データベース実習

MySQL補足

株式会社ジードライブ

今回学ぶこと

- データの追加
- DISTINCT
- BETWEEN / IN
- 関数(文字列操作、算術、日時操作)
- IF / IFNULL
- HAVING
- UNION
- サブクエリ
- VIEW
- ユーザーと権限

データの追加

- 直前にAUTO_INCREMETNT で自動生成されたIDは、LAST_INSERT_ID() で取得できる

```
INSERT INTO members (id, name, age, address, created)
VALUES (NULL, '斎藤静子', 25, '東京都', NOW());

SELECT LAST_INSERT_ID();
```

id にNULLを指定
⇒ AUTO_INCREMENTでidが自動生成される

- ON DUPLICATE KEY UPDATE 構文を使用すると、データを追加する際に、主キーやユニークキーに重複がある場合は、更新処理にすることができる

例：idが重複する場合、name, age, addressを更新する（createdは更新しない）

```
INSERT INTO members (id, name, age, address, created)
VALUES (10, '大田洋介', 33, '宮城県', NOW()) AS params
ON DUPLICATE KEY UPDATE
name = params.name, age = params.age, address = params.address;
```

BETWEEN / IN

- WHERE句で範囲を指定する場合、BETWEENが便利

```
-- WHERE age >= 30 AND age <= 39 と同義
SELECT * FROM members
WHERE age BETWEEN 30 AND 39;
```

- WHERE句で複数の条件を列挙する場合、INが便利

```
-- WHERE age = 20 OR age = 30 OR age = 40 OR age = 50 と同義
SELECT * FROM members
WHERE age IN (20, 30, 40, 50);
```

```
-- WHERE age != 20 AND age != 30 AND age != 40 AND age != 50 と同義
SELECT * FROM members
WHERE age NOT IN (20, 30, 40, 50);
```

文字列操作の関数

関数	説明
CONCAT(str1, str2, str3, ...)	文字列結合
LOWER(str) / UPPER(str)	小文字に変換 / 大文字に変換
LENGTH(str)	文字列のバイト数を返す → utm8mb4の場合、日本語１文字は３としてカウントされる → 実際の文字数は、CHAR_LENGTH()を使用
TRIM(str)	行頭・行末のスペースを除去
REPLACE(str, s1, s2)	s1をs2で置き換える
SUBSTRING(str, position, length)	文字列の一部を抜き出す
POSITION(substr IN str)	substrの出現箇所を返す
SUBSTRING_INDEX(str, delim, count)	delimは区切り文字。区切り文字で文字列を分割し、そのうちの何番目をを返すかを第三引数で指定する → 第三引数をマイナスにすることで、後ろから数えることができる

https://dev.mysql.com/doc/refman/8.0/ja/string-functions.html

算術用の関数

関数	説明
PI()	円周率を返す
POWER(num, n)	num の n乗 を返す
RAND()	１未満のランダムな数値を返す
ROUND(num)	小数点第一位を四捨五入
SIGN(num)	正の数の場合１を返す 負の数の場合ー１を返す ０の場合０を返す
SQRT(num)	２乗した値を返す

https://dev.mysql.com/doc/refman/8.0/ja/numeric-functions.html

日時操作の関数

関数	説明
CURDATE()	現在の年月日を返す
CURTIME()	現在時刻を返す
NOW()	現在日時を返す
YEAR(date)	年を返す
MONTH(date)	月を返す
MONTHNAME(date)	月の名前を返す（January, Februaryなど）
DAY (date)	日を返す
DAYNAME(date)	曜日名を返す（Sunday, Mondayなど）
DATE_FORMAT(date, format_str)	日付をフォーマットして返す
TIMESTAMPDIFF(unit, date1, date2)	２つの日時の差分を返す

