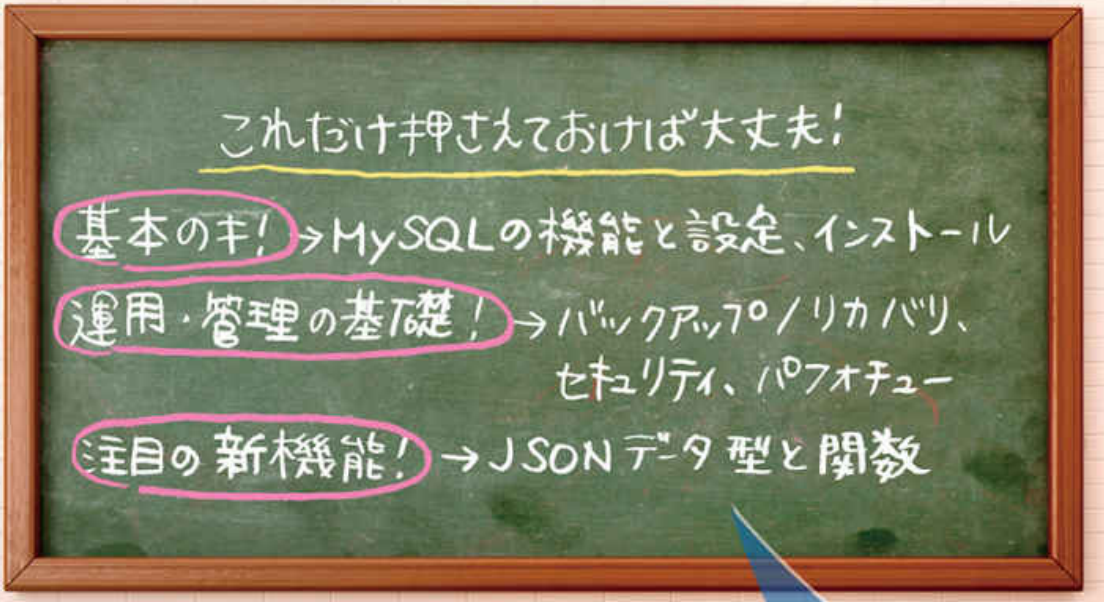
やさしく学べる

MySQL

運用・管理入門

5.7対応

梶山 隆輔／山﨑 由章（日本オラクル株式会社）・著



1レッスン45分

「セミナー感覚」で
すんなり身に付く！

インプレス

やさしく学べる

MySQL

運用・管理入門

5.7 対応

梶山 隆輔／山﨑 由章（日本オラクル株式会社）・著

インプレス

目次

●サンプルプログラムについて
本書で説明したサンプルプログラムは、以下のURLからすべてダウンロードできます。
http://book.impress.co.jp/books/1116101014

はじめに

本書はMySQLの運用・管理の現場で役立つポイントを基礎から学習できるように執筆しました。

開発から20年以上が経ち、MySQLは着実に進化を続け、さまざまな環境で利用されています。そして「使いやすさ／ease of use」は、引き続きMySQLの重要な開発コンセプトの1つとなっています。本書では、この「使いやすさ」と「体系的な知識」を融合することで、皆様がより確実にMySQLを運用・管理できることを狙いとしています。

MySQLの技術を順番に解説する書籍ではなく、各章をセミナーの1レッスンとして学習できるスタイルにしました。また各レッスンの末尾には演習問題を用意し、レッスン内容を復習できるようになっています。MySQL 5.7では、150を超える新機能の追加や機能改善が行われており、本書はMySQL 5.7をベースとした運用・管理方法について解説しています。

本書がこれからMySQLの運用・管理業務に携わる方や、すでに運用・管理を行っていて改めて体系的に学習したい方などのお役に立てることを願っております。

梶山隆輔

ある時「MySQLは初心者向けの情報が不足しているから、もっと初心者向けの情報も発信した方がいい」と、日本MySQLユーザ会の方から言われたことがあります。

そこで、初心者向けにMySQLの入門者セミナーを東京で企画したところ、予想を上回る申込みがあり、定員オーバーのために途中で申込みを締め切る事態が発生しました。

その後も「MySQLを使用する人がもっと増えて欲しい」との思いもあり、東京で同様のセミナーを再開催したり、大阪、名古屋、福岡、徳島など東京以外の地域でも同様のセミナーを開催したりしていますが、セミナーを開催できる機会には限りがあります。そのため、このような書籍で初心者向けに情報発信できる機会を頂けたことを大変嬉しく思っています。

本書によってMySQLを使用する方が増えたり、本書が皆様のMySQL学習のお役に立てれば幸いです。MySQLを使用する際のお供に、是非本書をご活用ください！！

山﨑由章

レッスン

0

RDBMS の基礎

このレッスンでは、レッスン 1 以降の解説を読み進めるための環境の準備や、理解しておきたい RDBMS の基礎知識を学習します。

0.1 本書の目的

多くのシステムではデータを蓄積し、そのデータを分析しています。データベースはこの「データの蓄積」を担う機能で、コンピューター上でデータベースを構築・運用するシステムをデータベース管理システム（DBMS。後述）と呼びます。リレーショナルモデルを採用したリレーショナルデータベース管理システム（RDBMS）は、DBMSの中でも幅広いシステムで利用されています。MySQLサーバーは、ソーシャルゲームやWebシステム、クラウドインフラなど大規模なシステムのバックエンドとなるデータベースから、小規模な社内システムのデータベースまで、さまざまな環境で採用されています。

本書は、MySQLサーバーのアーキテクチャの理解、MySQLサーバーを用いたインフラ設計、運用時に求められる知識など、MySQLの基礎知識を体系的に学べるセミナー形式となっています。アプリケーション開発だけではなく、バックアップ、セキュリティ、高可用性構成など、システムのインフラ設計やデータベースの運用管理に役立つ知識も学習していきます。各レッスンを読んでコマンドを試してみるまでがおおむね45分で終わるように構成されています。また各レッスンの終わりには各自で理解度を確認するための演習も用意しました。なお、各コマンドは、MySQLサーバーの最新版「MySQL 5.7」をベースにしています。MySQLサーバーで利用できるSQL文の構文については『できるProシリーズ MySQL』（インプレス）を参考にしてください。

0.2 MySQL環境の準備

本書では、MySQLサーバーをPCにインストールし、実際にコマンドを実行することを想定しています。MySQLサーバーのインストールや実行に必要なマシンスペックは公式には規定されていませんが、とりあえず動作させるためには、メモリーが500MB強、ディスクは1.5GB程度の空きがあれば十分です。

本番環境でのMySQLサーバーはLinux上で利用されることが多いため、本書でのコマンド例も特に断りがない場合はLinuxのものとします。なお、本書に掲載するコマンドは、実行環境によってリスト1のように、行頭の表示を変えてあります。

リスト1　本書でのコマンドの行頭表示

```
Windowsでのコマンド実行例
>
Linuxでのコマンド実行例
$
mysqlクライアントプログラムでのコマンド実行例
mysql>
各プロンプト内でのコメント
#
```

Column　仮想化ソフトウェアでLinux環境を構築できる

学習や動作検証、アプリケーション開発の段階では、Windows上でMySQLサーバーを利用している方も多くいます。コマンドの

動作確認をLinuxで行いたい方は、お手持ちのPCのOSを入れ替えることなく、VirtualBoxなどの仮想化ソフトウェアを用いて、WindowsやMacのOS上にLinux環境を構築できます。またLive CDやLive USBを利用すると、一時的にLinuxでブートできます。特にLive USBでは、インストールしたプログラムや作成したデータをUSBメモリー内に保管することが可能です。

VirtualBox上に仮想マシンを構築した例として、以下が参考になります。

- **Think IT 連載：読んで試す！ゼロからはじめるWordPress入門**
  VirtualBox上に仮想マシンを準備する
  https://thinkit.co.jp/story/2015/03/17/5678

0.3 RDBMSとは

リレーショナルデータベースでは、関係モデルの概念をベースに、表でデータを格納していきます。言い方を変えると、リレーショナルデータベースは、データを格納した表の集まりを管理するシステムとなります。各表では、関係モデルの「組」に該当するレコードを「行」で、「属性」を「列」と「データ型」で表現しています。これらの複数の表で共通な列の値を用いてレコードを結合し、新たな集合演算の結果を得ることが可能です。リレーショナルモデルの詳細は、『理論から学ぶデータベース実践入門』（奥野幹也著、技術評論社）で学ぶことができます。

リレーショナルデータベースは、データそのものを格納する表以外にも、各種のデータベースオブジェクトを管理しています。データベースオブジェクトの論理的なグループを「スキーマ」または「データベース」と呼びます。スキーマにはデータの重複を防ぐことやデータの抽出の効率化を目的とした「インデックス」、データベースに対する複数の操作をまとめたプログラムである「ストアドプロシージャ」、ユーザーが作成した「SQL関数」などが格納されています。データベース製品によって格納されるオブジェクトの種類は変わります。

0.3.1 RDBMSで求められる機能

RDBMSには表でのデータ管理に加えて、複数の役割が求められます。

格納するデータに対する制約

役割の1つは、不正なデータが格納されることを防いで、データの完全性を各種の制約によって担保することです。インデックスによる制約に加えて、表や列の定義を利用することができます。主な制約に関する機能は表1の通りです。

表1　主な制約

制約	機能
主キー	行ごとに値の重複を防ぐ。必ず値を持つ
ユニークキー	行ごとに値の重複を防ぐ。NULL の格納を許容する
外部キー	別の表の主キーまたはユニークキーの値のみを許容する
CHECK 制約	指定された条件式にあう値のみを許容する
NOT NULL 制約	NULL の格納を許容しない

データの操作

RDBMSのデータに対する操作は、通常、SQL文を利用します。SQL文は以下のカテゴリに分類されます。

- **Data Definition Language（DDL）:データベースオブジェクトの定義**
  例）CREATE文、ALTER文、DROP文
- **Data Manipulation Language（DML）:データの操作**
  例）INSERT文、UPDATE文、DELETE文
- **Data Control Language（DCL）:データアクセス権限の定義**
  例）GRANT文
- **Transaction Control Language（TCL）:トランザクションの管理**
  例）BEGIN、COMMIT、SAVE POINT、ROLLBACK

MySQLでは、分散キャッシュのmemcachedを組み込むことで、SQL文を使わずにキーバリュー型でのアクセスも可能にしています。SQL文だけではないデータの操作が可能な製品も複数あります。

ACIDトランザクション

データベースにおけるトランザクションは、分割できないアプリケーション処理の単位で、データを操作する一連のDMLを1つのグループとした論理的操作のことです。処理を確実に完了させるかどうかを制御してデータの変更が中途半端な状態にならないようにするため、また、ほかのクライアントの処理に邪魔されずに処理を完了させるために、トランザクションが必要となります。

トランザクションに求められる特性を「ACID特性」といいます。ACIDの各要素は表2の通りです。

表2 ACID特性

要素	英語	日本語	意味
A	Atomic	原子性	トランザクション内の処理を「すべて」完了させるか否かのいずれかであること
C	Consistent	一貫性	トランザクションの前後でデータの整合性が保たれ、矛盾がないこと
I	Isolated	独立性	処理中のトランザクションでの変更は完了するまで他のトランザクションから見えないこと
D	Durable	耐障害性	完了したトランザクションは障害が発生しても失われないこと

RDBMSでは、これらの特性が満たされているのが一般的です。しかし、旧バージョンのMySQLサーバーでは、明示的に指定しないとトランザクションをサポートしない表が作られ、これによってMySQLはトランザクションをサポートしていないと誤解されていたこともありました。

0.4 データベースサーバーの構築および運用時に考慮すべき事項

データベースサーバーは、システムで必要となるデータを保管するため、問題が起こった場合でもデータを失わない工夫が必要となります。また、不正なアクセスによってデータが改ざんまたは流出しないこと、多くのアクセスがあった場合でも滞りなく処理を継続することなども求められます。これらの考慮すべき事項を整理すると表3のようになります。

表3 データベースサーバー構築運用時のポイント

高可用性	データベースサーバーに障害が発生した際に、データを喪失せず、処理を継続する構成
バックアップ&リカバリ	データをデータベースサーバー外に保管し、障害復旧の際に利用する設定や仕組み
セキュリティ	アクセスの認証や暗号化などにより不正なアクセスやデータ改ざんなどを防ぐ設定や仕組み
パフォーマンス	想定した処理性能を発揮し、アクセスやデータ量増加時にも対処できる設定や仕組み
監視／管理	データベースサーバーの稼働状況を監視し、設定や構成の変更を行う仕組み

これらの項目は互いに独立したものではなく、例えば高可用性とパフォーマンスの向上を両立できる構成もあれば、逆に高可用性を確保するためにパフォーマンス面を犠牲にしなければならないケースなどがあり得ます。レッスン1以降では、MySQL 5.7のアーキテクチャや新機能に加え、MySQLを利用したデータベースサーバーを構築するために必要となる情報を解説していきます。

0.5 MySQLについて

MySQLサーバーは、1994年にスウェーデンとフィンランドのエンジニアによって開発が始まり、1995年に最初のバージョンが公開されました。コミュニティベースで開発されるオープンソースソフトウェアとは異なり、MySQLは企業が製品ロードマップの策定、製品開発および知財の管理を行い、ソフトウェアをGPLで公開するモデルを取っています。開発の初期からオープンソースソフトウェアとして公開しながら、同時に開発元として有償のサポートサービスを提供しています。開発当初はスウェーデンのMySQL ABの製品として、2008年のサン・マイクロシステムズによる買収、さらに2010年のオラクルによる買収を経て、現在はオラクルのデータベース製品ポートフォリオの1つとなっています。

MySQLは、Webアプリケーションやクラウドのバックエンドデータベースとして活用されることが多いオープンソースのリレーショナルデータベースです。MySQLは、シンプルで手軽に利用できるデータベースとして利用者から認識され、LAMP（Linux、Apache、MySQL、PHP/Perl/Python）スタックの一部として、Webアプリケーションのバックエンドで利用されています。またMySQLは、セキュリティパッケージや携帯電話網の通信機器などに組み込まれて利用される事例も多数あります。クラウド環境においては、クラウド基盤を構築するためのソフトウェアOpenStackのバックエンドストレージとして採用されており、各社のDataBase as a Service（DBaaS）でもMySQLが利用できるようになっています。

0.5.1 MySQLサーバーのエディションとライセンス

MySQLサーバーには、GPLの基で提供される「コミュニティ版」と、サポートサービスや拡張機能を含んだ商用ライセンスの基で提供される「商用版」があります。一般的にMySQLサーバーというと、コミュニティ版を指すことが多いのですが、本書で特に断りのない場合は、コミュニティ版および商用版に共通の情報です。

MySQLのコミュニティ版を自分で開発したソフトウェアに組み込んで配布（有償／無償を問わず）する場合は、GPLの規定に従うことが求められます。ただし、自分で開発するソフトウェアをオープンソースライセンスの基で配布する場合は、データベースに接続するためのライブラリやドライバーに関しては、FLOSS Exception※1という例外規定が用意されており、GPLの継承が必須ではなくなります。

商用版のMySQLサーバーでは、コンサルティングサポートというSQLチューニングやパラメータチューニングの支援を含んだサポートサービスのほか、GUIの稼働監視ツール、データベースファイアウォール機能、より強力な暗号管理機能などセキュリティ関連の拡張機能が提供されています。

0.5.2 MySQLサーバーのバージョンと主な機能

MySQLサーバーのバージョン番号は、5.7.12のように小数点2つで区切った表記となります。基本的に最初の2つの数字（MySQL 5.7.12の場合は5.7）がメジャーバージョンを表し、最後の1つがマイナーバージョンを表しています。リリースの間隔は固定されていませんが、メジャーバージョンがおおむね2、3年に1回、マイナーバージョンは1～3ヶ月に1回リリースされてきています。

2016年までにリリースされた主なメジャーバージョンと主な機能は図1の通りです。

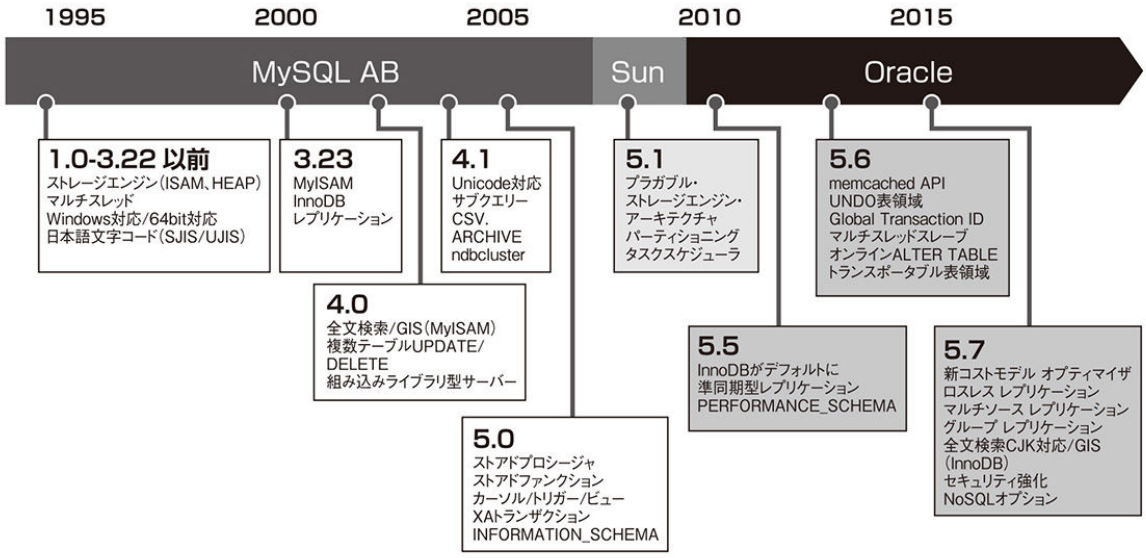


図1　MySQL サーバーの開発元企業、主なメジャーバージョンと実装された機能

製品リリース直前にMySQL 5.5にメジャーバージョン番号が変更されたMySQL 5.4や、アルファ版が公開されたものの開発が中止されてしまったMySQL 6.0など製品出荷にいたらかなった幻のバージョンも存在しています。

2016年9月には、次期MySQLサーバーのメジャーバージョンとして、MySQL 8.0の最初の開発途上版（Development Milestone Release）がリリースされました。MySQL 8.0はMySQL 5.7の次のメジャーバージョンです。

最大の変更点は、テーブル定義などのメタデータを新たにデータディクショナリによって管理することです。またユーザーアカウントやアクセス権限を管理するテーブル群のストレージエンジンも、従来のMyISAMからInnoDBに変更されています。システム系の情報の管理方法が大きく変わるということで、バージョン番号も明確に変更することが議論され、サン・マイクロシステムズ時代に開発中止となった6.0やMySQL Clusterのバージョン番号と重複する7.0を飛ばして8.0となりました。

表4 MySQL 8.0の主な変更点

データディクショナリ	テーブル定義やパーティション、トリガーの設定をフラットファイルからInnoDBを使用したデータディクショナリに変更
UTF-8	utf8mb4がデフォルトの文字コードに。Unicode 9.0サポート
権限テーブルのInnoDB化	権限管理テーブルをMyISAMからInnoDBに変更
ロール	ユーザーの権限管理にロールを追加。ROLES_GRAPHML()にて可視化
パフォーマンススキーマにインデックス追加	性能統計情報の分析を効率化
SQLコマンドによる設定変更永続化	SETコマンドで行っていたパラメータ変更を永続化可能に
パラメータ設定箇所の可視化	コンパイル時に設定値が設定ファイル、コマンドライン引数やSETコマンドのどれで設定されたか確認可能

MySQL 5.7向けとしても開発中である、MySQLサーバーをドキュメントデータベースとして利用するためのドキュメントストア機能もMySQL 8.0に組み込まれています。

このほかオプティマイザヒントの追加やGIS機能の拡張などが行われています。

2016年9月にリリースされたMySQL 8.0.0 DMR1には含まれず、Labsリリースとして公開されたのがCommon Table Expressions（共通テーブル式）およびオプティマイザからインデックスを隠す機能Invisible Indexです。Invisible Indexは、インデックスの削除／再作成の処理を行わずに、インデックス削除の影響を確認することが可能になります。どちらもMySQL 8.0のメインの開発ツリーに統合されることが想定されます。

Column　MySQLの活用事例　Web&オンラインゲーム&クラウド

MySQLの有名な利用事例として挙げられるのが、Webやオンラインゲームです。イベントなどでMySQLの利用方法や運用ツールなどを紹介している企業には、「Facebook」や「Twitter」といった超大規模ソーシャルネットワーク、4億人以上のユーザーを持つオンラインストレージ「Dropbox」、1,000万ダウンロードを超えるゲーム「キャンディークラッシュ」などがあります。新しいビジネスのインフラとして採用されることも多く、タクシー配車サービスの「Uber」や民泊仲介サイトの「Airbnb」などもMySQLを活用してビジネスを成長させています。

日本でも国内のソーシャルネットワークの草分けともいえる「mixi」や「GREE」、ゲームを中心としたDeNAの「Mobage」、料理レシピのコミュニティ「クックパッド」、さらにはコミュニケーションアプリの「LINE」なども、各社のエンジニアによる情

報発信などでMySQLの利用形態を確認できます。数多くのヒット作品を持つスクウェア・エニックスでは、同社の認証や課金のための共通インフラにMySQLを採用し、サポートサービスを活用しているMySQLの顧客事例の1つとなっています。また、Yahoo!Japanや楽天などは、他のデータベース製品との組み合わせではあるものの、大量のMySQLサーバーを運用している企業として知られています。

MySQLがWebの領域で広く採用されてきた理由の1つが、MySQLサーバーのレプリケーション機能にあります（レプリケーションについてはレッスン9およびレッスン10で解説しています）。レプリケーション機能を利用すると、1台のサーバーに書き込んだデータを複数のサーバーにコピーできます。Webシステムでは、アクセスの大半が参照処理となることが多く、MySQLのレプリケーションを利用してデータのコピー先を参照させることで、システム全体の性能向上を図れる点がWebの要件にうまく適合しました。MySQLは2000年にレプリケーション機能を実装し、Webの急速な拡大にあわせて普及が進んでいきました。

MySQLは、クラウドの中でもさまざまな使われ方をしています。Webアプリケーションから発展したSoftware as a Service（SaaS）では、ビジネス向けファイル共有クラウドサービスの「Box」や人事システムの「Workday」、マーケティングオートメーションの「Marketo」など業種やサービスの種類を問わず幅広いアプリケーションのタイプで利用されています。クラウドプラットフォームを構築するために利用される「OpenStack」のメタデー

タリポジトリにもMySQLが使われています。また、プラットフォームレベルのクラウドサービスを提供する企業の多くは、MySQLをベースにしたDatabase as a Service（DBaaS）を用意しています。

　MySQLの導入事例は、これらの領域に限らず、業務向けのシステムやミッションクリティカルなシステムでもバラエティに富む事例が見られます。これについてはレッスン1以降のコラムで紹介します。

※1　URL: FOSS License Exception
http://www-jp.mysql.com/about/legal/licensing/foss-exception/

レッスン

1

MySQL 5.7 で新しくなったインストール方法

このレッスンでは、MySQL 5.7 で新しくなった MySQL サーバーのインストール方法を学習します。MySQL 5.7 ではインストール時に利用するコマンドが新しくなっているほか、インストール後の設定値も従来よりセキュリティを重視した設定となっています。

1.1 MySQLのダウンロードサイト

MySQLサーバーなどは、以下の複数のサイトからダウンロード可能となっています。サイトによって利用できるエディションが異なるので、用途や契約状況に応じたサイトを選択します。

なお、Linuxディストリビューションによっては、インストールメディアにMySQLが含まれているケースもありますが、多くの場合はバージョンが古いため、本書で紹介する手順でダウンロードし、インストールすることをおすすめします。　本書では基本的にコミュニティ版のMySQLサーバーを利用しています。

表1　MySQL 製品のダウンロードサイト

サイト名	URL	概要	バイナリ	ライセンス
MySQL 公式サイト	www.mysql.com	商用版およびコミュニティ版ダウンロード先へのリンクのみ		
MySQL デベロッパーゾーン	dev.mysql.com	GA および DMR（開発途上版）、Labs（機能実験版）などすべてのコミュニティ版バイナリおよびソースコードがダウンロード可能	コミュニティ版	GPL
My Oracle Support	support.oracle.com	サブスクリプションやライセンス購入者向けサポートサイト。過去にリリースされたすべてのマイナーバージョンを含む。コミュニティ版と同じ頻度でバージョンアップ	商用版	商用ライセンス
Oracle Software Delivery Cloud	edelivery.oracle.com	最新バージョンの試用版のダウンロードサイト。機能はサブスクリプションやライセンス購入後と同じ。他のサイトよりバージョンアップの頻度がやや低いため最新版でないことがあり得る	商用版	30 日間試用ライセンス

1.1.1 対応プラットフォーム

MySQLサーバーは、Linuxの各ディストリビューション、Solaris、WindowsおよびMac OS Xなど、主要なOS上での動作がサポートされています。MySQLサーバーの各バージョンとサポート対象OSのリストは次のサイトを参照してください。このページのOSのバージョンは、メジャーバージョンに相当するもののみが記載されています。例えば、Red Hat Enterprise Linuxの7.1や7.2は、リストの「Red Hat Enterprise Linux 7 / CentOS 7」に該当するので、MySQL 5.5から5.7までのバージョンの動作がサポートされています。

- **URL : Supported Platforms: MySQL Database**
  http://www-jp.mysql.com/support/supportedplatforms/

MySQLサーバーには特別なハードウェア要件はありません。簡単な動作検証であれば、最近のノートPC程度のスペックでも十分にインストール可能です。また、MySQLサーバーの動作環境はOSのみをサポート対象としており、仮想化環境やハードウェア、クラウドサービスに対しては特に動作保証の有無はありません。

1.1.2 インストールパッケージ

コミュニティ版MySQLサーバーのインストールパッケージは以下のページからダウンロード可能です。「Select Platform:」のプルダウンから対象のOSを選択し、必要なファイルをダウンロードします。このページでは、通常、現在の最新版が最初に表示されています。開発中のバージョンが公開されている時期には、ページ中段に「Generally Available (GA) Release」タブとは別に、「Development Releases」タブが表示されます。また、MySQL 5.7よりも古いバージョンのダウンロードが必要な場合には、ページ内の「Looking for previous GA versions?」にあるリンクから移動します。

- **URL : Download MySQL Community Server**
  http://dev.mysql.com/downloads/mysql/

ダウンロードの際は、「Login >>」ボタンをクリックしてoracle.comのアカウントでログインすることも可能です。アカウントを持っていない場合は、「Sign Up >>」ボタンからアカウントを新規登録できます。oracle.comにお客様情報を登録すると、MySQLのニュースレターをはじめ、各オラクル製品の最新情報の案内などを受け取れます。また、ログインや新規登録をせずにダウンロードすることもできます。その場合は「No thanks, just start my download.」のリンクを押して進んでください。

なお、このページではソースコードのダウンロードも可能となっています。MySQLの開発チームは、最適なコンパイルオプションを利用したバイナリを作成し配布しています。MySQLサーバー機能の

修正を行うなど特別な場合を除いて、ソースコードからビルドしてインストールを行わなくても問題ありません。

1.1.3 サポート期間

MySQLサーバーのバグ修正やメンテナンス・リリースが行われる期間は、基本的にはオラクルのライフタイム・サポートのポリシーに準じます。MySQLに関するオラクルのライフタイム・サポートでは、新規のパッチはPremier Supportとして一般提供開始(GA)日から5年間、さらにExtended SupportとしてPremier Support期間終了後3年間提供されます。その後のSustaining Support期間に入ると、原則的には新規のパッチは提供されません。各バージョンのサポート期間については、以下のページの下部にある「Oracle Lifetime Support Policy: Technology Products」を参照してください。

- **URL：MySQL　テクニカル・サポート**
  http://www-jp.mysql.com/support/

1.2 MySQLのドキュメント

MySQLの各製品のリファレンスマニュアルや各バージョンのリリースノートは、次のページに掲載されています。

- **URL : MySQL Documentation**
  http://dev.mysql.com/doc/

MySQL 5.7のリファレンスマニュアルには、日本語版がありません。日本語の情報が必要な場合は、MySQL 5.6のリファレンスマニュアルを参照することになります。リファレンスマニュアルの各ページの右上にはプルダウンメニューがあり、ほかのバージョンや別の言語のページを選択できるようになっています。MySQL 5.7の英語のページとMySQL 5.6の日本語のページを併用することも検討してください。

1.3 Windows環境へのインストール方法

Windows環境向けには、GUIインストーラ（MSI Installer）と、必要なファイルをまとめたZipファイルの2種類が用意されています。インストールされたファイルの管理やサービスとして起動する場合は、GUIインストーラが便利です。一方、Zipファイルは展開するだけで利用でき、レジストリも汚さないため、検証などの用途に使いやすいほか、手動でOSサービスに登録するなど起動停止方法を用意すれば本番環境での利用にも問題ありません。

1.3.1 GUIインストーラでWindows環境にインストールする

MySQL 5.6からWindows版インストーラの名称が「MySQL Installer for Windows」となり、GUIのデザインを含めて一新されました。ウィザード形式のこのインストーラでは、以下の機能をまとめてインストール可能になっています。また、インストール後には、MySQLサーバーのrootユーザーのパスワード設定や、Windowsサービスとしての登録、ログファイル名の指定などが可能です。インストールできるコンポーネントは表2の通りです。

表2 MySQL Installer for Windows でインストール可能なコンポーネント

コンポーネント名	内容
MySQL Server	MySQL サーバー本体
MySQL Workbench	運用開発を支援する GUI ツール
MySQL Connectors	アプリケーションプログラムからの接続部品
MySQL Notifier	MySQL サーバーのステータスを表す通知機能
MySQL for Excel	MySQL を操作するための Excel プラグイン
MySQL for Visual Studio	MySQL を操作するための Visual Studio プラグイン
Sample Databases	サンプルデータベース
MySQL Documentation	リファレンスマニュアル

インストーラのパッケージには、表3のようにFull版とWeb版があります。

表3 MySQL Installer for Windows のパッケージ

パッケージ	ファイル名の書式	ファイルサイズ	概要
Full 版	mysql-installer-community-VERSION.N.msi	150MB～200MB	インストール可能なすべてのコンポーネントを含む
Web 版	mysql-installer-community-web-VERSION.N.msi	2MB	インストールするコンポーネントを個別にダウンロード

ファイル名のVERSIONは5.7.12などMySQLサーバーのバージョン、Nはインストーラパッケージのバージョンです。MySQL Installer for

Windowsは、コミュニティ版だけではなく、商用版にも用意されています。また、インストーラそのものは32bitバイナリですが、実際にインストールするMySQLサーバーなどは、OSの環境に応じて32bitバイナリか64bitバイナリかを選択可能です。

MySQL Installer for Windowsでのインストール手順は以下の通りです。

ダウンロードしたインストーラをダブルクリックして起動します。

「I accept the license terms」にチェックを入れて、「Next」ボタンをクリックします（図1）。

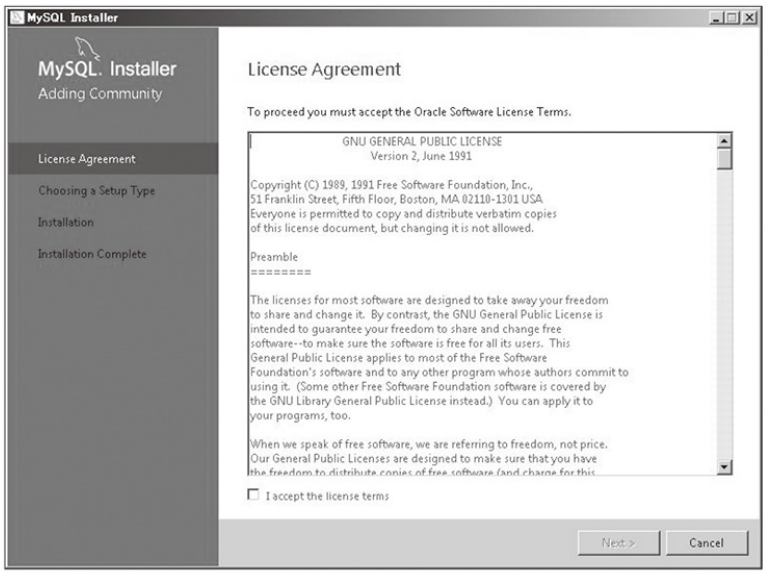


図1　ライセンスに合意

次にインストールタイプを選択します（図2）。デフォルトで選択されている「Developer Default」はアプリケーション開発者向けに構成されており、MySQLサーバーに加えてMySQL Workbenchをはじ

め、ExcelやVisual Studioのプラグインなどが含まれています。通常はデフォルトの「Developer Default」を選択すれば問題ありません。MySQLサーバーのみをインストールする場合は「Server only」を選択します。個別にコンポーネントをインストールするには、図2で「Custom」を選択して次に進み、コンポーネントを指定します（図3）。以下では「Custom」でコンポーネントを選択してインストールしています。

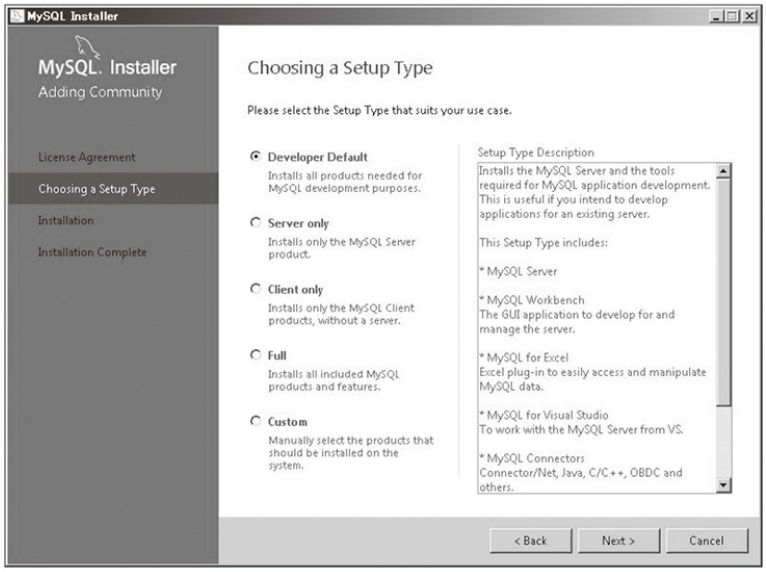


図2　インストールタイプを選択

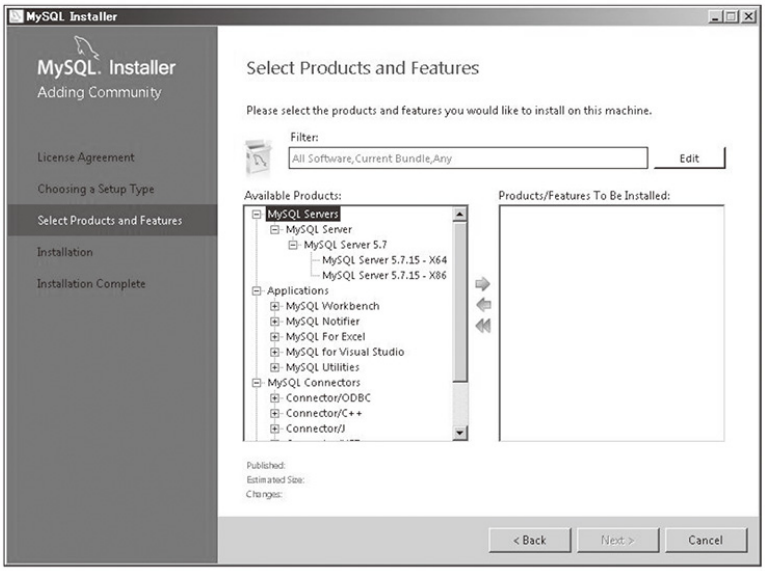


図 3　Custom インストールでコンポーネントを選択

　各コンポーネントのインストールにあたって、事前に必要なソフトウェアが見つからない場合に、ソフトウェア要件を確認する画面が表示されます（図4）。コンポーネント名の前にある○をクリックすると、必要なソフトウェアが表示されます。「Status」欄が「Manual」のものは手動で先に必要なソフトウェアをインストールする必要があります。空欄の場合はインストーラが自動で問題の解決を試みます。

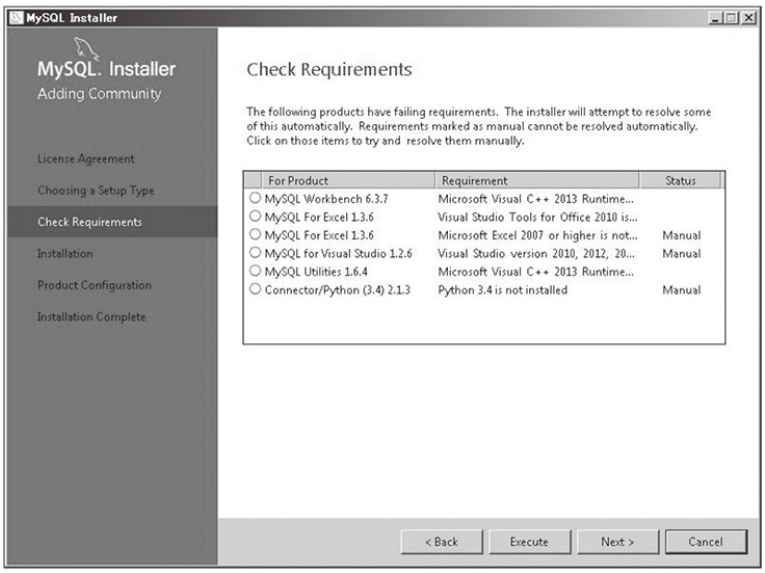


図4　ソフトウェア要件を確認

準備ができると「Ready to Install」と表示されるので、「Execute」ボタンをクリックします（図5）。

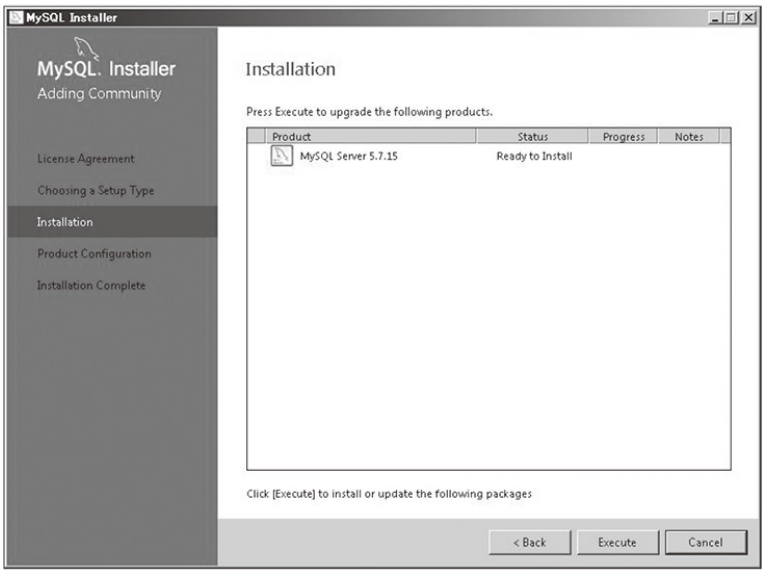


図5　インストール準備ができたことを確認

インストールの進捗状況は、「Progress」を見るとわかります（図6）。さらに「Show Details」ボタンを押すと、詳細な進捗状況を確

認できます（図7）。

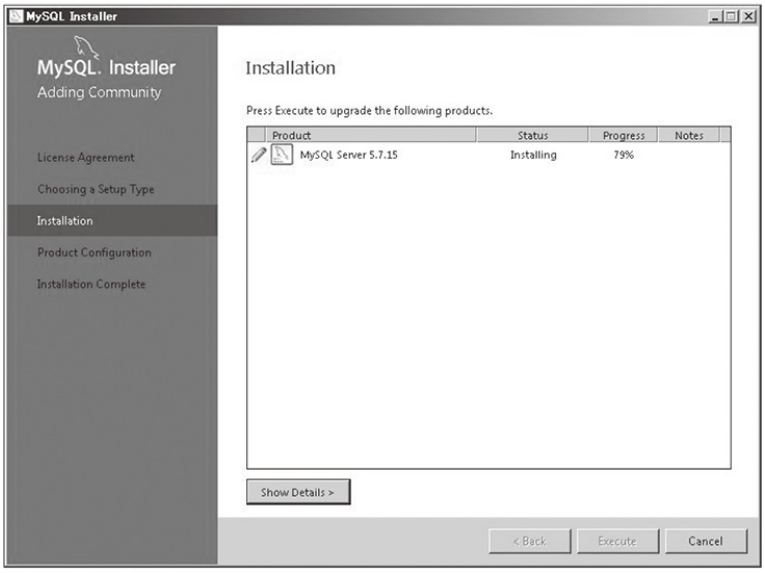


図6　インストールの進捗を確認

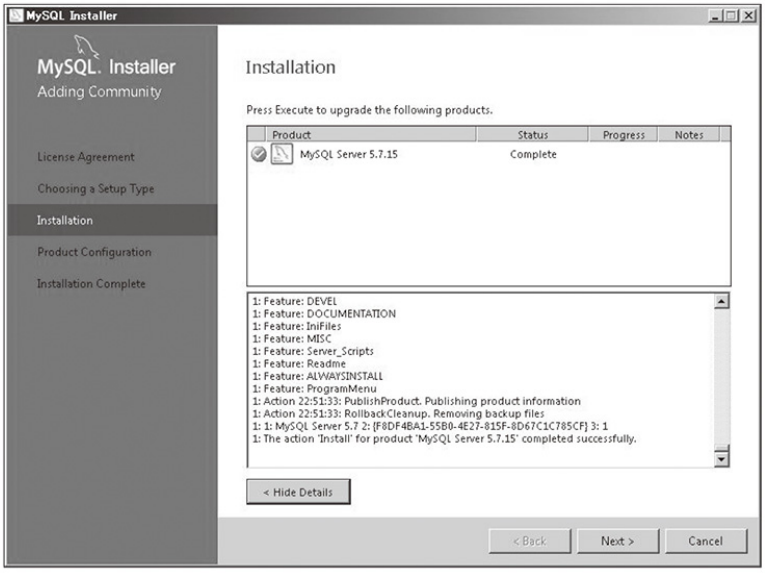


図7　「Show Details」ボタンを押してさらに進捗の詳細を確認

インストールが完了したら、MySQLサーバーの設定を行います。「Next」をクリックします（図8）。

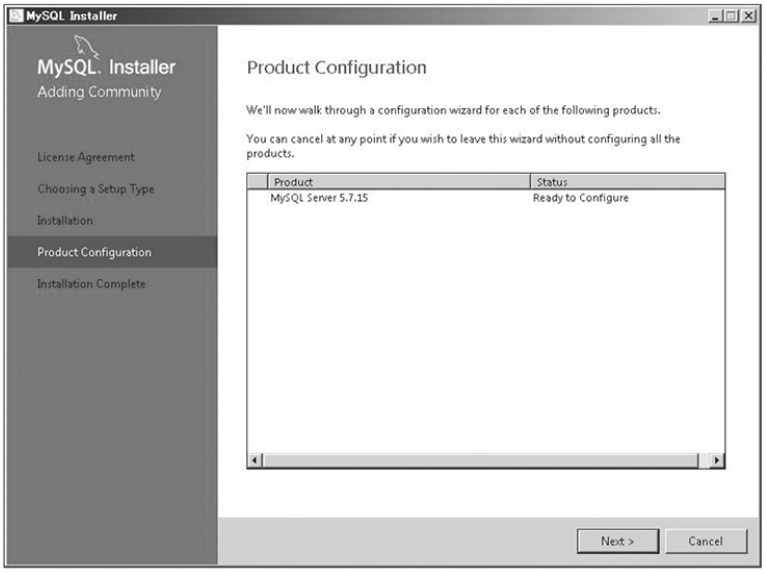


図 8　インストール完了後、MySQL サーバーの設定を行う

まず、ネットワーク接続を設定し、「Next」をクリックします（図9）。

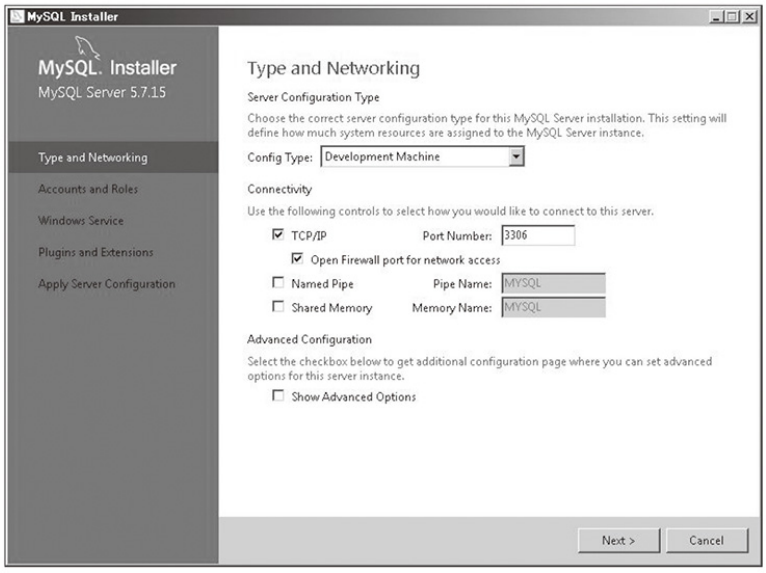


図 9　TCP/IP ポートなどネットワーク接続を設定

ルートアカウントのパスワードを設定して、「Next」をクリックします（図10）。

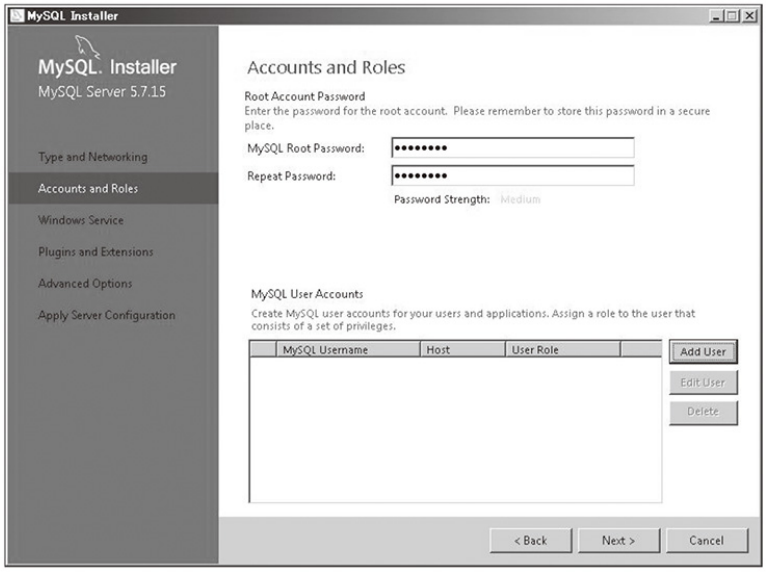


図 10　ルートアカウントのパスワードを設定

MySQLユーザーアカウントを追加します（図11）。

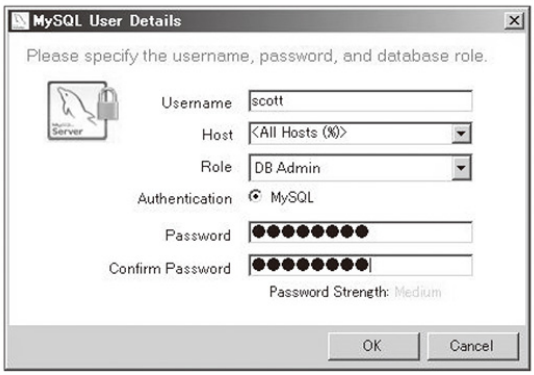


図 11　MySQL ユーザーアカウントを追加

Windowsのサービス登録を設定し、「Next」をクリックします（図12）。

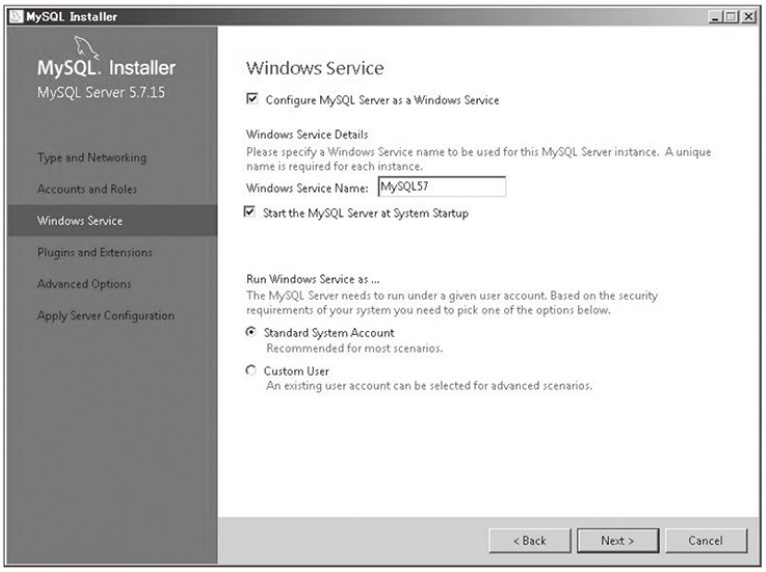


図 12　Windows のサービス登録を設定

MySQLドキュメントストア機能を設定します（図13）。MySQLドキュメントストアは、MySQL 5.7.15時点では開発中の、MySQLサーバーをドキュメントデータベースとして利用するための機能です。

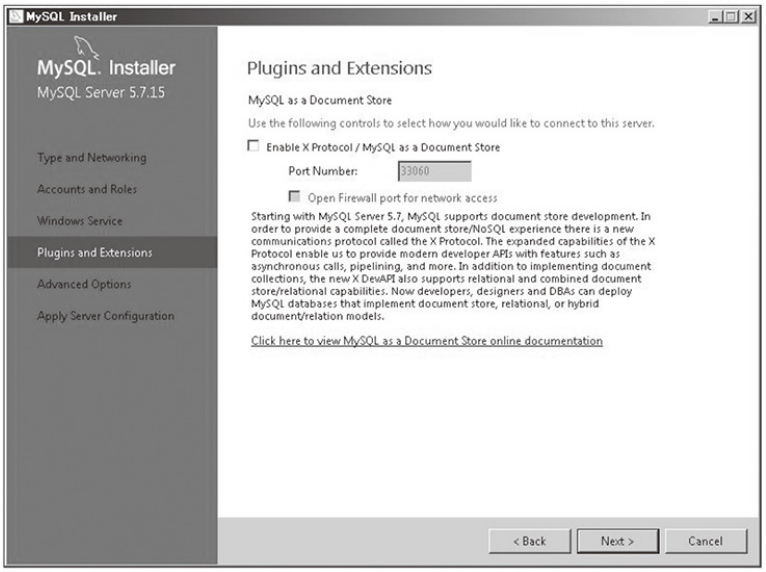


図 13　MySQL ドキュメントストア機能の設定

ログファイルを設定します（図14）。それぞれのログに関してはレッスン3の「3.1.6　ログ」で解説します。

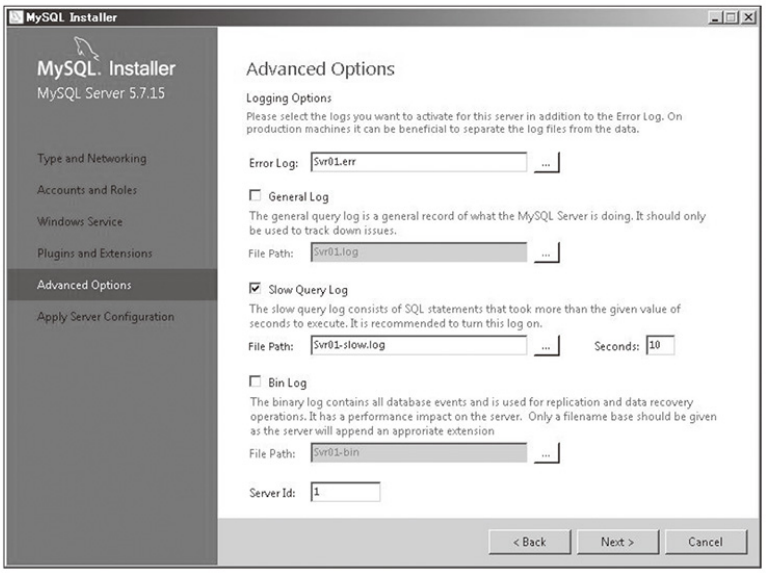


図 14　ログファイルの設定

設定した内容を適用します。「Execute」ボタンをクリックします（図15）。設定中の進捗状況も見られます（図16）。

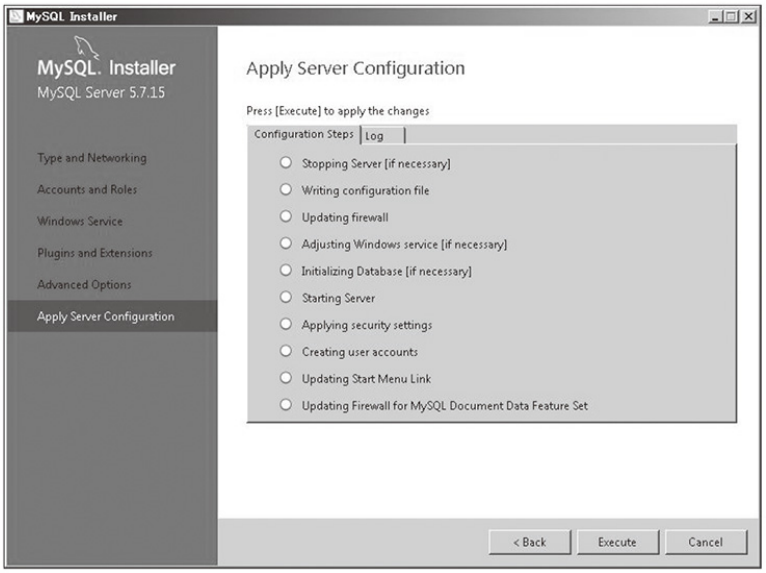


図 15　各設定項目を適用

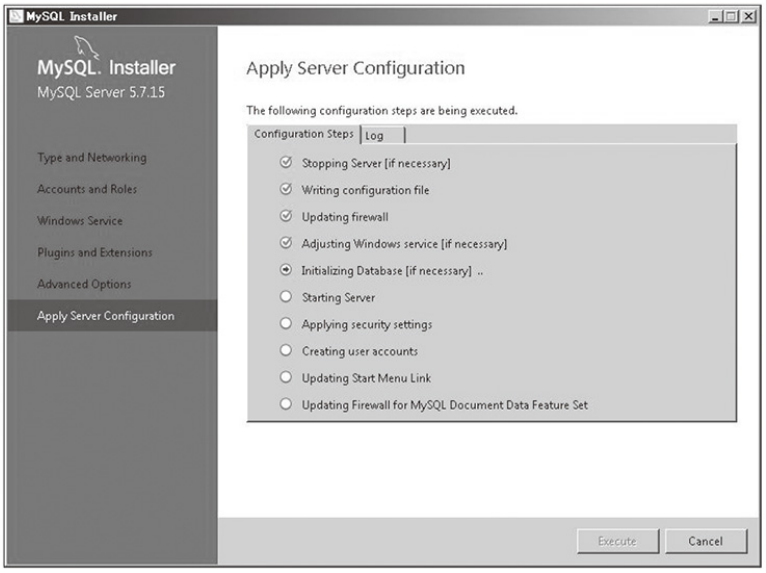


図 16　設定の進捗を確認

　これでインストールと設定は完了です。「Copy Log to Clipboard」ボタンをクリックすると、インストールと設定のログをクリップボードにコピーできますので、作業履歴として保管しておくことが可能です（図17）。

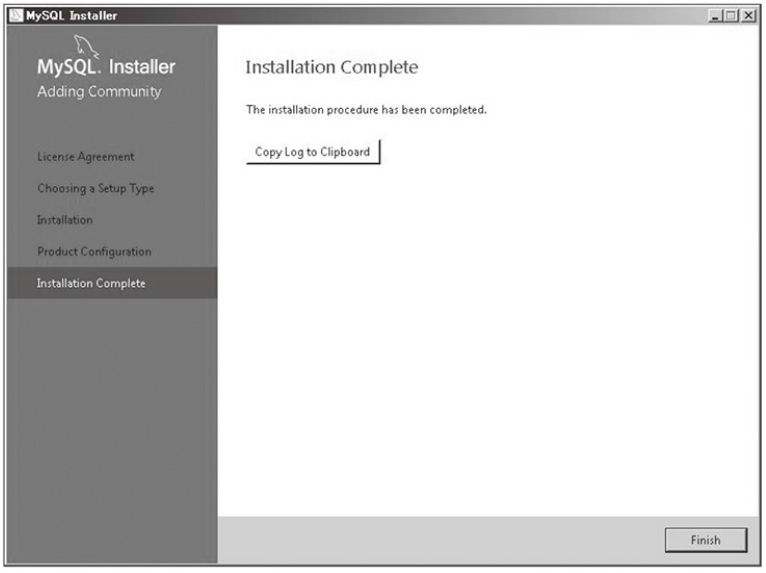


図 17　インストールと設定の完了

1.3.2 Windows上でZipファイルでのインストール

MySQL 5.6までは、Zipファイルを展開してMySQLサーバーの実行ファイルを起動するだけでインストールできましたが、MySQL 5.7では、新たにデータベースの初期化作業が必要となりました。インストールは以下のように進めます。

1. Zipアーカイブを任意のインストールディレクトリに展開する

任意のディレクトリでZipファイルを展開する。

例）C:￥mysql￥mysql-5.7.12-winx64

2. 設定ファイル（my.ini）を作成する

Zipファイルを展開したディレクトリにmy.iniを配置する。設定ファイルについてはレッスン2で説明する。設定ファイルの作成は必須ではないが、設定値を毎回サーバーの起動時に指定するのは効率的ではなく、ミスを誘発することにもなるので、設定ファイルを作成しておくことが望ましい。

