countrylanguage |
+-----+
3 rows in set (0.00 sec)

mysql> desc city;
+----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+----+-----+-----+-----+-----+-----+
| ID    | int  | NO   | PRI | NULL    | auto_increment |
| Name  | char(35) | NO | MUL |          |
| CountryCode | char(3) | NO |      |          |
| District | char(20) | NO |      |          |
| Population | int | NO |      | 0       |
+----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> select * from city where District='Chiba';
+----+-----+-----+-----+-----+
| ID | Name      | CountryCode | District | Population |
+----+-----+-----+-----+-----+
| 1544 | Chiba     | JPN         | Chiba    | 863930     |
| 1551 | Funabashi | JPN         | Chiba    | 545299     |
| 1560 | Matsudo  | JPN         | Chiba    | 461126     |
| 1563 | Ichikawa | JPN         | Chiba    | 441893     |
| 1593 | Kashiwa  | JPN         | Chiba    | 320296     |
| 1607 | Ichihara | JPN         | Chiba    | 279280     |
| 1653 | Sakura   | JPN         | Chiba    | 168072     |
| 1662 | Yachiyo  | JPN         | Chiba    | 161222     |
| 1669 | Narashino | JPN        | Chiba    | 152849     |
| 1676 | Nagareyama | JPN       | Chiba    | 147738     |
| 1698 | Urayasu  | JPN         | Chiba    | 127550     |
| 1703 | Abiko     | JPN         | Chiba    | 126670     |
| 1711 | Kisarazu | JPN         | Chiba    | 121967     |
| 1712 | Noda      | JPN         | Chiba    | 121030     |
| 1754 | Kamagaya | JPN         | Chiba    | 100821     |
| 1757 | Nishio    | JPN         | Chiba    | 100032     |
| 1771 | Kimitsu  | JPN         | Chiba    | 93216      |
| 1776 | Mobara    | JPN         | Chiba    | 91664      |
| 1777 | Narita    | JPN         | Chiba    | 91470      |
+----+-----+-----+-----+-----+
19 rows in set (0.00 sec)

mysql>
```

世界の都市のデータ(4000以上)等が含まれる

ウィンドウ関数(分析関数)

- 現在の行に関するテーブル全体を舐める計算をする
- 集約関数と同じく集計動作を行う
 - 集約関数は1行に集約するが, ウィンドウ関数を使った場合は対象の行はそのまま残る
 - 指定対象の行全てに対して処理が行われる(範囲指定は可能)
- OVER句を用いて表される
- 構文
 - ウィンドウ関数 over (partition by 列名) from テーブル名 →fromの前にover句を書く
 - ウィンドウ関数 over (order by 列名) from テーブル名
 - ※ ウィンドウ関数でないavg等の集約関数も使える(overを使えばウィンドウ関数になる)
- 範囲対象に使用する列をPARTITION BY句, ORDER BY句で指定する
 - PARTITION BY句は集約関数のGROUP BY句とほぼ同じ動作をする

集約関数

```
mysql> select *, avg(value) as avg from student natural join test group by dep_code;
```

id	name	dep_code	email	value	avg
1	Alice	63	alice@is.jp	80	77.5000
2	Bob	64	bob@bs.jp	70	65.0000

group by は
一行に集約

2 rows in set (0.01 sec)

```
mysql>
```

ウィンドウ関数

```
mysql> select *, avg(value) over (partition by dep_code) as avg from student natural join test;
```

id	name	dep_code	email	value	avg
1	Alice	63	alice@is.jp	80	77.5000
3	Charlie	63	char@is.jp	75	77.5000
2	Bob	64	bob@bs.jp	70	65.0000
4	Dave	64	dave@bs.jp	60	65.0000

2行残る

4 rows in set (0.00 sec)

```
mysql>
```

```
mysql> select *, avg(value) over (order by dep_code desc) as avg from student natural join test;
```

id	name	dep_code	email	value	avg
2	Bob	64	bob@bs.jp	70	65.0000
4	Dave	64	dave@bs.jp	60	65.0000
1	Alice	63	alice@is.jp	80	71.2500
3	Charlie	63	char@is.jp	75	71.2500

77.5ではない？

4 rows in set (0.00 sec)

```
mysql>
```

ウィンドウ関数のorder by

- order by XXXは, XXXによって順番を並び替える ←ここまでは同じ
- ウィンドウ関数のorder by句は
行を順番に並べた上で最初の行から現在行までのみを集計の対象にする

