MySQL 8.0 の薄い本

hmatsu47　著

2021-02-08 MySQL 8.0.23 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 2021 年 2 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。
    - 8.0.21 以降については、主な変更点を「MySQL 8.0.21 以降の主な変更点」に記載しています。
  - 本書では、MySQL NDB Cluster 8.0 の機能は原則として紹介しません。
- 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。

[*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。

電子版 PDF・最新版ダウンロード URL

MySQL 8.0.20 対応版より、印刷版は電子版 PDF の要約版（電子版 PDF から参考ブログ記事等の URL を省いたもの）になりました。ブログ記事等の URL は電子版 PDF をダウンロードしてご利用ください。

図 1: https://github.com/hmatsu47/mysql80_no_usui_hon/raw/master/PDF_ebook/book_ebook.pdf

実行例・サンプル

Linux コマンドライン・SQL 実行例がテキストファイルで入っています。

図 2: https://github.com/hmatsu47/mysql80_no_usui_hon/tree/master/examples

MySQL 8.0.21 以降の主な変更点

8.0.21（リリース日：2020-07-13）

- 管理用クライアント専用のネットワーク設定が可能に（一般クライアント設定と分離）（P.21）
- `CREATE USER`・`ALTER USER` で JSON 形式のユーザコメントが登録可能に（P.25）
- `JSON_VALUE()` 関数（P.47）
- バイナリログのチェックサムをサポート（P.67）
- 可用性向上のために 2 つのシステム変数のデフォルト値を変更（P.67）
- 論理ダンプ・リストアツール（P.69）
- オプティマイザスイッチ `subquery_to_derived`・`prefer_ordering_index` の追加（P.77）
- `UPDATE`・`DELETE` でセミジョイン（準結合）・マテリアライズ（実体化）最適化をサポート（P.81）
- Redo ログの無効化が可能に（P.90）
- Undo テーブルスペースの処理性能向上と安定化・ACID Undo DDL のサポート（P.90）
- テーブルスペースのパス検証の無効化が可能に（P.90）
- KEY パーティショニングでカラムインデックスプレフィックスを使ったときに正しく警告・エラーを出力するようになった（P.105）
- 文字列型と数値型・時間型の型変換（キャスト）がどのように行われたかを `EXPLAIN ANALYZE`・`EXPLAIN FORMAT=JSON`・`EXPLAIN FORMAT=TREE` で可視化（P.105）

8.0.22（リリース日：2020-10-19）

- 権限テーブルのノンロッキング読み取りが可能に（P.25）
- `CREATE`・`DROP`・`RENAME USER` で存在しない `DEFINER` に対するチェックを厳格化（P.25）
- `ALTER DATABASE` で読み取り専用オプションをサポート（P.36）
- MySQL Router ／ HTTP サーバプラグインと REST API（8.0.22 でデフォルト有効に）（P.68）
- 論理ダンプ・リストアツールの拡充（P.69）
- 新しい非同期レプリケーション接続フェイルオーバーメカニズム（P.70）
- 「ホワイトリスト」「スレーブ」（用語）の非推奨化による変更（P.70）
- オプティマイザスイッチ `hypergraph_optimizer` の追加（デバッグビルドのみ／ P.77）
- Derived Condition Pushdown 最適化（P.83）
- `PREPARE` の実行アルゴリズムが変更（`ORDER BY ?`での列番号指定などが無視されるように／ P.85）
- `SELECT ～ FOR SHARE` が `SELECT` 権限のみで実行可能に（P.90）
- Linux 環境においてテーブルスペース配置を効率化（`innodb_extend_and_initialize` ／ P.90）
- 並列度の低いシステムで dedicated log writer threads の無効化が可能に（`innodb_log_writer_threads` ／ P.90）

- Information Schema に SCHEMATA_EXTENSIONS テーブルを追加（P.93）
- Information Schema の TABLESPACES テーブルが非推奨に（P.94）
- Performance Schema にエラーログテーブルを追加（P.100）
- SHOW PROCESSLIST の性能改善（Performance Schema.processlist テーブルを利用／ P.101）
- 括弧付きクエリ式のサポート（P.104）
- CAST() 関数または CONVERT() 関数による YEAR 型へのキャスト（P.104）
- CAST(value AT TIME ZONE specifier AS DATETIME) による TIMESTAMP 列値のタイムゾーン変換（P.104）
- 監査プラグインで audit_log_read() を使用したログ読み込み操作を改善（Enterprise 版／ P.105）
- InnoDB memcached プラグインが非推奨に（P.105）
- Oracle Cloud Infrastructure（OCI）用のキーリングプラグイン（P.105）
- C API で mysql_real_connect_dns_srv() をサポート（P.105）
- ネットワーク名前空間指定子のサポート（P.105）

8.0.23（リリース日：2021-01-18）

- RELOAD 権限の追加（FLUSH 処理だけを可能に）（P.25）
- ユーザ権限の IP アドレス照合ルール（照合順）の変更（P.25）
- SASL LDAP 認証プラグインが SCRAM-SHA-256 をサポート（Enterprise 版）（P.25）
- 不可視カラム（P.36）
- FLUSH HOSTS が非推奨に（P.37）
- GIS 関数 ST_HausdorffDistance()・ST_FrechetDistance() を追加（P.55）
- 非同期接続フェイルオーバ機能がグループレプリケーション・トポロジをサポート（P.67）
- AdminAPI のクラスタ診断機能強化（P.69）
- Parallel Table Import Utility が複数データファイルからのインポートをサポート（P.69）
- マルチスレッドレプリカ（スレーブ）レプリケーションにおけるデッドロック検出機構の改善（P.70）
- GTID を使用しないソースから GTID を使用するレプリカへのレプリケーションが可能に（P.70）
- master_info_repository と relay_log_info_repository が非推奨に（P.70）
- 「マスター」（用語）の非推奨化による変更（P.70）
- Hash Join 高速化（P.79）
- ダブルライトファイル暗号化（P.89）
- AUTOEXTEND_SIZE オプション（P.90）
- テーブルスペース DROP・TRUNCATE の高速化（P.90）
- mysql_bind_param() C API 関数によるクエリ属性の定義（P.105）
- サーバ変数 temptable_max_mmap 追加（P.105）
- ユーザ定義関数 gen_blacklist() を gen_blocklist() に変更（Enterprise 版／ P.105）
- MySQL Enterprise Firewall にグループプロファイル機能を追加（Enterprise 版／ P.105）

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 8 に新規インストールする場合の例を示しておきます。

【注】SELinux および firewalld などの設定は適切に行っておきます。

```
[root@mysql80cent8 ~]# wget https://dev.mysql.com/get/mysql80-community-release-el8-1.noarch.r
pm
※ダウンロードするバージョン（el8-1）はその時点のものを指定
--2021-02-01 22:15:45--  https://dev.mysql.com/get/mysql80-community-release-el8-1.noarch.rpm
（中略）
2021-02-01 22:15:46 (15.2 MB/s) - ‘mysql80-community-release-el8-1.noarch.rpm’ saved [30388/3
0388]

[root@mysql80cent8 ~]# dnf localinstall mysql80-community-release-el8-1.noarch.rpm
（中略）
Dependencies resolved.
================================================================================
 Package                       Arch        Version     Repository         Size
================================================================================
Installing:
 mysql80-community-release     noarch      el8-1       @commandline       30 k

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

```
Total size: 30 k
Installed size: 29 k
Is this ok [y/N]: y
Downloading Packages:
（中略）
Installed:
  mysql80-community-release-el8-1.noarch

Complete!
[root@mysql80cent8 ~]# dnf module disable mysql
※ CentOS 8 標準の MySQL 8.0 のパッケージを無効化する
（中略）
Dependencies resolved.
================================================================================
 Package           Architecture     Version             Repository         Size
================================================================================
Disabling modules:
 mysql

Transaction Summary
================================================================================

