アプリ開発者向け SQL 入門

2024/06/29

SRA OSS LLC 松坂大地

会社紹介

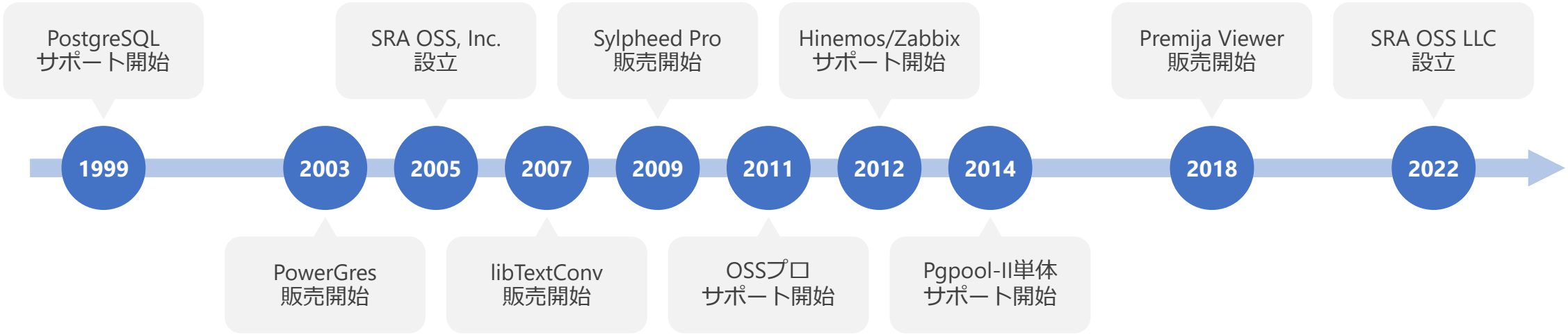
SRA OSS合同会社

所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
出　　資: 株式会社SRA
資 本 金: 7,000万円
社　　長: 稲葉 香理

事業内容

- オープンソースソフトウェア (OSS) 関連のサポート、製品開発・販売、構築・コンサル
- OSSの教育、開発、コミュニティ運営支援
- ソフトウェアの研究開発

顧　　問: 石井 達夫
技術顧問: 増永 良文 (お茶の水女子大学名誉教授)



目次

- アプリ開発中のSQLトラブル
- SQL入門
  - アプリ発行のSQL確認
  - SQL発行
  - SELECT文

アプリ開発中のSQLトラブル

- 目的のデータが得られない

  ⇒ ORM の使い方の問題か？ 発行されているSQLの確認が必要

- 実装してみたけど、画面表示に時間がかかる
  - 集計レポートを表示するような画面を作成したい時や、
    複雑な条件でデータを取得したい時
    ORM と向き合って色々頑張っても、目的の性能が出ない

