MySQL 8.0 の薄い本

hmatsu47　著

2019-05-02 MySQL 8.0.16 版　　発行

はじめに

本書の目的

「**MySQL 5.7 より最大 2 倍高速**」と Oracle がアナウンスしている MySQL 8.0 を取り上げた本です[*1]。

2016 年 9 月に MySQL 8.0.0 がはじめてリリースされ、2018 年 4 月リリースの MySQL 8.0.11 から GA[*2]となり、MySQL 8.0 も徐々にプロダクトへの採用事例が増えてきました。その間、公式リファレンスマニュアル[*3]・MySQL Server Team による MySQL Server Blog[*4]のほか MySQL パートナーや個人のブログに MySQL 8.0 の新機能を紹介する記事が多数掲載されており、今もその数を増やしています。

この「**MySQL 8.0 の薄い本**」では、MySQL 8.0 で導入された新機能をページ数の制約（および著者の能力）の範囲でできるだけ取り上げるとともに、紹介記事の URL を提供します[*5]。

想定読者

MySQL 5.7 までのバージョンの利用経験があり、MySQL 8.0 の新機能に興味がある方です。なお、この本では従来の MySQL について丁寧な説明は行いません。はじめて MySQL に触れる方は、まず MySQL 5.7 までの入門書・解説書などを読んで MySQL を実際に起動・操作し、全体像を掴んでおくことをお勧めします。

ライセンスについて

[*1] 性能・パフォーマンスについて知るには、MySQL 界隈で「ベンチマークおじさん」として有名？ な Dimitri さんのブログや資料がお勧めです。「日本の Dimitri（おじ）さん」こと@i_rethi さんによるこちらの解説記事をご確認ください。http://hiroi10.hatenablog.com/entry/2018/12/24/000138

[*2] General Availability

[*3] https://dev.mysql.com/doc/refman/8.0/en/

[*4] https://mysqlserverteam.com 一部日本語記事あり。また、Yakst | 人力翻訳コミュニティ https://yakst.com/ja に日本語訳されている記事もあります。

[*5] URL を入力するのは面倒なので、各章末に関連リンク集への QR コードを掲載します。

[*6] 情報の出所がわからなくなることを避けるため

商標について

- Oracle と Java、JavaScript、JDK および MySQL は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります[*7]。
- その他記載の会社名、製品名等は、それぞれの会社・組織の商標もしくは登録商標です。

その他免責事項、制限事項等

- 本書記載の内容は無保証です。本書の利用により生じた一切の損害等を著者は負わないものとします。
- 本書記載の内容は著者個人の調査等によるものであり、所属する組織とは無関係です。
- 本書の内容は 2019 年 4 月現在の情報をもとに構成しています。
  - MySQL 8.0 は Continuous Delivery Model（継続提供モデル）を採用しており、マイナーバージョンが上がるごとに機能が追加されていくことが想定されています。
  - 本書で紹介する機能は途中のマイナーバージョンで追加・変更されたものを含みますが、煩雑になるため追加・変更されたマイナーバージョンは原則として記載しません。
- 本書の内容に誤りや記載 URL のリンク切れ、不適切な URL 等が見つかった場合は、こちらにご連絡ください。
  - E-Mail： hmatsu47@gmail.com
  - Twitter： @hmatsu47

謝辞

本書のレビューを快く引き受けてくださった@taka_yuki_04 さん、また執筆中に進捗を見守ってくださった MySQL ユーザ会界隈[*8]やその他の皆様、ありがとうございました。

リンク集 URL

図 1: https://hmatsu47.hatenablog.com/book_mysql80_001

[*7] https://www.oracle.com/jp/legal/trademarks.html
[*8] 若い方もいらっしゃるので、「MySQL おじさん」の括りではありません。

目次

目次

第 1 章

MySQL 8.0 のインストールと設定パラメータ

1.1 新規インストール

新規インストールについては公式リファレンスマニュアルに手順が記載されており、基本的には MySQL 5.7 とほぼ同じです。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/installing.html

実行例

参考として、CentOS 7 に新規インストールする場合の例を示しておきます。

```
[root@mysql80cent ~]# wget http://dev.mysql.com/get/mysql80-community-release-el7-2.noarch.rpm
※ダウンロードするバージョン（el7-2）はその時点のものを指定
--2019-04-30 09:34:53--  http://dev.mysql.com/get/mysql80-community-release-el7-2.noarch.rpm
（中略）
HTTP request sent, awaiting response... 200 OK
Length: 25892 (25K) [application/x-redhat-package-manager]
Saving to: ‘mysql80-community-release-el7-2.noarch.rpm’

100%[======================================>] 25,892      --.-K/s   in 0.001s

2019-04-30 09:34:54 (28.8 MB/s) - ‘mysql80-community-release-el7-2.noarch.rpm’
saved [25892/25
892]

[root@mysql80cent ~]# yum localinstall mysql80-community-release-el7-2.noarch.rpm
Loaded plugins: fastestmirror
（中略）
Dependencies Resolved

================================================================================
 Package             Arch   Version
                                  Repository                                Size
================================================================================
```

```
Installing:
 mysql80-community-release
                     noarch el7-2 /mysql80-community-release-el7-2.noarch  31 k

Transaction Summary
================================================================================
Install  1 Package

Total size: 31 k
Installed size: 31 k
Is this ok [y/d/N]: y
Downloading packages:
（中略）
Installed:
  mysql80-community-release.noarch 0:el7-2

Complete!
[root@mysql80cent ~]# yum update mysql-community-libs
※ mariadb-libs を mysql-community-libs で置き換える
Loaded plugins: fastestmirror
（中略）
Dependencies Resolved

================================================================================
 Package                      Arch    Version          Repository          Size
================================================================================
Installing:
 mysql-community-libs         x86_64  8.0.16-1.el7     mysql80-community  3.0 M
     replacing  mariadb-libs.x86_64 1:5.5.60-1.el7_5
 mysql-community-libs-compat  x86_64  8.0.16-1.el7     mysql80-community  2.1 M
     replacing  mariadb-libs.x86_64 1:5.5.60-1.el7_5
Installing for dependencies:
 mysql-community-common       x86_64  8.0.16-1.el7     mysql80-community  575 k

Transaction Summary
================================================================================
Install  2 Packages (+1 Dependent package)

Total download size: 5.6 M
Is this ok [y/d/N]: y
Downloading packages:
warning: /var/cache/yum/x86_64/7/mysql80-community/packages/mysql-community-common-8.0.16-1.el
7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Public key for mysql-community-common-8.0.16-1.el7.x86_64.rpm is not installed
(1/3): mysql-community-common-8.0.16-1.el7.x86_64.rpm      | 575 kB   00:00
(2/3): mysql-community-libs-compat-8.0.16-1.el7.x86_64.rpm | 2.1 MB   00:00
(3/3): mysql-community-libs-8.0.16-1.el7.x86_64.rpm        | 3.0 MB   00:00
--------------------------------------------------------------------------------
Total                                              43 MB/s | 5.6 MB  00:00
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
Importing GPG key 0x5072E1F5:
 Userid     : "MySQL Release Engineering <mysql-build@oss.oracle.com>"
 Fingerprint: a4a9 4068 76fc bd3c 4567 70c8 8c71 8d3b 5072 e1f5
 Package    : mysql80-community-release-el7-2.noarch (installed)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
Is this ok [y/N]: y
Running transaction check
```

```
（中略）
Installed:
  mysql-community-libs.x86_64 0:8.0.16-1.el7
  mysql-community-libs-compat.x86_64 0:8.0.16-1.el7

Dependency Installed:
  mysql-community-common.x86_64 0:8.0.16-1.el7

Replaced:
  mariadb-libs.x86_64 1:5.5.60-1.el7_5

Complete!
[root@mysql80cent ~]# yum install mysql-community-client
※クライアントをインストールする
Loaded plugins: fastestmirror
（中略）
Dependencies Resolved

================================================================================
 Package                    Arch      Version         Repository            Size
================================================================================
Installing:
 mysql-community-client     x86_64    8.0.16-1.el7    mysql80-community     32 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 32 M
Installed size: 143 M
Is this ok [y/d/N]: y
Downloading packages:
（中略）
Installed:
  mysql-community-client.x86_64 0:8.0.16-1.el7

Complete!
[root@mysql80cent ~]# yum install mysql-community-server
※サーバをインストールする
Loaded plugins: fastestmirror
（中略）
Dependencies Resolved

================================================================================
 Package                   Arch     Version            Repository           Size
================================================================================
Installing:
 mysql-community-server    x86_64   8.0.16-1.el7       mysql80-community   403 M
Installing for dependencies:
 libaio                    x86_64   0.3.109-13.el7     base                 24 k

Transaction Summary
================================================================================
Install  1 Package (+1 Dependent package)

Total download size: 403 M
Installed size: 1.8 G
```

```
Is this ok [y/d/N]: y
Downloading packages:
（中略）
Installed:
  mysql-community-server.x86_64 0:8.0.16-1.el7

Dependency Installed:
  libaio.x86_64 0:0.3.109-13.el7

Complete!
[root@mysql80cent ~]# systemctl enable mysqld.service
※自動起動 ON
[root@mysql80cent ~]# systemctl start mysqld.service
※起動
[root@mysql80cent ~]# ps aux | fgrep mysqld
mysql     3617  7.2  4.8 1831836 387416 ?      Ssl  09:40   0:00 /usr/sbin/mysqld
root      3662  0.0  0.0 112708   896 pts/0    S+   09:40   0:00 grep -F --color=auto mysqld
[root@mysql80cent ~]# fgrep assword /var/log/mysqld.log
※ログからサーバ初期パスワードを確認
2019-04-30T01:40:30.118228Z 5 [Note] [MY-010454] [Server] A temporary password is generated
for root@localhost: JaN::_YS%83f
```

mysql_secure_installation も使えます。

- https://dev.mysql.com/doc/refman/8.0/en/mysql-secure-installation.html

```
[root@mysql80cent ~]# mysql_secure_installation

Securing the MySQL server deployment.

Enter password for user root:先ほど確認したサーバ初期パスワードを入力

The existing password for the user account root has expired. Please set a new password.

New password:新しいパスワードを入力

Re-enter new password:同じパスワードを入力
The 'validate_password' component is installed on the server.
The subsequent steps will run with the existing configuration
of the component.
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : n

 ... skipping.
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.
```

```
Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

1.1.1 Dedicated Server Mode

MySQL 5.7 までは、リソースが乏しいサーバ環境でも動作するよう各種バッファ容量のデフォルト設定は小さめでした。MySQL 8.0 では、MySQL 専用サーバとして設定する場合 Dedicated Server Mode によって、以下の項目の自動設定を行うことが可能です。

- `innodb_buffer_pool_size`
- `innodb_log_file_size`
- `innodb_log_files_in_group`
- `innodb_flush_method`

起動オプションとして`--innodb-dedicated-server=ON` を付けてサーバを起動することで自動設定されます。具体的な設定値は公式リファレンスマニュアル（以下の 1 つ目の URL）に記載されています。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/innodb-dedicated-server.html
- https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html#sysvar_innodb_dedicated_server

ブログ記事等

- https://yakst.com/ja/posts/4781
- https://www.s-style.co.jp/blog/2018/08/2281/

1.2 アップグレードインストール

アップグレードインストールする方法としては、

- インプレースアップグレードする方法
- 新環境を別途用意し、旧環境で `mysqldump` した内容をリストアする方法

の 2 つがあります。

- https://speakerdeck.com/yoshiakiyamasaki/20181201-mysqlbaziyonatupufalseji-chu-zhi-shi?slide=26

1.2.1 インプレースアップグレード

こちらの資料の 8〜17 ページを参照してください（要 Oracle シングル・サインオンアカウント[*1]）。

- https://www.mysql.com/jp/why-mysql/presentations/mysql-80-upgrade-checker-201811-jp/

シンプルなケースにおけるインプレースアップグレードの流れ

- MySQL 5.7 系列の最新バージョンまでアップグレードする
- MySQL Shell 8.0 をインストールして Upgrade Checker（後述）を実行し、問題点を抽出する
- 問題となる設定やアプリケーションを修正する
  - 設定のうち、MySQL 8.0 で改名されたパラメータ等については `--loose` 接頭辞を付けると良い
- バックアップを取得する
- MySQL Server 8.0 を上書きインストールして起動する
- 必要に応じてユーザと権限設定等を修正する
  - アプリケーションで使用するユーザを新規に作り直す場合、第 2 章で説明する認証プラグインの指定に注意する

【注】MySQL 8.0.16 から `mysql_upgrade` が不要になりました。

- https://dev.mysql.com/doc/refman/8.0/en/upgrading-what-is-upgraded.html

1.2.2 `mysqldump` →新環境へのリストアを行う場合の注意点

MySQL 8.0 の仕様変更により、旧バージョンで取得したダンプファイルをリストアする際にエラーが発生する場合があります。

[*1] 無料で登録可能です。登録するとセミナー受講申し込みや、ホワイトペーパー・各種資料の閲覧等が可能になります。

- https://hit.hateblo.jp/entry/MYSQL/MYSQL8/SETTING

個人的には、サーバ全体のダンプファイルを一括取得するのではなく、以下のようにするのが良いのではないかと考えています。

- ユーザは DB のデータとは別に移行する
  - https://speakerdeck.com/yoshiakiyamasaki/20181201-mysqlbaziyonatupufalseji-chu-zhi-shi?slide=70
- DB のデータはスキーマ（DB）別に分割して取得し、移行する
  - 意図しない情報まで新環境に引き継がないようにする

1.2.3　レプリケーションを利用するアップグレードの注意点

`mysqldump` →新環境へのリストアなどで移行する場合、システム停止時間の短縮のためにレプリケーションを利用する方法があります。ところが最近、レプリケーションにおいて複数バージョンが混在する場合のサポートポリシーが変わり、3 バージョン混在[*2]の環境がサポート外となりました[*3]。

- https://qiita.com/hmatsu47/items/2cfbb7dec89ce5ddd647

1.2.4　Upgrade Checker

MySQL 5.7 環境からのアップグレード時に互換性で問題になりそうな箇所を抽出するための Upgrade Checker があります。

前掲のこちらの資料 25〜30 ページ

- https://www.mysql.com/jp/why-mysql/presentations/mysql-80-upgrade-checker-201811-jp/

公式リファレンスマニュアル

- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-upgrade.html

ブログ記事等

- https://yakst.com/ja/posts/5190
- https://gihyo.jp/dev/serial/01/mysql-road-construction-news/0076
- https://mysqlserverteam.com/mysql-shell-8-0-4-introducing-upgrade-checker-utility/

実行例

```
[root@mysql57to80 yum.repos.d]# yum-config-manager --disable mysql57-community
※ MySQL Shell 8.0 をインストールするため、MySQL 5.7 のリポジトリを無効にする
```

[*2] マイナーバージョンであっても 3 バージョン混在はサポート外となります。
[*3] サポート外ではありますが、必ずしも「できなくなった」わけではありません。

```
Loaded plugins: fastestmirror
============================ repo: mysql57-community ============================
[mysql57-community]
async = True
（中略）
username =

[root@mysql57to80 yum.repos.d]# yum-config-manager --enable mysql80-community
※ MySQL 8.0 のリポジトリを有効にする
Loaded plugins: fastestmirror
============================ repo: mysql80-community ============================
[mysql80-community]
async = True
（中略）
username =

[root@mysql57to80 yum.repos.d]# yum install mysql-shell
※ MySQL Shell 8.0 をインストールする
Loaded plugins: fastestmirror
（中略）
Dependencies Resolved

================================================================================
 Package         Arch       Version             Repository                 Size
================================================================================
Installing:
 mysql-shell     x86_64     8.0.16-1.el7        mysql-tools-community      14 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 14 M
Installed size: 47 M
Is this ok [y/d/N]: y
Downloading packages:
（中略）
Installed:
  mysql-shell.x86_64 0:8.0.16-1.el7

Complete!
[root@mysql57to80 yum.repos.d]# mysqlsh -u root -S /var/lib/mysql/mysql.sock
Please provide the password for 'root@/var%2Flib%2Fmysql%2Fmysql.sock': パスワードを入力
Save password for 'root@/var%2Flib%2Fmysql%2Fmysql.sock'? [Y]es/[N]o/Ne[v]er (default No):
[Enter] キーを押す
MySQL Shell 8.0.16
（中略）
No default schema selected; type \use <schema> to set one.

 MySQL  localhost  JS > util.checkForServerUpgrade()
The MySQL server at /var%2Flib%2Fmysql%2Fmysql.sock, version 5.7.26 - MySQL
Community Server (GPL), will now be checked for compatibility issues for
upgrade to MySQL 8.0.16...

1) Usage of old temporal type
 No issues found
```

```
2) Usage of db objects with names conflicting with reserved keywords in 8.0
 No issues found

3) Usage of utf8mb3 charset
 No issues found

4) Table names in the mysql schema conflicting with new tables in 8.0
 No issues found

5) Foreign key constraint names longer than 64 characters
 No issues found
(中略)
17) New default authentication plugin considerations
  Warning: The new default authentication plugin 'caching_sha2_password' offers
    more secure password hashing than previously used 'mysql_native_password'
    (and consequent improved client connection authentication). However, it also
    has compatibility implications that may affect existing MySQL installations.
    If your MySQL installation must serve pre-8.0 clients and you encounter
    compatibility issues after upgrading, the simplest way to address those
    issues is to reconfigure the server to revert to the previous default
    authentication plugin (mysql_native_password). For example, use these lines
    in the server option file:

    [mysqld]
    default_authentication_plugin=mysql_native_password

    However, the setting should be viewed as temporary, not as a long term or
    permanent solution, because it causes new accounts created with the setting
    in effect to forego the improved authentication security.
    If you are using replication please take time to understand how the
    authentication plugin changes may impact you.
  More information:
    https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-cachin
g-sha2-password-compatibility-issues
    https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-cachin
g-sha2-password-replication

Errors:   0
Warnings: 1
Notices:  0

No fatal errors were found that would prevent an upgrade, but some potential issues were detec
ted. Please ensure that the reported issues are not significant before upgrading.