Is this ok [y/N]: y
Complete!
[root@mysql80cent8 ~]# dnf install mysql-community-server
Last metadata expiration check: 0:00:26 ago on Mon 01 Feb 2021 10:16:22 PM JST.
Dependencies resolved.
================================================================================
 Package                         Arch    Version       Repository          Size
================================================================================
Installing:
 mysql-community-server          x86_64  8.0.23-1.el8  mysql80-community   53 M
Installing dependencies:
 mysql-community-client          x86_64  8.0.23-1.el8  mysql80-community   12 M
 mysql-community-client-plugins  x86_64  8.0.23-1.el8  mysql80-community  108 k
 mysql-community-common          x86_64  8.0.23-1.el8  mysql80-community  624 k
 mysql-community-libs            x86_64  8.0.23-1.el8  mysql80-community  1.4 M

Transaction Summary
================================================================================
Install  5 Packages

Total download size: 67 M
Installed size: 319 M
Is this ok [y/N]: y
Downloading Packages:
（中略）
Importing GPG key 0x5072E1F5:
 Userid     : "MySQL Release Engineering <mysql-build@oss.oracle.com>"
 Fingerprint: A4A9 4068 76FC BD3C 4567 70C8 8C71 8D3B 5072 E1F5
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
Is this ok [y/N]: y
Key imported successfully
（中略）
Installed:
  mysql-community-client-8.0.23-1.el8.x86_64
  mysql-community-client-plugins-8.0.23-1.el8.x86_64
```

```
  mysql-community-common-8.0.23-1.el8.x86_64
  mysql-community-libs-8.0.23-1.el8.x86_64
  mysql-community-server-8.0.23-1.el8.x86_64

Complete!
[root@mysql80cent8 ~]# systemctl enable mysqld.service
※自動起動 ON
[root@mysql80cent8 ~]# systemctl start mysqld.service
※起動
[root@mysql80cent8 ~]# ps aux | fgrep mysqld
mysql       2025 20.0  9.8 1764528 364860 ?      Ssl  22:17   0:00 /usr/sbin/mysqld
root        2069  0.0  0.0  12108  1084 pts/0    S+   22:17   0:00 grep -F --color=auto mysql
d
[root@mysql80cent8 ~]# fgrep assword /var/log/mysqld.log
※ログからサーバ初期パスワードを確認
2021-02-01T13:17:44.417951Z 6 [Note] [MY-010454] [Server] A temporary password is generated
for root@localhost: ,+Eucy2:Nl?.
```

mysql_secure_installation も使えます。

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

```
[root@mysql80cent8 ~]# 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.
（中略）
Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.
（中略）
Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.
（中略）
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.
（中略）
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

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

■コラム: `Windows` 環境におけるアップグレード

こちらが参考になります。

- https://lefred.be/content/upgrading-from-mysql-5-7-to-8-0-on-windows/

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】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

なお、アップグレードインストール後、`mysql_upgrade_info`ファイルの所有者・アクセス権が原因でサーバ起動に失敗することがあります。

- https://blog.pinkumohikan.com/entry/could-not-start-mysql8-after-version-up
- https://mgng.mugbum.info/1542

【注】MySQL 8.0 ではダウングレードがサポートされなくなっています。

- https://mita2db.hateblo.jp/entry/MySQL_8.0_%E3%81%AF%E3%83%80%E3%82%A6%E3%83%B3%E3%82%B0%E3%83%AC%E3%83%BC%E3%83%89%E3%81%A7%E3%81%8D%E3%81%AA%E3%81%84

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。

- 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。

[*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。

- 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

実行例（CentOS 7 上）

```
[root@mysql57to80 ~]# 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
Loaded plugins: fastestmirror
（中略）
username =

[root@mysql57to80 ~]# yum-config-manager --enable mysql80-community
※ MySQL 8.0 のリポジトリを有効にする
（省略）
[root@mysql57to80 ~]# yum install mysql-shell
※ MySQL Shell 8.0 をインストールする
Loaded plugins: fastestmirror
（中略）
Dependencies Resolved
```

[*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。

```
================================================================================
 Package         Arch       Version             Repository                 Size
================================================================================
Installing:
 mysql-shell     x86_64     8.0.23-1.el7        mysql-tools-community      32 M

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

Total download size: 32 M
Installed size: 142 M
Is this ok [y/d/N]: y
Downloading packages:
（中略）
Installed:
  mysql-shell.x86_64 0:8.0.23-1.el7

Complete!
[root@mysql57to80 ~]# 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 8.0.23
（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 5.7.33 - MySQL
Community Server (GPL), will now be checked for compatibility issues for
upgrade to MySQL 8.0.23...

1) Usage of old temporal type
  No issues found
（中略）
21) New default authentication plugin considerations
（中略）
Errors:   0
Warnings: 1
Notices:  1

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 のチェック項目は、8.0.15 時点で 15 項目だったのが 8.0.23 時点では 21 項目になりました。また、8.0.16 から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

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

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://mysql-params.tmtms.net/

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-system-variables.html#sysvar_event_scheduler

上限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_max_prepared_stmt_count

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
- 管理専用ポートの追加
- 管理用クライアント専用のネットワーク設定が可能に（一般クライアント設定と分離）
- サーバに `mysqld_safe` 機能を追加
- TLS 1.3 サポート
- サーバステータス変数の追加・廃止

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

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

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

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

加えて、8.0.17 から照合順序 `utf8mb4_0900_bin` がサポートされ、`utf8mb4_bin` と比べてソートが高速化されています[*4]。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/charset-unicode-sets.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

[*4] 但し、条件によっては遅くなることもあるようです。詳細は著者ブログを参照してください。

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

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                              |
（中略）
+------------------+-----------+-----------------------+-------------------------------------
-----------------------------------+
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 |
（中略）
+------------------+-----------+-----------------------+-------------------------------------
-----------------------------------+
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 |
（中略）
+------------------+-----------+-----------------------+-------------------------------------
-----------------------------------+
6 rows in set (0.00 sec)
```

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

2.2.2　ランダムパスワードの設定をサポート

8.0.18 から、ユーザ作成・変更およびパスワード変更時にランダムパスワードの設定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/password-management.html#random-password-generation

2.2.3　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

- ログイン試行回数（FAILED_LOGIN_ATTEMPTS）とパスワードロック時間（PASSWORD_LOCK_TIME）の追加
- SUPER 権限を動的権限に分割
- 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
- ストアドルーチンの定義や属性にアクセスするための SHOW_ROUTINE 権限
- RELOAD 権限の追加（FLUSH 処理だけを可能に）
- GRANT ステートメントによるユーザ作成の廃止
- GRANT TABLE のホスト名が 255 文字まで指定可能に
- ユーザ権限の IP アドレス照合ルール（照合順）の変更
- データベースオブジェクトに対する部分的な権限の取り消し（REVOKE）
- ALTER USER ／ SET PASSWORD 時に変更前パスワードの入力を要求
- CREATE USER・ALTER USER で JSON 形式のユーザコメントが登録可能に
- print_identified_with_as_hex システム変数
- セキュアセッション変数の設定（MYSQL_SESSION_ADMIN 権限）
- --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
- ACL ステートメントをアトミックにする
- ログイン失敗時に認証を遅延させる
- LDAP 認証プラグインに関する機能追加（Enterprise 版）
- SASL LDAP 認証プラグインが SCRAM-SHA-256 をサポート（Enterprise 版）
- 権限テーブルのノンロッキング読み取りが可能に
- CREATE・DROP・RENAME USER で存在しない DEFINER に対するチェックを厳格化

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。

なお、8.0.18 からは yaSSL・wolfSSL のサポートは廃止され、OpenSSL のみサポートしています。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/encrypted-connections.html
- https://dev.mysql.com/doc/refman/8.0/en/fips-mode.html

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

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;
（中略）
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)
```

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

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
```

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-type-defaults-explicit

[*1] テーブルの設計情報など

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`

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='use_invisible_indexes=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