    ⇒ 生のSQLを実行する、という選択肢を視野に

アプリ開発者でもSQLを知っていて損はない

SRA OSS

アプリ発行のSQL確認編

まず、フレームワークが発行しているSQLを確認します

ログを確認①

```
2024-06-03 17:23:19 debug: connection=default duration=4 rows=1 SELECT 1 AS "existing" FROM users Users WHERE (Users.name_id = XXXX AND Users.is_latest = TRUE) LIMIT 1
2024-06-03 17:23:19 debug: connection=default duration=5 rows=1 INSERT INTO audit_logs (al_us_id, al_us_name_id, al_update, al_path, al_post_data) VALUES (XXXX, XXXX, '2024-06-03 17:23:19.396546', '/questions/XXXX/ /admin-api/current_staff_roles.json', '"[]"') RETURNING *
2024-06-03 17:23:19 debug: connection=default duration=0 rows=0 COMMIT
2024-06-03 17:23:19 debug: connection=default duration=0 rows=0 BEGIN
2024-06-03 17:23:19 debug: connection=default duration=4 rows=1 SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS "Users__password", Users.te_id AS "Users__te_id", Users.cm_id AS
"Users__cm_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS
"Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.us_id AS "Users__us_id", Users.us_name_id AS "Users__us_name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE Users.id = 100666 LIMIT 1
2024-06-03 17:23:19 debug: connection=default duration=4 rows=1 SELECT 1 AS "existing" FROM users Users WHERE (Users.name_id = XXXX AND Users.is_latest = TRUE) LIMIT 1
2024-06-03 17:23:19 debug: connection=default duration=4 rows=1 UPDATE audit_logs SET al_update = '2024-06-03 17:23:19.431670' , al_response = '200 OK' WHERE al_id = XXXX
2024-06-03 17:23:19 debug: connection=default duration=0 rows=0 COMMIT
2024-06-03 17:23:23 debug: connection=default duration=7 rows=1 SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS "Users__password", Users.te_id AS "Users__te_id", Users.cm_id AS
"Users__cm_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS
"Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.us_id AS "Users__us_id", Users.us_name_id AS "Users__us_name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = 848 AND Users.is_latest = TRUE) LIMIT 1
2024-06-03 17:23:23 debug: connection=default duration=0 rows=0 BEGIN
2024-06-03 17:23:23 debug: connection=default duration=6 rows=1 SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS "Users__password", Users.te_id AS "Users__te_id", Users.cm_id AS
"Users__cm_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS
"Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.us_id AS "Users__us_id", Users.us_name_id AS "Users__us_name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE Users.id = 100657 LIMIT 1
2024-06-03 17:23:23 debug: connection=default duration=5 rows=1 SELECT 1 AS "existing" FROM users Users WHERE (Users.name_id = XXXX AND Users.is_latest = TRUE) LIMIT 1
2024-06-03 17:23:23 debug: connection=default duration=5 rows=1 INSERT INTO audit_logs (al_us_id, al_us_name_id, al_update, al_path, al_post_data) VALUES (XXXX, XXXX, '2024-06-03 17:23:23.116022', '/questions/ /admin-api/current_staff_roles.json', '"[]"') RETURNING *
2024-06-03 17:23:23 debug: connection=default duration=0 rows=0 COMMIT
2024-06-03 17:23:23 debug: connection=default duration=0 rows=0 BEGIN
2024-06-03 17:23:23 debug: connection=default duration=4 rows=1 SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS "Users__password", Users.te_id AS "Users__te_id", Users.cm_id AS
"Users__cm_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS
"Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.us_id AS "Users__us_id", Users.us_name_id AS "Users__us_name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE Users.id = 100657 LIMIT 1
2024-06-03 17:23:23 debug: connection=default duration=4 rows=1 SELECT 1 AS "existing" FROM users Users WHERE (Users.name_id = XXXX AND Users.is_latest = TRUE) LIMIT 1
2024-06-03 17:23:23 debug: connection=default duration=9 rows=1 UPDATE audit_logs SET al_update = '2024-06-03 17:23:23.152106' , al_response = '200 OK' WHERE al_id = XXXX
2024-06-03 17:23:23 debug: connection=default duration=0 rows=0 COMMIT
2024-06-03 17:23:35 debug: connection=default duration=7 rows=1 SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS "Users__password", Users.te_id AS "Users__te_id", Users.cm_id AS
"Users__cm_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS
"Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.us_id AS "Users__us_id", Users.us_name_id AS "Users__us_name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = 4021 AND Users.is_latest = TRUE) LIMIT 1
2024-06-03 17:23:35 debug: connection=default duration=0 rows=0 BEGIN
2024-06-03 17:23:35 debug: connection=default duration=4 rows=1 SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS "Users__password", Users.te_id AS "Users__te_id", Users.cm_id AS
"Users__cm_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS
"Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.us_id AS "Users__us_id", Users.us_name_id AS "Users__us_name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE Users.id = 100805 LIMIT 1
```

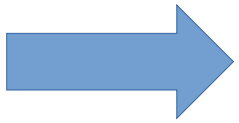
CakePHPの
クエリログの場合
こんな感じ

```
SELECT Users.id AS “Users__id”, Users.name AS
“Users__name”, Users.mail AS “Users__mail”, ...
FROM users WHERE Users.id = 100657 LIMIT 1
```