3. データベースの初期化作業

MySQL 5.7で加わったMySQLサーバーのコマンドラインオプション（--initialize）を付けてMySQLサーバーを実行する。

リスト1　データベースの初期化コマンド例（MySQLサーバーのrootユーザーにランダムなパスワードを設定）

```
C:￥> C:￥mysql￥mysql-5.7.12-winx64￥bin￥mysqld --initialize
```

rootユーザーの初期パスワードはコンソールに表示されるので確認する。
MySQLサーバー起動後、rootユーザーでの初回接続時にパスワードの変更を求められる。

リスト2　データベースの初期化コマンド例（MySQLサーバーのrootユーザーにパスワードを設定しない）

```
C:￥> C:￥mysql￥mysql-5.7.12-winx64￥bin￥mysqld --initialize-insecure
```

4. MySQLサーバーを起動する（MySQLをサービス登録する）

毎回コマンドラインで起動／停止もできるが、Windowsのサービスとして登録することも可能。

リスト3　コマンドラインでの起動例

```
# defaults-fileオプションで設定ファイルを明示的に指定している例
C:￥> START C:￥mysql￥mysql-5.7.12-winx64￥bin￥mysqld --defaults-file=C:￥mysql￥mysql-5.7.12-winx64￥data￥my.ini
```

リスト4　コマンドラインでの停止例

```
# MySQLサーバーのrootユーザーとして実行するために-uオプションでユーザー指定、-pオプションでパスワード入力プロンプト表示
```

```
C:￥> C:￥mysql￥mysql-5.7.12-winx64￥bin￥mysqladmin -u root -p shutdown
```

リスト5　サービス登録コマンド例

```
# オプションファイルを明示的に指定し、mysql57というサービス名で登録
C:￥> C:￥mysql￥mysql-5.7.12-winx64￥bin￥mysqld --install mysql57 --defaults-file=C:￥mysqlmysql-5.7.12-winx64￥data￥my.ini
```

1.4 Linux環境へのインストール

Linux環境には、各ディストリビューション向けのパッケージ（rpmやdeb）と、ディストリビューション共通のtar.gzファイルが用意されています。ソフトウェアの管理をパッケージで行っている環境では、rpmなどのファイルを利用することになります。またYumやAPTのリポジトリを利用する方法も用意されています。一方で、tarファイルを利用したインストールは、同一環境上で複数のバージョンをインストールでき、シンボリックリンクを利用することでバージョンの切り替えや切り戻しなども簡単に行えます。

1.4.1 Linux上でRPMファイルでのインストール

Red Hat Package Manager（RPM）を利用できるLinux上では、MySQLデベロッパーゾーンからダウンロードしたRPMファイルでMySQLサーバーのインストールが可能です。RPMファイルでインストールした時の各ファイルの配置は以下の通りとなります。

表4 RPMでインストールした際のファイルレイアウト

ディレクトリ	概要
/usr/bin	クライアントプログラムおよびスクリプト
/usr/sbin	MySQL サーバープログラム
/var/lib/mysql	ログファイル、データファイル
/usr/share/info	Info フォーマットのマニュアル
/usr/share/man	Unix Man フォーマットのマニュアル Unix manual pages
/usr/include/mysql	ヘッダーファイル
/usr/lib/mysql	ライブラリ
/usr/share/mysql	各種ファイル（エラーメッセージ、文字コード定義、サンプル設定ファイルなど）

通常の利用環境では、mysql-serverとmysql-clientのパッケージをインストールしておけば十分です。ただし、RedHat Enterprise LinuxならびにOracle LinuxやCentOSなど互換のディストリビューションでは、mysql-libsが競合してインストールできない場合があります。そこで、事前にmysql-community-libs-compatパッケージでアップグレードすることで競合を解消できます。

リスト6　任意のパッケージをダウンロード後、rpm／yumコマンドでインストールする例

```
# rpmコマンドで互換性問題解消パッケージやライブラリをアップグレード
$ rpm -Uvh mysql-community-libs-compat-5.7.12.el6.x86_64.rpm
mysql-community-libs-5.7.12.el6.x86_64.rpm mysql-community-
common-5.7.12.el6.x86_64.rpm
```

```
# rpmコマンドでMySQLサーバーならびにクライアントコマンドをインストール
$ rpm -ivh mysql-community-server-5.7.12.el6.x86_64.rpm mysql-community-client-5.7.12.el6.x86_64.rpm

# 自動的に登録されたサービスを起動
$ service mysqld start

# サービスを停止
$ service mysqld stop
```

　インストール中にMySQLサーバーのrootユーザーにランダムなパスワードが自動生成され、MySQLサーバーのエラーログに記録されます。

1.4.2 Linux上でYumリポジトリを利用したインストール

MySQLの公式Yumリポジトリから最新のMySQLサーバーをインストールすることが可能です。MySQLサーバー本体やクライアントプログラムに加え、MySQL WorkbenchやMySQL Utilitiesもあわせてインストールできます。インストールの準備作業として、以下のページから利用中のディストリビューションおよびバージョン用のMySQL公式Yumリポジトリの情報をインストールします。

- **URL : Download MySQL Yum Repository**

  http://dev.mysql.com/downloads/repo/yum/

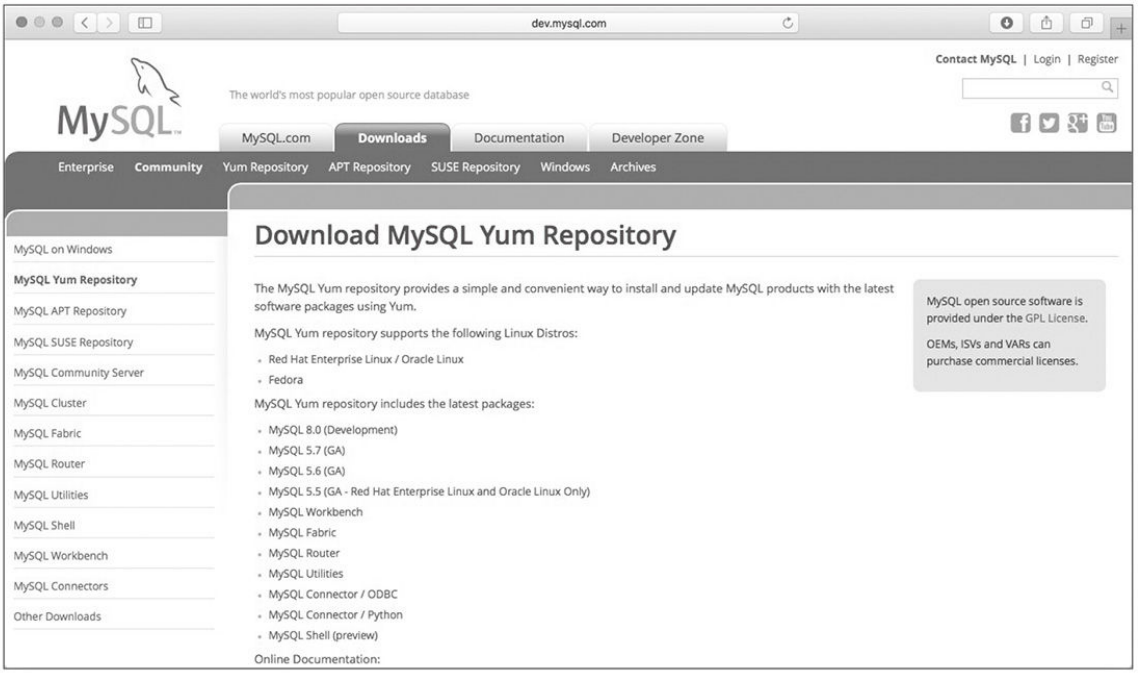


図 18　MySQL 公式 Yum リポジトリの情報のインストール

リスト7　CentOS 7にMySQL 5.7のYumリポジトリをインストール

```
$ sudo rpm -Uvh mysql57-community-release-el7-8.noarch.rpm
```

MySQL公式Yumリポジトリの情報のインストール後は、以下の手順でインストール、および起動／停止を行います。

リスト8　MySQLサーバーのインストール

```
$ yum repolist enabled | grep "mysql.*-community.*"
$ sudo yum install mysql-community-server
```

リスト9　MySQLサーバー起動／停止（初回起動時にデータベースも作成される）

```
# サービスを起動
$ service mysqld star

t# サービスを停止
$ service mysqld stop
```

RPMファイルでのインストール時と同様に、MySQLサーバーのrootユーザーに自動生成されたパスワードはエラーログ/var/log/mysqld.logに記録されます。

1.4.3 Linux上でtarファイルでのインストール

tarファイルをダウンロードするには、ダウンロードページの「Select Platform:」のプルダウンから「Linux - Generic」を選択します。なお、MySQL 5.7.15の時点では、ファイルの説明としてTARとなっているファイルと、Compressed TAR Archiveとなっているファイルがありますが、TARとなっているファイルはMySQLサーバー本体とテストプログラムが含まれています。通常はCompressed TAR Archiveを選択すれば問題ありません。

Windows上でZipファイルでのインストールと同様の手順で、Linux上にもインストールが可能です。tarファイルを展開したディレクトリにあるINSTALL-BINARYにも手順が掲載されています。

データベースの初期化作業では、Windows版Zipファイルと同様に、MySQLサーバーの--initializeまたは--initialize-insecureオプションを利用します。MySQL 5.6まではこのオプションがなかったため、mysql_install_dbコマンドを利用していました。また、OSのサービスへの登録が手動になるのもWindows版Zipファイルと同様です。

この方法でのインストール時、MySQLサーバーは/usr/local/mysqlにファイルが展開されたと想定して動作しようとします。ほかのディレクトリにtarファイルを展開した場合は、/usr/local/mysqlでシンボリックリンクを設定します。

リスト10　tarファイルでのインストール

```
# tarファイルの展開先が/home/mysql/mysql57の場合
# /usr/localにファイル作成権限のあるユーザーで下記コマンドを実行
$ cd /usr/loca
```

```
l# シンボリックリンクの設定
$ ln -s /home/mysql/mysql57 mysql

# mysqldなど実行コマンドの格納されたディレクトリの確認
$ cd mysql/bin
$ ls
```

シンボリックリンクを設定しない場合は、mysqld実行時にbasedirオプションで展開先のディレクトリを指定します。

リスト11　データベースの初期化コマンド例（MySQLサーバーのrootユーザーにパスワードを設定しない）

```
# tarファイルの展開先が/home/mysql/mysql57の場合
$ cd /home/mysql/mysql57/bin
$ ./mysqld --basedir=/home/mysql/mysql57 --initialize-insecure

# もしくは
# ディレクトリを絶対パスでの指定も可能
$ /home/mysql/mysql57/bin/mysqld --basedir=/home/mysql/mysql57 --initialize-insecure
```

リスト12　コマンドラインでの起動例

```
# --defaults-fileオプションで設定ファイルを明示的に指定している例
# 下記コマンドは改行を入れずに1行で
# コマンドの最後に半角の&(アンパサンド)を付けてバックグラウンドで実行
```

```
$ ./mysqld --defaults-file=/home/mysql/mysql57/my.cnf --basedir=/home/mysql/mysql57 &

# クライアントプログラムから接続確認 出力省略
$ ./mysql -uroot

# Windowsでは￥q (半角の￥と小文字Q)、Linuxでは＼q(半角バックスラッシュと小文字Q)でコネクションを切断
mysql> ＼q
```

MySQLサーバーを停止する方法は、MySQL 5.6まではmysqladminコマンドのサブコマンドであるshutdownが利用されていました。しかし、MySQL 5.7では、mysqlクライアントで接続した状態で、shutdown命令を出すことができるようになりました。以下のいずれかの方法で、MySQLサーバーを停止することができます。

リスト13　コマンドラインでの停止例（MySQLサーバーのrootユーザーとして実行するために-uオプションでユーザー指定、-pオプションでパスワード入力プロンプト表示）

```
# mysqladminコマンドのサブコマンドshutdownでMySQLサーバーを停止する場合
$ ./mysqladmin -uroot shutdown

# mysqlクライアントでMySQLサーバーに接続して停止する
$ ./mysql -uroot
mysql> SHUTDOWN;
```

```
# mysqlクライアントの-eで指定したSQL文やコマンドを実行
# 上記のMySQLサーバーに接続の上でSHUTDOWNを実行しているのと同じ
$ ./mysql -uroot -e"SHUTDOWN"
```

1.5 MySQL 5.7のインストール時の留意事項

これまで見てきたように、OSやインストールパッケージによって、手順やインストール後の状態が異なるため注意が必要です。特にLinuxでYumやRPMでインストールした場合、MySQLサーバーのrootユーザーのパスワードが自動生成され、MySQLサーバー起動後の初回ログイン時にパスワードの変更が求められます。この際、パスワード検証がデフォルトで有効になっているため、「8文字以上、大文字／小文字／数字／記号を1文字以上含む」パスワードの設定が必要です。なお、ほかの方法でインストールした場合にはパスワード検証は有効になっていません。

Column MySQLの活用事例 “堅い”システム

MySQLはWeb領域での利用事例が有名ですが、実際には業務向けのシステムやミッションクリティカルなシステムでもさまざまな活用事例があります。

例えば、オープンソースの利用促進を政策として掲げているインドでは、日本のマイナンバーに類似する「Aadhaar」と呼ばれる国民ID制度の基幹データベースとしてMySQLサーバーを採用しています。各国の軍事防衛系のシステムでは、ごく一部の例外を除いて詳細は公表されませんが、例えば米国海軍の航空母艦上で管制を行うミッションクリティカルなシステムはMySQL Clusterを採

用し運用していることが過去のMySQLのイベントなどで発表されています。

金融システムにおいても、従来型のシステムではなく「Fintech」と呼ばれる新しいタイプの金融サービスの領域でMySQLが活用されます。この領域もWebシステムの技術をベースとしていることが多く、インターネット経由での決済サービスを提供する「PayPal」では、特にマネーロンダリングのリアルタイム検出のためにMySQL Clusterを採用しています。ほかにもスマートフォンやタブレットでクレジットカード決済を行う仕組みを提供する「Square」、中国版のPayPalともいえる「Alipay」（アリペイ／支付宝）や「Tenpa」（テンペイ／財付通）でも大規模なMySQLプラットフォームを構築しています。

また、MySQLの別の利用方法として、ソフトウェアやアプライアンスに組み込んでいる例が多数あります。大規模向けのグループウェアである「サイボウズガルーン」やメディカル・データ・ビジョンの医療向けパッケージ、ソニーの映像データを管理するオプティカルディスク・アーカイブシステムなど、それぞれの製品の利用者がMySQLの存在に気付かない使われ方です。MySQLは、サーバー機でなくても十分なほどインストール時に必要となるスペック要件が高くないこと、各種のOS上で動作すること、さらには商用ライセンスの中で知財補償が含まれている点が、ほかのRDBMSと総合的に比較した場合のメリットとなり選択されています。

MySQLもソースコードを公開するプラットフォームとして利用しているリポジトリのホスティングサービス「GitHub」もデータ管理にMySQLサーバーを採用しています。GitHubは数多くのオープンソースプロジェクトがリポジトリとして利用しており、業務系とは異なった観点でMySQLが支えるきわめて重要なシステムといえます。

1.6 演習

このレッスンでは、WindowsとLinux環境でのMySQLサーバーのインストール方法を、インストールパッケージ別に学習しました。学習した内容を演習問題で確認しましょう。本編のOS別のインストール手順を確認しながら以下の作業を行ってください。なお、コマンドの実行時にはカレントディレクトリ内に実行ファイルがあることを確認してください。

1. dev.mysql.comから最新のコミュニティ版MySQLサーバーをダウンロードする

Windows上ならZipファイル、Linux上ならtarファイルを利用する

ヒント：レッスン1の「1.1　MySQLのダウンロードサイト」からダウンロード先ならびに利用するプラットフォーム別の情報を確認

2. 手元の環境にMySQLサーバーをインストールする

ヒント：レッスン1の「1.3.2　Windows上でZipファイルでのインストール」のリスト2および「1.4.3　Linux上でtarファイルでのインストール」のリスト11

3. コマンドラインからmysqldを実行して、MySQLサーバーを起動する

Windowsでの例

```
> START C:￥mysql￥mysql-5.7.12-winx64￥bin￥mysqld
```

Linuxでの例

```
# tarファイルの展開先ディレクトリに注意$ ./mysqld &
```

4. クライアントプログラムから接続する。接続確認後、コネクションを切断する

ヒント：レッスン1の「1.4.3　Linux上でtarファイルでのインストール」のリスト12の後半部分

5. コマンドラインからmysqladminコマンドを使用してMySQLサーバーを停止する

ヒント：レッスン1の「1.3.2　Windows上でZipファイルでのインストール」のリスト4および「1.4.3　Linux上でtarファイルでのインストール」のリスト13

解説

1. MySQLサーバーのファイルダウンロード時には、OSが32ビット版か64ビット版かにあわせて、MySQLサーバーのバイナリファイルをダウンロードします。ビット数を確認するには、Windowsの場合には下記URLを参考にしてください。

参照URL

http://faq.ricoh.jp/app/answers/detail/a_id/58/
https://121ware.com/qasearch/1007/app/servlet/relatedqa?QID=018022

Linuxの場合は、uname -aの出力にx86_64またはamd64の文字列が含まれていれば64ビット版となります。

ダウンロードするファイル名は以下の通りです。マイナーバージョンは演習時点での最新のものに置き換えてください。

- 32ビット版Windowsの場合：mysql-5.7.15-win32.zip
- 64ビット版Windowsの場合：mysql-5.7.15-winx64.zip
- 32ビット版Linuxの場合：mysql-5.7.15-linux-glibc2.5-i686.tar.gz
- 64ビット版Linuxの場合：mysql-5.7.15-linux-glibc2.5-x86_64.tar.gz

2. 「1.3.2　Windows上でZipファイルでのインストール」のリスト2と「1.4.3　Linux上でtarファイルでのインストール」のリスト11を参考に、Windows上ならZipファイル、Linux上ならtarファイルを利用してインストールします。なお、以降の作業でrootユーザーのパスワードなしで作業を行うため、--initialize-insecureオプションを利用します。

3. WindowsでのSTARTコマンド、Linuxでの&を付け忘れて起動した場合は、別のコンソールウィンドウを開いて以降の作業を継続してください。なお、Linux環境では、Ctrl+Zでサスペンドし、bgコマンドを使用してバックグラウンドで実行させることもできます。

レッスン

2

MySQL サーバー アーキテクチャ概要

このレッスンでは MySQL サーバーの製品概要とアーキテクチャを学習します。MySQL サーバーのアーキテクチャ上の最大の特徴は、表ごとに機能や特徴を変えることができる独自のストレージエンジン機能を持っていることです。

2.1 MySQLサーバーのアーキテクチャ

MySQLサーバーは、OS上では1つのプロセスとして動作し、クライアントから実行されたSQL文の処理やディスク上のデータの読み書きは内部の複数のスレッドが担当する「シングルプロセスマルチスレッド型」です。内部の実装は、接続の管理やSQL文の構文解析などを行う部分と、データやトランザクションの管理を行う部分の2層構造になっています。後者は「プラガブル・ストレージエンジン機能」と呼ばれ、用途に応じてどのストレージエンジンを利用するか、表ごとに選択が可能です。また、標準で含まれている以外に、プラグインとしてストレージエンジンも追加できる構造になっています。MySQLサーバーの運用管理やパフォーマンスチューニングのためにも、各ストレージエンジンが持つ機能や特徴の理解が重要です。

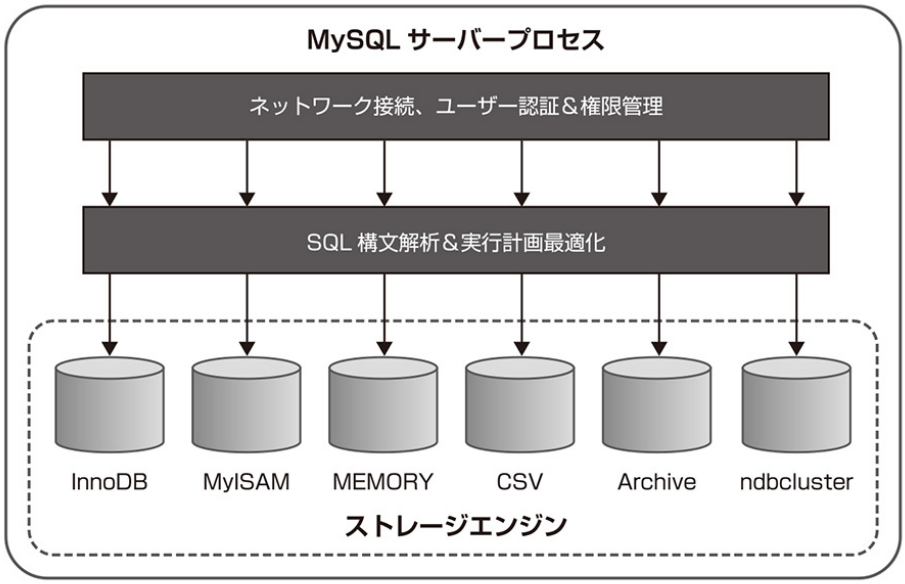


図1 2層構造となるMySQLサーバーのアーキテクチャ

2.1.1 ネットワーク接続管理

クライアントアプリケーションとMySQLサーバーの間は、通常TCP/IPで接続します。MySQL 5.7では基本的にSSLで通信の暗号化を行うようになっています※1。

2.1.2 ユーザー認証&権限管理

MySQLサーバーでは、許可に関する情報であるアクセスコントロールリスト（ACL）をベースに、ユーザーアカウントとデータベースオブジェクトへのアクセスの認証や権限の管理を行っています。商用版の MySQL Enterprise Edition では、 Lightweight Directory Access Protocol（LDAP）やKerberos Authentication（ケルベロス認証）、Windows Active DirectoryなどMySQLサーバーの外部で管理されるユーザー情報によって認証を行うことも可能です。ユーザーアカウントや認証、権限に関してはレッスン11、12のセキュリティの章で紹介します。

MySQLサーバーでは、新規の接続が行われると処理を担当するスレッドが生成され、ユーザー認証後にSQL文が発行されるとすぐに実行します。接続が切断されると処理スレッドは破棄されます。ただし、スレッドのプール（スレッドキャッシュ機能）を利用している場合には、スレッドは破棄されずにプールに格納され、次の接続時に再利用されることでスレッド生成や破棄のオーバーヘッドを削減できます。

2.1.3 SQL構文解析 & 実行計画最適化

実行されたSQL文を処理するために、データベースサーバーではパーサーによって構文解析を行って内部のAPIに変換し、オプティマイザによって最も効率的なデータへのアクセス経路（アクセスパス）を導き出す最適化を行います。

MySQL 5.7では、古い実装のパーサーのリファクタリングを行いました。オプティマイザは、処理の重みを見積もる際、個別の要素での処理負荷を数値化したコストを、従来のハードコーディングから設定パラメータとして可変にするなどの改良を行っています。

2.1.4 キャッシュ

データベースサーバーが稼働するハードウェア上で、処理のボトルネックとなりがちなのが、ディスクやストレージです。このためMySQLサーバーに限らずデータベースサーバーでは、ディスクやストレージへのアクセスを削減するために、データをメモリー上に蓄えるキャッシュ（またはバッファとも呼ばれる）の仕組みを持っています。MySQLサーバーでは、ストレージエンジン固有のキャッシュや、サーバー全体で利用するキャッシュなど複数の仕組みが用意されています。主なキャッシュについてはレッスン13、14のパフォーマンスチューニングの章で紹介します。

2.1.5 ストレージエンジン

ストレージエンジンの役割には、大きく分けて、データフォーマットの定義、データ永続化、インデックス管理、トランザクション管理、ロックと排他制御があります。さらに、InnoDBストレージエンジンの全文検索機能や自動クラッシュリカバリ機能などストレージエンジン固有の機能があります。MySQLサーバーでは、複数のストレージエンジンからアプリケーションに最適なものを選択できることがメリットですが、MySQL 5.7ではInnoDBに対する改善や機能追加が数多く行われています。また、InnoDBは多くのアプリケーションで必要となるACID特性を持ったトランザクションを、MySQL 5.7で唯一サポートしているストレージエンジンです。特別な理由がない限り、標準のInnoDBを利用しておけば十分なケースがほとんどです。以前は、参照性能が高いなどの理由でMyISAMストレージエンジンが利用されることもありましたが、実装が古いため、CPUコアが複数ある環境では性能が出ないこともあり、最近では利用者がほとんどいない状態です。

主なストレージエンジンと特徴は表1の通りです。※2

表1 代表的なMySQLのストレージエンジン

ストレージエンジン	特徴
InnoDB	MySQLのデフォルトのストレージエンジン。トランザクション対応、外部キーサポート、クラッシュリカバリ機能
MyISAM	以前のデフォルトのストレージエンジン。トランザクション非対応、テーブルレベルロック
MEMORY	指定した表をインメモリーデータベース化。アクセス頻度の高いデータの参照用
Archive	データの追加と参照のみ可能。データを自動的に圧縮。ログ蓄積や監査用のテーブルなどに利用
ndbcluster	MySQL Cluster※2のベースとなる技術。リモートのサーバーにデータを多重化して格納。トランザクション対応
CSV	データファイルの形式がCSV。ログやフラットファイルデータをSQLで分析する用途など
Blackhole	NULLデバイスのように書き込んだデータをすべて廃棄

現在稼働中のMySQLサーバーで利用可能なストレージエンジンのリストは次のコマンドで確認できます（リスト1）。

リスト1　利用可能なストレージエンジンを確認

```
mysql> SHOW ENGINES;
```

2.2 ストレージエンジンの使い方

ストレージエンジンの指定は、テーブル作成時のCREATE TABLE文のテーブルオプションで行います（リスト2）。

リスト2　InnoDBストレージエンジンを利用するテーブルを作成

```
mysql> CREATE TABLE t (i INT) ENGINE = InnoDB;
```

指定がない場合は、システムのデフォルトを利用します。何も設定していない場合、MySQL 5.5以降ではInnoDBが利用されます。明示的にデフォルトのストレージエンジンを指定する場合には、MySQLサーバーのシステム変数default_storage_engineにストレージエンジンを設定します。

既存の表のストレージエンジンを変更するには、ALTER TABLE文を利用します（リスト3）。

リスト3　MEMORYストレージエンジンを利用するテーブルに変更

```
mysql> ALTER TABLE t ENGINE = MEMORY;
```

このように非常にシンプルなコマンドでストレージエンジンの変更ができてしまいますが、内部的には変更したい定義の空のテーブルを作成し、既存の表からデータをロードするため、データサイズが大きい場合には処理時間がかかる点に注意が必要です。

表に設定されているストレージエンジンの確認は、SHOW CREATE TABLE 文またはSHOW TABLE STATUS 文 を利用します（リスト4）。

リスト4　InnoDBを利用している表の各コマンドの出力例

```
mysql> SHOW CREATE TABLE City ＼G
*************************** 1. row ***************************
       Table: City
Create Table: CREATE TABLE `City` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

　また、メタデータ管理スキーマであるINFORMATION_SCHEMA のTables表でも確認は可能です（リスト5）。

リスト5　INFORMATION_SCHEMAのTables表での確認方法の例

```
mysql> SELECT TABLE_NAME, ENGINE FROM TABLES
    -> WHERE TABLE_NAME LIKE 'City';
+------------+--------+
| TABLE_NAME | ENGINE |
+------------+--------+
| City       | InnoDB |
+------------+--------+
1 row in set (0.00 sec)
```

2.3 InnoDBストレージエンジン

MySQL 5.5以降での標準のストレージエンジンが、InnoDBストレージエンジンです。InnoDBは、当初フィンランドのInnobase社によって開発され、MySQL社に供給されていましたが、2005年にはInnobase社がオラクル社によって買収されました。その後、サン・マイクロシステムズ社の買収を通じてMySQLの開発部隊もオラクル社の一部となったため、先に買収されていたInnoDBの開発部隊と合併され、1つの組織として開発が進められることになりました。

InnoDBは、ACID特性に沿ったトランザクションをサポートしており、ANSI/ISOで定義されている4つのトランザクション分離レベルをすべてサポートしています（デフォルトはREPEATABLE READ）。ロックの粒度は、行レベルロックで、デッドロック検知も持っています。また、MySQL 5.7のストレージエンジンとして、外部キーをサポートしているのはInnoDBだけです。

2.3.1 InnoDBのデータファイル

InnoDBを利用している表には、表ごとに1つの表領域ファイル（拡張子.ibd）が作成され、そこにデータとインデックスの情報が格納されます。InnoDBや各表のメタデータは、共有表領域ファイルに格納されます。システム変数innodb_file_per_tableの値をOFFに設定すると、MySQL 5.5以前と同様に、すべての表のデータとインデックスを共有表領域ファイルに格納する方式に変更されます。

MySQL 5.7では、General Tablespaceという仕組みが追加され、表ごとに格納先の表領域ファイルを指定できるようになっています。いずれの形式の場合にも、InnoDBのメタデータを格納するシステム表領域やトランザクションによる変更前のデータのコピーを保持するUNDOログには、独立した表領域ファイルが作成されます。

2.3.2 InnoDBのトランザクション管理

トランザクションの内容は、実行されるたびにInnoDBのログバッファというメモリー上の領域に格納されます。コミット時にログバッファ内のトランザクション内容を記録するファイルがInnoDBログです。コミット時には、InnoDBログのみが変更され、メモリー上のバッファプールという領域にキャッシュされたデータのみが変更されます。変更されたキャッシュ上のデータは、チェックポイントという一定間隔でファイルに反映されます。これによりコミット時のディスクIOを削減して応答性能を高めつつ、行われた変更処理は確実にディスクに記録しておくことができます。このようにデータファイルの変更に先行して書き込まれるログは一般的にWrite Ahead Log（WAL）と呼ばれています。

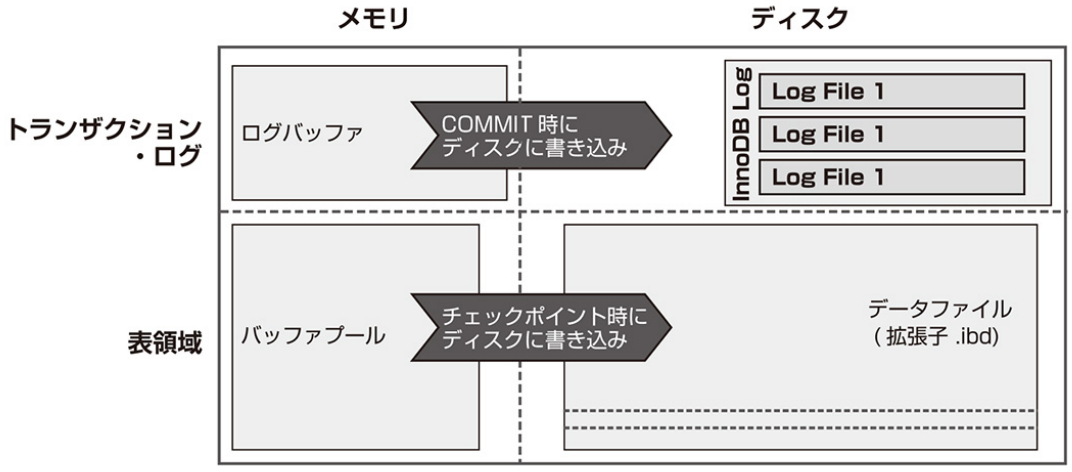


図 2　InnoDB のトランザクション管理

2.4 MEMORYストレージエンジン

特定のテーブルをインメモリーデータベースのように利用できるのは、MEMORYストレージエンジンです。データおよびインデックスはメインメモリー上に配置し、ディスク上にはテーブル定義ファイル（拡張子.frm）のみ存在します※3。

データがメモリーに格納されるため、処理性能は高速です。ただし、トランザクションはサポートせず、BLOB型およびTEXT型は格納できません。インデックスのアルゴリズムは、一意検索に適したHASHインデックス、または汎用性のあるBTREEが選択可能です。

コード1　MEMORYストレージエンジンの2種類のインデックスを作成

```
# ハッシュインデックスの作成例
CREATE TABLE lookup
      (id INT, INDEX USING HASH (id))
      ENGINE = MEMORY;

# Bツリーインデックスの作成例
CREATE TABLE lookup
      (id INT, INDEX USING BTREE (id))
      ENGINE = MEMORY;
```

MEMORYストレージエンジンは、MySQLサーバーが停止するとデータが消える仕様になっているため、データの永続化先の表には利用できません。一方で、高速なデータ参照先やSQL文も使えるキャッシュのような使い方に向いています。またディスク上にデータを蓄積する

ストレージエンジンを使った表にトリガーを追加することにより、データの永続性とMEMORYストレージエンジンによる高速な参照を両立させる工夫も可能です。なお、MySQLサーバーの起動時には、システム変数のinit-fileで指定するファイル内にデータをロードするコマンドを記載しておくことが可能です（リスト6）。

リスト6　トリガーによるInnoDBストレージエンジンの表とMEMORYストレージエンジンの表の連携

```
# InnoDBストレージエンジンを利用したTbl_I表を作成
mysql> CREATE TABLE Tbl_I (id INT(8), val_A INT(8), val_B INT(8)) ENGINE = InnoDB;

# MEMORYストレージエンジンを利用したTbl_I表を作成
mysql> CREATE TABLE Tbl_M (id INT(8), val_Sum INT(8)) ENGINE = MEMORY;

# Tbl_I表に値を挿入するとTbl_Mにval_Aとval_Bの合計値を格納するトリガーを作成
mysql> CREATE TRIGGER I_to_M AFTER INSERT ON Tbl_I FOR EACH ROW
    -> INSERT INTO Tbl_M VALUES (NEW.id, NEW.val_A + NEW.val_B);

# Tbl_I表に値を挿入
mysql> INSERT INTO Tbl_I VALUES (1, 10, 20);

# Tbl_I表の値を確認
mysql> SELECT * FROM Tbl_I;
```

```
+------+-------+-------+
| id   | val_A | val_B |
+------+-------+-------+
|    1 |    10 |    20 |
+------+-------+-------+

# Tbl_M表の値を確認
mysql> SELECT * FROM Tbl_M;
+------+---------+
| id   | val_Sum |
+------+---------+
|    1 |      30 |
+------+---------+
```

2.5 設定ファイル

MySQLサーバーやクライアントプログラムの設定オプションは、起動時のコマンドラインオプションか設定ファイルで指定できます。リファレンスマニュアル等では、MySQLサーバーの設定値をシステム変数と呼んでいます。設定ファイルを使うことで、サーバーやクライアントプログラムを起動するたびにコマンドライン上でオプションを指定する必要がなくなり、作業の手間とミスを削減することが期待できます。また、MySQLサーバーをOSのサービスとして登録して起動する場合には、コマンドラインオプションでシステム変数の指定ができないため、設定ファイルの利用が必須です。

標準の設定ファイル名は、Windowsではmy.ini、LinuxやSolarisではmy.cnfとなっていますが、任意のファイル名にすることが可能です。設定ファイルには、MySQLサーバーのデータファイルやログの格納先、各種メモリーサイズ、MySQLサーバーやクライアントプログラムで利用するTCP/IPポート、SSLのネットワーク設定などの情報を格納します。設定ファイルに記述するオプションは、大括弧でプログラム名などを囲んだグループと呼ばれる行から始まるカテゴリで構成されます。通常、グループ名はオプションのグループが反映するプログラムの名前です。MySQLサーバーの設定は[mysqld]の行以降に記述します。[client]はクライアントプログラム共通の設定を記述するグループです。

コード2　設定ファイルの例

```
# MySQLサーバーの設定
[mysqld]
```

```
# InnoDBストレージエンジンのデータバッファ領域のサイズ 1GB
innodb_buffer_pool_size = 1024M

# ソート用のメモリー領域 1MB
sort_buffer_size = 1M

# データファイルの格納先ディレクトリ
datadir = /u01/data

# Windowsのディレクトリの区切りの＼は/(スラッシュ)で表現
# ディレクトリ名などに空白を含む場合は引用符で囲む
#basedir="C:/Program Files/MySQL/MySQL Server 5.7/Data"

# TCP/IPポート番号
port = 3306

#同時最大接続数
max_connections = 4096

# MySQLサーバーの文字コードをUTF8(最大長4バイト)
character_set_server = utf8mb4

# クライアントプログラム共通の設定は[client]の項目で指定
[client]
# 文字コードをUTF8(最大長4バイト)
default_character_set = utf8mb4
```

MySQLサーバーは、--defaults-fileオプションで明示的に利用する設定ファイルが指定されていない場合は、起動時に自動的にいくつかの場所に設定ファイルが置かれていないか検索します。どの設定ファイルの項目を、どの順番で検索するかは、mysqldのヘルプの冒頭箇所に表示されます（リスト7）。

リスト7　MySQLサーバーが検索する設定ファイルの確認（ヘルプ内の該当部分のみ抜粋）

```
# Windowsの場合の出力例
C:＼>mysqld --verbose --help

Default options are read from the following files in the given order:
C:￥windows￥my.ini C:￥windows￥my.cnf C:￥my.ini C:￥my.cnf C:
￥mysql￥mysql-5.7.12-winx64＼my.ini C:￥mysql￥mysql-5.7.12-
winx64￥my.cnf

# Linuxの場合の出力例
$ mysqld --verbose --help
Default options are read from the following files in the given order:
/etc/my.cnf /etc/mysql/my.cnf
/usr/local/mysql/etc/my.cnf~/.my.cnf
```

実際の運用時には、読み込ませる設定ファイルを明示的にすることをおすすめします。自動的に設定ファイルを検索させると、想定外の設定ファイルを読み込んで起動してしまい、誤った設定値で動作してしまう可能性があるからです。

MySQLサーバーが読み込む設定ファイルは、起動時のコマンドラインオプション--defaults-fileで指定します。設定ファイルに記述されていない項目は、プリコンパイルされた設定値を使用します。もし、同じ設定項目が複数回記述されていて、異なる値が設定されていた場合は、後に書かれた値が使用されます。設定ファイルとコマンドラインオプションで同じ設定項目に異なる値が設定されていた場合は、コマンドラインオプションの値が使われます。

2.5.1 設定の確認

設定ファイルで指定した設定値がMySQLサーバーに反映されているかを含め、MySQLサーバーがどのような設定で動作しているかを確認するには、SHOW VARIABLESコマンドを利用します（リスト8）。

リスト8　MySQLサーバーに設定されている値を確認

```
mysql> SHOW GLOBAL VARIABLES;
# LIKE句で絞り込んで特定の設定項目の値を確認 %をワイルドカードとして利用可能
mysql> SHOW GLOBAL VARIABLES LIKE 'character_set_%';
```

Column　MyISAMストレージエンジン

MyISAMストレージエンジンは、MySQL 5.1以前のデフォルトのストレージエンジンでした。MyISAMは非常にシンプルで軽量なストレージエンジンです。MySQL 5.7でも、ユーザーや権限管理のシステム系のテーブルにMyISAMが利用されています。

MyISAMはシンプルで軽量な実装でしたが、トランザクションに対応していないため、データの変更中にMySQLサーバーに障害が発生するとデータが破損するリスクがあります。更新処理の際は、テーブルレベルロックのため同時実行性に大きな制約があります(InnoDBは行レベルロックです)。また、MyISAMは「軽い」ので性能が良いといわれることもありましたが、アーキテクチャが古いため、CPUのコア数を増やしても処理スループットが伸びないか、場合によっては低下してしまうこともありました。

MyISAMにはInnoDBにはない機能が複数ありましたが、現在のInnoDBには、同様またはより進化した機能が実装されています。以下は、InnoDBにはなくMyISAMにはあった主な機能の一覧と、InnoDBに実装されたMySQLサーバーのバージョンおよびその機能名や特徴です。

MyISAM にあった機能	InnoDB での機能名・特徴と実装されたバージョン
OS のコマンドでデータファイルを他の MySQL サーバーにコピー	MySQL 5.6、InnoDB トランスポータブル表領域
地理空間 (Geospatial) インデックス	MySQL 5.6、その後 MySQL 5.7 で Boost.Geometry のライブラリを採用し機能追加
全文検索インデックス	MySQL 5.6、その後 MySQL 5.7 で MyISAM では非対応だった日本語に対応
テーブル圧縮	MySQL 5.1、MyISAM では圧縮後に参照専用となるが InnoDB は更新可能

このような背景から、特別な理由がない限りはMyISAMを積極的に採用する機会は少なくなっています。

また、MyISAMを使った複数のテーブルを擬似的にパーティションのように扱うMRG_MYISAMストレージエンジンも、MySQLサーバー本体にパーティショニング機能が実装されたことで利用機会はほぼなくなっています。

MySQLサーバーの開発チームの方向性も、InnoDBを中心とした機能強化となっており、MySQL 5.7の次のメジャーバージョンであるMySQL 8.0 ではシステム系のテーブルにも InnoDB を採用することとなっています。

2.6 演習

このレッスンでは、MySQLサーバーのアーキテクチャ、ストレージエンジンの役割、設定ファイルについて学習しました。ここでは設定ファイルについて演習問題で確認しましょう。

1. tarを展開したディレクトリにmy.cnfという名称で上記設定ファイルを作成する

ヒント：設定ファイルや内容についてはレッスン2の「2.5　設定ファイル」を参照

「データファイルの格納先ディレクトリ」を表すdatadirは、Zipファイルまたはtarファイルを展開したディレクトリにあわせる

例）ファイルの展開先がC:￥mysql￥mysql-5.7.12-winx64の場合は、C:￥mysql￥mysql-5.7.12-winx64￥data

ファイルの展開先が/home/mysql/mysql57の場合は、/home/mysql/mysql57/data

2. 作成した設定ファイルを指定してMySQLサーバーをコマンドラインから起動する

ヒント：レッスン1の「1.3.2　Windows上でZipファイルでのインストール」のリスト3または「1.4.3　Linux上でtarファイルでのインストール」のリスト12を参考に、--defaults-fileオプションで設定ファイルを指定

3. 指定した設定値が反映されていることを確認する

ヒント：レッスン2の「2.5.1 設定の確認」のリスト8を参照

4. 現在稼働中のMySQLサーバーで利用可能なストレージエンジンを確認する

ヒント：レッスン2の「2.1.5 ストレージエンジン」のリスト1を参照

解説

3. 指定した設定値の確認にはSHOW GLOBAL VARIABLES;コマンドを使います。LIKE句を使わない時は全設定値を表示します。LIKE句に設定値を指定すると、特定の設定値だけを絞り込むことができます。出力が横長になって見づらい場合は、;（セミコロン）の代わりに￥G（半角円サインと半角大文字のG）または＼G（半角バックスラッシュと半角大文字のG）を使用します。

```
# 全設定値を出力
mysql> SHOW GLOBAL VARIABLES＼G
*************************** 1. row ***************************
Variable_name: auto_increment_increment
        Value: 1
*************************** 2. row ***************************
Variable_name: auto_increment_offset
        Value: 1

<中略>

*************************** 505. row ***************************
Variable_name: wait_timeout
        Value: 28800
505 rows in set (0.01 sec)
```

```
# 同時最大接続数の確認
mysql> SHOW GLOBAL VARIABLES LIKE 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 4096  |
+-----------------+-------+
1 row in set (0.01 sec)

# 文字コード関連の設定をワイルドカード % を使って確認
SHOW GLOBAL VARIABLES LIKE 'character_set_%'￥G

<出力省略>
```

4. 利用可能なストレージエンジンの確認にはSHOW ENGINESコマンドを使います。

```
mysql> SHOW ENGINES＼G
*************************** 1. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
```

```
   Savepoints: YES
*************************** 2. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO  Savepoints: NO

<中略>

*************************** 9. row ***************************
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
9 rows in set (0.01 sec)
```

※1 Windows上では、OpenSSLのセットアップと手動での鍵や証明書の作成が必要。Linux上ではインストール形態によっては手動での作業が必要。詳細はセキュリティのレッスン（レッスン12）で解説します。

※2 MySQL ClusterはMySQLサーバーを1コンポーネントとして含む分散型高速データベースクラスタ。アーキテクチャや運用方法、またリリースサイクルなどは通常のMySQLサーバーとは異なる独立した製品として開発されています。

※3 テーブル定義ファイルはどのストレージエンジンを利用している表にも必ず存在します。

レッスン

3

MySQL サーバーの主な機能と設定オプション

このレッスンでは MySQL サーバーの主要な機能と
その代表的な設定オプションについて学習します。

3.1 mysqld MySQLサーバープログラム

MySQLサーバー本体にあたるプログラムが「mysqld」です。mysqldは、データベースを管理し、各種クライアントからの接続を受けてデータベースにアクセスします。

3.1.1 ディレクトリ設定

MySQLサーバーのディレクトリに関する主なシステム変数は表1の通りです

表1 MySQL サーバーのディレクトリ関連システム変数

システム変数	意味
basedir	MySQL インストールディレクトリ。各ディレクトリ関連の設定値が相対パスの場合はこのディレクトリを基準とする
datadir	データファイルを配置するデータディレクトリ
innodb_data_file_path	InnoDB のシステム表領域の各ファイルとサイズ。自動拡張設定の指定も可能
innodb_data_home_dir	InnoDB のシステム表領域の各ファイルのディレクトリ

basedirを指定しない場合、以下がデフォルト値となります。これら以外の場所にインストールした場合は明示的に指定が必要です。

- Windows上　C:￥Program Files￥MySQL￥MySQL Server 5.7
- Linux上　/usr/local/mysql

MySQL 5.7のデフォルト設定であるinnodb_file_per_table有効時には、innodb_data_home_dirの設定は各テーブルのデータファイルには影響しません。innodb_file_per_table無効時には、innodb_data_file_pathで指定した表領域ファイルに各テーブルのデータも格納されます。

3.1.2 接続設定

MySQLサーバーに別のマシンなどのクライアントプログラムからリモートでアクセスする場合には、TCP/IP接続を使用します。

同一OS上でローカル接続する場合は、OSによって接続方法が異なります。Linuxを含むUnix系OSでは、ソケットファイル経由（システム変数名：socket）で高速に通信が可能です。Windowsでは、ローカル接続の場合にも基本的にはTCP/IP接続となります。MySQLサーバーの設定を変更すれば共有メモリー（システム変数名：shared_memory）や名前付きパイプ（システム変数名：enable-named-pipe）も使用できますが、現在はあまり一般的ではありません。

システムとしてMySQLサーバーに対するリモートアクセスを必要としていない場合は、ソケットファイルなどのローカル接続のみを有効とし、skip-networkingオプションを有効にしてTCP/IP接続を利用しない設定にできます。特にユーザー認証を行わずにデータベースへのアクセスを可能にするskip-grant-tablesを有効にして特殊なメンテナンスを行う場合などには、skip-networkingオプションの利用が強く推奨されます。

3.1.3 メモリー

MySQLサーバーが利用するメモリーサイズの設定は、次の2つに分類できます。

- データのソート用の領域など各クライアント接続の処理を行うスレッドごとに割り当てられるメモリーサイズの設定
- ディスク上のデータをキャッシュしておくなどサーバー全体で利用するメモリーサイズの設定

メモリーサイズに関する設定の多くは、必要に応じてMySQLサーバーの稼働中に動的に変更可能です。各パラメータの役割や設定値の見積もり方などはパフォーマンスチューニングに関するレッスン13とレッスン14で紹介します。

3.1.4 文字コード

MySQLでは、文字コードの設定は「Character Set」と呼ばれています。MySQLでは、MySQLサーバー全体、各データベース、各テーブル、テーブルの各行、クライアントなど各所で文字コードを設定可能です。MySQLサーバーやクライアントの文字コードを設定ファイルで指定する例は、レッスン2で紹介したサンプルの設定ファイルの後半に記載されています（コード1）。

コード1　サンプル設定ファイル内の文字コード関連設定

```
# MySQLサーバーの設定
[mysqld]
# MySQLサーバーの文字コードをUTF8（最大長4バイト）
character_set_server = utf8mb4

# クライアントプログラム共通の設定は[client]の項目で指定
[client]
# 文字コードをUTF8（最大長4バイト）
default_character_set = utf8mb4
```

MySQLは、デフォルトでは半角英数字と記号のみを扱う「latin1」という文字コードを使う設定になっています。日本語を含むデータを取り扱う場合は、サーバーおよびクライアントの文字コードをデフォルトのlatin1から変更する必要があります。日本語を利用する際に多く使われてきた文字コードは「cp932」や「eucjpms」ですが、最近ではUNICODE（UTF-8）の利用が一般的になっており、今後は

「utf8mb4」の利用が多くなると考えられます。MySQL 5.7の次のメジャーバージョンであるMySQL 8.0ではデフォルトの文字コードをutf8mb4にすることが予定されています。

- **URL：MySQL Server Team Blog: Planning the defaults for MySQL 5.8**
  http://mysqlserverteam.com/planning-the-defaults-for-mysql-5-8/

MySQLの日本語関連文字コードは表2の通りです。

表 2　MySQL の日本語関連文字コード

文字コード名	1 文字当たりの最大バイト数	説明
utf8mb4	4	UTF-8 (BMP 外サロゲートペアに対応。JIS X 0213:2004 文字セットを含む)
cp932	2	シフト JIS (JIS X 0208:1997+NEC 特殊文字・IBM 拡張文字)
eucjpms	3	EUC-JP (JIS X 0208:1997+NEC 特殊文字・IBM 拡張文字)
utf8	3	UTF-8 (uft8mb3 と同じ。BMP のみ対応した UTF-8 エンコーディング) ※非推奨
sjis	2	シフト JIS (JIS X 0208:1997) ※非推奨
ujis	3	EUC-JP (JIS X 0208:1997) ※非推奨

3.1.5 文字の照合順序（COLLATION）

文字コードと同時に考慮しなければいけないのが、文字の照合順序（COLLATION）です。utf8mb4で利用できるCOLLATIONは、各言語用に26種類あります。このうち名称の末尾がciのものは大文字小文字を区別しない（Case Insensitive）照合順序になっています。大文字小文字を区別するには、末尾がcs（Case Sensitive）のCOLLATIONか、文字バイナリコード値に従ってすべての文字の比較を行う末尾がbinのものを利用します。（リスト1）。

リスト1　utf8mb4で利用できるCOLLATIONの確認

```
mysql> SHOW COLLATION LIKE 'utf8mb4%';
+-----------------------+---------+-----+---------+----------+---------+
| Collation             | Charset | Id  | Default | Compiled | Sortlen |
+-----------------------+---------+-----+---------+----------+---------+
| utf8mb4_general_ci    | utf8mb4 |  45 | Yes     | Yes      |       1 |
| utf8mb4_bin           | utf8mb4 |  46 |         | Yes      |       1 |
| utf8mb4_unicode_ci    | utf8mb4 | 224 |         | Yes      |       8 |
| utf8mb4_icelandic_ci  | utf8mb4 | 225 |         | Yes      |       8 |
<中略>
| utf8mb4_unicode_520_ci | utf8mb4 | 246 |         | Yes      |       8 |
```

```
| utf8mb4_vietnamese_ci   | utf8mb4 | 247 |             | Yes         |
8 |
+-----------------------+---------+-----+---------+----------+---------+
26 rows in set (0.00 sec)
```

MySQL 5.7.11の時点では、utf8mb4のデフォルトのCOLLATIONであるutf8mb4_general_ciおよびutf8mb4_unicode_ciではUnicode 6.0で利用可能となった絵文字を正しく比較照合できないため、Unicodeの絵文字を含むデータの比較照合が必要な場合は、utf8mb4_unicode_520_ciまたはutf8mb4_binを利用します。次の例では、DISTINCT句の中でCOLLATEを指定して、同一文字と見なされるものの有無について確認しています（リスト2）。なお、Unicodeの絵文字に対応したフォントがない場合、Unicode絵文字が□で表示されてしまいます。Segoe UI SymbolやNoto Sans CJKなどのフォントを利用してください。

リスト2　COLLATIONによる文字比較の違い　Unicode絵文字

```
# 明示的に文字コードを指定してクライアントプログラムを起動
$ ./mysql -uroot --default-character-set=utf8mb4 world

mysql> DROP TABLE IF EXISTS Unicode_Test;
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Unicode_Test (id SERIAL, chars CHAR(1),
memo VARCHAR(32));
Query OK, 0 rows affected (0.11 sec)
```

```
# 大文字、小文字、絵文字、ひらがな、カタカナ、半濁音付き、半角カタカナを挿入
mysql> INSERT INTO Unicode_Test(chars, memo) VALUES
    -> ('a', 'Lower case A'),
    -> ('A', 'Upper case A'),
    -> ('🍣', 'Emoji Sushi'),
    -> ('🍺', 'Emoji Beer'),
    -> ('ハ', '30CF, Katakana HA'),
    -> ('パ', '30D1, Katakana PA'),
    -> ('は', '306F, Hiragana HA'),
    -> ('ﾊ', 'FF8A, Halfwidth Katakana HA');
Query OK, 8 rows affected (0.06 sec)
Records: 8  Duplicates: 0  Warnings: 0

# utf8mb4_bin: 文字バイナリコード値で比較 -> すべて別の文字として評価
mysql> SELECT DISTINCT chars COLLATE utf8mb4_bin AS bin FROM Unicode_Test;
+------+
| bin  |
+------+
| a    |
| A    |
| 🍣   |
| 🍺   |
| ハ   |
| パ   |
```

```
| は    |
| ハ    |
+------+
8 rows in set (0.00 sec)

# UCA(Unicode Collation Algorithm) v5.2.0の照合ルールで比較かつ大文字小文字を区別しない
mysql> SELECT DISTINCT chars COLLATE utf8mb4_unicode_520_ci AS unicode_520_ci FROM Unicode_Test;
+----------------+
| unicode_520_ci |
+----------------+
| a              |
| 🥥             |
| 👍             |
| ハ             |
+----------------+
4 rows in set (0.00 sec)

# Unicodeの絵文字など非サポートかつ大文字小文字を区別する
mysql> SELECT DISTINCT chars COLLATE utf8mb4_general_ci AS general_ci FROM Unicode_Test;
+------------+
| general_ci |
+------------+
| a          |
| 🥥         |
| ハ         |
```

```
| パ           |
| は           |
| ハ           |
+------------+
6 rows in set (0.00 sec)

# Unicodeの絵文字など非サポートのUCA v4.0.0の照合ルールで比較かつ大文字小文字を区別しない
mysql> SELECT DISTINCT chars COLLATE utf8mb4_unicode_ci AS unicode_ci FROM Unicode_Test;
+------------+
| unicode_ci |
+------------+
| a            |
| 🍆           |
| ハ           |
+------------+
3 rows in set (0.00 sec)
```

3.1.6 ログ

MySQLサーバーの運用やトラブルシューティングで利用するログには、4種類あります。これらのログは、システム変数で明示的にファイル名の一部を指定できます。ファイル名を指定しない場合は、ファイル名の一部にホスト名が利用されます（表3）。

表3 ホスト名が Svr01 でファイル名を明示的に指定しない場合のログファイル名の例

ログファイル名	ログの種類
Svr01.err	エラーログ
Svr01-bin.000001	バイナリログ。最後の数字部分はローテーションのたびにカウントアップ
Svr01-slow.log	スロークエリーログ
Svr01.log	一般クエリーログ

なお、MySQL Enterprise EditionのEnterprise Auditプラグインを利用すると、これらのログに加えて監査用ログを出力することができます。

エラーログ

MySQLサーバーの起動／停止や、サーバーサイドでのエラーに関連するログです。サーバーの問題解析に必要な情報がデフォルトで出力されています。

デフォルトの出力先はOSによって異なります。Windowsでは、データディレクトリのhost_name.errファイルおよびWindowsイベントログに出力されます。Linuxを含むUnix系OSの場合には、MySQLサーバーを起動したコンソールに出力され、ログファイルには記録されないため、エラーログが出力されているコンソールを閉じてしまった場合や、シェルスクリプトまたはサービスとして起動した場合にエラーログが確認できなくなります。そこで、システム変数

log_errorを設定し、ファイルに記録することを推奨します。MySQL 5.7では、log_syslogを有効に設定すると、syslogにエラーログを出力できます。

バイナリログ

実行されたトランザクションのうち、更新系の処理の内容と実行時刻などのメタデータを記録しているログです。バイナリログは、レッスン9、10で紹介するレプリケーションで必須となるほか、レッスン8で紹介するポイントインタイムリカバリのためにも必須となるため、多くの環境で出力する設定となっています。バイナリログはデフォルトでは出力されていないので、システム変数log_binで出力の有効化とファイル名の指定を行います。なお、MySQL 5.7からはバイナリログを出力する場合はserver_idでサーバーを識別する番号の設定が必須となっています。

MySQL 5.7では、バイナリログにはトランザクションによって変更された行イメージが記録されるROW（行ベース）形式がデフォルトになっています。行ベース形式ではバイナリログに記録されるデータ量が大きくなりがちですが、MySQL 5.6までのデフォルト設定だったSTATEMENT（文ベース）形式では、レプリケーション時にデータ不整合の可能性があった「非決定的な処理」の問題は回避できます。

- **URL：MySQL5.6リファレンスマニュアル：17.1.2.1 ステートメントベースおよび行ベースレプリケーションのメリットとデメリット**
  https://dev.mysql.com/doc/refman/5.6/ja/replication-sbr-rbr.html

バイナリログは名前の通りバイナリ形式でログが書かれていますが、mysqlbinlogコマンドでバイナリログを指定するとテキスト化が可能です。行ベース形式のバイナリログをテキスト化しても、行イメージが出力されるだけでそのままではどのようなSQL文で変更されたかわかりません（コード2）。しかし、-vオプションを付けてmysqlbinlogコマンドを実行すると、同じ更新結果を得るためのSQL文があわせて出力されます。行ベース形式のバイナリログをテキスト化した例は次の通りとなります（コード3）。

コード2　mysqlbinlogコマンドを-vオプションなしで実行した場合

```
# at 384
#160401 16:37:46 server id 1 end_log_pos 444 CRC32
0xb4cc9047 Write_rows: table id 116 flags: STMT_END_F