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

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。

[*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。

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

3.1.7 CHECK 制約

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/information-schema-table-constraints-table.html
  - INFORMATION_SCHEMA.TABLE_CONSTRAINTS テーブル
    - CONSTRAINT_TYPE 列
- https://dev.mysql.com/doc/refman/8.0/en/alter-table.html#alter-table-foreign-key
  -「As of MySQL 8.0.19, …」
    - ALTER TABLE での変更・削除をサポート

実行例

```
mysql> CREATE TABLE t1 (
    ->   id INT PRIMARY KEY AUTO_INCREMENT,
    ->   prefecture VARCHAR(4) NOT NULL,
    ->   city_town_village VARCHAR(10),
    ->   ward VARCHAR(10),
    ->   CONSTRAINT ctv_ward_blank CHECK (
    ->     (prefecture = '東京都' AND city_town_village IS NULL AND ward IS NOT NULL) OR
    ->     (prefecture = '東京都' AND city_town_village IS NOT NULL AND ward IS NULL) OR
    ->     (prefecture <> '東京都' AND city_town_village IS NOT NULL)));
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO t1 SET prefecture = '東京都', ward = '千代田区';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t1 SET prefecture = '東京都', city_town_village = '八王子市';
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO t1 SET prefecture = '愛知県', city_town_village = '豊田市';
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO t1 SET prefecture = '愛知県', city_town_village = '名古屋市', ward = '中
区';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t1 SET prefecture = '東京都', city_town_village = '町田市', ward = '多摩
区';
```

[*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。

```
ERROR 3819 (HY000): Check constraint 'ctv_ward_blank' is violated.
※エラーの原因は「町田は神奈川」ではなく、東京都なのに市町村と区が同時に指定されたから。
mysql> INSERT INTO t1 SET prefecture = '東京都';
ERROR 3819 (HY000): Check constraint 'ctv_ward_blank' is violated.
※東京都では市町村または区のいずれかが指定されていないとエラーになる。
mysql> INSERT INTO t1 SET prefecture = '愛知県';
ERROR 3819 (HY000): Check constraint 'ctv_ward_blank' is violated.
※東京都以外の場合は市町村が指定されていないとエラーになる。
mysql> SELECT * FROM t1 ORDER BY id;
+----+------------+-------------------+--------------+
| id | prefecture | city_town_village | ward         |
+----+------------+-------------------+--------------+
|  1 | 東京都     | NULL              | 千代田区     |
|  2 | 東京都     | 八王子市          | NULL         |
|  3 | 愛知県     | 豊田市            | NULL         |
|  4 | 愛知県     | 名古屋市          | 中区         |
+----+------------+-------------------+--------------+
4 rows in set (0.00 sec)
```

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
- インプレース処理でのキャラクタセット変換
- ADD DATAFILE を伴わない CREATE TABLESPACE
- LOCK TABLES を伴う RENAME TABLE
- ALTER DATABASE で読み取り専用オプションをサポート
- The ddl_rewriter Plugin
- 不可視カラム

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

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

3.2.3 その他の管理用 SQL 変更点

- FLUSH HOSTS が非推奨に

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

実行例

著者ブログ（https://qiita.com/hmatsu47/items/01211556089b19913d05）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。
```

■コラム: 再帰 CTE の LIMIT 句対応

8.0.19 から、再帰 CTE で LIMIT 句が使えるようになりました。

- https://dev.mysql.com/doc/refman/8.0/en/with.html#common-table-expressions-recursive-examples
  - 「Prior to MySQL 8.0.19, …」
- https://dev.mysql.com/doc/refman/8.0/en/with.html#common-table-expressions-recursion-limits
  - 「Beginning with MySQL 8.0.19, …」

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

関数名	説明
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

実行例

著者ブログ（https://qiita.com/hmatsu47/items/7976e81100604f8984d2）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)
```

第 5 章

JSON とドキュメントストアの新機能

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 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
JSON_SCHEMA_VALID()	JSON スキーマに対する JSON 文書の検証
JSON_SCHEMA_VALIDATION_REPORT()	検証に関するレポートを JSON 形式で提供
MEMBER OF()	検索用オペレータ／値の全部が含まれるか？
JSON_OVERLAPS()	検索用関数／値の一部が含まれるか？
JSON_VALUE()	JSON ドキュメントから値を取り出す

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

実行例

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

```
mysql> CREATE TABLE agg_test (id INT PRIMARY KEY AUTO_INCREMENT, j_key VARCHAR(20) UNIQUE NOT
```

[*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）。

```
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)
```

続いて、JSON_VALUE() の利用例です。関数インデックスとして使ってみます。

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

mysql> USE jsontest;
Database changed
mysql> CREATE TABLE t1(
    ->   j JSON,
    ->   INDEX i1 ( (JSON_VALUE(j, '$.id' RETURNING UNSIGNED)) )
    -> );
Query OK, 0 row affected (0.02 sec)

※ここでデータを投入。

mysql> SELECT * FROM t1;
+----------------------------------+
| j                                |
+----------------------------------+
| {"id": 100, "val": [1, 2, 3]}    |
| {"id": 101, "val": [4, 5, 6, 7]} |
| {"id": 110, "val": [8, 9, 0]}    |
| {"id": 120, "val": [1, 2]}       |
| {"id": 122, "val": 3}            |
| {"id": 130, "val": [4, 5]}       |
| {"id": 140, "val": [6, 7, 8]}    |
```

```
| {"id": 150, "val": [9, 0, 1, 2]} |
| {"id": 200, "val": [3, 4, 5]}    |
| {"id": 220, "val": [6, 7]}       |
+----------------------------------+
10 rows in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE JSON_VALUE(j, '$.id' RETURNING UNSIGNED) = 150\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: Using where
1 row in set, 1 warning (0.00 sec)
```

※インデックスが使われなかったのでテーブル定義を確認。

```
mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `j` json DEFAULT NULL,
  KEY `i1` ((json_value(`j`, _utf8mb4'$.id' returning unsigned)))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)
```

※「_utf8mb4」が補完されていた。

```
mysql> EXPLAIN SELECT * FROM t1 WHERE json_value(`j`, _utf8mb4'$.id' returning unsigned) = 150
\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
   partitions: NULL
         type: ref
possible_keys: i1
          key: i1
      key_len: 9
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0.00 sec)
```

※インデックスの定義に合わせて WHERE 句を書いてみたところ、うまく動いた。

■コラム: JSON_SCHEMA_VALID() で CHECK 制約

8.0.19 から、JSON_SCHEMA_VALID() が CHECK 制約に対応しています。

- https://dev.mysql.com/doc/refman/8.0/en/json-validation-functions.html#json-validation-functions-constraints

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

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 {
```

[*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。

```
        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");

                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
}
```

なお、例では person_id 列にインデックスを追加していますが、主キー[*3]を手動で指定する場合は_id 列に値を入れます。

また、以下は 8.0.17 で非推奨になりました。

- `Collection.find().where()`
- `Collection.modify().where()`
- `Collection.remove().where()`

5.3 その他の JSON 新機能

- MySQL Shell ／ JSON・BSON データのインポート
- JSON パス表現の拡張
- JSON オブジェクト値の高速ソート
- JSON オブジェクト値のインプレース更新
- 複数値インデックス（Multi-Valued Indexes）

[*3] 通常は自動で値が入ります。

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。`ST_Length()` は 8.0.16 で単位の指定ができるようになり、8.0.18 で異なるジオメトリタイプ（型）間の距離を計算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_FrechetDistance()	離散フレシェ距離を返す
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

関数名	説明
ST_GeometryType()	ジオメトリタイプの名前を返す
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_HausdorffDistance()	離散ハウスドルフ距離を返す
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

関数名	説明
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
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

実行例

- 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]。

```
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。
```