MySQL  localhost  JS > \q
Bye!
```

■コラム: Upgrade Checker のチェック項目

Upgrade Checker はバージョンアップのたびにチェック項目が増えています。8.0.15 時点で 15 項目だったのが 8.0.16 では 17 項目になりました。また、8.0.16 ではターゲットバージョンを指定してチェックすることができるようになりました。

なお、MySQL 5.5・5.6 からの移行で利用可能な非公式 Upgrade Checker（yoku0825 さん作）もあります。

- https://github.com/yoku0825/p5-mysql-upgrade-checker

ブログ記事等

- https://yoku0825.blogspot.com/2018/07/mysql-shellupgrade-checkerperl-5.html
- http://next4us-ti.hatenablog.com/entry/2018/12/05/085115

1.2.5　データディクショナリの InnoDB 化

前掲の資料にも説明がありましたが、MySQL 8.0 からデータディクショナリが InnoDB 化されました。トランザクション対応という触れ込みですが、今のところ DDL は基本的にトランザクション非対応です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/data-dictionary.html

ブログ記事等

- https://mysqlserverteam.com/mysql-8-0-data-dictionary-architecture-and-design/
- https://lefred.be/content/mysql-8-0-data-dictionary-tables-and-why-they-should-stay-protected/

InnoDB テーブル作成時、以前は`.ibd` ファイルとともに`.frm` ファイルが生成されましたが、MySQL 8.0 では`.frm` ファイルは生成されません。

MySQL 5.7 からのインプレースアップグレード時、サーバを最初に起動したタイミングで変換が行われます。

1.3　設定パラメータ・起動パラメータの変更

以下を確認して、設定パラメータの変更を計画します。

前掲のこちらの資料 19〜24 ページ

- https://www.mysql.com/jp/why-mysql/presentations/mysql-80-upgrade-checker-201811-jp/
  - 特に非推奨化・廃止された機能（21〜22 ページ）に注意。`sql_mode`、アカウント管理など。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html
- https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html

1.3.1　対象となるサーバ設定パラメータ・起動パラメータ

前述の資料で示されているもののほか、いくつか変更点があります（デフォルトの変更・廃止など）。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/added-deprecated-removed.html

公式サーババージョンリファレンス

- https://dev.mysql.com/doc/mysqld-version-reference/en/

とみたまさひろさん作・バージョン間パラメータ比較ができるページ

- https://tmtm.github.io/mysql-params/

デフォルト値が変更されたパラメータの例

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_explicit_defaults_for_timestamp
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_allowed_packet
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_table_open_cache
- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_event-scheduler

サーバ変数名が変更されたパラメータの例

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_transaction_read_only
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_transaction_isolation

1.3.2　その他の変更点

`bind-address` サーバ変数（起動オプション）で複数のアドレスをサポート

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_bind_address

管理専用ポートの追加

- http://mita2db.blogspot.com/2019/05/mysqladministrative-network-interface.html
- https://dev.mysql.com/doc/refman/8.0/en/client-connections.html
- https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_service-connection-admin
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_admin_address

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_admin_port
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_create_admin_listener_thread

サーバーに mysqld_safe 機能を追加

- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_daemonize
- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_initialize
- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_basedir

TLS 1.3 サポート

- https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-protocols-ciphers.html

1.4　キーワードと予約語

SQL の中で予約語をテーブル名・カラム名等に使用する場合、バッククォート等で囲む必要があります。MySQL 8.0 で増えた予約語がテーブル名等に使われている場合は要注意です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/keywords.html

公式サーババージョンリファレンス

- https://dev.mysql.com/doc/mysqld-version-reference/en/keywords.html

ブログ記事等

- 著者ブログ
  - https://qiita.com/hmatsu47/items/a1da0e06f0597acd6502

1.5　キャラクタセットと照合順序

MySQL 8.0 では Unicode 9.0 がサポートされるとともに、デフォルトのキャラクタセットが utf8mb4 に変更されました。あわせて照合順序（COLLATION）も拡張されています。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/charset-charsets.html

ブログ記事等

- https://mysqlserverteam.com/mysql-8-0-1-japanese-collation-for-utf8mb4-ja_jp/
- https://mysqlserverteam.com/mysql-8-0-kana-sensitive-collation-for-japanese-ja_jp/
- https://tmtms.hatenablog.com/entry/201805/mysql-innovation-day-tokyo

公式リファレンスマニュアル／設定パラメータ等

- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_character-set-server
- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#option_mysqld_collation-server
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_character_set_database
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_character_set_client
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_character_set_connection
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_character_set_results
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_collation_for_utf8mb4

デフォルトの変更とあわせて、`utf8mb4` 指定時の処理高速化も行われています。

ブログ記事等

- http://dimitrik.free.fr/blog/archives/2018/04/mysql-performance-80-and-utf8-impact.html

ブログ記事等／照合順序による差

- https://yoku0825.blogspot.com/2018/12/utf8mb40900aici.html

また、正規表現ライブラリの変更とあわせて、正規表現で Unicode がサポートされました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/regexp.html

ブログ記事等

- https://www.s-style.co.jp/blog/2018/09/2519/

1.6　リンク集 URL

図 1.1: https://hmatsu47.hatenablog.com/book_mysql80_011

第 2 章

ユーザ管理・認証・権限設定の変更と新機能

2.1 認証プラグイン

MySQL 8.0 では Caching sha2 authentication プラグインが導入され、デフォルトとなりました。従来の MySQL Native Password プラグインと比べて、以下の点が優れています。

- 安全なパスワード暗号化
- 高いパフォーマンス

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/caching-sha2-pluggable-authentication.html
- https://dev.mysql.com/doc/refman/8.0/en/mysql-command-options.html#option_mysql_get-server-public-key

ブログ記事等

- https://www.s-style.co.jp/blog/2018/05/1807/
- https://yoku0825.blogspot.com/2018/01/mysql-804.html
- https://yoku0825.blogspot.com/2018/10/mysql-80cachingsha2password-ssl.html

アプリケーションからの接続に使うコネクタによっては、Caching sha2 authentication プラグインに対応していないことがあります。その場合は従来の MySQL Native Password プラグインをデフォルトにするか、接続ユーザに対する認証プラグインとして指定します（ブログ記事等の 1 つ目）。

その他、認証プラグインの注意点についてはブログ記事等の 2 つ目・3 つ目を参照してください。

実行例

MySQL 5.7 からアップグレードした環境で確認してみます。

```
mysql> SELECT user, host, plugin, authentication_string FROM mysql.user;
+------------------+-----------+-----------------------+-------------------------------------
----------------------------------+
```

```
| user             | host      | plugin                | authentication_string
                                 |
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
| mysql.infoschema | localhost | caching_sha2_password | $A$005$THISISACOMBINATIONOFINVALIDSAL
TANDPASSWORDTHATMUSTNEVERBRBEUSED |
| mysql.session    | localhost | mysql_native_password | *THISISNOTAVALIDPASSWORDTHATCANBEUSED
HERE                              |
| mysql.sys        | localhost | caching_sha2_password | $A$005$THISISACOMBINATIONOFINVALIDSAL
TANDPASSWORDTHATMUSTNEVERBRBEUSED |
| root             | localhost | mysql_native_password | *0DB50A1723AF0498B6B290CD9E827AAB206D
4BC7                              |
| testuser         | localhost | mysql_native_password | *A4C8C2403A2F78879BE15B52489D5F3D24B6
6C62                              |
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
5 rows in set (0.00 sec)

mysql> CREATE USER 'hmatsu47'@'localhost' IDENTIFIED WITH mysql_native_password BY 'HOgeFug@';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT user, host, plugin, authentication_string FROM mysql.user;
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
| user             | host      | plugin                | authentication_string
                                 |
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
| hmatsu47         | localhost | mysql_native_password | *5528FA7F88CFC88E779DAE7C94511C249878
B7F8                              |
| mysql.infoschema | localhost | caching_sha2_password | $A$005$THISISACOMBINATIONOFINVALIDSAL
TANDPASSWORDTHATMUSTNEVERBRBEUSED |
| mysql.session    | localhost | mysql_native_password | *THISISNOTAVALIDPASSWORDTHATCANBEUSED
HERE                              |
| mysql.sys        | localhost | caching_sha2_password | $A$005$THISISACOMBINATIONOFINVALIDSAL
TANDPASSWORDTHATMUSTNEVERBRBEUSED |
| root             | localhost | mysql_native_password | *0DB50A1723AF0498B6B290CD9E827AAB206D
4BC7                              |
| testuser         | localhost | mysql_native_password | *A4C8C2403A2F78879BE15B52489D5F3D24B6
6C62                              |
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
6 rows in set (0.00 sec)

mysql> ALTER USER 'hmatsu47'@'localhost' IDENTIFIED WITH caching_sha2_password;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT user, host, plugin, authentication_string FROM mysql.user;
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
| user             | host      | plugin                | authentication_string
                                 |
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
| hmatsu47         | localhost | caching_sha2_password |
|
| mysql.infoschema | localhost | caching_sha2_password | $A$005$THISISACOMBINATIONOFINVALIDSAL
```

```
TANDPASSWORDTHATMUSTNEVERBRBEUSED |
| mysql.session    | localhost | mysql_native_password | *THISISNOTAVALIDPASSWORDTHATCANBEUSED
HERE                               |
| mysql.sys        | localhost | caching_sha2_password | $A$005$THISISACOMBINATIONOFINVALIDSAL
TANDPASSWORDTHATMUSTNEVERBRBEUSED |
| root             | localhost | mysql_native_password | *0DB50A1723AF0498B6B290CD9E827AAB206D
4BC7                               |
| testuser         | localhost | mysql_native_password | *A4C8C2403A2F78879BE15B52489D5F3D24B6
6C62                               |
+------------------+-----------+-----------------------+-----------------------------------
----------------------------------+
6 rows in set (0.00 sec)
```

ブログ記事等の 2 つ目に示されている通り、ALTER USER 〜 IDENTIFIED WITH を実行するとパスワードが消えてしまいます。

2.2 ユーザ・パスワードと権限の管理

2.2.1 ユーザアカウントごとに 2 つのアクティブパスワードをサポート

ユーザパスワードの変更を行う際、変更ミスがあると認証ができなくなるため、変更には神経を使います。また、アプリケーションで利用するユーザアカウントでは、パスワードの変更とアプリケーション（もしくはアプリケーション設定）の変更を同じタイミングで実施しないといけないため、レプリカを多数使う環境ではメンテナンス停止なしにパスワードを変更するのが困難でした。

MySQL 8.0 では、ユーザーアカウントごとに 2 つのアクティブパスワードをサポートするようになりました。最初にユーザパスワードを変更し、全てのアプリケーション（もしくはアプリケーション設定）の変更を段階的に進め、完了後に古いパスワードを無効化する、という運用が可能です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/password-management.html#dual-passwords

ブログ記事等

- https://gihyo.jp/dev/serial/01/mysql-road-construction-news/0090

2.2.2 その他のユーザ・パスワード管理、権限管理に関わる変更点

SUPER 権限を動的権限に分割

- https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#privileges-provided-dynamic

外部キー制約を作成するには親テーブルに対する REFERENCES 権限が必要に

- https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html

GRANT ステートメントによるユーザ作成の廃止

- http://next4us-ti.hatenablog.com/entry/2018/07/13/123322

ALTER USER ／ SET PASSWORD 時に変更前パスワードの入力を要求

- https://dev.mysql.com/doc/refman/8.0/en/password-management.html

セキュアセッション変数の設定（MYSQL_SESSION_ADMIN 権限）

- https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#
  priv_session-variables-admin

--skip-grant-tables オプション付きで起動したときに--skip-networking も有効化する

- https://dev.mysql.com/doc/refman/8.0/en/server-options.html#
  option_mysqld_skip-grant-tables

ACL ステートメントをアトミックにする

- https://www.ospn.jp/osc2018-osaka/pdf/osc2018_osaka_20180126_MySQL_Update.pdf
  – 72 ページを参照

ログイン失敗時に認証を遅延させる

- http://masato.ushio.org/blog/index.php/2018/05/08/whats_new_in_mysql8_translation/
  –「ユーザパスワードへのブルートフォース攻撃を低速化する」

LDAP 認証プラグインに関する機能追加（Enterprise）

- https://dev.mysql.com/doc/refman/8.0/en/ldap-pluggable-authentication.html

2.3　yaSSL から OpenSSL に移行し動的リンク化

認証そのものではありませんが、認証機能から利用されるため関連項目としてあげておきます。

SSL/TLS ライブラリが yaSSL から OpenSSL に変更され、ライブラリのリンク方式が動的になりました。TLS 1.3 に対応するなどセキュリティ強化につながっています。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/ssl-libraries.html
- https://dev.mysql.com/doc/refman/8.0/en/fips-mode.html

ブログ記事等

- https://www.s-style.co.jp/blog/2018/07/2112/

2.4 ロール

MySQL 8.0 では権限に関わる機能としてロール（ROLE）がサポートされました。ロールの基本的な使い方は以下の通りです。

- 特別な権限（スキーマ・テーブル・ユーザの CREATE・DROP などの管理業務に必要な権限）は、ユーザ個人に直接付与するのではなくロールに付与する
- それぞれのユーザが適用できるロールをあらかじめ指定しておく
- ユーザは特別な権限を必要とする操作を実行するときに、ロールを適用してから実行する

MySQL 8.0 でサポートされた主なロール機能は以下の通りです。

- ロールの作成と削除
- ロールに対する権限の付与と剥奪
- 適用するロールの切り替え
- ロールに関する情報の表示
- ログイン（接続）時に適用されるデフォルトロールの指定
- 必須ロール（mandatory_roles）の指定
- 必須ロールを含めたログイン（接続）時適用ロールの指定

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/roles.html
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_mandatory_roles
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_activate_all_roles_on_login

ブログ記事等

- https://yoku0825.blogspot.com/2016/09/mysql-800role.html
- https://www.s-style.co.jp/blog/2018/07/2123/
- http://next4us-ti.hatenablog.com/entry/2019/04/25/223048
  - MySQL 8.0.16 での変更点（DROP ROLE でユーザアカウントを誤削除しないように）
- 著者ブログ
  - https://qiita.com/hmatsu47/items/e4a49d32685220d492a9
  - https://qiita.com/hmatsu47/items/0cf831b6a39086dfdb0e

実行例

```
[root@mysql57to80 ~]# mysql -u root -p
Enter password:パスワードを入力
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

```
（中略）
mysql> GRANT SELECT ON test.sales_person TO ‘hmatsu47‘@‘localhost‘;
Query OK, 0 rows affected (0.00 sec)
※ユーザには SELECT 権限のみ付与

mysql> CREATE ROLE ‘account_admin‘;
Query OK, 0 rows affected (0.01 sec)
※ロールを作成

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON test.sales_person TO ‘account_admin‘;
Query OK, 0 rows affected (0.01 sec)
※ロールには更新権限も付与

mysql> GRANT ‘account_admin‘ TO ‘hmatsu47‘@‘localhost‘;
Query OK, 0 rows affected (0.00 sec)
※ユーザにロールを割り当て

mysql> QUIT
Bye
[root@mysql57to80 ~]# mysql -u hmatsu47 -p
Enter password:パスワードを入力
Welcome to the MySQL monitor.  Commands end with ; or \g.
（中略）
mysql> USE test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM sales_person;
+----+--------+
| id | name   |
+----+--------+
|  1 | 田中   |
|  2 | 坂井   |
|  3 | 富田   |
|  4 | 三谷   |
+----+--------+
4 rows in set (0.00 sec)
※ SELECT は可能

mysql> INSERT INTO sales_person SET name='梶山';
ERROR 1142 (42000): INSERT command denied to user 'hmatsu47'@'localhost' for table 'sales_pers
on'
※ INSERT はできない
mysql> SET ROLE ‘account_admin‘;
Query OK, 0 rows affected (0.00 sec)
※ロールを有効化

mysql> INSERT INTO sales_person SET name='梶山';
Query OK, 1 row affected (0.01 sec)
※ INSERT もできるようになった

mysql> SELECT * FROM sales_person;
+----+--------+
| id | name   |
+----+--------+
|  1 | 田中   |
```

```
|  2 | 坂井   |
|  3 | 富田   |
|  4 | 三谷   |
|  5 | 梶山   |
+----+--------+
5 rows in set (0.00 sec)
```

2.5 リンク集 URL

図 2.1: https://hmatsu47.hatenablog.com/book_mysql80_021

第 3 章

DDL と管理用 SQL の新機能

3.1 DDL

MySQL 8.0 ではインスタント DDL のサポートなど、DDL 関連の機能が向上しています。

3.1.1 インスタント DDL

`ALTER TABLE` でカラムの追加等を行う際、実データの更新を行わずメタデータ[*1]の更新のみを行う機能です。なお、全ての DDL 処理がインスタント DDL として処理できるわけではありません。インスタント DDL に対応している処理については、公式マニュアルで確認してください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/innodb-online-ddl-operations.html

ブログ記事等

- https://www.s-style.co.jp/blog/2018/09/2525/
- https://www.walksocket.com/archives/715
- https://mysqlserverteam.com/mysql-8-0-innodb-now-supports-instant-add-column/
- http://kenken0807.hatenablog.com/entry/2019/03/18/173600
  - `ADD COLUMN` の方式の違い、およびその後の `UPDATE` 方式の違いによるデータサイズと所要時間の差異調査

実行例

```
mysql> ALTER TABLE test ADD COLUMN str VARCHAR(100), ALGORITHM=INSTANT;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

[*1] テーブルの設計情報など

3.1.2 カラムのデフォルト値指定の拡張（関数・式の利用）

以前は日付型カラムにおいて CURRENT_TIMESTAMP を指定できる程度でしたが、MySQL 8.0 からカラムのデフォルト値として関数や式を指定できるようになりました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/data-type-defaults.html#data-types-defaults-explicit

ブログ記事等

- https://yoku0825.blogspot.com/2018/10/mysql-8013default.html
- https://mysqlserverteam.com/the-mysql-8-0-13-maintenance-release-is-generally-available/

実行例