ログを確認②

- AS – テーブルとカラムの別名
  - users はこのクエリ内では Users と呼ぶことにしている
  - “AS” は省略できる

```
SELECT
  Users.id AS "Users__id",
  Users.name AS "Users__name",
  Users.mail AS "Users__mail"
FROM users Users
WHERE ...
```



別名を取り除くとこうなる

```
SELECT
  users.id,
  users.name,
  users.mail
FROM users
WHERE ...
```

- 取得カラムは、”テーブル名.カラム名” の形で記述
  - CakePHPでは “Users__id” などの形で命名していることがわかる

PostgreSQL のログ ①

- データベース側のログからわかること
  - 実際に実行されたSQL
  - その実行時間（スロークエリの特定）

- データベースクラスタ内に作成されている postgresql.conf にてログ設定を確認

```
# echo $PGDATA
/var/lib/postgresql/data

# ls $PGDATA/postgresql.conf
/var/lib/postgresql/data/postgresql.conf
```

PostgreSQL のログ ②

- ここではデータベースクラスタ内に保存する方法※ を紹介
  - logging_collector を有効にする

```
# - Where to Log -

#log_destination = 'stderr'

# This is used when logging to stderr:
#logging_collector = off
logging_collector = on

# These are only used if logging_collector is on:
#log_directory = 'log'
#log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
```

※ (Linux) syslog や (Windows) eventlog に送る方法もあります

PostgreSQL のログ ③

- SQL文のログ出力

  log_min_duration_statement

  指定した時間(ミリ秒)以上かかったクエリをログに記録

  - “0” と指定した場合、すべてのクエリが出力される
  - ※スロークエリを探す場合、大きめの値から始めるとよい
    - ms 単位で指定する（単位を書いてしまってもよい）

```
# - When to Log -

#log_min_duration_statement = -1   # -1 is disabled, 0 logs all statements
log_min_duration_statement = 0
#log_min_duration_statement = 1000
#log_min_duration_statement = 1000ms
#log_min_duration_statement = 1s
```

PostgreSQL のログ ④

- 設定変更の反映

```
$ pg_ctl restart -D /var/lib/postgresql/data/
```

- 設定ファイルに再起動が必要と記載がなければ、再読み込みでもOK

```
logging_collector = on                  # (change requires restart)
```

```
$ pg_ctl reload -D /var/lib/postgresql/data/
```

PostgreSQL のログ ⑤

```
2024-05-14 05:46:09.130 UTC [506] LOG:  duration: 2.669 ms  bind pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password
AS "Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS
"Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled",
Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.130 UTC [506] DETAIL:  parameters: $1 = '1166', $2 = 't'
2024-05-14 05:46:09.130 UTC [506] LOG:  execute pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS
"Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges",
Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS
"Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS "Users__mfa_secret_key" FROM users
Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.130 UTC [506] DETAIL:  parameters: $1 = '1166', $2 = 't'
2024-05-14 05:46:09.130 UTC [507] LOG:  duration: 1.623 ms  parse pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name",
Users.password AS "Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS
"Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled",
Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.130 UTC [506] LOG:  duration: 0.104 ms
2024-05-14 05:46:09.130 UTC [508] LOG:  duration: 1.885 ms  parse pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name",
Users.password AS "Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS
"Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled",
Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.132 UTC [508] LOG:  duration: 1.750 ms  bind pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password
AS "Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS
"Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled",
Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.132 UTC [508] DETAIL:  parameters: $1 = '1166', $2 = 't'
2024-05-14 05:46:09.132 UTC [508] LOG:  execute pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS
"Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges",
Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS
"Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS "Users__mfa_secret_key" FROM users
Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.132 UTC [508] DETAIL:  parameters: $1 = '1166', $2 = 't'
2024-05-14 05:46:09.132 UTC [508] LOG:  duration: 0.107 ms
2024-05-14 05:46:09.133 UTC [507] LOG:  duration: 2.423 ms  bind pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password
AS "Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS
"Users__privileges", Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled",
Users.update AS "Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS
"Users__mfa_secret_key" FROM users Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.133 UTC [507] DETAIL:  parameters: $1 = '1166', $2 = 't'
2024-05-14 05:46:09.133 UTC [507] LOG:  execute pdo_stmt_00000001: SELECT Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mail AS "Users__mail", Users.name AS "Users__name", Users.password AS
"Users__password", Users.te_table_id AS "Users__te_table_id", Users.cm_table_id AS "Users__cm_table_id", Users.cm_name_id AS "Users__cm_name_id", Users.slack_id AS "Users__slack_id", Users.privileges AS "Users__privileges",
Users.lang_enum AS "Users__lang_enum", Users.settings_json AS "Users__settings_json", Users.news_enum AS "Users__news_enum", Users.note AS "Users__note", Users.is_enabled AS "Users__is_enabled", Users.update AS
"Users__update", Users.is_latest AS "Users__is_latest", Users.id AS "Users__id", Users.name_id AS "Users__name_id", Users.mfa_is_enabled AS "Users__mfa_is_enabled", Users.mfa_secret_key AS "Users__mfa_secret_key" FROM users
Users WHERE (Users.name_id = $1 AND Users.is_latest = $2) LIMIT 1
2024-05-14 05:46:09.133 UTC [507] DETAIL:  parameters: $1 = '1166', $2 = 't'
```

