データベース実習

04. MySQLによるデータ操作

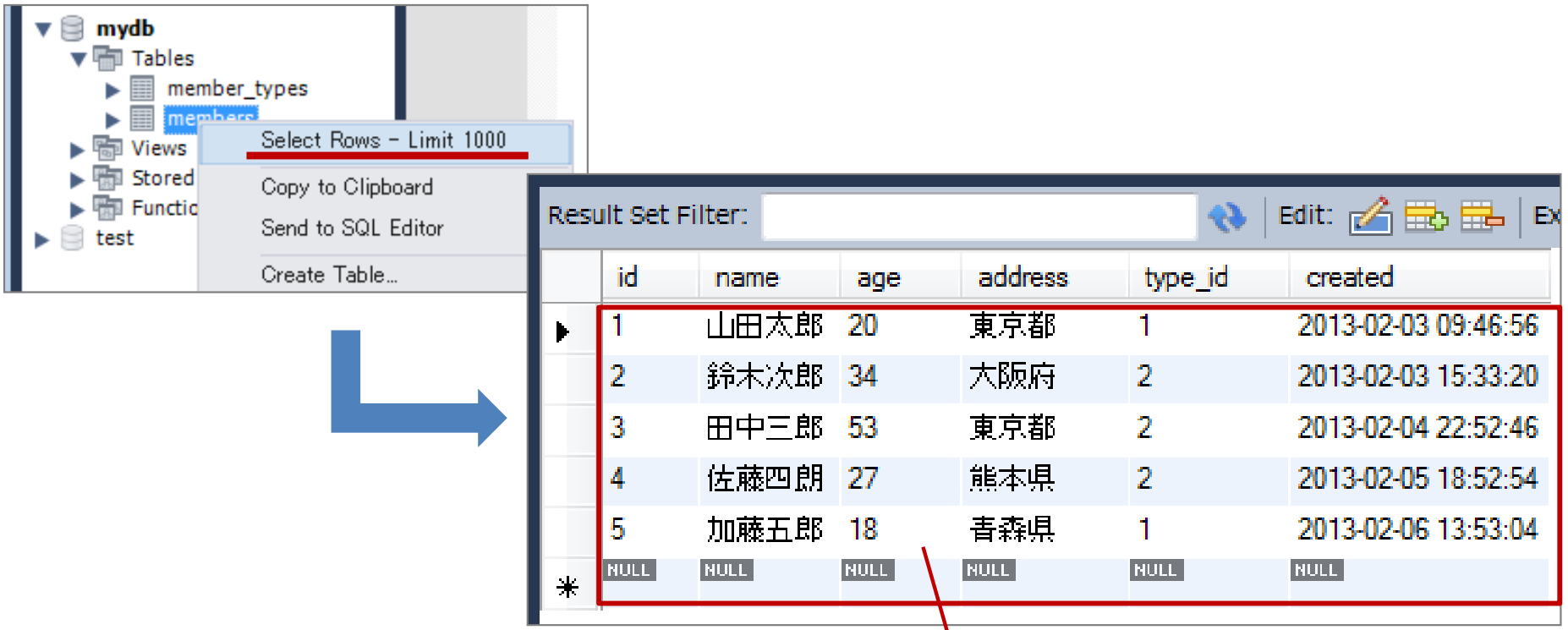
株式会社ジードライブ

今回学ぶこと

- データの追加、更新、削除
- データの検索
- データの集計やソート
- データベースのエクスポートとインポート

データの表示と操作

- SCHEMASパネルでテーブルを右クリックして[Select Rows - Limit 1000]を選択する



画面上でデータの操作が可能
(操作後はApplyボタンで確定する必要がある)

SQLによるデータ操作

データ操作

- SQLを使用したデータ操作
  - **INSERT** … データの追加
  - **UPDATE** … データの更新
  - **DELETE** … データの削除

データの追加（1）

- 書式

```
INSERT INTO テーブル名 VALUES (値１,値２,…);
```

- 全てのカラムの値をテーブルの定義通りの順番で指定する必要がある
  - 数値やNULLはそのまま記述する
  - 文字列や日時はシングルクォートかダブルクォートで囲む