```
※ sql_require_primary_key=0 の環境で実行した結果です。

mysql> CREATE TABLE def_test (org_str VARCHAR(100), sha_str VARCHAR(64) DEFAULT (SHA2(org_str
, 256)));
Query OK, 0 rows affected (0.03 sec)
※関数を使う

mysql> INSERT INTO def_test SET org_str='abc';
Query OK, 1 row affected (0.01 sec)
※中略

mysql> SELECT * FROM def_test;
+---------+------------------------------------------------------------------+
| org_str | sha_str                                                          |
+---------+------------------------------------------------------------------+
| abc     | ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad |
| 123     | a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3 |
| XYZ     | ade099751d2ea9f3393f0f32d20c6b980dd5d3b0989dea599b966ae0d3cd5a1e |
+---------+------------------------------------------------------------------+
3 rows in set (0.00 sec)

mysql> CREATE TABLE def_test2 (val INT NOT NULL, calc BIGINT DEFAULT (val*(val+1)));
Query OK, 0 rows affected (0.02 sec)
※式を使う

mysql> INSERT INTO def_test2 SET val=10;
Query OK, 1 row affected (0.01 sec)
※中略

mysql> SELECT * FROM def_test2;
+-----+-------+
| val | calc  |
+-----+-------+
|  10 |   110 |
| 100 | 10100 |
```

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

3.1.3 不可視インデックス

不可視インデックス（Invisible Index）は、インデックスをオプティマイザから使われないようにする機能です。

インデックスを運用していると、データの増加や値の偏りなどによって有効に利用されなくなることがありますが、非効率なインデックスだからといっていきなり削除してしまうと、削除に時間が掛かったり意図しない実行計画の変化をもたらす場合があります。

不可視インデックスを使うと、インデックスを削除する前に（インデックスが削除された状態での）オプティマイザの判断を確認することができます。

反対に、インデックスを追加する際にいきなり有効にするのではなく無効（不可視）の状態で追加し、影響を確認してから有効（可視）化する使い方も可能です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/invisible-indexes.html
- https://dev.mysql.com/doc/refman/8.0/en/switchable-optimizations.html
  - オプティマイザスイッチ `use_invisible_indexes`

ブログ記事等

- https://www.s-style.co.jp/blog/2018/08/2275/
- https://hit.hateblo.jp/entry/MYSQL8/INVISIBLE_INDEX

実行例

```
mysql> CREATE TABLE iv_test (id INT PRIMARY KEY AUTO_INCREMENT, val INT, INDEX idx_val (val));
Query OK, 0 rows affected (0.03 sec)
※通常のインデックスを作成

mysql> INSERT INTO iv_test SET val=FLOOR(RAND()*100);
Query OK, 1 row affected (0.01 sec)
※中略

mysql> SELECT * FROM iv_test ORDER BY id;
+----+------+
| id | val  |
+----+------+
|  1 |   39 |
|  2 |   93 |
※中略
| 50 |   73 |
+----+------+
50 rows in set (0.00 sec)
```

```
mysql> EXPLAIN SELECT * FROM iv_test WHERE val BETWEEN 40 AND 59;
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  |
 rows | filtered | Extra                    |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
|  1 | SIMPLE      | iv_test | NULL       | range | idx_val       | idx_val | 5       | NULL |
   10 |   100.00 | Using where; Using index |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
```

※インデックス idx_val が効いている

```
mysql> ALTER TABLE iv_test ALTER INDEX idx_val INVISIBLE;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

※インデックス idx_val を不可視にする。可視化するときは VISIBLE

```
mysql> EXPLAIN SELECT * FROM iv_test WHERE val BETWEEN 40 AND 59;
+----+-------------+---------+------------+------+---------------+------+---------+------+----
--+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | row
s | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+---------+------+----
--+----------+-------------+
|  1 | SIMPLE      | iv_test | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   5
0 |    11.11 | Using where |
+----+-------------+---------+------------+------+---------------+------+---------+------+----
--+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

※インデックス idx_val が効かなくなった

```
mysql> SET optimizer_switch='index_merge=on,index_merge_union=on,index_merge_sort_union=on,ind
ex_merge_intersection=on,engine_condition_pushdown=on,index_condition_pushdown=on,mrr=on,mrr_c
ost_based=on,block_nested_loop=on,batched_key_access=off,materialization=on,semijoin=on,looses
can=on,firstmatch=on,duplicateweedout=on,subquery_materialization_cost_based=on,use_index_exte
nsions=on,condition_fanout_filter=on,derived_merge=on,use_invisible_indexes=on,skip_scan=on';
Query OK, 0 rows affected (0.00 sec)
```

※オプティマイザスイッチ use_invisible_indexes を on に変更

```
mysql> EXPLAIN SELECT * FROM iv_test WHERE val BETWEEN 40 AND 59;
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  |
 rows | filtered | Extra                    |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
|  1 | SIMPLE      | iv_test | NULL       | range | idx_val       | idx_val | 5       | NULL |
   10 |   100.00 | Using where; Using index |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
```

※インデックス idx_val が効くようになった

3.1.4 降順インデックス

従来、MySQL ではインデックス作成時に DESC を指定しても無視されましたが、MySQL 8.0 から降順インデックスを作成できるようになりました。通常は複合インデックスで昇順ソートしたいカラムと降順ソートしたいカラムが混在する場合に使用します。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/descending-indexes.html

ブログ記事等

- http://variable.jp/2017/04/13/mysql8-0-descending-index%E3%81%AE%E3%82%B5%E3%83%9D%E3%83%BC%E3%83%88/
- http://mysqlserverteam.com/mysql-8-0-labs-descending-indexes-in-mysql/
- 著者ブログ
  - https://qiita.com/hmatsu47/items/8c5e7abe204f7ecc5084

実行例

```
mysql> CREATE TABLE di_test (id INT PRIMARY KEY AUTO_INCREMENT, val1 INT, val2 INT, INDEX id
x_val(val1 ASC, val2 ASC));
Query OK, 0 rows affected (0.02 sec)
※複合インデックス idx_val を両カラムとも昇順で作成

mysql> INSERT INTO di_test SET val1=FLOOR(RAND()*100), val2=FLOOR(RAND()*100);
Query OK, 1 row affected (0.01 sec)
※中略

mysql> SELECT * FROM di_test ORDER BY id;
+----+------+------+
| id | val1 | val2 |
+----+------+------+
|  1 |   62 |   91 |
|  2 |   71 |   82 |
※中略
| 50 |   66 |   41 |
+----+------+------+
50 rows in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM di_test ORDER BY val1 ASC, val2 ASC;
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  |
 rows | filtered | Extra       |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-------------+
|  1 | SIMPLE      | di_test | NULL       | index | NULL          | idx_val | 10      | NULL |
   50 |   100.00 | Using index |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-------------+
```

```
1 row in set, 1 warning (0.00 sec)

mysql> EXPLAIN SELECT * FROM di_test ORDER BY val1 ASC, val2 DESC;
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-----------------------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  |
 rows | filtered | Extra                       |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-----------------------------+
|  1 | SIMPLE      | di_test | NULL       | index | NULL          | idx_val | 10      | NULL |
   50 |   100.00 | Using index; Using filesort |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-----------------------------+
1 row in set, 1 warning (0.00 sec)
※ Using filesort が表示された

mysql> ALTER TABLE di_test DROP INDEX idx_val, ADD INDEX idx_val(val1 ASC, val2 DESC);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN SELECT * FROM di_test ORDER BY val1 ASC, val2 DESC;
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  |
 rows | filtered | Extra       |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-------------+
|  1 | SIMPLE      | di_test | NULL       | index | NULL          | idx_val | 10      | NULL |
   50 |   100.00 | Using index |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
※ Using filesort が表示されなくなった
```

3.1.5 関数・式インデックス

MySQL 8.0 より、インデックスの定義として関数や式を使うことができるようになりました。MySQL 5.7 でも生成列（Generated Column）によって同様の機能を利用することができましたが、列ではなくインデックスとして定義できるのがポイントです[*2]。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/create-index.html#create-index-functional-key-parts

ブログ記事等

- https://yoku0825.blogspot.com/2018/10/mysql-8013_25.html
  - 式インデックスの例

[*2] MySQL 5.7 の生成列は「更新できない列」であり、ORM（オブジェクト関係マッピング）との相性が悪い、という問題があります。

- 著者ブログ／ MySQL 5.7 の生成列を使う例
  - https://qiita.com/hmatsu47/items/128ece7276e4deac1477

実行例

※関数インデックスの例。降順インデックスの実行例で使ったテーブルを流用。

```
mysql> EXPLAIN SELECT * FROM di_test WHERE MOD(val1, 10) < 3;
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  |
 rows | filtered | Extra                    |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
|  1 | SIMPLE      | di_test | NULL       | index | NULL          | idx_val | 10      | NULL |
   50 |   100.00 | Using where; Using index |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+
------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
※ val1 が idx_val の 1 列目に定義されているのでインデックスフルスキャンになっている。

mysql> ALTER TABLE di_test ADD INDEX idx_func((MOD(val1, 10)));
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
※関数インデックス idx_func を定義。

mysql> EXPLAIN SELECT * FROM di_test WHERE MOD(val1, 10) < 3;
+----+-------------+---------+------------+-------+---------------+----------+---------+------
+------+----------+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key      | key_len | ref
| rows | filtered | Extra       |
+----+-------------+---------+------------+-------+---------------+----------+---------+------
+------+----------+-------------+
|  1 | SIMPLE      | di_test | NULL       | range | idx_func      | idx_func | 5       | NULL
|   14 |   100.00 | Using where |
+----+-------------+---------+------------+-------+---------------+----------+---------+------
+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
※ idx_func の range スキャンに変わった。
```

3.1.6 主キーのないテーブルの禁止（sql_require_primary_key）

MySQL 8.0 より、主キーのないテーブルの作成を禁止するサーバシステム変数 sql_require_primary_key が新設されました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_sql_require_primary_key

ブログ記事等

- https://yoku0825.blogspot.com/2018/10/mysql-8013primary-key.html

3.1.7 CHECK 制約

MySQL 8.0.16 より、CHECK 制約がサポートされました[*3]。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html
- https://dev.mysql.com/doc/refman/8.0/en/table-constraints-table.html
  - INFORMATION_SCHEMA.TABLE_CONSTRAINTS テーブル
    - CONSTRAINT_TYPE 列

ブログ記事等

- https://mysqlserverteam.com/mysql-8-0-16-introducing-check-constraint/
- https://yoku0825.blogspot.com/2019/04/mysql-8016checknot-enforced.html
- http://next4us-ti.hatenablog.com/entry/2019/04/25/133554

3.1.8 その他の DDL 新機能

アトミックな DDL・バイナリログからのアトミックな DDL リカバリ

- https://dev.mysql.com/doc/refman/8.0/en/atomic-ddl.html

インプレース処理でのキャラクタセット変換

- https://dev.mysql.com/doc/refman/8.0/en/alter-table.html#alter-table-performance
- https://dev.mysql.com/doc/refman/8.0/en/alter-table.html#alter-table-character-set
  - INPLACE との組み合わせが可能

ADD DATAFILE を伴わない CREATE TABLESPACE

- https://dev.mysql.com/doc/refman/8.0/en/create-tablespace.html
  - ADD DATAFILE

LOCK TABLES を伴う RENAME TABLE

- https://dev.mysql.com/doc/refman/8.0/en/rename-table.html
  -「As of MySQL 8.0.13, …」

[*3] 過去のバージョンでは制約として定義した内容がエラーにならない場合でも、処理上は無視されていました。

The ddl_rewriter Plugin

- https://dev.mysql.com/doc/refman/8.0/en/ddl-rewriter.html

3.2 管理用 SQL

MySQL 8.0 では、MySQL 5.7 で始まった「**OS レベルではなく SQL レベルの操作でサーバの管理を行う**」機能の実装がさらに進みました。

3.2.1 RESTART ステートメント

MySQL 5.7 で導入された SHUTDOWN ステートメントに続いて、MySQL 8.0 では RESTART ステートメントが使えるようになりました。OS 操作レベルではなく、MySQL に接続して SQL 操作レベルでのサーバ再起動が可能です。

なお、実行には SHUTDOWN 権限が必要です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/restart.html

ブログ記事等

- https://www.s-style.co.jp/blog/2018/09/2527/

3.2.2 SET PERSIST ステートメント

従来、SET ステートメントで設定した設定値は、サーバを再起動すると消えてしまっていました。MySQL 8.0 では、SET PERSIST ステートメントにより設定値が保存され、サーバを再起動しても維持されるようになりました。

また、SET PERSIST_ONLY ステートメントによって、動作中のサーバには影響を与えず、次回（再）起動時に有効になる形で設置値を変更できるようになりました。

そして、RESET PERSIST ステートメントによって設定値をデフォルトに戻すことができます。

必要な権限など細かい仕様については、公式リファレンスマニュアル・ブログ記事等を参照してください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/persisted-system-variables.html
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_persist_only_admin_x509_subject
  - SET PERSIST_ONLY するユーザに対する追加の認証設定について

ブログ記事等

- https://www.s-style.co.jp/blog/2018/08/2262/
- https://yoku0825.blogspot.com/2019/04/mysql-8015-set-persistonly.html
  - SET PERSIST_ONLY についての注意喚起

3.3 リンク集 URL

図 3.1: https://hmatsu47.hatenablog.com/book_mysql80_031

第 4 章

CTE とウィンドウ関数

4.1 CTE（Common Table Expressions）

CTE（共通テーブル式）は、主たる SQL の問い合わせを実行するために補助的に使う一時テーブルを定義するものです。`WITH` で記述を始めるので `WITH` 句とも呼びます。

単純なテーブル構造だけではなく、`WITH RECURSIVE` で再帰的に記述することもできるのがポイントです。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/with.html
- https://dev.mysql.com/doc/refman/8.0/en/with.html#common-table-expressions-recursive

ブログ記事等

- https://www.s-style.co.jp/blog/2017/07/884/
- https://yakst.com/ja/posts/4322
- https://yoku0825.blogspot.com/2018/04/mysql-80ctewith-recursive1000.html
- https://tombo2.hatenablog.com/entry/2019/03/10/222732
- http://next4us-ti.hatenablog.com/entry/2019/03/21/003927
- 著者ブログ
  - https://qiita.com/hmatsu47/items/01211556089b19913d05

実行例

著者ブログで使ったデータのうち、今回は `order_id=3` のスタイリッシュパッケージを抽出・更新してみます。

```
mysql> CREATE TABLE order_detail (
    ->   detail_id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    ->   order_id INT UNSIGNED NOT NULL,
    ->   parent_id INT UNSIGNED,
    ->   product_name VARCHAR(100),
    ->   cancel_flag INT UNSIGNED NOT NULL,
    ->   INDEX (order_id),
```

```
    ->   INDEX (parent_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO order_detail VALUES( 1,1,NULL,'車両本体 S グレード'        ,0);
Query OK, 1 row affected (0.01 sec)
（中略）

mysql> INSERT INTO order_detail VALUES(22,3,  19,'リアスポイラー'            ,0);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM order_detail;
+-----------+----------+-----------+----------------------------------------+-------------+
| detail_id | order_id | parent_id | product_name                           | cancel_flag |
+-----------+----------+-----------+----------------------------------------+-------------+
|         1 |        1 |      NULL | 車両本体 S グレード                     |           0 |
|         2 |        1 |         1 | セーフティーパッケージ                  |           0 |
|         3 |        1 |         2 | 衝突回避ブレーキシステム                |           0 |
|         4 |        1 |         2 | 追加エアバッグセット                    |           0 |
|         5 |        1 |         3 | サイドエアバッグ                        |           0 |
|         6 |        1 |         3 | カーテンエアバッグ                      |           0 |
|         7 |        1 |         1 | 18 インチセット                         |           0 |
|         8 |        1 |         7 | 225/40R18 ラジアルタイヤ                |           0 |
|         9 |        1 |         7 | 18 インチアルミホイール                 |           0 |
|        10 |        1 |         1 | フロアマット                            |           0 |
|        11 |        2 |      NULL | 車両本体 B グレード                     |           0 |
|        12 |        2 |        11 | サイドバイザー                          |           0 |
|        13 |        2 |        11 | フロアマット                            |           0 |
|        14 |        3 |      NULL | 車両本体 X グレード                     |           0 |
|        15 |        3 |        14 | スタイリッシュパッケージ                |           0 |
|        16 |        3 |        15 | 18 インチセット                         |           0 |
|        17 |        3 |        16 | 225/40R18 ラジアルタイヤ                |           0 |
|        18 |        3 |        16 | 18 インチアルミホイール                 |           0 |
|        19 |        3 |        15 | エアロセット B                          |           0 |
|        20 |        3 |        19 | フロントアンダースポイラー              |           0 |
|        21 |        3 |        19 | サイドステップ                          |           0 |
|        22 |        3 |        19 | リアスポイラー                          |           0 |
+-----------+----------+-----------+----------------------------------------+-------------+
22 rows in set (0.00 sec)
```

※データの構造は以下の通り（著者ブログより）。

```
order#1-+-車両本体 S グレード
        |
        +-+-セーフティーパッケージ
        | |
        | +---衝突回避ブレーキシステム
        | |
        | +-+-追加エアバッグセット
        |   |
        |   +---サイドエアバッグ
        |   |
        |   +---カーテンエアバッグ
        |
        +-+-18 インチセット
        | |
        | +---225/40R18 ラジアルタイヤ
```

```
        | |
        | +---18 インチアルミホイール
        |
        +---フロアマット

order#2-+-車両本体 B グレード
        |
        +---サイドバイザー
        |
        +---フロアマット

order#3-+-車両本体 X グレード
        |
        +-+-スタイリッシュパッケージ
          |
          +-+-18 インチセット
          | |
          | +---225/40R18 ラジアルタイヤ
          | |
          | +---18 インチアルミホイール
          |
          +-+-エアロセット B
            |
            +---フロントアンダースポイラー
            |
            +---サイドステップ
            |
            +---リアスポイラー

mysql> WITH RECURSIVE product_order AS
    -> (
    ->   SELECT detail_id, parent_id, product_name, cancel_flag
    ->    FROM ctetest.order_detail
    ->     WHERE order_id = 3 AND product_name = 'スタイリッシュパッケージ'
    ->   UNION ALL
    ->   SELECT child.detail_id, child.parent_id, child.product_name, child
.cancel_flag
    ->    FROM ctetest.order_detail AS child, product_order
    ->     WHERE product_order.detail_id = child.parent_id
    -> )
    -> SELECT * FROM product_order;
+-----------+-----------+---------------------------------------+-------------+
| detail_id | parent_id | product_name                          | cancel_flag |
+-----------+-----------+---------------------------------------+-------------+
|        15 |        14 | スタイリッシュパッケージ              |           0 |
|        16 |        15 | 18 インチセット                       |           0 |
|        19 |        15 | エアロセット B                        |           0 |
|        17 |        16 | 225/40R18 ラジアルタイヤ              |           0 |
|        18 |        16 | 18 インチアルミホイール               |           0 |
|        20 |        19 | フロントアンダースポイラー            |           0 |
|        21 |        19 | サイドステップ                        |           0 |
|        22 |        19 | リアスポイラー                        |           0 |
+-----------+-----------+---------------------------------------+-------------+
8 rows in set (0.02 sec)
※ order_id=3 に含まれるスタイリッシュパッケージと、その子・孫にあたる行が抽出された。

mysql> WITH RECURSIVE product_order AS
```

```
    -> (
    ->   SELECT detail_id, parent_id, product_name, cancel_flag
    ->    FROM ctetest.order_detail
    ->     WHERE order_id = 3 AND product_name = 'スタイリッシュパッケージ'
    ->   UNION ALL
    ->   SELECT child.detail_id, child.parent_id, child.product_name, child
.cancel_flag
    ->    FROM ctetest.order_detail AS child, product_order
    ->     WHERE product_order.detail_id = child.parent_id
    -> )
    -> UPDATE ctetest.order_detail
    ->  SET cancel_flag = 1
    ->   WHERE detail_id IN
    ->    (SELECT detail_id FROM product_order);
Query OK, 8 rows affected (0.01 sec)
Rows matched: 8  Changed: 8  Warnings: 0
```

※ CTE で直接 UPDATE はできないので、CTE をサブクエリで使う。DELETE の場合も同じ。