PostgreSQL のログ ⑥

- 拡張問い合わせ について
  - それぞれの段階で要した時間が記録されます

    Parse： Prepared Statement (準備された文)を割り当てます
    Bind： パラメータの値を渡す
    Execute： 実行

```
LOG:  duration: 0.893 ms parse pdo_stmt_00000001: SELECT Users.id AS "Users__id", users.name …
LOG:  duration: 1.261 ms bind pdo_stmt_00000001: SELECT Users.id AS "Users__id", users.name …
DETAIL:  parameters: $1 = '1166', $2 = 't'
LOG:  duration: 1.070 ms execute pdo_stmt_00000001: SELECT Users.id AS "Users__id", users.name …
DETAIL:  parameters: $1 = '1166', $2 = 't'
LOG:  duration: 0.045 ms  statement: DEALLOCATE pdo_stmt_00000001
```

    - 知りたいのは Execute の実行時間
      ※ log_min_duration_statement が大きめに設定されていれば
      大抵は Execute しかログ出力されないはず

  - DEALLOCATE 文： Prepared Statement の割り当てを解除するために発行されている

拡張問い合わせは SQLインジェクションの 対策にもなっている

PostgreSQL のログ ⑦

- ログからクエリを拾ってきて実行する場合、
  拡張問い合わせを簡易問い合わせに書き換える必要があります

```
LOG:  execute pdo_stmt_00000001:
  SELECT
    Users.id AS "Users__id",
    Users.name AS "Users__name",
    Users.mail AS "Users__mail",
    (省略)
  FROM users Users
  WHERE (Users.mail = $1 AND Users.is_enabled = $2)
DETAIL:  parameters: $1 = 'sraoss.co.jp', $2 = 't'
```

```
SELECT
  Users.id AS "Users__id",
  Users.name AS "Users__name",
  Users.mail AS "Users__mail",
  (省略)
FROM users Users
WHERE (Users.mail = 'sraoss.co.jp' AND Users.is_enabled = 't')
```

- PREPARE 文を使う方法も取れます

```
PREPARE test_001(text, bool) AS
  SELECT
    Users.id AS "Users__id",
    Users.name AS "Users__name",
    Users.mail AS "Users__mail",
    (省略)
  FROM users Users
  WHERE (Users.mail = $1 AND Users.is_enabled = $2);
EXECUTE test_001('sraoss.co.jp', 't');
```

```
PREPARE 名前 [ ( データ型 [, …] ) ] AS クエリ
```

```
EXECUTE 名前 [ ( 値 [, …] ) ]
```

SQL発行編

発行されている SQLの確認ができたら、直接実行してみます
SQL の発行には psql コマンドをおすすめします

実行履歴がテキストの形で
残るため報告しやすい

SQLの発行①

- psql ＝ SQL発行ツール
- psql の起動方法

```
psql [ オプション… ] [ DB名 [ ユーザ名 ] ]
```

  - DB名: 省略すると、環境変数の値、またはユーザ名と同じ値を使う
  - ユーザ名: 省略すると、環境変数の値、またはOSユーザと同じ値を使う

```
$ psql prjname prjname
psql (14.1)
Type "help" for help.

prjname=>
```