https://dev.mysql.com/doc/refman/8.0/ja/date-and-time-functions.html

関数の利用例

例：姓と名を結合する。生年月日をフォーマットする。年齢を算出する

```
SELECT
  CONCAT(lname, fname) AS name,
  DATE_FORMAT(birthday, '%Y年%m月%d日') AS birthday,
  TIMESTAMPDIFF(YEAR, birthday, CURDATE()) AS age
FROM students ORDER BY age;
```

id	lname	fname	birthday
1	山田	太郎	2000-01-01
2	佐藤	葉子	1999-03-15
3	三沢	花子	1999-11-27
4	本吉	信二	2001-10-18
5	草野	史郎	2000-07-29

name	birthday	age
本吉信二	2001年10月18日	20
山田太郎	2000年01月01日	21
三沢花子	1999年11月27日	21
草野史郎	2000年07月29日	21
佐藤葉子	1999年03月15日	22

IF / IFNULL

- IFやIFNULLを使用することで、条件に応じたデータを返すことができる

```
SELECT name,
    IFNULL(score, 0) AS score,
    IF(score >= 70, '合格', '不合格') AS result
FROM scores;
```

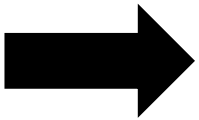
score が NULLの場合のデータ

条件

trueの場合

falseの場合

id	name	score
1	山田太郎	80
2	鈴木次郎	60
3	田中花子	
4	太田充希	70
5	藤本信二	50



name	score	result
山田太郎	80	合格
鈴木次郎	60	不合格
田中花子	0	不合格
太田充希	70	合格
藤本信二	50	不合格

DISTINCT

- 重複したデータをまとめるにはDISTINCTを使用する

id	name	club	grade
1	山田太郎	生物部	2
2	田中花子	科学部	1
3	佐藤光一	文芸部	3
4	木村洋子	科学部	1
5	本田紳助	科学部	2
6	真田光	生物部	2
7	大谷みゆき	将棋部	1
8	志村智	生物部	1
9	藤原充希	科学部	1
10	黒川裕也	将棋部	3

```
SELECT DISTINCT(club)
FROM students;
```

生徒が所属する部活の種類

club
生物部
科学部
文芸部
将棋部

```
SELECT DISTINCT(club)
FROM students
WHERE grade = 1;
```

1年生が所属する部活の種類

club
科学部
将棋部
生物部

HAVING

- 集計関数でグループ化されたデータに対しては、WHEREではなく、HAVINGで絞り込みの条件を指定する

id	name	club	score
1	山田太郎	生物部	80
2	田中花子	科学部	75
3	佐藤光一	文芸部	75
4	木村洋子	科学部	80
5	本田紳助	科学部	60
6	真田光	生物部	40
7	大谷みゆき	将棋部	55
8	志村智	生物部	65
9	藤原充希	科学部	70
10	黒川裕也	将棋部	60

```
SELECT club, AVG(score)
FROM students GROUP BY club;
```

部活ごとの平均点

club	AVG(score)
生物部	61.6667
科学部	71.2500
文芸部	75.0000
将棋部	57.5000

```
SELECT club, AVG(score)
FROM students GROUP BY club
HAVING AVG(score) >= 70;
```

平均点が70点以上の部活

club	AVG(score)
科学部	71.2500
文芸部	75.0000

UNION

- SELECT文の結果をテーブルを縦に結合することができる

```
SELECT 's' AS type, CONCAT(lname, fname) AS name FROM students
UNION
SELECT 'c' AS type, name FROM clubs;
```

studentsテーブル

id	lname	fname	birthday
1	山田	太郎	2000-01-01
2	佐藤	葉子	1999-03-15
3	三沢	花子	1999-11-27

clubsテーブル

id	name	teacher
1	卓球部	本田恵一
2	茶道部	木下光

type	name
s	山田太郎
s	佐藤葉子
s	三沢花子
c	卓球部
c	茶道部

サブクエリ

- FROM句やJOIN句などでは、別のSELECT文を使用することができる
  - サブクエリと呼ぶ

```
SELECT club, average
FROM (SELECT club, AVG(grade) AS average FROM students
      GROUP BY club) AS avg_table
WHERE average >= 70;
```