```
mysql> SELECT * FROM ctetest.order_detail WHERE cancel_flag = 1;
+-----------+----------+-----------+--------------------------------------+-------------+
| detail_id | order_id | parent_id | product_name                         | cancel_flag |
+-----------+----------+-----------+--------------------------------------+-------------+
|        15 |        3 |        14 | スタイリッシュパッケージ             |           1 |
|        16 |        3 |        15 | 18 インチセット                      |           1 |
|        17 |        3 |        16 | 225/40R18 ラジアルタイヤ             |           1 |
|        18 |        3 |        16 | 18 インチアルミホイール              |           1 |
|        19 |        3 |        15 | エアロセット B                       |           1 |
|        20 |        3 |        19 | フロントアンダースポイラー           |           1 |
|        21 |        3 |        19 | サイドステップ                       |           1 |
|        22 |        3 |        19 | リアスポイラー                       |           1 |
+-----------+----------+-----------+--------------------------------------+-------------+
8 rows in set (0.00 sec)
```

※ CTE 抽出行のみ、cancel_flag が 1 に更新されている。

4.2　ウィンドウ関数（Window Function）

MySQL 8.0 ではウィンドウ関数も利用できるようになりました。ウィンドウ関数は、テーブルに存在する複数の行を、区間に分割して集計する機能です。集約関数（GROUP BY）とは違い、複数の行がまとめられることはなく、個々の行が返却されます。

関数名	説明
CUME_DIST()	累積分布値
DENSE_RANK()	パーティション内の現在行の順位（ギャップなし）
FIRST_VALUE()	ウィンドウフレームの最初の行の値
LAG()	パーティション内の前行の値
LAST_VALUE()	ウィンドウフレームの最終行の値
LEAD()	パーティション内の次行の値
NTH_VALUE()	ウィンドウフレームの N 行目の値
NTILE()	パーティション内の現在行が含まれるバケット番号
PERCENT_RANK()	パーセントランク値
RANK()	パーティション内の現在行の順位（ギャップあり）
ROW_NUMBER()	パーティション内の現在の行番号

利用可能な関数の詳細については、公式リファレンスマニュアルおよびブログ記事等の 2 つ目・3 つ目を参照してください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/window-functions.html

ブログ記事等

- http://blog.kimuradb.com/?eid=877509
- http://next4us-ti.hatenablog.com/entry/2019/03/24/225924
- https://tombo2.hatenablog.com/entry/2019/03/12/231618
- 著者ブログ
  - https://qiita.com/hmatsu47/items/6cc0e69f3895f3e4a486
  - https://qiita.com/hmatsu47/items/7976e81100604f8984d2

実行例

著者ブログの記事と同様の集計を、愛知県ではなく静岡県のデータで実行してみました。

```
mysql> CREATE TABLE shizuoka (id INT PRIMARY KEY AUTO_INCREMENT,
    -> ctv_name VARCHAR(50) NOT NULL,
    -> population INT NOT NULL,
    -> ctv_type INT NOT NULL) ENGINE innodb;
Query OK, 0 rows affected (0.05 sec)
※著者ブログで示した例のテーブル構造のうち一部を省略。

mysql> INSERT INTO shizuoka SET ctv_name='静岡市', population=704043, ctv_type=1;
Query OK, 1 row affected (0.00 sec)
（中略）

mysql> INSERT INTO shizuoka SET ctv_name='周智郡森町', population=18507, ctv_type=5;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM shizuoka;
+----+----------------------+------------+-----------+
| id | ctv_name             | population | ctv_type |
```

```
+----+----------------------+------------+----------+
|  1 | 静岡市               |     704043 |        1 |
|  2 | 浜松市               |     804989 |        1 |
|  3 | 沼津市               |     196530 |        3 |
|  4 | 熱海市               |      37225 |        4 |
|  5 | 三島市               |     110505 |        4 |
|  6 | 富士宮市             |     133290 |        4 |
|  7 | 伊東市               |      69597 |        4 |
|  8 | 島田市               |      98909 |        4 |
|  9 | 富士市               |     254203 |        3 |
| 10 | 磐田市               |     169931 |        4 |
| 11 | 焼津市               |     140189 |        4 |
| 12 | 掛川市               |     117605 |        4 |
| 13 | 藤枝市               |     145789 |        4 |
| 14 | 御殿場市             |      88494 |        4 |
| 15 | 袋井市               |      87938 |        4 |
| 16 | 下田市               |      21937 |        4 |
| 17 | 裾野市               |      52332 |        4 |
| 18 | 湖西市               |      59861 |        4 |
| 19 | 伊豆市               |      31089 |        4 |
| 20 | 御前崎市             |      32996 |        4 |
| 21 | 菊川市               |      47850 |        4 |
| 22 | 伊豆の国市           |      49082 |        4 |
| 23 | 牧之原市             |      46102 |        4 |
| 24 | 賀茂郡東伊豆町       |      12418 |        5 |
| 25 | 賀茂郡河津町         |       7339 |        5 |
| 26 | 賀茂郡南伊豆町       |       8456 |        5 |
| 27 | 賀茂郡松崎町         |       6768 |        5 |
| 28 | 賀茂郡西伊豆町       |       8083 |        5 |
| 29 | 田方郡函南町         |      37901 |        5 |
| 30 | 駿東郡清水町         |      32606 |        5 |
| 31 | 駿東郡長泉町         |      43185 |        5 |
| 32 | 駿東郡小山町         |      18815 |        5 |
| 33 | 榛原郡吉田町         |      29679 |        5 |
| 34 | 榛原郡川根本町       |       7002 |        5 |
| 35 | 周智郡森町           |      18507 |        5 |
+----+----------------------+------------+----------+
35 rows in set (0.00 sec)

mysql> SELECT RANK() OVER (ORDER BY population DESC) AS pop_rank,
    -> ctv_name, population FROM shizuoka;
```

※全市町村のランキングを抽出。

```
+----------+----------------------+------------+
| pop_rank | ctv_name             | population |
+----------+----------------------+------------+
|        1 | 浜松市               |     804989 |
|        2 | 静岡市               |     704043 |
|        3 | 富士市               |     254203 |
```

（中略）

```
|       34 | 榛原郡川根本町       |       7002 |
|       35 | 賀茂郡松崎町         |       6768 |
+----------+----------------------+------------+
35 rows in set (0.02 sec)

mysql> SELECT ctv_type,
    -> RANK() OVER (PARTITION BY ctv_type ORDER BY population DESC)
    ->  AS pop_rank,
```

```
    -> ctv_name, population FROM shizuoka;
```

※市町村種類別のランキングを抽出。なお、静岡県には中核市（ctv_type=2）・村（ctv_type=6）は存在しない。

```
+----------+----------+----------------------+------------+
| ctv_type | pop_rank | ctv_name             | population |
+----------+----------+----------------------+------------+
|        1 |        1 | 浜松市               |     804989 |
|        1 |        2 | 静岡市               |     704043 |
|        3 |        1 | 富士市               |     254203 |
|        3 |        2 | 沼津市               |     196530 |
|        4 |        1 | 磐田市               |     169931 |
|        4 |        2 | 藤枝市               |     145789 |
（中略）
|        5 |       11 | 榛原郡川根本町       |       7002 |
|        5 |       12 | 賀茂郡松崎町         |       6768 |
+----------+----------+----------------------+------------+
35 rows in set (0.00 sec)

mysql> SELECT RANK() OVER w AS pop_rank,
    -> ctv_name, population,
    -> (SUM(population) OVER w / SUM(population) OVER w2)*100
    ->  AS sum_pct,
    -> FORMAT(CUME_DIST() OVER w, 3) AS c_dist FROM shizuoka
    ->  WINDOW w AS (ORDER BY population DESC), w2 AS ();
```

※ sum_pct は、人口が多い市町村から集計した総和が県全体の人口に占める割合。c_dist=0.2 のところが「上位 2 割（35 市区町村のうちの 7 番目）」。愛知県同様、60% 台中盤だった。

```
+----------+----------------------+------------+----------+--------+
| pop_rank | ctv_name             | population | sum_pct  | c_dist |
+----------+----------------------+------------+----------+--------+
|        1 | 浜松市               |     804989 |  21.5743 | 0.029  |
|        2 | 静岡市               |     704043 |  40.4431 | 0.057  |
|        3 | 富士市               |     254203 |  47.2559 | 0.086  |
|        4 | 沼津市               |     196530 |  52.5231 | 0.114  |
|        5 | 磐田市               |     169931 |  57.0774 | 0.143  |
|        6 | 藤枝市               |     145789 |  60.9846 | 0.171  |
|        7 | 焼津市               |     140189 |  64.7418 | 0.200  |
|        8 | 富士宮市             |     133290 |  68.3140 | 0.229  |
|        9 | 掛川市               |     117605 |  71.4659 | 0.257  |
|       10 | 三島市               |     110505 |  74.4275 | 0.286  |
|       11 | 島田市               |      98909 |  77.0784 | 0.314  |
|       12 | 御殿場市             |      88494 |  79.4501 | 0.343  |
|       13 | 袋井市               |      87938 |  81.8069 | 0.371  |
（中略）
|       35 | 賀茂郡松崎町         |       6768 | 100.0000 | 1.000  |
+----------+----------------------+------------+----------+--------+
35 rows in set (0.00 sec)

mysql> SELECT s.pop_rank, s.ctv_name, s.population,
    -> s.sum_pct, s.c_dist FROM
    -> (
    ->   SELECT RANK() OVER w AS pop_rank,
    ->   ctv_name, population,
    ->   (SUM(population) OVER w / SUM(population) OVER w2)*100
    ->    AS sum_pct,
    ->   FORMAT(CUME_DIST() OVER w, 3) AS c_dist FROM shizuoka
    ->    WINDOW w AS (ORDER BY population DESC), w2 AS ()
    -> ) AS s
    ->  WHERE s.c_dist <= 0.2;
```

※ウィンドウ関数は WHERE 句に書くことができないので、「上位 2 割」で打ち切るときは FROM 句のサブクエリとして書く。

```
+----------+-----------+------------+---------+--------+
| pop_rank | ctv_name  | population | sum_pct | c_dist |
+----------+-----------+------------+---------+--------+
|        1 | 浜松市    |     804989 | 21.5743 | 0.029  |
|        2 | 静岡市    |     704043 | 40.4431 | 0.057  |
|        3 | 富士市    |     254203 | 47.2559 | 0.086  |
|        4 | 沼津市    |     196530 | 52.5231 | 0.114  |
|        5 | 磐田市    |     169931 | 57.0774 | 0.143  |
|        6 | 藤枝市    |     145789 | 60.9846 | 0.171  |
|        7 | 焼津市    |     140189 | 64.7418 | 0.200  |
+----------+-----------+------------+---------+--------+
7 rows in set (0.00 sec)
```

4.3　リンク集 URL

図 4.1: https://hmatsu47.hatenablog.com/book_mysql80_041

第 5 章

JSON とオブジェクトストアの新機能

5.1 JSON 関数

MySQL 5.7 でサポートされた JSON 関数ですが、MySQL 8.0 では新たに以下の関数がサポートされました[*1]。

関数名	説明
JSON_ARRAYAGG()	GROUP BY での集約時に結果セットを単一の JSON 配列として返す
JSON_MERGE_PATCH()	重複したキー値を置き換えて<idx>{JSON ドキュメント}を結合する
JSON_OBJECTAGG()	GROUP BY での集約時に結果セットを単一の JSON オブジェクトとして返す
JSON_PRETTY()	人間が読める形式で JSON 文書を表示する
JSON_STORAGE_FREE()	部分更新後の JSON 列値のバイナリ表記内の解放容量
JSON_STORAGE_SIZE()	JSON ドキュメントのバイナリ表記の格納に使用される容量
JSON_TABLE()	JSON 形式の値をリレーショナルテーブルとして返す

なお、JSON_MERGE() は非推奨になりました（代わりに JSON_MERGE_PRESERVE() を使います）。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/json-utility-functions.html

ブログ記事等

- https://hit.hateblo.jp/entry/MYSQL/8.0/JSON
- https://symfoware.blog.fc2.com/blog-entry-2140.html
- https://masayuki14.hatenablog.com/entry/2018/07/25/080000
- https://masayuki14.hatenablog.com/entry/2018/10/17/170000
- https://yoku0825.blogspot.com/2018/04/mysql-80-select-for-update-skip-locked.html

実行例

JSON_OBJECTAGG()・JSON_STORAGE_SIZE()・JSON_STORAGE_FREE()・JSON_TABLE() の利用例です。

[*1] JSON_STORAGE_FREE()・JSON_TABLE() を除き MySQL 5.7 系列でもサポートされました（5.7.22）。

```
mysql> CREATE TABLE agg_test (id INT PRIMARY KEY AUTO_INCREMENT, j_key VARCHAR(20) UNIQUE NOT
NULL, j_val VARCHAR(100));
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO agg_test SET j_key='NEC', j_val='PC-8801';
Query OK, 1 row affected (0.01 sec)
（中略）

mysql> SELECT * FROM agg_test;
+----+---------+-----------------+
| id | j_key   | j_val           |
+----+---------+-----------------+
|  1 | NEC     | PC-8801         |
|  2 | FUJITSU | FM-8            |
|  3 | SHARP   | MZ-2000         |
|  4 | HITACHI | BASIC MASTER L3 |
+----+---------+-----------------+
4 rows in set (0.00 sec)

mysql> SELECT id, JSON_OBJECTAGG(j_key, j_val) AS old_pc FROM agg_test GROUP BY id ORDER BY i
d;
※ JSON_OBJECTAGG() で JSON オブジェクトに変換。
+----+---------------------------------+
| id | old_pc                          |
+----+---------------------------------+
|  1 | {"NEC": "PC-8801"}              |
|  2 | {"FUJITSU": "FM-8"}             |
|  3 | {"SHARP": "MZ-2000"}            |
|  4 | {"HITACHI": "BASIC MASTER L3"} |
+----+---------------------------------+
4 rows in set (0.00 sec)

mysql> CREATE TABLE storage_test (id INT PRIMARY KEY AUTO_INCREMENT, j_obj JSON);
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO storage_test SET j_obj=JSON_OBJECT('corp', 'NEC', 'pc', 'PC-8801');
Query OK, 1 row affected (0.00 sec)
（中略）

mysql> SELECT *, JSON_STORAGE_SIZE(j_obj) FROM storage_test;
※ JSON_STORAGE_SIZE() で JSON 列のサイズを取得。
+----+-----------------------------------------------+--------------------------+
| id | j_obj                                         | JSON_STORAGE_SIZE(j_obj) |
+----+-----------------------------------------------+--------------------------+
|  1 | {"pc": "PC-8801", "corp": "NEC"}              |                       37 |
|  2 | {"pc": "FM-8", "corp": "FUJITSU"}             |                       38 |
|  3 | {"pc": "MZ-2000", "corp": "SHARP"}            |                       39 |
|  4 | {"pc": "BASIC MASTER L3", "corp": "HITACHI"} |                       49 |
+----+-----------------------------------------------+--------------------------+
4 rows in set (0.00 sec)

mysql> UPDATE storage_test SET j_obj=JSON_REPLACE(j_obj, '$.pc', 'MZ-80B') WHERE id=3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT *, JSON_STORAGE_FREE(j_obj) FROM storage_test;
※ JSON_STORAGE_FREE() で JSON 列を部分更新した際の空きサイズを取得。id=3 の行が 1 文字減少している。
```

```
+----+-----------------------------------------------+--------------------------+
| id | j_obj                                         | JSON_STORAGE_FREE(j_obj) |
+----+-----------------------------------------------+--------------------------+
|  1 | {"pc": "PC-8801", "corp": "NEC"}              |                        0 |
|  2 | {"pc": "FM-8", "corp": "FUJITSU"}             |                        0 |
|  3 | {"pc": "MZ-80B", "corp": "SHARP"}             |                        1 |
|  4 | {"pc": "BASIC MASTER L3", "corp": "HITACHI"} |                        0 |
+----+-----------------------------------------------+--------------------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM JSON_TABLE(
    ->   '[{"name":"青木","dept":"IT 事業部"},{"name":"前田","dept":"コンサル事業部"},{"name":"山
本","dept":["IT 事業部","コンサル事業部"]}]',
    ->   "$[*]"
    ->   COLUMNS(
    ->     name VARCHAR(40) PATH "$.name",
    ->     dept VARCHAR(60) PATH "$.dept"
    ->   )
    -> ) AS tbl_test;
※ JSON_TABLE() で JSON オブジェクトをテーブル形式に変換。name="山本"の非正規列は NULL になっている。
+--------+----------------------+
| name   | dept                 |
+--------+----------------------+
| 青木   | IT 事業部            |
| 前田   | コンサル事業部       |
| 山本   | NULL                 |
+--------+----------------------+
3 rows in set (0.00 sec)
```

5.2 X DevAPI とドキュメントストア

5.2.1 X DevAPI の機能向上

X DevAPI 自体は MySQL 5.7 でサポートされましたが、MySQL 8.0 では Connector の対応も進み、より使いやすくなりました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/document-store.html
  - ドキュメントストア全般
- https://dev.mysql.com/doc/refman/8.0/en/x-plugin.html
  - X プラグイン
- https://dev.mysql.com/doc/x-devapi-userguide/en/
  - X DevAPI ユーザガイド

ブログ記事等

- http://blog.64p.org/entry/2018/07/08/233944
- 著者ブログ
  - https://qiita.com/hmatsu47/items/2de98cd0c9472e72a52a

5.2.2　コード例／ MySQL Connector/J 8.0 を使ったドキュメントストアの利用

X DevAPI によるドキュメントストアの利用例です。MySQL Connector/J 8.0 を使い、Java 8 から実行します[*2]。

リスト 5.1: DocDbTest.java