[*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（https://www.geocoding.jp/）で調べたものです。

```
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
- DROP SPATIAL REFERENCE SYSTEM ステートメント
- シェープファイル・GeoJSON ファイルのインポート

■コラム: MySQL 8.0 にシェープファイルをインポートするツール・shp2mysql

MySQL 8.0 の登場以来課題だったシェープファイルのインポートですが、宮内さん作の shp2mysql によってかなり楽になりました。

- https://qiita.com/miyauchi/items/b4e810b3becf2cf07e2f

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
- https://dev.mysql.com/doc/refman/8.0/en/alter-instance.html
  - マスターキーのローテーション（`ALTER INSTANCE`）

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、Source（Master）・Replica（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。

Source（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
```

Replica（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、Replica（Slave）を Source（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。

Source（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)
```

Replica（Slave）側操作

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

```
mysql> CHANGE REPLICATION SOURCE TO
    ->     SOURCE_HOST='【ホスト名】',
    ->     SOURCE_USER='repl',
    ->     SOURCE_PASSWORD='T35+U53r',
    ->     SOURCE_LOG_FILE='binlog.000001',
    ->     SOURCE_LOG_POS=4;
Query OK, 0 rows affected, 2 warnings (0.01 sec)

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

mysql> SHOW REPLICA STATUS\G
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source 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_Source: NULL
Source_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:
             Source_Server_Id: 1
（中略）
       Get_source_public_key: 0
1 row in set (0.00 sec)
※先ほど Source（Master）でユーザを作成したことが原因。Replica（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 REPLICA;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW REPLICA STATUS\G
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
（中略）
              Source_Log_File: binlog.000001
          Read_Source_Log_Pos: 1856
（中略）
```

```
       Seconds_Behind_Source: 0
（中略）
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
（中略）
       Get_source_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　バイナリログトランザクション圧縮

8.0.20 でバイナリログトランザクション圧縮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/binary-log-transaction-compression.html

7.3　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

■コラム: その他のバイナリログ関連情報

MySQL 8.0 のバイナリログについては、以下の情報も参考になります。

- https://www.s-style.co.jp/blog/2019/11/5440/
- https://labs.gree.jp/blog/2019/10/19616/
- https://labs.gree.jp/blog/2019/10/19628/
- https://labs.gree.jp/blog/2019/11/19752/
- https://labs.gree.jp/blog/2019/11/19832/
- https://labs.gree.jp/blog/2019/11/19898/

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

なお、8.0.17 でクローンプラグインとの組み合わせで自動ノードプロビジョニングをサポートしました。

- https://mysqlhighavailability.com/mysql-innodb-cluster-automatic-node-provisioning/
- https://mysqlhighavailability.com/a-breakthrough-in-usability-automatic-node-provisioning/

7.5 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

7.5.1 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）
- 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
- 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
- 応答のないメンバーをグループから追放
- プライマリフェイルオーバー時の一貫読み取り
- プライマリフェイルオーバー候補の優先順位設定
- 非同期接続フェイルオーバ機能がグループレプリケーション・トポロジをサポート
- メンバーの書き込み許可を自動で OFF（`super_read_only` のチェック）
- 複数バージョンが混在するグループの運用性を向上
- メンバーのオンラインアップデート
- グループレプリケーションでの自動プロビジョニング
- TLS 1.3 のサポート
- IPv6 のサポート
- 圧縮の設定
- フロー制御を微調整するためのオプション
- 許可リストでのホスト名のサポート

- メッセージング関連
  - 設定可能なメッセージングパイプライン
  - メッセージ受け渡しのトレース
  - メッセージの最大サイズ
  - メッセージのキャッシュサイズ
- トランザクションセーブポイントのサポート
- Performance Schema 項目の追加
  - threads.INSTRUMENTED
  - memory_summary_global_by_event_name
  - 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
- レプリケーション接続オプション
- バイナリログのチェックサムをサポート
- 可用性向上のために 2 つのシステム変数のデフォルト値を変更
  - `group_replication_member_expel_timeout`・`group_replication_autorejoin_tries`

7.6 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

7.6.1 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
- 接続成功時に `max_connect_errors` をリセット
- `mysqlrouter_plugin_info` ツールを追加
- メタデータキャッシュの TTL を 300 秒から 500 ミリ秒に短縮
- ルーティングストラテジを追加（`routing_strategy` オプション）
- 起動オプション追加
- 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
- HTTP サーバプラグインと REST API
  - 8.0.17 で追加、8.0.22 でデフォルト有効に
- MySQL Server のソースツリーの一部として Router を構築

7.7 MySQL Shell

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

[*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。

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）。

AdminAPI

- https://dev.mysql.com/doc/mysql-shell/8.0/en/admin-api-userguide.html
  - 8.0.17 で機能向上、さらに 8.0.19 で InnoDB ReplicaSet（後述）などをサポート
  - 8.0.20 で InnoDB Cluster・InnoDB ReplicaSet の管理者およびルータアカウント設定などをサポート
  - 8.0.23 でクラスタ診断機能が強化

InnoDB ReplicaSet

AdminAPI の機能として 8.0.19 で追加されました。InnoDB Cluster とは違い、非同期レプリケーションによるクラスタを構成します。

- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html

新しい Shell プラグイン構造

- https://mysqlserverteam.com/mysql-shell-plugins-introduction/

Clone Plugin（クローンプラグイン）

別ノードへのデータのオンラインコピーや自動プロビジョニングがサポートされました。

- https://dev.mysql.com/doc/refman/8.0/en/clone-plugin.html
- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster-clone-deployment.html

論理ダンプ・リストアツール

- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-table-export.html
  - Table Export Utility
- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-parallel-table.html
  - Parallel Table Import Utility
  - 8.0.23 で複数データファイルからのインポートをサポート
- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html
  - Instance Dump Utility・Schema Dump Utility・Table Dump Utility
- https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html
  - Dump Loading Utility

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

Python 3 サポート

- https://dev.mysql.com/doc/refman/8.0/en/mysql-shell-tutorial-python.html

その他の新機能と変更

- Centralized Help System（組み込みヘルプの改善）
- クエリ結果におけるカラムタイプ表示
- X DevAPI サポートをアップデート
- スクリーンページング
- Auto-Completion（自動補完）
- Prompt Themes（カスタマイズ可能なプロンプト）
- Command Line History Persistence（コマンド履歴永続化）
- シェル API の直接コマンドライン実行
- Connection Compression Options（コネクション圧縮オプション）

7.8　その他のレプリケーション新機能・変更

- チャネルフィルタ毎のマルチソースレプリケーション
- トランザクション内テンポラリテーブルの GTID サポート
- JSON 列の部分アップデート対応レプリケーション
- `RESET MASTER TO`
- `CHANGE REPLICATION SOURCE TO`（`CHANGE MASTER TO`）のホスト名が 255 文字まで指定可能に
- 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
- 受信側スレッドと適用側スレッドの間の競合の削減
- 拡張テーブルメタデータのバイナリログ記録
- `GTID_EXECUTED` が空でない場合に設定可能な `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
- トランザクション長のバイナリログへの記録
- 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
- `START REPLICA UNTIL`（`START SLAVE UNTIL` ／マルチスレッドレプリケーションへの対応）
- マルチスレッドレプリカ（スレーブ）で `SOURCE_AUTO_POSITION`（`MASTER_AUTO_POSITION=1`）のときのリレーログスキップ処理
- マルチスレッドレプリカ（スレーブ）レプリケーションにおけるデッドロック検出機構の改善
- 遅延レプリケーションのマイクロ秒対応
- `binlog-row-event-max-size` システム変数
- バイナリログキャッシュサイズの指定
- `mysqlbinlog` で圧縮をサポート
- 権限を限定したレプリケーション
- 新しい非同期レプリケーション接続フェイルオーバーメカニズム
- レプリカ（スレーブ）での主キーのチェックポリシーが設定可能に
- `auto_increment_increment`・`auto_increment_offset` のセッション値設定
- GTID を使用しないソースから GTID を使用するレプリカへのレプリケーションが可能に