例：membersテーブルに「山田太郎」のデータを追加

```
INSERT INTO members VALUES  (NULL,'山田太郎',20,'東京都',1,NOW());
```

- AUTO_INCREMENTのカラムはNULLを指定可能
- NOW() は現在の日時を取得する関数

データの追加 (2)

- 書式

```
INSERT INTO テーブル名（カラム１,カラム２,…） VALUES（値１,値２,…);
```

- NOT NULL に設定されておりDEFAULTが設定されていないカラムは省略不可
- AUTO_INCREMENTが設定されているカラムは省略可

例：membersテーブルに「鈴木次郎」のデータを追加

```
INSERT INTO members (name,type_id,created) VALUES ('鈴木次郎',1,NOW());
```

データの更新

- 書式

```
UPDATE テーブル名 SET カラム1=値1,カラム2=値2 WHERE 条件;
```

- WHEREで更新対象データを指定する

  ⇒ 指定がない場合、全データが変更されてしまう
- 複数のカラムを同時に更新する場合はカンマで区切って列挙する

例：idが1番の、ageカラムのデータを21に変更する

```
UPDATE members SET age=21 WHERE id=1;
```

これがない場合、membersテーブル内の全ageカラムが21になる

データの削除

- 書式

```
DELETE FROM テーブル名 WHERE 条件;
```

- WHEREで削除対象データを指定する
  ⇒ 指定がない場合、全データが削除されてしまう

例：membersテーブルからidが2番の会員データを削除

```
DELETE FROM members WHERE id=2;
```

これがない場合、membersテーブル内の全データが削除される

データの削除について

- 削除したデータは元に戻すことができない
  ⇒ 実際にデータを削除する(物理削除)のではなく、
  状態を管理するためのカラムを用意しておき、
  削除状態に更新する方法(論理削除)もある

id	name	email	status	created
1	山田太郎	taro@abc.com	ACTIVE	2020-01-05
2	田中花子	hana@abc.com	ACTIVE	2020-01-06
3	本田次郎	honda@abc.com	DELETE	2020-02-10
4	鈴木三郎	suzuki@abc.com	ACTIVE	2020-02-11

データを削除するのではなく、UPDATE文を使い、削除状態を設定する

物理削除	`DELETE FROM members WHERE id=3;`
論理削除	`UPDATE members SET status = "DELETE" WHERE id = 3;`

練習

- 練習04-1
- 練習04-2
- 練習04-3

データの検索

- 書式

```
SELECT カラム名 FROM テーブル名 WHERE 条件;
```

- カラム名はカンマで区切って列挙する
- カラム名を「*」とすると全カラムを取得
- WHEREで検索対象を指定する
  - 指定が無い場合は全データが取得される

例：idが１番の会員氏名と年齢を取得

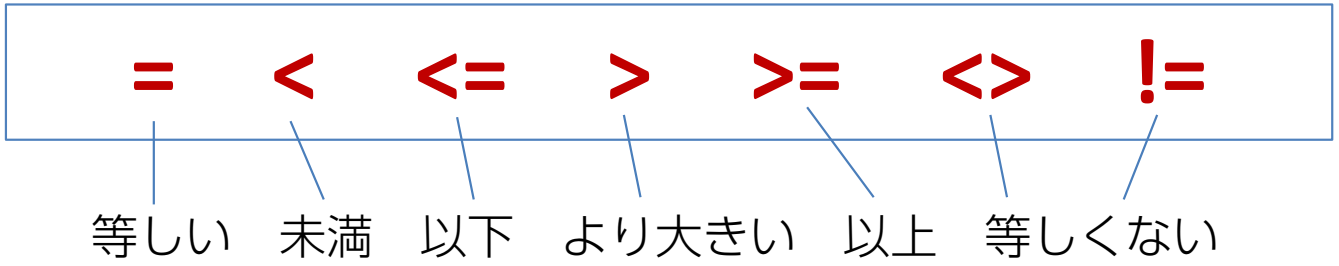
```
SELECT name, age FROM members WHERE id=1;
```