サブクエリ

FROM句やJOIN句でサブクエリを使用する場合は、ASでテーブル名を付ける

サブクエリの結果、一時的に生成されたavg_table

club	average
生物部	61.6667
科学部	71.2500
文芸部	75.0000
将棋部	57.5000

WHERE average >= 70

最終的な結果

club	average
科学部	71.2500
文芸部	75.0000

サブクエリ

- サブクエリの使用例

```
SELECT * FROM students WHERE club IN (
    SELECT club FROM (
        SELECT club, AVG(score) AS average FROM students
        GROUP BY club HAVING average >= 70) AS avg_table);
```

サブクエリ1

club	average
科学部	71.2500
文芸部	75.0000

サブクエリ2

club
科学部
文芸部

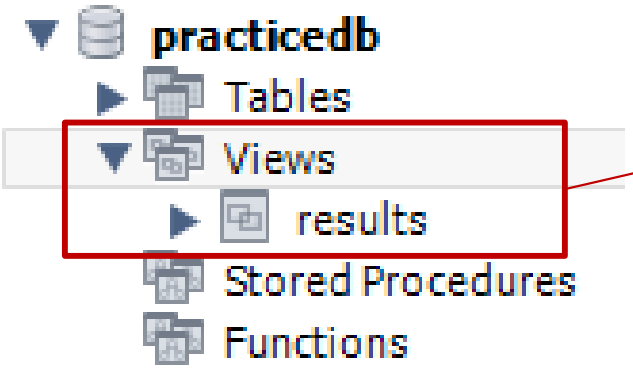
最終結果

id	name	club	score
2	田中花子	科学部	75
3	佐藤光一	文芸部	75
4	木村洋子	科学部	80
5	本田紳助	科学部	60
9	藤原充希	科学部	70

VIEW

- 同じSELECT文を何度も書く必要がある場合は、VIEWとして登録しておくと便利
  - 仮想的なテーブルとして登録することができる

```
CREATE VIEW results AS
    SELECT name,
      IFNULL(score, 0) AS score,
      IF(score >= 70, '合格', '不合格') AS result
    FROM scores;
```



登録されているVIEW

VIEW

- VIEWは、通常のテーブルと同様にSELECT文で利用することができる

```
SELECT * FROM results WHERE score >= 50;
```