- Deprecated（非推奨）になった設定等
  - `--master-info-file`
  - `master_info_repository`
  - `relay_log_info_file`
  - `relay_log_info_repository`
  - `slave_compressed_protocol`
  - `slave_rows_search_algorithms`
  - `log_bin_use_v1_row_events`
  - `WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS`
- 「ホワイトリスト」「マスター」「スレーブ」（用語）の非推奨化による変更
  - `group_replication_ip_whitelist` が `group_replication_ip_allowlist` に
  - `MASTER` が `SOURCE`（または `REPLICATION SOURCE`）に
  - `SLAVE` が `REPLICA` に

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

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

実行例

```
mysql> SELECT * FROM mysql.engine_cost\G
*************************** 1. row ***************************
  engine_name: default
  device_type: 0
    cost_name: io_block_read_cost
   cost_value: NULL
  last_update: 2020-05-02 23:41:29
      comment: NULL
default_value: 1
*************************** 2. row ***************************
  engine_name: default
  device_type: 0
    cost_name: memory_block_read_cost
   cost_value: NULL
  last_update: 2020-05-02 23:41:29
      comment: NULL
default_value: 0.25
2 rows in set (0.00 sec)
※ストレージの読み取りコスト=1 に対して、メモリの読み取りコスト=0.25 がデフォルト。上書き変更したら
「FLUSH OPTIMIZER_COSTS;」。

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

mysql> USE cost_test;
Database changed
mysql> CREATE TABLE t1 (id INT PRIMARY KEY AUTO_INCREMENT, val INT);
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO t1 SET val = 100;
Query OK, 1 row affected (0.01 sec)
（中略）
mysql> SELECT * FROM t1;
+----+------+
| id | val  |
+----+------+
|  1 |  100 |
（中略）
| 10 | 1000 |
+----+------+
10 rows in set (0.00 sec)

mysql> CREATE TABLE t2 (id INT PRIMARY KEY AUTO_INCREMENT, val INT);
Query OK, 0 rows affected (0.02 sec)
（中略）
mysql> SELECT * FROM t2;
+----+-------+
| id | val   |
+----+-------+
|  1 |  1000 |
（中略）
| 10 | 10000 |
+----+-------+
```

```
10 rows in set (0.00 sec)
※バッファプールをクリアするため、ここで MySQL サーバを停止。/etc/my.cnf の末尾に
「innodb-buffer-pool-load-at-startup=OFF」を追記してから起動。

mysql> USE cost_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> CHECKSUM TABLE t1;
+--------------+------------+
| Table        | Checksum   |
+--------------+------------+
| cost_test.t1 | 3821923307 |
+--------------+------------+
1 row in set (0.00 sec)
※ t1 テーブルのみバッファプールに読み込み。t1 テーブルの行読み取りコストが t2 テーブルの 1/4 に。

mysql> EXPLAIN SELECT * FROM t1, t2 WHERE t1.id = t2.id AND t1.id > 2 AND t2.id < 9\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t2
   partitions: NULL
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 6
     filtered: 100.00
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: cost_test.t2.id
         rows: 1
     filtered: 100.00
        Extra: NULL
2 rows in set, 1 warning (0.00 sec)
※ t2 テーブルが駆動表になった。

mysql> CHECKSUM TABLE t2;
+--------------+------------+
| Table        | Checksum   |
+--------------+------------+
| cost_test.t2 | 1155527278 |
+--------------+------------+
1 row in set (0.00 sec)
※ t2 テーブルもバッファプールに読み込み。t1 テーブルと t2 テーブルの行読み取りコストが同じに。
```

```
mysql> EXPLAIN SELECT * FROM t1, t2 WHERE t1.id = t2.id AND t1.id > 2 AND t2.id < 9\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
   partitions: NULL
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 6
     filtered: 100.00
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: t2
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: cost_test.t1.id
         rows: 1
     filtered: 100.00
        Extra: NULL
2 rows in set, 1 warning (0.00 sec)
```

※駆動表と内部表が逆転した。

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 …」

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
  - HASH_JOIN, NO_HASH_JOIN ※ 8.0.18 のみ有効。8.0.20 以降では BNL, NO_BNL でハッシュジョインを ON ／ OFF
  - MERGE, NO_MERGE

- https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-index-level
  - GROUP_INDEX, NO_GROUP_INDEX
  - INDEX, NO_INDEX
  - INDEX_MERGE, NO_INDEX_MERGE
  - JOIN_INDEX, NO_JOIN_INDEX
  - ORDER_INDEX, NO_ORDER_INDEX
  - SKIP_SCAN, NO_SKIP_SCAN
- 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-resource-group
  - RESOURCE_GROUP
- https://dev.mysql.com/doc/refman/8.0/en/optimizer-hints.html#optimizer-hints-set-var
  - SET_VAR

8.1.5 オプティマイザスイッチ

subquery_to_derived・prefer_ordering_index（および hypergraph_optimizer：8.0.22 時点ではデバッグビルドのみ）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/server-system-variables.html#sysvar_optimizer_switch
- https://dev.mysql.com/doc/refman/8.0/en/switchable-optimizations.html

8.1.6 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

実行例

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

mysql> USE skipscan_test;
Database changed
mysql> CREATE TABLE t1 (f1 INT NOT NULL, f2 INT NOT NULL, PRIMARY KEY(f1, f2));
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO t1 VALUES
    ->   (1,1), (1,2), (1,3), (1,4), (1,5),
    ->   (2,1), (2,2), (2,3), (2,4), (2,5);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO t1 SELECT f1, f2 + 5 FROM t1;
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO t1 SELECT f1, f2 + 10 FROM t1;
Query OK, 20 rows affected (0.00 sec)
Records: 20  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM t1;
+----+----+
| f1 | f2 |
+----+----+
|  1 |  1 |
（中略）
|  2 | 20 |
+----+----+
40 rows in set (0.00 sec)

mysql> ANALYZE TABLE t1;
+------------------+---------+----------+----------+
| Table            | Op      | Msg_type | Msg_text |
+------------------+---------+----------+----------+
| skipscan_test.t1 | analyze | status   | OK       |
+------------------+---------+----------+----------+
1 row in set (0.00 sec)
※テーブルの統計情報を更新しておく。

mysql> EXPLAIN SELECT f1, f2 FROM t1 WHERE f2 > 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
   partitions: NULL
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 8
          ref: NULL
         rows: 13
```