BINLOG '
SpjWVhMBAAAANAAAAIABAAAAAHQAAAAAAAEABXdvcmxkAAJ0dQA
DCA8PBIAAAAEGqPBsHg==
SpjWVh4BAAAAPAAAALwBAAAAAHQAAAAAAAEAAgAD//gJAAAAAAA
AAAFBDABVcHBlciBjYXNlIEFH
kMy0'
/*!*/;
```

コード3　mysqlbinlogコマンドを-vオプション付きで実行した場合

```
# at 384
#160401 16:37:46 server id 1 end_log_pos 444 CRC32
0xb4cc9047 Write_rows: table id 116 flags: STMT_END_F

BINLOG '
SpjWVhMBAAAANAAAAIABAAAAAHQAAAAAAAEABXdvcmxkAAJ0dQA
DCA8PBIAAAAEGqPBsHg==
SpjWVh4BAAAAPAAAALwBAAAAAHQAAAAAAAEAAgAD//gJAAAAAAA
AAAFBDABVcHBlciBjYXNlIEFH
kMy0
'/*!*/;
### INSERT INTO `world`.`tu`
### SET
### @1=9
### @2='A'
### @3='Upper case A'
```

スロークエリーログ

実行時間が指定した時間以上のクエリーを出力するログです。デフォルトでは出力されていませんが、処理時間が長い、問題となり得るSQL文を見つけることができるため、log_slow_queriesを設定して出力することを推奨します。スロークエリーログに関するシステム変数は以下の通りです（表4）。

表 4　スロークエリーログの設定に関するシステム変数

パラメータ名	概要
slow_query_log	スロークエリーログの出力の有無、およびログファイル名を指定
long_query_time	処理時間の閾値を秒単位で指定（デフォルトは 10 秒、0.5 と指定すれば 500 ミリ秒）
log_queries_not_using_indexes	処理時間が閾値未満の場合でも、インデックスを使っていない SQL 文をすべて出力
min_examined_row_limit	SQL 文の処理行数の閾値（デフォルトは 0 行）
log_slow_admin_statements	各管理系の SQL 文を記録しない（ALTER TABLE、ANALYZE TABLE、CHECK TABLE、CREATE INDEX、DROP INDEX、OPTIMIZE TABLE、REPAIR TABLE）

一般クエリーログ（または一般ログ）

クライアントからの接続および実行されたすべてのSQL文、各処理の実行時に内部的に発行されたコマンドを出力するログです。ログに記録する情報が多いため、デフォルトでは出力されていませんが、問題の解析などのために一時的に general_logを設定して出力することができます。以下は3つの一連の操作を行った際の一般ログの例です（コード4～6）。

コード4　ローカルホスト上のmysqlクライアントプログラムにてユーザー名scottでログイン

```
Time                 Id Command    Argument
160112 01:02:34      1 Connect      scott@localhost on
                     1 Query        select @@version_comment
limit 1
```

コード5　`USE world`コマンドを実行してカレントデータベースを切り替え

```
160112 01:02:45      1 Query        SELECT DATABASE()
                     1 Init DB      world
                     1 Query        show databases
```

```
                    1 Query         show tables
                    1 Field List    city
                   1 Field List    country
                    1 Field List    countrylanguage
```

コード6　SELECT文を実行

```
160112 01:03:12          1 Query    SELECT * FROM City LIMIT 1
```

注目すべきは、コード5のカレントデータベースの切り替えで、実行したコマンドよりも多くの内部コマンドが実行されているのがわかります。一般ログはこのように大量の情報を記録しますので、本番環境ではトラブルシューティングなどの際のみに利用するのが一般的です。

スロークエリーログと一般クエリーログの出力先

スロークエリーログと一般クエリーログの出力を有効にすると、デフォルトの設定ではそれぞれログファイルに出力されます。log_outputでログの出力先を次の3パターンから設定することができます。

1. ログファイルのみ
2. ログテーブルのみ
3. ログファイルとログテーブルの両方

ログテーブルは、MySQLのシステム系データベースmysqlにあり、スロークエリーログはslow_logテーブル、一般クエリーログはgeneral_logテーブルとなっています。sedやawkなどでログを分析するスクリプトを書く場合はログファイルを、SQL文でログの分析を行いたい場合はログテーブルを使うなどの使い分けが可能です。なお、ログテーブルはCSVストレージエンジンを利用しており、データファイルがCSV形式になっているため、データファイルをコピーしてExcelなどで集計や分析を行うことも可能になっています。

以下は、意図的にlong_query_timeの設定値を超えるようにスリープしたSELECT文を実行した際の、ログファイルに出力したスロークエリーログの例（コード7）と、テーブルに出力したスロークエリーログの例（コード8）です。

コード7　スロークエリーログをログファイルに出力した例

```
# Time: 2016-09-02T01:23:45.995674Z
# User@Host: root[root] @ localhost []  Id:       3
#  Query_time:  11.012354     Lock_time:  0.000000  Rows_sent:  1
Rows_examined: 0
SET timestamp=1474373406;
SELECT SLEEP(11);
```

コード8　スロークエリーログをログテーブルに出力した例

```
mysql> SELECT * FROM mysql.slow_log＼G
```

```
*************************** 1. row ***************************
      start_time: 2016-10-02 10:23:45.995674
       user_host: root[root] @ localhost []
      query_time: 00:00:11.012354
       lock_time: 00:00:00.000000
       rows_sent: 1 rows_examined: 0
                 db: last_insert_id: 0
       insert_id: 0
       server_id: 1
         sql_text: SELECT SLEEP(11)
       thread_id: 3
1 row in set (0.00 sec)
```

ログ出力設定の動的な変更

本番運用環境では、エラーログ、バイナリログ、スロークエリーログは常に出力しておき、一般クエリーログは開発環境やテスト環境ですべての処理内容の確認などの際に出力するのが一般的です。エラーログとバイナリログは、MySQLサーバーの起動時のみに出力の有無や設定を指定できますが、スロークエリーログと一般クエリーログはSETコマンドで動的に出力の有無や出力先を変更できます。

リスト3　SETコマンドで一般ログを一時的に有効にする

```
mysql> SET GLOBAL general_log = 'ON';
```

ログのローテーション

ログファイルのサイズが大きくなった場合などにログファイルの切り替えを行います。バイナリログは、FLUSH BINARY LOGSコマンドや、管理クライアントプログラムのmysqladminのflush-logsまたはrefreshサブコマンド、データのダンプを取得するクライアントプログラムmysqldumpを--flush-logsオプション付きで実行した際にログが切り替わり、ファイル末尾の数字がカウントアップします。

一方、エラーログ、スロークエリーログ、一般クエリーログは、ログのフラッシュを行ってもファイルを閉じて開くだけの動作になります。このため、まずログファイルの名前を変更し、FLUSH LOGSコマンドまたはmysqladminのflush-logsまたはrefreshサブコマンドでログをフラッシュします。なお、SQLコマンドの場合、すべてのログをフラッシュするFLUSH LOGSコマンドのほかに、ログを個別にフラッシュするlog_typeオプションが利用できます。

リスト4　Linuxのコマンドラインでスロークエリーログと一般クエリーログを切り替え

```
$ mv Svr01.log Svr01.old
$ mv Svr01-slow.log Svr01-slow.old
$ mysqladmin flush-logs
```

Column　**各種のストレージエンジン**

MySQLサーバーではこれまで解説してきたInnoDB、MEMORY、MyISAM以外のストレージエンジンも利用可能です。レッスン2の「ストレージエンジン」のリストに掲載してあります。ただしInnoDBとMySQL Clusterで利用されるndbcluster以外はトランザクションをサポートしていません。

この中でもCSVストレージエンジンは、一般クエリーログやスロークエリーログの出力先としてテーブルを選択した場合に、それぞれのログテーブルが利用しています。

CSVストレージエンジンを使ったテーブルは、データファイルにCSV（カンマ区切りのテキストファイル）を利用します。このファイルをコピーして利用することができます。行数などのメタデータを格納するファイルもテーブルごとに作成されます。作成したテーブルの列の定義と同じCSVファイルをデータファイルと入れ替え、データファイルを開き直すFLUSH TABLESコマンドを実行すると、CSVファイル内のデータをMySQLサーバー内で利用できます。

なおCSVストレージエンジンはインデックスとパーティショニングが利用できず、すべての列がNOT NULLである必要があるという制約があります。

ARCHIVEストレージエンジンは、データの追加と参照のみ可能で、変更や削除ができないテーブルを作ることができます。SELECT文、INSERT文、REPLACE文は実行可能ですがUPDATE文とDELETE文はエラーとなります。データは追加時に自動的にzlib圧縮して格納されます。アプリケーションからデータの変更がされない特性を生かしてログ蓄積や監査用のテーブルなどに利用され

ることがあります。DELETE文でのデータの削除はできませんが、パーティショニングが利用できるのでパーティション単位でデータを削除する運用が可能です。

Blackholeストレージエンジンは/dev/null（NULLデバイス）のように書き込んだデータをすべて捨てます。用途は「データを変更したトランザクション履歴は記録したいがデータは不要」という場合です。データを変更するとバイナリログにトランザクションが記録されます。これをレプリケーションに利用するケースやバックアップなどに利用することが想定されます。

FEDERATED ストレージエンジンは他のMySQLサーバーのテーブルに接続できるストレージエンジンです。ただし機能制約が多くデフォルトでは有効になっていません。

このほかバイナリ版には含まれていませんが、MySQLサーバーのソースコードパッケージにはストレージエンジンを開発するにあたってのサンプルとしてExampleストレージエンジンが用意されています。

3.2 演習

このレッスンではMySQLサーバーの主な機能と設定オプションについて学習しました。ここではサンプルの設定ファイルを利用して、文字コードについて確認しましょう。

1. レッスン2の「2.5　設定ファイル」にある設定ファイルの例（コード2）を使ってMySQLサーバーを起動し、クライアントプログラムから接続する

ヒント：MySQLコマンドラインクライアントの起動時には、レッスン3の「3.1.5　文字の照合順序（COLLATION）」のリスト2のように、コマンドラインクライアントが利用する文字コードの指定が必要

2. 現在利用可能な文字コードの一覧を次のコマンドで取得する

```
mysql> SHOW CHARACTER SET;
```

ヒント：レッスン3の「3.1.4　文字コード」および表2を参照

3. 現在稼働中のMySQLサーバーおよび接続するクライアントプログラムが利用している文字コードを次のコマンドで取得する

```
mysql> SHOW VARIABLES LIKE '%char%';
```

ヒント：レッスン2の「2.5.1　設定の確認」のリスト8を参照

解説

文字コードに関する演習の際は、OSやコマンドプロンプト、ターミナルコンソールの文字コードの設定、利用しているフォントがUnicode（UTF-8）に対応しているかを確認してください。

1. **クライアントプログラムが利用する文字コードの指定には、コマンドラインの引数で--default-character-setオプションを使用するか、「client」の項目で文字コードが指定してある設定ファイルを--defaults-fileで指定する方法があります。**

2. **SHOW CHARACTER SET;コマンドの出力には、各文字コードがデフォルトで使用する文字の照合順序（COLLATION）および、1文字あたりの最大バイト数が含まれているので確認しておいてください。ほかのSHOWコマンドと同様、LIKE句での絞り込みも可能です。**

```
mysql> SHOW CHARACTER SET LIKE 'utf8%';
+---------+--------------+--------------------+--------+
| Charset | Description  | Default collation  | Maxlen |
+---------+--------------+--------------------+--------+
| utf8    | UTF-8 Unicode | utf8_general_ci    |      3 |
| utf8mb4 | UTF-8 Unicode | utf8mb4_general_ci |      4 |
+---------+--------------+--------------------+--------+
2 rows in set (0.01 sec)
```

3. 文字化けを起こすことになるので、実際の利用時には推奨できませんが、例えば以下のコマンドのように、MySQLコマンドラインクライアントの文字コードを一切指定しない状態でMySQLサーバーに接続し、SHOW VARIABLES LIKE '%char%';を実行するとどうなるでしょうか。日本語のデータを追加した上で検索するとどうなるか試してみてください。

```
$ ./mysql -uroot world
```

レッスン

4

MySQL のクライアントプログラム

このレッスンでは、MySQL で使用できる各種のクライアントプログラムの機能と設定オプションについて学習します。

4.1 クライアントプログラム

MySQLのクライアントプログラムは、基本的にMySQLサーバーに接続してコマンドを実行しています。例えば、レッスン3の「バイナリログ」の項で出てきたmysqlbinlogコマンドは、ログファイルに直接アクセスするユーティリティのため、クライアントプログラムとは呼びません。クライアントプログラムに共通する代表的なコマンドラインオプションは以下の通りです（表1）。

表1 MySQL クライアントプログラム共通の主な設定オプション

オプション名	短縮型	デフォルト値	概要
--protocol	なし	TCP	通信プロトコルは TCP、SOCKET、PIPE、MEMORY のいずれかを指定
--host	-h	localhost	MySQL サーバーが稼働するマシンのホスト名または IP アドレス
--port	-P	3306	MySQL サーバーの TCP/IP のポート番号
--user	-u	なし	MySQL サーバーに接続するユーザー名
--password	-p	なし	MySQL サーバーに接続するユーザーのパスワード
--default-character-set	なし	latin1	クライアントプログラムが使用する文字コード
--compress	-c	なし	通信プロトコルの圧縮

オプション名と短縮形の比較は以下の通りです（リスト1）。ここではローカルホストの3306ポートに対して、rootユーザーで通信プロトコルの圧縮あり、文字コードはutf8mb4で接続している例です。

リスト1　MySQLコマンドラインクライアントのオプション名

```
# すべて長いオプション名で明示的に指定した場合
$ ./mysql --user=root --host=localhost --port=3306 --compress --default-character-set=utf8mb4

# 可能な部分は短いオプション名で明示的に指定した場合
$ ./mysql -u root -h localhost -P 3306 -c --default-character-set=utf8mb4
```

```
# 短いオプション名と値の間はスペースがなくても良い
$ ./mysql -uroot -hlocalhost -P3306 -c --default-character-set=utf8mb4

# 可能な部分はデフォルト値を利用した場合
$ ./mysql -uroot -c --default-character-set=utf8mb4
```

Windows上では、クライアントサーバー間は通常TCP/IPで接続します。Linuxの場合は、ローカルホスト上での接続にはデフォルトでUnixソケットが利用され、リモートのクライアントサーバー間はTCP/IPで接続します。また各クライアントプログラムからMySQLサーバーにはSSLを利用してセキュアな接続で通信を行うことができます。

以降で紹介するクライアントプログラムは、MySQLサーバーのインストール時にbinディレクトリに格納されます。Linux上にrpmでインストールした場合には、/usr/binに格納されています。

4.2 MySQLコマンドラインクライアントmysql

mysqlは、MySQLサーバーに接続して、SQL文やコマンドを実行するコマンドラインクライアントです。mysqlコマンドで利用可能なコマンドラインオプションは次のURLを参照してください。

- **URL：MySQL 5.6 Reference Manual ：4.5.1.1　mysqlのオプション**
  http://dev.mysql.com/doc/refman/5.6/ja/mysql-command-options.html
  http://dev.mysql.com/doc/refman/5.7/en/mysql-command-options.html

mysqlコマンドの利用を終了する際は、＼q（半角のバックスラッシュと小文字のQ）またはEXITまたはQUITと入力します。Windows上ではバックスラッシュを￥（半角円サイン）に置き換えてください。

4.2.1 mysqlのインタラクティブモードとバッチモード

mysqlコマンドでのSQL実行結果の表示は、利用方法によって異なります。毎回SQL文を入力する方法などを「インタラクティブモード」と呼び、罫線で区切られた表による出力形式になります。

一方で、SQL文が格納されたファイルを読み込む方法を「バッチモード」と呼び、列をタブ区切りで表現した出力形式となります。バッチモードでも--tableまたは-tオプションを付けることで、罫線で区切られた出力形式を利用できます。ほかにもHTML形式で出力する--htmlまたは-hオプション、XML形式で出力する--xmlまたは-xオプションも利用できます。モードによる標準の出力形式の違いは次の通りです（リスト2～5）。

リスト2　インタラクティブモード（表形式）

```
$ ./mysql -uroot world
...
mysql> SELECT * FROM City WHERE District = 'Okinawa';
+------+---------+-------------+----------+------------+
| ID   | Name    | CountryCode | District | Population |
+------+---------+-------------+----------+------------+
| 1597 | Naha    | JPN         | Okinawa  |     299851 |
| 1723 | Okinawa | JPN         | Okinawa  |     117748 |
| 1764 | Urasoe  | JPN         | Okinawa  |      96002 |
+------+---------+-------------+----------+------------+
```

リスト3　バッチモード（タブ区切り）

```
$ ./mysql -uroot world -e "SELECT * FROM City WHERE District = 'Okinawa'"
ID      Name      CountryCode   District   Population
1597   Naha      JPN   Okinawa   299851
1723   Okinawa   JPN   Okinawa   117748
1764   Urasoe    JPN   Okinawa   96002
```

リスト4　バッチモード(HTML形式)

```
$ ./mysql -uroot world -H -e "SELECT * FROM City WHERE District = 'Okinawa'"
<TABLE          BORDER=1><TR><TH>ID</TH><TH>Name</TH>
<TH>CountryCode</TH><TH>District</TH>
<TH>Population</TH></TR><TR><TD>1597</TD>
<TD>Naha</TD><TD>JPN</TD><TD>Okinawa</TD>
<TD>299851</TD></TR><TR><TD>1723</TD>
<TD>Okinawa</TD><TD>JPN</TD><TD>Okinawa</TD>
<TD>117748</TD></TR><TR><TD>1764</TD>
<TD>Urasoe</TD><TD>JPN</TD><TD>Okinawa</TD>
<TD>96002</TD></TR></TABLE>
```

リスト5　バッチモード(XML形式)

```
$ ./mysql -uroot world -X -e "SELECT * FROM City WHERE District = 'Okinawa'"
<?xml version="1.0"?>

<resultset statement="SELECT * FROM City WHERE District = 'Okinawa'
```

```
" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <field name="ID">1597</field>
    <field name="Name">Naha</field>
    <field name="CountryCode">JPN</field>
    <field name="District">Okinawa</field>
    <field name="Population">299851</field>
  </row>

  <row>
    <field name="ID">1723</field>
    <field name="Name">Okinawa</field>
    <field name="CountryCode">JPN</field>
    <field name="District">Okinawa</field>
    <field name="Population">117748</field>
  </row>

  <row>
    <field name="ID">1764</field>
    <field name="Name">Urasoe</field>
    <field name="CountryCode">JPN</field>
    <field name="District">Okinawa</field>
    <field name="Population">96002</field>
  </row>
</resultset>
```

--executeまたは-eオプションでは、オプションの後に指定したSQL文だけを実行して終了します。この場合は標準の出力形式は表形式となります。また、実行結果は>を使ってファイルに格納することや、｜（パイプ）経由で他のコマンドに渡すことができます。

4.2.2 mysqlコマンドの実行時ヘルプとSQL構文の確認

mysqlコマンドを実行中に、＼qまたはHELPと入力すると、表示形式や文字コードの変更などコマンドについてのヘルプが利用可能です（リスト6）。

リスト6　mysqlコマンドの実行時ヘルプ

```
mysql> HELP
<略>

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (＼?) Synonym for `help'.
clear     (＼c) Clear the current input statement.
<略>

ego           (＼G) Send command to mysql server, display result vertically.
<略>

charset   (＼C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings  (＼W) Show warnings after every statement.
nowarning (＼w) Don't show warnings after every statement.

For server side help, type 'help contents'
```

リスト6に表示されている中でも利用頻度の高いものは、表示形式を変更する「＼G」と、警告が発生するたびに自動的に表示される「＼W」です。特に＼Gは、表形式にすると表示が横長になりすぎる時や、改行が挟まって読みにくくなってしまった場合に便利です（リスト7）。

リスト7　標準の表形式の出力と＼G利用時の比較

```
mysql> SELECT VERSION();
+-----------------------------------+
| VERSION()                         |
+-----------------------------------+
| 5.7.4-community |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> SELECT VERSION()＼G
*************************** 1. row ***************************
VERSION(): 5.7.4-community
1 row in set (0.00 sec)
```

また、SQL文の構文や利用可能な句、関数、データ型などの確認ができます。これらのヘルプは、HELPに続けてSQL文のキーワードを入力するか、HELP CONTENTSと入力するとカテゴリのリストが表示されます（リスト8）。

リスト8　UPDATE文の構文を確認

```
mysql> HELP UPDATE
Name: 'UPDATE'
Description:
Syntax:
Single-table syntax:

UPDATE [LOW_PRIORITY] [IGNORE] table_reference
        SET col_name1={expr1|DEFAULT} [, col_name2=
{expr2|DEFAULT}] ...
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
<略>
```

4.3 MySQLサーバーの運用管理に必要となるクライアントプログラムmysqladmin

MySQLサーバーの稼働状況の確認などに必要なのが、mysqladminコマンドです。このコマンドもMySQLサーバーに接続して処理を行います。mysqladminコマンド内で利用できる主なサブコマンドは以下の通りです（表2）。

表2　mysqladmin コマンドの主なサブコマンド

サブコマンド	概要
create	データベースを作成する
password	パスワードを変更する
processlist	MySQL サーバー上でアクティブな処理を確認する
status	稼働状況の概要を表示する
shutdown	MySQL サーバーを停止する

mysqladminコマンドの全サブコマンドおよびオプションは--helpオプションで確認できるほか、次のリファレンスマニュアルを参照してください。

- **URL：MySQL 5.6 Reference Manual：4.5.2 MySQLサーバーの管理を行うクライアント**
  http://dev.mysql.com/doc/refman/5.6/ja/mysqladmin.html
  http://dev.mysql.com/doc/refman/5.7/en/mysqladmin.html

MySQL 5.6までは、OSのサービスで運用していないMySQLサーバーを停止するにはmysqladminコマンドの利用が必要でしたが、MySQL 5.7ではSQLコマンドとしてSHUTDOWNが加わりました。

4.4 簡易ベンチマークツールmysqlslap

mysqlslapは、MySQLサーバーに対してクライアントアプリケーションの負荷をかけ、各処理フェーズでの処理時間を確認できる簡易的なベンチマークツールです。

mysqlslapでは、ベンチマークに使うSQL文をあらかじめ用意して負荷をかける方法と、自動的にSQL文を生成する方法が選択できます。mysqlslapの処理は、テーブル生成とテストに利用するデータをロードする準備フェーズ、負荷テストを行うフェーズ、そしてテーブルの削除などを行う3つのフェーズに分かれています。負荷テストでは、マルチスレッドで複数のクライアントからの同時アクセスをシミュレートできます。

mysqlslapでは、MySQLクライアントプログラム共通のオプションに加えて、多重度（-cオプションまたは--concurrencyオプション）や実行回数（-iオプションまたは--iterationsオプション）、利用するSQL文を指定してベンチマークテストを行います（リスト9）。

リスト9　SQL文を指定してmysqlslapを実行

```
$ ./mysqlslap -uroot --delimiter=";" --create="CREATE TABLE tbl_1 (id INT, name VARCHAR(32)); INSERT INTO tbl_1 VALUES (1, 'aaa')" --query="SELECT * FROM tbl_1" --concurrency=5 --iterations=10
Benchmark
        Average number of seconds to run all queries: 0.001 seconds
        Minimum number of seconds to run all queries: 0.001 seconds
        Maximum number of seconds to run all queries: 0.002 seconds
```

```
Number of clients running queries: 5
Average number of queries per client: 1
```

リスト9の例では、--createオプションでテーブルを作成し、1行だけ登録した後、全件取得するSELECT文を5多重で各スレッドから10回ずつ、計50回実行するベンチマークを行っています。

ユーザーがSQL文を用意する場合は、--createおよび--queryオプションを利用します。この2つのオプションの引数には、SQL文を直接指定することも、SQL文が含まれたテキストファイルを指定することもできます。ファイルの場合、実行する複数のSQL文を改行または--delimiterオプションで指定された区切り文字で分けておきます。

自動生成したSQL文を利用してベンチマークテストを行うには、-aオプション（または--auto-generate-sqlオプション）を利用します。--auto-generate-sql-load-typeオプションで指定できるSQL文の種類は、デフォルトではmixedモードとして、参照と更新を混在させたものになります。SQL文の種類は以下のものが選択可能になっています。

- ●mixed：半分はINSERT文、半分はSELECT文によるテーブルスキャン
- ●read：テーブルスキャンするSELECT文のみ
- ●write：INSERT文のみ
- ●key：主キーによるSELECT文
- ●update：主キーによるUPDATE文

テーブル定義や自動生成するSQL文は、以下のオプションで制御できます（表3～4）。

表 3　自動生成されるテーブルの定義

オプション名	概要
--auto-generate-sql-add-autoincrement	AUTO_INCREMENT の列を自動生成されたテーブルに追加
--auto-generate-sql-guid-primary	GUID(UUID の一種) ベースの主キーを自動生成されたテーブルに追加
--auto-generate-sql-secondary-indexes	セカンダリインデックス (主キー以外インデックス) の数を指定。デフォルト値は 0
--number-char-cols	VARCHAR データ型の列の数
--number-int-cols	INT データ型の列の数

表 4　自動生成される SQL 文の定義

オプション名	概要
--auto-generate-sql-execute-number	自動生成される SQL 文の数を厳密に指定
--auto-generate-sql-load-type	SQL 文の種類を指定 [mixed, update, write, key, read] デフォルト値は mixed
--auto-generate-sql-unique-query-number	検索を行う SQL 文の種類数を指定 デフォルト値は 10
--auto-generate-sql-unique-write-number	書き込みを行う SQL 文の種類数を指定 デフォルト値は 10
--auto-generate-sql-write-number	各スレッドで行う書き込み行数 デフォルト値は 100

表5のオプションで指定するベンチマークテスト前後に行う処理は、ベンチマークテストの処理時間には含まれません。

表 5　ベンチマークテスト前後に行う処理の指定

オプション名	概要
--pre-query	ベンチマークテスト前に実行する SQL 文の指定
--pre-system	ベンチマークテスト前に実行する OS コマンドの指定
--post-query	ベンチマークテスト後に実行する SQL 文の指定
--post-system	ベンチマークテスト後に実行する OS コマンドの指定

MySQLサーバー向けに作られたベンチマークテストとして特徴的なのが、SQL文を自動生成して、ベンチマークの際に使うテーブルの

ストレージエンジンを-e（または--engineオプション）で複数指定して比較できる点です（リスト10）。

リスト10　SQL文を自動生成してmysqlslapを実行

```
$ ./mysqlslap -uroot -a --auto-generate-sql-add-autoincrement --number-int-cols=3 --number-char-cols=2 --engine=InnoDB,MEMORY,MyISAM -c 5 -i 10
Benchmark
	Running for engine InnoDB
	Average number of seconds to run all queries: 0.978 seconds
	Minimum number of seconds to run all queries: 0.829 seconds
	Maximum number of seconds to run all queries: 1.279 seconds
	Number of clients running queries: 5
	Average number of queries per client: 0

Benchmark	Running for engine MEMORY
	Average number of seconds to run all queries: 1.653 seconds
	Minimum number of seconds to run all queries: 1.570 seconds
	Maximum number of seconds to run all queries: 1.740 seconds
	Number of clients running queries: 5
	Average number of queries per client: 0

Benchmark
	Running for engine MyISAM
	Average number of seconds to run all queries: 1.732 seconds
	Minimum number of seconds to run all queries: 1.541 seconds
	Maximum number of seconds to run all queries: 2.299 seconds
```

```
Number of clients running queries: 5
Average number of queries per client: 0
```

この例では、INTデータ型が3列、CHARデータ型が2列、AUTO_INCREMENT属性の列を持つテーブルに対して、mixedの処理を5多重で各スレッドから10回ずつ、計50回実行するベンチマークを行っています。この際に作成されるテーブルは、InnoDBストレージエンジン、MEMORYストレージエンジンとMyISAMストレージエンジンとして、ストレージエンジン間での性能比較を行っています。

Column **歴史に消えたストレージエンジン**

MySQL 5.5でInnoDBがデフォルトのストレージエンジンになった以降は、MySQLサーバー本体に含まれるストレージエンジンに変更はありません。一方でMySQLの歴史の中では現在使われなくなったストレージエンジンもあります。

MySQL 3.23でMyISAMがデフォルトとなるまでは、ISAMというMyISAMの祖先となるストレージエンジンが使われていました。その後、MySQL 4.1からはソースコードパッケージのみに含まれるようになり、MySQL 5.0で廃止となりました。

BDB（BerkeleyDB）ストレージエンジンは、MySQL 5.0まで利用可能だったトランザクション対応ストレージエンジンでした。BerkeleyDBはSleepy Cat社によって開発され、その後、同社の買収

を経て、現在はOracle Corporationのモバイル機器などをターゲットとした組み込みデータベースとなっています。

　FalconストレージエンジンはInnoDBに代わるトランザクション対応ストレージエンジンとして開発されましたが、完成されることはありませんでした。

　MySQL 5.1では、プラガブル・ストレージエンジン・アーキテクチャが導入され、新たなストレージエンジンを開発しやすくするAPIが整備されました。それにともない、Amazon S3にデータを格納するものやIBM DB2 for iと連携するもの、データウェアハウス用途に特化したもの、MyISAMをトランザクションさせようと試みたものなど、多彩なストレージエンジンがコミュニティや企業で開発されました。2007年春に米国で開催されたMySQL Conference&Expo Reportでは、これらのストレージエンジンをテーマに多数の講演が行われていました。

- **URL：Think IT （2007/5/16） 次期バージョンの姿が見え、より飛躍するMySQLを追う**
  https://thinkit.co.jp/cert/article/0705/4/2.htm

　残念ながら、このころ開発されたストレージエンジンは、データウェアハウス向けのInfobrightなど一部の製品を除いて、開発が止まってしまったか、当初の設計目標を実現できなかったものがほとんどとなっています。その後もプラガブル・ストレージエンジンAPIを利用して、コミュニティを中心に各種のストレージエンジンが新たに生まれています。

ちなみに、MySQL 5.7でデフォルトとなっているInnoDBも、現在のアーキテクチャのInnoDBはMySQL 5.1の途中からInnoDB Pluginという名称で登場したものです。旧来のInnoDBに加えて、新しいファイルフォーマットや圧縮機能の追加、性能やCPUスケーラビリティの改善が行われたものでした。

4.5 演習

このレッスンではMySQLの主なクライアントプログラムの機能と設定オプションについて学習しました。ここではmysqladminコマンドとmysqlコマンドを利用してサンプルデータを登録してみましょう。

1. mysqladminを利用してMySQLサーバー上に新しいデータベースworldを作成する

```
./mysqladmin -uroot create world
```

2. 以下のURLからサンプルデータベースのデータ「world database」をダウンロードし、展開する

●URL：Setting Up the world Database　/　Installation

https://dev.mysql.com/doc/world-setup/en/world-setup-installation.html

3. 展開したファイル内のデータをMySQLサーバーにロードする

サンプルデータのファイルの展開先が/home/mysql/mysql57の場合

```
./mysql -uroot world < /home/mysql/mysql57/world.sql
```

4. MySQLコマンドラインクライアントからworldデータベース内のCityテーブル内のレコードを10件だけ取得する