```
mysql> select *, avg(value) over (order by dep_code desc) as avg from student natural join test;
```

id	name	dep_code	email	value	avg
2	Bob	64	bob@bs.jp	70	65.0000
4	Dave	64	dave@bs.jp	60	65.0000
1	Alice	63	alice@is.jp	80	71.2500
3	Charlie	63	char@is.jp	75	71.2500

4 rows in set (0.00 sec)

```
mysql>
```

1行目, 2行目のvalue値も
加味された平均値となる

ウィンドウ関数専用の関数

- ROW_NUMBER()
- RANK()
- DENSE_RANK()
- PERCENT_RANK()
- CUME_DIST()
- NTILE()
- LAG()
- LEAD()
- FIRST_VALUE()
- LAST_VALUE()
- NTH_VALUE()

ROW_NUMBER関数

- 1からNまでの現在行の数を返す

```
mysql> select *, row_number() over (order by dep_code) as RN from student natural join test;
```

id	name	dep_code	email	value	RN
1	Alice	63	alice@is.jp	80	1
3	Charlie	63	char@is.jp	75	2
2	Bob	64	bob@bs.jp	70	3
4	Dave	64	dave@bs.jp	60	4

```
4 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> select *, row_number() over (order by Population desc) as Ranking from city limit 20;
```

ID	Name	CountryCode	District	Population	Ranking
1024	Mumbai (Bombay)	IND	Maharashtra	10500000	1
2331	Seoul	KOR	Seoul	9981619	2
206	São Paulo	BRA	São Paulo	9968485	3
1890	Shanghai	CHN	Shanghai	9696300	4
939	Jakarta	IDN	Jakarta Raya	9604900	5
2822	Karachi	PAK	Sindh	9269265	6
3357	Istanbul	TUR	Istanbul	8787958	7
2515	Ciudad de México	MEX	Distrito Federal	8591309	8
3580	Moscow	RUS	Moscow (City)	8389200	9
3793	New York	USA	New York	8008278	10
1532	Tokyo	JPN	Tokyo-to	7980230	11
1891	Peking	CHN	Peking	7472000	12
456	London	GBR	England	7285000	13
1025	Delhi	IND	Delhi	7206704	14
608	Cairo	EGY	Kairo	6789479	15
1380	Teheran	IRN	Teheran	6758845	16
2890	Lima	PER	Lima	6464693	17
1892	Chongqing	CHN	Chongqing	6351600	18
3320	Bangkok	THA	Bangkok	6320174	19
2257	Santafé de Bogotá	COL	Santafé de Bogotá	6260862	20

```
20 rows in set (0.00 sec)
```

```
mysql>
```

例:世界の都市の人口ランキング上位20都市

RANK関数

- 現在行の順位を返す(同率の番号を飛ばす)

```
mysql> select *, rank() over (order by dep_code) as R from student natural join test;
```

id	name	dep_code	email	value	R
1	Alice	63	alice@is.jp	80	1
3	Charlie	63	char@is.jp	75	1
2	Bob	64	bob@bs.jp	70	3
4	Dave	64	dave@bs.jp	60	3

```
4 rows in set (0.00 sec)
```

```
mysql> _
```

DENSE_RANK関数

- 現在行の順位を返す(同率の番号を飛ばさない)

```
mysql> select *, dense_rank() over (order by dep_code) as R from student natural join test;
```

id	name	dep_code	email	value	R
1	Alice	63	alice@is.jp	80	1
3	Charlie	63	char@is.jp	75	1
2	Bob	64	bob@bs.jp	70	2
4	Dave	64	dave@bs.jp	60	2

```
4 rows in set (0.00 sec)
```

```
mysql>
```

PERCENT_RANK関数

- 現在行の相対順位比率
- 計算方法は(ランキング-1)/(全行数-1)

```
mysql> select *, percent_rank() over (order by id) as R from student natural join test;
```

id	name	dep_code	email	value	R
1	Alice	63	alice@is.jp	80	0
2	Bob	64	bob@bs.jp	70	0.3333333333333333
3	Charlie	63	char@is.jp	75	0.6666666666666666
4	Dave	64	dave@bs.jp	60	1