```
     filtered: 100.00
        Extra: Using where; Using index for skip scan
1 row in set, 1 warning (0.00 sec)
※「Using index for skip scan」が表示された。
```

■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…」

8.1.7 Hash Join（ハッシュジョイン）

8.0.18 にて、Hash Join をサポートしました（8.0.20 にて等結合以外の内部結合や外部結合、セミジョイン・アンチジョインにも対応)。

なお、EXPLAIN FORMAT=TREE または EXPLAIN ANALYZE で Hash Join を含む実行計画を表示することが可能です。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/hash-joins.html
- https://dev.mysql.com/doc/refman/8.0/en/explain.html#explain-analyze
  - EXPLAIN FORMAT=TREE・EXPLAIN FORMAT=JSON・EXPLAIN ANALYZE では内部的に Query cast injection が使用される（8.0.18〜）
- https://dev.mysql.com/worklog/task/?id=13459
  - 8.0.23 で高速化

実行例

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

mysql> USE hashjoin_test;
Database changed
mysql> CREATE TABLE t1 (col1 INT);
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE t2 (col1 INT);
Query OK, 0 rows affected (0.03 sec)
※ここでデータを投入。各テーブル 10 万行ずつ、ランダムな値で。

mysql> SELECT COUNT(*) FROM t1;
```

```
+----------+
| COUNT(*) |
+----------+
|   100000 |
+----------+
1 row in set (0.01 sec)

mysql> SELECT COUNT(*) FROM t2;
+----------+
| COUNT(*) |
+----------+
|   100000 |
+----------+
1 row in set (0.02 sec)

mysql> SELECT * FROM t1 LIMIT 3;
+------+
| col1 |
+------+
|  895 |
|  944 |
|   36 |
+------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM t2 LIMIT 3;
+------+
| col1 |
+------+
|  964 |
|  237 |
|  296 |
+------+
3 rows in set (0.00 sec)

mysql> EXPLAIN FORMAT=tree SELECT * FROM t1 LEFT JOIN t2 ON t1.col1 = t2.col1\G
*************************** 1. row ***************************
EXPLAIN: -> Left hash join (t2.col1 = t1.col1)  (cost=1005095728.81 rows=10050955650)
    -> Table scan on t1  (cost=10072.15 rows=100159)
    -> Hash
        -> Table scan on t2  (cost=0.10 rows=100350)

1 row in set (0.00 sec)

mysql> EXPLAIN SELECT * FROM t1 LEFT JOIN t2 ON t1.col1 = t2.col1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: t1
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 100159
     filtered: 100.00
```

```
       Extra: NULL
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: t2
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 100350
     filtered: 100.00
        Extra: Using where; Using join buffer (hash join)
2 rows in set, 1 warning (0.00 sec)
※「hash join」が表示されている。

mysql> SELECT COUNT(t1.col1) FROM t1 LEFT JOIN t2 ON t1.col1 = t2.col1;
+----------------+
| COUNT(t1.col1) |
+----------------+
|       10002205 |
+----------------+
1 row in set (0.68 sec)
※ 8.0.19 は LEFT(OUTER)JOIN に対応していないので、同じ処理に 50 秒以上掛かる。
```

8.1.8 UPDATE・DELETE でセミジョイン（準結合）・マテリアライズ（実体化）最適化をサポート

8.0.21 にて、IN・EXISTS 句のサブクエリを使った更新・削除が高速化されました。

公式リファレンスマニュアル

- https://dev.mysql.com/doc/refman/8.0/en/semijoins.html
  - 「In MySQL 8.0.21 and later, …」

実行例

UPDATE で試してみます。

```
mysql> USE multitable_test;
Database changed

※テーブル構造は以下の通り。

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int NOT NULL AUTO_INCREMENT,
  `key` int NOT NULL,
  PRIMARY KEY (`id`),
```

```
  KEY `key` (`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE t2\G
*************************** 1. row ***************************
       Table: t2
Create Table: CREATE TABLE `t2` (
  `id` int NOT NULL AUTO_INCREMENT,
  `val` int NOT NULL,
  `key` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `valkey` (`val`,`key`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)
```

※それぞれ 50 行のデータを投入。

```
mysql> EXPLAIN UPDATE t1 SET t1.key = 0 WHERE t1.key IN (SELECT t2.key FROM t2 WHERE t2.val IN
(40, 60, 100))\G
*************************** 1. row ***************************
           id: 1
  select_type: UPDATE
        table: t1
   partitions: NULL
         type: ALL
possible_keys: key
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 44
     filtered: 100.00
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: <subquery2>
   partitions: NULL
         type: eq_ref
possible_keys: <auto_distinct_key>
          key: <auto_distinct_key>
      key_len: 4
          ref: multitable_test.t1.key
         rows: 1
     filtered: 100.00
        Extra: NULL
*************************** 3. row ***************************
           id: 2
  select_type: MATERIALIZED
        table: t2
   partitions: NULL
         type: range
possible_keys: valkey
          key: valkey
      key_len: 4
          ref: NULL
         rows: 15
```

```
     filtered: 100.00
        Extra: Using where; Using index
3 rows in set, 1 warning (0.00 sec)
```

※実行計画としてサブクエリのマテリアライズが選択されている。8.0.20 までは「DEPENDENT SUBQUERY」だった。

※データを約 10 万行まで増やして時間計測。いずれもデータがバッファプールに載っている状態で実行。

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

mysql> UPDATE t1 SET t1.key = 0 WHERE t1.key IN (SELECT /*+ SUBQUERY(INTOEXISTS) */ t2.key FRO
M t2 WHERE t2.val IN (40, 60, 100));
Query OK, 24576 rows affected (2.71 sec)
Rows matched: 24576  Changed: 24576  Warnings: 0
```

※マテリアライズを無効化して実行。実行計画は 8.0.20 以前と同じ。3 回実行した平均は 2.78 秒。

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.95 sec)

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

mysql> UPDATE t1 SET t1.key = 0 WHERE t1.key IN (SELECT t2.key FROM t2 WHERE t2.val IN (40, 60,
100));
Query OK, 24576 rows affected (1.08 sec)
Rows matched: 24576  Changed: 24576  Warnings: 0
```

※マテリアライズ有効（デフォルト）で実行。3 回実行した平均は 1.07 秒。2〜3 倍高速化した。

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.99 sec)
```

8.1.9 Derived Condition Pushdown 最適化

8.0.22 にて、主クエリ側にある WHERE 句などの条件をサブクエリに直接適用することで処理を効率化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/derived-condition-pushdown-optimization.html

実行例

```
mysql> CREATE DATABASE subquery_test;
Query OK, 1 row affected (0.02 sec)

mysql> USE subquery_test;
Database changed
```

```
mysql> CREATE TABLE t1 (i INT NOT NULL, j INT NOT NULL, k INT NOT NULL);
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO t1 VALUES(1000 * RAND(), 1000 * RAND(), 1000 * RAND());
※これで 2 万行挿入後、INSERT INTO ... SELECT を 5 回繰り返して 64 万行に増やした。

mysql> ALTER TABLE t1 ADD INDEX (i, j);
※ INDEX がないと高速化しづらいので INDEX を作成。

mysql> EXPLAIN SELECT /*+ NO_DERIVED_CONDITION_PUSHDOWN() */ * FROM (SELECT i, j, SUM(k) FROM
t1 GROUP BY i, j) AS dt WHERE i < 10 AND j > 990\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: <derived2>
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 638976
     filtered: 11.11
        Extra: Using where
*************************** 2. row ***************************
           id: 2
  select_type: DERIVED
        table: t1
   partitions: NULL
         type: index
possible_keys: i
          key: i
      key_len: 8
          ref: NULL
         rows: 638976
     filtered: 100.00
        Extra: NULL
2 rows in set, 1 warning (0.00 sec)

※プッシュダウン無効の場合は、サブクエリの処理（id:2）でフル INDEX スキャンが行われる。

mysql> EXPLAIN SELECT * FROM (SELECT i, j, SUM(k) FROM t1 GROUP BY i, j) AS dt WHERE i < 10 AN
D j > 990\G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: <derived2>
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2058
     filtered: 100.00
        Extra: NULL
*************************** 2. row ***************************
```

```
           id: 2
  select_type: DERIVED
        table: t1
   partitions: NULL
         type: range
possible_keys: i
          key: i
      key_len: 4
          ref: NULL
         rows: 6176
     filtered: 33.33
        Extra: Using index condition
2 rows in set, 1 warning (0.00 sec)
```

※プッシュダウン有効の場合はサブクエリの処理（id:2）で INDEX による絞り込みが行われる。

8.1.10 その他のオプティマイザ新機能

- 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
- LIKE 検索時の部分インデックスの適正利用
- INSERT ／ UPDATE ／ REPLACE ／ DELETE に対する EXPLAIN EXTENDED
- IN・EXISTS サブクエリのアンチジョイン・セミジョイン最適化
- PREPARE の実行アルゴリズムが変更（ORDER BY ?での列番号指定などが無視されるように）

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。

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

実行例

※クライアント 1 で実行（準備）。

```
mysql> USE skiplock_test;
Database changed
mysql> CREATE TABLE seat (
    ->   id INT PRIMARY KEY AUTO_INCREMENT,
    ->   flight_date DATE NOT NULL,
```