例：membersテーブルから全データを取得

```
SELECT * FROM members;
```

WHERE句で使える演算子

- WHERE句で以下の記号を使い、条件に応じたデータを検索できる



例：氏名が「山田太郎」の会員データを取得

```
SELECT * FROM members WHERE name='山田太郎';
```

例：年齢が20歳以上の会員を検索

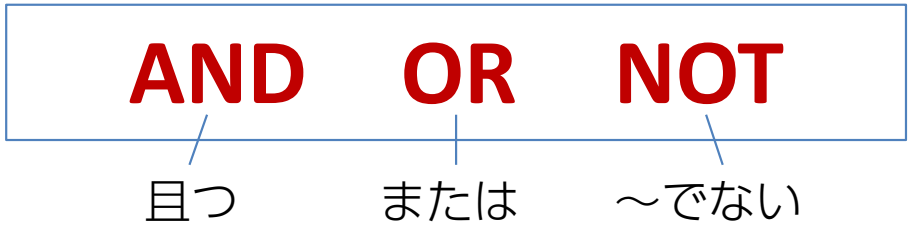
```
SELECT * FROM members WHERE age>=20;
```

例：createdカラムの日付が2005年5月5日以降の会員情報を取得

```
SELECT * FROM members WHERE created>='2005-05-05';
```

WHERE句で使える演算子

- 条件式を接続するために以下の演算子を使用することができる



  - ANDとORではANDの方が優先順位が高い

例：20歳以上で且つ2005年5月5日以降に登録された会員を検索

```
SELECT * FROM members WHERE age>=20 AND created>='2005-05-05';
```

例：20歳以上、または住所が東京都でない会員を検索

```
SELECT * FROM members WHERE age>=20 OR NOT address='東京都';
```

LIKE

- WHERE句に以下の書式を記述することで、特定の文字列を含むデータを検索することができる

```
WHERE カラム名 LIKE 'パターン'
```

- % や _ の記号を使い、パターンを示す
  - % … 0個以上の任意の文字列を表す
  - _ … 任意の1文字を表す

例：住所に「渋谷区」を含む会員を取得

```
SELECT * FROM members WHERE address LIKE '%渋谷区%';
```

例：氏名が「山田○○」である会員を取得

```
SELECT * FROM members WHERE name LIKE '山田__';
```

練習

- 練習04-4
- 練習04-5

集計関数

- MAX(カラム名)
  - 検索されたデータの最大値を取得する

例：東京都の会員の最高年齢を取得

```
SELECT MAX(age) FROM members WHERE address='東京都';
```

- MIN(カラム名)
  - 検索されたデータの最小値を取得する

例：全会員の中の最少年齢を取得

```
SELECT MIN(age) FROM members;
```

集計関数

- SUM(カラム名)
  - 検索されたデータの合計を取得する

例：売上の合計金額を取得

```
SELECT SUM(price) FROM sales;
```

- AVG(カラム名)
  - 検索されたデータの平均値を取得する

例：全会員の平均年齢を取得

```
SELECT AVG(age) FROM members;
```

集計関数

- COUNT(カラム名)
  - 検索されたデータの個数を取得する

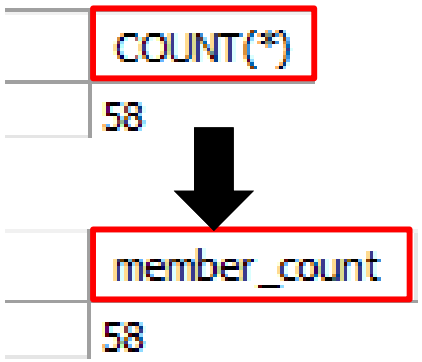
例：未成年の会員の人数を取得

```
SELECT COUNT(*) FROM members WHERE age < 20;
```

「カラム名 **AS** 別名」でカラムに別名を定義することができる

例： COUNT(*) にmember_countという別名をつける

```
SELECT COUNT(*) AS member_count FROM members;
```