```
# worldデータベースに接続
```

```
$ ./mysql -uroot world
mysql> SELECT * FROM City LIMIT 10
```

解説

1. データベースの作成は、mysqladminコマンドのcreateサブコマンドだけではなく、mysqlコマンドラインクライアントでMySQLサーバーに接続した上でCREATE DATABASE文を発行することも可能です。

2. ここで利用しているサンプルデータのworldデータベースは、Oracle Universityによる公式研修のテキストでも利用されているものです。フィンランド政府の統計局の実際のデータを元にしていますが、最初に作成されたのが2005年よりも前になるため、Countryテーブルに東ティモールや南スーダンなどの新しい独立国は含まれていません。以前は各テーブルがMyISAMストレージエンジンを利用していましたが、MySQLサーバーのデフォルトがInnoDBストレージエンジンになってから、worldデータベースの各テーブルもInnoDBストレージエンジンに変更されています。

3. worldデータベースのファイル内容をテキストエディタで開いてみましょう。サンプルデータを格納するテーブルを作成するCREATE TABLE文や、データを追加するためのINSERT文が並んでいるのを確認できます。「<」を使うと、SQL文が含まれるファイルをでmysqlコマンドラインクライアントに読み込ませて実行することが可能です。

レッスン

5

GUI ツール MySQL Workbench

MySQL Workbench は、MySQL の運用や開発を支援するオープンソースの GUI ツールです。このレッスンでは MySQL Workbench について学習します。

5.1 MySQL Workbenchのインストール

MySQL Workbench は、もともとは、スキーマ設計のためのE/R図作成ツールとして開発され、個別のツールだったSQL開発機能や運用管理機能、マイグレーション機能が統合されました。

MySQL Workbenchは、MySQLサーバーとは独立した製品のため、個別にインストールする必要があります。MySQLサーバーと同じ筐体にインストールすることも可能ですが、リモート接続もできるため、通常はクライアントPCにインストールして使用します。サポートされているOSの詳細は以下のページで確認できますが、Windows、Mac OS X、Red Hat Linux/Cent OS/Oracle Linux、Fedora、Ubuntuにインストール可能です。

- **URL : Supported Platforms: MySQL Workbench**
  http://www.mysql.com/support/supportedplatforms/workbench.html

MySQL Workbenchは以下のページからダウンロードできます。

- **URL : Download MySQL Workbench**
  http://dev.mysql.com/downloads/workbench/

5.1.1 Windows版のインストール

Windows版のMySQL Workbenchのインストール方法には、インストーラを使用する方法と、Zipファイルを展開する方法があります。

インストーラを使用する場合は、ダウンロードページからMySQL Workbench用のインストーラをダウンロードしてインストールします。インストーラを使用してMySQL Serverをインストールする際に、MySQL Workbenchもあわせてインストールすることもできます。

Zipファイルを使用する場合は、ダウンロードページからダウンロードしたZipファイルを任意のディレクトリに展開するだけでインストールが完了します。

なお、Windows上にMySQL Workbenchをインストールする場合は、ダウンロードページに掲載されている以下のツールのインストールも必要であることに注意してください。

- Microsoft .NET Framework 4 Client Profile
- Visual C++ Redistributable for Visual Studio 2013

5.1.2 Mac OS X版のインストール

Mac OS X版は、ダウンロードページからダウンロードしたファイルをダブルクリックし、MySQL WorkbenchのアイコンをApplicationsアイコンにドラッグ＆ドロップすることでインストール可能です。

5.1.3 Linux版のインストール

Linux版のインストールは、ダウンロードページから.rpmファイルや.debファイルをダウンロードして、rpmやdpkgコマンドでインストールできます。また、以下のページでは、YumとAPTの公式リポジトリが公開されていますので、公式リポジトリをセットアップ後、yumやapt-getでインストールすることも可能です。

- **URL : Download MySQL Yum Repository**
  http://dev.mysql.com/downloads/repo/yum/
- **URL : Download MySQL APT Repository**
  http://dev.mysql.com/downloads/repo/apt/

5.2 MySQL Workbenchの主な機能

MySQL Workbenchの主な機能には、「MySQLサーバーの管理」「MySQLデータベースを使った開発支援」「E/R図を用いたスキーマ設計」「他DBからMySQLへのテーブル／データ移行支援」があります。それぞれについて、どのようなことができるか概要を解説します。なお、パフォーマンスチューニングに関連する機能についてはレッスン13、14で紹介するため、ここでは解説を割愛しています。

5.2.1 新規接続定義の作成

MySQL Workbench起動後、はじめにMySQLデータベースとの接続を定義する必要があります。そこでまず、MySQLデータベースとの接続定義を作成する方法について解説します。図1はMySQL Workbench起動後の画面です。

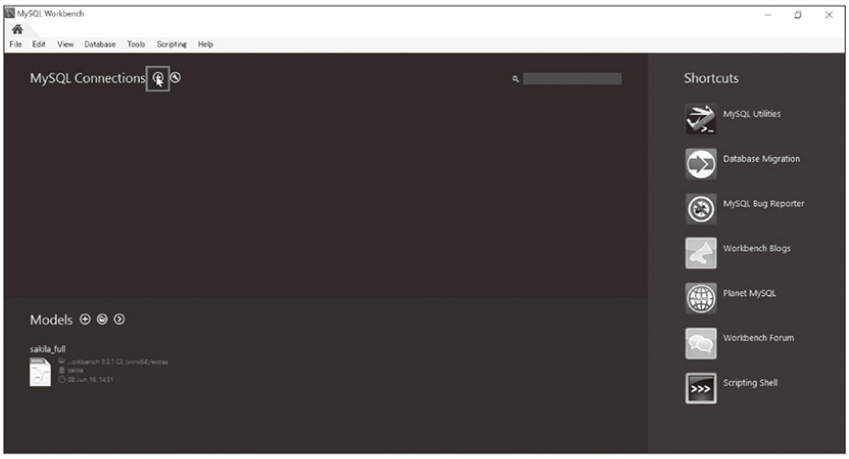


図 1 MySQL Workbench 起動後の画面

新しい接続定義を作成するために、左上の「MySQL Connections」の右側にある「+」ボタンをクリックします。「+」ボタンをクリックすると、図2の「Setup New Connection」が開くので、任意のコネクション名とMySQLデータベースへの接続情報を入力して「OK」をクリックすると接続定義が作成できます。図2はTCP/IPでの接続定義を作成する画面ですが、「Connection Method」を変更することで、SSHでの接続やソケットファイルを使ったローカル接続も選択可能です。

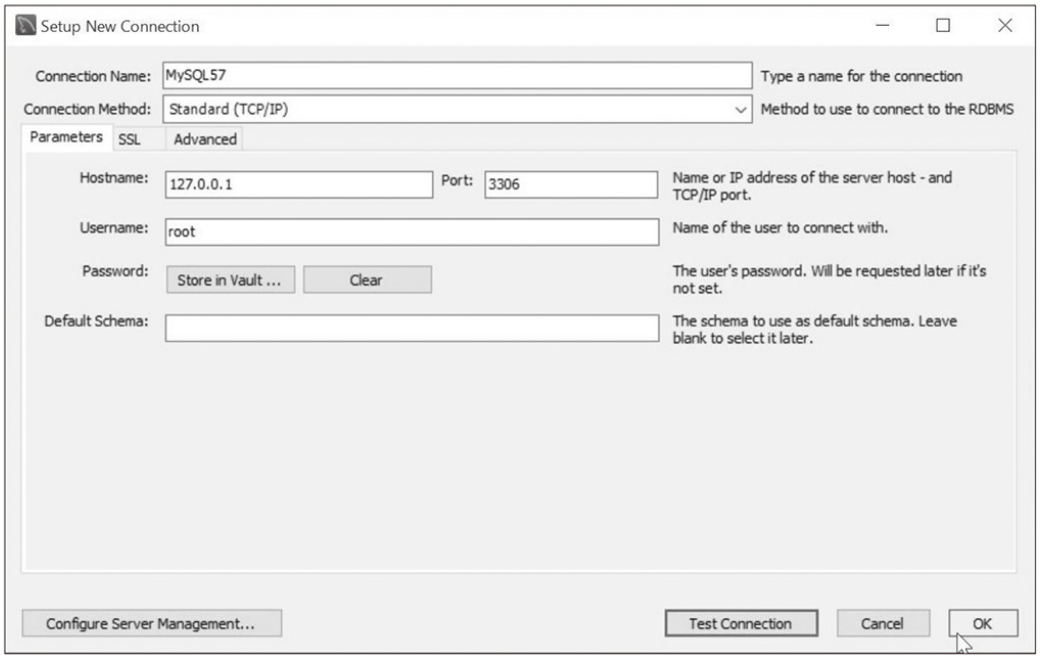


図 2　新しい接続定義の作成

　接続定義を作成すると、図3のようにアイコンが追加されるので、そのアイコンをダブルクリックします。すると図4のようにパスワード入力が促されるので、パスワードを入力して接続します。接続定義作成時にパスワードを保存しておいて、接続時にパスワード入力を省略することも可能です。

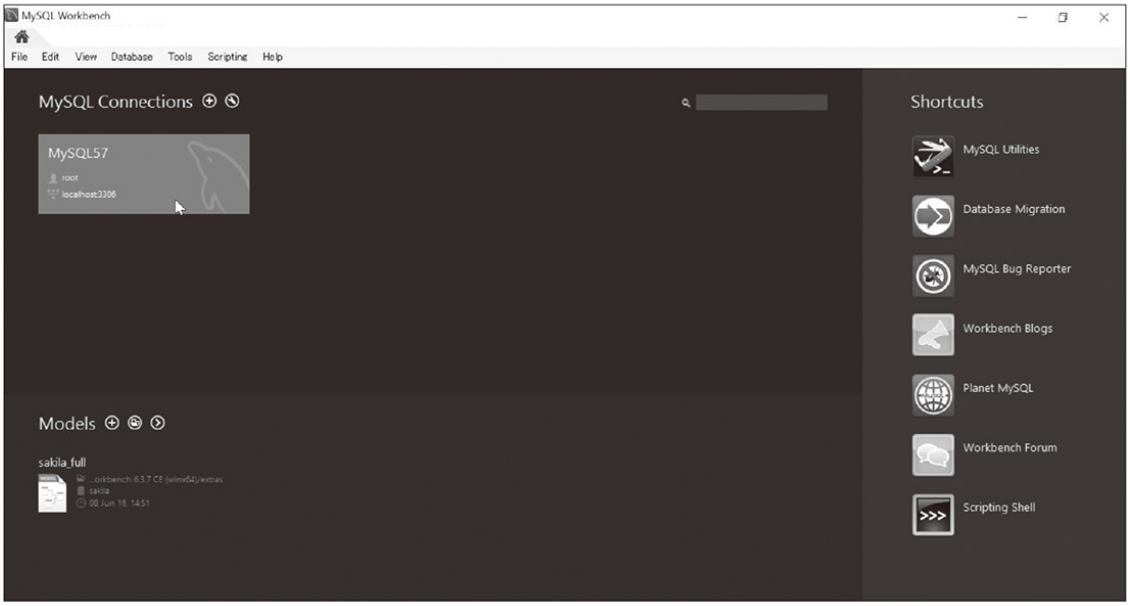


図 3　接続定義作成後の MySQL Workbench の画面

図 4　パスワード入力画面

接続後の画面は図5です。ここから、MySQLサーバーの管理系機能とMySQLデータベースを使った開発支援機能が使用できます。

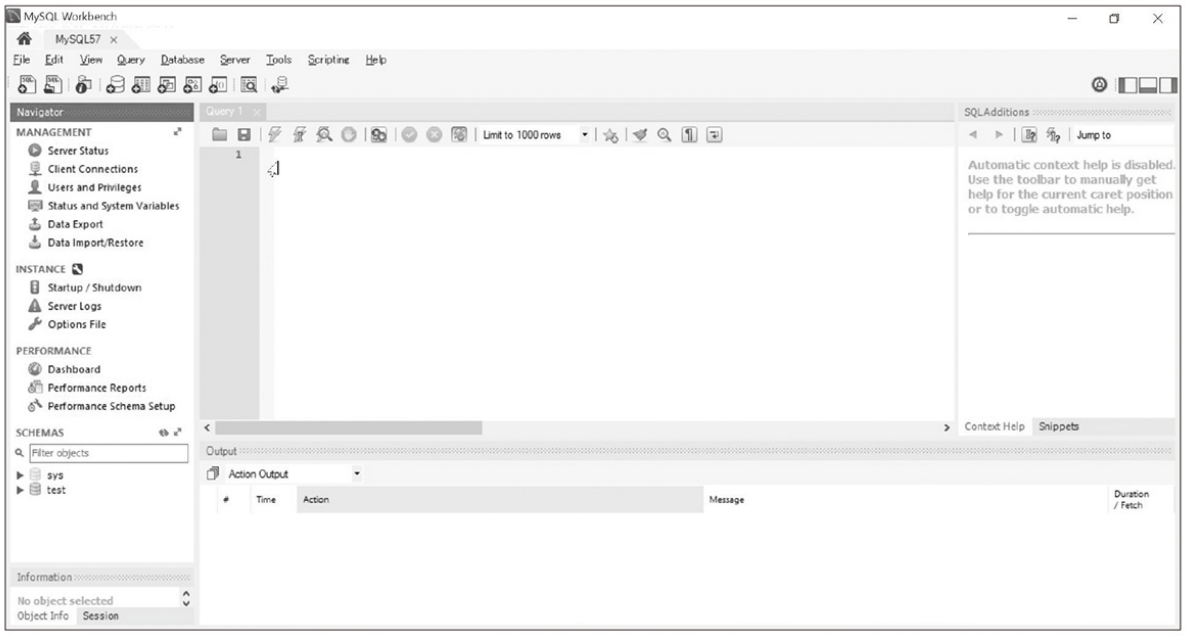


図 5　MySQL サーバー接続後の画面

図5の左側にナビゲーターがあり、MANAGEMENT、INSTANCE、PERFORMANCE、SCHEMASにカテゴリ分けされています。図6のようにナビゲーターをManagementタブ、Schemasタブに分けて表示することもできます。

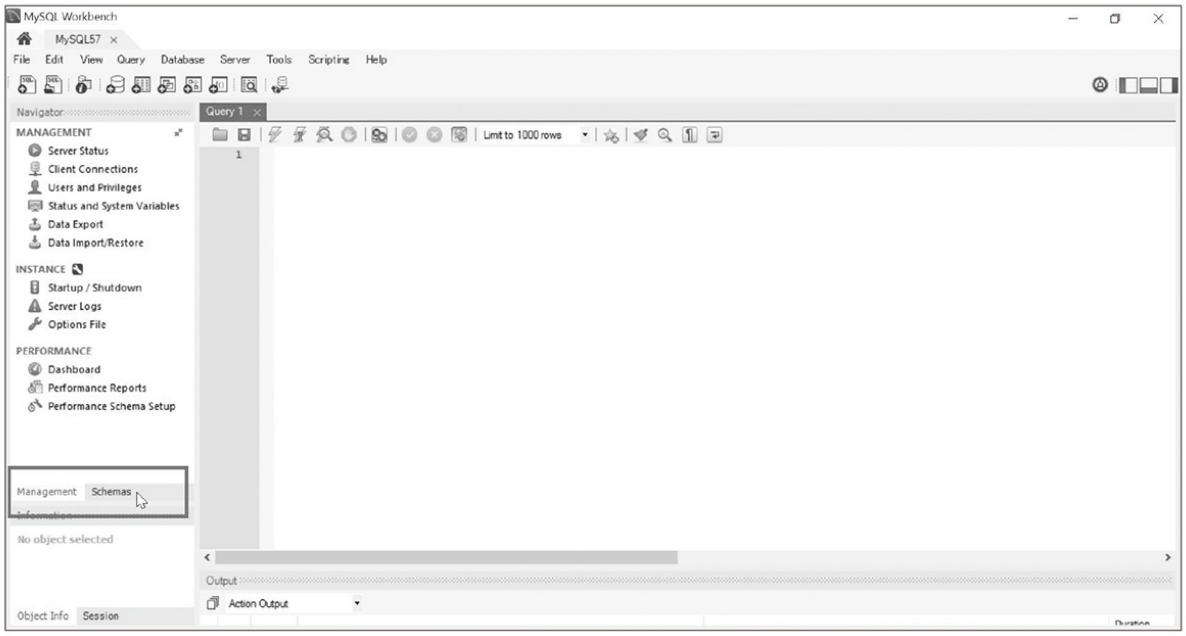


図 6　Management タブと Schemas タブ

5.2.2 MySQLサーバーの管理系機能

MySQLサーバーの管理系機能として、ここではMANAGEMENTとINSTANCEについて解説します。なお、PERFORMANCEおよびMANAGEMENTのパフォーマンスチューニング系機能については、レッスン14で解説します。

MANAGEMENTの「Server Status」をクリックすると、図7の右のようにサーバーの稼働状況をグラフで確認したり、使用可能な機能、ディレクトリ配置などを確認したりできます。

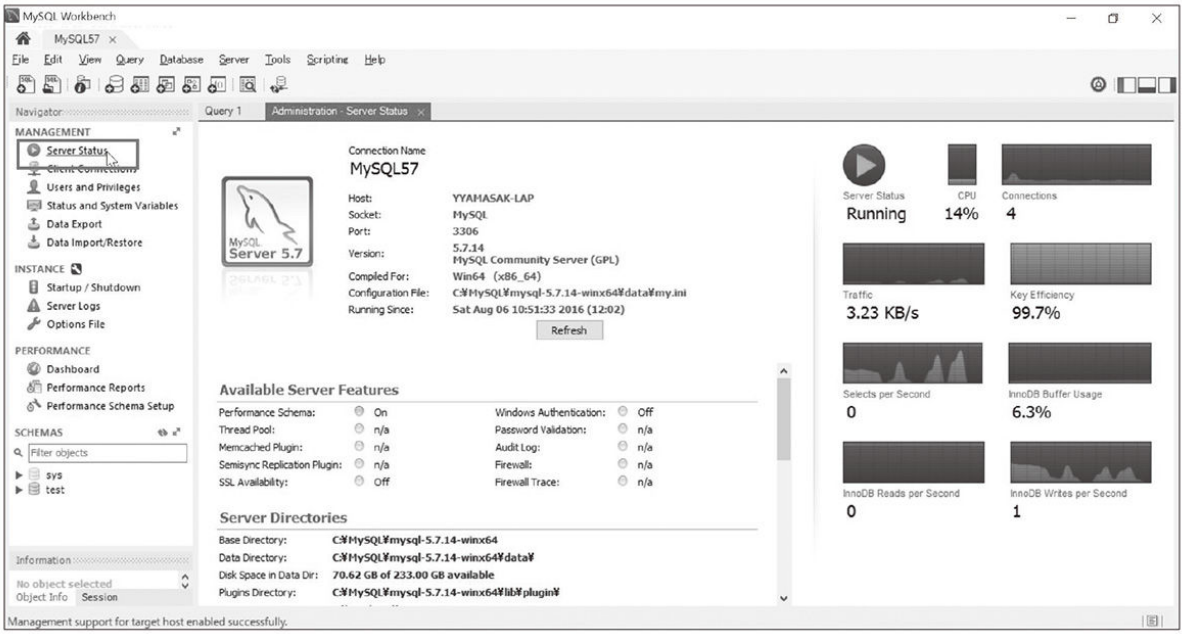


図7 Server Status

「Users and Privileges」からはユーザー管理ができます（図8）。新しいユーザーを追加したり、権限を付与したり剥奪したりすることができます。

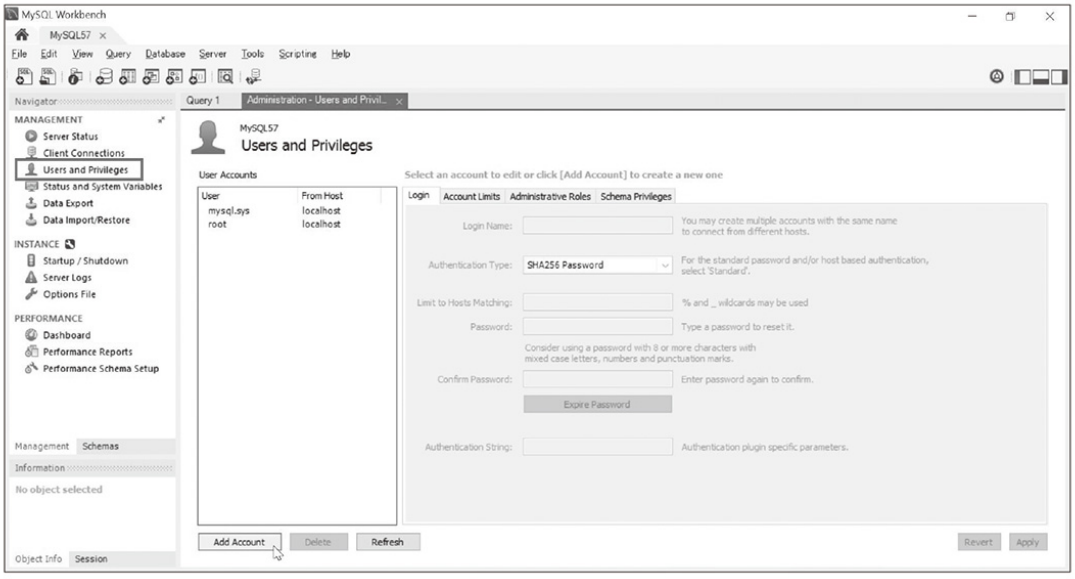


図 8 Users and Privileges

「Data Export」では、MySQLデータベースからデータを抜き出すことができます。図9のように、どのスキーマからどのテーブルのデータを抜き出すか選択します。内部的には、mysqldumpが実行されてデータを抜き出します。右上の「Advanced Options…」をクリックすることで、mysqldump実行時のオプションを詳細に制御することも可能です。また、抜き出したデータは「Data Import/Restore」でインポートすることもできます。

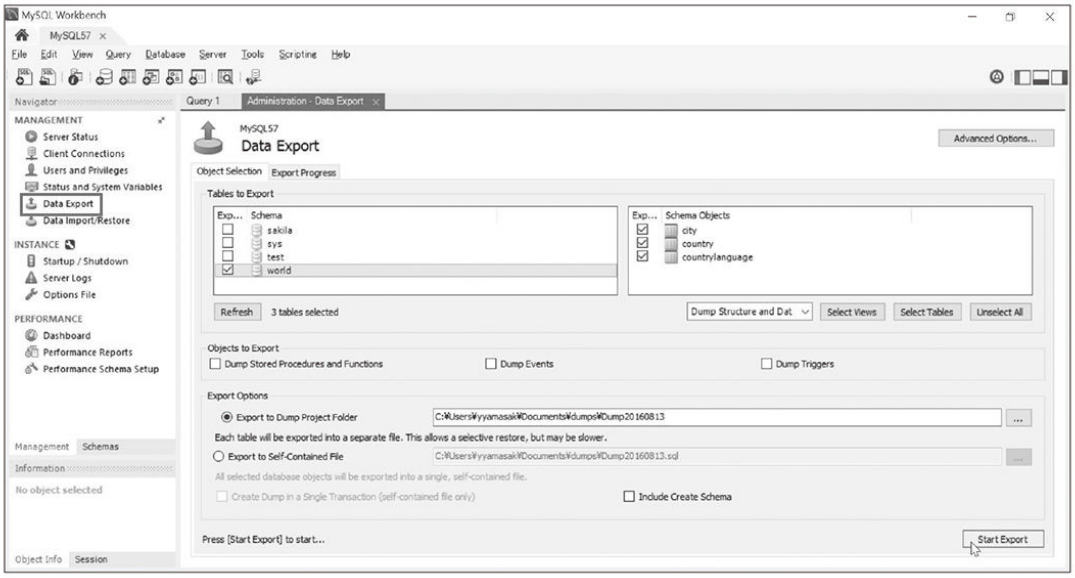


図 9 Data Export

INSTANCEの「Startup/Shutdown」では、MySQLサーバーの起動、停止が行えます（図10）。この機能を利用するためには、OSのサービスとしてMySQLサーバーが登録されている必要があります。

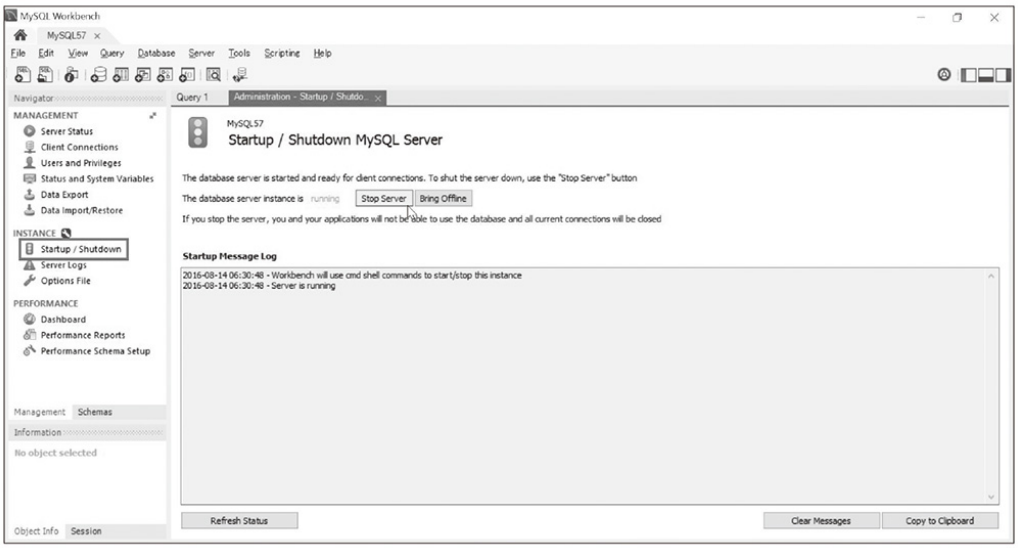


図 10 Startup/Shutdown

「Server Logs」では、エラーログを確認できます（図11）。「Options File」では、設定ファイルの確認や編集が可能です（図12）。

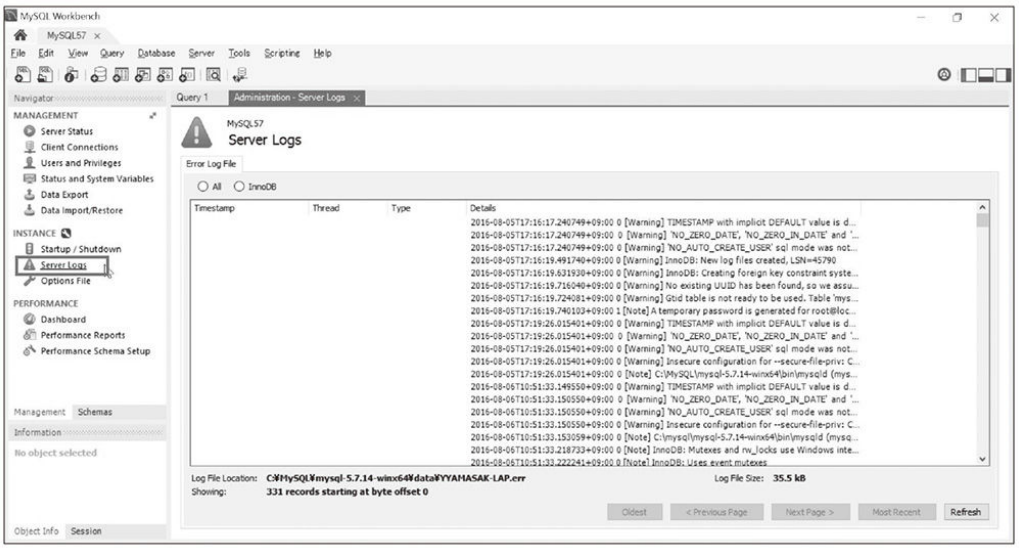


図11 Server Logs

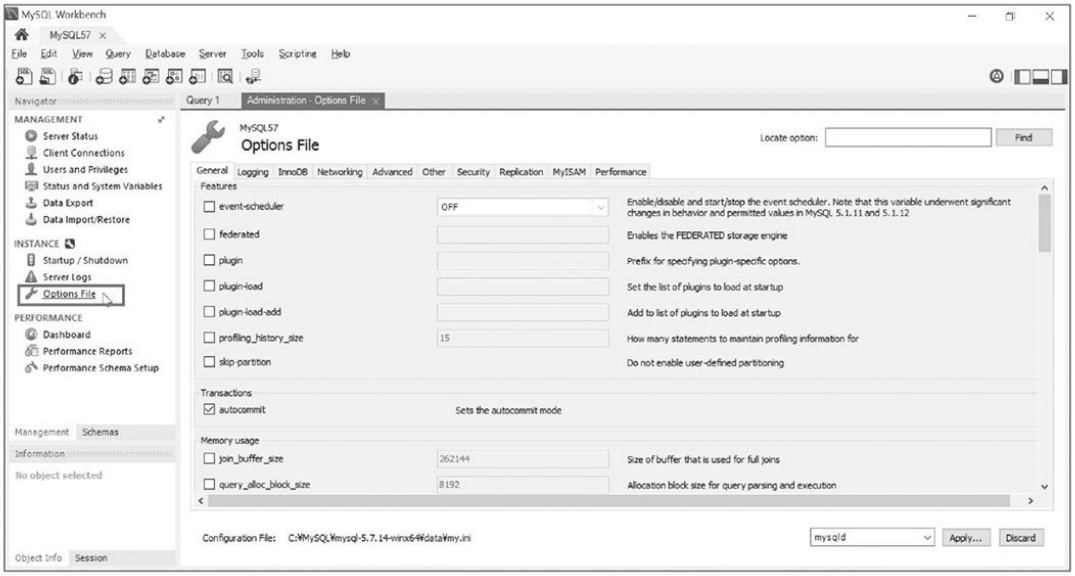


図12 Options File

5.2.3 MySQLサーバーのパフォーマンスチューニング系機能

パフォーマンスチューニング系機能については、レッスン14 で解説します。

5.2.4 MySQLデータベースを使った開発支援機能

左側のナビゲーターのSCHEMASでは、図13のようにエクスプローラーライクなツリー構造のObject Browserから、MySQLサーバー内のオブジェクトを確認できます。

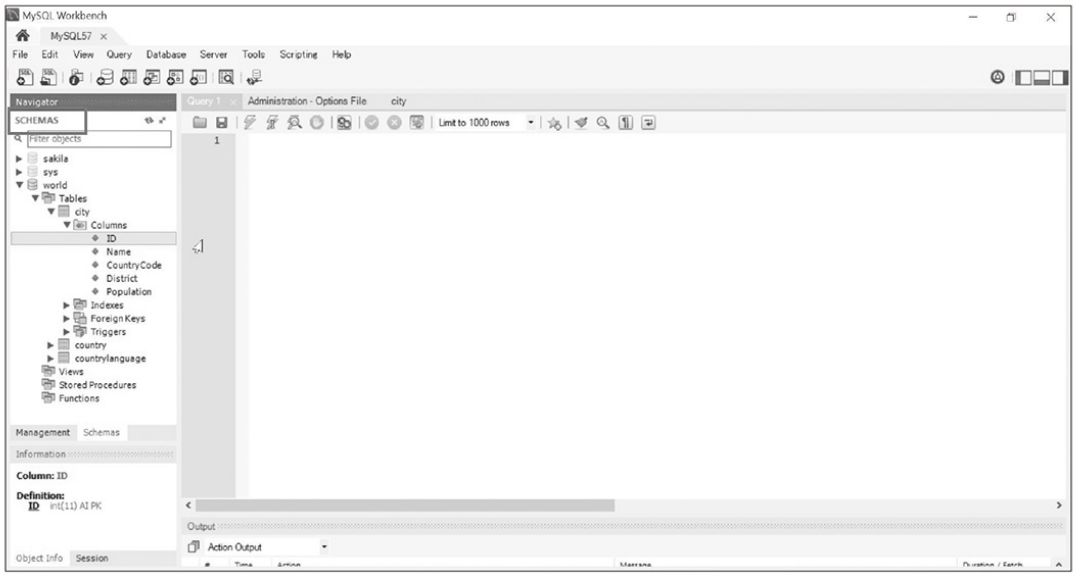


図 13 Object Browser

スキーマ内に存在するオブジェクトを一覧で確認することもできます。一覧で確認したい場合は、スキーマ名を右クリックし、「Schema Inspector」を選択します。例えば、図14はworldデータベース（スキーマ）内に存在するオブジェクトを一覧で確認している例です。図14ではColumnsタブを選択していますが、ほかにもTables、Indexes、Triggersなどのタブがあるため、スキーマ内のオブジェクト一覧を素早く確認できます。

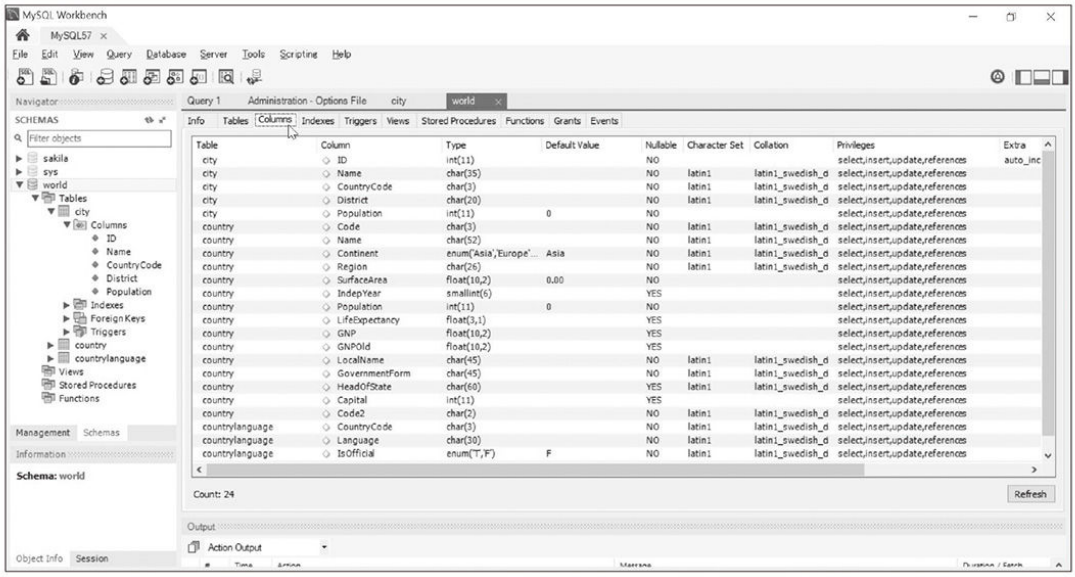


図 14 Schema Inspector

テーブル内に格納されているデータを確認したい場合は、図15のようにテーブル名を右クリックして「Select Rows」を選択することでSELECT文を自動的に作成し、実行できます。SQLの実行結果は、図16のように画面下部のスプレッドシートに表示されます。データをSELECTする時に、デフォルトでは1000行のデータが取得されるようにLIMITが設定されていますが、LIMITの設定をはずしたり、取得する行数を変更したりすることも可能です。

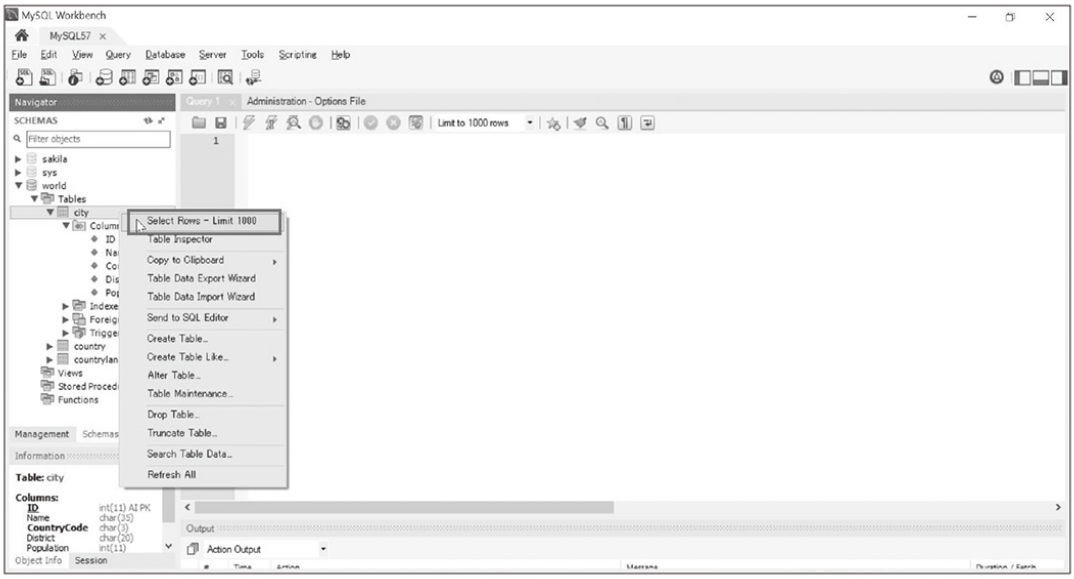


図 15　Select Rows 実行画面

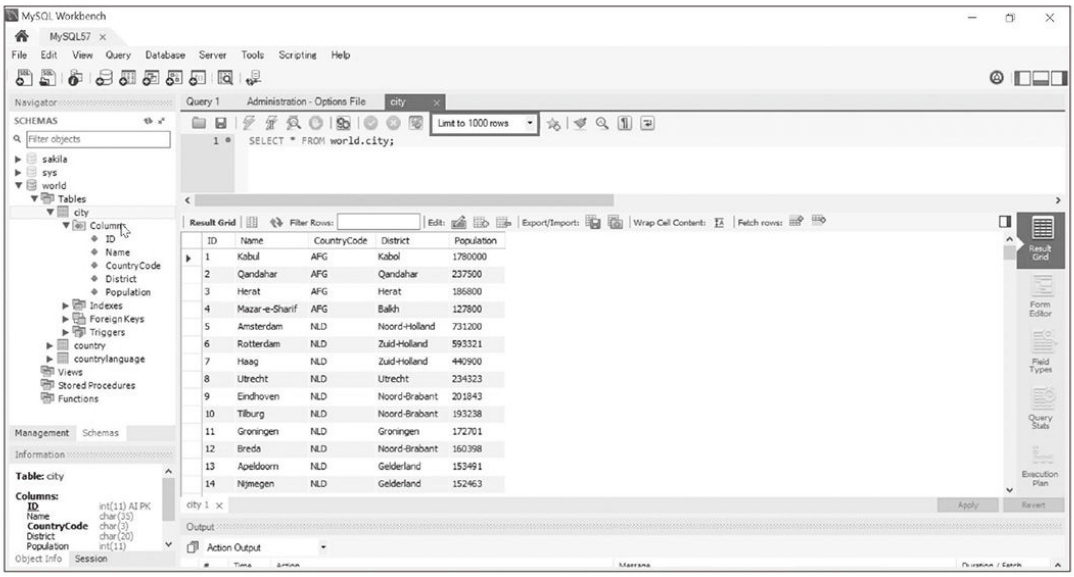


図 16　SQL 実行後の画面

SQLエディタに直接SQL文を記入して、任意のSQLを実行することもできます。SQLエディタには、SQL文、関数、テーブル名、列名などオブジェクト名の入力中に候補が表示される補完機能があります。また、記述したSQL文のキーワードがハイライトされるだけでなく、

SQLの整形機能も使えるため、SQLの視認性が向上します。ほかにも、SQLの構文を確認するためのContext Helpや、よく使うSQL文を簡単に呼び出せる（コピー＆ペーストできる）Snippetsが利用できるなど、SQL作成を効率化する機能が多数あります。

SQLの整形機能は、図17中央上部の枠で囲んだホウキのようなアイコンをクリックすることで利用できます。また、右上にある枠で囲んだアイコンをクリックするとContext HelpやSnippetsを表示できます。下部にあるタブで、Context HelpとSnippetsを切り替えられます。

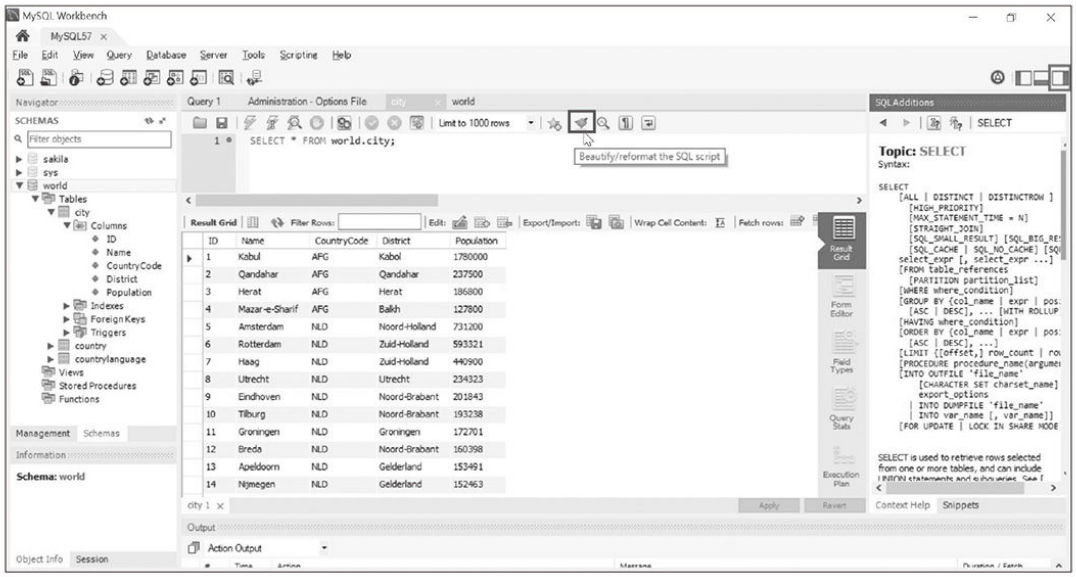


図17 SQLエディタ

このほか、スプレッドシートに表示された実行結果を外部のファイルにエクスポートしたり、集計関数やJOINを利用していない場合は結果を直接編集してデータベースに反映したりすることも可能です。

5.2.5 E/R図を用いたスキーマ設計

MySQL Workbenchには、E/R図を用いたスキーマ設計機能もあります。この機能を使用する時は、MySQL Workbench起動後の画面からModelsのメニューを使用します。新規にE/R図を作成する時は、図18の「+」ボタンをクリック後、図19の「Add Diagram」をダブルクリックしてEER Diagramの画面（図20）を開くことでE/R図を作成できます。

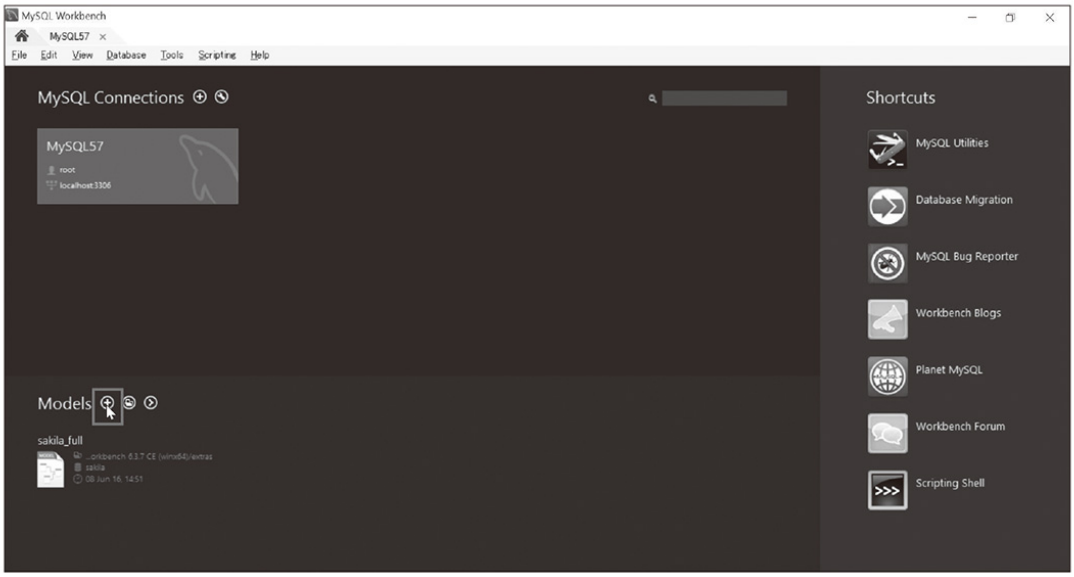


図 18 E/R 図作成機能を使用するためのアイコン

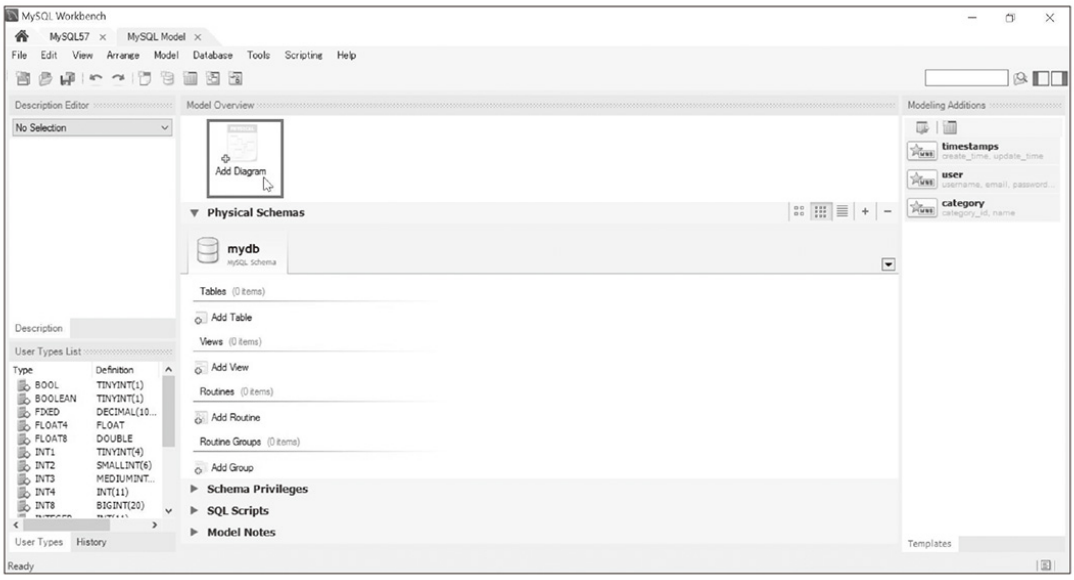


図 19　Model の管理画面

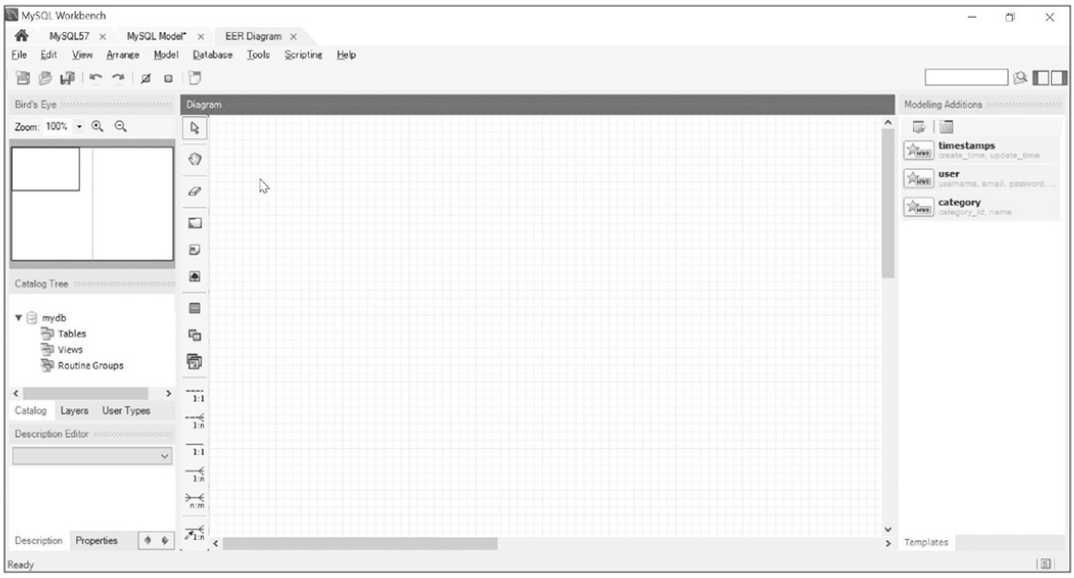


図 20　E/R 図を作成できる画面

E/R図作成後は「フォワード・エンジニアリング・ウィザード」を使用して、作成したテーブルなどをMySQLサーバーへ反映できます。フォワード・エンジニアリング・ウィザードは、図20の「Database」メニューにある「Forward Engineer...」メニューを選択することで利用

できます。フォワード・エンジニアリング・ウィザードでは、直接MySQLデータベース上にオブジェクトを作成するだけでなく、オブジェクトを作成するためのSQLスクリプトを生成することも可能です。

　また、既にMySQLデータベース上にテーブル等が作成されている場合、図20の「Database」メニューにある「Reverse Engineer…」からリバース・エンジニアリングすることで、既存のオブジェクト定義をE/R図化することもできます。

5.2.6 他DBからMySQLへのテーブル／データ移行支援

MySQL Workbenchの「マイグレーション・ウィザード」は、Microsoft SQL ServerやPostgreSQLなどのデータベース製品からMySQLサーバーへの移行を支援するツールです。また別環境のMySQLサーバーからの移行や、ODBCに対応した各種データベース製品からの移行も可能となっています。マイグレーション・ウィザードで移行できるのは、テーブル定義とデータ、ビューです。ストアドプロシージャなどのサーバーサイドプログラムはMySQLサーバー間での移行の場合のみ対応しています。

マイグレーション・ウィザードは、図18の右側にある「Database Migration」から実行できます。

5.3 MySQL Workbenchの商用版限定機能

ここまで紹介してきた機能は、すべてMySQL Workbenchのコミュニティ版でも利用できますが、MySQL Workbenchの商用版では、商用版限定の機能も使えます。

5.3.1 データモデルのドキュメント出力機能

テーブル定義書のようなドキュメントを自動作成できる機能です。E/R図を作成してDB設計した内容を、HTMLもしくはテキストのフォーマットで、ドキュメントに出力できます。テンプレートやフォーマットをカスタマイズすることも可能です。リバース・エンジニアリング機能と組み合わせれば、最新のテーブル定義書を短時間で自動作成できます。

ドキュメント出力機能を使用する時は、図21のようにE/R図作成画面から「Model」の「DBDoc – Model Reporting…」を選択します。すると図22の画面が表示されるので、フォーマットや出力先（Output Path）などを選択して、「Generate」をクリックするとドキュメントを出力できます。

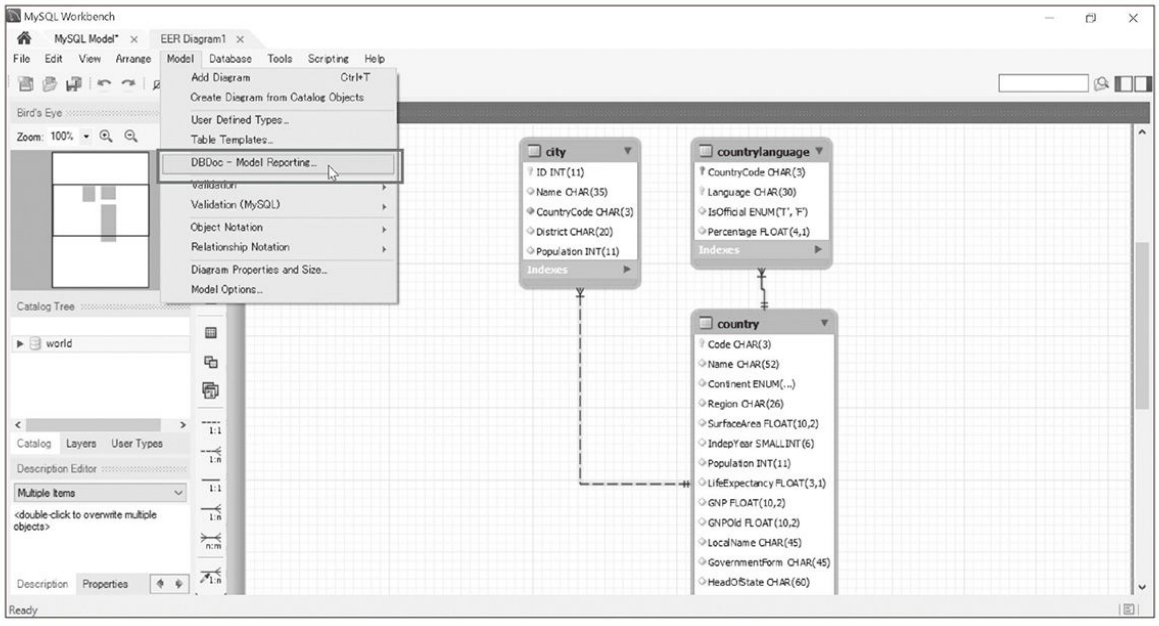


図 21 ドキュメント出力機能 1

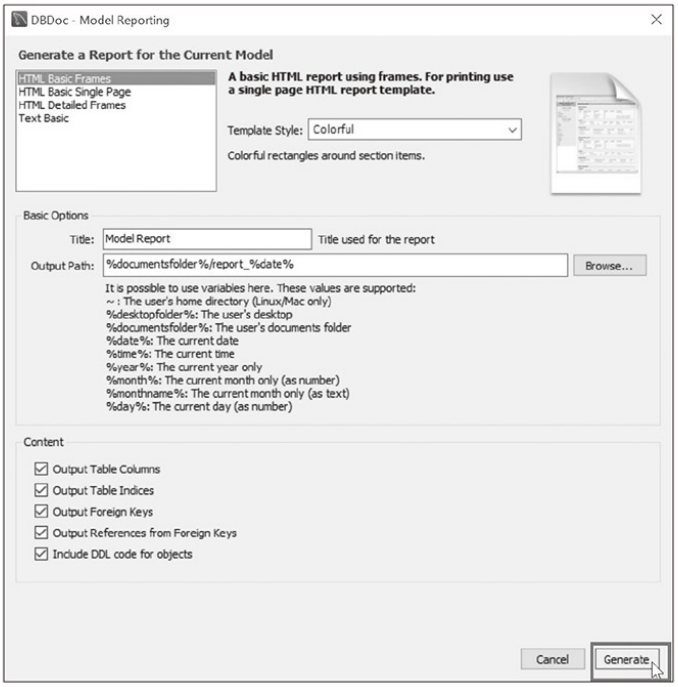


図 22　ドキュメント出力機能 2

ドキュメントのテンプレートは、HTMLが3種類（Basic Frames、Basic Single Page、Detailed Frames）とテキストフォーマットが用意されています。図23は、HTMLのBasic Framesでの出力例です。左側のフレームでオブジェクトを選択すると、右側のフレームでオブジェクトの定義を確認できます。

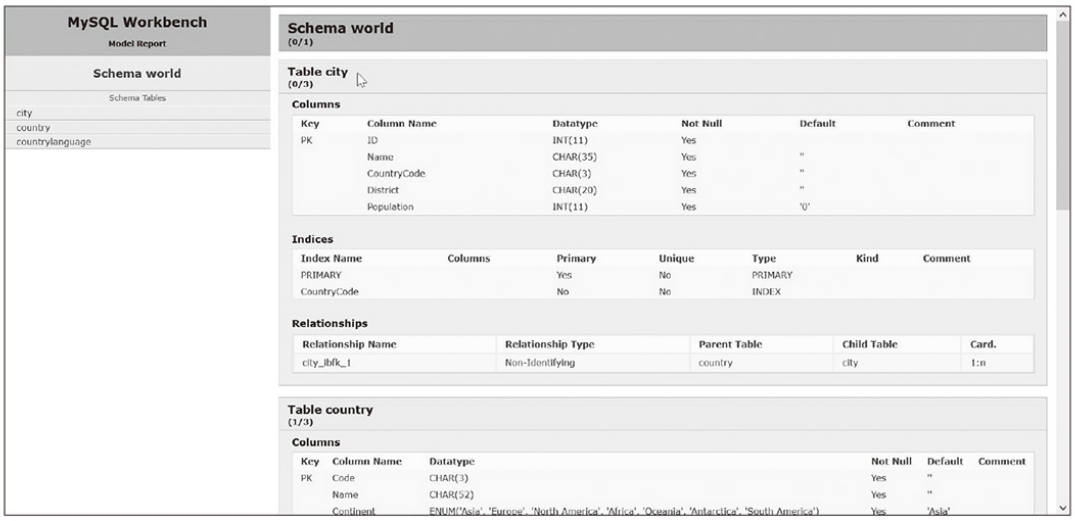


図 23　ドキュメント出力結果 (HTML：Basic Frames)

図24は、HTMLのDetailed Framesでの出力例です。Detailed Framesでは、オブジェクトを作成するためのDDL文も出力結果に含まれます。

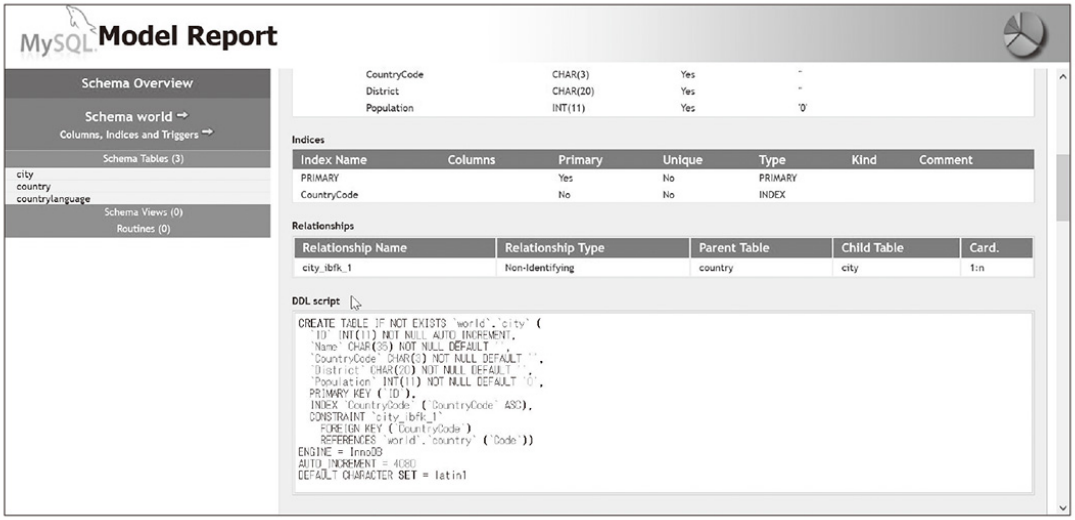


図 24　ドキュメント出力結果 (HTML：Detailed Frames)

リスト1は、テキストフォーマットの出力例です。サンプルデータベースのworldデータベースをドキュメント出力すると、このようになります。

リスト1　ドキュメント出力結果（テキストフォーマット）

```
+-----------------------------------------+
|    Model Report                         |
+-----------------------------------------+

Total number of Schemata: 1
=========================================================
===

. Schema: world
-------------------------------------------
## Tables (3) ##
. Table: city
## Columns ##
      Key       Column  Name       Datatype       Not  Null       Default
Comment
     PK       ID                INT(11)       Yes
              Name              CHAR(35)      Yes           ''
              CountryCode       CHAR(3)       Yes           ''
              District          CHAR(20)      Yes           ''
              Population        INT(11)       Yes           '0'

## Indices ##
       Index  Name        Columns        Primary        Unique        Type
Kind      Comment
     PRIMARY                          Yes          No          PRIMARY
     CountryCode                      No           No          INDEX
```

```
## Relationships ##
        Relationship Name         Relationship Type         Parent Table
Child Table      Cardinality
      city_ibfk_1               Non-Identifying         country              city
1:n

-----------------------------------------

. Table: country
## Columns ##
       Key        Column Name        Datatype        Not Null        Default
Comment
      PK       Code               CHAR(3)       Yes             ''
               Name               CHAR(52)      Yes             ''
                 Continent          ENUM('Asia', 'Europe', 'North America',
'Africa', 'Oceania', 'Antarctica', 'South America')      Yes      'Asia'
               Region             CHAR(26)      Yes             ''
               SurfaceArea        FLOAT(10,2) Yes             '0.00'
               IndepYear          SMALLINT(6) No               NULL
               Population         INT(11)       Yes             '0'
               LifeExpectancy FLOAT(3,1)    No               NULL
               GNP                FLOAT(10,2) No               NULL
               GNPOld             FLOAT(10,2) No               NULL
               LocalName          CHAR(45)      Yes             ''
               GovernmentForm CHAR(45)      Yes             ''
               HeadOfState        CHAR(60)      No               NULL
               Capital            INT(11)       No              NULL
```

```
            Code2           CHAR(2)     Yes         ''

## Indices ##
    Index Name      Columns     Primary     Unique      Type      Kind
Comment
    PRIMARY                     Yes         No          PRIMARY

## Relationships ##
    Relationship Name       Relationship Type   Parent Table    Child
Table       Cardinality
    city_ibfk_1             Non-Identifying     country         city
1:n
        countryLanguage_ibfk_1      Non-Identifying         country
countrylanguage   1:n

-----------------------------------------

. Table: countrylanguage
## Columns ##
    Key     Column Name     Datatype            Not Null    Default
Comment
    FK      CountryCode     CHAR(3)             Yes         ''
    PK      Language        CHAR(30)            Yes         ''
            IsOfficial      ENUM('T', 'F')  Yes         'F'
            Percentage      FLOAT(4,1)      Yes         '0.0'

## Indices ##
```

```
      Index Name        Columns        Primary        Unique        Type
Kind      Comment
     PRIMARY                    Yes          No          PRIMARY
     CountryCode                No           No          INDEX

## Relationships ##
      Relationship Name          Relationship Type       Parent Table
Child Table          Cardinality
       countryLanguage_ibfk_1     Identifying                country
countrylanguage      1:n

---------------------------------------

=========================================
===
End of MySQL Workbench Report
```

5.3.2 データモデルの検証機能

E/R図を作成してDB設計した内容を検証し、DB設計上の懸念事項があればアドバイスを提示できる機能です。データモデルの検証を行う場合は、図25のようにE/R図作成画面から「Model」の「Validation」、または「Validation（MySQL）」から「Validate All」を選択します。一般的なRDBMSとしての検証を行う時は「Validation」を選択し、MySQL固有の観点で検証を行う場合は「Validation（MySQL）」を選択します。

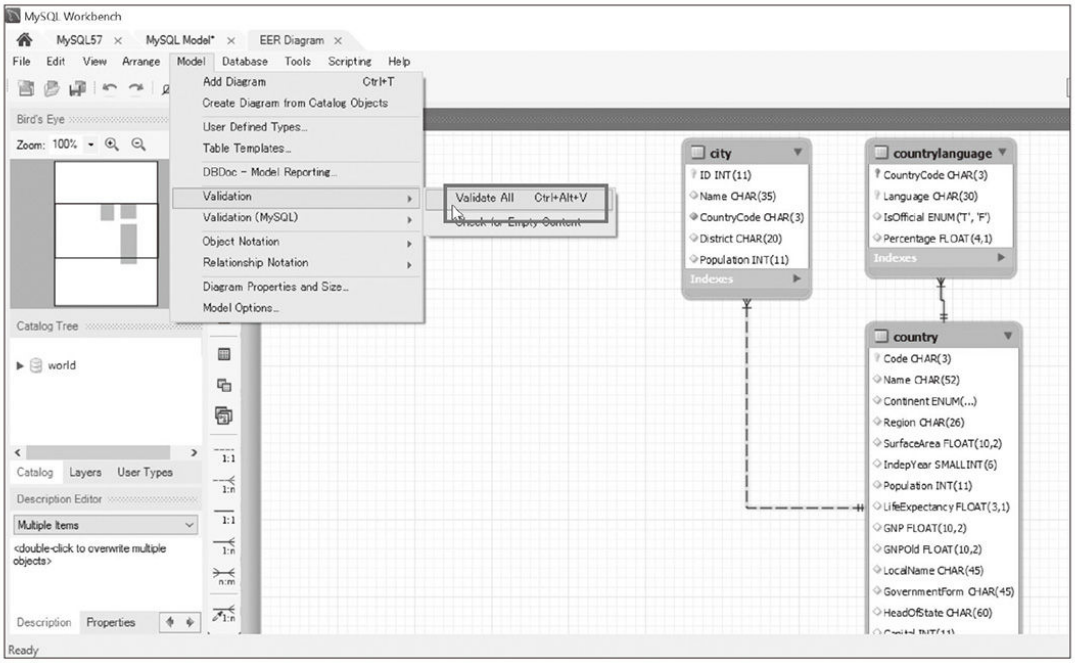


図25 データモデルの検証機能

図26は、サンプルデータベースのworldデータベースを「Validation」で検証した結果です。例えば、カーソルで選択している行では、「Nameという名前の列がcountryテーブルとcityテーブルに存在するが、データ型の定義が異なる」ことを知らせてくれています。同じ名

前の列は同じ情報を保持する可能性があるため、念のためデータ型の定義が異なることを検知して知らせてくれているのです。

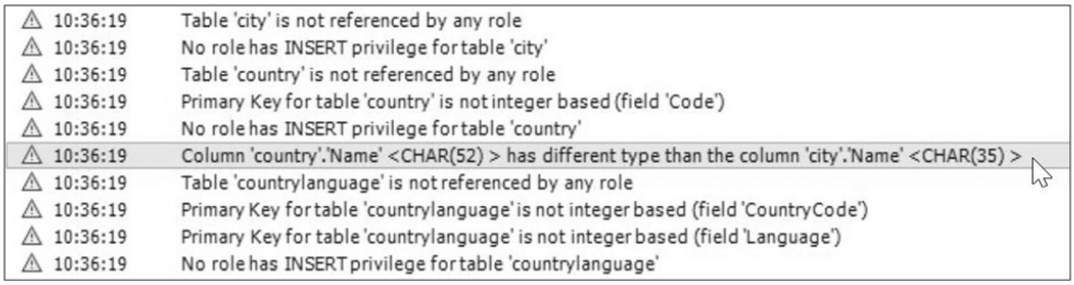


図 26　データモデルの検証結果

5.3.3 MySQL Enterprise Edition限定機能に対するGUI

MySQL Enterprise Editionで使用できる以下の機能に対するGUIが付属しています。

- MySQL Enterprise Backup
- MySQL Enterprise Audit
- MySQL Enterprise Firewall

MySQL Enterprise Backup

MySQL Enterprise Backupを使ったバックアップジョブの作成や、スケジューリング、バックアップジョブの実行状況確認、バックアップデータのリストアなどが行えます。

図27の画面で、MySQL Enterprise Backupの実行ファイルのパス、バックアップ出力先、バックアップの実行ユーザーを事前に設定しておきます。そして、図28の画面で「New Job」ボタンをクリックするとバックアップジョブの作成を開始します。　Scheduleなど詳細を設定したい時は、図28の右上にある「Settings…」ボタンをクリックします。次に表示される図29の画面で詳細を設定することができます。

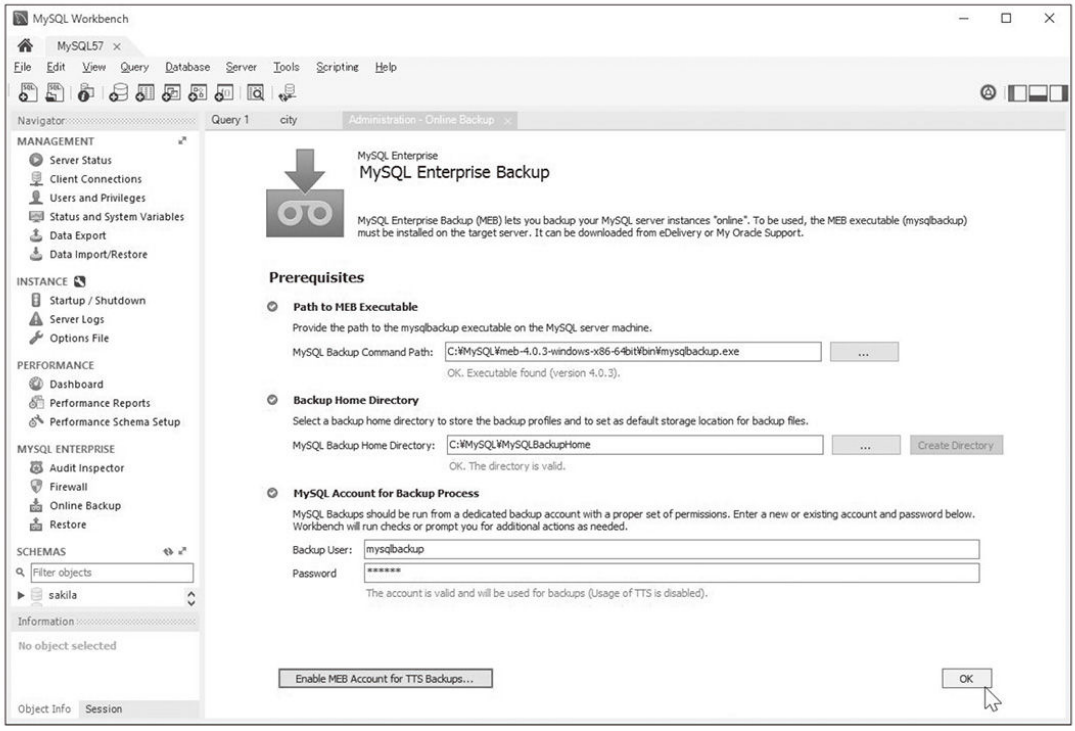


図 27　MySQL Enterprise Backup 用の GUI（事前設定画面）

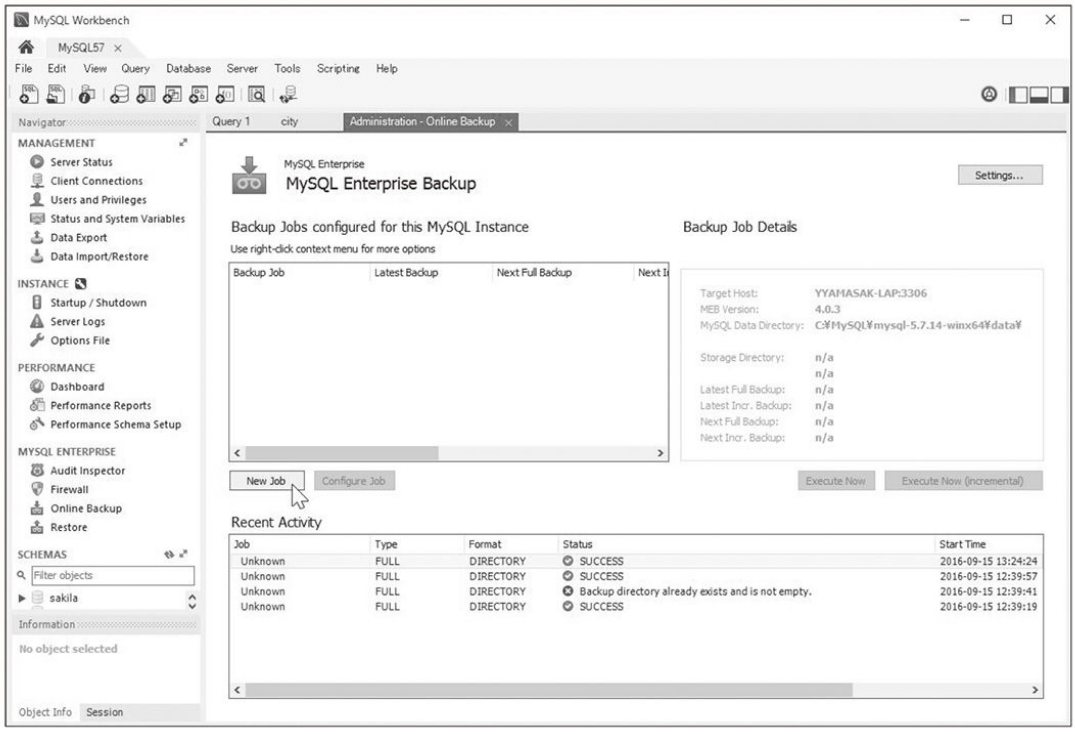


図 28　MySQL Enterprise Backup 用の GUI（メインの画面）

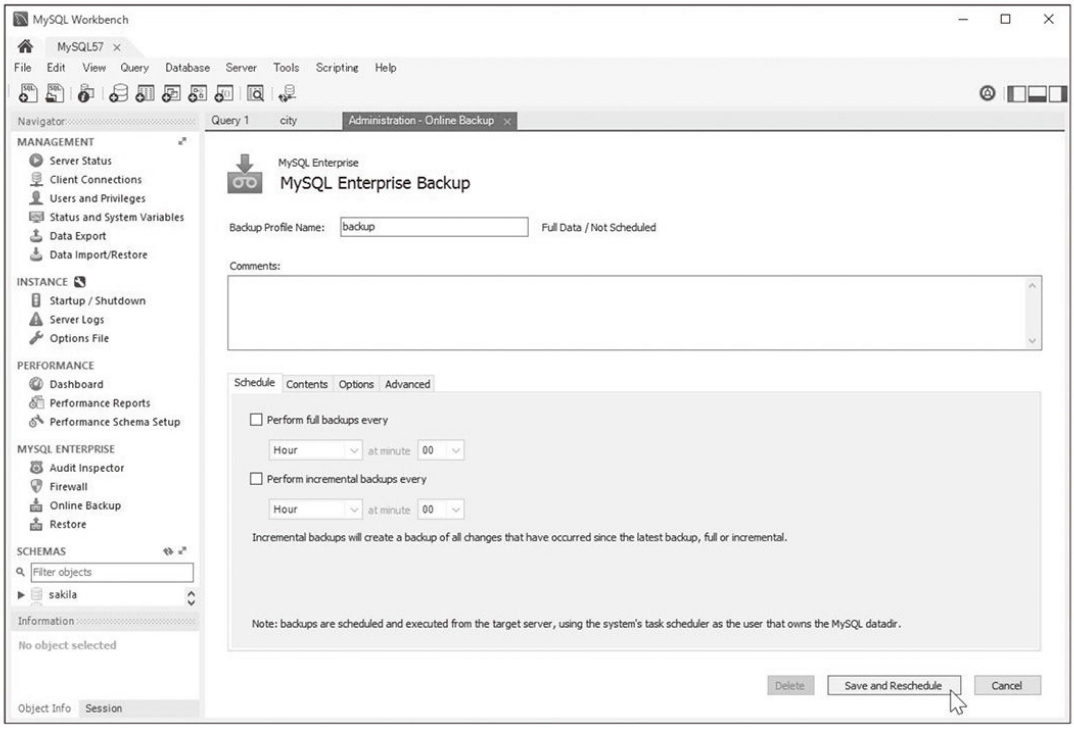


図 29　MySQL Enterprise Backup 用の GUI（バックアップジョブの設定画面）

MySQL Enterprise Audit

　MySQL Enterprise Auditで取得した監査ログを確認できる簡易のGUIです。イベントの種類や監査ログに出力されている内容で絞り込んで表示することも可能です。図30は監査ログを「SELECT」というキーワードで絞り込んで表示している例です。

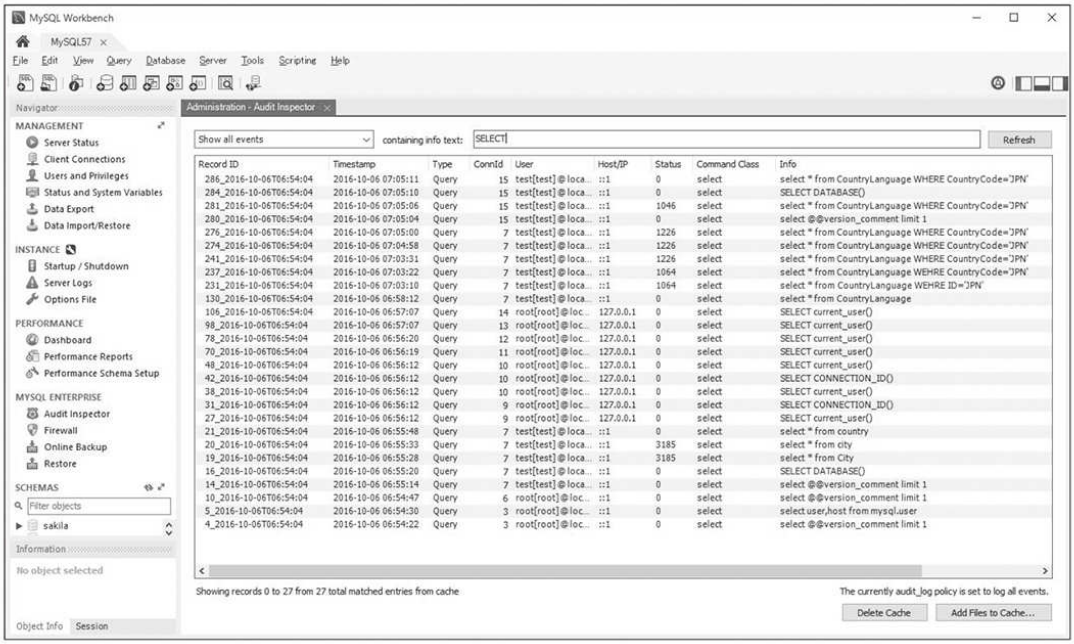


図 30　MySQL Enterprise Audit 用の GUI（監査ログの確認画面）

なお、監査ログをより詳細に分析したい場合は、Oracle Audit Vault and Database Firewall（AVDF）との連携も可能です。AVDFを使えば、監査ログを分析するだけでなく、監査ログを監視して怪しいアクセスがあれば警告を通知したり、DBAによる不正を防ぐために監査ログをDBサーバーとは異なる別のサーバーで保管したりすることも可能です。

MySQL Enterprise Firewall

MySQL Enterprise Firewallの設定を管理できます。ユーザーごとに設定されているホワイトリスト（警戒する必要のない安全なSQL文）の内容の確認や編集も可能です。図31は、MySQL Enterprise Firewallの設定と、関連するステータス変数を一覧で確認できる画面です。

図32は、ホワイトリストの内容確認と編集画面です。このようにユーザー管理の画面に「Firewall Rules」タブが追加され、そこからホワ

イトリストの内容確認＆編集が可能です。ホワイトリストはユーザーごとに保持されています。

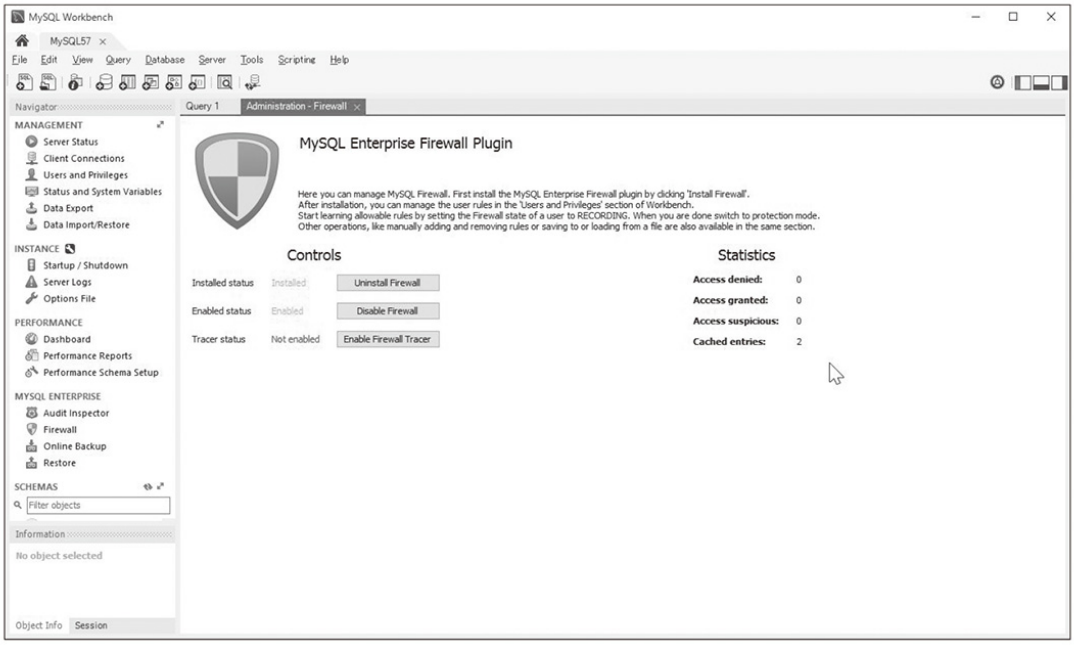


図 31　MySQL Enterprise Firewall 用の GUI（設定＆ステータス変数の確認画面）

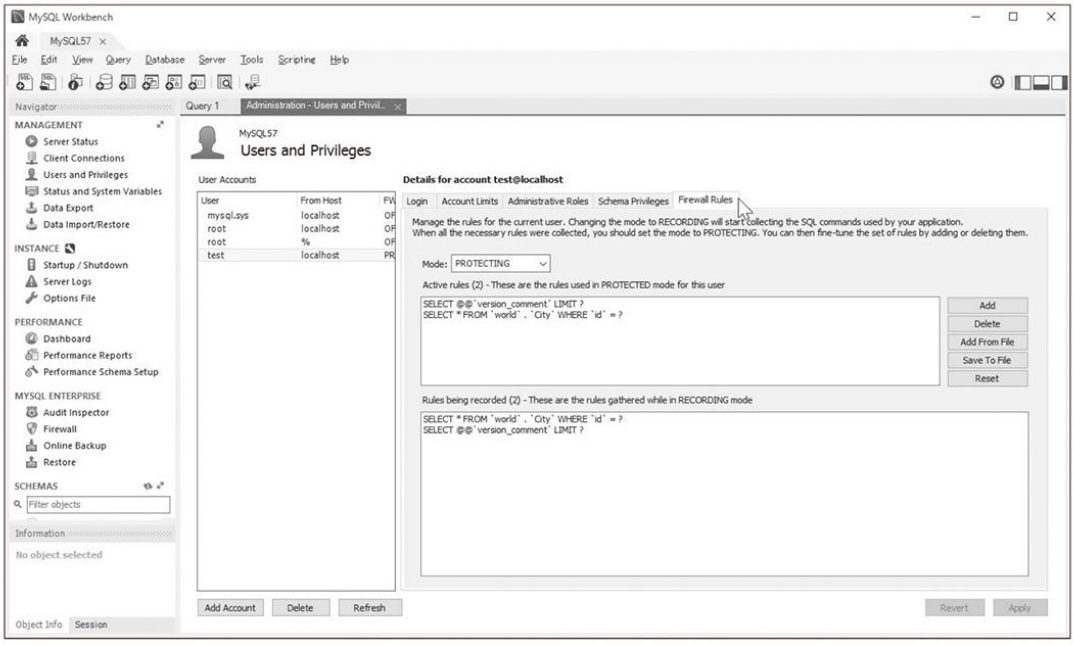


図 32　MySQL Enterprise Firewall 用の GUI（ホワイトリストの確認画面）

Column　MySQL Utilitiesとは

MySQL Utilitiesは、MySQLの運用管理を支援するPythonで書かれたスクリプト集です。複雑な操作を1コマンドで実行できることを目的としています。これらのスクリプトで使われるPythonのライブラリも同梱されており、独自の運用管理ツールの作成をより簡単にします。

MySQL Utilitiesには大きく以下のカテゴリのツールが含まれています。

- データ管理
- サーバー運用支援
- レプリケーション管理
- サーバー設定管理

データ管理ツールには、以下のものがあります。

ツール	機能概要
mysqldbcompare	同一サーバー内または 2 つのサーバー間のデータや定義の比較
mysqldbcopy	別のサーバーにデータベースをコピー
mysqldbexport	データとメタデータをエクスポート
mysqldbimport	データとメタデータをインポート
mysqldiff	2 つのサーバー間のテーブルなどオブジェクトの定義を比較

レプリケーション管理では、GTID（レッスン9、10で解説）をフル活用しており、GITDが有効になっている環境に新規にレプリケーションを構築するmysqlreplicate、レプリケーション構成を図示するmysqlrplshow、サーバー間のGTIDを比較して遅延の有無を確認するmysqlrplsyncなどが利用できます。さらに、mysqlfailoverを使うと

レプリケーション環境でのフェイルオーバー設定が可能となります。

MySQL Utilitiesは、スクリプトごとにオプションが異なりますが、1つポイントになるのが、接続する対象サーバーやユーザーの指定方法です。通常のMySQLクライアントプログラム（mysqlやmysqldumpなど）とは書式が異なっています。

```
# 接続先サーバーやユーザー、パスワードの指定方法の違い
# MySQLクライアントプログラムのオプション表記
mysqldump -uユーザー -pパスワード -hホスト名 -Pポート番号...
mysqldump --user=ユーザー --password=パスワード --host=ホスト名 -port=ポート番号 ...