```
package site.hmatsu47.DocDbTest;

import java.util.List;

import com.mysql.cj.xdevapi.Collection;
import com.mysql.cj.xdevapi.DbDoc;
import com.mysql.cj.xdevapi.DocResult;
import com.mysql.cj.xdevapi.Schema;
import com.mysql.cj.xdevapi.Session;
import com.mysql.cj.xdevapi.SessionFactory;


public class Main {

        public static void main(String args[]) {
                // サーバに接続
                Session session = new SessionFactory().getSession("mysqlx://localhost:33060/te
st_db?user=testuser&password=T35_U53r");

                // DB に接続
                Schema db = session.getSchema("test_db");

                // コレクション'test_collection' を作成
                Collection col = db.createCollection("test_collection", true);

                // コレクションにドキュメントを追加
                col.add("{\"person_id\":1, \"name\":\"青木\", \"dept\":\"IT 事業部\"}")
                .execute();
                col.add("{\"person_id\":2, \"name\":\"前田\", \"dept\":\"コンサル事業部\"}")
                .execute();
                col.add("{\"person_id\":3, \"name\":\"山本\", \"dept\":[\"IT 事業部\",\"コンサ
ル事業部\"]}")
                .execute();

                // コレクションの「person_id」列にインデックスを追加
                col.createIndex("pid_index", "{\"fields\": [{\"field\": \"$.person_id\", \"typ
e\": \"INT\"}]}");

                // コレクションから「dept LIKE '%IT 事業部 %'」を探して表示
                searchDept(col, "IT 事業部");

                System.out.println();

                // コレクションから「dept LIKE '% コンサル事業部 %'」を探して表示
                searchDept(col, "コンサル事業部");
```

[*2] DB のテーブル定義等は著者ブログ記事中のものと同じです。

```
                System.out.println();

                // コレクションから「person_id=2」を探して表示
                searchPid(col, 2);

                System.out.println();

                // コレクションを削除
                db.dropCollection("test_collection");
        }

        // コレクションから対象ドキュメントの「dept」を文字列検索して表示する
        private static void searchDept(Collection col, String keyword) {

                System.out.println("Search: " + keyword);
                DocResult docs = col.find("dept like :dept")
                        .bind("dept", "%" + keyword + "%").execute();

                // 結果を取得して表示
                List<DbDoc> docl = docs.fetchAll();
                docl.forEach(doc -> System.out.println(doc.toFormattedString()));
        }

        // コレクションから対象ドキュメントの「person_id」を数値検索して表示する
        private static void searchPid(Collection col, long value) {

                System.out.println("Search: " + value);
                DocResult docs = col.find("person_id = :pid")
                        .bind("pid", value).execute();

                // 結果を取得して表示
                System.out.println(docs.fetchOne().toFormattedString());
        }
}
```

コードの実行結果

```
Search: IT 事業部
{
"_id" : "00005c93179d0000000000000001",
"dept" : "IT 事業部",
"name" : "青木",
"person_id" : 1
}
{
"_id" : "00005c93179d0000000000000003",
"dept" : ["IT 事業部", "コンサル事業部"],
"name" : "山本",
"person_id" : 3
}

Search: コンサル事業部
```

```
{
"_id" : "00005c93179d0000000000000002",
"dept" : "コンサル事業部",
"name" : "前田",
"person_id" : 2
}
{
"_id" : "00005c93179d0000000000000003",
"dept" : ["IT 事業部", "コンサル事業部"],
"name" : "山本",
"person_id" : 3
}

Search: 2
{
"_id" : "00005c93179d0000000000000002",
"dept" : "コンサル事業部",
"name" : "前田",
"person_id" : 2
}
```

5.3 その他の JSON 新機能

MySQL Shell ／ JSON・BSON データのインポート

- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-json.html
- https://qiita.com/miyamadoKL/items/8d255c5faaeed671b58c
- https://mysqlserverteam.com/mysql-shell-8-0-14-whats-new/

JSON パス表現の拡張

- https://dev.mysql.com/worklog/task/?id=9831
- https://dev.mysql.com/doc/refman/8.0/en/json.html#json-path-syntax
  - 「MySQL 8.0.2 and later also supports range notation for subsets of JSON arrays …」

JSON オブジェクト値の高速ソート

- https://dev.mysql.com/worklog/task/?id=8741

JSON オブジェクト値のインプレース更新

- https://dev.mysql.com/worklog/task/?id=10570
- https://dev.mysql.com/worklog/task/?id=8963

5.4 リンク集 URL

図 5.1: https://hmatsu47.hatenablog.com/book_mysql80_051

第 6 章

GIS（地理情報システム）の新機能

MySQL 8.0 では、地理情報を扱う GIS 機能が MySQL 5.7 と比較して大きく拡張されました。

- Spatial 関数の追加
- MySQL 5.7 で非推奨になった関数の廃止[*1]
- Geography サポート
- Spatial Data・Spatial Index・Spatial 関数の SRID サポート／地理座標系サポート

地理座標系をサポートしたことで、MySQL 8.0 では地球を回転楕円体として扱うことができるようになり、GIS 機能が利用しやすくなりました。
概要については以下の資料の 1 つ目、もう少し深く知りたい場合は 2 つ目をご確認ください。

- https://www.slideshare.net/yoyamasaki/mysql-80gisfoss4g-2018-hokkaido
- https://www.slideshare.net/sakaik/mysql-gis-clubmysql-4

6.1 GIS 関数

MySQL 8.0 がサポートする GIS 関数は以下の通りです。

【注】

- WKT：Well-Known Text 形式
- WKB：Well-Known Binary 形式
- MBR：Minimum Bounding Rectangle（最小境界矩形または最小外接矩形）

■コラム: GA 後の GIS 関数

MySQL 8.0 がサポートする GIS 関数は GA 後も追加・機能改善が進んでいます。8.0.16 では `ST_Length()` で単位の指定ができるようになりました。

[*1] プレフィクスに `ST_`・`MBR` が付かない GIS 関数

関数名	説明
GeomCollection()	ジオメトリからジオメトリコレクションを構築
GeometryCollection()	ジオメトリからジオメトリコレクションを構築
LineString()	Point 値から LineString を構築
MBRContains()	あるジオメトリの MBR に別のジオメトリの MBR が含まれているか？
MBRCoveredBy()	ある MBR が別の MBR によって覆われているか？
MBRCovers()	ある MBR が別の MBR をカバーするか？
MBRDisjoint()	2 つの形状の MBR が交差していないか？
MBREquals()	2 つの形状の MBR が等しいか？
MBRIntersects()	2 つの形状の MBR が交差するか？
MBROverlaps()	2 つの形状の MBR が重複するか？
MBRTouches()	2 つの形状の MBR が接触するか？
MBRWithin()	あるジオメトリの MBR が別のジオメトリの MBR 内にあるか？
MultiLineString()	LineString 値から MultiLineString を構築
MultiPoint()	Point 値から MultiPoint を構築
MultiPolygon()	Polygon 値から MultiPolygon を構築
Point()	座標から Point を構築
Polygon()	LineString 引数から Polygon を構築
ST_Area()	多角形または多角形領域を返す
ST_AsBinary() ST_AsWKB()	内部ジオメトリ形式から WKB に変換
ST_AsGeoJSON()	ジオメトリから GeoJSON オブジェクトを生成
ST_AsText() ST_AsWKT()	内部ジオメトリ形式から WKT に変換
ST_Buffer()	ジオメトリから指定距離内にある点のジオメトリを返す
ST_Buffer_Strategy()	ST_Buffer() の戦略オプションを生成する
ST_Centroid()	重心を点として返す
ST_Contains()	あるジオメトリが別のジオメトリを含むか？
ST_ConvexHull()	ジオメトリの凸包を返す
ST_Crosses()	あるジオメトリが別のジオメトリと交差するか？
ST_Difference()	2 つのジオメトリの違いを Point Set として返す
ST_Dimension()	ジオメトリの次元
ST_Disjoint()	あるジオメトリが別のジオメトリと交差しないか？
ST_Distance()	あるジオメトリから別のジオメトリまでの距離
ST_Distance_Sphere()	2 つのジオメトリ間の地球上の最小距離
ST_EndPoint()	LineString の終点
ST_Envelope()	ジオメトリの MBR を返す
ST_Equals()	あるジオメトリが別のジオメトリと等しいか？
ST_ExteriorRing()	Polygon の外装リングを返す
ST_GeoHash()	ジオハッシュ値を生成する
ST_GeomCollFromText() ST_GeometryCollectionFromText() ST_GeomCollFromTxt()	WKT からジオメトリコレクションを返す
ST_GeomCollFromWKB() ST_GeometryCollectionFromWKB()	WKB からジオメトリコレクションを返す
ST_GeometryN()	ジオメトリコレクションから N 番目のジオメトリを返す
ST_GeometryType()	ジオメトリタイプの名前を返す

関数名	説明
ST_GeomFromGeoJSON()	GeoJSON オブジェクトからジオメトリを生成する
ST_GeomFromText() ST_GeometryFromText()	WKT からジオメトリを返す
ST_GeomFromWKB() ST_GeometryFromWKB()	WKB からジオメトリを返す
ST_InteriorRingN()	Polygon の N 番目の内部リングを返す
ST_Intersection()	2 つの形状が交差する Point Set を返す
ST_Intersects()	あるジオメトリが別のジオメトリと交差するか？
ST_IsClosed()	ジオメトリが閉じているか？
ST_IsEmpty()	プレースホルダー機能
ST_IsSimple()	形状が単純か？
ST_IsValid()	ジオメトリが有効か？
ST_LatFromGeoHash()	ジオハッシュ値から緯度を返す
ST_Latitude()	ポイントの緯度を返す
ST_Length()	LineString の長さを返す
ST_LineFromText() ST_LineStringFromText()	WKT から LineString を構築
ST_LineFromWKB() ST_LineStringFromWKB()	WKB から LineString を構築
ST_LongFromGeoHash()	ジオハッシュ値から経度を返す
ST_Longitude()	ポイントの経度を返す
ST_MakeEnvelope()	2 点を囲む四角形
ST_MLineFromText() ST_MultiLineStringFromText()	WKT から MultiLineString を構築
ST_MLineFromWKB() ST_MultiLineStringFromWKB()	WKB から MultiLineString を構築
ST_MPointFromText() ST_MultiPointFromText()	WKT から MultiPoint を構築
ST_MPointFromWKB() ST_MultiPointFromWKB()	WKB から MultiPoint を構築
ST_MPolyFromText() ST_MultiPolygonFromText()	WKT から MultiPolygon を構築
ST_MPolyFromWKB() ST_MultiPolygonFromWKB()	WKB から MultiPolygon を構築
ST_NumGeometries()	ジオメトリコレクション内のジオメトリ数を返す
ST_NumInteriorRing() ST_NumInteriorRings()	Polygon の内部リングの数を返す
ST_NumPoints()	LineString のポイント数を返す
ST_Overlaps()	あるジオメトリが別のジオメトリと重なるか？
ST_PointFromGeoHash()	ジオハッシュ値を Point 値に変換
ST_PointFromText()	WKT からポイントを構築
ST_PointFromWKB()	WKB から Point を構築
ST_PointN()	LineString から N 番目の点を返す
ST_PolyFromText() ST_PolygonFromText()	WKT から Polygon を構築
ST_PolyFromWKB() ST_PolygonFromWKB()	WKB から Polygon を構築
ST_Simplify()	単純化された形状を返す
ST_SRID()	ジオメトリの SRID を返す

関数名	説明
ST_StartPoint()	LineString の始点
ST_SwapXY()	X / Y 座標を入れ替えて引数を返す
ST_SymDifference()	2 つのジオメトリの対称差を Point Set として返す
ST_Touches()	あるジオメトリが別のジオメトリに接するか？
ST_Transform()	ジオメトリの座標を変換する
ST_Union()	2 つのジオメトリの和集合を Point Set として返す
ST_Validate()	検証済みのジオメトリを返す
ST_Within()	あるジオメトリが別のジオメトリの中にあるか？
ST_X()	Point の X 座標を返す
ST_Y()	Point の Y 座標を返す

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/spatial-reference-systems.html

ブログ記事等

- https://mysqlserverteam.com/spatial-reference-systems-in-mysql-8-0/
- https://mysqlserverteam.com/geographic-spatial-reference-systems-in-mysql-8-0/
- https://mysqlserverteam.com/projected-spatial-reference-systems-in-mysql-8-0/
- https://mysqlserverteam.com/geography-in-mysql-8-0/
- https://mysqlserverteam.com/geographic-indexes-in-innodb/
  - Geographic R-tree インデックス
- https://qiita.com/advent-calendar/2018/rdbms_gis
  - RDBMS-GIS(MySQL,PostgreSQL など) Advent Calendar 2018（この中に MySQL 8.0 の記事多数）
- http://next4us-ti.hatenablog.com/entry/2019/01/23/100858
  - MySQL 8.0.14 の ST_Distance() 機能強化について
- http://atsuizo.hatenadiary.jp/entry/2018/09/01/161717
  - MySQL Workbench ／結果の図表示について
- 著者ブログ
  - https://qiita.com/hmatsu47/items/97839fd9c3db1d2e9557

実行例

サンプルとして、

- 距離の計測（東京駅～大阪駅）
- ある地点が、複数の地点（政令指定都市の市役所）を結んだ領域の範囲内にあるかどうかの検索[*2]

を行ってみます。

なお、いずれも測地系として WGS84（SRID：4326）を使用しています。MySQL 8.0 の場合、SRID：4326 では地点の座標を「緯度 経度」の順に指定します[*3]。

[*2] この例では地点間を直線で結んでいますが、実際には都道府県・市区町村界などを領域として定義し、検索地点がどこに属するか判定する使い方のほうが一般的です。

[*3] 実行例の緯度・経度は、Geocoding（http://www.geocoding.jp/）で調べたものです。

```
mysql> SELECT ST_Distance(ST_GeomFromText('POINT(35.681236 139.767125)', 4326), ST_GeomFromTex
t('POINT(34.702485 135.495951)', 4326)) AS dist;
※東京駅と大阪駅の間の距離を計測。約 403.8km。
+------------------+
| dist             |
+------------------+
| 403826.6344217672 |
+------------------+
1 row in set (0.12 sec)

mysql> CREATE TABLE geom (id INT PRIMARY KEY AUTO_INCREMENT, t TEXT,g GEOMETRY NOT NULL SRID 4
326);
Query OK, 0 rows affected (0.02 sec)

mysql> SET @g = 'POLYGON((43.06208 141.354361,38.268195 140.869418,37.916124 139.036371,43.062
08 141.354361))';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO geom SET t='札幌・仙台・新潟', g=ST_GeomFromText(@g, 4326);
※札幌市役所～仙台市役所～新潟市役所～札幌市役所の三角形の領域を設定。
Query OK, 1 row affected (0.01 sec)

mysql> SET @g = 'POLYGON((35.861793 139.64551,35.607285 140.106495,35.530807 139.702997,35.443
674 139.637964,35.571257 139.373427,35.861793 139.64551))';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO geom SET t='さいたま・千葉・川崎・横浜・相模原', g=ST_GeomFromText(@g, 4326);
※さいたま市役所～千葉市役所～川崎市役所～横浜市役所～相模原市役所～さいたま市役所の五角形の領域を設定。
Query OK, 1 row affected (0.01 sec)

mysql> SET @g = 'POLYGON((34.975567 138.382677,34.710865 137.726117,35.181438 136.90642,34.975
567 138.382677))';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO geom SET t='静岡・浜松・名古屋', g=ST_GeomFromText(@g, 4326);
※静岡市役所～浜松市役所～名古屋市役所～静岡市役所の三角形の領域を設定。
Query OK, 1 row affected (0.01 sec)

mysql> SET @g = 'POLYGON((35.011564 135.768149,34.573362 135.483048,34.689486 135.195739,34.69
3725 135.502254,35.011564 135.768149))';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO geom SET t='京都・堺・神戸・大阪', g=ST_GeomFromText(@g, 4326);
※京都市役所～堺市役所～神戸市役所～大阪市役所～京都市役所の四角形の領域を設定。
Query OK, 1 row affected (0.00 sec)

mysql> SET @g = 'POLYGON((34.655531 133.919795,32.803216 130.707937,33.590184 130.401689,33.88
3498 130.875177,34.385289 132.455306,34.655531 133.919795))';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO geom SET t='岡山・熊本・福岡・北九州・広島', g=ST_GeomFromText(@g, 4326);
※岡山市役所～熊本市役所～福岡市役所～北九州市役所～広島市役所～岡山市役所の五角形の領域を設定。
Query OK, 1 row affected (0.01 sec)

mysql> SET @p = ST_GeomFromText('POINT(40.82222 140.747352)', 4326);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT id, t FROM geom WHERE ST_Contains(g, @p);
※青森市役所の位置は 1 番目の領域内にある。
+----+--------------------------+
| id | t                        |
+----+--------------------------+
|  1 | 札幌・仙台・新潟         |
+----+--------------------------+
1 row in set (0.00 sec)

mysql> SELECT id, t FROM geom WHERE ST_Within(@p, g);
※ ST_Within() は、結果的に ST_Contains とは引数が逆になる。
+----+--------------------------+
| id | t                        |
+----+--------------------------+
|  1 | 札幌・仙台・新潟         |
+----+--------------------------+
1 row in set (0.00 sec)

mysql> SET @p = ST_GeomFromText('POINT(33.284461 131.490709)', 4326);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT id, t FROM geom WHERE ST_Contains(g, @p);
※別府市役所の位置は 5 番目の領域内にある。
+----+-------------------------------------------------+
| id | t                                               |
+----+-------------------------------------------------+
|  5 | 岡山・熊本・福岡・北九州・広島                  |
+----+-------------------------------------------------+
1 row in set (0.01 sec)

mysql> EXPLAIN SELECT id, t FROM geom WHERE ST_Contains(g, @p);
※フルスキャンになっている。
+----+-------------+-------+------------+------+---------------+------+---------+------+------
+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows
| filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+------+------
+----------+-------------+
|  1 | SIMPLE      | geom  | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    5
|   100.00 | Using where |
+----+-------------+-------+------------+------+---------------+------+---------+------+------
+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
※ R-tree インデックスを作成。
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN SELECT id, t FROM geom WHERE ST_Contains(g, @p);
※ 1 行に絞り込まれた。
+----+-------------+-------+------------+-------+---------------+------+---------+------+-----
-+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key  | key_len | ref  | rows
 | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+------+---------+------+-----
-+----------+-------------+
```

```
|  1 | SIMPLE      | geom  | NULL       | range | g             | g    | 34      | NULL |    1
 |   100.00 | Using where |
+----+-------------+-------+------------+-------+---------------+------+---------+------+-----
-+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql>  SET @p = ST_GeomFromText('POINT(39.701956 141.15433)', 4326);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT id, t FROM geom WHERE ST_Contains(g, @p);
※盛岡市役所の位置は、どの領域の範囲内にもない（1 番目の領域からわずかに東に外れた位置にある）。