カラムごとの集計

- GROUP BY カラム名
  - カラムごとにグループ化して集計を行う

例：年齢ごとの会員数を取得

```
SELECT age,COUNT(*) AS member_count FROM members GROUP BY age;
```

age	member_count
14	3
18	3
23	3
24	2
25	2
26	3
27	2
33	2
34	3
35	5

練習

- 練習04-6
- 練習04-7
- 練習04-8
- 練習04-9

並べ替え

- ORDER BY カラム1 ASC/DESC, カラム2 …

  ⇒ 指定したカラムの値で結果を並べ替える

  - カラム名の後に**ASC**を付けると昇順で並べ替え、**DESC**を付けると降順で並べ替える（省略時は昇順）
  - 複数の基準で並び替えを行う場合、カラム名をカンマで区切って列挙する
  - 日時については、新しい日時が大きい値とみなされる

例：会員を年齢の高い順に並べて取り出す

```
SELECT * FROM members ORDER BY age DESC;
```

並び順用のカラム

- 並び順を管理するカラムを更新する場合のSQLは、以下のように記述することができる

並び順を管理するカラム

id	name	ranking
1	山田太郎	1
2	田中花子	2
3	本田次郎	3
4	鈴木三郎	4
5	菊池葉子	5
6	沢田律子	6

田中さんの順位を5位に変更

id	name	ranking
1	山田太郎	1
2	田中花子	~~2~~ ⇒ 5 ①
3	本田次郎	~~3~~ ⇒ 2 ②
4	鈴木三郎	~~4~~ ⇒ 3 ②
5	菊池葉子	~~5~~ ⇒ 4 ②
6	沢田律子	6

```
UPDATE members SET ranking = 5 WHERE id = 2;                    ①
UPDATE members SET ranking = ranking - 1
       WHERE ranking > 2  AND ranking <= 5 AND id != 2;         ②
```

取り出すデータの範囲を制限する

- LIMIT 開始位置，件数
  - 検索結果の「開始位置」件目から「件数」件だけを取り出す

  ⇒ 新着情報の取得や、ページ分割機能の実装等で利用

  - 先頭から取り出す場合、開始位置は「0」になる

    ※ 先頭からデータを取得する場合、「開始位置,」は省略できる

例：会員の201人目から10人分を取り出す

```
SELECT * FROM members LIMIT 200,10;
```

COLLATEの指定

- データ検索時や並び替えの際にもCOLLATEの指定が可能
  - データベース作成時とは異なるCOLLATE設定で検索や並び替えをすることができる

例：名前がカタカナの「アリス」である会員を検索する

```
SELECT * FROM members
WHERE name COLLATE utf8mb4_0900_as_cs LIKE "%アリス";
```

※ COLLATEが utf8mb4_0900_ai_ci の場合、ひらがなの「ありす」や濁音を含む「ありず」「アリズ」といったデータも取得されてしまう

例: 大文字・小文字やひらがな・カタカナを意識して並び替える

```
SELECT * FROM members
ORDER BY name COLLATE utf8mb4_0900_as_cs;
```

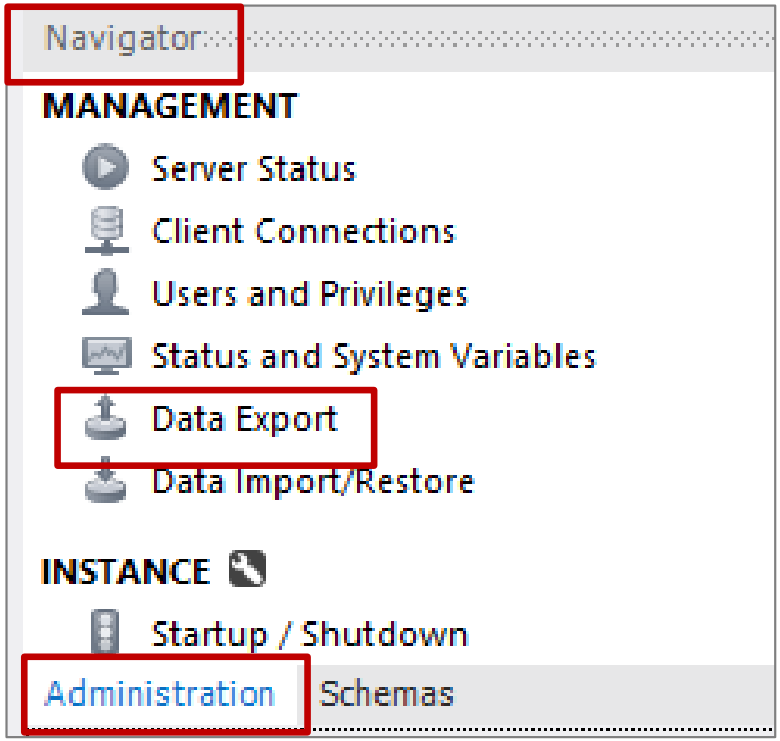
※「ありす」と「アリス」がある場合、ひらがなの方が先になる
※「alice」と「Alice」がある場合、小文字の方が先になる