VIEWの名前

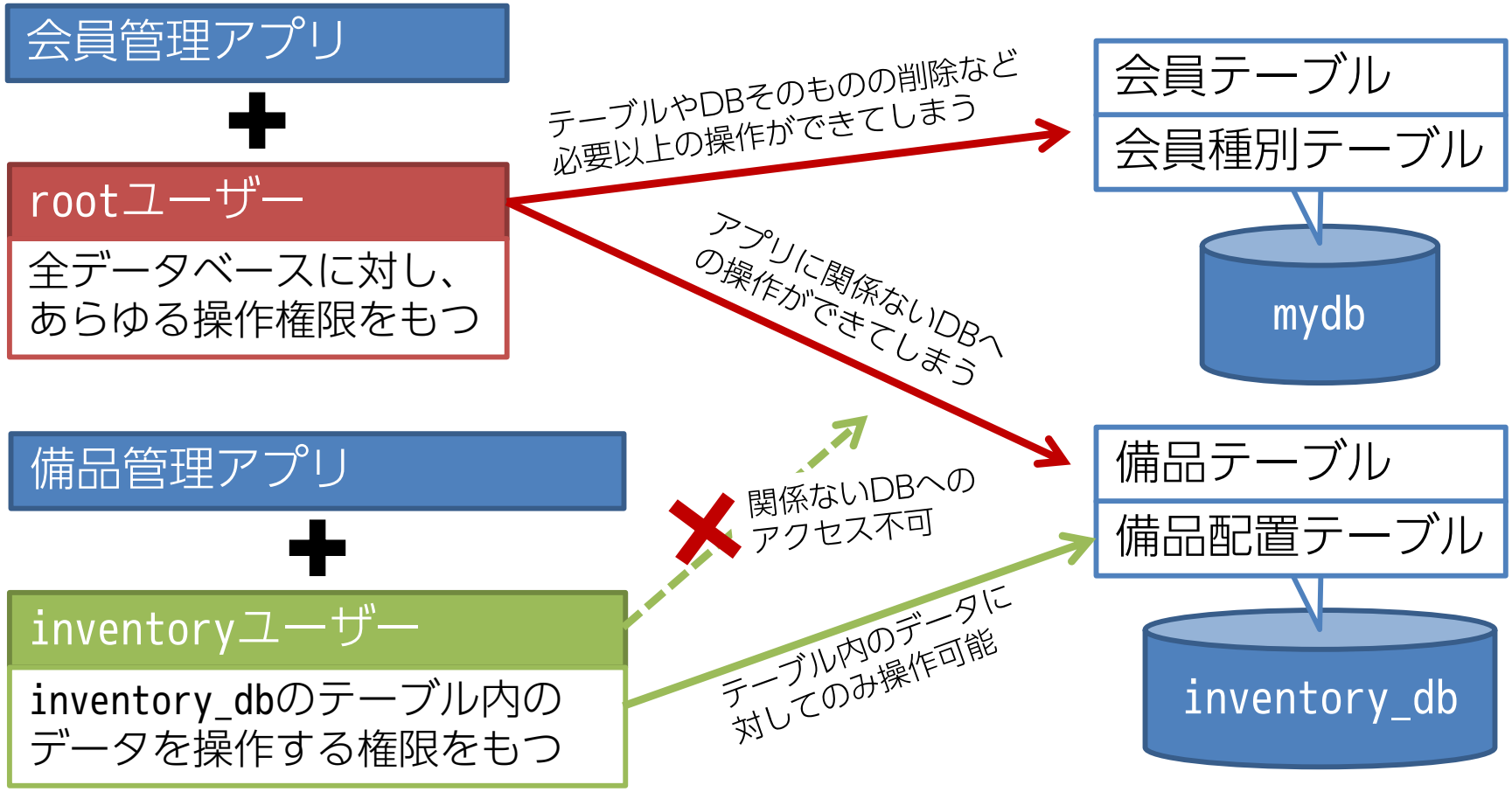
- VIEWは、DROP VIEWコマンドで削除できる

```
DROP VIEW results;
```

削除するVIEWの名前

ユーザーと権限

- アプリケーションごとに、適切な**ユーザー**を紐付けるすることで、データベースに対する操作を制限することができる



ユーザーの追加

- ユーザーの一覧は、mysql.user テーブルで確認できる

```
SELECT * FROM mysql.user;
```

- ユーザーはCREATE USERコマンドで追加する
  - ユーザーは、ユーザー名@ホスト の形式で指定する
  - パスワードは IDENTIFIED BY で設定する

例： taroというユーザーを作成。パスワードは abcd

```
CREATE USER taro@localhost
IDENTIFIED BY 'abcd';
```

- ユーザーの削除は DROP USERコマンドで行う

```
DROP USER taro@localhost;
```

権限の付与

- ユーザーに付与されている権限の確認

```
SHOW GRANTS FOR taro@localhost;
```

表示例

```
GRANT USAGE ON *.* TO taro@localhost
```

  - USAGE ON *. * ：全データベースの全テーブルに対して権限がないことを示す

- GRANT構文で権限を付与することができる

例：mydbの全テーブルに対し、SELECT, INSERT, UPDATE, DELETEを実行できる

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON mydb.* TO taro@localhost;
```

可能な操作を列挙:
テーブルそのものの操作権限を与えたい場合は、
CREATE, DROP, INDEX, ALTER 等も記載する

データベース名　　テーブル名: 全テーブルを指定する場合には * を記述する