PostgreSQL接続しているデータベース名

```
$ psql -d prjname -U prjname
```

-d，-U と明示的にオプション指定してもOK

SQLの発行②

- psql オプション

```
psql [ オプション… ] [ DB名 [ ユーザ名 ] ]
```

  - **-h** : ホスト名（省略すると localhost へ接続しようとする）
  - **-l** : データベース一覧を表示

```
$ psql -h XXXX.XXXXXXXXX.XXX -U prjname -l
 データベース一覧
          名前          |  所有者   | エンコーディング  |
------------------------+----------+-------------------+---
 postgres               | postgres | UTF8              |
 prjname                | postgres | UTF8              |
 template0              | postgres | UTF8              |
 template1              | postgres | UTF8              |
(4 行)
```

(snip)

SQLの発行③

- psql オプション（続き）
  - **-c**：SQL文を指定して実行

```
$ psql -c “SELECT * FROM users LIMIT 1"
  id  |    name    | mail              | is_enabled |
------+------------+-------------------+------------+-
 1000 | 札幌　太郎 | taro@example.com | t          |
(1 行)
```

(snip)

  - **-f**：SQL文を記述したファイルを指定して実行
  - **-o**：すべての問い合わせの出力を指定したファイルへ書き込む

```
$ psql -f 001.sql -f 002.sql -o output.txt
$ ls
001.sql  002.sql  output.txt
```

SQLの発行④

- バックスラッシュコマンド(メタコマンド)
  - **¥dt** ：データベース内の
    テーブル一覧を確認

```
$ psql prjname
prjname=> ¥dt
                    リレーション一覧
 スキーマ |          名前          |  タイプ  | 所有者
---------+------------------------+---------+---------
 public  | companies              | テーブル | prjname
 public  | users                  | テーブル | prjname
 (2 行)
```

  - **¥d** テーブル名：テーブルスキーマ情報を確認

```
prjname=> ¥d users
 テーブル"public.users"
     列      |          タイプ          | 照合順序 | Null 値を許容 |                  デフォルト
-------------+--------------------------+---------+--------------+-----------------------------------
 id          | integer                  |         | not null     | nextval('users_id_seq'::regclass)
 name        | text                     |         | not null     |
 mail        | text                     |         | not null     |
 :
(snip)
```

SQLの発行⑤

- バックスラッシュコマンド(メタコマンド)（続き）
  - ¥x：拡張テーブル形式の表示モード on/off を切り替えます
    - 例）

```
prjname=> SELECT * FROM users LIMIT 2;
 id   |    name    |       mail       | is_enabled | team_id | company_id | slack_id    |        privileges
| language |         settings_json         | news_enum |          updated           | mfa_is_enabled |
mfa_secret_key
------+------------+------------------+------------+---------+------------+-------------+--------------------
+----------+-------------------------------+-----------+----------------------------+----------------+-------
-----------+----------------
 1000 | 札幌　太郎  | taro@example.com | t          | 23      | 55         | XXXXXXXXXX | 10000011100011100000
00001010 | 1        | {"send_news_by_mail": “true"} | 1         | 2024-04-01 10:19:32.579579 | 1          | XX
XXXXXXXXXXXXXXXXXXXXXXXXXX
 1001 | 札幌　次郎  | ziro@example.com | t          | 23      | 55         | XXXXXXXXXX | 10000011100011100000
00001010 | 1        | {"send_news_by_mail": “true"} | 1         | 2024-04-01 10:23:33.445234 | 1          | XX
XXXXXXXXXXXXXXXXXXXXXXXXXX
(2 行)

prjname=> ¥x
prjname=> SELECT * FROM users LIMIT 2;
-[ RECORD 1 ]--------------------------
id         | 1000
name       | 札幌　太郎
mail       | taro@example.com
is_enabled | t
team_id    | 23
company_id | 55
 :
(snip)
```