# MySQL Utilitiesのツールでのオプション表記
mysqldbexport --server=ユーザー:パスワード@ホスト名:ポート番号 ...
```

5.4 演習

このレッスンではMySQL Workbenchについて学習しました。ここではMySQL Workbenchの使用方法について確認しましょう。

1. MySQL Workbenchをダウンロードしインストールする

ヒント：レッスン5の「5.1　MySQL Workbenchのインストール」を参照

2. MySQL WorkbenchからMySQLサーバーに接続し、Server Statusを確認する

ヒント：レッスン5の「5.2.1　新規接続定義の作成」、「5.2.2 MySQLサーバーの管理系機能」を参照

3. MySQL WorkbenchからMySQLサーバーに接続し、SQLエディタから次のコマンドを実行する

```
SELECT * FROM world.City;
```

ヒント：レッスン5の「5.2.4 MySQLデータベースを使った開発支援機能」を参照

4. MySQL WorkbenchからMySQLサーバーに接続し、リバース・エンジニアリング機能でworldデータベースをE/R図化する

ヒント：レッスン5の「5.2.5 E/R図を用いたスキーマ設計」を参照

解説

各演習問題のヒントで紹介している解説を参照してください。

レッスン

6

JSON データ型と JSON 関数

このレッスンでは MySQL 5.7 の新機能である
JSON データ型と JSON 関数について学習します。

6.1 JSON対応のメリットとは

MySQL 5.7でのJSON対応は、間違いなく目玉機能の1つとなっています。JSONドキュメントによるデータの管理や交換は急速な広がりを見せており、MySQLの機能やツールでもJSONドキュメントを利用しているものが複数出てきています。

JSONドキュメントをデータ保管の形式としているドキュメントデータベースが利用される場面も出てきましたが、既存のRDBMSと新たなドキュメントデータベースを併用すると、管理すべき対象が増えます。また、異なるデータベース間でのデータとして整合性を持たせることや関連を利用するには、すべてアプリケーション開発者が実装しなくてはならないため、処理が煩雑になります。

そこでMySQLでは、MySQLサーバーをRDBMSとドキュメントデータベースの「ハイブリッド型」データベースとして扱えるような改良を進めています。JSONデータ型とJSON関数が利用できるようになったこともその一環です。JSONドキュメント内の値とテーブル内の値をJOINすることや、JSONドキュメントとテーブルを1つのトランザクションで更新することなど、複数のデータベースでは実現しにくい点をMySQL内で処理できることは大きなメリットになります。

6.2 JSONデータ型

MySQL 5.7で実装したJSONデータ型によって、行と列の組み合わせである表で管理されるデータと、JSONドキュメントによる構造が不定なデータを同時に管理することが可能になりました。

6.2.1 JSONとは

JSONは、「JavaScript Object Notation」の略で、JavaScriptのオブジェクトの表記をベースとしたデータ表記形式です。キーと値の組み合わせや配列をテキストで記述します。JSONの記述方式には以下の特徴があります。

- JSONの配列は、値をカンマで区切り、大括弧（[]）で囲む
- JSONオブジェクトはコロンで区切られたキーと値のペアをカンマで区切り、全体を中括弧（{}）で囲む
- キーの名前と文字列は、ダブルクォーテーションで囲む
- JSON内の値としては、文字列型や数値型、ブール型、nullが利用可能
- オブジェクトを配列の要素とすることや、配列をオブジェクトの値とするようなネストが可能

JSONは、その形式からJavaScriptとの親和性が高いとされていましたが、現在はJavaScriptに限らず多くの開発言語で利用するためのライブラリが整備されたため、データ交換のフォーマットとして幅広く利用されています。

6.2.2 データ型としてのJSONの用途

RDBMSの表では表現しにくいデータを格納するためにJSONが利用されることがあります。例えば、属性情報の数が一定ではない場合、1つのカラムにカンマ区切りなどでデータを入れるか、属性情報の数を多めに見積もった予備カラムを用意するか、もしくは各属性情報を行として格納する別の表を用意することになります。ところが、JSONは配列を利用できネストすることもできるため、このようなデータが表現しやすい形式といえます（リスト1）。

リスト1　属性テーブルを使用した表現とJSONを使用した表現の例

・属性テーブルを使用した場合

```
mysql> SELECT * FROM pizza;
+------+-----------------+-------+
| code | name            | price |
+------+-----------------+-------+
| CLA  | Classic Pizza   |   400 |
| MAR  | Margherita Pizza|   500 |
+------+-----------------+-------+

mysql> SELECT * FROM toppings;
+--------+--------------+
| p_code | name         |
+--------+--------------+
| CLA    | Pepperoni    |
| CLA    | Gouda Cheese |
| MAR    | Basil        |
```

```
| MAR      | Mozzarella      |
+--------+---------------+

mysql> SELECT * FROM additionals;
+--------+-------+-------+
| p_code | name   | price |
+--------+-------+-------+
| MAR      | Olive |    100 |
+--------+-------+-------+
```

・JSONを使用した場合

```
{
    "name":"Classic Pizza","price":400,
    "toppings":[
        "Pepperoni","Gouda Cheese"
    ]
}
{
    "name":"Margherita Pizza","price":500,
    "toppings":[
        "Basil","Mozzarella"
    ],
    "additional":[
        {
            "name":"Olive","price":100
        }
    ]
}
```

RDBMSの表を利用するためには、あらかじめデータ構造を定義しておく必要があります。ところが、JSONのようなデータ形式では、あらかじめ構造を定義せずに利用できるため、アプリケーション開発者がすぐに開発に取りかかれるというメリットがあります。

しかし、データ構造の定義が不要で、自由な形式が取れるということは、どのようなデータが格納されるかはアプリケーション開発者に依存することになります。さらに、正規化や外部キーなどの概念もないため、データの重複の可能性や冗長なデータ構造に対する制限、関連するデータ間での制約などを設けることができない点に注意が必要です。

6.2.3 MySQLのJSONデータ型の特徴

MySQL 5.7から利用可能となったJSON型は、格納されるデータ形式が正しいかを自動的にチェックするDocument Validation機能を持ちます。MySQLでは、JSONデータ型内のデータを、文字コードはutf8mb4で、照合順序はutf8mb4_binとして扱います。ほかの文字コードで生成された値をJSONデータ型に格納する時は、自動的にutf8mb4に変換されます。ただし、文字コードasciiおよびutf8は、utf8mb4のサブセットとなるため、変換は行われません（リスト2）。

リスト2　JSONデータ型の列を持つテーブルの作成とデータの格納

```
# 同名のテーブルがあれば削除
mysql> DROP TABLE IF EXISTS pz;

# JSON型の列menuを持つテーブルを作成
mysql> CREATE TABLE pz (menu JSON);

# データを追加
# 複数行にわたるJSONの場合、改行を入れても特に問題はない
# JSON全体をシングルクォーテーションで囲む
# JSON内の文字列はダブルクォーテーションで囲む
mysql> INSERT INTO pz(menu) VALUES ('{
    '> "Name":"Plain Pizza",
    '> "price":300
    '> }');

# 配列を含むJSONを追加
mysql> INSERT INTO pz(menu) VALUES ('{
```

```
    '> "Name":"Cheesy Pizza",
    '> "price":400,
    '> "toppings":"More Cheese",
    '> "additionals":[{"Name":"B Cheese","price":100}]
    '> }');

# JSONではない文字列を追加しようとするとエラーになる
mysql> INSERT INTO pz(menu) VALUES ('some text');
ERROR 3140 (22032): Invalid JSON text: "Invalid value." at position
0 in value for column 'pz.menu'.
```

JSONドキュメントをそのままテキストとして格納するのではなく、バイナリ化して格納します。ネストされたオブジェクトやデータを探す際に、データ全体を走査することなく、直接キーや配列のインデックスを参照させることで、検索性能の向上を図っています。MySQLのJSONデータ型内データとして、JSONの文字列型や数値型などに加えて、独自に日時型もサポートしています。

MySQL 5.7で追加されたJSON関数およびJSON演算子によって、JSONデータ型の列の値の取得や変更が可能です（リスト3）。

リスト3　JSON関数およびJSON演算子の利用例

```
# JSONの内容を展開するJSON_EXTRACT関数の利用例
mysql> SELECT JSON_EXTRACT(menu, "$.Name") FROM pz;
+-----------------------------+
| JSON_EXTRACT(menu, "$.Name") |
+-----------------------------+
| "Plain Pizza"                |
```

```
| "Cheesy Pizza"                  |
| "Classic Pizza"                 |
+---------------------------------+

# JSON_EXTRACT関数と同様の動作をするJSON演算子 -> の利用例
mysql> SELECT menu->"$.Name" FROM pz;
+-----------------+
| menu->"$.Name"    |
+-----------------+
| "Plain Pizza"     |
| "Cheesy Pizza"    |
| "Classic Pizza" |
+-----------------+
```

もう1つの大きな特徴が、ほかのデータ型と同様に、インデックスに利用できることです。MySQL 5.7で加わった生成列（Generated Column）にJSONから抽出した値を格納して、その生成列にインデックスを作成することで実現できます（リスト4）。

リスト4　生成列の作成とインデックスの追加

```
# 生成列の作成
# JSON内のName要素を展開し、その値をVIRTUAL生成列とする
# VIRTUAL生成列の場合、毎回演算が行われ、値そのものは格納されない
# STORED生成列の場合、データの追加更新時に演算後の値が格納される
mysql> ALTER TABLE pz ADD COLUMN pz_name VARCHAR(32)
    -> GENERATED ALWAYS AS
    -> (JSON_UNQUOTE(JSON_EXTRACT(menu, '$.Name')))
    -> VIRTUAL;
```

```
# 作成した生成列に対してインデックスを作成
mysql> ALTER TABLE pz ADD INDEX(pz_name);

# レッスン14で説明するEXPLAINで実行計画を確認
# keyの項目でインデックスpz_nameが利用できていることが確認できる
mysql> EXPLAIN SELECT * FROM pz WHERE pz_name = "Cheesy Pizza"￥G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: pz
   partitions: NULL
         type: ref
possible_keys: pz_name
          key: pz_name
      key_len: 35
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0.00 sec)

Note (Code 1003): /* select#1 */ select `world`.`pz`.`menu` AS `menu`,`world`.`pz`.`pz_name` AS `pz_name` from `world`.`pz` where (`world`.`pz`.`pz_name` = 'Cheesy Pizza')
```

6.3 MySQLのJSON関数とJSON演算子

MySQL 5.7では、JSON文字列およびJSONデータ型を処理するための関数や演算子が複数用意されています。また多くのJSON関数では、JSONドキュメント内の要素を指定しますが、このためのJSONドキュメント内の階層（パス／Path）を表現する方法が用意されています。

6.3.1 JSONデータのパス式

MySQLのJSON関数で利用するパス式では、まず利用するJSONドキュメントをドル記号（$）で表します。その後にピリオドで区切って階層を表現していきます。配列内の要素は[n]で表現します。大括弧内のnは、0以上の整数または全要素を表すアスタリスク（*）を使えます。キーの名前はutf8mb4（またはasciiかutf8）でエンコードされた文字列をダブルクォーテーションで囲みます。

以下のようなJSON配列とJSON_EXTRACT関数を例にパス式の挙動を確認してみます（リスト5）。

[10,{"a":[20,30],"b":40},[50,60]]

リスト5　JSON配列に対するパス式の挙動

```
# セッション変数@j_ex1にJSON配列を格納する
mysql> SET @j_ex1='[10, {"a": [20, 30], "b": 40}, [50, 60]]';

# JSONドキュメント全体を取得
mysql> SELECT JSON_EXTRACT(@j_ex1, "$");
+----------------------------------------+
| JSON_EXTRACT(@j_ex1, "$")              |
+----------------------------------------+
| [10, {"a": [20, 30], "b": 40}, [50, 60]] |
+----------------------------------------+

# 配列のすべてを取得
mysql> SELECT JSON_EXTRACT(@j_ex1, "$[*]");
+----------------------------------------+
```

```
| JSON_EXTRACT(@j_ex1, "$[*]")                  |
+-----------------------------------------+
| [10, {"a": [20, 30], "b": 40}, [50, 60]] |
+-----------------------------------------+

# 配列の先頭の要素を取得
mysql> SELECT JSON_EXTRACT(@j_ex1, "$[0]");
+----------------------------+
| JSON_EXTRACT(@j_ex1, "$[0]") |
+----------------------------+
| 10                                  |
+----------------------------+

# 配列の2番目の要素を取得
# →JSONオブジェクトが返る
mysql> SELECT JSON_EXTRACT(@j_ex1, "$[1]");
+----------------------------+
| JSON_EXTRACT(@j_ex1, "$[1]") |
+----------------------------+
| {"a": [20, 30], "b": 40}       |
+----------------------------+

# 配列の2番目の要素（JSONオブジェクト）のうち、
# キーの名前がaの値を取得
# →JSON配列が返る
mysql> SELECT JSON_EXTRACT(@j_ex1, "$[1].a");
+----------------------------+
| JSON_EXTRACT(@j_ex1, "$[1].a") |
```

```
+------------------------------+
| [20, 30]                     |
+------------------------------+

# 配列の2番目の要素（JSONオブジェクト）のうち、
# キーの名前がaの値（JSON配列）の先頭の要素を取得
mysql> SELECT JSON_EXTRACT(@j_ex1, "$[1].a[0]");
+---------------------------------+
| JSON_EXTRACT(@j_ex1, "$[1].a[0]") |
+---------------------------------+
| 20                              |
+---------------------------------+
```

　別のJSONオブジェクトとJSON_EXTRACT関数を例に、パス式の挙動を確認してみます（リスト6）。

{"a": {"b": 1}, "c": {"b": [2,3]}, "d": [5, 6]}

リスト6　JSONオブジェクトに対するパス式の挙動

```
# セッション変数@j_ex2にJSONオブジェクトを格納する
mysql> SET @j_ex2='{"a": {"b": 1}, "c": {"b": [2,3]}, "d": [5, 6]}';

# キーの名前がaの値を取得→JSONオブジェクトが返る
mysql> SELECT JSON_EXTRACT(@j_ex2, '$.a');
+---------------------------+
| JSON_EXTRACT(@j_ex2, '$.a') |
+---------------------------+
```

```
| {"b": 1}                          |
+-----------------------------+

# キーの名前がbの値を取得
# →bの値は2階層目のキーなのでnullが返る
mysql> SELECT JSON_EXTRACT(@j_ex2, '$.b');
+-----------------------------+
| JSON_EXTRACT(@j_ex2, '$.b') |
+-----------------------------+
| NULL                              |
+-----------------------------+

# キーの名前がaの値（JSONオブジェクト）のうち
# キーの名前がbの値を取得
mysql> SELECT JSON_EXTRACT(@j_ex2, '$.a.b');
+-------------------------------+
| JSON_EXTRACT(@j_ex2, '$.a.b') |
+-------------------------------+
| 1                                  |
+-------------------------------+

# キーの名前がaの値（JSONオブジェクト）のうち
# キーの名前がcの値を取得
mysql> SELECT JSON_EXTRACT(@j_ex2, '$.c.b');
+-------------------------------+
| JSON_EXTRACT(@j_ex2, '$.c.b') |
+-------------------------------+
| [2, 3]                            |
```

```
+------------------------------+

# アスタリスク2つ（**）を指定するとa.bおよびc.bのように複数のパスを評価
mysql> SELECT JSON_EXTRACT(@j_ex2, '$**.b');
+------------------------------+
| JSON_EXTRACT(@j_ex2, '$**.b') |
+------------------------------+
| [1, [2, 3]]                  |
+------------------------------+

# さらにアスタリスク（*）を組み合わせてすべての値を展開
mysql> SELECT JSON_EXTRACT(@j_ex2, '$**.*');
+-------------------------------------------+
| JSON_EXTRACT(@j_ex2, '$**.*')             |
+-------------------------------------------+
| [{"b": 1}, {"b": [2, 3]}, [5, 6], 1, [2, 3]] |
+-------------------------------------------+
```

6.3.2 JSONデータを作成する関数

JSONデータを作成するための関数は以下の通りです。

表1　JSON データを作成する関数

関数名	概要
JSON_OBJECT	カンマで区切られた引数をキーと値として JSON オブジェクトを作成
JSON_ARRAY	カンマで区切られた引数から JSON 配列を作成
JSON_QUOTE	引数の JSON をクォーテーションで囲む

JSONオブジェクトを作成する例は、次の通りです（リスト7）。

リスト7　JSON_OBJECT関数によるJSONオブジェクトの作成

```
# JSON_OBJECT関数の引数として各種のSQL関数も利用可能
mysql>  SELECT  JSON_OBJECT("Title",  "Foobar",  "Timestamp",
CURTIME());
+-----------------------------------------------------+
| JSON_OBJECT("Title", "Foobar", "Timestamp", CURTIME()) |
+-----------------------------------------------------+
| {"Title": "Foobar", "Timestamp": "14:54:23.000000"}      |
+-----------------------------------------------------+
```

JSON_OBJECT関数は、SELECT文による検索結果をJSONオブジェクトとして返すためにも利用できます（リスト8）。

リスト8　SELECT文の結果をJSONオブジェクトとする例

```
mysql> SELECT
    ->   JSON_OBJECT("CityName", Name, "Country", CountryCode)
AS J
    -> FROM City WHERE district = 'Okinawa' LIMIT 3;
```

```
+----------------------------------------+
| J                                      |
+----------------------------------------+
| {"Country": "JPN", "CityName": "Naha"}      |
| {"Country": "JPN", "CityName": "Okinawa"} |
| {"Country": "JPN", "CityName": "Urasoe"}   |
+----------------------------------------+
```

6.3.3 JSONデータを検索する関数および演算子

JSONデータを検索するための関数および演算子は以下の通りです（表2）。JSONドキュメントの内容を展開するJSON_EXTRACT関数以外にも値が存在するか、またそのパスの情報などを確認するための関数が用意されています。リスト9に利用例を示します。

表2　JSON データを検索する関数

関数名	概要
JSON_CONTAINS	該当する値が指定したパスに存在するか否かの判定
JSON_CONTAINS_PATH	指定した複数のパスのうち、1 つまたはすべてが存在するか否かの判定
JSON_EXTRACT	指定したパスの値を展開
JSON_KEYS	キーを抽出。パスの指定も可能
JSON_SEARCH	与えられた文字列が存在するパスを返す

リスト9　JSON_CONTAINS関数およびJSON_CONTAINS_PATH関数の利用例

```
# 該当する値が指定したパスに存在するか
mysql> SELECT JSON_CONTAINS(menu, '400', '$.price') FROM pz;
+-------------------------------------+
| JSON_CONTAINS(menu, '400', '$.price') |
+-------------------------------------+
|                                   0 |
|                                   1 |
+-------------------------------------+

# 指定した複数のパスが1つでも存在するか
mysql>  SELECT  JSON_CONTAINS_PATH(menu,  'one',  '$.price',
'$.toppings') FROM pz;
+-----------------------------------------------------+
| JSON_CONTAINS_PATH(menu, 'one', '$.price', '$.toppings') |
+-----------------------------------------------------+
|                                                   1 |
```

```
|                                                          1 |
+------------------------------------------------------------+

# 指定した複数のパスがすべて存在するか
mysql> SELECT JSON_CONTAINS_PATH(menu, 'all', '$.price',
'$.toppings') FROM pz;
+-----------------------------------------------------------+
| JSON_CONTAINS_PATH(menu, 'all', '$.price', '$.toppings') |
+-----------------------------------------------------------+
|                                                          0 |
|                                                          1 |
+-----------------------------------------------------------+
```

JSON_EXTRACT関数で値を取得する代わりに、->演算子を使うことができます。JSON_EXTRACT関数および->演算子ともに、取得した値が文字列の場合はダブルクォーテーションで囲まれた値が返ります。クォーテーションをはずすためには、JSON_UNQUOTE関数を使います。MySQL 5.7.13で加わった->>演算子（大なり記号2個）は、クォーテーションをはずした値が返ります。以下の結果はいずれも同じです。

- SELECT JSON_UNQUOTE(JSON_EXTRACT(menu, "$.Name")) FROM pz;
- SELECT JSON_UNQUOTE(menu->"$.Name") FROM pz;
- SELECT menu->>"$.Name" FROM pz;

6.3.4 JSONデータを変更する関数

JSONドキュメントの変更した値を取得するための関数は、表3の通りです。JSONデータを検索する関数と同様にJSONドキュメント内のパスを指定して、変更や追加を行う位置を指定できます（リスト10）。

表3 JSON データを変更する関数

関数名	概要
JSON_ARRAY_APPEND	JSON ドキュメント内の配列の末尾に値を追加。追加位置のパスを指定可能
JSON_ARRAY_INSERT	JSON ドキュメント内の配列の先頭に値を追加。追加位置のパスを指定可能
JSON_INSERT	JSON ドキュメント内の指定したパスに値を追加
JSON_MERGE	複数の JSON ドキュメントを連結
JSON_REMOVE	JSON ドキュメント内の指定したパスの値を削除
JSON_REPLACE	JSON ドキュメント内の指定したパスの値を更新
JSON_SET	JSON ドキュメント内の指定したパスに値を更新、該当する値がなければ追加
JSON_UNQUOTE	JSON の値のクォーテーションをはずす

リスト10　JSON_ARRAY_APPEND関数とJSON_ARRAY_INSERTの比較

```
# 変数@j_ex1にJSON配列を設定
# 配列の要素の番号は0から始まることに注意
mysql> SET @j_ex1='[10, {"a": [20, 30], "b": 40}, [50, 60]]';

# JSON配列の2番目の要素の後に値を追加
# 配列の要素として追加される
mysql> SELECT JSON_ARRAY_APPEND(@j_ex1, '$[1]', 'x');
+-------------------------------------------+
| JSON_ARRAY_APPEND(@j_ex1, '$[1]', 'x')    |
+-------------------------------------------+
| [10, [{"a": [20, 30], "b": 40}, "x"], [50, 60]] |
+-------------------------------------------+

# JSON配列の2番目の要素の前に値を追加
```

```
mysql> SELECT JSON_ARRAY_INSERT(@j_ex1, '$[1]', 'x');
+-------------------------------------------+
| JSON_ARRAY_INSERT(@j_ex1, '$[1]', 'x')    |
+-------------------------------------------+
| [10, "x", {"a": [20, 30], "b": 40}, [50, 60]] |
+-------------------------------------------+

# 存在しない配列の位置を指定した場合はエラーとはならず、一番後に追加される
mysql> SELECT JSON_ARRAY_INSERT(@j_ex1, '$[10]', 'x');
+-------------------------------------------+
| JSON_ARRAY_INSERT(@j_ex1, '$[10]', 'x')   |
+-------------------------------------------+
| [10, {"a": [20, 30], "b": 40}, [50, 60], "x"] |
+-------------------------------------------+

# JSON配列の2番目の要素のキーaが持つ配列の2番目の要素(30)の後に値を追加
mysql> SELECT JSON_ARRAY_APPEND(@j_ex1, '$[1].a[1]', 'y');
+-------------------------------------------+
| JSON_ARRAY_APPEND(@j_ex1, '$[1].a[1]', 'y')  |
+-------------------------------------------+
| [10, {"a": [20, [30, "y"]], "b": 40}, [50, 60]] |
+-------------------------------------------+

# JSON配列の2番目の要素のキーaが持つ配列の2番目の要素(30)の前に値を追加
mysql> SELECT JSON_ARRAY_INSERT(@j_ex1, '$[1].a[1]', 'y');
```

```
+-------------------------------------------+
| JSON_ARRAY_INSERT(@j_ex1, '$[1].a[1]', 'y')    |
+-------------------------------------------+
| [10, {"a": [20, "y", 30], "b": 40}, [50, 60]] |
+-------------------------------------------+
```

JSON_INSERT関数、JSON_REPLACE関数、JSON_SET関数は類似した関数ですが、それぞれJSONドキュメントに対するINSERT文、UPDATE文、REPLACE文に該当すると考えると違いを理解できるかと思います（リスト11）。

リスト11　JSON_REPLACE関数による値の変更

```
# リスト2で作成したデータの"Plain Pizza"の"price"を確認
mysql> SELECT JSON_EXTRACT(menu, '$.price') FROM pz WHERE
menu->>"$.Name" = "Plain Pizza";
+-----------------------------+
| JSON_EXTRACT(menu, '$.price') |
+-----------------------------+
| 300                         |
+-----------------------------+
1 row in set (0.00 sec)

# "Plain Pizza"の"price"を300から400に変更した値を取得
mysql> SELECT JSON_REPLACE(menu, '$.price', 400) FROM pz
WHERE menu->>"$.Name" = "Plain Pizza";
+-------------------------------------------------------------------------
+
```

```
|              JSON_REPLACE(menu,              '$.price',              400)
|
+-------------------------------------------------------------------------
+
|  {"Name":  "Plain  Pizza",  "price":  400,  "toppings":  null,
"additionals": null} |
+-------------------------------------------------------------------------
+
1 row in set (0.00 sec)

# テーブルに格納されている値の再確認
# あくまでもJSON_REPLACE関数の実行結果として表示されている値のみ
が変わっており、
# テーブルのデータの変更はUPDATE文などを利用する
mysql>  SELECT  JSON_EXTRACT(menu,  '$.price')  FROM  pz  WHERE
menu->>"$.Name" = "Plain Pizza";
+-----------------------------+
| JSON_EXTRACT(menu, '$.price') |
+-----------------------------+
| 300                                |
+-----------------------------+
1 row in set (0.00 sec)
```

Column　**NoSQL APIとmemcachedプラグイン**

　MySQLには、キーバリュー型のNoSQLインターフェースが存在します。MySQLのデータは通常SQL文を使って読み書きをしますが、

このインターフェースではSQL文を使わないデータアクセスが可能です。MySQLのNoSQLインターフェースはオープンソースの分散キャッシュmemcachedを活用しています。

memcachedは、ブログサイトの「LiveJournal」で利用が始まった分散キャッシュの仕組みで、現在では多くのWebサイトでMySQLと組み合わせて利用されています。単純にMySQLとmemcachedを組み合わせた場合では、データの取得時にアプリケーションからmemcachedにアクセスして、対象のデータがキャッシュされているか確認し、データがない場合は改めてMySQLにアクセスする必要があります。また、データの更新時には、永続化のためにMySQLにアクセスしつつ、さらにキャッシュを更新する処理をなんらかの形で実装する必要があります。

memcachedはMySQLサーバーのプラグインとしてMySQLサーバーと同一プロセス内部で稼働しています。データはInnoDBのテーブルに永続化されます。この場合、memcachedのプロトコルを使ったクライアントプログラムは、memcachedに対してデータを読み書きするように動作しますが、実際にはInnoDBのテーブルのデータを読み書きする仕組みとなります。

設定の中で、キーのプレフィックスによってどのテーブルを利用するか、テーブル内のどの列をキーおよびバリューとするかを定義できます。テーブルの複数の列をまとめてバリューとしてmemcachedプロトコルから扱うこともできます。

memcachedのプロトコルでデータを追加更新する場合、「memcachedへのキャッシュのみ」「InnoDBへの永続化のみ」「キャッシュと永続化」を選択できます。これによってキャッシュ

のデータとMySQLのデータの同期を心配せず、自動的に同期が取れる構成にすることが可能です。また、MySQLサーバーのバイナリログにはデータの変更が記録されるため、レプリケーションも可能です。

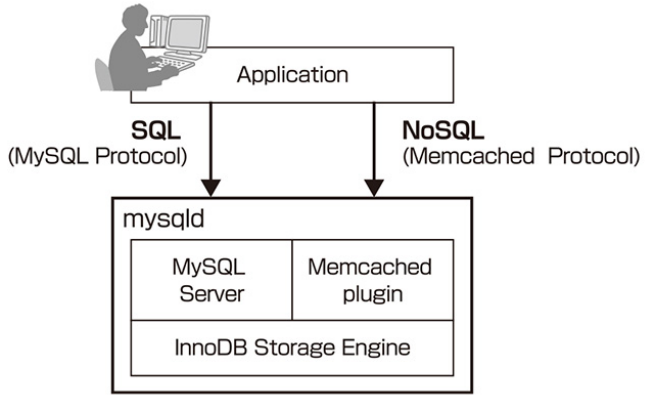


図1　InnoDB memcached プラグイン

同じデータに対してキーバリュー型のアクセスが適しているシンプルなデータアクセスをするアプリケーションからはmemcachedのプロトコルで、JOINなど複雑なデータ操作の場合にはMySQLサーバーのプロトコルでSQL文を利用します。特にデータをまとめてロードする場合などのオーバーヘッドを最小限に抑えることができます。

6.4 演習

このレッスンではMySQL 5.7で加わったJSONデータ型とJSON関数について学習しました。学習した内容を演習で確認しましょう。

1. world_xデータベースを作成し、JSONデータを含むサンプルデータ world_x database をダウンロードする。world_x databaseは以下のページの「Example Databases」の一覧で、TGZまたはZipファイルが選択可能

●URL：Other MySQL Documentation

http://dev.mysql.com/doc/index-other.html

ヒント：レッスン4の演習を参照

2. world_xデータベース内でサンプルデータの一部を確認する

```
mysql> use world_x;

mysql> SELECT * FROM CountryInfo LIMIT 2;
```

3. JSON_EXTRACT関数、->演算子および、->>演算子の挙動を確認する

```
mysql> SELECT JSON_EXTRACT(doc, "$.Name") FROM CountryInfo LIMIT 2;

mysql> SELECT doc->"$.Name" FROM CountryInfo LIMIT 2;
```

```
mysql> SELECT doc->>"$.Name" FROM CountryInfo LIMIT 2;
```

4. サンプルデータ内のCountryInfoテーブルから以下の条件で値を抽出しJSONオブジェクトを作成する

●GNPの値が1,000,000より大きい国

●国名と人口をJSONオブジェクトとして取得

```
mysql> SELECT JSON_OBJECT
    ->      ("Country Name", doc->"$.Name",
    ->        "Population", doc->"$.demographics.Population"
    ->      )
    -> FROM CountryInfo WHERE doc->"$.GNP" > 1000000;
```

5. CountryInfoテーブルとCityテーブルの値をJOINして値を取得する

●都市の人口が1,000万を超える都市名とその国名を取得する

```
mysql> SELECT
    ->      CountryInfo.doc->>"$.Name" AS CountryName,
    ->      City.Name AS CityName
    -> FROM CountryInfo, City
    -> WHERE
    ->      CountryInfo.doc->>"$._id" = City.CountryCode
    ->      AND
    ->      City.Info->>"$.Population" > 10000000;
```

解説

この演習では、JSONドキュメントデータを実際にMySQLサーバーに格納して、JSON関数や演算子を利用しています。このサンプルデータは、すべての項目数が同じドキュメントばかりが含まれており、JSONドキュメントを利用したデータベースの特性の1つである不定型な構造のデータにはなっていません。

例えば、中国の特別行政区である香港のデータが含まれていますが、独立国家ではないため独立年を表すIndepYearがnullになっています。RDBMSのテーブルの列としてIndepYearが定義されていた場合はnullを格納することになりますが、JSONドキュメントでは項目をそろえる必要性がないため、次のように変更した上で格納することも可能です。

```
# サンプルデータの内容
{"GNP": 166448, "_id": "HKG", "Name": "Hong Kong", "IndepYear": null, "geography": {"Region": "Eastern Asia", "Continent": "Asia", "SurfaceArea": 1075}, "government": {"HeadOfState": "Jiang Zemin", "GovernmentForm": "Special Administrative Region of China"}, "demographics": {"Population": 6782000, "LifeExpectancy": 79.5}}

# 変更後の内容
{"GNP": 166448, "_id": "HKG", "Name": "Hong Kong", "geography": {"Region": "Eastern Asia", "Continent": "Asia", "SurfaceArea": 1075}, "government": {"HeadOfState": "Jiang Zemin", "GovernmentForm": "Special Administrative Region of China"}, "demographics": {"Population": 6782000, "LifeExpectancy": 79.5}}
```

1. 実行するコマンドの例は以下の通りとなります。world_x.sqlファイルの指定は展開したディレクトリにあわせてください。

```
# world_xデータベースの作成
$ ./mysqladmin -uroot create world_x

# world_xデータベースにサンプルデータをロード
$ ./mysql -uroot world_x ￥
```

2. 演算子の違いで文字列のクォーテーションの有無が異なります。クォーテーションが付くのは文字列型のみです。

3. この演習では、JSONドキュメントの内容を抽出した上で、JSON_OBJECT関数を使って改めてJSONオブジェクトを生成しています。以下のSQL文にすると、JSONドキュメントの内容を通常のSELECT文の結果セットして受け取ることができます。演習3にもあるように国名の取得には->>演算子を用いてクォーテーションをはずしている点も確認してください。

```
mysql> SELECT
    ->   doc->>"$.Name" AS "Country Name",
    ->   doc->"$.demographics.Population" AS "Population"
    -> FROM CountryInfo WHERE doc->"$.GNP" > 1000000;
+---------------+------------+
| Country Name    | Population |
+---------------+------------+
| Germany           | 82164700    |
```

```
| France         | 59225700    |
| United Kingdom | 59623400    |
| Italy          | 57680000    |
| Japan          | 126714000   |
| United States  | 278357000   |
+----------------+-------------+
```

4. この例では、国のコードが格納されているCountryInfoテーブルのJSONドキュメント内の_idの値とCityテーブルのCountryCode列の値でJOINを行っています。このように、MySQLではJSONドキュメント内の値とテーブル内の値でJOINが可能となっています。

レッスン

7

バックアップとリカバリ 基礎編

システム全体のデータを蓄積する観点からデータベースの保護は非常に重要な運用時の要件となります。このレッスンでは MySQL サーバーのバックアップとリカバリの基礎について学習します。

7.1 バックアップの重要性

パソコンやスマートフォン、アプリケーション開発リポジトリなど、データベースに限らずさまざまな領域でバックアップの重要性がうたわれています。特に、データベースにはシステム全体で必要となるデータが格納されており、データを失ってしまうと事業が継続できなくなる恐れが出てきます。

そこで、ディスクのRAID構成やデータ格納先を複数にすることで、機器の故障やクラウドサービスの停止に対応する方法も考えられます。しかし、アプリケーションの更新時に埋め込まれたバグによって、誤ったSQL文が実行されて不正なデータができてしまうケースや、故意か過失かを問わずデータベース管理者の操作によってデータを失ってしまうケースもあり得ます。このような時に過去の一定時点までデータを復元するためにも、データのバックアップが必要となります。

7.1.1 バックアップ取得時の考慮点

運用中のシステムのバックアップを行うにあたって考慮すべき事項は以下の通りです。

- いつ：1日の中で何時頃？　どのぐらいの間隔で？
- なにを：OSからデータベース全体まですべて？　変更されたデータだけ？
- どこに：同じサーバー上？　テープデバイス？　別のデータセンター？　クラウドストレージ？
- どのように：システムは停止必要？　どのツールや手段を使う？
- どれだけ：何世代のバックアップデータを残しておく？　どれだけのストレージが必要か？

7.1.2 バックアップ対象の検討

バックアップ対象のデータと設定は、以下のように検討します。

「いつ」バックアップするか

バックアップ処理は、データベースの内容をなんらかの方法で複製することになります。複製を行うためにデータを読み取る必要があり、そのぶんのディスクアクセスが発生する可能性があります。また、単にデータのコピーを行うのではないケースではCPUにも負荷がかかり、バックアップデータを別のサーバーやクラウドストレージなどに転送する場合はネットワークにも負荷がかかります。アプリケーションからのアクセスへの影響を最小限に抑えるためにも、バックアップを実行するタイミングの検討が重要です。

一般的には、アプリケーションからデータベースへのアクセスが少なくなることが想定される時間帯にバックアップを行います。業務系のシステムでは業務時間外にはアプリケーション利用者が減りますが、分析やデータの洗い替えといった夜間バッチなどが実行される可能性がある点も考慮する必要があります。オンラインゲームのようなコンシューマー向けのサービスでは、夜間も負荷が高いままのシステムもあり得ます。バックアップが別の処理で負荷が高まる前に設計することも重要です。

「なにを」バックアップするか

バックアップ対象は、データベースに格納されたアプリケーションのデータだけではありません。バックアップや復旧にかかる時間、データ変更の頻度や有無によって、データベース全体をバック

アップするのか、変更点のみをバックアップするのか、またはデータベースの特定のテーブルだけをバックアップするのかなどの設計を行います。

　また、設定を変更する前には、設定ファイルのバックアップを取っておくべきです。MySQL製品のバイナリは、多くの場合でMySQLの公式サイトからダウンロード可能となっていますが、例えばバージョンアップ後になんらかの事情で元のバージョンに戻すことも考えられますので、環境をバックアップすることも検討に含めます。

　過去のMySQL製品は以下のサイトからダウンロードできます。ただし、MySQLサーバーについてはMySQL 5.0.15が最も古いものになります。

- **URL：MySQL Product Archives**
  http://downloads.mysql.com/archives/

「どこに」保管するか

　バックアップしたデータを稼働中のホスト上に置いていた場合、障害に巻き込まれると復旧のためにバックアップデータを利用することができません。また、アプリケーションデータの格納先に使えたはずのストレージの容量を使ってしまうことにもなりかねません。そこで、通常、バックアップデータは、外部ストレージやテープデバイス、クラウドストレージに待避しておきます。

　同時に考えておかなければいけないのは、復旧が必要となった際に、早急にバックアップデータを利用できるかです。バックアップデータを置いたストレージの性能や、クラウドストレージとの間の

ネットワークの転送性能などによって、取得に時間がかかってしまうと、そのぶんシステムの復旧が遅れてしまう懸念もあります。

「どのように」バックアップするか

　このレッスンとレッスン8では、特にこの項目について解説します。単にツールの機能だけではなく、ここで挙げられているほかの項目についても検討した上で最適な方法を選択してください。

「どれだけ」残すのか

　バックアップを行うと、なんらかのストレージの容量を必要とします。しかし、容量を節約するため、最新のバックアップデータだけを持っていると、例えばそのバックアップデータに誤ったデータが含まれている場合や、必要なデータが削除された状態でバックアップが行われた場合の復旧には使えません。そのため、一般的には、過去の世代のバックアップデータも保管されています。現在は、物理ストレージデバイスやクラウドストレージの容量が大きく、かつ単価がきわめて低くなっているので、以前ほどシビアな設計は不要かもしません。しかし、それでも物理ストレージデバイスには上限がありますし、また運用コストの観点からも、際限なく過去のバックアップを残すべきではありません。

7.2 バックアップ用語の整理

バックアップの種類や運用方法は、運用の要件によって変わってきます。ここでバックアップ手法の用語を整理しておきましょう。なお、用語に関しては製品によって異なる場合もありますが、本レッスンにおける用語の用法を紹介します。

7.2.1 オンラインバックアップとオフラインバックアップ

「オンラインバックアップ」は、対象となるシステムを停止せずに行うバックアップのことです。バックアップ中もアプリケーションの処理は通常通り継続できるタイプを「ホットバックアップ」と呼ぶこともあります。

「オフラインバックアップ」は、バックアップ作業中に一時的にシステムを停止する方法を指し、「コールドバックアップ」と呼ぶこともあります（図1）。

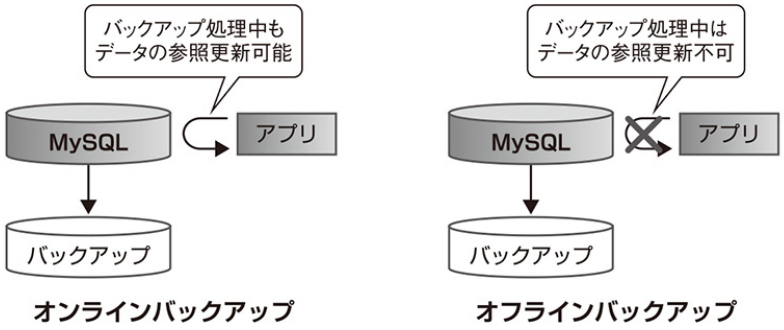


図1　オンラインバックアップとオフラインバックアップ

システムは稼働中ではあるものの、データを読み取り専用として運用するなど、アプリケーションの機能に制限を加えた状態で行うバックアップを、MySQLの資料ではホットとコールドの間ということで「ウォーム（Warm）バックアップ」と表現しています（図2）。

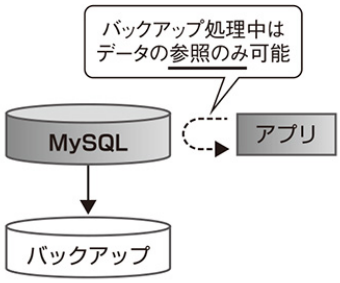


図 2　ウォームバックアップ

7.2.2 物理バックアップと論理バックアップ

データファイルやデータイメージをそのままバックアップする方式を「物理バックアップ」、データをテキストファイルなどにエクスポートしてバックアップする方式を「論理バックアップ」といいます（図3）。

一般的に、物理バックアップの方が高速にバックアップを行えますが、異なるデータベースに復元することはできず、異なったバージョン間でデータフォーマットが変更されていると、復元が難しいこともあります。

論理バックアップでエクスポートしたデータは、基本的にテキストデータとなるため、エディタなどで編集することもでき、柔軟性は高くなります。

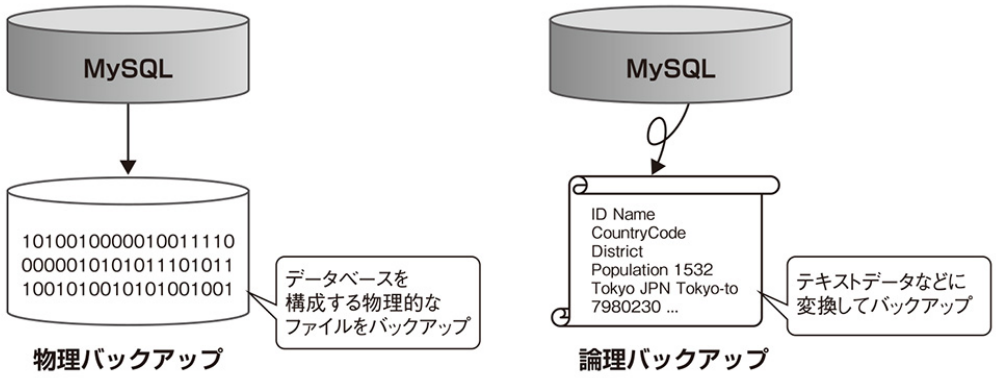


図3　物理バックアップと論理バックアップ

7.2.3　フルバックアップと増分バックアップと差分バックアップ

「フルバックアップ」は、データベース全体をまとめてバックアップします。データを復元するリストアの作業は、基本的にフルバックアップのデータを戻すだけで完了します。フルバックアップは、シンプルな操作でバックアップとリストアができますが、バックアップすべきデータサイズが大きく、バックアップにかかる時間も長くなります（図4）。

前回バックアップした時点から変更されたデータのみをバックアップするのが、「増分バックアップ」です。増分バックアップを行った場合のリストア作業では、フルバックアップしたデータがあれば、まずそのデータを戻し、その後で増分バックアップを行った順に1つずつデータを元に戻していきます。

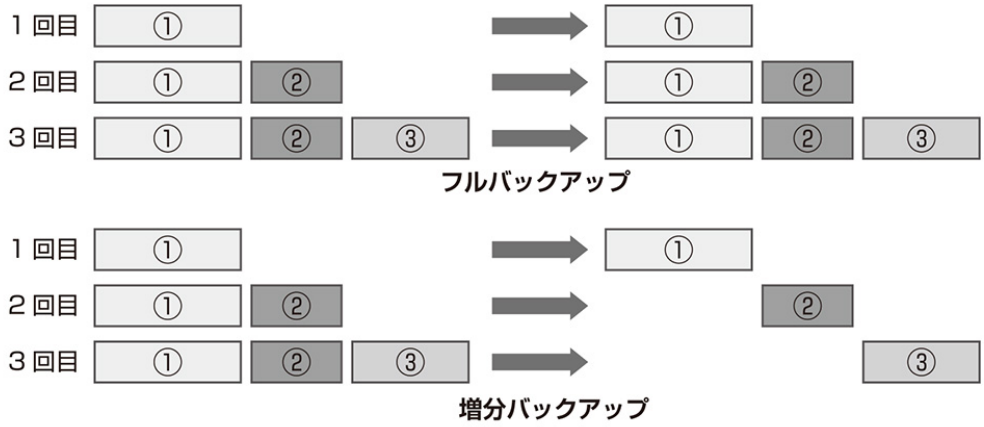


図４　フルバックアップと増分バックアップ

「差分バックアップ」では、前回のフルバックアップ以降に変更されたデータをまとめてバックアップします（図5）。

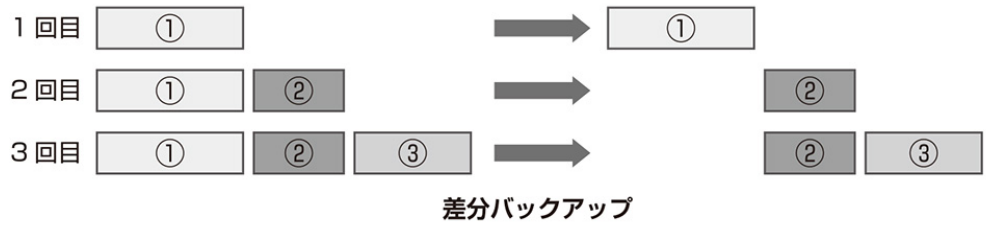


図5　差分バックアップ

フルバックアップと増分バックアップ、差分バックアップのメリットやデメリットは次の通りです（表1）。

表1　処理イメージとメリット／デメリット

	フルバックアップ	増分バックアップ	差分バックアップ
データサイズ	大	小	中
バックアップ作業の煩雑さ	低	高（前回バックアップ時点からの変更点の確認が必要）	高（前回フルバックアップ時点からの変更点の確認が必要）
リストア作業の煩雑さ	低（フルバックアップデータのみを復元）	高（フルバックアップデータとすべての増分バックアップを順番に復元）	中（フルバックアップデータと差分バックアップを順番に復元）

7.2.4 ローカルバックアップとリモートバックアップ

「ローカルバックアップ」は、同一ホスト上でバックアップ作業を行います（図6）。mysqldumpなど各種ツールでは、リモートのMySQLサーバーからのバックアップが行えます。MySQL Enterprise Backupでは、ローカルバックアップとして作業を行うものの、バックアップファイルの格納先をファイルサーバーやクラウドストレージといったリモートホストにする「リモートバックアップ」も選択できます。

バックアップしたデータは、MySQLサーバーと同一ホスト上に保管しておくべきではありません。バックアップ処理直後に、一時的に置いておくのであれば大きな問題にはならないかもしれませんが、障害が発生した際に、バックアップデータも巻き込まれてしまうと、復旧に利用できないため意味がありません。ローカルバックアップの場合でも、バックアップデータは外部ストレージやテープデバイスなど、物理的に別の装置や、クラウドストレージなどに保管することをおすすめします。

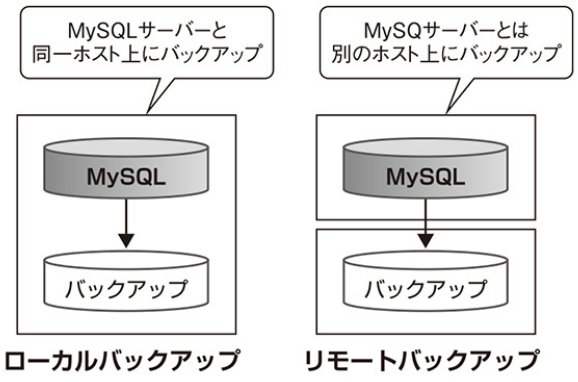


図6　ローカルバックアップとリモートバックアップ

7.3 データの復旧

バックアップしたデータを使ってシステムの復旧を行う作業は、バックアップしたデータを元のデータファイルに戻す「リストア」と、障害直前の状態まで戻す「リカバリ」に分類されます。データ復旧の際には、復旧にかけることができる時間およびどの時点まで復旧する必要があるのかを、システムの運用要件から定義しておくことが必要です。

7.3.1 リストアとリカバリ

ディスク障害などによってデータファイルが破損した時や、MySQLサーバーのデータファイルを喪失した場合などに、バックアップしたデータを元に戻す作業がリストアです。物理バックアップを行った場合には、基本的にバックアップファイルを元のデータファイルのあったディレクトリにコピーします。論理バックアップは、データをロードし直すため、復旧時間は物理バックアップよりかなり長くなる傾向にあります。

リカバリでは、データがバックアップされた以降に行われた変更をバイナリログから抽出して再現し、障害発生直前の状態にします。MySQLでは、リカバリのためにバイナリログが有効になっていることが必須です。また、データファイルとバイナリログが同時に壊れてしまうとリカバリができないため、物理的なディスクが複数利用できる場合には、別々のディスクに配置します（図7）。

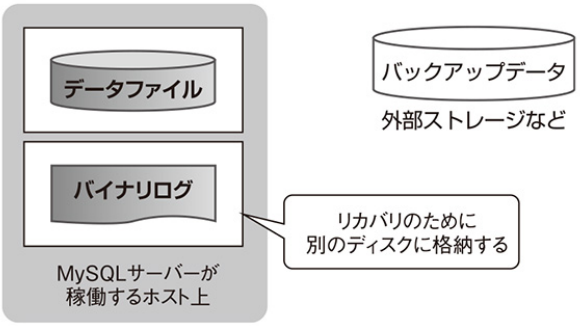


図7 データファイルとバイナリログ、バックアップデータの配置

MySQLのInnoDBストレージエンジンには、MySQLサーバーの起動時に、自動的にInnoDBのデータファイルとログファイルの同期を取るクラッシュリカバリ処理があります（図8）。プロセス障害などでMySQLサーバーが正しく停止されていない状態では、InnoDBログが

COMMIT時にトランザクションを記録したものの、チェックポイントでデータファイルが変更される前にMySQLサーバーが停止している可能性があります。その差分がないかをチェックし、差分があればクラッシュリカバリ処理を行います。クラッシュリカバリが行われている様子は、MySQLサーバーの起動時に出力されるログに「Starting crash recovery...」および「Crash recovery finished.」などと表示されることで確認できます。

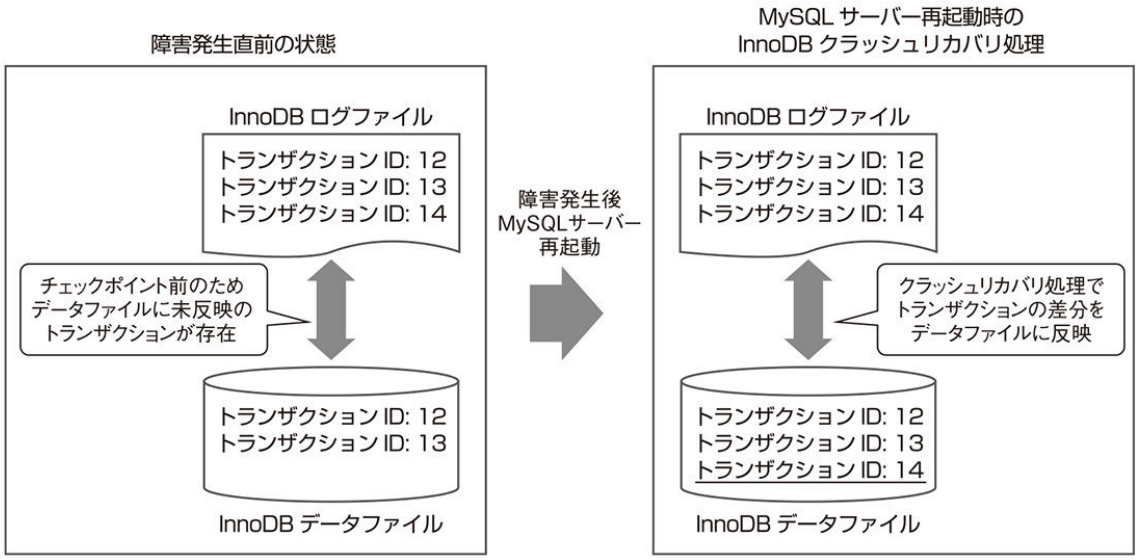


図 8　InnoDB のクラッシュリカバリ処理

7.3.2 RTOとRPO

リカバリの要件として、「復旧作業として許容できる時間」Recovery Time Objective（RTO、目標復旧時間）と、「データを復旧する必要がある障害発生前の時点」Recovery Point Objective（RPO、目標復旧時点）の2つを定義します（図9）。例えば、RTOの要件が1時間以内の場合、リストアとリカバリをあわせて1時間以内に終わる方法を利用する必要があります。RPOが前回のバックアップ時点までのデータに戻れば十分な場合は、リストアのみを行えば要件は満たせるため、リカバリのためのバイナリログがなくても問題ありません。一方で障害発生直前の状態までデータを復旧することが必要な要件では、バイナリログは必須となります。

MySQLでの具体的なバックアップリカバリコマンド等はレッスン8の応用編で紹介します。

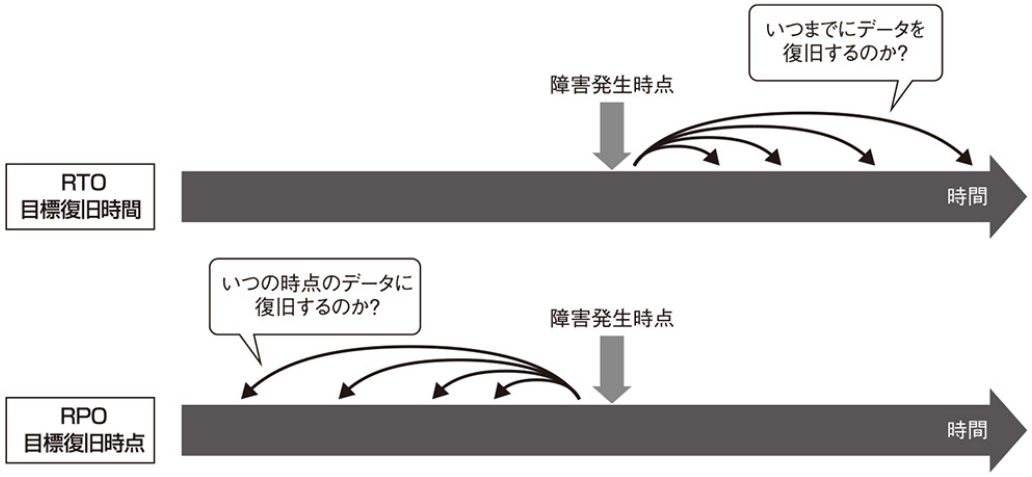


図9 RTO と RPO

Column 日本のMySQLコミュニティ

1997年から1998年にかけて、MySQLのリファレンスマニュアルの翻訳や日本語対応パッチが公開され、メーリングリストが開始されたのが日本でのMySQLコミュニティの黎明期となります。

2000年3月には、「日本MySQLユーザ会」（MySQL Nippon Association、略称MyNA）が設立されました。メーリングリストでの意見交換や、不定期に開催される勉強会、全国各地で開催されるオープンソースカンファレンスでのブース出展や講演が活動の中心となっています。日本語に関する機能要望をまとめてMySQL開発チームに渡すことも行われてきました。ユーザ会メンバーが共同で執筆した書籍も複数あります。MySQLに興味がある方なら誰でも参加でき、メーリングリストに参加することで入会したことになります。

- **URL：日本MySQLユーザ会**
  http://mysql.gr.jp

2010年には、MySQLについてカジュアルな雰囲気の中でつながりを作るという方向性のコミュニティ「MySQL Casual」としての動きが始まりました。最先端の現場でのとがった利用事例から特定の機能について深く掘り下げて調査した内容、さまざまなパッチやツールの情報などを共有するイベントMySQL Casual Talksが開催されてきました。主な活動は、コミュニケーションツールのSlack上で行われています。なにげないMySQLの話題からバグと思

われる挙動の詳細な調査まで、ゆるい雰囲気の中でディープなやりとりが行われています。

- **URL：MySQL Casual**
  http://mysql-casual.org

これらは独立したコミュニティではなく、どちらにも参加している方々が多く、どちらもゆるい雰囲気で運営しているので、参加しやすいと思います。

レッスン

8

バックアップとリカバリ 応用編

レッスン７ではバックアップとリカバリの基礎について学習しました。このレッスンでは、関連するMySQLのバックアップリカバリ方法、およびバックアップリカバリツールの具体的な使い方をパターンごとに学習します。

8.1 データのフルバックアップに関する方法とツール

MySQLサーバーのデータのフルバックアップには、複数の方法が選択できます。

1. 物理バックアップかつオフラインバックアップ

⇒OSのコピーコマンド等

2. 物理バックアップかつオンラインバックアップ

⇒MySQLEnterpriseBackup

3. 論理バックアップかつオンラインバックアップ

⇒mysqldumpまたはmysqlpump

なお、MySQLの論理バックアップにはMySQLサーバーが稼働している必要があるため、MySQLでは「論理バックアップかつオフラインバックアップ」は不可能です。

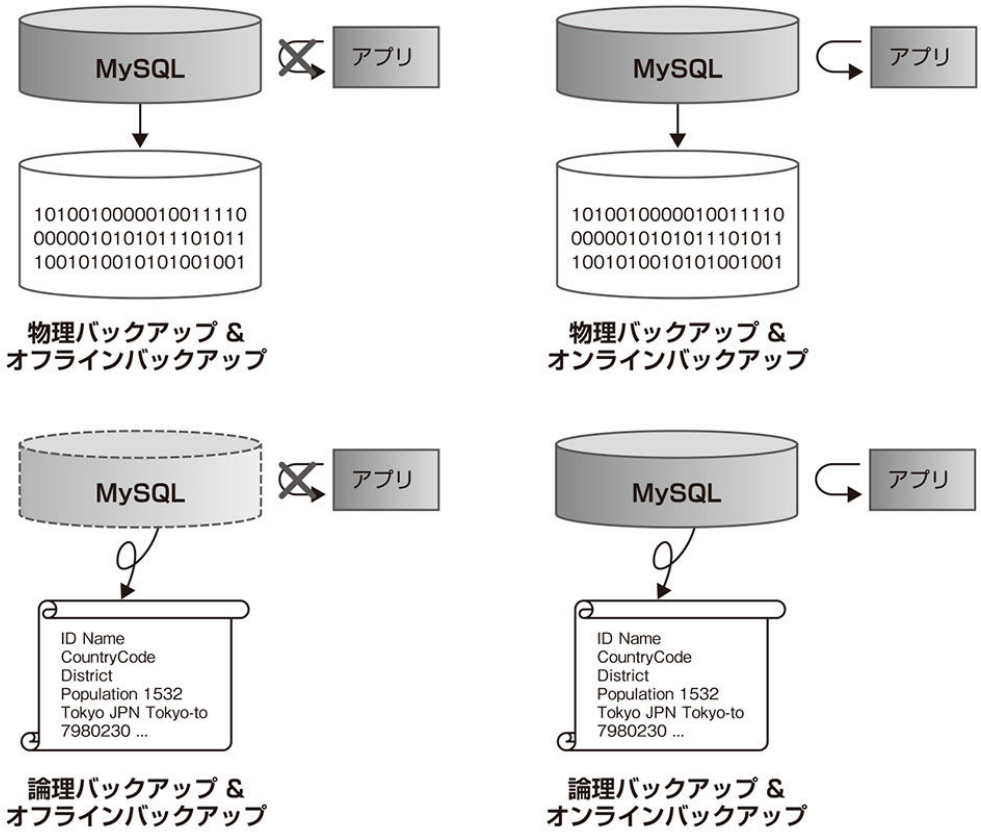


図 1　データのフルバックアップに関する方法とツール

8.2 物理バックアップかつオフラインバックアップ

MySQLのバックアップで最もシンプルな方法は、MySQLサーバーを停止して、データディレクトリ全体をOSのコピーコマンド等でコピーすることです。バックアップ作業中にMySQLサーバーを停止できる運用要件であれば、この方法が一番楽です。リストアもバックアップしたファイルをMySQLサーバーのデータディレクトリに戻すだけなので、かかる時間は基本的にファイルのコピー時間だけで済みます。

8.3 物理バックアップかつオンラインバックアップ

MySQL Enterprise Editionで提供されるMySQL Enterprise Backupは、MySQLサーバーの稼働中でもアプリケーションの処理に影響を与えずに、InnoDBのデータファイルの物理バックアップが行えます。

内部的には3段階でバックアップを行っています。まず、InnoDBのデータファイルのコピーを行います。データファイルのコピー中もデータの変更が行われている可能性があるため、データファイル内ではデータの整合性が取れていない可能性があります。そこで次の段階として、データファイルのコピー開始時点以降のInnoDBのトランザクション情報をInnoDBログから取得します。最後にInnoDBログのトランザクション情報を基に、データファイルコピー中に行われた変更点をデータに反映し、バックアップ終了時点のデータの整合性が取れたデータとなります。

また、MySQL Enterprise Backupは、後述するデータファイルの差分バックアップのほか、コマンド1つでポイントインタイムリカバリなども行うことが可能です。

リスト1　バックアップ取得とログ適用を1回の操作で実行