```
4 rows in set (0.00 sec)
```

```
mysql> _
```

CUME_DIST関数

- 現在行の相対順位比率
- 計算方法は(ランキング/全行数)

```
mysql> select *, cume_dist() over (order by id) as R from student natural join test;
```

id	name	dep_code	email	value	R
1	Alice	63	alice@is.jp	80	0.25
2	Bob	64	bob@bs.jp	70	0.5
3	Charlie	63	char@is.jp	75	0.75
4	Dave	64	dave@bs.jp	60	1

```
4 rows in set (0.00 sec)
```

```
mysql>
```

NTILE関数

- 引数で指定した数に分割する
- 100行を5分割する場合は20行ずつ採番される

```
mysql> select *, ntile(2) over (order by id) as NT from student natural join test;
```

id	name	dep_code	email	value	NT
1	Alice	63	alice@is.jp	80	1
2	Bob	64	bob@bs.jp	70	1
3	Charlie	63	char@is.jp	75	2
4	Dave	64	dave@bs.jp	60	2

```
4 rows in set (0.00 sec)
```

```
mysql> _
```

LAG関数

- 前の行を返す

```
mysql> select *, lag(dep_code) over (order by id) as LG from student natural join test;
```

id	name	dep_code	email	value	LG
1	Alice	63	alice@is.jp	80	NULL
2	Bob	64	bob@bs.jp	70	63
3	Charlie	63	char@is.jp	75	64
4	Dave	64	dave@bs.jp	60	63

```
4 rows in set (0.00 sec)
```

```
mysql> S
```

LEAD関数

- 後の行(または後の行の1つ)を返す

```
mysql> select *, lead(dep_code) over (order by id) as LG from student natural join test;
```

id	name	dep_code	email	value	LG
1	Alice	63	alice@is.jp	80	64
2	Bob	64	bob@bs.jp	70	63
3	Charlie	63	char@is.jp	75	64
4	Dave	64	dave@bs.jp	60	NULL

```
4 rows in set (0.00 sec)
```

```
mysql> _
```


FIRST_VALUE関数

- 最初の行の値を返す

```
mysql> select *, first_value(name) over (order by id) as FV from student natural join test;
```

id	name	dep_code	email	value	FV
1	Alice	63	alice@is.jp	80	Alice
2	Bob	64	bob@bs.jp	70	Alice
3	Charlie	63	char@is.jp	75	Alice
4	Dave	64	dave@bs.jp	60	Alice

```
4 rows in set (0.00 sec)
```

```
mysql> _
```

LAST_VALUE関数

- 最後の行の値を返す(ただしその行までの最後の値)

```
mysql> select *, last_value(email) over (order by id) as LV from student natural join test;
```

id	name	dep_code	email	value	LV
1	Alice	63	alice@is.jp	80	alice@is.jp
2	Bob	64	bob@bs.jp	70	bob@bs.jp
3	Charlie	63	char@is.jp	75	char@is.jp
4	Dave	64	dave@bs.jp	60	dave@bs.jp

```
4 rows in set (0.00 sec)
```

```
mysql> _
```

NTH_VALUE関数

- 1から数えたN番目の行の値を返す
- Nが3の場合, 1~2行目はNULL

```
mysql> select *, nth_value(email,3) over (order by id) as NV from student natural join test;
```

id	name	dep_code	email	value	NV
1	Alice	63	alice@is.jp	80	NULL
2	Bob	64	bob@bs.jp	70	NULL
3	Charlie	63	char@is.jp	75	char@is.jp
4	Dave	64	dave@bs.jp	60	char@is.jp

```
4 rows in set (0.00 sec)
```

```
mysql> _
```

副問合せ

- SQL文の中に入れ子でSELECT文を入れることができる

```
mysql> select * from student where id in (select id from test where value>70);
```

id	name	dep_code	email
1	Alice	63	alice@is.jp
3	Charlie	63	char@is.jp

```
2 rows in set (0.00 sec)
```

```
mysql> _
```

- ()内のSELECT文を副問合せ,
冒頭のSELECT文を主問合せという

ストアドプロシージャ

- いくつかのSQL文を1つにまとめ、CALLというコマンドで実行できるようにしたもの

- 構文