SQLの発行⑥

- バックスラッシュコマンド(メタコマンド) (続き)
  - /timing : 各SQL文にかかった時間の表示機能 on/off を切り替えます

```
prjname=> ¥timing
タイミングは on です。
prjname=> SELECT * FROM users LIMIT 2;
-[ RECORD 1 ]--------------------------
id         | 1000
name       | 札幌　太郎
mail       | taro@example.com
is_enabled | t
team_id    | 23
company_id | 55
 :
(snip)
 :
時間: 2.108 ミリ秒
```

SQL文解説編

SQLの発行ができたら、
今度は、自身でSQLを解読または作成していきます

SELECT文①

- SELECT - テーブルまたはビューから行を検索する

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

```
=> SELECT name, price FROM items WHERE price < 3000;
                          name                          | price
--------------------------------------------------------+-------
 SQLクイックリファレンス                                |  2800
 SQLポケットリファレンス                                |  1980
 Database-SQL-Rdbms Howto                               |  2634
  :
 (省略)
```

SELECT文②

- ORDER BY - 出力を並び替える

```
SELECT カラム名 FROM テーブル名 WHERE 検索条件 ORDER BY カラム名 [ ASC | DESC ]
```

  - ASC：昇順（デフォルト）
  - DESC：降順

```
=> SELECT name, price FROM items ORDER BY price ASC;
                          name                          | price
--------------------------------------------------------+-------
 SQLポケットリファレンス                                 |  1980
 Database-SQL-Rdbms Howto                               |  2634
 SQLクイックリファレンス                                 |  2800
   :
 (省略)
```

```
=> SELECT name, price FROM items ORDER BY price DESC;
                          name                          | price
--------------------------------------------------------+-------
                                                        |  4980
                                                        |  4640
                                                        |  4540
                                                          :
                                                       (省略)
```

SELECT文③

- LIMIT, OFFSET - **検索結果を範囲指定する**

```
SELECT … WHERE 検索条件 ORDER BY カラム名 LIMIT 件数 OFFSET 開始行;
```

  - 検索結果のうち「開始行から件数だけ」を返す
  - 開始（OFFSET）は 0 から始まる
  - LIMIT と OFFSET は SQL 標準ではない
  - ORDER BY と組み合わせて使う

```
=> SELECT name, price FROM items ORDER BY price ASC
-> LIMIT 2 OFFSET 0;
                          name                          | price
--------------------------------------------------------+-------
 SQLポケットリファレンス                                 |  1980
 Database-SQL-Rdbms Howto                               |  2634
(2 行)
```

SELECT文④

- DISTINCT -検索結果から指定した列の重複を取り除く

```
SELECT DISTINCT [ ON (カラム名) ] カラム名 FROM …
```

  - 例：価格と著者の組み合わせ一覧を表示

```
=> SELECT id, price, description FROM items ORDER BY price ASC;
 id  | price |                          description
-----+-------+---------------------------------------------------------------
          -- 省略 --
  11 |  6800 | ソフトバンクパブリッシング  ジム グレイ／アンドレアス ロイター 共著
  12 |  6800 | ソフトバンクパブリッシング  ジム グレイ／アンドレアス ロイター 共著
          -- 省略 -
(15 rows)

=> SELECT DISTINCT ON (price, description) id, price, description FROM items ORDER BY price ASC;
          -- 省略 --
  11 |  6800 | ソフトバンクパブリッシング  ジム グレイ／アンドレアス ロイター 共著
          -- 省略 -
(14 rows)
```

SELECT文⑤

- GROUP BY, HAVING - 指定した列でグループ単位に分類・集計

```
SELECT … GROUP BY カラム名 HAVING 検索条件;
```

  - 例：売れた書籍の合計を求め、そこから10冊以上売れた商品を抽出

```
=> SELECT item_id, sum(number) FROM sales GROUP BY item_id;
 item_id | sum
---------+-----
      15 |  19
      14 |   4
   (以下略)

=> SELECT item_id, sum(number) FROM sales GROUP BY item_id
-> HAVING sum(number) >= 10 ORDER BY sum(number) DESC;
 item_id | sum
---------+-----
       1 |  64
       2 |  54
       6 |  34
   (以下略)
```