```
$ mysqlbackup --user=root -p --backup-dir=/home/admin/backups backup-and-apply-log
```

リスト2　ログ適用済みのバックアップをリストア

```
$ mysqlbackup --defaults-file=/usr/local/mysql/my.cnf --backup-dir=/home/admin/backups copy-back
```

リスト3　バックアップ取得のみ実行

```
$ mysqlbackup --user=root -p --backup-dir=/home/admin/backups backup
```

リスト4　ログの適用のみ実行

```
$ mysqlbackup --user=root -p --backup-dir=/home/admin/backups apply-log
```

リスト5　ログ適用前のバックアップにログを適用し、リストア

```
$ mysqlbackup --defaults-file=/usr/local/mysql/my.cnf --backup-dir=/export/backups/full copy-back-and-apply-log
```

8.4 論理バックアップかつオンラインバックアップ

MySQLのコミュニティ版では、論理バックアップのクライアントプログラムとしてmysqldumpとmysqlpumpがあります。どちらのツールでも、データの一貫性を保ったバックアップが可能です。バックアップ対象のテーブルがすべてInnoDBストレージエンジンのみを利用している場合はオンラインバックアップを行い、MyISAMやMEMORYなどトランザクションをサポートしないストレージエンジンのテーブルが含まれる場合にはウォームバックアップを行います。

mysqldumpは、MySQL 3.23から利用できるツールです。一方で、mysqlpumpはMySQL 5.7から利用できるようになった新しいツールで、データのエクスポート処理の並列化や進捗を表示できます。これらのツールでエクスポートしたデータなどをMySQLサーバーにロードするためには、mysqlやmysqlimportコマンドを利用します。

8.4.1 MySQLの論理バックアップツールmysqldump

mysqldumpは、MySQLの論理バックアップのためのクライアントプログラムとして使われてきました。デフォルトでは、テーブルとデータを復元するためのCREATE TABLE文とINSERT文の形式で、データをエクスポートします。--all-databasesオプションによって、MySQLサーバー全体、またはデータベース単位、個別のテーブル単位まで指定して部分をバックアップするか選択できます。また、タブ区切りやカンマ区切りなど任意の文字で区切ったテキストファイルをテーブル単位で出力することもできます。

リスト6　データベース単位とテーブル単位での出力方法

```
# データベース単位での指定
$ ./mysqldump -uroot world

# --databasesまたは-Bオプションではスペースで区切りで複数のデータベース指定可能
$ ./mysqldump -uroot --databases world world_x
<略>

# --tableオプションでテーブル単位で指定
# スペースで区切りで複数指定可能
$ ./mysqldump -uroot --databases world --tables City Country
<略>
# または
$ ./mysqldump -uroot world City Country
<略>
```

バックアップ対象のテーブルがすべてInnoDBストレージエンジンのみを利用している場合、mysqldumpの--single-transactionオプションを指定してバックアップ作業を1つのトランザクションとすることで、データの一貫性を保った上でオンラインバックアップが可能です（リスト7）。MyISAMやMEMORYなどトランザクションをサポートしないストレージエンジンのテーブルが含まれる場合には、--lock-all-tablesオプションを指定してすべてのテーブルを読み取り専用にして一貫性を保った上でバックアップができます（リスト8）。これがMySQLでのウォームバックアップに該当します。

mysqldumpは、MySQLサーバーに添付されているツールでもあり、多くのシステムで利用実績があります。一方でバックアップやリストアの性能は、物理バックアップと比較すると大きく劣るため、処理時間が要件を満たせないことが危惧されます。システム構成にもよりますが、データサイズがおおむね数十GB程度以上の場合には、mysqldumpではなく各種の物理バックアップでの運用を検討してください。

リスト7　基本：InnoDBのトランザクションを使用してデータベース全体のバックアップを取得

```
$ ./mysqldump --user=root --password=root --master-data=2 ＼
    --socket=/usr/local/mysql/data/mysql.sock ＼
    --hex-blob --default-character-set=utf8 --all-databases ＼
    --single-transaction > mysql_bkup_dump.sql
```

リスト8　例外：すべてのテーブルをロックしてデータベース全体のバックアップを取得

```
$ ./mysqldump --user=root --password=root --master-data=2 ＼
      --socket=/usr/local/mysql/data/mysql.sock ＼
      --hex-blob --default-character-set=utf8 --all-databases ＼
      --lock-all-tables > mysql_bkup_dump.sql
```

mysqldumpの代表的なオプションは以下の通りです（表1）。

表 1　mysqldump の代表的なオプション

オプション名	概要
--add-drop-database	各 CREATE DATABASE 文に DROP DATABASE を追加
--add-drop-table	各 CREATE TABLE 文に DROP TABLE を追加
--add-drop-trigger	各 CREATE TRIGGER 文に DROP TRIGGER を追加
--add-locks	各テーブルのデータ追加前後に LOCK TABLES 文と UNLOCK TABLES 文を追加。データ登録性能を向上
--no-autocommit	各 INSERT 文の前後に SET autocommit = 0 と COMMIT を追加
--no-create-db	CREATE DATABASE 文を出力しない
--no-create-info	CREATE TABLE 文を出力しない
--no-data	データを出力しない
--no-tablespaces	CREATE LOGFILE GROUP 文や CREATE TABLESPACE 文を出力しない
--routines	ストアドプロシージャやストアドルーチンを出力する
--triggers	トリガーを出力する。デフォルトで有効。出力しない場合は --skip-triggers を指定
--master-data	レプリケーションのスレーブで必要となるバイナリログ情報を出力
--hex-blob	バイナリデータ型の内容を 16 進数で出力

-Tまたは--tabオプションを使用すると、データをタブ区切りのファイルとして出力できます。列がタブで区切られ、行が改行で区切られたフォーマットとなっています。またインポート時も明示的に指定しない限りこの形式をデフォルトとしています。さらにフォーマットを変更するオプションを組み合わせて任意のフォーマットで出力することもできます（表2）。なお、このオプションでは、出力先ディレクトリを明示的に指定する必要があります。

表2　-tオプション利用時の出力フォーマットの指定

コマンドラインオプション意味--fields-terminated-by列の区切り文字--fields-enclosed-by列の囲み文字--lines-terminated-by行の区切り文字

表２　-t オプション利用時の出力フォーマットの指定

コマンドラインオプション	意味
--fields-terminated-by	列の区切り文字
--fields-enclosed-by	列の囲み文字
--lines-terminated-by	行の区切り文字

```
$ ./mysqldump <略> world.City

# 出力例
INSERT INTO `City` VALUES (1,'Kabul','AFG','Kabol',
1780000);
INSERT INTO `City` VALUES
(2,'Qandahar','AFG','Qandahar',237500);
INSERT INTO `City` VALUES (3,'Herat','AFG','Herat',
186800);
INSERT INTO `City` VALUES (4,'Mazar-e-
Sharif','AFG','Balkh',127800);
INSERT INTO `City` VALUES (5,'Amsterdam','NLD','Noord-
Holland',731200);
INSERT INTO `City` VALUES (6,'Rotterdam','NLD','Zuid-
Holland',593321);
INSERT INTO `City` VALUES (7,'Haag','NLD','Zuid-
Holland',440900);
INSERT INTO `City` VALUES (8,'Utrecht','NLD','Utrecht',
234323);
...
```

-t オプションなし

```
$ ./mysqldump <略> -T world.City

# 出力例
ID    Name    CountryCode  District  Population
1532  Tokyo   JPN  Tokyo-to  7980230
1533  Jokohama [Yokohama] JPN  Kanagawa  3339594
1534  Osaka   JPN  Osaka   2595674
1535  Nagoya JPN  Aichi   2154376
1536  Sapporo     JPN     Hokkaido  1790886
1537  Kioto   JPN  Kyoto   1461974
1538  Kobe    JPN  Hyogo   1425139
1539  Fukuoka     JPN     Fukuoka   1308379
1540  Kawasaki    JPN     Kanagawa  1217359
1541  Hiroshima   JPN     Hiroshima 1119117
1542  Kitakyushu  JPN     Fukuoka   1016264
1543  Sendai JPN  Miyagi  989975
1544  Chiba   JPN  Chiba   863930
...
```

-t オプションあり

図２　mysqldump コマンドの -t オプション利用イメージ

mysqldumpからのリストア例

mysqldumpからのリストア例を見ていきましょう。

1. マシンやディスクが壊れている場合は復旧する

2. MySQLサーバーが起動している場合は停止する

3. 既存のデータベース領域を削除後、再作成する（必要に応じて事前にバックアップを取得）

```
$   rm -rf /usr/local/mysql/data

$   mkdir /usr/local/mysql/data
```

4. バイナリログの出力を停止（my.cnfからlog-binをコメントアウト）

```
$ ./mysqld --defaults-file=/usr/local/mysql/data/my.cnf ＼
--skip-networking --skip-grant-tables &
```

（この例の場合は、バックアップしておいたmy.cnfを/usr/local/mysql/data配下に配置してから、log-binをコメントアウト。権限テーブルとネットワーク接続を無効化した状態でMySQLサーバーを起動）

5. バックアップファイルに記述されたSQL文を実行する

```
$ ./mysql --default-character-set=utf8 ＼
--socket=/usr/local/mysql/data/mysql.sock ￥
```

6. バイナリログの出力を再開して、正常に再起動する

```
$ ./mysqladmin --user=root --password=root ＼
--socket=/usr/local/mysql/data/mysql.sock shutdown
```

```
$ ./mysqld --defaults-file=/usr/local/mysql/data/my.cnf &
```

7. MySQLサーバー再起動後、mysql_upgradeを実行する

```
$ ./mysql_upgrade   --user=root --password=root ＼
                --socket=/usr/local/mysql/data/mysql.sock
```

この例で、mysql_upgrade実行時に以下のエラーが出力されることがあります（リスト9）。これは、InnoDBストレージエンジンを利用するテーブルの統計情報を管理するmysql.innodb_table_stats、mysql.innodb_index_statsの各テーブルがリストアされるよりも前に、ユーザーが作成したテーブルがリストアされるためです。これらの統計情報管理テーブルはリストア処理中に作成されるため特に問題はなく、無視してかまいません。

リスト9　エラー出力例

```
2016-11-22T12:34:56.418759+09:00 2 [ERROR] InnoDB: Table `mysql`.`innodb_table_stats` not found.

2016-11-22T12:34:56.420733+09:00 2 [ERROR] InnoDB: Fetch of persistent statistics requested for table `<<DB名>>`.`<<テーブル名 >>` but the required system tables mysql.innodb_table_stats and mysql.innodb_index_stats are not present or have unexpected structure. Using transient stats instead.
```

2016-11-22T12:34:56.061321+09:00 0 [Warning] InnoDB: Recalculation of persistent statistics requested for table `<<DB名>>`.`<<テーブル名>>` but the required persistent statistics storage is not present or is corrupted. Using transient stats instead.

8.4.2 MySQLの新しい論理バックアップツール mysqlpump

MySQL 5.7から加わったmysqlpumpは、設計を一新し、mysqldumpにはなかったスキーマごとにバックアップ用の並列処理数を指定する機能や、エクスポート対象のテーブル数とエクスポート済みのテーブル数を基にした進捗の表示機能などを追加しています。--default-parallelismオプションで並列処理数を指定し、かつ--single-transactionオプションを指定した場合にも、データの一貫性を保った状態でオンラインバックアップが可能となっています。

mysqlpumpからMySQLサーバーに5本の接続がある場合に、並列度4（--default-parallelism=4）で実行する例を以下に示します。

1. 全体を制御する接続で「FLUSH TABLES WITH READ LOCK」実行
2. 他の4接続で「SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ」、「START TRANSACTION WITH CONSISTENT SNAPSHOT」を実行し、一貫性のあるバックアップが取得できる状態を担保
3. 全体を制御する接続で「UNLOCK TABLES」を実行しロックを解除
4. 他の4接続でバックアップ取得処理を実行

8.4.3　MySQLサーバーにデータをロードするCUI mysqlimport

mysqldumpで-tまたは--tabオプション付きの論理バックアップを行ったデータや、CSVファイルなどテキストデータをロードするためのツールがmysqlimportです。mysqldumpと同じく、バックアップされたデータの区切り文字などを「8.4.1　MySQL の論理バックアップツール mysqldump」の表1にあるオプションで指定できます。

インポートするデータに見出し行などが含まれる場合、--ignore-linesで指定した行数をスキップすることができます。インポート先のテーブルに主キーやユニークキーなどが定義されており、インポートするデータに該当する列に重複する値が含まれている場合、--ignoreオプションが指定されていれば該当のレコードはインポートされず、--replaceオプションが指定されていれば既存のレコードを破棄して新しいレコードをインポートします。I/O性能が高いストレージを利用している場合は、--use-threadsで指定したスレッド数で並列にデータをインポートすることもできます。

8.5 増分バックアップの方法とツール

MySQLの増分バックアップはバイナリログを利用する方法とMySQL Enterprise Backupの機能を使う方法が選択できます。

8.5.1 バイナリログによる増分バックアップ

バイナリログには、各トランザクションによって行われたデータに対する変更が記録されています。リストアしたデータに対して、該当のバックアップ以降のバイナリログの内容を再実行していくことで、例えば障害発生の直前など任意の時点までデータの状態を復旧することができます。このように特定の時点の状態にデータを復旧することを「ポイントインタイムリカバリ」（PITR）と呼びます。復旧させる時点は、ディスク障害などシステムの障害発生の直前とは限らず、誤ってデータを削除してしまった場合はその直前までや、アプリケーションのバグで不正なデータが混入するようになる前まで戻すようなこともあります。

バイナリログの内容を使ってリカバリする際は、まず該当のバックアップがされた時点の時刻またはバイナリログのポジション以降のmysqlbinlogコマンドでバイナリログの内容をテキスト化し、テキスト化されたデータをmysqlクライアントプログラムに読み込ませてデータの変更を行います。バイナリログのポジションは、mysqldumpであれば--master-dataオプションを有効にすると出力内容にバックアップ時点のバイナリログ名とポジションが記録されます。なお--master-dataオプションは、レプリケーション構成のマスターのデータをダンプしてスレーブにロードしてそのままレプリケーションを開始できるようにする出力があるため、この情報をコメントとして出力できる--master-data=2を使う方がバックアップとして扱うには便利です。

ポイントインタイムリカバリの手順は、リカバリの開始と終了時点をバイナリログのポジションまたは時刻で指定する以外は、mysqldumpからのリストアと同じです。詳細な手順は以下のリファレンスマニュアルを参照してください。

- **URL：MySQL 5.6 リファレンスマニュアル：7.5 バイナリログを使用したポイントインタイム (増分) リカバリ**
  http://dev.mysql.com/doc/refman/5.6/ja/point-in-time-recovery.html

8.5.2 MySQL Enterprise Backupによる増分バックアップ

MySQL Enterprise Backupでは、前回のバックアップ以降にInnoDBのデータファイルに対して行われた変更を自動的に抽出して、以降の変更点をバックアップすることができます。前回のバックアップデータが格納されたディレクトリを明示的に指定する方法と、バックアップ対象のMySQL内にあるmysqlデータベースに記録されたMySQL Enterprise Backup実行履歴を基に前回のバックアップデータを見つける方法があります。

以下に、/opt/mysql/backupディレクトリの曜日別ディレクトリにバックアップデータを置いている場合の例を示します。

リスト10　前回のバックアップデータが格納されたディレクトリを明示的に指定する方法

```
$ mysqlbackup --incremental ＼ > --incremental-base=dir:/opt/mysql/backup/monday ＼ > --incremental-backup-dir=/opt/mysql/backup/tuesday backup
```

リスト11　バックアップ履歴を基に前回分を見つける方法

```
$ mysqlbackup --incremental ＼ > --incremental-base=history:last_backup --with-timestamp ＼ > --incremental-backup-dir=/opt/mysql/backup backup
```

MySQL Enterprise Backupを利用している場合でも、バイナリログによるポイントインタイムリカバリが可能です。詳細な手順は以下のリファレンスマニュアルを参照してください。

●**URL：MySQL Enterprise Backup ユーザーズガイド (バージョン 3.11)**
：4.3　ホットバックアップからのポイントインタイム・リカバリ
https://dev.mysql.com/doc/mysql-enterprise-backup/3.11/ja/advanced.point.html

Column　**アジアのMySQLコミュニティ**

日本だけでなく、アジア各地でもMySQLの利用は急速に広がっており、普及にあわせてコミュニティも拡大しています。MySQLのユーザーグループのいくつかはFacebookのグループ機能を利用しています。

アジアで最大のMySQLユーザーグループは、インドネシアの「MySQL Indonesia」で、22,000名以上のメンバーを誇っています。インドネシアは、アジアでも3番目に人口の多い国で、若年人口も多く、バイクタクシーを使った個人間の貨物輸送サービスなど独自のモバイルサービスも次々と生まれています。MySQLユーザーグループには、動作に関する質問が多数寄せられ、中には学生が学校へ提出する課題の解法を質問しているケースもたびたび見受けられます。また、求人情報が頻繁に投稿されるのもこのグループの特徴です。

Facebookのグループ機能を利用しているのは、韓国と台湾がそれぞれ約2,000名、アゼルバイジャンが1,000名強、カンボジアと香港がそれぞれ100名以上と急速にメンバーが増加中です。比較的新しいベトナムは、30名強となっています。いずれも現地語での

情報交換やイベントの案内、質問と対応がメインとなっています。

中国には、「China MySQL User Group」が存在し、中国各地に支部を持って活動しています。中国の大手SI企業の1つChinasoft（CSS）がメインスポンサーとなり活動を支援しています。メンバー同士のコミュニケーションは、中国でユーザーが非常に多いインスタントメッセンジャーのテンセントQQのグループ機能を利用しています。

インドには複数のグループがあります。バンガロールを拠点とする「MySQL User Camp India」は、機能の解説を直接行う勉強会を年に3、4回開催しています。これはバンガロールオフィスのMySQL開発チームのエンジニアが担当しています。また、ビジネス向けのソーシャルネットワークLinkedIn上のグループである「MySQL India」では、イベントの告知やMySQL利用上のTipsなどが紹介されています。このほか、ハイデラバードやチェンナイ、ブバネーシュワルなどの各都市にもグループがあります。特定の形に縛られずさまざまなグループができあがり形を変えていく様はインドらしいともいえます。

MySQLのコミュニティチームでは全世界のユーザーグループのリストを作成しており、随時情報を更新しています。

- **URL：List of MySQL User Groups**
  https://community.oracle.com/docs/DOC-917215

8.6 演習

このレッスンではMySQLの各種バックアップリカバリ方法についてコマンドとあわせて学習しました。ここでは演習として、論理バックアップとリカバリを行いましょう。

1. mysqldumpですべてのテーブルの論理バックアップを取得する

```
$ ./mysqldump --user=root --master-data=2 --hex-blob --default-character-set=utf8 --all-databases --single-transaction > ../mysql_bkup_dump.sql
```

2. worldデータベースのCityテーブルにデータを追加する

```
# 2016年6月現在で人口が日本で最も少ない市である北海道歌志内市を追加
mysql> INSERT INTO City VALUES(NULL, 'Utashinai', 'JPN', 'Hokkaido', 3590);

# 追加した北海道歌志内市とサンプルデータに含まれていた岡山県津山市が表示されることを確認
mysql> SELECT * FROM City WHERE CountryCode = 'JPN' ORDER BY ID DESC LIMIT 2;
```

3. 最新のバイナリログをコピーしてバックアップとする

```
# tarファイルの展開先が/home/mysql/mysql57の場合
$ cd /home/mysql/mysql57/data
```

```
# バイナリログの一覧を確認
$ more *-bin.index

# バイナリログファイル名の末尾の数字が最大のファイルをコピー
# 以下はバイナリログのファイル名が MySQL-bin.* の場合
$ cp MySQL-bin.000002 ../MySQL-bin.000002_bak
```

4. MySQLサーバーを停止する

ヒント：MySQLサーバーの停止方法はレッスン1のリスト13を参照

5. ディレクトリを破損した障害を再現するため既存のデータベース領域を削除。その後、データディレクトリを再作成し、mysqld --initialize-insecureで初期化処理を再度行う

```
# データディレクトリの削除
$ rm -rf /home/mysql/mysql57/data

# 空のデータディレクトリを作成
$ mkdir /home/mysql/mysql57/data
```

ヒント：データベースの初期化処理はレッスン1のリスト11を参照

6. バイナリログの出力を停止する（my.cnfからlog-binをコメントアウト）

ヒント：ここでバイナリログの出力を停止しておかないと、以降の復旧作業の処理内容がバイナリログに記録され、処理履歴として重複した状態になるため

7. 権限テーブルとネットワーク接続を無効化した状態でMySQLサーバーを起動する

```
$ ./mysqld --defaults-file=/home/mysql/mysql57/my.cnf --skip-networking --skip-grant-tables --basedir=/home/mysql/mysql57 --socket=/home/mysql/mysql57/mysql.sock --server-id=1 &
```

ヒント：ソケットによる通信についてはレッスン3の「3.1.2 接続設定」を参照

8. バックアップファイルに記述されたSQL文を実行する

```
# 権限テーブルとネットワーク接続を無効化しているので、ユーザー無指定およびソケットを明示的に指定している点に注目
$ ./mysql --socket=/home/mysql/mysql57/mysql.sock < ../mysql_bkup_dump.sql

# mysqldumpによる論理バックアップからリストアしたデータのみが存在することを確認
```

```
# 岡山県津山市のデータは表示されるが、バックアップ以降に追加した北海道歌志内市はない
$ ./mysql --socket=/home/mysql/mysql57/mysql.sock world -e "SELECT * FROM City WHERE CountryCode = 'JPN' ORDER BY ID DESC LIMIT 2"

# メタデータを復旧するためにmysql_upgradeコマンドを実行
$ ./mysql_upgrade --socket=/home/mysql/mysql57/mysql.sock --force
```

9. 出力したバックアップファイルmysql_bkup_dump.sqlから、バイナリログの情報を確認する

```
# バイナリログ内で"point-in-time"の出力を含む行の後に、バイナリログの情報が記録されている。
# ファイル名とバックアップされた時点で最後のトランザクションを示すログ内のポジション。
#
# 以下の例ではgrepコマンドの-Aコマンドで"point-in-time"から4行分を表示
# Windows上などでgrepコマンドが使えない場合は、バックアップファイルの30行目付近の出力を確認
$ more ../mysql_bkup_dump.sql |grep -A 4 point-in-time

-- Position to start replication or point-in-time recovery from
--
```

```
-- CHANGE MASTER TO MASTER_LOG_FILE=MySQL-bin.000002', MASTER_LOG_POS=690665
```

10.バックアップしたバイナリログの内容をmysqlbinlogコマンドでテキスト化する

```
# 演習問題2で確認したMASTER_LOG_POS以降の内容が復旧すべきデータを含む処理
# -vオプションでデータを変更するSQL文に該当する情報を表示
$ ./mysqlbinlog -v --disable-log-bin --start-position=690665 ../MySQL-bin.000002_bak

# 復旧に利用すべき処理をmysqlクライアントプログラムに渡してリカバリ
$ ./mysqlbinlog --disable-log-bin --start-position=690665 ../MySQL-bin.000002_bak | ./mysql --socket=/home/mysql/mysql57/mysql.sock

# バイナリログからリカバリしたデータである北海道歌志内市のデータが存在することを確認
$ ./mysql --socket=/home/mysql/mysql57/mysql.sock world -e "SELECT * FROM City WHERE CountryCode = 'JPN' ORDER BY ID DESC LIMIT 2"
```

11.コメントアウトしたlog-binを元に戻し、バイナリログの出力を再開して再起動する

12.演習2で追加したデータを含めてすべてのデータが復旧していることを確認する

```
mysql> SELECT * FROM City WHERE CountryCode = 'JPN' ORDER BY ID DESC LIMIT 2;
```

解説

この演習では「フルバックアップ後にデータの変更があり、差分バックアップを行った直後に障害が発生して復旧が必要」というシナリオを再現しています。

演習1がmysqldumpによるオンラインでのフルバックアップ、演習2でデータの変更を行い、演習3がバイナリログによる差分バックアップに該当します。演習8でフルバックアップからのリストアを行っています。

さらに、演習9で障害直前まで復旧を行うポイントインタイムリカバリの準備として、適用すべきバイナリログの内容を確認しています。このMASTER_LOG_FILEとMASTER_LOG_POSの確認はきわめて重要で、バイナリログファイルの誤ったポジションからリカバリを開始してしまうと、実行済みのトランザクションが重複してしまったり、必要なトランザクションが反映されなかったりしてしまいます。

レッスン

9

レプリケーション 基礎編

障害発生の影響を最小限に抑えるための構成を高可用性構成と呼びます。可用性を向上するための手段は複数ありますが、その中でも簡単な設定で構成ができ、さらにサーバー台数を追加して参照処理性能の拡張性を持てるのがレプリケーションです。このレッスンでは、レプリケーションを中心に学習します。

9.1 MySQL高可用性構成パターン

クラスタリング構成のデータベースでのデータ管理は、それぞれのサーバーに格納する「データミラー型」と、共有ストレージに格納する「ディスク共有型」に大別できます。また、すべてのサーバーがアプリケーションから利用可能かどうかで分類することができます。特徴をまとめます（表1）。

表1 データベースのクラスタリング構成

	データミラー型＆アクティブ／アクティブ型	データミラー型＆アクティブ／スタンバイ型	ディスク共有型＆アクティブ／アクティブ型	ディスク共有型＆アクティブ／スタンバイ型
レスポンス	△〜○	△〜○	○	○
性能拡張性	◎	×	○	×
フェイルオーバー時間	○	△	○	△
構成例	MySQL Cluster	DRBD、SIOS LifeKeeper他	Oracle Real Application Clusters	Oracle Clsuterware、SIOS LifeKeeper、NEC Clusterpro など多数

9.1.1 データミラー型&アクティブ／アクティブ型

MySQL Clusterに代表されるのが、共有ディスクを使わず、かつすべてのノードがアクセス可能な構成です。データベースへの同時アクセス数やデータ量に応じてサーバーを追加し、柔軟に拡張できることが最大のメリットです。

また、サーバー台数が2台に限定されますが、双方向にMySQLのレプリケーションを行う構成も類型といえます。ただし、トランザクションは、それぞれのノードで管理され、非同期でデータが相互に転送されるため、データの変更順が保証されずデータの矛盾が起きてしまう可能性があります。MySQLのレプリケーションでは、この矛盾の検知や解決はできません。ID番号別にデータ変更するサーバーを固定するなど「同一のレコードは同時に別のサーバーで変更しない」アプリケーション側での作り込みが必要です。この課題を解決するのが、MySQL 5.7向けに開発中のグループレプリケーションです。

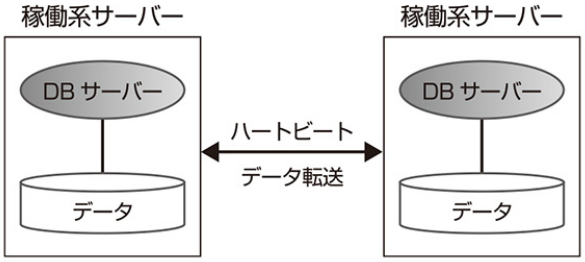


図1　データミラー型&アクティブ／アクティブ型

9.1.2 データミラー型＆アクティブ／スタンバイ型

Distributed Replicated Block Device（DRBD）やSIOS Lifekeeperなどのクラスタリングソフトウェアで実現可能な構成です。これらのようなデバイスレベルの実装では、スタンバイサーバーは、アクティブサーバーからデータがコピーされるパーティションをアンマウントしておく必要があります。また、フェイルオーバー時にはファイルシステムのマウントが行われるので、時間を要することがある点に注意が必要です。また、DRBD単体では、ハートビートパケットによる障害検知や障害時のフェイルオーバーが行えないため、ノードの死活監視を行うCorosyncやクラスタリソースの切り替えなどを行うPacemakerと組み合わせて利用されます。

MySQLのレプリケーションは、MySQLのデータイメージまたはSQL文の複製によって複数サーバー間のデータを同期します。データのコピー先でもMySQLサーバーが起動しており、コピーされたデータを参照することが可能となっています。

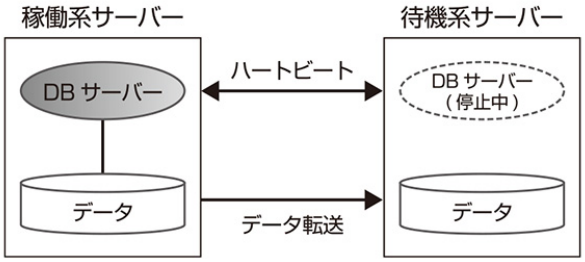


図 2 データミラー型＆アクティブ／スタンバイ型

9.1.3 ディスク共有型＆アクティブ／アクティブ型

Oracle Real Application Clustersに代表される構成です。データを共有ストレージに格納し、すべてのデータベースサーバーがアプリケーションから利用可能になっています。1カ所にデータが集約されているため、テーブルを広くスキャンするような処理や、多くのテーブルを結合する処理などは高速です。一方で、複数のアプリケーションが同時に同じデータや同じデータブロック内のレコードを変更する処理の場合、データの一貫性を保つために行われるデータベースサーバーのキャッシュの最新化処理による性能低下が起きないように、並列実行を避ける工夫が必要です。また、共有ディスクが単一障害点にならないようにストレージデバイス、ファイバチャネルおよびスイッチの多重化を行うため、コストが増大する傾向にあります。

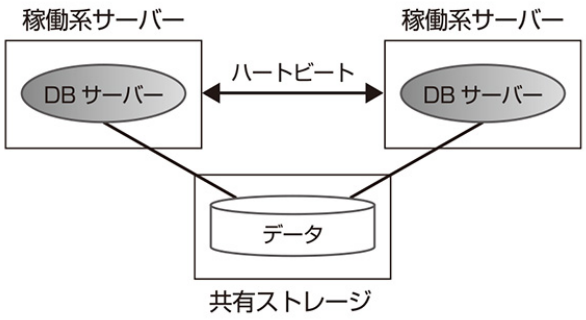


図3 ディスク共有型＆アクティブ／アクティブ型

9.1.4 ディスク共有型＆アクティブ／スタンバイ型

Oracle Clusterwareを始めとする多くのクラスタリングソフトウェアで利用でき、小規模から大規模まで幅広いシステムで利用されている実績のある構成です。類型としては、仮想マシンイメージを共有ディスクに置いて運用する方法もありますが、フェイルオーバー時には、仮想マシン上のOSの起動からとなるため、フェイルオーバー時間がかかる可能性があります。

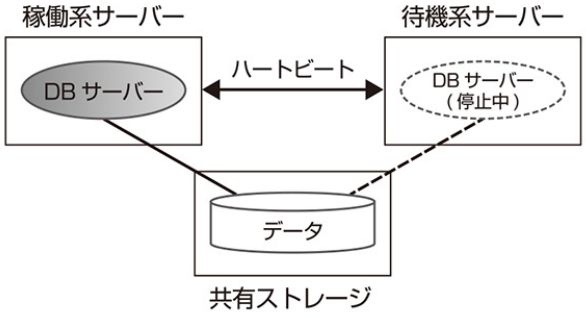


図 4　ディスク共有型＆アクティブ／スタンバイ型

9.2 MySQLレプリケーション

MySQLのレプリケーションは、バイナリログをほかのサーバーに転送し、その内容をデータに反映することで、データのコピーを行います。MySQLのレプリケーションでは、コピー元を「マスター」、コピー先を「スレーブ」と呼びます。レッスン3の「3.1.6　ログ」で解説した通り、バイナリログには各トランザクションによる変更点を記録しています。MySQL 5.7では、ROW（行ベース）形式がデフォルトとなり、バイナリログにはデータ変更後の行イメージが記録されています。また、MySQL 5.6からトランザクションを一意に識別するためのGlobal Transaction Identifier（GTID）が利用できるようになり、レプリケーションの設定や運用が簡素化されています。

MySQLのレプリケーションの主な用途は次の4点です。

- ●参照の負荷分散
- ●耐障害性構成
- ●ディザスタリカバリ構成
- ●バックアップ

Webアプリケーションでは、アクセス全体に対して、データを参照する処理の比率が高いことが多くあります。MySQLのレプリケーションでは、複数のサーバーにデータを転送することで、構成全体の参照処理性能の向上を図ることができます。このようにWebアプリケーションのアクセス特性とMySQLのレプリケーションの特徴の相性がとても良いため、以前からMySQLはWebアプリケーションの

バックエンドとして広く使われてきました。また、分析や帳票作成などの処理を行うための別サーバーとしての利用もよくあるパターンです。

　データの複製が存在することで、耐障害性を持たせた構成として利用されることがあります。レッスン10で解説をしますが、デフォルトの挙動は非同期型となり耐障害性構成には向きません。耐障害性を目的としてMySQLのレプリケーションを利用するためには、準同期型に設定します。

　また、耐障害性構成の1つになりますが、別のデータセンターやクラウドサービスによっては、ドメインやゾーン、可用性セットなどと呼ばれる物理的に隔離された領域、さらには地域を越えてデータを転送する、ディザスタリカバリ構成を取ることができます。この場合には、ネットワークの応答性能を考慮して、非同期でのレプリケーションが設定されることが多くなります。

　スレーブのデータをバックアップとして活用することも考えられます。ただし、レプリケーションではあらゆる変更がコピーされるため、操作ミスなどがあった場合には意図しない変更までコピーされてしまいます。そのためスレーブそのものをバックアップデータとして扱うことは不適切です。スレーブでバックアップを行うことで、バックアップ処理中にもマスターへの負荷がかからない運用は有用と考えられます。考慮すべき点としては、非同期でのレプリケーションの場合にマスターの最新の情報がスレーブのデータに反映されていないことについて、どう運用するかということが挙げられます。

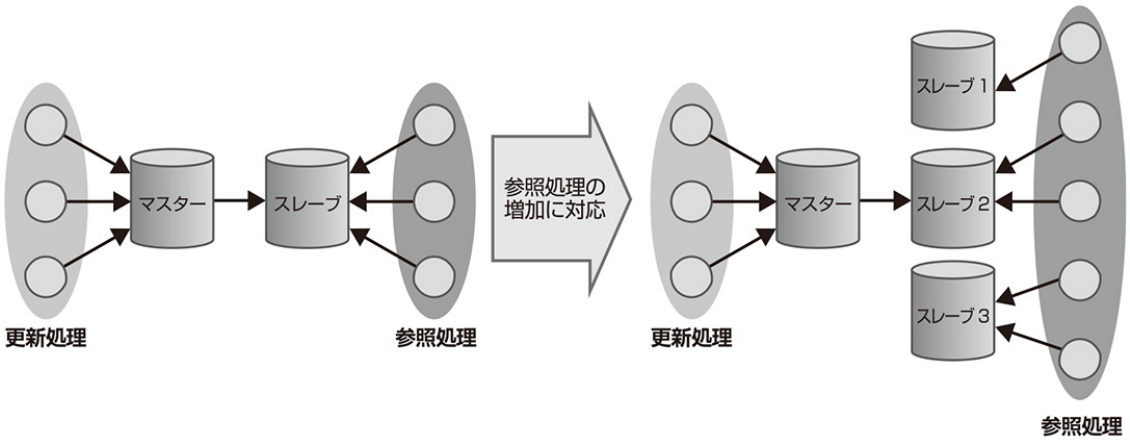


図 5　参照の負荷分散

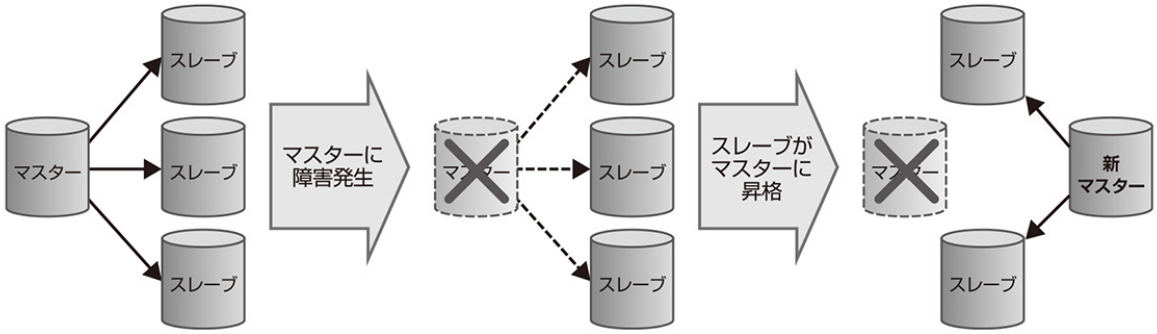


図 6　耐障害性構成

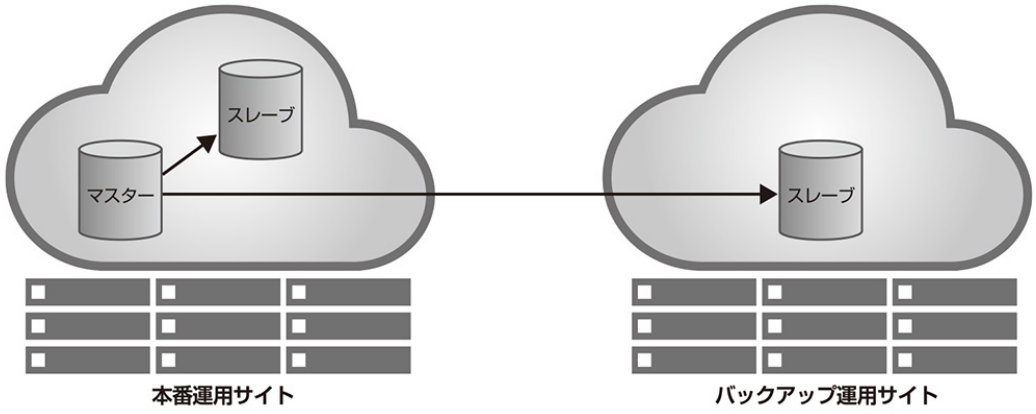


図 7　ディザスタリカバリ構成

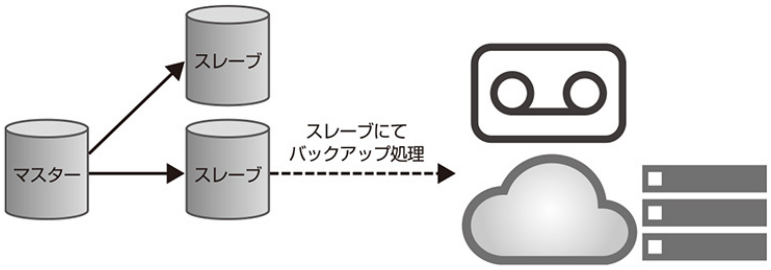


図 8　バックアップ

9.2.1 レプリケーションにおけるMySQLサーバーの各スレッドとファイルの役割

MySQLのレプリケーションでは、マスターおよびスレーブともにMySQLサーバーの複数のスレッドが役割を分担しています。各スレッドの役割は以下の通りです。

表2 MySQL サーバーのレプリケーション関連スレッド

スレッド名	マニュアル等での名称	役割
Binlog ダンプスレッド	Sender スレッド	バイナリログ内容をスレーブに送信
ACK Receiver スレッド	ACK Receiver スレッド	スレーブから ACK を受信
スレーブ I/O スレッド	Receiver スレッド	バイナリ内容を受信しリレーログに記録
スレーブ SQL スレッド	Applier スレッド	リレーログ内容をデータに反映

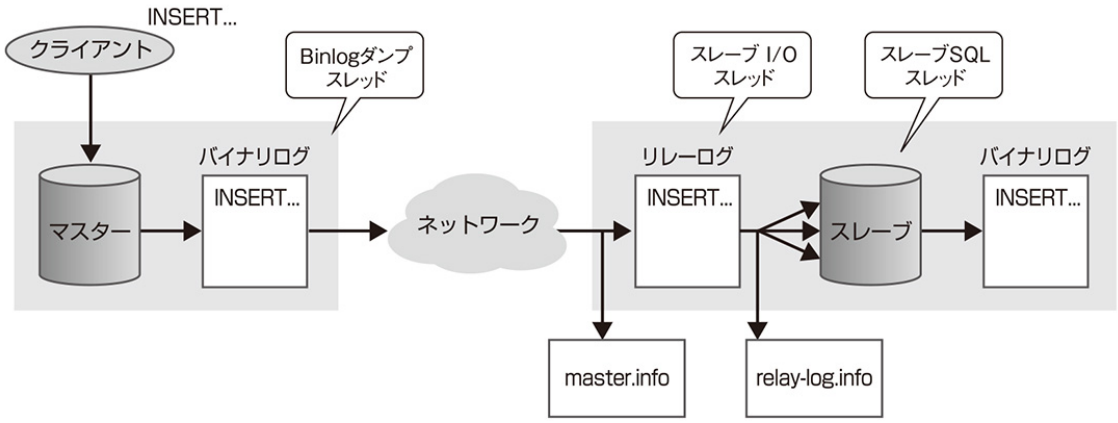


図9 レプリケーションにおける各スレッドとファイル

表3 MySQL サーバーのレプリケーション関連ファイル

ファイル名	役割
リレーログファイル	マスターから受信した変更点を記録したファイル
master.info	マスターへの接続に必要な情報や、読み取りを開始するバイナリログの位置情報（バイナリログファイル名とポジション）が記録されている OS 上のファイル（MySQL 5.6 からテーブル内に格納可能）
relay-log.info	リレーログをどこまで適用したかを記録している OS 上のファイル（MySQL 5.6 からテーブル内に格納可能）

9.3 レプリケーションの基本的なセットアップ方法

ここでは同一のホスト上に新規にレプリケーションを構成する手順を解説します。まずマスターのデータをスレーブにコピーします。以降では、マスターとスレーブでの作業や設定を[マスター]および[スレーブ]で表示しています。

1. マスターを停止した状態でデータディレクトリをスレーブにコピー

2. [スレーブ]データディレクトリのauto.cnfを削除

auto.cnfはサーバーを一意に識別するためのUUIDを格納するためにMySQL 5.6から自動的に作成されるようになった設定ファイルです。削除しない場合はSTART SLAVE時にUUIDの重複としてエラーになります。

リスト1　auto.cnfを削除

```
# MySQLサーバーのシャットダウン
mysql> SHUTDOWN;

# mysqlクライアントプログラム終了
mysql> ＼q

# 初期化済みデータディレクトリのバックアップ
# 自動生成設定ファイルauto.confの削除
```

```
$ cd ../; cp -R data data_slave; cd data_slave; rm auto.cnf; cd ../bin
```

3. [マスター]マスターとなるための設定の追加

マスターになるために最低限必要となる設定は、バイナリログを有効にすることとバイナリログを有効にするためにサーバーIDを設定することの2点、およびGTIDを有効にするための設定です。

リスト2　マスターとなるための設定例

```
[mysqld]
# サーバーIDの設定
server_id = 1

# バイナリログ有効（実運用環境ではログファイル名の指定を推奨）
log-bin

# GTID有効
gtid_mode = on

# GTID利用時に必須となる設定（GTIDの一貫性を担保できないSQLの実行を禁止）
enforce_gtid_consistency = on
```

4. [スレーブ]スレーブとなるための設定

スレーブになるためには、コピー元となるマスターに接続するための情報をあらかじめ設定しておくこともできますが、ここでは後ほどコマンドで設定していきます。同一ホスト上で複数のMySQLサーバーを起動するためには、ほかのMySQLサーバーとデータディレクトリ、TCP／IPポート、Linuxなどの場合はUnixソケットファイルを分けておく必要があります。

リスト3　スレーブとなるための設定例

```
[mysqld]
# サーバーIDの設定
# マスターとは異なる値を指定
server_id = 2

# バイナリログ有効（実運用環境ではログファイル名の指定を推奨）
# スレーブとして稼働する場合は必須ではない
log-bin

# GTID有効
gtid_mode = on

# GTID利用時に必須となる設定（GTIDの一貫性を担保できないSQLの実行を禁止）
enforce_gtid_consistency = on

# 以下はマスターとスレーブを同一OS上で稼働させる場合に必要な構成
# データディレクトリ
```

```
datadir = /home/mysql/mysql57/data-slave
port=3307
socket = /tmp/mysqlslave.sock
```

5. [マスター]レプリケーションスレーブ用のユーザーを設定

レプリケーションを開始する際に、まずスレーブからマスターに接続する必要があります。そのためのユーザーをマスター上に作成し、レプリケーションのスレーブとしての権限（Replication Slave権限）を付与します。

リスト4　レプリケーションスレーブ用のユーザーの作成と権限付与

```
mysql> CREATE USER 'repl'@'localhost' IDENTIFIED BY 'repl';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'localhost';
```

6. [スレーブ]レプリケーションスレーブの設定と開始

スレーブ上で、CHANGE MASTER TOコマンドでスレーブからマスターへの接続情報を設定し、START SLAVEコマンドでレプリケーションを開始します。

リスト5　レプリケーションスレーブの設定と開始

```
# マスターがロカールホスト上で動作している場合
# マスターに作成したユーザーとパスワードで接続
mysql> CHANGE MASTER TO MASTER_HOST='localhost',
MASTER_USER='repl', MASTER_PASSWORD='repl',
```

```
MASTER_AUTO_POSITION=1;

# レプリケーションスレーブの開始
mysql> START SLAVE;
```

7. レプリケーションの動作確認

リスト6　レプリケーションの稼働を確認

```
# それぞれのMySQLサーバー上に存在するデータベースを確認
# [マスター]
mysql> SHOW DATABASES;

# [スレーブ]
mysql> SHOW DATABASES;

# マスター上でデータベースを作成し、スレーブに反映されることを確認
# [マスター]
mysql> CREATE DATABASE rpltest;
# [スレーブ]
mysql> SHOW DATABASES;

# 作成したデータベースにテーブルを作成し、値を格納。スレーブに反映されることを確認
# [マスター]
mysql> use rpltest;
mysql> CREATE TABLE t1 (col1 INT);
mysql> INSERT INTO t1 VALUES (1),(2);
```

```
# [スレーブ]
mysql> SELECT * FROM rpltest.t1;
```

MySQLのレプリケーション方式の詳細や構成パターンはレッスン10で紹介します。

Column　バグデータベース

バグや問題となる挙動があった場合は、再現方法とあわせてバグデータベース（http://bugs.mysql.com/）から報告します。また、機能追加の要望を登録することもできます。ほかの方が報告済みのバグの影響を受けている場合には、「Affects Me」ボタンを押すこともできます。

バグ報告の手順は、最初に報告済みのバグではないかを検索します。同様のバグであっても細かな部分に差がある場合には、既存のバグ報告にコメントを追加します。類似するバグが報告されていない場合は、バグ再現方法、発生した環境の詳細、バグの緊急度をあわせて報告します。

報告されたバグが開発者などの環境で再現すると、ステータスが「Verified」に変更され、以降、開発チームで対処方法が検討されます。この際にバグデータベースの管理者や開発者から、状況の詳細などの追加質問が報告者宛に届く場合がありますので回答してください。

ソフトウェアのバグだけではなく、ドキュメントの誤りもバグとして管理しています。リファレンスマニュアルの記載内容の誤りや不足事項は、バグデータベースのDocumentationカテゴリとして管理しています。なお、日本語版リファレンスマニュアルの翻訳の誤りは「MySQL Server: Japanese Documentation」カテゴリを選択してバグ報告してください。

バグ修正や機能追加のために作成したパッチをMySQL開発チームに取り込んでもらうためには、まず著作権などの知財の取り扱いを定めたOracle Contributor Agreement（OCA）に署名し、送付しておく必要があります。OCAの場合、パッチの開発者自身が著作権を保持し、それを行使する権利は維持しつつ、MySQL開発チームも同様に著作権所有者としてパッチを活用します。

- **URL：The Oracle Contributor Agreement**
  http://www.oracle.com/technetwork/community/oca-486395.html

MySQLの利用者や独自機能を開発しているコミュニティとの連携を円滑にするため、コミュニティ版のMySQLサーバー、アプリケーションからの接続部品であるConnectorや、レッスン5で紹介したGUI製品のMySQL Workbenchなどでは、ソースコードをGitHubにリポジトリを設けて公開しています。機能追加要望やバグ修正のためのパッチは、GitHubリポジトリからプルリクエストの形で提供することも可能になっています。

- **URL：GitHub MySQL公式リポジトリ**

https://github.com/mysql/

バグデータベースなどでパッチを送った際に、OCAの署名送付済みかがわからない場合には、確認の連絡がなされます。mysql.comのユーザーアカウントとOCAのステータスはリンクされるので、バグデータベースのユーザー名にOCAの表示がされているのを見ることもあります。

9.4 演習

このレッスンではMySQLにおける各種の高可用性構成の特徴とレプリケーションについて学習しました。ここでは本文の内容を確認しながら実際にレプリケーション環境を構築し、その動作について確認しましょう。

1. 同一ホスト上で1台のマスターと2台のスレーブの構成を構築する

ヒント：レッスン9の「9.3　レプリケーションの基本的なセットアップ方法」を参照

2. マスターで新しいテーブルを作成し、その内容がスレーブに反映されているかを確認する

ヒント：以降のスレーブにmysqlクライアントシェルプログラムから接続する場合には、TCP/IPポート番号の指定を忘れずに

3. 1台のスレーブを停止した上で、マスターのテーブルに値を追加。稼働中のスレーブのテーブル内容を確認する

4. 停止していたスレーブを再起動し、起動後にテーブルの内容を確認する

解説

1．同一ホストで複数のMySQLサーバーを動かすためには、以下の設定を分けておく必要があります。なお、サーバーIDは複数動かす際に必須ではありませんが、設定ファイルごとに別の値にすることが望ましいほか、レプリケーション構成内では重複しないようにする必要があります。

- **サーバーID：server_id**
- **データディレクトリ：datadir**
- **TCP/IPポート：port**
- **Linuxなどの場合はUnixソケットファイル：socket**

2～4．複数のMySQLサーバーに対してmysqlクライアントシェルプログラムから接続して作業を行うと、どのコンソールがどのサーバーに接続しているのか、わからなくなることがあります。コンソールそのものの背景色や文字色などを変えることも有用ですが、mysqlクライアントシェルプログラムのプロンプトを変えておくのも便利です。プロンプトの変更にはpromptコマンドを使います。例えば「＼u」はユーザー名の表示、「＼h」はMySQLサーバーのホスト名、「＼p」はMySQLサーバーのTCP/IPポート番号（またはソケットファイル名）となります。このほかの利用できる特殊なシーケンスについてはリファレンスマニュアルを参照してください。

- **URL：MySQL 5.7 Reference Manual 5.5.1.2 mysql Commands**

http://dev.mysql.com/doc/refman/5.7/en/mysql-commands.html#idm140162200992352

- **URL：MySQL 5.6 リファレンスマニュアル 4.5.1.2 mysql コマンド**

http://dev.mysql.com/doc/refman/5.6/ja/mysql-commands.html#idm140179253656752

次にpromptコマンドでのプロンプトの表示の変更例を示します。

```
# ユーザー名、ホスト名、TCP/IP番号(またはソケットファイル名)を表示
# コロンやアットマークを間に挟んで表示させることもできる
# 下記の例では最後の大なり記号の後に半角スペースを入れてある
mysql> prompt ＼u@＼h:＼p>
PROMPT set to '＼u@＼h:＼p> '

# シーケンスを使わずに文字列を設定することも可能
root@localhost:3307> prompt スレーブ1>
PROMPT set to 'スレーブ1> '

# promptのみを入力するとデフォルトの表示に戻る
スレーブ1> prompt
Returning to default PROMPT of mysql>
mysql>
```

レッスン

10

レプリケーション応用編

このレッスンでは、レプリケーション方式を比較します。レプリケーション方式として、マスターからスレーブにバイナリログが転送されるタイミングとマスターからスレーブに転送される際のバイナリログの形式を考慮すべきです。あわせて、レプリケーションの構成パターンも学習します。

10.1　タイミング　非同期型＆準同期型

MySQLのレプリケーションでのデータ転送のタイミングは、デフォルトでは「非同期型」となっています。設定を変更することにより、より可用性を高めた「準同期型」を選択できます。

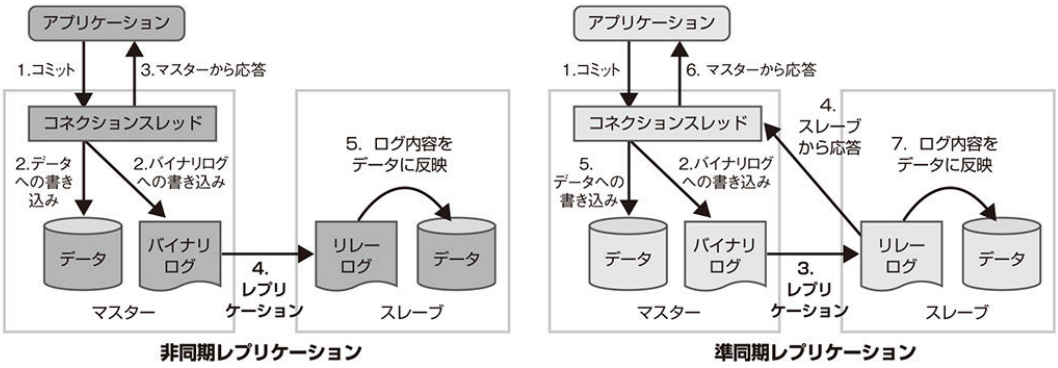
- 非同期型：スレーブへのバイナリログの転送とデータへの反映を待たずに、マスターからアプリケーションに応答を返す
- 準同期型：スレーブへのバイナリログの転送は待つが、コピー内容がデータに反映されるのを待たずに、マスターのサーバーからアプリケーションに応答を返す

性能の観点では、非同期型がデータの変更処理に対する応答時間が短く、準同期型が長くなっています。一方で、データの耐障害性の観点では、バイナリログの内容がスレーブに転送されてからアプリケーションに応答を返す準同期型の方が優れています。

非同期型の場合、アプリケーションは応答したものの、バイナリログの転送が完了していないタイミングでマスターが停止してしまうと、「アプリケーションではコミットが成功したのにデータを失ってしまう」という事象が発生し得ます。

準同期型は、このデメリットを克服できます。ただし、コミットの応答がアプリケーションに返ってきた時点では、コピー先のデータがまだ変更されていない可能性があります。

MySQLのレプリケーションでは、1台のサーバーには準同期型としてデータをコピーし、他のサーバーには非同期型として同時に配信することもできます。



非同期レプリケーションの処理フロー

1. アプリケーションからマスターにコミット
2. ストレージエンジンとバイナリログに書き込み
3. マスターからアプリケーションに応答
4. **バイナリログの内容をスレーブに転送**
5. スレーブでログ内容をデータに反映

準同期レプリケーションの処理フロー

1. アプリケーションからマスターにコミット
2. バイナリログに書き込み
3. **バイナリログの内容をスレーブに転送**
4. スレーブからマスターに応答
5. ストレージエンジンにデータの書き込み
6. マスターからアプリケーションに応答
7. スレーブでログ内容をデータに反映

図1　非同期レプリケーションと準同期レプリケーション

10.1.1 準同期レプリケーションの設定項目

準同期レプリケーションを利用するには、準同期レプリケーションプラグインのインストールと、マスターとスレーブそれぞれに追加の設定が必要です。

リスト1　プラグインのインストール

```
# マスターへのプラグインのインストール INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';