```
    ->   flight_number VARCHAR(10) NOT NULL,
    ->   seat_number VARCHAR(10) NOT NULL,
    ->   reserved BOOLEAN NOT NULL DEFAULT false,
    ->   UNIQUE (flight_date, flight_number, seat_number));
Query OK, 0 rows affected (0.04 sec)

mysql> SELECT * FROM seat ORDER BY id;
+----+-------------+---------------+-------------+----------+
| id | flight_date | flight_number | seat_number | reserved |
+----+-------------+---------------+-------------+----------+
|  1 | 2020-05-15  | RAC 801       | A2          |        0 |
|  2 | 2020-05-15  | RAC 801       | A3          |        0 |
（中略）
| 12 | 2020-05-15  | RAC 801       | A13         |        0 |
| 13 | 2020-05-15  | RAC 801       | C2          |        0 |
| 14 | 2020-05-15  | RAC 801       | C3          |        0 |
（中略）
| 24 | 2020-05-15  | RAC 801       | C13         |        0 |
| 25 | 2020-05-15  | RAC 801       | H1          |        0 |
| 26 | 2020-05-15  | RAC 801       | H2          |        0 |
| 27 | 2020-05-15  | RAC 801       | H3          |        0 |
（中略）
| 37 | 2020-05-15  | RAC 801       | H13         |        0 |
| 38 | 2020-05-15  | RAC 801       | K1          |        0 |
| 39 | 2020-05-15  | RAC 801       | K2          |        0 |
| 40 | 2020-05-15  | RAC 801       | K3          |        0 |
（中略）
| 50 | 2020-05-15  | RAC 801       | K13         |        0 |
+----+-------------+---------------+-------------+----------+
50 rows in set (0.00 sec)

※クライアント 1 で実行。「K3」席の行ロックを獲得する。
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM seat WHERE flight_date = '2020-05-15' AND flight_number = 'RAC 801'
    ->   AND seat_number = 'K3' AND reserved IS false FOR UPDATE;
+----+-------------+---------------+-------------+----------+
| id | flight_date | flight_number | seat_number | reserved |
+----+-------------+---------------+-------------+----------+
| 40 | 2020-05-15  | RAC 801       | K3          |        0 |
+----+-------------+---------------+-------------+----------+
1 row in set (0.00 sec)

※クライアント 2 で実行。「K3」席の行ロック獲得を「NOWAIT」で試みる。
mysql> USE skiplock_test;
Database changed
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM seat WHERE flight_date = '2020-05-15' AND flight_number = 'RAC 801'
    ->   AND seat_number = 'K3' AND reserved IS false FOR UPDATE NOWAIT;
ERROR 3572 (HY000): Statement aborted because lock(s) could not be acquired immediately and NO
WAIT is set.
※行ロックが獲得できなかったので即座ににエラーが返る。

※クライアント 3 で実行。「A3」「C3」「H3」「K3」席をまとめて、行ロック獲得を「SKIP LOCKED」で試みる。
```

```
mysql> USE skiplock_test;
Database changed
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM seat WHERE flight_date = '2020-05-15' AND flight_number = 'RAC 801'
    ->   AND seat_number LIKE '_3' AND reserved IS false FOR UPDATE SKIP LOCKED;
+----+-------------+---------------+-------------+----------+
| id | flight_date | flight_number | seat_number | reserved |
+----+-------------+---------------+-------------+----------+
|  2 | 2020-05-15  | RAC 801       | A3          |        0 |
| 14 | 2020-05-15  | RAC 801       | C3          |        0 |
| 27 | 2020-05-15  | RAC 801       | H3          |        0 |
+----+-------------+---------------+-------------+----------+
3 rows in set (0.00 sec)
※「K3」席以外の行ロックが即座に獲得できた。
```

8.2.2 ノンロッキング並列読み取り

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, …」
- https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html#sysvar_innodb_parallel_read_threads

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

実行例

```
mysql> CREATE DATABASE autoinc_test;
Query OK, 1 row affected (0.00 sec)
```

[*1] Cluster 構成でない通常の状態では最大値 +1 です。

```
mysql> USE autoinc_test;
Database changed
mysql> CREATE TABLE t1 (id int(10) PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100)) ENGINE=innod
b;
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO t1 SET name = 'first';
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO t1 SET name = 'second';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO t1 SET name = 'third';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM t1;
+----+--------+
| id | name   |
+----+--------+
|  1 | first  |
|  2 | second |
|  3 | third  |
+----+--------+
3 rows in set (0.00 sec)

mysql> DELETE FROM t1 WHERE id = 3;
Query OK, 1 row affected (0.00 sec)
※ id=3 の行を削除したところで、MySQL サーバを再起動。

mysql> USE autoinc_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 t1;
+----+--------+
| id | name   |
+----+--------+
|  1 | first  |
|  2 | second |
+----+--------+
2 rows in set (0.00 sec)

mysql> INSERT INTO t1 SET name = 'fourth';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM t1;
+----+--------+
| id | name   |
+----+--------+
|  1 | first  |
|  2 | second |
|  4 | fourth |
+----+--------+
3 rows in set (0.00 sec)
※ MySQL 5.7 までとは異なり、再起動後に id=3 は再利用されずに id=4 の行が挿入された。
```

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／システムテーブル等の暗号化

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）
- ダブルライトファイル

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-data-encryption.html
  - 透過的暗号化（TDE）全体の解説
- https://dev.mysql.com/doc/refman/8.0/en/innodb-data-encryption.html#innodb-data-encryption-redo-log
  - Redo ログ暗号化
- https://dev.mysql.com/doc/refman/8.0/en/innodb-data-encryption.html#innodb-data-encryption-undo-log
  - Undo ログ暗号化
- https://dev.mysql.com/doc/refman/8.0/en/innodb-data-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-data-encryption.html#innodb-mysql-tablespace-encryption-enabling-disabling
  - システムテーブル暗号化
- https://dev.mysql.com/doc/refman/8.0/en/innodb-data-encryption.html#innodb-doublewrite-file-encryption
  - ダブルライトファイル

実行例

テーブルスペース／ Redo・Undo ログ／システムテーブル／ダブルライトファイ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
# 8.0.23 からダブルライトを無効にしなくても正式にダブルライトファイルに平文で書き出されることがなくなった
(↑の 1 行が不要に)。
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。

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。
```

■コラム: InnoDB テーブルスペース管理について

こちらが参考になります。

- https://mysqlserverteam.com/innodb-tablespace-space-management/

8.2.5 その他の InnoDB 新機能

- CREATE ／ ALTER ／ DROP UNDO TABLESPACE
- ダブルライトバッファが InnoDB システムテーブルスペースから分離可能に
- 適応型のスキャンバッファサイズ調整
- 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

- ラッチフリーでスケーラブルな Redo ログ
- Redo ログアーカイブ処理
- Redo ログの無効化が可能に
- Undo テーブルスペースの処理性能向上と安定化・ACID Undo DDL のサポート
- LOB 列の再設計・改良による高速化
- CATS（the Contention-Aware Transaction Scheduling ＝新しいロックスケジューラ）
- テーブルスペースのパス検証の無効化が可能に
- `AUTOEXTEND_SIZE` オプション
- InnoDB テーブルスペースバージョン管理サポート（アップグレード用）
- 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
- バッファプールの Mutex 削除
- パージの改善
- テーブルスペース `DROP`・`TRUNCATE` の高速化
- デッドロック検出を自動的に有効化／無効化
- 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）
- より小さなコアファイルを生成するための新設定
- 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
- 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
- XA トランザクションロールバック時の権限チェック
- アイドル時書き込み IOPS の調整
- `SELECT 〜 FOR SHARE` が `SELECT` 権限のみで実行可能に
- Linux 環境においてテーブルスペース配置を効率化（`innodb_extend_and_initialize`）
- 並列度の低いシステムで dedicated log writer threads の無効化が可能に（`innodb_log_writer_threads`）