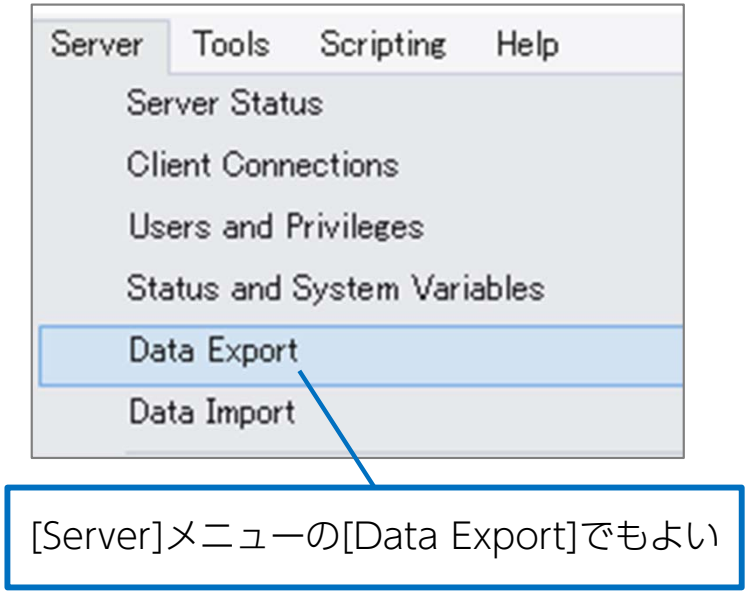
練習

- 練習04-10
- 練習04-11

データベースのエクスポート

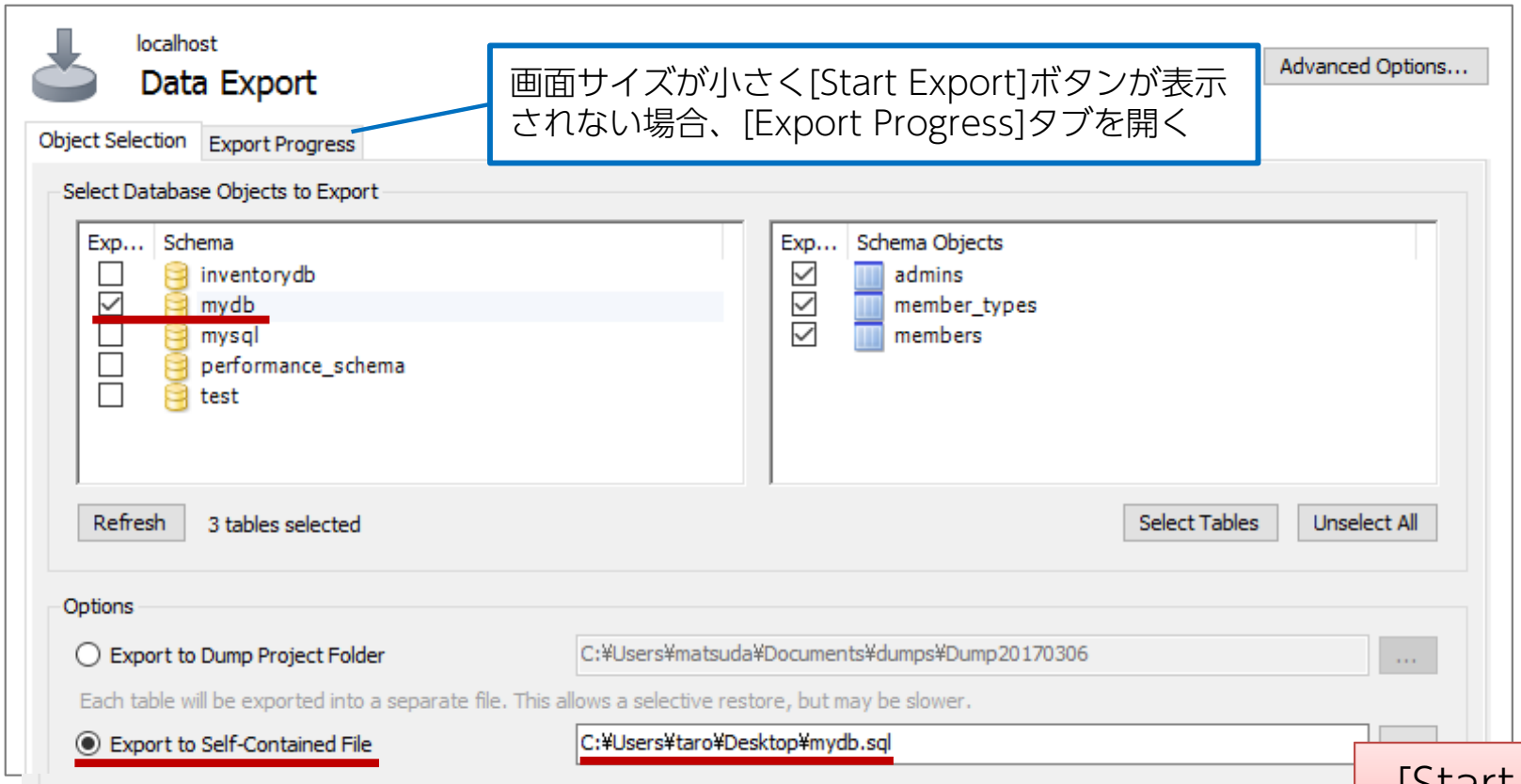
- MySQL Workbenchで、データベースの内容(構造とデータ)をSQLファイルに書き出すこと(エクスポート)ができる
  - Navigatorパネルの[Data Export] をクリック