Empty set (0.00 sec)

mysql> SELECT id, t FROM geom WHERE MBRContains(g, @p);
※ MBR では外接する矩形を境界に用いて判定するため、「範囲内」にあたる領域が広くなる。
+----+--------------------------+
| id | t                        |
+----+--------------------------+
|  1 | 札幌・仙台・新潟         |
+----+--------------------------+
1 row in set (0.00 sec)
```

6.2　その他の GIS 新機能

CREATE SPATIAL REFERENCE SYSTEM ステートメント

- https://dev.mysql.com/doc/refman/8.0/en/create-spatial-reference-system.html
- https://mysqlserverteam.com/creating-your-own-spatial-reference-systems-in-mysql-8-0/

DROP SPATIAL REFERENCE SYSTEM ステートメント

- https://dev.mysql.com/doc/refman/8.0/en/drop-spatial-reference-system.html

6.3　リンク集 URL

図 6.1: https://hmatsu47.hatenablog.com/book_mysql80_061

第 7 章

レプリケーションの新機能

7.1　バイナリログ／リレーログ暗号化

MySQL 8.0 では InnoDB のテーブルおよび各種ログファイルの透過的暗号化サポートが進んでいますが、バイナリログとリレーログの暗号化にも対応しました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/replication-binlog-encryption.html

ブログ記事等

- https://www.s-style.co.jp/blog/2019/03/3771/
- https://mysqlhighavailability.com/binary-log-encryption-at-rest/
- https://mysqlhighavailability.com/how-to-manually-decrypt-an-encrypted-binary-log-file/
- 著者ブログ
  - https://qiita.com/hmatsu47/items/96980d508f79f9510aa2

7.1.1　実行例

第 8 章 InnoDB とオプティマイザの新機能 8.2 InnoDB「テーブルスペース／ Redo・Undo ログと一般テーブルスペースの暗号化」を参考に、あらかじめキーリング用プラグインの導入を行っておきます。

次に、Master・Slave それぞれのサーバの`/etc/my.cnf` にレプリケーション関連の設定を記述します。

Master 側設定（関連部分のみ）

リスト 7.1: /etc/my.cnf

```
server-id=1
binlog_format=MIXED
# バイナリログ形式を MIXED に変更しているのは暗号化の確認をしやすくするため。実運用では ROW 推奨。
binlog_encryption=ON
binlog_rotate_encryption_master_key_at_startup=ON
```

Slave 側設定（同上）

リスト 7.2: /etc/my.cnf

```
server-id=2
binlog_format=MIXED
binlog_encryption=ON
binlog_rotate_encryption_master_key_at_startup=ON
super_read_only
#skip-slave-start
```

なお、Slave を Master のディスクイメージからコピーして立てた場合、レプリケーション開始時にエラーが発生することがあります。データディレクトリにある auto.cnf の server-uuid が重複していることが原因かもしれません。

その場合、auto.cnf を削除してからサーバを起動すると server-uuid が自動生成され、正しくレプリケーションを開始することができます。

Master 側操作

```
mysql> CREATE DATABASE enc_test;
Query OK, 1 row affected (0.01 sec)

mysql> USE enc_test;
Database changed
mysql> CREATE TABLE enc_test (id int(10) PRIMARY KEY AUTO_INCREMENT, value VARCHAR(100)) ENGIN
E=innodb ENCRYPTION='Y';
※テストテーブルを作成する。ファイルシステム検索で紛らわしくないよう暗号化テーブルで。
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO enc_test SET value='hoge';
※テストデータを挿入する。
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO enc_test SET value='fuga';
Query OK, 1 row affected (0.00 sec)

※ここで OS Shell からファイルシステムに対し hoge・fuga を grep 検索しても引っかからないことが確認できる。

mysql> SHOW BINARY LOGS;
※ Encrypted が Yes になっている。
+---------------+-----------+-----------+
| Log_name      | File_size | Encrypted |
+---------------+-----------+-----------+
| binlog.000001 |      1855 | Yes       |
+---------------+-----------+-----------+
1 row in set (0.00 sec)

mysql> CREATE USER 'repl'@'%' IDENTIFIED BY 'T35+U53r';
※レプリケーション用ユーザを作成する。
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
※レプリケーションスレーブ権限を付与する。
Query OK, 0 rows affected (0.01 sec)
```

Slave 側操作

Master 側と同様に mysql コマンドを root ユーザで操作します。

```
mysql> CHANGE MASTER TO
    ->     MASTER_HOST='【ホスト名】',
    ->     MASTER_USER='repl',
    ->     MASTER_PASSWORD='T35+U53r',
    ->     MASTER_LOG_FILE='binlog.000001',
    ->     MASTER_LOG_POS=4;
Query OK, 0 rows affected, 2 warnings (0.01 sec)
※ warnings の内容については著者ブログを参照。

mysql> START SLAVE;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
 (中略)
                   Last_Errno: 1410
                   Last_Error: Error 'You are not allowed to create a user with GRANT' on quer
y. Default database: ''. Query: 'GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%''
 (中略)
        Seconds_Behind_Master: NULL
Master_SSL_Verify_Server_Cert: No
 (中略)
               Last_SQL_Errno: 1410
               Last_SQL_Error: Error 'You are not allowed to create a user with GRANT' on quer
y. Default database: ''. Query: 'GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%''
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
 (中略)
       Get_master_public_key: 0
1 row in set (0.00 sec)
※先ほど Master でユーザを作成したことが原因。Slave で同じユーザを作成する必要はないのでスキップする。

mysql> SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1;
Query OK, 0 rows affected (0.00 sec)

mysql> START SLAVE;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
 (中略)
              Master_Log_File: binlog.000001
          Read_Master_Log_Pos: 1856
 (中略)
```

```
       Seconds_Behind_Master: 0
（中略）
               Last_IO_Errno: 0
               Last_IO_Error:
              Last_SQL_Errno: 0
              Last_SQL_Error:
（中略）
       Get_master_public_key: 0
1 row in set (0.00 sec)
```

※今度は成功。ここでファイルシステムに対し hoge・fuga を grep 検索しても引っかからないことが確認できる。

7.2　バイナリログ有効期限の指定方法変更

細かい点ですが、原則として日単位ではなく秒単位で有効期限を設定する仕様になったのでご注意ください。なお、MySQL 8.0 の DMR ／ RC 版を試した経験がある方は、GA 前に二度の仕様変更があった点にもご注意ください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html sysvar_binlog_expire_logs_seconds

ブログ記事等

- https://yoku0825.blogspot.com/2018/04/mysql-803-expirelogsdays.html

7.3　InnoDB Cluster

InnoDB Cluster は MySQL 5.7 で導入された MySQL の高可用性ソリューションです。グループレプリケーション・MySQL Router・MySQL Shell の 3 つのコンポーネントで構成されています。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/mysql-innodb-cluster-userguide.html
- https://dev.mysql.com/doc/refman/8.0/en/mysql-innodb-cluster-working-with-cluster.html

ブログ記事等

- https://www.s-style.co.jp/blog/2018/11/2722/
- https://www.s-style.co.jp/blog/2018/11/2890/
- https://www.s-style.co.jp/blog/2018/11/2899/
- https://www.s-style.co.jp/blog/2018/11/2904/
- https://www.s-style.co.jp/blog/2018/12/2962/
- https://www.s-style.co.jp/blog/2018/12/3028/
- https://www.s-style.co.jp/blog/2019/02/3489/

- https://mysqlserverteam.com/mysql-shell-8-0-whats-new/
  - InnoDB Cluster Improvements
- https://mysqlserverteam.com/mysql-innodb-cluster-changing-cluster-topology-modes-live/
  - InnoDB Cluster 用のリモート MySQL サーバの設定と再設定
  - 拡張 Cluster ステータス表示（複製遅延時間を含む）
  - InnoDB Cluster での手動によるプライマリスイッチオーバーとトポロジの再設定
  - より多くのユースケースと環境のための高度なクラスターのカスタマイズ

7.4 グループレプリケーション

グループレプリケーションは、Master サーバの冗長化を目的として MySQL 5.7 から導入された機能です。InnoDB Cluster のベースとなる機能の 1 つです。

詳細は、公式リファレンスマニュアルおよびブログ記事等（注：MySQL 5.7 時点のものです）を確認してください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/group-replication.html

ブログ記事等

- http://mita2db.blogspot.com/2016/12/group-replication-1.html
- http://mita2db.blogspot.com/2017/01/group-replication-2.html
- http://mita2db.blogspot.com/2017/01/group-replication-3.html
- http://mita2db.blogspot.com/2017/01/group-replication-4.html

7.4.1 グループレプリケーションの新機能

オンラインおよびユーザーによるプライマリ切り替え／選出（Election）

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-changing-primary-member.html
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-new-primary.html

オンラインおよびユーザーによるシングルプライマリ／マルチプライマリの切り替え

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-changing-group-mode.html
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-mode.html

サーバがグループから削除されたときにサーバをシャットダウンする

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_exit_state_action
  - 8.0.16 から初期値が変更になっているので注意（8.0.15 まで `ABORT_SERVER`、8.0.16 から `READ_ONLY`）

応答のないメンバーをグループから追放

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_member_expel_timeout

メンバーのグループ追放・コンタクト不能時自動再試行回数

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_autorejoin_tries

プライマリフェイルオーバー時の一貫読み取り

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_consistency

プライマリフェイルオーバー候補の優先順位設定

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_member_weight
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-single-primary-mode.html

メンバーの書き込み許可を自動で OFF（`super_read_only` のチェック）

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-adding-instances.html
  - Tip「When Group Replication starts successfully and the server joins the group …」
- https://dev.mysql.com/doc/refman/8.0/en/start-group-replication.html
- https://dev.mysql.com/doc/refman/8.0/en/stop-group-replication.html

IPv6 のサポート

- https://dev.mysql.com/doc/refman/8.0/en/ipv6-support.html
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-ipv6.html

フロー制御を微調整するためのオプション

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-flow-control.html
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_applier_threshold
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_certifier_threshold
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_hold_percent
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_max_commit_quota
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_member_quota_percent

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_min_quota
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_flow_control_min_recovery_quota

ホワイトリストでのホスト名のサポート

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-ip-address-whitelisting.html

設定可能なメッセージングパイプライン

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-group-write-consensus.html
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-maximum-consensus.html

メッセージ受け渡しのトレース

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_communication_debug_options

メッセージの最大サイズとキャッシュサイズ

- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_communication_max_message_size
- https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html#sysvar_group_replication_message_cache_size

トランザクションセーブポイントのサポート

- https://mysqlserverteam.com/the-complete-list-of-new-features-in-mysql-8-0/
  - Group Replication 1 項目目
  - リンク先のマニュアルに記載なし

Performance Schema 項目の追加

- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-summary-tables.html
- https://dev.mysql.com/doc/refman/8.0/en/threads-table.html
  - threads.INSTRUMENTED
- https://dev.mysql.com/doc/refman/8.0/en/cond-instances-table.html
  - cond_instances.INSTRUMENTED
- https://dev.mysql.com/doc/refman/8.0/en/mutex-instances-table.html
  - mutex_instances.INSTRUMENTED
- https://dev.mysql.com/doc/refman/8.0/en/memory-summary-tables.html
  - memory_summary_global_by_event_name
- https://dev.mysql.com/doc/refman/8.0/en/setup-instruments-table.html
  - setup_instruments（列追加）

同上／グループ全体の認証と Applier 統計のモニタリング

- https://dev.mysql.com/doc/refman/8.0/en/replication-group-member-stats-table.html
- https://dev.mysql.com/doc/refman/8.0/en/replication-group-members-table.html

7.5 MySQL Router

MySQL Router はアプリケーションサーバ〜MySQL サーバ間の透過的なルーティングを提供する軽量なミドルウェアです。前述の通り InnoDB Cluster の主要コンポーネントの 1 つです。

- https://www.mysql.com/jp/products/enterprise/router.html

7.5.1 MySQL Router の新機能

最後に利用したサーバアドレス等の永続化

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysql-router-conf-options.html#option_mysqlrouter_dynamic_config

接続成功時に `max_connect_errors` をリセット

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysql-router-conf-options.html#option_mysqlrouter_max_connect_errors

`mysqlrouter_plugin_info` ツールを追加

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysqlrouter_plugin_info.html

メタデータキャッシュの TTL を 300 秒から 500 ミリ秒に短縮

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysql-router-conf-options.html#option_mysqlrouter_ttl

ルーティングストラテジを追加（`routing_strategy` オプション）

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysql-router-conf-options.html#option_mysqlrouter_routing_strategy

起動オプション`--report-host` を追加

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysqlrouter.html#option_mysqlrouter_report-host

起動オプション`--account-host` を追加

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysqlrouter.html#option_mysqlrouter_account-host

プライマリからセカンダリに降格したサーバーノードへのクライアント接続を解除

- https://dev.mysql.com/doc/mysql-router/8.0/en/mysql-router-conf-options.html#option_mysqlrouter_destinations

MySQL Server のソースツリーの一部として Router を構築

- https://dev.mysql.com/worklog/task/?id=10799

7.6 MySQL Shell

MySQL 5.7 で導入された MySQL Shell も機能向上しています[*1]。

InnoDB Cluster に対する MySQL 8.0 のサポート

- 前述（InnoDB Cluster のセクションを参照)。

セキュアなパスワード管理

- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-pluggable-password-store.html
- https://mysqlserverteam.com/mysql-shell-8-0-12-whats-new/
  - Pluggable Password Store

改善された組み込みヘルプ

- https://mysqlserverteam.com/mysql-shell-8-0-12-whats-new/
  - Centralized Help System

クエリ結果におけるカラムタイプ表示

- https://mysqlserverteam.com/mysql-shell-8-0-14-whats-new/
  -「Ability to show column type information when executing SQL.」

X DevAPI サポートをアップデート

- https://dev.mysql.com/doc/x-devapi-userguide/en/

スクリーンページング

- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-using-pager.html

自動補完

- https://mysqlserverteam.com/mysql-shell-8-0-whats-new/
  - Auto-Completion

[*1] InnoDB Cluster のコンポーネントの 1 つですので、この章で説明します。

カスタマイズ可能なプロンプト

- https://mysqlserverteam.com/mysql-shell-8-0-whats-new/
  - Prompt Themes

コマンド履歴永続化

- https://mysqlserverteam.com/mysql-shell-8-0-whats-new/
  - Command Line History Persistence

シェル API の直接コマンドライン実行

- https://mysqlserverteam.com/mysql-shell-api-command-line-integration-for-devops/

7.7 その他のレプリケーション新機能

チャネルフィルタ毎のマルチソースレプリケーション

- https://dev.mysql.com/doc/refman/8.0/en/replication-rules-channel-based-filters.html
- https://gihyo.jp/dev/serial/01/mysql-road-construction-news/0088
  -「MySQL8.0 では，`performance_schema` の `replication_applier_global_filters` と…」

トランザクション内テンポラリテーブルの GTID サポート

- https://dev.mysql.com/doc/refman/8.0/en/replication-gtids-restrictions.html

JSON 列の部分アップデート対応レプリケーション

- https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_row_value_options

`RESET MASTER TO`

- https://dev.mysql.com/doc/refman/8.0/en/reset-master.html

書き込みセットベースのトランザクション依存関係追跡

- https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_transaction_dependency_tracking
- https://dev.mysql.com/doc/refman/8.0/en/replication-options-slave.html#option_mysqld_slave-parallel-type

受信側スレッドと適用側スレッドの間の競合の削減

- https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-1.html
  -「To avoid potential race conditions, …」

拡張テーブルメタデータのバイナリログ記録

- https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#sysvar_binlog_row_metadata
- https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html#option_mysqlbinlog_print-table-metadata

`GTID_EXECUTED` が空でない場合に設定可能な `GTID_PURGED`

- https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html#sysvar_gtid_purged

空き容量がなくなったときの安全（ノンブロッキング）なレプリケーションモニタリング

- https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-2.html
  - 「Replication: The receiver thread has been improved to no longer block other thread's …」

トランザクション長のバイナリログへの記録

- https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-2.html
- 「Replication: A new transaction length field has been added to the Gtid_log_event …」

各トランザクションのサーババージョンをバイナリログに記録

- https://dev.mysql.com/doc/refman/8.0/en/replication-compatibility.html
  - 「From MySQL 8.0.14, the server version is recorded in the binary log…」
- https://dev.mysql.com/doc/refman/8.0/en/replication-options-master.html#sysvar_original_server_version
- https://dev.mysql.com/doc/refman/8.0/en/replication-options-master.html#sysvar_immediate_server_version

`START SLAVE UNTIL`（マルチスレッドレプリケーションへの対応）

- https://dev.mysql.com/doc/refman/8.0/en/start-slave.html

遅延レプリケーションのマイクロ秒対応

- https://dev.mysql.com/doc/refman/8.0/en/replication-delayed.html
  - 「original_commit_timestamp: the number of microseconds since epoch when …」

`binlog-row-event-max-size` システム変数

- https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html

7.8 リンク集 URL

図 7.1: https://hmatsu47.hatenablog.com/book_mysql80_071

第 8 章

オプティマイザと InnoDB の新機能

8.1　オプティマイザ

MySQL 8.0 では、SQL の実行計画を最適化するオプティマイザが進化しました。

【注】「非公式 MySQL 8.0 オプティマイザガイド」という非常に有用な資料があります。ご確認ください。

- https://yakst.github.io/unofficialmysqlguide-ja/

8.1.1　ヒストグラム

カラム値のヒストグラム統計を使い、インデックスがないカラムでも値の分布から行の絞り込みを可能にする機能です。RDBMS によってはインデックスの一種としてヒストグラムを利用するものがありますが、MySQL 8.0 ではインデックスとは別の機能として提供されます。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/optimizer-statistics.html
- https://dev.mysql.com/doc/refman/8.0/en/analyze-table.html#analyze-table-histogram-statistics-analysis
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_histogram_generation_max_mem_size

ブログ記事等

- https://yakst.com/ja/posts/4873
- http://masato.ushio.org/blog/index.php/2017/09/25/uco-tech_mysql-8-0-rc-histogram-optimizing/
- 著者ブログ
  - https://qiita.com/hmatsu47/items/3cfc6762bca766c5d9a1

8.1.2 メモリとディスクの I/O コスト

MySQL 5.7 までは、データページをメモリ（バッファプール）から読み取る場合もディスクから読み出す場合も同じコストが掛かるものとしてコスト計算を行っていました。MySQL 8.0 では、メモリとディスクのコスト係数を別々に設定してコスト計算を行うことができるようになりました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/cost-model.html
  - メモリとディスクの I/O コスト係数を分離
- https://dev.mysql.com/doc/refman/8.0/en/cost-model.html#cost-model-database
  - コスト係数テーブルの初期値

ブログ記事等

- 著者ブログ
  - https://qiita.com/hmatsu47/items/d53b0471c8f279130114

8.1.3 `FORCE INDEX` 時に不要なインデックスダイブを回避

`FORCE INDEX` を指定した場合のインデックス走査が効率的になりました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/range-optimization.html#equality-range-optimization
  -「In MySQL 8.0, index dive skipping is possible for queries that satisfy all …」

ブログ記事等

- https://mysqlserverteam.com/optimization-to-skip-index-dives-with-force-index/

8.1.4 ヒント句

新しいヒント句が追加されました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-table-level
  - `MERGE`, `NO_MERGE`
- https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-index-level
  - `INDEX_MERGE`, `NO_INDEX_MERGE`

- https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-join-order
  - JOIN_FIXED_ORDER
  - JOIN_ORDER
  - JOIN_PREFIX
  - JOIN_SUFFIX
- https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-set-var
  - SET_VAR

ブログ記事等

- https://yoku0825.blogspot.com/2017/04/mysql-801joinorder.html
- http://variable.jp/2017/09/28/mysql8-0%E3%81%AB%E3%81%8A%E3%81%91%E3%82%8B%E3%83%92%E3%83%B3%E3%83%88%E5%8F%A5%E3%81%AE%E6%8B%A1%E5%BC%B5/

8.1.5 Skip Scan Range Access Method

複合インデックスの 1 番目の列が検索条件に入っていない場合に当該インデックスを利用して検索する仕組みです。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/range-optimization.html#range-access-skip-scan

ブログ記事等

- 著者ブログ
  - https://qiita.com/hmatsu47/items/d83bda0360728d4f585a

■コラム: 暗黙の GROUP BY ソートの廃止

以下のリンクの通り MySQL 5.6 の時点で非推奨とされていましたが、暗黙の GROUP BY ソートは MySQL 8.0 で廃止になっています。

- https://dev.mysql.com/doc/refman/5.6/ja/order-by-optimization.html
  - 注記「MySQL 5.6 における暗黙の GROUP BY ソートへの依存は…」

8.2 InnoDB

MySQL 8.0 では、地味なものが多いですが InnoDB も細かい改良が進んでいます。

8.2.1 新しいロック：NOWAIT ／ SKIP LOCKED

SELECT 〜 FOR UPDATE 等によって行ロックの獲得を試みてすぐに獲得できなかったとき、獲得を待たずに処理を進める機能が追加されました。
詳細は公式リファレンスマニュアルとブログ記事等をご確認ください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/innodb-locking-reads.html#innodb-locking-reads-nowait-skip-locked

ブログ記事等

- https://mysqlserverteam.com/mysql-8-0-1-using-skip-locked-and-nowait-to-handle-hot-rows/
- https://yoku0825.blogspot.com/2018/04/mysql-80-select-for-update-skip-locked.html

8.2.2 ノンロッキング並列読み取り

MySQL 8.0.15 時点では CHECK TABLE・SELECT COUNT(*) など利用可能なケースがかなり限られますが、並列読み取り（パラレルスキャン）に対応しました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/check-table.html#check-table-innodb
  - 「As of MySQL 8.0.14, InnoDB supports parallel clustered index reads, …」

ブログ記事等

- http://atsuizo.hatenadiary.jp/entry/2019/01/23/112608
- http://atsuizo.hatenadiary.jp/entry/2019/01/24/090000
- http://atsuizo.hatenadiary.jp/entry/2019/01/26/090000
- http://atsuizo.hatenadiary.jp/entry/2019/01/28/090000
- http://atsuizo.hatenadiary.jp/entry/2019/01/29/090000
  - グループ化せずに SELECT COUNT(*) を高速化

8.2.3 AUTO_INCREMENT 値の永続化

MySQL 5.7 まではサーバを再起動すると各テーブルの AUTO_INCREMENT 値が「最大の値を持つ行の次の値」[*1]に自動的に設定されていました。これにより、行削除やトランザクションのロールバックなどが原

[*1] Cluster 構成でない通常の状態では最大値 +1 です。

因で AUTO_INCREMENT 列の値に空き番号が存在した場合に番号が巻き戻ることがありましたが、MySQL 8.0 では正しく AUTO_INCREMENT 値を保持するようになりました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/innodb-auto-increment-handling.html#innodb-auto-increment-initialization

ブログ記事等

- https://www.s-style.co.jp/blog/2018/08/2284/
- 著者ブログ
  - https://qiita.com/hmatsu47/items/4429171c1bbaba564774

8.2.4 テーブルスペース／ Redo・Undo ログ／一般テーブルスペース／システムテーブルの暗号化

MySQL 5.7 ではテーブルスペースだけが対象だった透過的暗号化機能が、MySQL 8.0 では

- Redo ログ
- Undo ログ
- 一般テーブルスペース
- バイナリログ／リレーログ（第 7 章参照）
- システムテーブル（mysql スキーマ）

まで対象が増えました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespace-encryption.html
  - 透過的暗号化（TDE）全体の解説
- https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespace-encryption.html#innodb-tablespace-encryption-redo-log
  - Redo ログ暗号化
- https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespace-encryption.html#innodb-tablespace-encryption-undo-log
  - Undo ログ暗号化
- https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespace-encryption.html#innodb-general-tablespace-encryption-enabling-disabling
  - 一般テーブルスペース暗号化
- https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespace-encryption.html#innodb-mysql-tablespace-encryption-enabling-disabling
  - システムテーブル暗号化

ブログ記事等

- https://mysqlserverteam.com/mysql-8-0-13-innodb-transparent-tablespace-encryption-for-general-tablespaces/
- 著者ブログ
  - https://qiita.com/hmatsu47/items/bae53fd0f6d09511732c
  - https://qiita.com/hmatsu47/items/f3519532c134ba0018af

実行例

テーブルスペース／ Redo・Undo ログ／システムテーブル暗号化の実行例です。一般テーブルスペース暗号化についてはブログ記事等の 1 つ目または著者ブログの 2 つ目をご確認ください。バイナリログ／リレーログ暗号化については**第 7 章 レプリケーションの新機能 7.1 バイナリログ／リレーログ暗号化**をご確認ください。

まず、`/etc/my.cnf` に設定を追加します。

リスト 8.1: /etc/my.cnf 追記部分