# 各スレーブへのプラグインのインストール INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';
```

準同期レプリケーションプラグインのインストール後、マスターではrpl_semi_sync_master_enabledを1に設定します。この段階では、準同期レプリケーションの準備ができている状態です。

準同期レプリケーションプラグインをインストールしていないスレーブからレプリケーションを開始した場合は、非同期レプリケーションとして動作します。準同期レプリケーションとして動作させるスレーブでは、rpl_semi_sync_slave_enabledを1に設定し、START SLAVEコマンドでレプリケーションを開始します。

1台のマスターに対して、非同期レプリケーションのスレーブと準同期レプリケーションのスレーブを混在させることも可能です。例えば、同じデータセンター内では準同期レプリケーションとして耐障害性を担保し、ディザスタリカバリに利用するスレーブは非同期レプリケーションとすることもできます。

表1　準同期レプリケーションのマスターでの主な設定値

オプション名	概要
rpl_semi_sync_master_enabled	準同期レプリケーションを有効にする
rpl_semi_sync_master_timeout	スレーブが応答しなかった場合のタイムアウト
rpl_semi_sync_master_wait_for_slave_count	応答を待つスレーブの数
rpl_semi_sync_master_wait_no_slave	接続している準同期スレーブが少ない場合の挙動
rpl_semi_sync_master_wait_point	Lossless 型準同期レプリケーションの設定

表2　準同期レプリケーションのスレーブでの設定値

オプション名	概要
rpl_semi_sync_slave_enabled	準同期レプリケーションを有効にする

rpl_semi_sync_master_timeoutは、マスターがどれだけスレーブの応答を待つかの設定です。デフォルト値は10秒（指定はミリ秒単位）です。設定値を超えると、以降は該当のスレーブを非同期レプリケーションとして扱います。

rpl_semi_sync_master_wait_for_slave_countは、MySQL 5.7で加わった設定パラメータで、指定した値のスレーブの応答を待ちます。MySQL 5.7でのデフォルト値も1となり、1台の準同期レプリケーションのスレーブが応答すると、トランザクションのコミットが成功したとしてアプリケーションに応答を返します。この値を2に設定した場合は、2台のスレーブが応答したところでアプリケーションに応答を返します。なお、MySQL 5.6ではこのパラメータは存在しません。

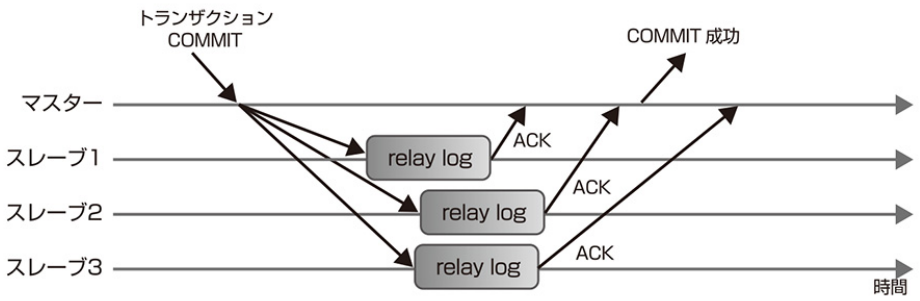


図2　rpl_semi_sync_master_wait_for_slave_count=2 の時の挙動

rpl_semi_sync_master_wait_no_slaveはデフォルトでONとなっています。準同期レプリケーションのスレーブの数がrpl_semi_sync_master_wait_for_slave_countの設定値より少ない場合でも、タイムアウトまで待つ動きとなります。rpl_semi_sync_master_wait_no_slaveをOFFにすると、タイムアウトを待たずに非同期レプリケーションの挙動に変わります。

MySQL 5.6の準同期レプリケーションでは、特定のタイミングでマスターに障害が発生した場合に、データの不整合が起こり得る課題が残っていました。準同期レプリケーションの流れの中でストレージエンジンにトランザクションが記録された時点（図1で示した準同期レプリケーションの2）で、別のクライアントからはコミットの内容を参照することができます。この後、スレーブに転送される前にマスターに障害が発生すると、ほかのクライアントから見えていたコミット済みのトランザクションがスレーブには存在せず、データの整合性が取れていないように見えてしまいます。この問題を解決したのが、MySQL 5.7の「Lossless準同期レプリケーション」です。Lossless準同期レプリケーションは、MySQL 5.7から新たに加わったパラメータrpl_semi_sync_master_wait_pointのデフォルト値AFTER_SYNCを利用することで実現できます。この値をAFTER_COMMITに変更した場合は、MySQL 5.6までの挙動と同じになります。

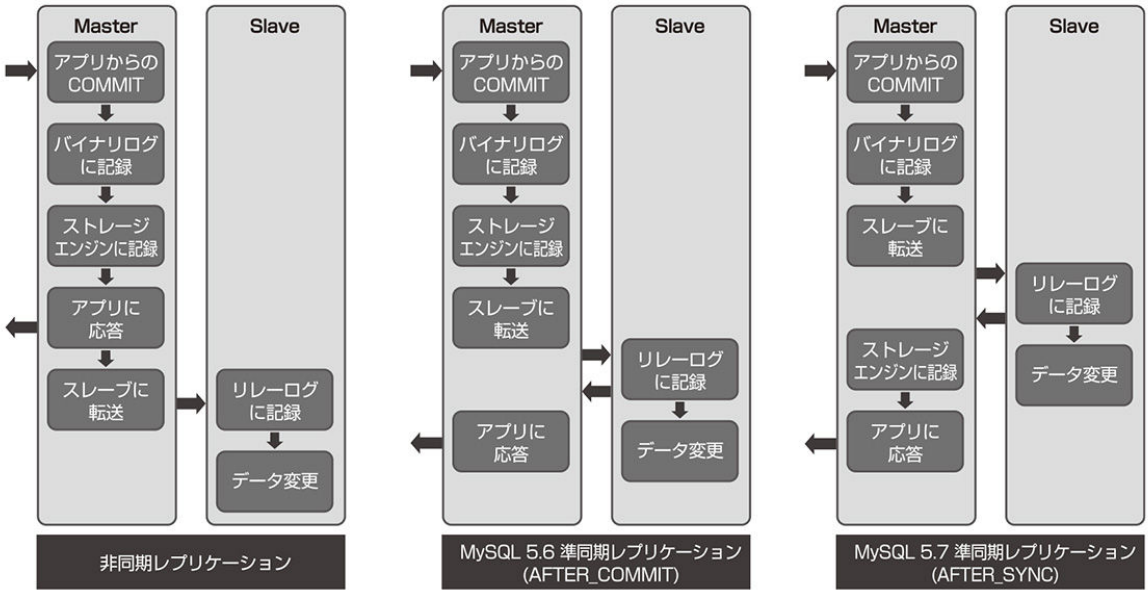


図 3　MySQL 5.7 の Lossless 準同期レプリケーション

10.2 バイナリログの形式　SQL文転送型＆行イメージ転送型

MySQL 5.7より、マスターからスレーブに転送される際のバイナリログは、変更後の行イメージを記録した「行ベース」（Row Based）がデフォルトとなりました。設定パラメータは、binlog_format=ROWがデフォルトです。MySQL 5.6までのデフォルト値は、実行されたSQL文を記録する「文ベース」（Statement Based）となっていました。

MySQL 5.6から導入されたサーバーオプションbinlog_rows_query_log_eventsを有効にすると、該当するトランザクションでのSQL文もバイナリログに記録されます。mysqlbinlogコマンドに-vv（小文字のvを2個続ける）オプションを付けて実行すると、記録されたSQL文をテキスト化できます。

行イメージをバイナリログに記録して転送されるため、「文ベース」と比較してサーバー間で渡されるデータ量が大きくなるケースがほとんどです。そこで、MySQL 5.6以降で利用可能なサーバーオプションbinlog_row_imageをminimalに設定すると、主キーと変更された列のみをバイナリログに記録するため、「行ベース」レプリケーションで転送されるデータ量を抑えることができます。このminimalの利用時は、レプリケーション対象の列の定義が順番を含めてすべて同じであることが必須です。「行ベース」のレプリケーションではスレーブでSQL文が実行されないのでスレーブ側でのトリガーが実行されません。binlog_row_imageについては、以下を参考にしてください。

- **URL：MySQL 5.6 リファレンスマニュアル：17.1.4.4バイナリログのオプションと変数　binlog_row_image**
  http://dev.mysql.com/doc/refman/5.6/ja/replication-options-binary-log.html#sysvar_binlog_row_image

「文ベース」のレプリケーションで注意すべき点は、SQL文で利用する関数によってはデータの不整合が起こり得る点です。例えば、サーバー固有のIDであるUUIDを取得するUUID()関数やシステム日付を取得するSYSDATE()関数などは、マスターとスレーブで実行されるタイミングによって返る値が異なるため、これらの関数を利用して値を変更する場合などに問題となります。問題となり得る関数やSQL文のリストは以下のURLを参考にしてください。これらの関数などを利用する場合には、MySQL 5.6までのデフォルトの「行ベース（Row Based Replication）」または両方の方式のうち最適なものを自動的に選択する「MIXED」に設定します。

- **URL：MySQL 5.6 リファレンスマニュアル：17.1.2.3バイナリロギングでの安全および安全でないステートメントの判断**
  http://dev.mysql.com/doc/refman/5.6/ja/replication-rbr-safe-unsafe.html

10.3 GTIDモード

MySQL 5.6から加わった機能の1つに「Global Transaction IDentifier」（GTID）があります。GTIDは、トランザクションを一意に識別するためのIDで、レプリケーションでも活用できます。GTIDはバイナリログに記録され、一度マスターでトランザクションに割り当てられたIDは、スレーブに渡されても値が変わりません。以下がGTIDを有効にした際の利点です。

- ●複数台のレプリケーション環境でも容易にGTIDを確認することで、トランザクションの追跡／比較が可能
- ●レプリケーション開始時にポジションを自動認識（master_auto_position=1）
- ●マスター障害時にスレーブを昇格させる際に最新のスレーブを自動認識

10.3.1 GTIDを設定しているバイナリログ内の出力例

GTIDを設定しているバイナリログの出力例を以下に示します。コロンより前の文字列がサーバーを一意に識別するUUID、後ろがそのサーバーでのトランザクション番号です。

リスト2　GTIDを設定しているバイナリログの出力例

```
SET @@SESSION.GTID_NEXT= '8560c2ac-e1dc-11e4-88ff-0800275399c1:6170'/*!*/;
```

GTIDを利用していない環境では、スレーブでのレプリケーション開始時に、マスターのバイナリログファイル名とバイナリログ内のどのポジションからレプリケーションを開始するかを明示的に指定する必要がありましたが、GTIDが有効の場合はスレーブが持っていないGTIDから自動的にレプリケーションを開始すべきトランザクションを見つけることができるようになりました。

また、MySQL 5.7では、GTIDを利用していない環境から新たにGTIDを有効にするためには、動的に設定値の変更が可能となりました。MySQL 5.6ではこの仕組みがないため、レプリケーション構成全体を一度停止する必要がありました。

- **URL : MySQL 5.7 Reference Manual : 18.1.5.2 Enabling GTID Transactions Online**
  http://dev.mysql.com/doc/refman/5.7/en/replication-mode-change-online-enable-gtids.html

10.4 MySQLのレプリケーション構成パターン

図4は、レプリケーション構成パターンです。MySQLのレプリケーションで広く利用されている構成は、1台のマスターに複数のスレーブを組み合わせる「1:n型」となっています。ほかにもMySQL 5.7では、複数のマスターから1台のスレーブにデータを集約するためのマルチソース型が追加されています。また原稿執筆時点では開発中ですが、レプリケーション環境内のすべてのサーバーをマスターとして運用するグループレプリケーションもあります。

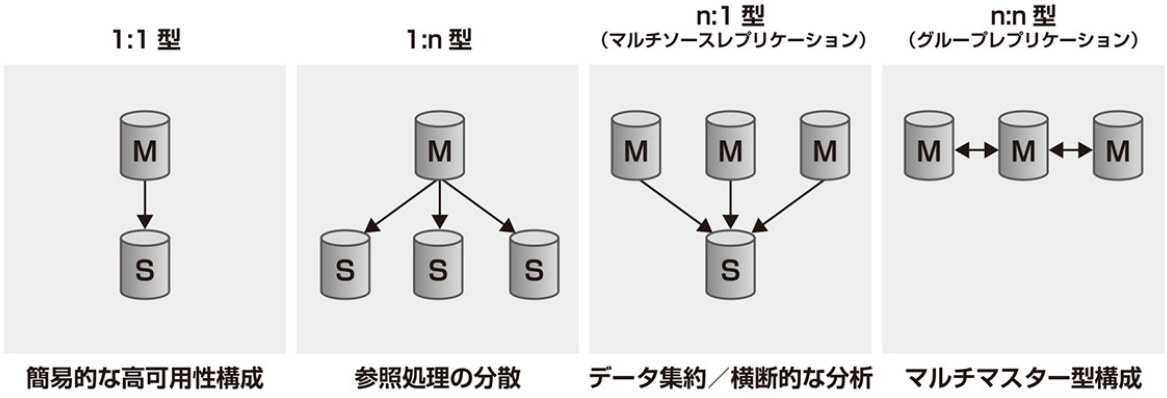


図4　レプリケーション構成パターン

10.4.1 1:1型と1:n型

2000年にリリースされたMySQL 3.23でのレプリケーション実装当初から広く使われている構成が、マスターとスレーブを1台ずつ利用する「1:1型」や1台のマスターに複数のスレーブを組み合わせる「1:n型」です。

1:1型は、簡易的な高可用性構成として利用できます。レッスン9の演習で構築した構成もこの1:1型となります。

1:n型では、参照処理を複数のスレーブに分散させることができるため、アクセスのうち参照比率が圧倒的に多いWebサイトのバックエンドデータベースとして広く活用されてきました。

10.4.2 n:1型（マルチソースレプリケーション）

MySQL 5.7から加わった新しい構成が、複数のソース（マスター）から1台のサーバー（スレーブ）にレプリケーションする「マルチソースレプリケーション」です。この構成では、1台のサーバーにデータを集約する構成となり、複数のMySQLサーバーのデータのバックアップを1台で実行することや、異なるデータを管理するMySQLサーバー群のデータをまとめて分析するなどの利用が想定できます。

スレーブからマスターに接続する際にマスターごとにチャネルを用意し、そのチャネルを通じて通信します。各チャネルに個別のスレッドやリレーログが用意され、複数のチャネルは個別に稼働／停止可能です。

リスト3　マルチソースレプリケーションのためのチャネルの利用

```
# マスターを指定するCHANGE MASTER TO実行時にFOR CHANNELでチャネルを指定
mysql> CHANGE MASTER TO
    MASTER_HOST='master1', MASTER_USER='rpl',
    MASTER_PORT=3451, MASTER_PASSWORD='',
    MASTER_AUTO_POSITION = 1 FOR CHANNEL 'master-1';

# 指定したチャネルの特定のスレッドレプリケーションを開始／停止することも可能
# 下記 thread_types には SQL_THREAD, IO_THREAD を指定可能
mysql> START SLAVE thread_types FOR CHANNEL 'master-1';

mysql> STOP SLAVE thread_types FOR CHANNEL 'master-1';
```

```
# 指定したチャネルだけレプリケーション設定をリセットすることも可能
mysql> RESET SLAVE thread_types FOR CHANNEL 'master-1';
```

10.4.3 n:n型（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、レプリケーション構成内のすべてのMySQLサーバーがマスターとなる「n:n型」またはマルチマスター型の構成となります。グループレプリケーションでは、グループメンバーの自動管理と障害検知が可能で、1台のサーバーに障害が起きても、レプリケーション構成から切り離されるだけで、フェイルオーバー処理が不要です。共有ディスクなしで単一障害点のないクラスタリング構成が可能となります。またInnoDBに対応しており、特殊なハードウェアが不要な構成です。

グループレプリケーションは、MySQLサーバーへのプラグインとして開発が進められており、MySQLサーバー本体のメジャーバージョンアップのタイミングを待たずにリリースが行われることが検討されています。2016年10月の時点では開発中となっており、今後のリリースが期待される構成です。

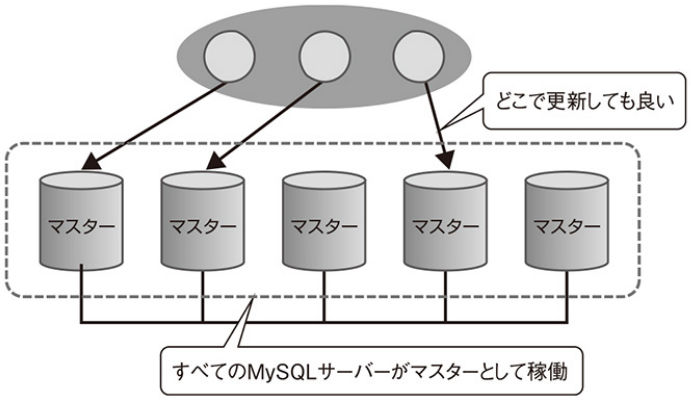


図5 MySQLのグループレプリケーション

10.4.4 MySQL InnoDB Cluster

マルチマスター型レプリケーションのグループレプリケーションをベースとして、MySQLサーバーの高可用性構成を実現するソフトウェアパッケージをまとめたのが、MySQL InnoDB Clusterです。MySQL InnoDB Clusterは、グループレプリケーションの構成をアプリケーションから利用しやすく、また運用をシンプルにするための機能を統合したソリューションとして提供することを目標としています。

- マルチマスター型レプリケーション：MySQLグループレプリケーション
- MySQLプロトコル用のソフトウェアルーター：MySQL Router
- 新しいクライアントシェルプログラム：MySQL X Shell

2016年10月時点では、MySQL Router 2.1およびMySQL X ShellのMySQL InnoDB Clusterの管理APIは、Labsリリース（実験室版）となっています。

アプリケーションからはMySQL Routerに接続し、MySQL Routerがグループレプリケーション構成内の各MySQLサーバーへの処理の転送と負荷分散などを担当します。また、MySQLサーバーに障害があった場合やサーバーが追加された場合など、構成に変更があると新しい構成にあわせた通信を行います。新しいクライアントシェルプログラムであるMySQL X Shellには、グループレプリケーションの新規構築や構成変更、監視を行うための管理APIが実装されています。

MySQL InnoDB Cluster Step 2と呼ばれるフェーズでは、これらの製品の統合とグループレプリケーションのサポートが行われます。

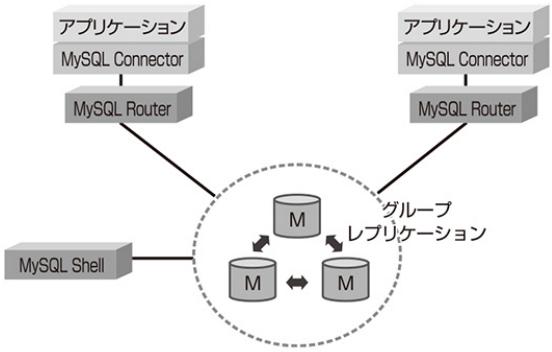


図 6 MySQL InnoDB Cluster - Step 2

次のStep 3では、グループレプリケーションにスレーブを追加して、参照性能の向上を図った構成のサポートが想定されています。

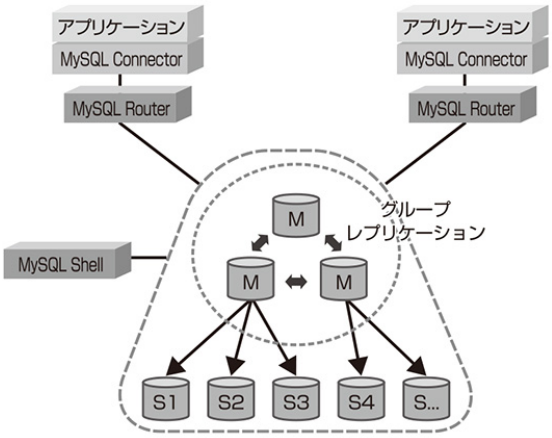


図 7 MySQL InnoDB Cluster - Step 3

さらにStep 4では、グループレプリケーションとスレーブの組み合わせをレプリカセットとして、複数のレプリカセットによるシャーディング構成のサポートを予定しています。

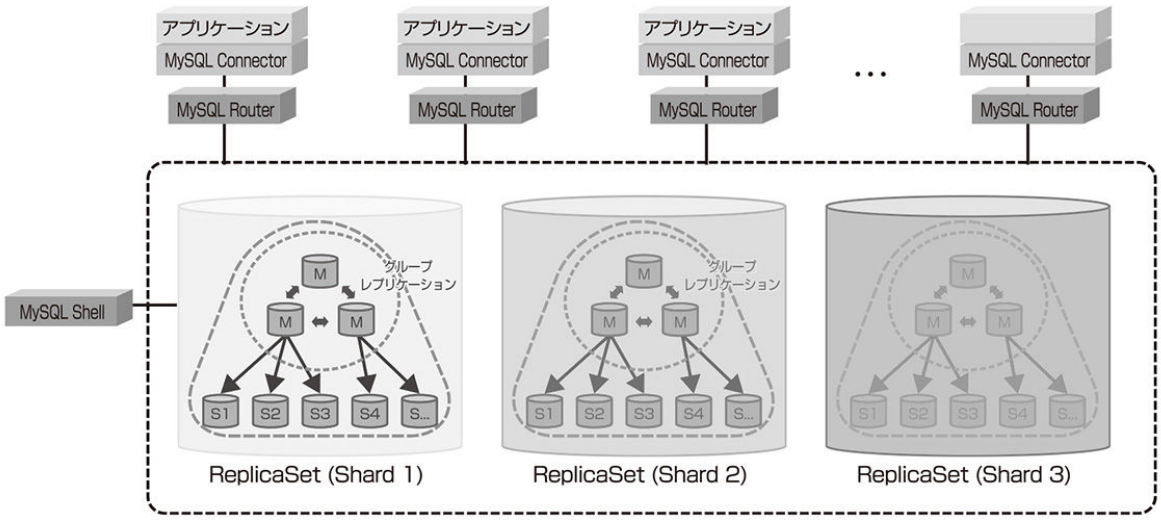


図 8 MySQL InnoDB Cluster - Step 4

Column　MySQL Cluster

MySQL Clusterは、共有ディスクを必要としない分散型のRDBMSクラスタで、すべてのノードがアクティブノードとして稼働します。もともとはスウェーデンの通信機器ベンダ「エリクソン」で携帯通信網の加入者データベース向けに開発されたEricsson Network DataBase（NDB）と呼ばれていた技術で、旧MySQL社がエンジニアや知財を取り込んでMySQLサーバーと組み合わせ、現在のMySQL Clusterになっています。このような経緯から、MySQL Clusterは単一障害点がないことによるミッションクリティカルシステムでも利用できる高い可用性を持っています。テーブル定義やパッチ適用、OSを含めたバージョンアップ等もクラスタ全体としては無停止で運用可能です。また、インメモリーデータベースとしても動作できることによるきわめて高い応答性能、オンライ

ンでノードを追加できることによる高い性能拡張性を兼ね備えています。

MySQL Clusterは分散型のアーキテクチャのため、参照性能の拡張性だけではなく、更新処理性能が求められる大規模システムでの運用実績が多数あります。大規模なオンラインゲームのバックエンドデータベース、国内外の主要な通信機器ベンダやPayPalの不正取引検出機能のほか、軍事防衛の領域でも導入されています。

MySQL Clusterの中では、MySQLサーバーのプロセスは「SQLノード」と呼ばれ、アプリケーションからの接続を受け付けてユーザー認証を行い、SQL文の構文解析や実行計画の最適化を行います。各テーブルはNDBストレージエンジンを利用し、データはデータノードと呼ばれる別のサーバーに格納するのが特徴です。データを追加や更新する際は複数のデータノードに記録されるため、1台のデータノードに障害が発生しても運用を継続可能です。

MySQL Clusterは、各種のNoSQL APIを備えており、トランザクション対応NoSQLとしても活用できます。分散キャッシュのmemcachedにMySQL開発チームが作成したプラグインを追加して、MySQL Clusterに対してキーバリュー型のアクセスを可能にしています。また、各種NoSQL APIからのアクセスもSQLでのアクセスもトランザクショナルに処理され、アプリケーション要件に応じて同一のデータに複数のアクセス手段を提供しています。さらに、NoSQL APIから行った変更点を外部のMySQLサーバーにレプリケーションが可能で、トランザクション対応キーバリューデー

タストアとしてのMySQL ClusterからRDBMSとしてのMySQLサーバーへのレプリケーションも可能です。

10.5 演習

このレッスンではレプリケーションの各種の方式や構成パターンについて学習しました。ここではバイナリログの内容を参照し、バイナリログの形式について確認しましょう。

1. 最新のバイナリログをSHOW BINARY LOGSで確認

```
# バイナリログの一覧を表示
mysql> SHOW BINARY LOGS;

# バイナリログのローテーション（切り替え）を行い、再度一覧を確認
mysql> FLUSH BINARY LOGS;
mysql> SHOW BINARY LOGS;

# バイナリログ内の情報を表示 末尾の数字が最大のものが最新のバイナリログファイル
mysql> SHOW BINLOG EVENTS IN 'MySQL-bin.000004';
```

2. バイナリログの現在のフォーマットをSHOW VARIABLESで確認

```
mysql> SHOW VARIABLES LIKE 'binlog_format';
```

ヒント：バイナリログのフォーマットについてはレッスン10「10.2　バイナリログの形式 SQL文転送型＆行イメージ転送型」を参照

3. Cityテーブルにデータを追加し、SHOW BINARY LOGSで最新のバイナリログの内容を確認

```
# 2016年6月現在で人口が日本で最も少ない村である東京都青ヶ島村を追加
mysql> INSERT INTO City VALUES(NULL, 'Aogashimamura', 'JPN', 'Tokyo', 201);

# 追加した東京都青ヶ島村が表示されることを確認
mysql> SELECT * FROM City WHERE CountryCode = 'JPN' ORDER BY ID DESC LIMIT 3;

# バイナリログ内の情報を表示
# バイナリログのフォーマットがROWの場合はテーブル名の情報は表示されるがSQL文そのものは表示されない
mysql> SHOW BINLOG EVENTS IN 'MySQL-bin.000004';
```

4. バイナリログのフォーマットをセッション単位で変更

```
# バイナリログのフォーマットを文ベースに切り替え
mysql> SET SESSION binlog_format = 'STATEMENT';
```

5. Cityテーブルにデータを追加し、SHOW BINARY LOGSで最新のバイナリログの内容を確認

```
# 2016年6月現在で人口が日本で最も少ない町である山梨県早川町を追加

```

```
mysql> INSERT INTO City VALUES(NULL, 'Hayakawamachi', 'JPN', 'Yamanashi', 1237);

# 追加した山梨県早川町が表示されることを確認
mysql> SELECT * FROM City WHERE CountryCode = 'JPN' ORDER BY ID DESC LIMIT 4;

# バイナリログ内の情報を表示
# バイナリログのフォーマットがSTATEMENTの場合はSQL文が表示されることを確認
mysql> SHOW BINLOG EVENTS IN 'MySQL-bin.000004';
```

解説

この演習では、MySQLのレプリケーションで重要となるバイナリログのコマンドと内容の確認を行いました。

MySQL 5.7では、デフォルトのフォーマット（オプション名binlog_format）が行ベース（ROW）となっています。フォーマットの違いによるそれぞれのメリットとデメリットについては、「10.2 バイナリログの形式 SQL 文転送型&行イメージ転送型」を確認してください。

MySQLグループレプリケーションは、行ベース（ROW）のバイナリログのみをサポートしているほか、MySQL Cluster構成内でも行ベースのみがサポートされているため、行ベースのバイナリログの特徴や制限事項を確認しておいてください。

レッスン

11

セキュリティ Part1

このレッスンでは、セキュリティの強化機能について学習します。インストールやユーザー関連のセキュリティ対策について解説します。

11.1 データベース・セキュリティ概論

データベースのセキュリティを強化することは、機密データを保護したり、システム障害を防いだりするために非常に重要です。

Symantec社が2016年4月に発表した「2016 Internet Security Threat Report」によると、以下の事象が報告されています。

- 2015年の調査では、78%のWebサイトに脆弱性があり、さらにそのうち15%は深刻な脆弱性を抱えていた
- 2015年には4億レコード以上の個人情報が流出し、2014年と比べて23%増加した
- 2015年に1,000万レコード以上の被害にあった不正アクセス事件は9件

このようにセキュリティ対策が甘いシステムが多いことや、それにより多くの被害が発生していることがわかります。被害にあわないためにも、データベースにおける脆弱性を理解し、それに応じた対策を取ることが重要です。

- **URL：2016 Internet Security Threat Report**
  https://www.symantec.com/security-center/threat-report

例えば、表1のようなデータベースの脆弱性とその対策が想定されます。

表 1　データベースの脆弱性とその対策

脆弱性の原因	対策例
設定の不備	デフォルト設定からの変更
過剰な権限の付与	適切な権限付与
貧弱な認証	複雑なパスワードの強制
認証情報の不十分な管理	mysql_config_editor 利用、外部認証の使用
貧弱な監査	監査ログ取得
暗号化の不足	データ／バックアップの暗号化、ネットワーク暗号化
監視の不足	ユーザー＆オブジェクト監視、監査ログの監視
アプリケーションの脆弱性	データベースファイアウォール

また、表2のような悪意のある攻撃への対策も必要です。

表 2　悪意のある攻撃とその対策

攻撃の種類	対策例
SQL インジェクション	データベースファイアウォール、入力バリデーション
バッファオーバーフロー	ソフトウェアの定期的な更新、データベースファイアウォール、入力バリデーション
ブルートフォースアタック（総当たり攻撃）	設定回数を超えた回数ログインを試みたアカウントをロック
ネットワーク傍受	すべての接続とデータ転送に SSL/TLS を使用
マルウェア	強固なアクセスコントロール、接続元 IP アドレスの制限、デフォルト設定の変更
情報漏洩	データやネットワークの暗号化、強固なアクセス制御、監査ログ取得による内部犯行の抑止
DoS（サービス拒否）攻撃	各種のリソース利用制限（最大接続数、セッション数、タイムアウトなど）
不正な改変	認証の強化、監査、監視、バックアップ

このような攻撃からデータやシステムを保護するために、適切なセキュリティ設定を実施する必要があります。

続いて、MySQLサーバーのセキュリティを強化するために留意すべき点について、具体的に解説します。

11.2 MySQLサーバーのセキュリティ対策

MySQLサーバーのセキュリティ対策には以下の対応があります。ここでは、インストール関連のセキュリティ対策とユーザー関連のセキュリティ対策について解説します。

インストール関連のセキュリティ対策

- ●MySQLのバイナリは常に最新版を利用する
- ●ファイルシステム上の権限を適切に設定する
- ●secure_file_privを設定して、FILE権限を持つユーザーがファイルの入出力に使えるディレクトリを制限する
- ●初期データベースはmysqld --initializeで作成する

ユーザー関連のセキュリティ対策

- ●MySQLデータベースのユーザーは、最低限のホストからのアクセスを可能にし、最低限の権限のみを付与する
- ●ユーザーが利用するリソースを制限する
- ●パスワード検証プラグインを有効にしてパスワードポリシーを設定する
- ●mysql_config_editorを使って認証情報を管理する
- ●外部認証を使用しユーザー管理を一元化する

ネットワーク関連のセキュリティ対策

- ●必要最低限のネットワークインターフェースを使用する
- ●ネットワーク通信をSSL/TLSにより暗号化する

暗号化によるセキュリティ対策

- ●バックアップファイルを暗号化する
- ●透過的データ暗号化により、データファイルを保護する

●機密データは暗号化してデータベースに格納する

その他のセキュリティ対策（監査、データベース・ファイアウォール）

●監査ログを取得してデータの改ざんなどの不正操作を監視する

●監査ログを取得することで抑止力により内部犯行を防ぐ

●データベース・ファイアウォールにより想定しないアクセスをブロックする

11.3 インストール関連のセキュリティ対策

インストールに関連するセキュリティ対策を解説します。

11.3.1 MySQLのバイナリは常に最新版を利用する

MySQLに脆弱性が発見された場合、修正はマイナーバージョンアップによって提供されます。そのため、MySQLのバイナリは常に最新版を利用することが、セキュリティ上の観点からも望ましいです。

レッスン1で解説した通り、MySQLはLinux環境用にYumやaptのリポジトリを用意したり、Windows環境用にインストーラを用意したりしていますので、これらを用いることで、容易にバイナリを最新版に更新できます。

11.3.2 ファイルシステム上の権限を適切に設定する

ファイルシステムのセキュリティ対策として、以下の通り設定することが望ましいです。

- MySQL実行バイナリの所有者はOSのrootユーザーにする
- データディレクトリ、ログディレクトリ、ファイルの入出力に使用するディレクトリの所有者はOSの一般ユーザーにする（例：mysqlユーザー）
- 一般ユーザーはログイン不可にしておき、MySQLサーバーの起動／停止のみに使用する

例えば、リスト1は、Linux環境におけるtarファイルを使用したMySQLのインストール例です。一般ユーザーとしてログイン不可であるmysqlユーザーを作成し、MySQL実行バイナリの所有者はroot、データディレクトリ兼ログディレクトリ（/usr/local/mysql/data）、ファイルの入出力に使用するディレクトリ（/usr/local/mysql/mysql-files）の所有者はmysqlユーザーに設定しています。

リスト1　MySQLのインストール例（Linux環境でtarファイル使用）

```
$ groupadd mysql
$ useradd -r -g mysql -s /bin/false mysql
$ cd /usr/local
$ tar zxvf /path/to/mysql-VERSION-OS.tar.gz
$ ln -s full-path-to-mysql-VERSION-OS mysql
$ cd mysql
$ mkdir mysql-files
$ chmod 770 mysql-files
```

```
$ chown -R mysql .
$ chgrp -R mysql .
$ bin/mysqld --initialize --user=mysql
$ bin/mysql_ssl_rsa_setup
$ chown -R root .
$ chown -R mysql data mysql-files
```

　mysql-filesというディレクトリは、別途secure_file_privを設定することを想定したディレクトリです。この設定について解説します。

11.3.3 secure_file_privを明示的に設定する

MySQLサーバーでは、FILE権限を持つユーザーがLOAD_FILE関数やLOAD DATA文を使用することで、ファイルシステム上のファイルを読み込んで処理を実行できます。また、SELECT...INTO OUTFILE文を使用することで、ファイルシステム上のファイルに書き込みができます。

この時、システム変数secure_file_privに何も設定がない場合は、MySQLサーバーを起動している一般ユーザーがアクセス可能なすべてのディレクトリについて、ファイルの入出力が可能になります。これは、セキュリティ上望ましくない状態ですから、secure_file_privに任意のディレクトリを設定するか、ファイルの入出力自体が不要である場合は「NULL」を設定して、ファイルの入出力をすべて禁止することが推奨されます。

secure_file_privに何も設定がない場合、例えばFILE権限を持つユーザーで以下のコマンドを実行すると、/etc/passwdファイルを参照でき、MySQLサーバーが稼働しているOS上のユーザー一覧を確認できてしまいます。

リスト2　LOAD_FILE関数によってOS上のファイルを参照可能

```
mysql> SELECT LOAD_FILE('/etc/passwd')＼G
*************************** 1. row ***************************
LOAD_FILE('/etc/passwd'): root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

```
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
<後略>
```

　このような状態は望ましくないため、secure_file_privを明示的に設定することが推奨されます。

11.3.4 初期データベースはmysqld --initializeで作成する

MySQL 5.7では、初期データベースの作成方法が変更になり、mysqld --initializeコマンドで作成することが推奨されています。

mysqld --initializeコマンドで初期データベースを作成した場合、従来の方法よりもセキュリティが強化されています。具体的には、以下の特徴があります。

- **testデータベースは作成されない**
  ⇒従来は、初期データベースにtestデータベースが作成されていた
- **初期ユーザーは、'root'@'localhost'と'mysql.sys'@'localhost'のみ**
  ⇒従来は、匿名ユーザーやリモートから接続可能なrootユーザーも作成されていた
- **rootユーザーの初期パスワードにはランダムなパスワードが設定され、初回ログイン後パスワードを変更するまでは何も操作ができない**
  ⇒従来は、rootユーザーにパスワードが設定されていなかった

なお、初期ユーザー'mysql.sys'@'localhost'はMySQL 5.7で追加されたユーザーで、sysスキーマ用のユーザーです。sysスキーマについてはレッスン14で解説します。

rootユーザーに自動的に設定されたランダムなパスワードは、リスト3のようにエラーログに出力されるため、初期データベース作成後は、エラーログからパスワードを確認し、そのパスワードを使用してログインします。

リスト3　エラーログに出力される初期パスワードの例

```
2016-06-24T22:35:33.542937+09:00 1 [Note] A temporary password is generated for root@localhost: N9oGt?fow-p3
```

その後、ALTER USER文を用いてパスワードを変更します。11.4.3で取り上げるパスワード検証プラグインが有効になっている場合は、変更後のパスワードがポリシーを満たす必要があることにも注意してください。

リスト4　rootユーザーのパスワード変更例

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '変更後のパスワード';
```

パスワードの変更は、リスト5のようにSET PASSWORDコマンドを使用することも可能ですが、標準的なSQLであるALTER USER文によるパスワード変更が推奨されています。

リスト5　SET PASSWORDによるパスワードの変更

```
# 現在接続しているユーザーのパスワード変更
mysql> SET PASSWORD='変更後のパスワード';

# 任意のユーザーのパスワード変更（'root'@'localhost'の変更例）
SET PASSWORD FOR 'root'@'localhost'='変更後のパスワード';
```

```
# 備考：MySQL 5.7.5以前でのSET PASSWORD使用方法（現在はこのやり方は非推奨）
SET PASSWORD=PASSWORD('変更後のパスワード');
```

初期データベース作成に関する補足

MySQL 5.6までは、初期データベース作成にmysql_install_dbというコマンドを使用していました。この場合、以下の通り脆弱性となり得る事象を含む状態でデータベースが作成されていました（--random-passwordsオプション未使用の場合）。

- testデータベースが作成される
- rootユーザーにパスワードが設定されない
- リモートから接続可能なrootユーザーが作成される
- 匿名ユーザーが作成される

これらの問題をまとめて解決するために、mysql_secure_installtaionというコマンドが用意されています。mysql_secure_installtaionを使用すると以下の操作をまとめて実行し、セキュリティを強化できます。

- testデータベースを削除
- rootユーザーにパスワードを設定
- リモートから接続可能なrootユーザーを削除
- 匿名ユーザーを削除

　このように、MySQL 5.6までは、デフォルトはセキュリティが弱い状態で後から強化する、というアプローチでした。しかし、MySQL 5.7では、デフォルトでセキュリティが強化された状態に変更されています。

11.4 ユーザー関連のセキュリティ対策

ここでは、ユーザー関連のセキュリティ対策を解説します。

11.4.1 アクセス可能なホストと権限を制限する

MySQLでは、ユーザーアカウントは'username'@'host'のようにユーザー名と接続ホスト名の組み合わせで構成されます。図1のように同じユーザー名であっても、接続ホストが違えば別のユーザーアカウントとなります。

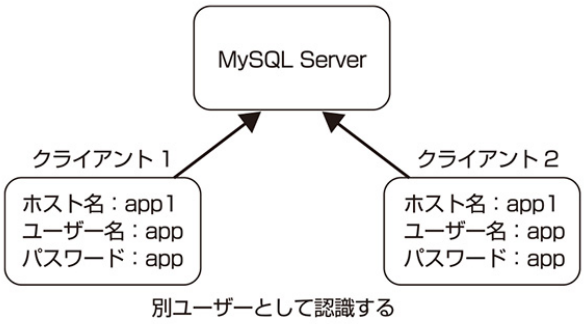


図１　ユーザーアカウントの定義

そのため、通常はこのようなことはしませんが、接続ホストごとに異なる権限を付与した同じ名前のユーザーを作成することも可能です。

ユーザーアカウントの例

- 'root'@'localhost'：localhostから接続するrootユーザー
- 'root'@'192.168.56.101'：192.168.56.101から接続するrootユーザー

ユーザーアカウントの定義に接続ホスト名を含むことで、特定のホストからのみアクセスできるユーザーを作成できます。例えば、以下のようにユーザーを定義することで、MySQLサーバーに対して意図しないアクセスが発生するリスクを低減できます。

- アプリケーションユーザーの接続ホスト名にはアプリケーションサーバーのホスト名を指定する
- MySQLサーバーのメンテナンス作業を行う管理者ユーザーのホスト名には'localhost'を指定し、リモートから作業する場合はsshでOSにログインしてから作業する

接続ホスト名をIPアドレスで指定すると、DNSやhostsファイルによる名前解決を回避できるため、ユーザー認証にかかる時間を短くできます。また、接続ホスト名には「%」をワイルドカードとして指定できるため、'root'@'192.168.56.%'のようにすれば、特定のサブネットからアクセスできるユーザーを定義できます。

ユーザー名を指定しない匿名ユーザーを作成することもできますが、セキュリティの観点から好ましくないため匿名ユーザーの使用は避けましょう。

ユーザー作成はCREATE USER文を使用し、ユーザー削除はDROP USER文を使用します。以下はlocalhostから接続するtestユーザーを作成する例、削除する例です（リスト6）。

リスト6　ユーザー作成、ユーザー削除

```
# ユーザー作成mysql> CREATE USER 'test'@'localhost' IDENTIFIED BY 'test';

# ユーザー削除
mysql> DROP USER 'test'@'localhost';
```

ユーザー作成後は、そのユーザーに必要な権限を付与します。権限の付与はGRANT文を用い、権限の剥奪はREVOKE文を用います。ユーザーに付与できる権限には、以下の5つのレベルがあります。

表3　権限レベルと権限の例

権限レベル	権限の例
グローバル	SUPER、CREATE USER、FILE、など
データベース	CREATE、DROP、LOCK TABLES、など
テーブル	ALTER、CREATE、SELECT、INSERT、など
カラム	SELECT、INSERT、UPDATE、など
ストアドルーチン	CREATE ROUTINE、EXECUTE、など

権限の詳細については、以下のリファレンスマニュアルを参照してください。

- **URL : MySQL 5.7 Reference Manual : 14.7.1.4 GRANT Syntax（Privileges Supported by MySQL以降を参照）**
  http://dev.mysql.com/doc/refman/5.7/en/grant.html

リスト7は、worldデータベースのCityテーブルに対してSELECT/INSERT/UPDATE/DELETE操作を許可する権限を'test'@'localhost'ユーザーに付与する例と、その権限を剥奪する例です。

リスト7　権限付与、権限剥奪

```
# GRANT文で権限を付与
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON world.City TO 'test'@'localhost';

# REVOKE文で権限を剥奪
```

```
mysql> REVOKE SELECT,INSERT,UPDATE,DELETE ON world.City FROM 'test'@'localhost';
```

余分な権限は脆弱性となり得るため、それぞれのユーザーには必要最低限の権限のみを付与し、余分な権限を付与しないようにします。

ユーザーに付与されている権限は、SHOW GRANTS文やインフォメーション・スキーマ、mysql.procs_privテーブルから確認できます。表4のインフォメーション・スキーマが利用可能です。

リスト8　権限の確認

```
# SHOW GRANTS 文による権限確認 mysql> SHOW GRANTS FOR 'test'@'localhost';
+-----------------------------------------------------------------------------+
| Grants for test@localhost                                                   |
+-----------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'test'@'localhost'                                    |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `world`.`City` TO 'test'@'localhost' |
+-----------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

```
# インフォメーション・スキーマによる権限確認
mysql> SELECT * FROM INFORMATION_SCHEMA.<table名>;

# ストアドプロシージャの権限確認
mysql> SELECT * FROM mysql.procs_priv;
```

表 4　権限を確認できるインフォメーション・スキーマ

テーブル名	確認できる情報
USER_PRIVILEGES	グローバルレベルで付与されている権限
SCHEMA_PRIVILEGES	スキーマ（データベース）レベルで付与されている権限
TABLE_PRIVILEGES	テーブルレベルで付与されている権限
COLUMN_PRIVILEGES	カラム（列）レベルで付与されている権限

ログインユーザーと現在のユーザーの違い

　ユーザーアカウントの接続ホスト名にワイルドカードを指定できることや、匿名ユーザーを使用できることなどから、ログイン時に指定されたユーザーと実際に認証に使用されたユーザーが異なることがあります。これらを確認するために、USER 関数とCURRENT_UESR関数が使用できます。以下の例では、ユーザーは「apl」というユーザー名で「192.168.56.101」のホストから接続し、実際に認証に使用されたユーザーは「apl@192.168.56.%」であることがわかります。

リスト9　ログインユーザーと現在のユーザーの違い

```
mysql> SELECT USER(),CURRENT_USER();
+-------------------+-----------------+
| USER()            | CURRENT_USER()  |
```

```
+-------------------+-----------------+
| apl@192.168.56.101 | apl@192.168.56.% |
+-------------------+-----------------+
1 row in set (0.00 sec)
```

そのため、このセッションではユーザー「apl@192.168.56.%」に権限が付与された操作のみが実行できます。

プロキシユーザー（≒ロール）の活用

ロール（役割）とは、権限のセットに名前を付けて管理し、その権限のセットをまとめて特定ユーザーに付与できる機能です。例えば、「apl_developer」というロールを作成し、そのロールにアプリケーション開発者に必要な権限を付与します。そうすることで、アプリケーション開発者が使用するデータベース・ユーザーに対して、apl_developerロールを付与するだけで、必要な権限をまとめて付与できます。アプリケーション開発者に必要な権限に変更があった場合も、apl_developerロールに対する権限を変更するだけで、apl_developerロールが付与されているすべてのデータベース・ユーザーの権限を変更できるため、同じ権限を複数ユーザーに与える場合の権限管理が容易になります。

MySQL 5.7では、このロール自体は実装されていませんが、プロキシユーザーが標準実装されました。プロキシユーザーを使うと、あるユーザーアカウントでログインした時に、別のユーザーアカウントに付与されている権限で処理を実行できます。そして、このプロ

キシユーザーを活用することで、ロールを使った権限管理に近いことができます。具体例を解説します。

まず、デフォルトの認証方式でプロキシユーザーを使用するために、以下のシステム変数を有効にします。

- check_proxy_users
- mysql_native_password_proxy_users

その後、リスト10の例のようにユーザー作成、権限付与することで、複数のユーザーに対して同じ権限をまとめて付与できます。

リスト10　プロキシユーザーの使用例（≒ロールの使用例）

```
# システム変数の確認
mysql> SHOW GLOBAL VARIABLES LIKE '%proxy_users%';
+---------------------------------+-------+
| Variable_name                   | Value |
+---------------------------------+-------+
| check_proxy_users               | ON    |
| mysql_native_password_proxy_users | ON    |
| sha256_password_proxy_users     | OFF   |
+---------------------------------+-------+
3 rows in set (0.00 sec)

# プロキシ対象ユーザーの作成（≒ロールの作成）
mysql> CREATE USER proxy_base@localhost;
Query OK, 0 rows affected (0.00 sec)
```

```
# プロキシユーザーの作成（＝個別のユーザー作成）
mysql> CREATE USER admin_1@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER admin_2@localhost;
Query OK, 0 rows affected (0.00 sec)

# プロキシ対象ユーザーに対するPROXY権限を個別のユーザーに対して付与（≒個別のユーザーにロールを付与）
mysql> GRANT PROXY ON proxy_base@localhost TO admin_1@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT PROXY ON proxy_base@localhost TO admin_2@localhost;
Query OK, 0 rows affected (0.00 sec)

# プロキシ対象ユーザーに対して必要な権限を付与（≒ロールへの権限付与）
mysql> GRANT SELECT ON world.* TO proxy_base@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT DELETE ON world.* TO proxy_base@localhost;
Query OK, 0 rows affected (0.00 sec)
```

　この状態で個別のユーザーでログインすると、現在のユーザーはプロキシ対象ユーザーになり、プロキシ対象ユーザーに付与されて

いる権限が有効になります。

リスト11　プロキシユーザーの権限確認例

```
# admin_1ユーザーで接続して権限確認
$ mysql -u admin_1

mysql> SELECT USER(), CURRENT_USER(), @@session.proxy_user;
+-------------------+----------------------+----------------------+
|   USER()                          |   CURRENT_USER()                 |
@@session.proxy_user    |
+-------------------+----------------------+----------------------+
|        admin_1@localhost        |        proxy_base@localhost        |
'admin_1'@'localhost' |
+-------------------+----------------------+----------------------+
1 row in set (0.00 sec)

mysql> SHOW GRANTS;
+-----------------------------------------------------------------+
| Grants for proxy_base@localhost                                 |
+-----------------------------------------------------------------+
|            GRANT            USAGE            ON            *.*            TO
'proxy_base'@'localhost'                        |
| GRANT SELECT, DELETE ON `world`.* TO 'proxy_base'@'localhost'
|
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> exit
```

```
# admin_2ユーザーで接続して権限確認
$ mysql -u admin_2

mysql> SELECT USER(), CURRENT_USER(), @@session.proxy_user;
+-------------------+---------------------+----------------------+
|   USER()                          |   CURRENT_USER()                  |
@@session.proxy_user    |
+-------------------+---------------------+----------------------+
|        admin_2@localhost        |        proxy_base@localhost        |
'admin_2'@'localhost' |
+-------------------+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> SHOW GRANTS;
+-----------------------------------------------------------------+
| Grants for proxy_base@localhost                                 |
+-----------------------------------------------------------------+
|           GRANT           USAGE           ON           *.*           TO
'proxy_base'@'localhost'                      |
| GRANT SELECT, DELETE ON `world`.* TO 'proxy_base'@'localhost'
|
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Column **ロール**

本文中で解説した通り、MySQL 5.7ではロールが実装されていませんが、MySQLの次バージョンであるMySQL 8.0ではロールの実装が計画されています。原稿執筆時点で公開されているMySQL 8.0.0 DMR（開発途上版）でも、既にロールが実装されており、MySQL 8.0ではより柔軟な権限管理が可能になる予定です。

プロキシユーザーと比較した場合、ロールを使用するとより柔軟な権限管理ができます。具体的には、プロキシユーザーではできない以下のことができます。

- ロールにロールを付与できる（階層構造の権限管理ができる）
- ユーザーが任意で権限のセットを切り替えられる（ユーザーに複数のロールを付与し、ユーザーが有効にするロールを選択できる。デフォルトロールの設定も可能）

プロキシユーザーを使用した権限管理をロールで例えると、ロールを階層構造で使用せず、ユーザーには1つのロールだけを付与し、デフォルトロールでそのロールを有効にしているような状態となります。

11.4.2 ユーザーが利用するリソースを制限する

MySQLサーバーでは、ユーザーごとに以下のリソース制限を設定できます。DoS攻撃を防ぐ目的などで、これらのリソース制限を活用します。

- MAX_QUERIES_PER_HOUR：アカウントが1時間ごとに発行できるクエリーの数
- MAX_UPDATES_PER_HOUR：アカウントが1時間ごとに発行できる更新の数
- MAX_CONNECTIONS_PER_HOUR：アカウントが1時間ごとにサーバーに接続できる回数
- MAX_USER_CONNECTIONS：アカウントによるサーバーへの同時接続の数

ユーザーにリソース制限をかける場合は、GRANT USAGE文で制限を設定します。また、制限を削除する場合は、その値を0に設定します。以下は制限の設定例と制限の削除例です。

リスト12　リソース制限の設定、解除

```
# リソース制限の設定
mysql> ALTER USER 'test'@'localhost' WITH MAX_QUERIES_PER_HOUR 100;

# リソース制限の解除
mysql> ALTER USER 'test'@'localhost' WITH MAX_QUERIES_PER_HOUR 0;
```

リソース制限は、クライアントごとではなくアカウントごとであることに注意してください。例えばクエリー数の制限が100の場合に2つのクライアントから同時に接続すると、2つのクライアントの合計クエリー数が100に到達した時点で制限がかかります。

11.4.3　パスワード検証プラグインを有効にしてパスワードポリシーを設定する

パスワード検証プラグインを使用することで、ユーザーのパスワードに対してパスワードポリシーを設定でき、強固なパスワードをユーザーに強制できます。パスワード検証プラグインは、YumやrpmでインストールしたTarを展開した場合や、Windows環境でインストールした場合は、デフォルトでは無効になっています。

プラグインが有効化されていない場合は、以下のコマンドで有効化できます。

リスト13　パスワード検証プラグインの有効化

```
# パスワード検証プラグインの有効化（Linux系OS）
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';

# パスワード検証プラグインの有効化（Windows系OS）
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password';

# プラグインが有効になっていることを確認
mysql> SHOW PLUGINS;
+----------------------------+----------+--------------------+-------------
---------+---------+
| Name                       | Status   | Type               |
Library                | License |
```

```
+---------------------------+----------+--------------------+------------
---------+---------+
<中略>
|  validate_password                    |  ACTIVE         |  VALIDATE
PASSWORD   | validate_password.so | GPL       |
+---------------------------+----------+--------------------+------------
---------+---------+
45 rows in set (0.01 sec)
```

パスワード検証プラグイン使用時、デフォルトで有効になるパスワードポリシーは以下の通りですが、これらは関連するシステム変数を変更することでカスタマイズ可能です。

●デフォルトで有効になるパスワードポリシー

⇒8文字以上で英語大文字／小文字／数字／特殊文字を1文字以上含む

関連するシステム変数は表5の通りです。これらのシステム変数を変更することで、パスワードポリシーを任意でカスタマイズすることができます。

表 5　パスワード検証プラグイン関連の設定

システム変数名	説明
validate_password_policy	有効化するパスワードポリシーを選択する。設定できる値は、次の 3 種類。 0 または LOW：文字数のみ制限 1 または MEDIUM：文字数と、大文字／小文字、数字、特殊文字の使用文字数を制限 2 または STRONG：文字数と、大文字／小文字、数字、特殊文字の使用文字数と、使用できない単語（辞書ファイルに含まれる単語）を制限
validate_password_length	パスワードの最少文字数
validate_password_mixed_case_count	パスワードに含める大文字／小文字の最少文字数
validate_password_number_count	パスワードに含める数字の最少文字数
validate_password_special_char_count	パスワードに含める特殊文字の最少文字数
validate_password_dictionary_file	辞書ファイルのパスを指定する。validate_password_policy=2 に設定した場合、辞書ファイルに含まれる単語はパスワードで使用できない

11.4.4 mysql_config_editorを使って認証情報を管理する

各種のコマンドラインツールを使ってMySQLサーバーにアクセスする場合、--userまたは-uオプションでユーザー名、--passwordまたは-pオプションでパスワードを指定して接続しますが、この方法では毎回パスワードを入力する必要があります。データベース管理者が複数人いて共通のMySQLユーザーを使用する場合などは、それぞれの人がパスワードを管理するため、パスワードの流出リスクが上がります。

mysql_config_editorを使用すると、パスワードを含む接続情報を.mylogin.cnfファイルに暗号化して格納しておき、コマンドラインツールからの接続時は--login-pathオプションで接続情報の名称のみを指定することで接続できるため、パスワードの流出リスクを下げることができます。

mysql_config_editorの詳細は、以下のリファレンスマニュアルを参照してください。

- **URL：MySQL 5.7 Reference Manual：5.6.6 mysql_config_editor**
  http://dev.mysql.com/doc/refman/5.7/en/mysql-config-editor.html

なお、詳細については言及しませんが、mysql_config_editorによるパスワードの暗号化は、強力ではありません。スキルのある攻撃者であればパスワードを復号できてしまうため、.mylogin.cnfファイルを流出させないように注意してください。

11.4.5 外部認証を使用しユーザー管理を一元化する

ユーザー管理が煩雑な環境において、外部認証機能を活用することで、ユーザー管理を一元化できます。

MySQL Enterprise Editionでは、MySQL Enterprise Authenticationという外部認証機能が用意されています。MySQL Enterprise Authenticationを使用すると、MySQLサーバーのユーザー認証をLDAPやWindows Active DirectoryなどMySQLサーバーの外部で管理されるユーザー情報によって認証できます。

例えば、開発環境においてアプリケーション開発者ごとにMySQLユーザーを用意していて、一人のアプリケーション開発者が複数システムに関わっている場合、開発者の異動や退職などが発生した時には不要になったMySQLユーザーをすべて削除することが望ましいですが、削除漏れがあると脆弱性につながる可能性があります。このような場合、ユーザー認証をLDAP等で一元管理しておけば、MySQLユーザーの削除漏れがあったとしても、認証時にエラーとなるため（そのユーザーではログインできないため）、不要なアクセスを防ぐことができます。

Column　全文検索（Full Text Search）

全文検索とは、特定のキーワードを含むテキストを、全文検索用のインデックスを使って検索することです。RDBMSにおいて一般的に使用される「Bツリーインデックス」は、前方一致による絞り込みでは使用できますが、中間一致や後方一致による絞り込み

では使用できません。そのため、特定のキーワードを含むテキストを検索する用途には使えません。そこで、文字数の多いテキストを格納している列から特定のキーワードを含むテキストを高速に検索するために、全文検索用のインデックスが必要となります。

全文検索インデックスは、MySQL 4.0のMyISAMストレージエンジンで使用可能になり、MySQL 5.6ではInnoDBストレージエンジンで使用可能になりました。しかし、これらの全文検索インデックスは、日本語や中国語、韓国語のようなスペースを区切り文字としない言語ではキーワードを切り出すことができません。そのため、日本語で全文検索を利用するには、事前にアプリケーション側で分かち書きをして、単語の間にスペースを追加したテキストをMySQLに格納する、といった工夫が必要でした。

MySQL 5.7では、InnoDBの全文検索インデックスが日本語や中国語、韓国語に対応し、スペースを区切り文字としないテキストからでもキーワードを切り出してインデックスを作成できるようになりました。日本語のテキストをそのままMySQLに格納して全文検索を利用できるため、手軽に活用できます。

テキストからキーワードを切り出すためのパーサーとしては、N-gramとMeCabがサポートされています。

N-gramは、一定の文字数でキーワードを切り出す手法です。Nには文字数が入り、2文字単位で切り出す場合はbi-gramと呼びます。MySQL 5.7ではデフォルトでN-gramパーサーが使用可能にな

っているため、MySQL 5.7をインストールすればすぐにN-gramを使った全文検索が使用できます。

MeCabは、オープンソースの形態素解析エンジンで、日本語の辞書を使ってキーワードを切り出します。MySQL 5.7でMeCabパーサーを使用する場合は、プラグインを追加でインストールする必要がありますが、必要なモジュールはMySQL 5.7のバイナリに含まれて提供されているため、簡単な設定で使用可能になります。

MySQL 5.7の全文検索機能の使用方法等については、以下の資料も参考にしてください。

- **URL：MySQL 5.7 InnoDB 日本語全文検索（その1）**
  http://www.slideshare.net/yoyamasaki/20160209-inno-dbftsjp
- **URL：MySQL 5.7 InnoDB 日本語全文検索（その2）**
  http://www.slideshare.net/yoyamasaki/mysql-57-innodb
- **URL: MySQL 5.7 InnoDB 日本語全文検索(その3)**
  http://www.slideshare.net/yoyamasaki/20160929-inno-dbftsjp

11.5 演習

このレッスンではMySQLサーバーのセキュリティ対策として、インストール関連のセキュリティ対策、ユーザー関連のセキュリティ対策について学習しました。ここではsecure_file_privの設定、ユーザー権限の付与、リソース制限の設定、パスワード検証プラグインについて確認しましょう。

1. secure_file_privを設定せずに以下のSQLを実行し、SQLによって/etc/passwdファイルが参照できることを確認する。続いて、secure_file_privを任意のディレクトリまたはNULLに設定し、同様の操作を行っても/etc/passwdファイルが参照できないことを確認する