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

- https://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/information-schema.html
  - `VIEW_TABLE_USAGE`
  - `VIEW_ROUTINE_USAGE`
  - `KEYWORDS`
  - `COLUMN_STATISTICS`
  - `ST_GEOMETRY_COLUMNS`
  - `ST_SPATIAL_REFERENCE_SYSTEMS`

  - ST_UNITS_OF_MEASURE
  - APPLICABLE_ROLES
  - ADMINISTRABLE_ROLE_AUTHORIZATIONS
  - ENABLED_ROLES
  - ROLE_TABLE_GRANTS
  - ROLE_COLUMN_GRANTS
  - ROLE_ROUTINE_GRANTS
  - SCHEMATA_EXTENSIONS

9.1.4　その他の Information Schema 変更

- INNODB_TABLESPACES の SERVER_VERSION
- SELECT NAME, SUBSYSTEM, COMMENT FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME LIKE '%truncate%';
- TABLESPACES テーブルが非推奨に

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

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

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/performance-schema-error-log-table.html
  - エラーログテーブル
- 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/performance-schema-log-status-table.html
  - ログステータステーブル
- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-keyring-keys-table.html
  - `keyring_keys` テーブル
- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-clone-status-table.html
  - `clone_status` テーブル
- https://dev.mysql.com/doc/refman/8.0/en/performance-schema-clone-progress-table.html
  - `clone_progress` テーブル

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（Sys Schema を含む）

- ステートメントダイジェストに `QUERY_SAMPLE_TEXT` を追加
- 行ベースレプリケーションのモニタリング
- `APPLYING_TRANSACTION`・`APPLYING_TRANSACTION_START_APPLY_TIMESTAMP` など
- `COMPRESSION_ALGORITHMS`・`ZSTD_COMPRESSION_LEVEL`
- Sys Schema の `ps_is_consumer_enabled()` 関数
- `sys.version` ビュー（非推奨に）

9.3 その他の変更と新機能

9.3.1 SHOW ステートメント

- `SHOW EXTENDED COLUMNS`（隠しカラムの表示）
- `SHOW INDEX` に表示される情報の追加
- `SHOW PROCESSLIST` の性能改善（`Performance Schema.processlist` テーブルを利用）

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

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

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

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. …」

10.2.4 その他の DML

- 標準 SQL 対応など（TABLE ステートメント・VALUES ステートメントほか）
- 括弧付きクエリ式のサポート
- INSERT ... ON DUPLICATE KEY UPDATE で新しい行に対する行エイリアス・列エイリアスをサポート（VALUES() による参照を非推奨に）

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

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

10.3.3 その他の関数

- UUID 関数
- BLOB 列に対するビット処理
- CAST() 関数または CONVERT() 関数による YEAR 型へのキャスト
- CAST(value AT TIME ZONE specifier AS DATETIME) による TIMESTAMP 列値のタイムゾーン変換

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

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 …」

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
- サーバエラーメッセージ

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
- スロークエリログへの `log-slow-extra` の追加
- 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）
- 監査ログの JSON 形式化・圧縮・暗号化（5.7.21 と同様／ Enterprise 版）
- 監査ログにデータを挿入するための SQL 関数
- 監査プラグインで `audit_log_read()` を使用したログ読み込み操作を改善（Enterprise 版）
- データマスキング機能（5.7.24 と同様／ Enterprise 版）

10.4.5 その他の変更と新機能

- クエリキャッシュの廃止
- オンディスクテンポラリテーブルストレージエンジン
- 255 文字までのホスト名に対応
- キャスト関数・オペレータ（演算子）の拡張
- 日時リテラルがタイムゾーンオフセット付きフォーマットに対応
- 非推奨（Deprecated）となった関数・演算子など（8.0.17 以降）
  - `FLOAT(M,D)`・`DOUBLE(M,D)`、`ZEROFILL` 属性、`FLOAT`・`DOUBLE`・`DECIMAL` に付加する `UNSIGNED` 属性
  - `FLOAT`・`DOUBLE` カラムに対する `AUTO_INCREMENT`
  - 文字列データ型における `BINARY` 属性
  - 論理演算子「`&&`」・「`||`」・「`!`」
  - `SQL_CALC_FOUND_ROWS`、`FOUND_ROWS()`
  - `YEAR` 型の表示文字数 `(n)` と `UNSIGNED` 属性

- InnoDB memcached プラグインが非推奨に
- `INTO OUTFILE`・`INTO DUMPFILE` で生成するファイルのパーミッションを `0640` に
- `LOAD DATA INFILE LOCAL` に関する変更
- 新しいバックアップロック
- コネクション圧縮関連
  - `Compression_algorithm`・`Compression_level`
  - `protocol_compression_algorithms`
  - `Compression`（非推奨に）
- サーバ側キーリング移行ツール
- キーリングでサポートするキー長の増加
- AWS KMS 用のキーリングプラグイン（5.7.19 と同様）
- HashiCorp Vault 用のキーリングプラグイン
- Oracle Cloud Infrastructure（OCI）用のキーリングプラグイン
- 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
- 外部キーのためのメタデータロックのサポート
- The LOCK_ORDER Tool
- 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
- `--binary-as-hex` オプションに関する挙動の変更
- 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
- レプリケーションストリームを読み取るための C API
- 非同期 C API
- C API で `mysql_real_connect_dns_srv()` をサポート
- `mysql_bind_param()` C API 関数によるクエリ属性の定義（ユーザ定義関数を使用して SQL 文内から属性値の取得が可能に）
- UDF 自動登録コンポーネントのための UDF 登録サービス
- MySQL サーバー文字列コンポーネントサービス
- 結果セットのメタデータ転送オプション
- コンポーネント用のステータス変数サービス
- サーバ変数 `temptable_max_mmap` 追加
- 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
- ソートバッファの動的割り当て
- `NO PAD` 照合順序（COLLATION）用の可変長ソートキー
- `lower_case_table_names` の有効化に `debconf-set-selections` を利用
- KEY パーティショニングでカラムインデックスプレフィックスを使ったときに正しく警告・エラーを出力するようになった
- 文字列型と数値型・時間型の型変換（キャスト）がどのように行われたかを `EXPLAIN ANALYZE`・`EXPLAIN FORMAT=JSON`・`EXPLAIN FORMAT=TREE` で可視化
- ネットワーク名前空間指定子のサポート
- ユーザ定義関数 `gen_blacklist()` を `gen_blocklist()` に変更（Enterprise 版）
- MySQL Enterprise Firewall にグループプロファイル機能を追加（Enterprise 版）
- ソースコードの改善
- 環境変数 `MYSQL_PWD` が非推奨に

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 以降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。

※ 8.0.20 対応版より、印刷版は電子版 PDF の要約版となりました（参考ブログ記事等の URL 掲載を省略）。一方で、印刷版・電子版 PDF とも、実行例（サンプル）の数を増やしました。

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

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

漢のコンピュータ道

- https://nippondanji.blogspot.com/

日々の覚書

- https://yoku0825.blogspot.com/

MySQL Weekly

- https://mysql-weekly.hatenablog.com/

mita2 database life

- https://mita2db.hateblo.jp/

41 から始めました

- https://next4us-ti.hatenablog.com/

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
2019 年 5 月 26 日　第 2 版第 2 刷　発行
2019 年 8 月 8 日　第 3 版第 1 刷　発行
2019 年 8 月 25 日　第 3 版第 2 刷　発行
2019 年 10 月 27 日　第 4 版第 1 刷　発行
2019 年 12 月 20 日　第 4 版第 2 刷　発行
2020 年 1 月 4 日　第 4 版第 3 刷　発行
2020 年 1 月 19 日　第 5 版第 1 刷　発行
2020 年 3 月 22 日　第 5 版第 2 刷　発行
2020 年 5 月 15 日　第 6 版第 1 刷　発行
2020 年 11 月 14 日　第 7 版第 1 刷　発行
2021 年 2 月 8 日　第 8 版第 1 刷　発行
著　者　hmatsu47