```
early-plugin-load=keyring_file.so
keyring_file_data=/var/lib/mysql-keyring/keyring
innodb_doublewrite=0
# ダブルライトを無効にしないとダブルライトバッファに平文で書き出されてしまうため。
innodb_redo_log_encrypt=1
innodb_undo_log_encrypt=1
```

次にサーバを再起動します。再起動後、暗号化テーブルを作成してみます。

```
mysql> CREATE TABLE enc_test (id int(10) PRIMARY KEY AUTO_INCREMENT, value VARCHAR(100)) ENGIN
E=innodb ENCRYPTION='Y';
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO enc_test SET value='1234567890ABCDEFGHIJKLMNOPQRSTabcdefghijklmnopqrst12345
67890ABCDEFGHIJKLMNOPQRSTabcdefghijklmnopqrst';
Query OK, 1 row affected (0.01 sec)
※複数行入れておく。

mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE enc_test SET value='ENCRYPTED' WHERE id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
※ここで OS Shell からファイルシステムに対し ENCRYPTED を grep 検索しても見つからないことが確認できる。但しテーブル名の enc_test を grep 検索すると見つかる。

mysql> ALTER TABLESPACE mysql ENCRYPTION = 'Y';
```

```
Query OK, 0 rows affected (0.21 sec)
※ここで OS Shell からファイルシステムに対し enc_test を grep 検索すると見つからなくなったことが確認できる。
```

8.2.5 その他の InnoDB 新機能

CREATE ／ ALTER ／ DROP UNDO TABLESPACE

- https://dev.mysql.com/doc/refman/8.0/en/innodb-undo-tablespaces.html#innodb-add-undo-tablespaces
- https://dev.mysql.com/doc/refman/8.0/en/innodb-undo-tablespaces.html#innodb-drop-undo-tablespaces
- https://mysqlserverteam.com/new-in-mysql-8-0-14-create-undo-tablespace/

適応型のスキャンバッファサイズ調整

- https://dev.mysql.com/worklog/task/?id=7093

ストレージエンジン API におけるサンプリングインターフェース

- https://dev.mysql.com/doc/dev/mysql-server/8.0.14/classhandler.html#a684f7429844b6a5061d5942e6f12b573
- https://dev.mysql.com/doc/dev/mysql-server/8.0.14/classhandler.html#a2e052d45aa09050fdfff6e10c67cbd1e
- https://dev.mysql.com/doc/dev/mysql-server/8.0.14/classhandler.html#a167d4bd2f1c5f353d77ed08dbed4c04a
- https://dev.mysql.com/doc/dev/mysql-server/8.0.14/sql__2handler__8h.html#ae954cfd46ba0b8300368dccf2bebc842

デフォルトのオプティマイザトレースバッファ容量の拡大

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_optimizer_trace_max_mem_size

LIKE 検索時の部分インデックスの適正利用

- https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-11.html
  - 「When using a partial index, the optimizer performed a more expensive table lookup …」
- https://bugs.mysql.com/bug.php?id=74359

INSERT ／ UPDATE ／ REPLACE ／ DELETE に対する EXPLAIN EXTENDED

- https://dev.mysql.com/doc/refman/8.0/en/explain-extended.html
  - 「As of MySQL 8.0.12, extended information is available for SELECT, DELETE, INSERT, …」

ラッチフリーでスケーラブルな Redo ログ

- https://dev.mysql.com/worklog/task/?id=10310
- https://mysqlserverteam.com/mysql-8-0-new-lock-free-scalable-wal-design/

LOB 列の再設計・改良による高速化

- https://dev.mysql.com/worklog/task/?id=8960
- https://mysqlserverteam.com/mysql-8-0-innodb-introduces-lob-index-for-faster-updates/
- https://mysqlserverteam.com/mysql-8-0-mvcc-of-large-objects-in-innodb/
- https://mysqlserverteam.com/mysql-8-0-optimizing-small-partial-update-of-lob-in-innodb/

CATS（新しいロックスケジューラ）

- http://masato.ushio.org/blog/index.php/2018/03/04/uco-tech_mysql-8-0_trx_scheduling_cats/
- https://mysqlserverteam.com/contention-aware-transaction-scheduling-arriving-in-innodb-to-boost-performance/

InnoDB テーブルスペースバージョン管理サポート（アップグレード／ダウングレード用）

- https://dev.mysql.com/worklog/task/?id=5989

シリアライズ辞書情報（SDI）を持つ自己記述型テーブルスペースと管理ツール

- https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_serialized_dictionary_information
- https://dev.mysql.com/doc/refman/8.0/en/ibd2sdi.html
- https://dev.mysql.com/doc/refman/8.0/en/serialized-dictionary-information.html

バッファプールの Mutex 削除

- https://dev.mysql.com/worklog/task/?id=8423

パージの改善

- https://dev.mysql.com/worklog/task/?id=9387

デッドロック検出を自動的に有効化／無効化

- https://dev.mysql.com/worklog/task/?id=9383

オフラインでの DB ポータビリティ提供（.isl ファイル不要化）

- https://dev.mysql.com/worklog/task/?id=8619

より小さなコアファイルを生成するための新設定

- https://mysqlserverteam.com/mysql-8-0-excluding-the-buffer-pool-from-a-core-file/

パーティションテーブルの共有テーブル領域を非推奨に

- https://dev.mysql.com/worklog/task/?id=11571

テンポラリテーブルが占有しているオンラインディスクスペースをオンラインで再利用

- https://dev.mysql.com/worklog/task/?id=11613

XA トランザクションロールバック時の権限チェック

- https://dev.mysql.com/doc/refman/8.0/en/xa.html

8.3 リンク集 URL

図 8.1: https://hmatsu47.hatenablog.com/book_mysql80_081

第 9 章

Information Schema・Performance Schema の変更と新機能

MySQL 8.0 における機能追加や変更、およびデータディクショナリの InnoDB 化に合わせて、Information Schema・Performance Schema にも大幅な変更が加えられました。

9.1 Information Schema

主なものを示します。

9.1.1 全般

ブログ記事等

- http://next4us-ti.hatenablog.com/entry/2018/11/14/175647
  - 廃止されたものと追加されたもの
- https://mysqlserverteam.com/mysql-8-0-improvements-to-information_schema/
- https://mysqlserverteam.com/further-improvements-on-information_schema-in-mysql-8-0-3/

9.1.2 データディクショナリテーブルと INFORMATION_SCHEMA 内テーブルの統合

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/data-dictionary-information-schema.html

9.1.3 新規追加テーブル

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/view-table-usage-table.html
  - `VIEW_TABLE_USAGE`
- https://dev.mysql.com/doc/refman/8.0/en/view-routine-usage-table.html
  - `VIEW_ROUTINE_USAGE`
- https://dev.mysql.com/doc/refman/8.0/en/keywords-table.html
  - `KEYWORDS`

- https://dev.mysql.com/doc/refman/8.0/en/column-statistics-table.html
  - COLUMN_STATISTICS
- https://dev.mysql.com/doc/refman/8.0/en/st-geometry-columns-table.html
  - ST_GEOMETRY_COLUMNS
- https://dev.mysql.com/doc/refman/8.0/en/st-spatial-reference-systems-table.html
  - ST_SPATIAL_REFERENCE_SYSTEMS
- https://dev.mysql.com/doc/refman/8.0/en/st-units-of-measure-table.html
  - ST_UNITS_OF_MEASURE

ブログ記事等

- https://www.s-style.co.jp/blog/2018/08/2270/
  - INNODB_CACHED_INDEXES

9.1.4　その他の Information Schema 変更

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/innodb-tablespaces-table.html
  - INNODB_TABLESPACES の SERVER_VERSION
- https://dev.mysql.com/doc/refman/8.0/en/innodb-undo-tablespaces.html
  - SELECT NAME, SUBSYSTEM, COMMENT FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME LIKE '%truncate%';

9.2　Performance Schema

同様に、主なものを示します。

9.2.1　InnoDB ロック関連テーブル等

InnoDB ロック関連テーブル・ビューの構成が大きく変更されました。詳細は著者ブログの記事を確認してください。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/sys-innodb-lock-waits.html
  - InnoDB ロック関連テーブル

ブログ記事等

- 著者ブログ
  - https://qiita.com/hmatsu47/items/607d176e885f098262e8
  - https://qiita.com/hmatsu47/items/b49bc18d49da5c6029e5

実行例

```
※クライアント 1 で実行（準備）。
mysql> CREATE DATABASE lock_test;
Query OK, 1 row affected (0.02 sec)

mysql> USE lock_test;
Database changed
mysql> CREATE TABLE lock_test (id int(10) PRIMARY KEY AUTO_INCREMENT, value VARCHAR(100)) ENGI
NE=innodb;
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO lock_test SET value='abc';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO lock_test SET value='def';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO lock_test SET value='ghi';
Query OK, 1 row affected (0.00 sec)

mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

※クライアント 2 で実行（準備）。
mysql> USE lock_test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

※クライアント 3 で実行。ロックが生じていないことを確認。
mysql> SELECT * FROM sys.innodb_lock_waits\G
Empty set (0.02 sec)

mysql> SELECT * FROM performance_schema.data_locks\G
Empty set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_lock_waits\G
Empty set (0.00 sec)

※クライアント 1 で実行。UPDATE でロックを発生させる。
mysql> UPDATE lock_test SET value='345' WHERE id>1;
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

※クライアント 3 で実行。ロックが生じていることがわかる。
mysql> SELECT * FROM sys.innodb_lock_waits\G
Empty set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_locks\G
*************************** 1. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578867504:1065:139903459448472
ENGINE_TRANSACTION_ID: 2585
```

```
           THREAD_ID: 46
            EVENT_ID: 23
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: NULL
OBJECT_INSTANCE_BEGIN: 139903459448472
            LOCK_TYPE: TABLE
            LOCK_MODE: IX
          LOCK_STATUS: GRANTED
            LOCK_DATA: NULL
*************************** 2. row ***************************
              ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578867504:8:4:1:139903459445432
ENGINE_TRANSACTION_ID: 2585
            THREAD_ID: 46
             EVENT_ID: 23
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459445432
            LOCK_TYPE: RECORD
            LOCK_MODE: X
          LOCK_STATUS: GRANTED
            LOCK_DATA: supremum pseudo-record
*************************** 3. row ***************************
              ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578867504:8:4:3:139903459445432
ENGINE_TRANSACTION_ID: 2585
            THREAD_ID: 46
             EVENT_ID: 23
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459445432
            LOCK_TYPE: RECORD
            LOCK_MODE: X
          LOCK_STATUS: GRANTED
            LOCK_DATA: 2
*************************** 4. row ***************************
              ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578867504:8:4:4:139903459445432
ENGINE_TRANSACTION_ID: 2585
            THREAD_ID: 46
             EVENT_ID: 23
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459445432
            LOCK_TYPE: RECORD
```

```
           LOCK_MODE: X
         LOCK_STATUS: GRANTED
           LOCK_DATA: 3
4 rows in set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_lock_waits\G
Empty set (0.00 sec)
```

※クライアント 2 で実行。INSERT がロック待ちになる。

```
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO lock_test SET value='ghi';
```

※クライアント 3 で実行。ロック待ちも確認できる。

```
mysql> SELECT * FROM sys.innodb_lock_waits\G
*************************** 1. row ***************************
                wait_started: 2019-05-02 18:35:07
                    wait_age: 00:00:13
               wait_age_secs: 13
                locked_table: `lock_test`.`lock_test`
         locked_table_schema: lock_test
           locked_table_name: lock_test
      locked_table_partition: NULL
   locked_table_subpartition: NULL
               locked_index: PRIMARY
                locked_type: RECORD
             waiting_trx_id: 2595
        waiting_trx_started: 2019-05-02 18:35:07
            waiting_trx_age: 00:00:13
     waiting_trx_rows_locked: 1
   waiting_trx_rows_modified: 0
                 waiting_pid: 9
               waiting_query: INSERT INTO lock_test SET value='ghi'
             waiting_lock_id: 139903578868400:8:4:1:139903459451384
           waiting_lock_mode: X,INSERT_INTENTION
             blocking_trx_id: 2594
                blocking_pid: 8
              blocking_query: NULL
            blocking_lock_id: 139903578867504:8:4:1:139903459445432
          blocking_lock_mode: X
        blocking_trx_started: 2019-05-02 18:34:54
            blocking_trx_age: 00:00:26
    blocking_trx_rows_locked: 3
  blocking_trx_rows_modified: 2
     sql_kill_blocking_query: KILL QUERY 8
sql_kill_blocking_connection: KILL 8
1 row in set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_locks\G
*************************** 1. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578868400:1065:139903459454424
ENGINE_TRANSACTION_ID: 2595
            THREAD_ID: 47
             EVENT_ID: 15
        OBJECT_SCHEMA: lock_test
```