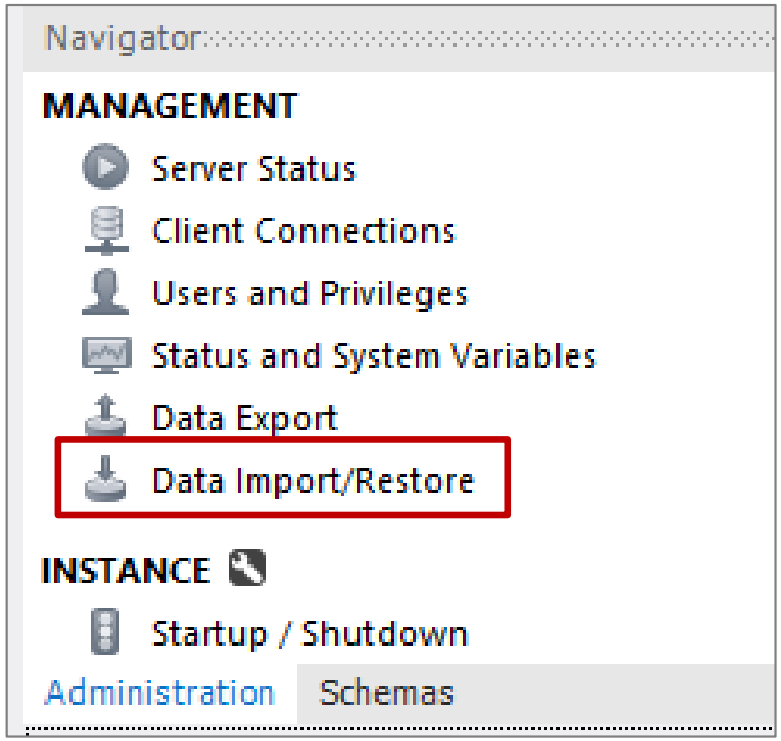
データベースのエクスポート

- エクスポートするデータベースを選び、[Export to Self-Contained File]で保存先のファイル名を指定する



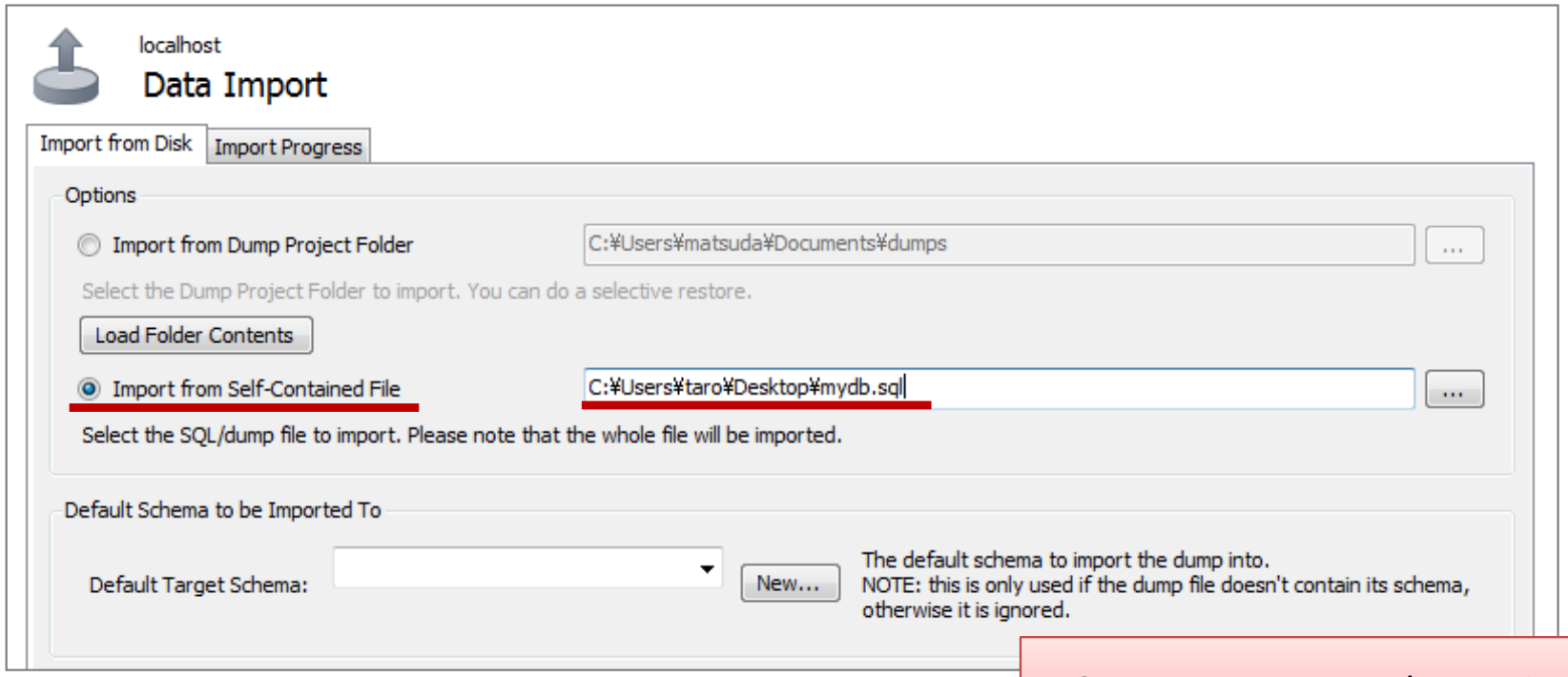
データベースのインポート

- 逆に、ファイルに保存されたSQLスクリプトをデータベースに読み込むこと(インポート)もできる
  - Navigatorパネルの[Data Import/Restore] をクリック



データベースのインポート

- [Import from Self-Contained File]でインポートするSQLファイルを指定する



[Start Import]をクリック