SELECT文⑥

- CASE - SQL文の中で条件を記述できる
  - 例：購入歴で、計10冊以上売れている書籍には「GOOD」
    それ未満は「BAD」という値で取得する

```
=> SELECT item_id,
->   CASE WHEN sum(number) >= 15 THEN 'GOOD'
->        ELSE 'BAD'
->   END AS score
-> FROM sales GROUP BY item_id;
 item_id | score
---------+-------
      15 |  GOOD
      14 |  BAD
 (以下略)
```

```
CASE
    WHEN 条件式1 THEN 結果1
    WHEN 条件式2 THEN 結果2
     . . .
    ELSE 結果3
END
```

結合①

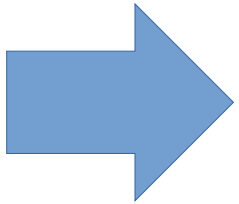
- JOIN 構文
  - 複数テーブルを結合し、ひとつの表にした結果を返す
  - 結合する方法によっていくつか種類がある
- CROSS JOIN : テーブル同士の直積を返す

```
SELECT … FROM テーブル名1 JOIN テーブル名2;
```

```
=> SELECT * FROM posts;
 id | text | user_id
----+------+---------
  1 | aaaa |       1
  2 | bbbb |       2
  3 | cccc |       2
  4 | dddd |
(4 行)
```

×

```
=> SELECT * FROM users;
 id | name
----+------
  1 | taro
  2 | jiro
(2 行)
```



```
=> SELECT * FROM posts CROSS JOIN users;
 id | text | user_id | id | name
----+------+---------+----+------
  1 | aaaa |       1 |  1 | taro
  2 | bbbb |       2 |  1 | taro
  3 | cccc |       2 |  1 | taro
  4 | dddd |         |  1 | taro
  1 | aaaa |       1 |  2 | jiro
  2 | bbbb |       2 |  2 | jiro
  3 | cccc |       2 |  2 | jiro
  4 | dddd |         |  2 | jiro
(8 行)
```

結合②

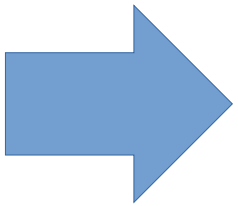
- INNER JOIN (内部結合): 直積結果から結合条件に一致した行を返す

```
SELECT … FROM テーブル名1 [ INNER ] JOIN テーブル名2 [ ON 結合条件 ];
```

```
=> SELECT * FROM posts;
 id | text | user_id
----+------+---------
  1 | aaaa |       1
  2 | bbbb |       2
  3 | cccc |       2
  4 | dddd |
(4 行)
```

×

```
=> SELECT * FROM users;
 id | name
----+------
  1 | taro
  2 | jiro
(2 行)
```



```
=> SELECT * FROM posts
->   INNER JOIN users ON (posts.user_id = users.id);
 id | text | user_id | id | name
----+------+---------+----+------
  1 | aaaa |       1 |  1 | taro
  2 | bbbb |       2 |  2 | jiro
  3 | cccc |       2 |  2 | jiro
(3 行)
```

結合③

- OUTER JOIN (外部結合): INNER JOIN の結果にプラスして
  指定したテーブルの行はすべて返す

```
SELECT … FROM テーブル名1
{ LEFT | RIGHT | FULL } [ OUTER ] JOIN テーブル名2 [ ON 結合条件 ];
```

```
=> SELECT * FROM posts;
 id | text | user_id
----+------+---------
  1 | aaaa |       1
  2 | bbbb |       2
  3 | cccc |       2
  4 | dddd |
(4 行)
```

×

```
=> SELECT * FROM users;
 id | name
----+------
  1 | taro
  2 | jiro
(2 行)
```

```
=> SELECT * FROM posts
->   LEFT JOIN users ON (posts.user_id = users.id);
 id | text | user_id | id | name
----+------+---------+----+------
  1 | aaaa |       1 |  1 | taro
  2 | bbbb |       2 |  2 | jiro
  3 | cccc |       2 |  2 | jiro
  4 | dddd |         |    |
(4 行)
```