```
CREATE PROCEDURE プロシージャ名(引数)  
BEGIN
```

```
    SQL文
```

```
END
```

ストアドプロシージャの例

```
mysql> delimiter //
```

```
mysql> create procedure p1()  
    -> begin  
    -> select * from student;  
    -> end  
    -> //
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> delimiter ;  
mysql> call p1;
```

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)

Query OK, 0 rows affected (0.02 sec)

ストアドプロシージャの例2

```
mysql> delimiter //  
mysql> create procedure p2(i integer)  
-> begin  
-> select * from student where id=i;  
-> end  
-> //
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> delimiter ;  
mysql> call p2(3);
```

id	name	dep_code	email
3	Charlie	63	char@is.jp

1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

ストアドプロシージャの利点

- 作成されたときにプリコンパイルされるのでSQL文を一つずつ呼ぶより高速に動作する
- プログラムから呼び出される場合などは, SQL文複数を送るよりプロシージャを一つ呼ぶ方が通信量削減に繋がる

ストアドプロシージャの一覧

```
SELECT
  ROUTINE_SCHEMA, /* ストアドプロシージャがあるデータベース */
  ROUTINE_NAME,   /* ストアドプロシージャの名前 */
  ROUTINE_TYPE    /* プロシージャとファンクションのどちらかを示す */
FROM
  information_schema.ROUTINES
WHERE
  ROUTINE_TYPE = 'PROCEDURE'; /* プロシージャのみ抽出 */
```

※ファイルにしておくと便利

実行例

```
mysql> source show_procedure.sql;
```

ROUTINE_SCHEMA	ROUTINE_NAME	ROUTINE_TYPE
sys	create_synonym_db	PROCEDURE
sys	execute_prepared_stmt	PROCEDURE
sys	diagnostics	PROCEDURE
sys	ps_statement_avg_latency_histogram	PROCEDURE
sys	ps_trace_statement_digest	PROCEDURE
sys	ps_trace_thread	PROCEDURE
sys	ps_setup_disable_background_threads	PROCEDURE
sys	ps_setup_disable_consumer	PROCEDURE
sys	ps_setup_disable_instrument	PROCEDURE
sys	ps_setup_disable_thread	PROCEDURE
sys	ps_setup_enable_background_threads	PROCEDURE
sys	ps_setup_enable_consumer	PROCEDURE
sys	ps_setup_enable_instrument	PROCEDURE
sys	ps_setup_enable_thread	PROCEDURE
sys	ps_setup_reload_saved	PROCEDURE
sys	ps_setup_reset_to_default	PROCEDURE
sys	ps_setup_save	PROCEDURE
sys	ps_setup_show_disabled	PROCEDURE
sys	ps_setup_show_disabled_consumers	PROCEDURE
sys	ps_setup_show_disabled_instruments	PROCEDURE
sys	ps_setup_show_enabled	PROCEDURE
sys	ps_setup_show_enabled_consumers	PROCEDURE
sys	ps_setup_show_enabled_instruments	PROCEDURE
sys	ps_truncate_all_tables	PROCEDURE
sys	statement_performance_analyzer	PROCEDURE
sys	table_exists	PROCEDURE
sakila	rewards_report	PROCEDURE
sakila	film_in_stock	PROCEDURE
sakila	film_not_in_stock	PROCEDURE
test_db	pr3	PROCEDURE
test_db	pr1	PROCEDURE

```
31 rows in set (0.00 sec)
```

ストアドファンクション

- 基本的にストアドプロシージャと同じであるが関数であるため、返り値が必要
- 構文
CREATE FUNCTION ファンクション名(引数) RETURNS 型
DETERMINISTIC
BEGIN
文
END
- DETERMINISTICは返り値が決定的(入力と同じ場合は出力が同じ)の場合に指定する
※違う場合はNOT DETERMINISTICを指定する

ストアドファンクションの例

```
mysql> delimiter //
mysql> create function f(i integer) returns integer deterministic
    -> begin
    -> return (i+i);
    -> end
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;
mysql> select f(10);
+-----+
| f(10) |
+-----+
|    20 |
+-----+
1 row in set (0.00 sec)

mysql> _
```

まとめ

- ウィンドウ関数を用いるとグループ化せずに様々な集計が可能になる
- SQL文(SELECT)を入れ子にして記述する問合せを副問合せと呼ぶ
- SQLを用いた一連の処理をまとめたものをストアドプロシージャと呼びCALLで実行できる
- ユーザ定義の関数(ストアドファンクション)を定義することができる

質問あればどうぞ