```
         OBJECT_NAME: lock_test
      PARTITION_NAME: NULL
   SUBPARTITION_NAME: NULL
          INDEX_NAME: NULL
OBJECT_INSTANCE_BEGIN: 139903459454424
           LOCK_TYPE: TABLE
           LOCK_MODE: IX
         LOCK_STATUS: GRANTED
           LOCK_DATA: NULL
*************************** 2. row ***************************
              ENGINE: INNODB
      ENGINE_LOCK_ID: 139903578868400:8:4:1:139903459451384
ENGINE_TRANSACTION_ID: 2595
           THREAD_ID: 47
            EVENT_ID: 15
       OBJECT_SCHEMA: lock_test
         OBJECT_NAME: lock_test
      PARTITION_NAME: NULL
   SUBPARTITION_NAME: NULL
          INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459451384
           LOCK_TYPE: RECORD
           LOCK_MODE: X,INSERT_INTENTION
         LOCK_STATUS: WAITING
           LOCK_DATA: supremum pseudo-record
*************************** 3. row ***************************
              ENGINE: INNODB
      ENGINE_LOCK_ID: 139903578867504:1065:139903459448472
ENGINE_TRANSACTION_ID: 2594
           THREAD_ID: 46
            EVENT_ID: 30
       OBJECT_SCHEMA: lock_test
         OBJECT_NAME: lock_test
      PARTITION_NAME: NULL
   SUBPARTITION_NAME: NULL
          INDEX_NAME: NULL
OBJECT_INSTANCE_BEGIN: 139903459448472
           LOCK_TYPE: TABLE
           LOCK_MODE: IX
         LOCK_STATUS: GRANTED
           LOCK_DATA: NULL
*************************** 4. row ***************************
              ENGINE: INNODB
      ENGINE_LOCK_ID: 139903578867504:8:4:1:139903459445432
ENGINE_TRANSACTION_ID: 2594
           THREAD_ID: 46
            EVENT_ID: 30
       OBJECT_SCHEMA: lock_test
         OBJECT_NAME: lock_test
      PARTITION_NAME: NULL
   SUBPARTITION_NAME: NULL
          INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459445432
           LOCK_TYPE: RECORD
           LOCK_MODE: X
         LOCK_STATUS: GRANTED
           LOCK_DATA: supremum pseudo-record
```

```
*************************** 5. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578867504:8:4:3:139903459445432
ENGINE_TRANSACTION_ID: 2594
            THREAD_ID: 46
             EVENT_ID: 30
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459445432
            LOCK_TYPE: RECORD
            LOCK_MODE: X
          LOCK_STATUS: GRANTED
            LOCK_DATA: 2
*************************** 6. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578867504:8:4:4:139903459445432
ENGINE_TRANSACTION_ID: 2594
            THREAD_ID: 46
             EVENT_ID: 30
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459445432
            LOCK_TYPE: RECORD
            LOCK_MODE: X
          LOCK_STATUS: GRANTED
            LOCK_DATA: 3
6 rows in set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_lock_waits\G
*************************** 1. row ***************************
                          ENGINE: INNODB
       REQUESTING_ENGINE_LOCK_ID: 139903578868400:8:4:1:139903459451384
REQUESTING_ENGINE_TRANSACTION_ID: 2595
            REQUESTING_THREAD_ID: 47
             REQUESTING_EVENT_ID: 15
REQUESTING_OBJECT_INSTANCE_BEGIN: 139903459451384
         BLOCKING_ENGINE_LOCK_ID: 139903578867504:8:4:1:139903459445432
  BLOCKING_ENGINE_TRANSACTION_ID: 2594
              BLOCKING_THREAD_ID: 46
               BLOCKING_EVENT_ID: 30
  BLOCKING_OBJECT_INSTANCE_BEGIN: 139903459445432
1 row in set (0.00 sec)

※クライアント 1 で実行。COMMIT する。
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

※クライアント 2 で確認。ロック獲得に成功したので INSERT できた。
Query OK, 1 row affected (24.45 sec)

※クライアント 3 で実行。クライアント 2 が獲得したロックが確認できる。
```

```
mysql> SELECT * FROM sys.innodb_lock_waits\G
Empty set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_locks\G
*************************** 1. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578868400:1065:139903459454424
ENGINE_TRANSACTION_ID: 2595
            THREAD_ID: 47
             EVENT_ID: 15
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
          INDEX_NAME: NULL
OBJECT_INSTANCE_BEGIN: 139903459454424
            LOCK_TYPE: TABLE
            LOCK_MODE: IX
          LOCK_STATUS: GRANTED
            LOCK_DATA: NULL
*************************** 2. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139903578868400:8:4:1:139903459451384
ENGINE_TRANSACTION_ID: 2595
            THREAD_ID: 47
             EVENT_ID: 15
        OBJECT_SCHEMA: lock_test
          OBJECT_NAME: lock_test
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
          INDEX_NAME: PRIMARY
OBJECT_INSTANCE_BEGIN: 139903459451384
            LOCK_TYPE: RECORD
            LOCK_MODE: X,INSERT_INTENTION
          LOCK_STATUS: GRANTED
            LOCK_DATA: supremum pseudo-record
2 rows in set (0.01 sec)

mysql> SELECT * FROM performance_schema.data_lock_waits\G
Empty set (0.00 sec)

※クライアント 2 で実行。ROLLBACK する。
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

※クライアント 3 で実行。ロックが解消された。
mysql> SELECT * FROM sys.innodb_lock_waits\G
Empty set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_locks\G
Empty set (0.00 sec)

mysql> SELECT * FROM performance_schema.data_lock_waits\G
Empty set (0.00 sec)
```

9.2.2　高速化について

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-optimization.html

ブログ記事等

- https://mysqlserverteam.com/mysql-8-0-performance-schema-now-with-indexes/

9.2.3　新規追加テーブル

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/error-summary-tables.html
  - エラー要約テーブル
- https://dev.mysql.com/doc/refman/8.0/en/statement-histogram-summary-tables.html
  - ステートメントヒストグラム要約テーブル
- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-thread-pool-tables.html
  - スレッドプールテーブル（Enterprise 版）
- https://dev.mysql.com/doc/refman/8.0/en/log-status-table.html
  - ログステータステーブル
- https://dev.mysql.com/doc/refman/8.0/en/keyring-keys-table.html
  - `keyring_keys` テーブル

ブログ記事等

- https://yoku0825.blogspot.com/2018/05/mysql-80performanceschemaeventsstatemen.html
  - ステートメントダイジェスト
- http://mita2db.blogspot.com/2018/12/mysql-80-events-histgram.html
  - ヒストグラム
- 著者ブログ
  - https://qiita.com/hmatsu47/items/075d8c4f19f7d75a605b
    - ステートメントダイジェスト
  - https://qiita.com/hmatsu47/items/2e4f7c4a09e4c6d4efe6
    - ヒストグラム

9.2.4　Performance Schema のビルトイン SQL 関数

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-functions.html
  - `FORMAT_BYTES()`
  - `FORMAT_PICO_TIME()`
  - `PS_CURRENT_THREAD_ID()`
  - `PS_THREAD_ID()`

9.2.5　その他の Performance Schema 変更

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-statement-digests.html
  - ステートメントダイジェストに QUERY_SAMPLE_TEXT を追加
- https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-rbr-monitoring.html
  - 行ベースレプリケーションのモニタリング
- https://dev.mysql.com/doc/refman/8.0/en/replication-applier-status-by-worker-table.html
  - APPLYING_TRANSACTION・APPLYING_TRANSACTION_START_APPLY_TIMESTAMP など

9.3　その他の変更と新機能

9.3.1　SHOW ステートメント

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/show-columns.html
  - 「The optional EXTENDED keyword causes the output to include information …」
  - SHOW EXTENDED COLUMNS（隠しカラムの表示）
- https://dev.mysql.com/doc/refman/8.0/en/show-index.html
  - 「MySQL 8.0.13 and higher supports functional key parts …」
  - SHOW INDEX に表示される情報の追加

9.4　リンク集 URL

図 9.1: https://hmatsu47.hatenablog.com/book_mysql80_091

第 10 章

その他の変更と新機能

第 9 章までに触れなかった MySQL 8.0 の変更点と新機能について簡単に紹介しておきます。

10.1 リソースグループ

MySQL サーバのスレッドが使用するリソース（CPU コアなど）に制限を掛ける機能です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/resource-groups.html

ブログ記事等

- https://www.s-style.co.jp/blog/2018/09/2549/
- http://mita2db.blogspot.com/2017/09/mysql-80.html

10.2 DML の新機能

10.2.1 `ORDER BY` 句／ `DICTINCT` 句と `WITH ROLLUP` の併用・`GROUPING()`

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/group-by-modifiers.html
- https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html#function_grouping

ブログ記事等

- https://mysqlserverteam.com/improvements-to-rollup-in-mysql/
- https://yakst.com/ja/posts/4564
- https://yoku0825.blogspot.com/2017/04/mysql-801grouping.html

10.2.2 `LATERAL` 句

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/lateral-derived-tables.html

ブログ記事等

- https://mysqlserverteam.com/support-for-lateral-derived-tables-added-to-mysql-8-0-14/
- http://mita2db.blogspot.com/2019/04/mysql-8-lateralsql.html
- https://tombo2.hatenablog.com/entry/2019/03/21/210806
- 著者ブログ
  - https://qiita.com/hmatsu47/items/040d65d118d0ecec6381

10.2.3 派生（Derived）テーブルからの外部テーブル参照

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/derived-tables.html
  -「Prior to MySQL 8.0.14, a derived table cannot contain outer references. …」

10.3 関数の変更と新機能

10.3.1 正規表現関数

利用ライブラリが ICU（International Components for Unicode）に変わるとともに、新しい正規表現関数が追加されました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/regexp.html
  - `REGEXP_INSTR`
  - `REGEXP_LIKE`
  - `REGEXP_REPLACE`
  - `REGEXP_SUBSTR`

ブログ記事等

- https://www.s-style.co.jp/blog/2018/09/2519/
- https://yoku0825.blogspot.com/2018/01/mysql-804mysql.html
- https://yoku0825.blogspot.com/2018/02/mysql-804regexpsubstr-regexpinstr.html
- https://yoku0825.blogspot.com/2018/04/vs-mysql-80.html
  - 危険な正規表現

10.3.2 `STATEMENT_DIGEST()` ／ `STATEMENT_DIGEST_TEXT()`

SQL ステートメントの正規化（Normalize）を行う関数です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html#function_statement-digest

- https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html#function_statement-digest-text

ブログ記事等

- https://yoku0825.blogspot.com/2018/04/mysql-80statementdigestsql.html
- 著者ブログ
  - https://hmatsu47.hatenablog.com/entry/2018/04/19/230021

10.3.3 その他の関数

UUID 関数

- https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html
  - `UUID_TO_BIN`・`BIN_TO_UUID`・`IS_UUID`

`BLOB` 列に対するビット処理

- https://dev.mysql.com/doc/refman/8.0/en/bit-functions.html

10.4 その他各種新機能

10.4.1 Query Rewrite プラグイン

Query Rewrite プラグインの書き換え対応（`INSERT`・`UPDATE`・`DELETE`）

- https://dev.mysql.com/doc/refman/8.0/en/rewriter-query-rewrite-plugin.html
- 著者ブログ
  - https://qiita.com/hmatsu47/items/a43db9fb8c0504f15a79

10.4.2 新しいメモリ内テンポラリテーブルストレージエンジン

- https://dev.mysql.com/doc/refman/8.0/en/internal-temporary-tables.html
  - 「Presence of a BLOB or TEXT column in the table. However, the TempTable …」
- http://nippondanji.blogspot.com/2018/06/temptable.html
- https://mysqlserverteam.com/mysql-8-0-support-for-blobs-in-temptable-engine/

10.4.3 エラーロギング

新しいエラーロギングインフラストラクチャ／エラーロギングの改善

- https://dev.mysql.com/doc/refman/8.0/en/error-log-component-configuration.html
- https://www.s-style.co.jp/blog/2018/07/2061/
- http://variable.jp/2018/03/14/mysql8-0-%E3%82%A8%E3%83%A9%E3%83%BC%E3%83%AD%E3%82%B0%E3%81%AE%E8%A8%AD%E5%AE%9A/
- https://yoku0825.blogspot.com/2018/01/mysql-804_25.html

- https://dev.mysql.com/doc/refman/8.0/en/error-log.html
- https://dev.mysql.com/doc/refman/8.0/en/error-log-format.html
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_log_error_verbosity
  - デフォルト変更：`log_error_verbosity=2`
- https://dev.mysql.com/doc/refman/8.0/en/error-log-filtering.html
  - エラーログのフィルタリング
- https://dev.mysql.com/doc/refman/8.0/en/error-log-json.html
  - JSON 形式のエラーログ
- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_log_error_suppression_list
  - `WARNINGS`・`NOTE` のエラーログを抑制

サーバエラーメッセージ

- https://dev.mysql.com/doc/refman/8.0/en/server-error-reference.html

10.4.4 ログ関連（エラーログ以外）

syslog・eventlog 関連のシステム変数をコンポーネント変数に指定

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_syseventlog.facility

スロークエリログへの `log-slow-extra` の追加

- https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_log_slow_extra

監査ログフィルタ：ルールベースの条件でクエリを中断（Enterprise 版）

- https://dev.mysql.com/doc/refman/8.0/en/audit-log-filtering.html
  -「Filter rules have the capability of blocking (aborting) execution of …」

監査ログの JSON 形式化・圧縮・暗号化（5.7.21 と同様／ Enterprise 版）

- https://dev.mysql.com/doc/refman/8.0/en/audit-log-file-formats.html

監査ログにデータを挿入するための SQL 関数

- https://dev.mysql.com/doc/refman/8.0/en/audit-api-message-emit.html

データマスキング機能（5.7.24 と同様／ Enterprise）

- https://dev.mysql.com/doc/refman/8.0/en/data-masking.html

10.4.5 その他の変更と新機能

クエリキャッシュの廃止

- https://yakst.com/ja/posts/4612

オンディスクテンポラリテーブルストレージエンジン

- https://dev.mysql.com/doc/refman/8.0/en/internal-temporary-tables.html
  - InnoDB のみ利用可能に（MyISAM 廃止）

新しいバックアップロック

- https://dev.mysql.com/doc/refman/8.0/en/lock-instance-for-backup.html

サーバ側キーリング移行ツール

- https://dev.mysql.com/doc/refman/8.0/en/keyring-key-migration.html

AWS KMS 用のキーリングプラグイン（5.7.19 と同様）

- https://dev.mysql.com/doc/refman/8.0/en/keyring-aws-plugin.html

復旧／切り離された準備済み XA トランザクションの MDL ロック有効化

- https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-13.html#mysqld-8-0-13-xa

外部キーのためのメタデータロックのサポート

- https://dev.mysql.com/doc/refman/8.0/en/lock-tables.html
  - 「If you lock a table explicitly with LOCK TABLES, any tables related by …」

SSL チェックを効率化するための`--ssl-mode` クライアントオプション

- https://dev.mysql.com/doc/refman/8.0/en/encrypted-connection-options.html#option_general_ssl-mode

サービスレジストリとコンポーネントインフラストラクチャ

- https://dev.mysql.com/doc/refman/8.0/en/server-components.html

レプリケーションストリームを読み取るための C API

- https://dev.mysql.com/doc/refman/8.0/en/c-api-binary-log-function-overview.html

非同期 C API

- https://dev.mysql.com/doc/refman/8.0/en/c-api-asynchronous-interface.html
  – mysql プロトコルを利用した非同期 API

UDF 自動登録コンポーネントのための UDF 登録サービス

- https://dev.mysql.com/doc/dev/mysql-server/latest/structs___mysql___udf___registration.html

MySQL サーバー文字列コンポーネントサービス

- https://dev.mysql.com/doc/dev/mysql-server/latest/mysql___string__8h.html

結果セットのメタデータ転送オプション

- https://dev.mysql.com/doc/refman/8.0/en/c-api-optional-metadata.html

コンポーネント用のステータス変数サービス

- https://dev.mysql.com/doc/dev/mysql-server/latest/structs___mysql___status___variable___registration.html

プラグインがプリペアドステートメントを使えるように

- https://dev.mysql.com/worklog/task/?id=8413

ソートバッファの動的割り当て

- https://dev.mysql.com/doc/refman/8.0/en/order-by-optimization.html#order-by-filesort

`NO PAD` 照合順序（COLLATION）用の可変長ソートキー

- https://dev.mysql.com/worklog/task/?id=9554

ソースコードの改善

- https://mysqlserverteam.com/mysql-8-0-source-code-improvements/

10.5 リンク集 URL

図 10.1: https://hmatsu47.hatenablog.com/book_mysql80_101

おわりに

The complete list of new features in MySQL 8.0（MySQL Server Blog）

- https://mysqlserverteam.com/the-complete-list-of-new-features-in-mysql-8-0/

この記事をもとに MySQL 8.0.15 までの変更と新機能を把握し、公式リファレンスマニュアルで確認した MySQL 8.0.16 の変更点を加えて本書を完成させました。

今後もバージョンアップに合わせて内容を加筆修正していく予定です。

なお、本文中では特に触れませんでしたが、以下のサイトに有用な情報が多数掲載されていますので、ご確認ください。

本家 MySQL.com の資料ダウンロードサイト

- https://www.mysql.com/jp/news-and-events/seminar/downloads.html

スマートスタイル TECH Blog

- https://www.s-style.co.jp/blog/tag/mysql8-0/

MySQL 道普請便り

- http://gihyo.jp/dev/serial/01/mysql-road-construction-news

漢のコンピュータ道[*1]

- http://nippondanji.blogspot.com/

日々の覚書

- https://yoku0825.blogspot.com/

リンク集 URL

図 2: https://hmatsu47.hatenablog.com/book_mysql80_Z01

[*1] 奥野幹也さんが MySQL 8.0 についても書籍を出されるという未確認情報もあります。期待！

索引

MySQL 8.0 の薄い本

2019 年 4 月 13 日　初版第 1 刷　発行
2019 年 5 月 2 日　第 2 版第 1 刷　発行
著　者　hmatsu47