サブクエリ①

```
users
 id | name
----+------
  1 | taro
  2 | jiro
```

```
posts
 id | text | user_id
----+------+---------
  1 | aaaa |       1
  2 | bbbb |       2
  3 | cccc |       2
  4 | dddd |
```

- 選択リスト部分で使うケース
  - サブクエリが返す結果は厳密に1要素でなければならない

```
SELECT (サブクエリ) FROM テーブル名;
```

```
=> SELECT
->   id,
->   text,
->   (SELECT name FROM users WHERE id = 1) AS admin
-> FROM posts;
 id | text | admin
----+------+-------
  1 | aaaa | taro
  2 | bbbb | taro
  3 | cccc | taro
  4 | dddd | taro
(4 行)
```

サブクエリ②

```
users
 id | name
----+------
  1 | taro
  2 | jiro
```

```
posts
 id | text | user_id
----+------+---------
  1 | aaaa |       1
  2 | bbbb |       2
  3 | cccc |       2
  4 | dddd |
```

- FROM句で使うケース
  - サブクエリが返す結果を一時的にテーブルとして扱う

```
=> SELECT
     users.id,
     users.name,
     counts.post_count
   FROM users
   INNER JOIN (
     SELECT user_id, count(*) AS post_count FROM posts GROUP BY user_id
   ) AS counts
   ON (users.id = counts.user_id);
 id | name | post_count
----+------+------------
  1 | taro |          1
  2 | jiro |          2
(2 行)
```

サブクエリ③

```
users
 id | name
----+------
  1 | taro
  2 | jiro
```

```
posts
 id | text | user_id
----+------+---------
  1 | aaaa |       1
  2 | bbbb |       2
  3 | cccc |       2
  4 | dddd |
```

- WITH句 （前ページの書き換え）
  - サブクエリを先に定義しておき、複数個所で利用できる

```
=> WITH counts AS (
     SELECT user_id, count(*) AS post_count FROM posts GROUP BY user_id
   )
   SELECT
     users.id,
     users.name,
     counts.post_count
   FROM users INNER JOIN counts ON (users.id = counts.user_id);
 id | name | user_id | post_count
----+------+---------+------------
  1 | taro |       1 |          1
  2 | jiro |       2 |          2
(2 行)
```

サブクエリ④

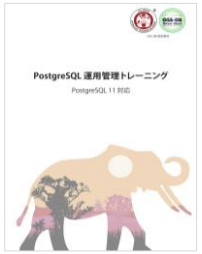
- WITH句（続き）
  - カンマ区切りにて複数の
    サブクエリを定義できます
  - 右の例では、必要な情報をサブクエリの
    形で揃えていき、最終的に results の
    結合で結果一覧を生成しています
  - 最後の 主問い合わせ（黄色）のみを
    切り替えることができるため、
    Web開発と相性が良い
    - 途中段階のサブクエリ毎に
      Unit Test が書けます

```
=> WITH list1 AS (
     SELECT ... FROM table1 WHERE ...
   ),
   list2 AS (
     SELECT ... FROM list1 WHERE ...
   ),
     :
   (省略)
     :
   results AS (
     SELECT ... FROM list2
       LEFT JOIN list3 ON ...
       LEFT JOIN list4 ON ...
   )
   SELECT * FROM results ORDER BY ...
   LIMIT XXX OFFSET XXX;
```

```
SELECT count(*) FROM results;
```

```
SELECT * FROM list1;
```

- 取り扱いトレーニング



PostgreSQL トレーニング

PostgreSQL を初めて導入される方から、内部構造を解説するものまで、幅広いコースを用意しています。

- 技術者 募集中

ホームページをご確認ください
https://www.sraoss.co.jp/personnel/

- ご清聴ありがとうございました

TEL：03-5979-2701
E-mail: sales@sraoss.co.jp
https://www.sraoss.co.jp/