```
mysql> SELECT LOAD_FILE('/etc/passwd')＼G
```

ヒント：レッスン11の「11.3.3　secure_file_privを明示的に設定する」を参照

2. localhostから接続するtestユーザーを作成し、worldデータベース内の全オブジェクトに対するSELECT、INSERT、UPDATE、DELETE権限を付与する。その後、testユーザーで接続し、worldデータベース内の全オブジェクトに対してSELECT、INSERT、UPDATE、DELETEが実行できることを確認する

ヒント：レッスン11の「11.4.1　アクセス可能なホストと権限を制限する」を参照

3. 2で作成したtestユーザーに対して、「MAX_QUERIES_PER_HOUR 10」のリソース制限を設定する。その後testユーザーでクエリーを連続して実行し、クエリー実行回数が10を超えた場合の挙動を確認する

ヒント：レッスン11の「11.4.2　ユーザーが利用するリソースを制限する」を参照

4. パスワード検証プラグインを有効にして、以下のパスワードポリシーを設定する

●10文字以上

●英語大文字／小文字／数字を1文字以上含む

ヒント：レッスン11の「11.4.3　パスワード検証プラグインを有効にしてパスワードポリシーを設定する」を参照

解説

1．secure_file_privの設定は、以下のようにSHOW GLOBAL VARIABLESコマンドで確認できます。secure_file_privは動的には変更できないシステム変数なので、変更するためにはmy.cnfファイルに「secure_file_priv=NULL」等を設定し、MySQLサーバーを再起動します。

```
mysql> SHOW GLOBAL VARIABLES LIKE 'secure_file_priv';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| secure_file_priv |       |
+------------------+-------+
1 row in set, 1 warning (0.00 sec)
```

2．testユーザーの作成方法、testユーザーでの接続後の確認例は以下の通りです。

```
# rootユーザーで以下を実行してtestユーザーを作成
mysql> CREATE USER 'test'@'localhost' IDENTIFIED BY 'test';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON world.*
TO 'test'@'localhost';
Query OK, 0 rows affected (0.00 sec)

# testユーザーで接続し、以下を実行して権限が付与されていることを確認
```

```
mysql> SELECT * FROM world.City LIMIT 1;
+----+-------+-------------+----------+------------+
| ID | Name  | CountryCode | District | Population |
+----+-------+-------------+----------+------------+
|  1 | Kabul | AFG         | Kabol    |    1780000 |
+----+-------+-------------+----------+------------+
1 row in set (0.01 sec)

mysql> INSERT INTO world.City VALUES(9999,'TEST','JPN','TEST',9999);
Query OK, 1 row affected (0.00 sec)

mysql> UPDATE world.City SET Population=0 WHERE ID=9999;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> DELETE FROM world.City WHERE ID=9999;
Query OK, 1 row affected (0.00 sec)
```

3. testユーザーへのリソース制限設定方法、クエリー実行回数が10回を超える前後の挙動は以下の通りです。

```
#testユーザーへのリソース制限設定方法（rootユーザーで実行）
mysql> ALTER USER 'test'@'localhost' WITH MAX_QUERIES_PER_HOUR 10;
Query OK, 0 rows affected (0.00 sec)
```

```
#testユーザーでクエリー実行回数が10回を超る前と、超えた後の挙動
mysql> SELECT * FROM world.City LIMIT 1;
+----+-------+-------------+----------+------------+
| ID | Name  | CountryCode | District | Population |
+----+-------+-------------+----------+------------+
|  1 | Kabul | AFG         | Kabol    |    1780000 |
+----+-------+-------------+----------+------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM world.City LIMIT 1;
ERROR 1226 (42000): User 'test' has exceeded the 'max_questions' resource (current value: 10)
```

4. パスワード検証プラグインを有効にする方法は以下の通りです。

```
#rootユーザーで実行(Linux系OS)
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';

#rootユーザーで実行(Windows系OS)
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password';
```

次に、my.cnfに以下の通り設定してMySQLサーバーを再起動します。

```
validate_password_policy=MEDIUM
validate_password_length=10
validate_password_mixed_case_count=1
validate_password_number_count=1
validate_password_special_char_count=0
```

レッスン

12

セキュリティ part2

レッスン 11 に続き、MySQL のセキュリティ強化機能について学習します。このレッスンでは、ネットワーク関連、暗号化、その他のセキュリティ対策などについて解説します。

12.1 ネットワーク関連のセキュリティ対策

ネットワーク関連のセキュリティ対策を解説します。

12.1.1 必要最低限のネットワークインターフェースを使用する

MySQLサーバーは、デフォルトではすべてのネットワークインターフェースを使用します。アプリケーションサーバーと通信するインターフェースが限定されている場合は、システム変数bind-addressを設定することで、接続可能なインターフェースを使用することができます。

また、レッスン3「3.1.2　接続設定」でも解説した通り、システム変数skip-networkingを設定するとTCP/IP経由でのアクセスをすべて無効化できます。リモートアクセス不要な環境や、MySQLサーバーのメンテナンス時などは、skip-networkingを設定することで不必要なリモートからのアクセスを防げます。

12.1.2 ネットワーク通信をSSLで暗号化する

ネットワークにおける通信は、盗聴に備えて暗号化することが推奨されます。MySQLでは、クライアントセッションとの通信だけでなく、各種のコマンドラインツールとの通信もSSLを使用して暗号化可能です。MySQL 5.7では、デフォルトでSSL関連の設定が有効化されるため、簡単にSSLを使用できます。

SSLを使用するためには、SSL証明書や鍵が必要ですが、MySQL 5.7ではmysql_ssl_rsa_setupを実行することで、datadir配下にSSL証明書や鍵を作成できます。レッスン11のリスト1でも、mysql_ssl_rsa_setupを実行しています。Windows環境の場合は、mysql_ssl_rsa_setup.exeが提供されていますが、事前にOpenSSLをインストールし、openssl.exeへのパスを通しておく必要があることに注意してください。

以下のリスト1は、mysql_ssl_rsa_setupを実行した後に、mysqlコマンドラインクライアントからSSLを使用して接続し、SSL関連の設定を確認している例です。mysqlコマンドラインクライアント以外からの接続やクライアントツールでもSSLを使用可能です。

リスト1　SSL接続の使用例

```
# mysqlコマンドラインクライアントからSSLを使って接続
# --ssl-mode=REQUIREDを指定することにより、SSLで通信できない場合は接続自体がエラーになる
$ mysql -u root -p --ssl-mode=REQUIRED

# SSL関連のシステム変数を確認
# have_sslがYESとなっており、SSLがサポートされていることがわかる
```

```
mysql> SHOW GLOBAL VARIABLES LIKE '%ssl%';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| have_openssl  | YES             |
| have_ssl      | YES             |
| ssl_ca        | ca.pem          |
| ssl_capath    |                 |
| ssl_cert      | server-cert.pem |
| ssl_cipher    |                 |
| ssl_crl       |                 |
| ssl_crlpath   |                 |
| ssl_key       | server-key.pem  |
+---------------+-----------------+
9 rows in set (0.01 sec)

# statusコマンドにより、暗号化スイートがDHE-RSA-AES256-SHAであることを確認
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.7.13, for linux-glibc2.5 (x86_64) using EditLine wrapper

Connection id:          3
Current database:
Current user:           root@localhost
SSL:                    Cipher in use is DHE-RSA-AES256-SHA
Current pager:          stdout
```

<後略>

12.2 暗号化によるセキュリティ対策

ここでは暗号化によるセキュリティ対策について解説します。

12.2.1 バックアップファイルを暗号化する

バックアップファイルの盗難に備える場合、バックアップファイルを暗号化します。特に、古いバックアップファイルをテープなどの別媒体にアーカイブする場合などは、盗難のリスクが高くなるため、バックアップファイルを暗号化することが推奨されます。

mysqldump/mysqlpumpで取得したダンプファイルや物理バックアップファイルは、任意の暗号化手法（opensslコマンドなど）で暗号化することができます。また、MySQL Enterprise Backupを使用している場合は、--encryptオプションを使用してバックアップファイルを暗号化することもできます。MySQL Enterprise Backupによるバックアップの暗号化についてはリファレンスマニュアルを参照してください。

- **URL：MySQL Enterprise Backup ユーザーズガイド（バージョン3.11）：第8章バックアップの暗号化**
  https://dev.mysql.com/doc/mysql-enterprise-backup/3.11/ja/meb-encryption.html

バックアップファイルを暗号化すると、障害が発生してバックアップからリストアする場合に、事前に復号してからリストアする必要があるため、リストアにかかる時間が延びることに注意してください。

12.2.2　透過的データ暗号化でデータファイルを保護する

MySQL 5.7では、InnoDBデータファイルの「透過的データ暗号化」（Transparent Data Encryption）が可能になりました。透過的データ暗号化とは、アプリケーション側で何もしなくても、データが暗号化／復号される暗号化機能のことです。

MySQLユーザーがデータを格納する際には、MySQLサーバーが自動的にデータを暗号化して、データファイルに書き込みます。また、MySQLユーザーがデータにアクセスした場合は、MySQLサーバーが自動的に暗号化されているデータを復号してユーザーに渡します。そのため、この暗号化機能はデータへのアクセス権限を持つMySQLユーザーの不正を防ぐための暗号化機能ではありません。

透過的データ暗号化で対策できるのは、以下のような脅威です。

- OSに侵入されてデータファイルを盗み見られた
- バックアップファイルを盗まれた

データへのアクセス権限を持つMySQLユーザーの不正を防ぐための暗号化機能としては、「MySQL Enterprise Encryption」があります。これについては「12.2.3　機密データは暗号化してデータベースに格納する」で解説します。

MySQLの透過的データ暗号化には、以下の特徴があります。

- AES（Advanced Encryption Standard）※1ブロック暗号化アルゴリズムを使用

●2層鍵管理アーキテクチャを採用しているため、鍵のローテーション時にデータの復号／再暗号化が発生しない
●鍵管理製品と連携可能（MySQL Enterprise Editionのみ）

「2層鍵管理アーキテクチャ」と「鍵管理製品との連携」について補足します。

2層鍵管理アーキテクチャ

InnoDBの透過的データ暗号化では、以下2つの鍵を使用します。

●表領域鍵
●マスター暗号化鍵

InnoDB上のデータを暗号化／復号するために、表領域鍵を使用します。この表領域鍵は、表領域のヘッダーに格納されますが、表領域鍵はマスター暗号化鍵によって暗号化された状態で、表領域のヘッダーに格納されます。

鍵のローテーションを実施した場合、マスター暗号化鍵はローテーションされますが、表領域鍵はローテーションされません（ローテーションの必要がありません）。つまり、データを暗号化／復号するための鍵は変更されないため、鍵のローテーション時であってもデータの復号／再暗号化は発生しません。

鍵のローテーションは、以下の場合に実施する必要があります。

●鍵が盗まれた疑いがある

●鍵の管理者が変更になった（前任者が不正に鍵を所有していて悪用されるリスクがあるため）

IPAによる「安全な暗号鍵のライフサイクルマネージメントに関する調査」でも、暗号鍵には有効期間を設定し、同一の鍵を利用し続けることを避けるように推奨されています。

●**URL：安全な暗号鍵のライフサイクルマネージメントに関する調査**
https://www.ipa.go.jp/files/000013895.pdf

鍵管理製品との連携

暗号化機能を使用する場合、暗号鍵をどうやって管理するかは、重要な考慮事項になります。

例えば、クレジットカード業界のセキュリティ基準であり、クレジットカード業界以外でも参考にされることの多いPayment Card Industry Data Security Standard（PSI-DSS）では、暗号化鍵の管理に以下の要件を設けています。

●安全な形で保管する
●最小限の場所に保管する
●最小限の管理者のみをアクセス可能にする
●定期的に鍵を変更する

MySQL Enterprise Editionで使用可能なMySQL Enterprise TDE（Transparent Data Encryption）では、鍵管理製品であるOracle Key Vaultと連携して暗号化鍵（マスター暗号化鍵）を管理できるため、暗

号化鍵を安全に管理できます。また、Key Management Interoperability Protocol（KMIP） v1.2に準拠しているため、KMIP v1.2に準拠しているほかの鍵管理製品とも連携可能です。

MySQL Enterprise TDEを使用しない場合、暗号化鍵はファイルシステム上に配置されます。

透過的データ暗号化の使用方法

透過的データ暗号化を使用する場合、事前に以下の設定を行います。

- システム変数early-plugin-loadにkeyring_file.soを指定して、MySQLサーバー起動（Windows系OSの場合はkeyring_fileを指定）
- 鍵の配置先としてシステム変数keyring_file_dataもしくはkeyring_okv_conf_dir（Oracle Key Vault使用時）を指定※2

事前設定ができていれば、CREATE TABLE／ALTER TABLE時にENCRYPTION='Y'を指定するだけで、該当テーブルのデータを暗号化できます。以下は、透過的データ暗号化を有効にしてworld.Cityテーブルを作成する例です。この場合、テーブル作成後にINSERTしたデータは、自動的に暗号化されます。

リスト2　透過的データ暗号化の使用例（新規テーブル）

```
mysql> CREATE TABLE world.City (
  ID int(11) NOT NULL AUTO_INCREMENT,
  Name char(35) NOT NULL DEFAULT '',
  CountryCode char(3) NOT NULL DEFAULT '',
```

```
  District char(20) NOT NULL DEFAULT '',
  Population int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY CountryCode (CountryCode),
    CONSTRAINT city_ibfk_1 FOREIGN KEY (CountryCode)
REFERENCES world.Country (Code)
) ENCRYPTION='Y';
```

ALTER TABLE文を使って、既存のテーブルに対して暗号化を有効にすることもできます（リスト3）。内部的にはテーブルを再作成するため、データ量が多いテーブルに対して実行する時は、時間がかかることに注意してください。

リスト3　透過的データ暗号化の使用例（既存テーブル）

```
mysql> ALTER TABLE world.City ENCRYPTION='Y';
```

また、以下のコマンドを実行することで、暗号化鍵をローテーションできます（リスト4）。実データを復号／再暗号化する必要はないため、暗号化鍵のローテーションは短時間で完了します。

リスト4　マスター暗号化鍵のローテーション

```
mysql> ALTER INSTANCE ROTATE INNODB MASTER KEY;
```

12.2.3 機密データは暗号化してデータベースに格納する

悪意のあるユーザーからデータを守るために、機密データは暗号化してデータベースに格納することが推奨されます。

高度な機能や管理ツールが含まれているMySQL Enterprise Editionでは、機密データを暗号化するための強力な暗号化機能「MySQL Enterprise Encryption」が用意されています。例えば、クレジットカード番号などの機密情報を保持する場合に、その情報をMySQLデータベースに格納する前にMySQL Enterprise Encryptionの暗号化関数を使用して、暗号化してから格納できます。これにより、データが盗まれた場合でも情報が暗号化されていることにより機密性を保持できます。

データを復号するためには、MySQL Enterprise Encryptionの復号関数を使用する必要がありますが、MySQL Enterprise Encryptionでは、暗号化鍵と復号鍵を異なる鍵にできるため、機密性がより向上します。例えば、暗号化鍵にアクセス可能なアプリケーション開発者が、その暗号化鍵を使用して本番環境の機密データを復号しようとしても、鍵が異なるため復号できません。機密データは、復号鍵にアクセス可能なユーザーのみが復号できます。

12.3 その他のセキュリティ対策（監査、ファイアウォール）

ここでは、監査やファイアウォールについて解説します。

12.3.1 監査ログで不正操作や内部犯行を防ぐ

データ改ざんなどの不正操作を監視するために、監査ログの取得が推奨されます。また、監査ログを取得していることを関係者に通達することによって、抑止力により内部犯行を防ぐ効果も期待できます。2014年に発生した大手通信教育事業者による大規模個人情報漏洩事件も、正当な権限を持ったユーザーによる内部犯行によって発生しています。このような内部犯行を防ぐために抑止力を働かせることも、セキュリティリスクを下げるためには重要なポイントです。

MySQL Enterprise Editionでは、監査ログを取得できる機能として「MySQL Enterprise Audit」が用意されています。MySQL Enterprise Auditを使用することで、ログオン、クエリーの情報を監査可能です。監査ログはXMLファイルに出力されますが、このXMLファイルのフォーマットは、オラクルの仕様にあわせており、「Oracle Audit Vault and Database Firewall」と連携可能になっています。Oracle Audit Vault and Database Firewallと連携することで、監査ログをMySQLサーバーから分離して集約して管理し、怪しいアクセスがあった時に警告を通知することなどができます。

また、MySQL 5.7.13以降では、特定の操作に対してのみ監査ログを取得するなど、取得する監査ログに対して詳細な絞り込み条件を定義可能になりました。

12.3.2 ファイアウォールで想定しないアクセスをブロックする

SQLインジェクション攻撃やクラッカーによる攻撃、悪意のあるユーザーによる内部犯行などによって、MySQLサーバーに対して想定しないパターンのSQLが実行される可能性があります。ファイアウォールを使用することで、このような想定外のアクセスをブロックでき、セキュリティリスクが低減できます。

MySQL Enterprise Editionには、「MySQL Enterprise Firewall」というホワイトリスト形式のファイアウォール機能が用意されています。あらかじめ実行可能なSQLをホワイトリストに定義しておくことで、ホワイトリストに定義されていないSQLの実行をブロックできます。このホワイトリストはユーザー単位で定義できるため、定型のSQLを実行するアプリケーションユーザーに対してのみファイアウォール機能を有効にし、非定型のSQLを実行するMySQLサーバーの管理者ユーザーに対してはファイアウォール機能を有効化しない、といった使い方も可能です。

ホワイトリストは、自動的に作成可能です。MySQL Enterprise Firewall使用時は、事前に学習モードに設定することで、その間に実行したSQLを自動的に記録し、ホワイトリストに追加します。SQLがホワイトリストに記録される時は、以下のようにパターン化されます（リスト5）。

リスト5　MySQL Enterprise Firewallのホワイトリスト記録例

```
# 実行したSQL文（fwuser@localhostで実行）
mysql> select * from world.City where id=1;
```

```
# ホワイトリストに記録されたSQL文の確認（インフォメーションスキーマから確認可能）
mysql> SELECT RULE FROM INFORMATION_SCHEMA.MYSQL_FIREWALL_WHITELIST WHERE USERHOST = 'fwuser@localhost';
+----------------------------------------------+
| RULE                                         |
+----------------------------------------------+
| SELECT * FROM `world` . `City` WHERE `id` = ?  |
+----------------------------------------------+
1 rows in set (0.00 sec)
```

WHERE id=1、WHERE id=2など、WHERE句の条件値の指定が異なるSQL文は同じSQLと見なされるため実行可能ですが、それ以外のパターンのSQLは実行できません。例えば、不正に情報を抜き出すためにSQLインジェクションでリスト6のようなSQLが実行されたとしても、これはホワイトリストに一致しないためブロックされます。

リスト6　MySQL Enterprise FirewallによるSQLブロック例

```
# SQLインジェクションにより、WHERE句に「OR 1=1」を追加したSQLが実行されたことを想定
mysql> SELECT * FROM world.City WHERE id=1 OR 1=1;
ERROR 1045 (28000): Statement was blocked by Firewall
```

システム変数mysql_firewall_traceをONに設定していると、ホワイトリストと一致しないSQLをブロックした時に、同時にエラーログにもSQLをブロックしたことを記録します。また、アプリケーションから実行されるSQLに非定型なものが含まれる場合や、ホワイトリストの登録漏れを懸念する場合は、SQLをブロックせずに、ホワイトリストに一致しないSQLが実行されたことをエラーログに出力するだけの「検知（DETECTING）モード」も使用可能です。これらの出力を監視することで、どのようなSQLによる想定外のアクセスがあったかを確認できます（リスト7）。

リスト7　ホワイトリストに一致しないSQL実行時のエラーログ出力例

```
# PROTECTINGモードの場合のエラーログ出力例
2016-06-25T12:01:37.673266+09:00 5 [Note] Plugin MYSQL_FIREWALL reported: 'ACCESS DENIED for 'fwuser@localhost'. Reason: No match in whitelist. Statement: SELECT * FROM `world` . `City` WHERE `id` = ? OR ? = ? '

# DETECTINGモードの場合のエラーログ出力例
2016-06-25T12:03:07.676878+09:00 5 [Note] Plugin MYSQL_FIREWALL reported: 'SUSPICIOUS STATEMENT from 'fwuser@localhost'. Reason: No match in whitelist. Statement: SELECT * FROM `world` . `City` WHERE `id` = ? OR ? = ? '
```

さらに、SQLをブロックしたり検知した時には、関連するステータス変数が増加するため、以下のステータス変数を監視することで、アタックを受けているなど、想定外のアクセスが発生していることを早

期に発見できます。監視ツールであるMySQL Enterprise Monitorと組み合わせれば、想定外のアクセスが発生している場合にメールやSNMP Trapで警告を通知することもでき、問題を迅速に検知できます。

表1 MySQL Enterprise Firewall 関連のステータス変数の例

ステータス変数	内容
Firewall_access_denied	SQL をブロックした数
Firewall_access_suspicious	SQL を検知した数

MySQL Enterprise Firewallは、完全にMySQLサーバーに統合されているため、使用する際に追加のサーバーやネットワーク機器を用意する必要もありません。追加のプロセスの起動も不要です。インストールが完了していれば、以下のコマンドでプロシージャをコールするだけで使用できるため、非常に手軽に使用できるファイアウォール機能となっています。

- ホワイトリストの学習：CALL sp_set_firewall_mode（'ユーザー名','RECORDING'）;
- 防御モード有効化：CALL sp_set_firewall_mode（'ユーザー名','PROTECTING'）;
- 検知モード有効化：CALL sp_set_firewall_mode（'ユーザー名','DETECTING'）;
- ファイアウォール無効化：CALL sp_set_firewall_mode（'ユーザー名','OFF'）;
- ホワイトリスト初期化：CALL sp_set_firewall_mode（'ユーザー名','RESET'）;

12.4 MySQL Enterprise Editionの機能を試す方法

レッスン11とこのレッスンで紹介した以下の機能は、MySQL Enterprise Editionのみで使用できる追加機能です。

- ●MySQL Enterprise Authentication（外部認証）
- ●MySQL Enterprise TDE（機密データの保護）※3
- ●MySQL Enterprise Encryption（非対称暗号化）
- ●MySQL Enterprise Audit（監査ログ取得）
- ●MySQL Enterprise Firewall（SQLインジェクション対策）

これらの機能は、MySQL Enterprise Editionを契約することで使用できますが、評価目的であれば30日間限定で試すことができます。以下のサイトにアクセスし、ログイン後、規約に同意することで、評価目的で製品版と同一のバイナリをダウンロードできます。

- ●**URL：Oracle Software Delivery Cloud**
  https://edelivery.oracle.com/

Platformで適切なOSを選択し、「MySQL Enterprise Edition」で検索して必要な製品をダウンロードします。MySQL Enterprise Backup、MySQL Enterprise MonitorはMySQL Databaseとバイナリが分かれていますが、それ以外のMySQL Enterprise Editionの機能は、MySQL Databaseのバイナリに同梱されて提供されています。

MySQL Enterprise Editionの各機能の使用方法は、以下のリファレンスマニュアルを参照してください。

MySQL Enterprise Authentication

- **URL : MySQL 5.7 Reference Manual : 7.5.1.5 The PAM Authentication Plugin**
  http://dev.mysql.com/doc/refman/5.7/en/pam-authentication-plugin.html
- **URL : MySQL 5.7 Reference Manual : 7.5.1.6 The Windows Native Authentication Plugin**
  http://dev.mysql.com/doc/refman/5.7/en/windows-authentication-plugin.html

MySQL Enterprise TDE

- **URL : MySQL 5.7 Reference Manual : 15.7.10 InnoDB Tablespace Encryption**
  http://dev.mysql.com/doc/refman/5.7/en/innodb-tablespace-encryption.html
- **URL : MySQL 5.7 Reference Manual : 7.5.3 The MySQL Keyring**
  http://dev.mysql.com/doc/refman/5.7/en/keyring.html

MySQL Enterprise Encryption

- **URL : MySQL 5.7 Reference Manual : 13.18 MySQL Enterprise Encryption Functions**
  https://dev.mysql.com/doc/refman/5.7/en/enterprise-encryption.html

MySQL Enterprise Audit

- **URL : MySQL 5.7 Reference Manual : 7.5.4 MySQL Enterprise Audit**
  http://dev.mysql.com/doc/refman/5.7/en/audit-log.html

MySQL Enterprise Firewall

- **URL : MySQL 5.7 Reference Manual : 7.5.5 MySQL Enterprise Firewall**
  http://dev.mysql.com/doc/refman/5.7/en/firewall.html

Column **GIS（地理情報システム）**

Geographic Information System（GIS）とは、地理情報を活用するシステムのことを指します。MySQLでは、MySQL 4.1からMyISAMにおいてGIS関連機能が実装されましたが、この実装はMySQL開発チームの独自実装であったため、複雑な使い方をすると正確に地理情報を処理できない場合がある、MyISAMなのでトランザクションが使用できない、などの問題を抱えていました。

そこで、MySQL 5.7ではGIS関連機能を刷新し、Boost.GeometryというC++のオープンソースライブラリを採用してInnoDBで再実装しました。これにより、InnoDBで以下の機能が使用可能になりました。

- ●GEOMETRYデータ型（POINT、LINE、POLYGON、GEOMETRYなど）
- ●空間インデックス（Rツリーインデックス）
- ●ST_Contains()、ST_Distance()などのSpatial関数
- ●経緯度の情報を文字列化し、文字列の桁数で精度を変えることができるGeoHash関数
- ●JSONでジオメトリデータを扱えるGeoJSON関数

また、各種のSpatial関数について命名規則が整理されました。Spatial関数にはST_というプレフィックスが付くものと、MBRというプレフィックスが付くもの、どちらのプレフィックスも付かないものが存在します。MySQL 5.7では、ジオメトリ同士の関係を比較する関数について、どちらのプレフィックスも付かない関数は挙動が不明確で混乱を招くため、廃止予定となりました。

ST_とMBRの違いは、図形の形に基づいた判定をするのか、MBRに基づいた判定をするのかです。MBR（Minimum Bounding Rectangle：最小外接矩形）とは、ある図形に隣接する最小の矩形です（図1）。

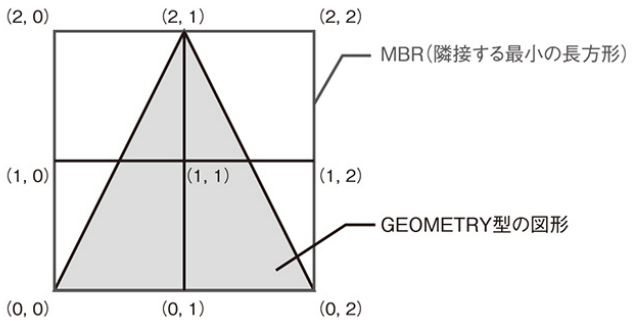


図1 MBR (Minimum Bounding Rectangle：最小外接矩形)

例えば、あるジオメトリが別のジオメトリに含まれているか否かを判定するContains()という関数は、それぞれ以下のように動作します。

- MBRContains(g1,g2)：g1の最小外接矩形にg2の最小外接矩形が含まれているかどうかを示す
- ST_Contains(g1,g2)：g1にg2が含まれているかどうかを示す

このため、MBRContains()は大まかな判定をすることに向いていますが、正確な判定が必要な場合はST_Contains()を使用する必要があります。

12.5 演習

このレッスンではMySQLサーバーのネットワーク関連のセキュリティ対策、暗号化／監査／ファイアウォールによるセキュリティ対策について学習しました。ここでは透過的データ暗号化、ファイアウォール、監査ログ取得について確認しましょう。

1. **透過的データ暗号化によりworld.Cityテーブルを暗号化し、Cityテーブルのデータファイル（City.ibd）が暗号化されていることを確認する。この時、システム変数keyring_file_dataを使用し、暗号化鍵はファイルシステム上に配置する**

   ヒント：暗号化されていることの確認方法例は、City.ibdファイルをviエディタで開き、「Tokyo」という文字列を検索する。暗号化前は「Tokyo」という文字列が見つかるが、暗号化後は見つからない

   ヒント：レッスン12の「12.2.2　透過的データ暗号化でデータファイルを保護する」を参照

2. **Oracle Software Delivery Cloudから評価目的でMySQL Enterprise EditionのMySQL Databaseをダウンロードし、インストールする**

   ヒント：レッスン12の「12.4　MySQL Enterprise Editionの機能を試す方法」およびレッスン1「MySQL 5.7で新しくなったインストール方法」を参照

3. MySQL Enterprise Firewallを使用して、ホワイトリストに掲載されていないSQLをブロックできることを確認する

ヒント：レッスン12の「12.3.2　ファイアウォールで想定しないアクセスをブロックする」を参照

4. MySQL Enterprise Auditを使用して、監査ログが取得できることを確認する

ヒント：レッスン12の「12.3.1　監査ログで不正操作や内部犯行を防ぐ」を参照

解説

1. 透過的データ暗号化を使用するために、my.cnfに以下を設定してMySQLサーバーを再起動します。

```
early-plugin-load=keyring_file.so　# Windows系OSの場合はkeyring_fileを指定
keyring_file_data=<<任意のパスを指定(ディレクトリではなく、ファイル名まで指定)>>
```

次に、以下のSQLを実行してworld.Cityテーブルを暗号化します。

```
mysql> ALTER TABLE world.City ENCRYPTION='Y';
```

暗号化後、City.ibdファイルをviエディタ等で開いて、暗号化されていることを確認してください。

2. ヒント部分の解説を参照してください。

3. MySQL Enterprise EditionのMySQLデータベースをインストールした環境にMySQL Enterprise Firewallをインストールするため、linux_install_firewall.sqlを実行します。Windows系OSの場合は、win_install_firewall.sqlを実行します。それぞれのSQLスクリプトは、MySQLをインストールしたディレクトリ配下にあるshareディレクトリの中にあります。

```
# mysql -u root -p ￥
```

次に、testユーザーに対してMySQL Enterprise Firewallの学習モードを有効にするために、rootユーザーで以下のコマンドを実行します。

```
# rootユーザーで実行
mysql>                                                    CALL
mysql.sp_set_firewall_mode('test@localhost','RECORDING');
+-----------------------------------------------+
| read_firewall_whitelist(arg_userhost,FW.rule) |
+-----------------------------------------------+
| Imported users: 0Imported rules: 0
              |
+-----------------------------------------------+
1 row in set (0.00 sec)Query OK, 0 rows affected (0.00 sec)
```

次に、testユーザーで接続し、以下のSQLを実行してホワイトリストに記録します。

```
# testユーザーで実行
mysql> SELECT * FROM world.City WHERE id=1;
+----+-------+-------------+----------+------------+
| ID | Name   | CountryCode | District | Population |
+----+-------+-------------+----------+------------+
|   1 | Kabul | AFG            | Kabol      |      1780000 |
+----+-------+-------------+----------+------------+
1 row in set (0.00 sec)
```

ホワイトリストが記録できたら、rootユーザーで以下を実行して、ファイアウォールを有効にします。

```
# rootユーザーで実行
mysql>                                                    CALL
sp_set_firewall_mode('test@localhost','PROTECTING');
Query OK, 2 rows affected (0.00 sec)
```

ホワイトリストを確認したい場合は、rootユーザーで以下を実行します。

```
# rootユーザーで実行
mysql> SELECT * FROM mysql.firewall_whitelist;
+----------------+-------------------------------------------------+
|        USERHOST                                  |        RULE
|
+----------------+-------------------------------------------------+
| test@localhost | SELECT * FROM `world` . `City` WHERE `id`
= ?   |
|  test@localhost  |  SELECT  @@`version_comment`  LIMIT  ?
|
+----------------+-------------------------------------------------+
2 rows in set (0.00 sec)
```

その後、testユーザーで接続し、以下のSQLを実行して、SQLがブロックされることを確認します。

```
# testユーザーで実行
mysql> SELECT * FROM world.City;
```

```
ERROR 1045 (28000): Statement was blocked by Firewall
```

4. MySQL Enterprise EditionのMySQLデータベースをインストールした環境で、MySQL Enterprise Auditを有効にするために以下のコマンドを実行します。

```
mysql> INSTALL PLUGIN audit_log SONAME 'audit_log.so';
# Windows系OSの場合はaudit_logを指定
```

その後、何かSQLを実行し、datadir配下のaudit.logに監査ログが出力されることを確認します。デフォルトで監査ログ取得が有効になっているため、明示的にシステム変数を設定しなくても監査ログが取得できます。
また、Audit関連のシステム変数は、以下の方法で確認できます。

```
mysql> SHOW GLOBAL VARIABLES LIKE 'audit_%';
+---------------------------+-------------+
| Variable_name             | Value       |
+---------------------------+-------------+
| audit_log_buffer_size     | 1048576     |
| audit_log_connection_policy | ALL       |
| audit_log_current_session | OFF         |
| audit_log_exclude_accounts |            |
| audit_log_file            | audit.log   |
| audit_log_filter_id       | 0           |
| audit_log_flush           | OFF         |
```

```
| audit_log_format                | NEW              |
| audit_log_include_accounts      |                  |
| audit_log_policy                | ALL              |
| audit_log_rotate_on_size        | 0                |
| audit_log_statement_policy      | ALL              |
| audit_log_strategy              | ASYNCHRONOUS |
+---------------------------------+--------------+
13 rows in set (0.00 sec)
```

※１ 米国の国立標準技術研究所によって制定されている、標準的な暗号化アルゴリズム。

※２ keyring_okv_conf_dirは、MySQL Enterprise TDEでのみ使用可能。

※３ 鍵管理製品との連携部分のみMySQL Enterprise Editionが必要。

レッスン

13

パフォーマンスチューニングの基礎

このレッスンでは、パフォーマンスチューニングの基礎について学習します。

13.1 MySQLパフォーマンスチューニング概論

パフォーマンスチューニングを実施する場合、チューニングの指標やボトルネックを明確にして適切なチューニングをすることが重要です。また、単一処理（単一SQL）のレスポンスタイムをチューニングすることも大切ですが、アプリケーション全体としてチューニングを実施する場合にはベンチマークテストを実施することも重要です。

13.1.1 チューニングの指標

チューニングの指標として代表的なものに、「スループット」と「レスポンスタイム」があります。

スループット

スループットは、単位時間当たりの処理能力を表す指標です。よく使われる指標に「Transaction Per Second」（TPS）があります。TPSは、1秒間で処理したトランザクション数を表します。

レスポンスタイム

レスポンスタイムは、ある処理を実行してから結果が返ってくるまでの時間です。データベースにおけるレスポンスタイムはSQLの実行時間ですが、アプリケーションとしてのレスポンスタイムには以下のような要因も含まれます。

- アプリケーションサーバーでの処理時間
- データベースサーバーとアプリケーションサーバーのネットワーク通信時間

ロック待ちやキューイングによる待ちが発生している場合、これらの待ち時間もレスポンスタイムに含まれることに注意が必要です。キューイングによる待ちとは、複数の同時実行処理がある場合に処理をさばききれず、実行される前にキューで待たされることです。実社会では、サポートセンターに電話した時に電話が集中している旨の音声ガイダンスが流れ、オペレーターにつながるまでに待

たされることがあると思いますが、これがキューイングによる待ちに該当します。

　チューニングにおいて、レスポンスタイムが向上することで結果としてスループットが向上する場合もありますが、そうならない場合もあることに注意してください。代表的な例として、インデックスによるチューニングがあります。データベースには検索処理のレスポンスタイムを向上するためにインデックスを付けることができますが、インデックスは更新処理に対してはオーバーヘッドとなり得ます。インデックスを追加することで、データを更新する際にテーブルデータだけでなくインデックスも更新する必要があるからです。そのため、不要なインデックスはできる限り作成しない方が、更新処理のスループットの向上につながります。

13.1.2 ボトルネックの特定とチューニング

チューニングを行う時はボトルネックを特定し、そのボトルネックを解消します。

ボトルネックを正確に特定するには、さまざまな情報を確認する必要があります。実行時間だけではなく、ある処理を実行した時のOSリソースの使用状況、MySQLサーバーの稼働状況などを確認し、ボトルネックを特定します。MySQLサーバーでは、ボトルネックの特定に役立つ稼働情報を確認するための仕組みとして、ステータス変数やパフォーマンス・スキーマなどの仕組みがあります。これらについては後ほど解説します。

ボトルネックを解消するためのチューニング方法は、1つとは限りません。例えば、先ほどのキューイングによる待ちが発生している場合には、以下のチューニング方法が考えられます。

- **同時実行処理数を増やすことで、キューイングせずに実行できる処理数を増やす**
  前述したサポートセンターの例では、対応するオペレーターの人数を増加することに該当
- **1つ1つの処理の処理時間を短縮することで、キューに入りにくくする**
  前述したサポートセンターの例では、オペレーターの対応手順を改善するなどして、オペレーターが1件の問合せにかかる時間を短縮することに該当

考えられるチューニング方法の中から、実現可能な方法でチューニング指標を達成できるようにチューニングします。

13.1.3 ベンチマークテスト

チューニングを実施する場合、単一処理（単一SQL）のレスポンスタイムをチューニングすることも重要ですが、アプリケーション全体をチューニングすることも重要です。アプリケーション全体のチューニングを実施する場合に、単一処理のテストでは発見できなかった性能特性を明らかにするために、ベンチマークテストを実施します。

ベンチマークテストを実施する際は、本番環境と同様の条件でテストを実施しないと、正しくボトルネックを特定できないことに注意しましょう。例えば、以下のような失敗例があります。

- **本番環境より少ないデータ量でテスト**
  ⇒テストでは全データがキャッシュに乗っていたのでパフォーマンスが良かったが、本番環境ではキャッシュアウトが発生し、想定していたパフォーマンスが出なかった
- **データやリクエストのばらつきを考慮しないでテスト**
  ⇒テストでは全データに均一にアクセスしていたが、本番環境ではデータのアクセスに偏りがあった。そのため、ベンチマークテストでは特定データを更新する際のロック待ちが大量に発生することに気付けなかった

このように条件の異なるベンチマークテストから得た性能特性を基にチューニングしても、本番環境において有効なチューニング方法となるとは限りません。ベンチマークテストを実施する際は、できる限り本番環境と同様の条件でテストを実施しましょう。

13.2 稼働状況と設定の確認

MySQLサーバー全体のチューニングを行う際は、MySQLサーバーの稼働状況や設定を確認して、必要に応じて設定（システム変数）を変更します。

13.2.1 ステータス変数

MySQLサーバーでは、稼働状況を確認する方法として「ステータス変数」が用意されています。例えば、以下のステータス変数が存在します。ステータス変数の変化を確認することにより、MySQLサーバーの稼働状況を確認できます。

表1　ステータス変数の例

ステータス変数	内容
Com_select	SELECT 文が実行された回数
Com_insert	INSERT 文が実行された回数
Innodb_buffer_pool_read_requests	InnoDB の論理読み取りリクエスト数
Innodb_buffer_pool_reads	InnoDB のディスク読み取りリクエスト数
Threads_connected	現在接続中のコネクション数

表1で紹介した以外にも、多数のステータス変数が存在します。詳細は以下のリファレンスマニュアルを参照してください。

●URL：MySQL 5.7 Reference Manual：6.1.7 Server Status Variables

https://dev.mysql.com/doc/refman/5.7/en/server-status-variables.html

大半のステータス変数は累積値を表示するため、ステータス変数を確認する時は、稼働状況を確認したい処理の前後でステータス変数を記録し、差分を取ることで該当処理による「ステータス変数の変化」を読み取ります。上で例に挙げたステータス変数の場合、Threads_connectedだけは現在の値（瞬間値）を表示するので差分を取る必要はありませんが、それ以外のステータス変数は累積値を表示するため、差分を取って変化を確認します。

ステータス変数は、グローバル単位のものとセッション単位のものがありますが、セッション単位のステータス変数はFLUSH STATUSコマンドを実行することで初期化できます。そのため、特定セッションでチューニング対象の処理を実行して、ステータス変数の変化を確認する場合は、FLUSH STATUSコマンド実行後に該当処理を実行し、その後ステータス変数を確認することで、差分を取らずに該当処理によるステータス変数の変化を確認できます。

ステータス変数を確認するには、以下の方法があります。

- mysqlコマンドラインクライアントからSHOW STATUSコマンドを実行する
- mysqladminのextended-statusコマンドを実行する
- MySQL Workbenchから確認する

MySQL Workbenchからステータス変数を確認する方法は、レッスン14で解説します。

SHOW STATUSコマンドによるステータス変数の確認

mysqlコマンドラインクライアントからSHOW STATUSコマンドを実行することで、ステータス変数を確認できます。構文は以下の通りです（リスト1）。

リスト1　SHOW STATUSコマンドの構文

```
mysql> SHOW [GLOBAL | SESSION] STATUS [LIKE 'pattern' | WHERE expr]
```

'pattern'部分に%をワイルドカードとして指定し、ステータス変数を絞り込んで表示できます。例えば、リスト2を実行すると、セッション単位のステータス変数の中から'Com_'という接頭辞を持つステータス変数のみを表示します。なお、「￥」はエスケープ文字です。「_」のみを指定すると文字数が1文字のワイルドカードと見なされてしまうため、エスケープ文字を使用しています。

リスト2　SHOW STATUSコマンドによるステータス変数の確認例

```
mysql> SHOW SESSION STATUS LIKE 'Com￥_%';
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| Com_admin_commands         | 0     |
| Com_assign_to_keycache     | 0     |
| Com_alter_db               | 0     |
<中略>
| Com_select                 | 13    |
<中略>
| Com_stmt_reprepare         | 0     |
+----------------------------+-------+
148 rows in set (0.01 sec)
```

mysqladminのextended-statusコマンドによるステータス変数の確認

mysqladminのextended-statusコマンドを実行することで、ステータス変数が確認できます。例えば、リスト3のように実行します。こ

のコマンドを実行すると、60秒ごとにステータス変数の差分を出力する動作を10回繰り返します。

リスト3　mysqladminによるステータス変数の確認例

```
$ mysqladmin -u root -p extended-status -i 60 -c 10 -r
```

　mysqladminのコマンドやオプションの詳細は以下のリファレンスマニュアルを参照してください。

- **URL：MySQL 5.7 Reference Manual：5.5.2 mysqladmin — Client for Administering a MySQL Server**
  https://dev.mysql.com/doc/refman/5.7/en/mysqladmin.html

13.2.2　システム変数

レッスン2の「2.5　設定ファイル」で解説した通り、システム変数はmy.cnfで設定します。稼働中に現在設定されているシステム変数を確認する方法としては、以下の方法があります。

- mysqlコマンドラインクライアントからSHOW VARIABLESコマンドを実行する
- mysqlコマンドラインクライアントからSELECT @@コマンドを実行する
- パフォーマンス・スキーマから確認する
- MySQL Workbenchから確認する

MySQL Workbenchからシステム変数を確認する方法は、レッスン14で解説します。

SHOW VARIABLESコマンドによるシステム変数の確認

SHOW VARIABLESコマンドによるシステム変数の確認方法は、レッスン2の「2.5.1　設定の確認」で解説していますので参照してください。

SELECT @@コマンドによるシステム変数の確認

SELECT文に以下の形式でシステム変数名を指定することで、システム変数を確認可能です。

- グローバル変数の確認：@@global.システム変数名
- セッション変数の確認：@@session.システム変数名

例えば、以下のコマンドを実行すると、システム変数sort_buffer_sizeの設定値をグローバル単位、セッション単位でそれぞれ確認できます（リスト4）。セッション単位で設定を変更している場合は、この例のようにグローバル単位の値とセッション単位の値が異なります。

リスト4　SELECT @@によるシステム変数の確認例

```
mysql>                                                    SELECT
@@global.sort_buffer_size,@@session.sort_buffer_size;
+--------------------------+--------------------------+
| @@global.sort_buffer_size | @@session.sort_buffer_size |
+--------------------------+--------------------------+
|                    262144 |                1073741824 |
+--------------------------+--------------------------+
1 row in set (0.00 sec)
```

パフォーマンス・スキーマによるシステム変数の確認

パフォーマンス・スキーマの以下のテーブルをSELECTすることで、グローバル単位、セッション単位のシステム変数を確認できます。

- performance_schema.global_variables
- performance_schema.session_variables

例えば、以下の例では、グローバル単位のシステム変数から接頭辞が「Innodb_buffer」であるシステム変数のみを確認できます（リ

スト5）。

リスト5　パフォーマンス・スキーマによるシステム変数の確認例

```
mysql> SELECT * FROM performance_schema.global_variables
WHERE VARIABLE_NAME LIKE 'innodb_buffer%';
+---------------------------------------+----------------+
| VARIABLE_NAME                         | VARIABLE_VALUE |
+---------------------------------------+----------------+
| innodb_buffer_pool_chunk_size         | 134217728      |
| innodb_buffer_pool_dump_at_shutdown | ON               |
| innodb_buffer_pool_dump_now           | OFF              |
| innodb_buffer_pool_dump_pct           | 25               |
| innodb_buffer_pool_filename           | ib_buffer_pool |
| innodb_buffer_pool_instances          | 1                |
| innodb_buffer_pool_load_abort         | OFF              |
| innodb_buffer_pool_load_at_startup    | ON               |
| innodb_buffer_pool_load_now           | OFF              |
| innodb_buffer_pool_size               | 134217728      |
+---------------------------------------+----------------+
10 rows in set (0.01 sec)
```

13.3 システム変数のチューニング例

システム変数のチューニングについて、いくつか例を説明します。

13.3.1 接続スレッドごとの設定値

MySQLサーバーにクライアントが接続して処理を実行すると、サーバー側では接続スレッドが生成され、接続スレッドごとにメモリー上に各種の領域を確保して処理を実行します。接続スレッドごとにメモリー上に確保する領域には、以下のものがあります。

表2 接続スレッドごとに確保するメモリー領域

システム変数名	概要
sort_buffer_size	ソート処理で利用するバッファ
join_buffer_size	インデックスを使用しない JOIN で利用するバッファ
tmp_table_size	一時テーブルの最大サイズ
read_buffer_size	MyISAM テーブルを全件検索する時のバッファ
read_rnd_buffer_size	ソート後にレコードを読み取る時のバッファ
binlog_cache_size	バイナリログに記録するトランザクションのバッファ
net_buffer_length	データ送受信のバッファの初期値
max_allowed_packet	データ送受信のバッファの最大値
thread_stack	スレッドのスタックサイズ

これらの設定の大半はデフォルトのままでも問題のないケースが多いのですが、必要に応じてチューニングを実施します。ここでは、チューニングすることが多い「sort_buffer_size」「tmp_table_size」について、チューニング方法を説明します。

sort_buffer_sizeのチューニング

sort_buffer_sizeは、ソート用のメモリー領域です。MySQLサーバーでソート処理を実行した場合、このメモリー領域のサイズを超えるソート処理は、ディスク上のファイルを利用して実行されます。つまり、ディスクI/Oが発生します。ディスクI/Oを削減することがレスポンスタイム向上につながるため、ファイルソートが発生して

いる場合にsort_buffer_sizeを拡張することで、チューニングできる可能性があります。

ファイルソートが発生しているか否かは、ステータス変数Sort_merge_passesから確認できます。Sort_merge_passesが増加している環境では、sort_buffer_sizeの増加を検討します。sort_buffer_sizeはセッション単位で動的に変更できるため、バッチ処理など特定の処理でだけSort_merge_passesが増加している環境では、その処理を実行するセッションだけsort_buffer_sizeを変更してチューニングすることも可能です。

tmp_table_sizeとmax_heap_table_sizeのチューニング

MySQLサーバーでは、UNION、ORDER BY、GROUP BY使用時など、クエリー処理中に内部的に一時テーブルを使用する場合があります。この時、一時テーブルのサイズが大きくなり、メモリー上の一時テーブルで処理しきれない場合は、ディスク上に一時テーブルを書き出します。メモリー上の一時テーブルのサイズの上限は、tmp_table_sizeおよびmax_heap_table_sizeの最小値で設定されているため、これらの設定値を増加することでディスクI/Oを削減し、チューニングできる可能性があります。

一時テーブルをディスク上に書き出しているか否かは、ステータス変数Created_tmp_disk_tablesで確認できます。Created_tmp_disk_tablesが増加している環境では、tmp_table_sizeおよびmax_heap_table_sizeの増加を検討します（両方とも増加させる必要があります）。tmp_table_sizeおよびmax_heap_table_sizeは、セッション単位で動的に変更可能です。

13.3.2 MySQLサーバー全体の設定値 thread_cache_sizeのチューニング

クライアントがサーバーに接続すると、コネクションスレッドが生成されます。コネクションスレッドは、コネクション切断時に毎回破棄せずキャッシュして使いまわす仕組みがあります。このキャッシュのサイズを設定しているのがthread_cache_sizeです。

thread_cache_sizeが足りているかどうかは、ステータス変数Threads_createdで確認できます。サーバー起動後にクライアントが接続するとコネクションスレッドが新規で作成されるため、Threads_createdに値が入っていること自体は問題ではありませんが、時間と共にThreads_createdの値が増加している場合はスレッドキャッシュが足りていないと判断できるため、thread_cache_sizeを増加させてチューニングします。

13.3.3 InnoDBストレージエンジンの設定値

InnoDBストレージエンジンの設定値について解説します。

innodb_buffer_pool_sizeのチューニング

innodb_buffer_pool_sizeには、InnoDBがデータとインデックスをキャッシュするメモリー領域のサイズを指定します。パフォーマンスを考慮すると、頻繁にアクセスするデータとインデックスはすべてメモリー上にキャッシュされることが望ましいため、可能な範囲でできる限り大きく設定するのがチューニングの定石となっています。ただし、実データ量以上に大きく設定してもメリットはありません。

キャッシュの空き状況や、どの程度キャッシュが有効活用できているかは、リスト6のようにSHOW ENGINE INNODB STATUSから確認できます。

リスト6　InnoDBのキャッシュヒット率の確認

```
mysql> SHOW ENGINE INNODB STATUS￥G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2016-06-30 09:09:32 0x7fae54db9700 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 7 seconds
```

```
<中略>
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 137428992
Dictionary memory allocated 350357
Buffer pool size    8192
Free buffers        7918
Database pages      274
Old database pages 0
Modified db pages   0
Pending reads       0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 240, created 34, written 36
0.86 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 956 / 1000, young-making rate 0 / 1000 not 0 /
1000
<中略>
----------------------------
END OF INNODB MONITOR OUTPUT
============================

1 row in set (0.00 sec)
```

ページサイズのデフォルトは16KBですから、上の例ではBuffer pool sizeの8192に16KBを乗じると、バッファプールのサイズが128MB（131,072KB）であることがわかります。同様に、Database pages、Free buffersから算出すると4,384KBだけ使用していて、126,688KBは空いていることがわかります。また、「Buffer pool hit rate 956 / 1000」という出力から、直近のキャッシュのヒット率が95.6%であったこともわかります。

SHOW ENGINE INNODB STATUSの出力は、直近n秒間のサマリーとなっているため、キャッシュヒット率の傾向を確認する場合は、1回の出力だけで判断せずに複数回出力を確認して傾向を読み取ります。冒頭に出力されている「Per second averages calculated from the last 7 seconds」から、直近何秒間のサマリーの情報が出力されているかを確認できます。

キャッシュに乗っているオブジェクトを詳細に確認できるINFORMATION_SCHEMA.INNODB_BUFFER_PAGEというテーブルもありますが、INFORMATION_SCHEMA.INNODB_BUFFER_PAGEの参照は大幅なパフォーマンス上のオーバーヘッドを発生させる可能性があるため、本番環境では基本的に参照しないでください。詳細は以下のリファレンスマニュアルを参照ください。

- **URL : MySQL 5.7 Reference Manual : 22.31.1 The INFORMATION_SCHEMA INNODB_BUFFER_PAGE Table**
  https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-page-table.html

innodb_log_file_sizeのチューニング

トランザクションをコミットした時に、InnoDBログファイルにトランザクション内容が記録されます。innodb_log_file_sizeには、InnoDBログファイルのサイズを設定します。InnoDBログファイルのサイズが大きくなると、そのぶんクラッシュリカバリにかかる時間は長くなりますが、ダーティページ（バッファプール上で変更されたが、まだディスク上に書き出されていないページ）をたくさんメモリー上に保持できるようになります。ダーティページをたくさん保持できるとディスクI/Oの削減につながるため、更新処理が多発する環境ではinnodb_log_file_sizeを拡張することでパフォーマンスの向上が見込めます。

InnoDBログファイルのサイズをinnodb_buffer_pool_size以上に拡張する必要はないため、一般的にはinnodb_buffer_pool_sizeの25%～100%の範囲でサイズを調整します。

なお、別途innodb_log_files_in_groupでログファイルの個数が設定されています。innodb_log_files_in_groupのデフォルト値は2であるため、デフォルト設定の場合、innodb_log_file_sizeはinnodb_buffer_pool_sizeの12.5%～50%の範囲でサイズを調整します。

InnoDBログファイルのサイズ変更は、以下の手順で行います。

1．MySQLサーバーを停止し、シャットダウン時にエラーが発生していないことを確認する
2．my.cnfを編集し、innodb_log_file_sizeを変更する（innodb_log_files_in_groupを増加してログファイルの個数を

増やすことも可能）

3. MySQLサーバーを起動する

なお、この手順はMySQL 5.6.8以降で実施可能です。MySQL 5.6.7以前で上の手順を実行すると、最悪の場合、データベースが破損することもあります。MySQL 5.6.7以前でのInnoDBログファイルのサイズ変更手順は、以下のリファレンスマニュアルを参照してください。

- **URL：MySQL 5.6 リファレンスマニュアル：14.5.7 InnoDB ログファイルの数またはサイズの変更、および InnoDB テーブルスペースのサイズの変更**
  https://dev.mysql.com/doc/refman/5.6/ja/innodb-data-log-reconfiguration.html

innodb_flush_methodのチューニング

innodb_flush_method=O_DIRECTに設定すると、データファイルへのアクセス時にファイルキャッシュを使用しなくなります。この設定は、Linux環境では多くの場合オーバーヘッドが低下してパフォーマンス向上につながります。

Column **MySQLのリリースサイクル**

MySQLの開発中のマイナーバージョンには、他のソフトウェアで見られるようなアルファ版やベータ版といったものはありませ

ん。MySQLではDevelopment Milestone Releaseというモデルで開発を進めています。省略形でDMRとも呼ばれます。

DMRのモデルは、MySQL 5.5の開発中に取り入れられました。MySQL 5.1までの開発モデルでは、いつまで経っても製品がリリースされないことがあったほか、リリースされたとしても品質に大きな問題があることがあり、MySQLの利用者に大きな不満がたまっていました。この反省を踏まえて、より安定したメジャーバージョン間でも一貫性のある開発を進めるために採用されたのがDMRです。

DMRの最大のポイントは、メインの開発ブランチは常に高い品質を保った安定したものを維持するということです。このブランチでは、デグレードが起きていないか確認するリグレッションテストを常に行い、何か問題が見つかればすぐに改修します。新しい機能を追加する際には、非常に多くの品質保証テスト（QAテスト）を行います。新しい機能の開発は独立したツリーで行われ、メインの開発ブランチでの変更点を取り込みながら進められます。開発が進むと品質保証テストとバグ修正を繰り返し、品質保証テストチームから合格が出た段階でメインの開発ブランチにマージされます。これによって新しい機能は品質保証テストを済ませてからマージされ、品質を保つことができるようになっています。

このようにして開発中のバージョンであるDMRが一部の品質が良くない機能に邪魔をされてうまく動作しないようなことを回避

し、利用者が早い段階から新しい機能を試すことが可能になりました。

　開発中の新機能に対して利用者のフィードバックをいち早く集めるために、品質保証テストが完了していないものをMySQL Labs（実験室）でリリースしています。このLabs版のソフトウェアは、本番環境での利用は非推奨とされています。実験室と名前は付いているものの、MySQL 5.7に導入されたマルチソースレプリケーションや実装を一新したオプティマイザなども、このLabs版で開発の早い段階から利用者のフィードバックを集めた上で、製品としてのリリースにつなげています。

13.4 演習

このレッスンではパフォーマンスチューニングの基礎について学習しました。ここではシステム変数、ステータス変数の参照方法について確認しましょう。

1. SHOW VARIABLESを利用して、以下の設定値を確認する

- sort_buffer_size
- thread_cache_size
- innodb_buffer_pool_size

ヒント：レッスン13の「13.2.1　ステータス変数」および「13.2.2　システム変数」の「SHOW VARIABLESコマンドによるシステム変数の確認」を参照

2. パフォーマンス・スキーマを参照して、1と同様の設定値を確認する

ヒント：レッスン13「13.2.2　システム変数」の「パフォーマンス・スキーマによるシステム変数の確認」を参照

3. SHOW STATUSを利用して、以下のSQLを実行した時のステータス変数の変化を確認する

```
SELECT * FROM world.City;
```

ヒント：レッスン13「13.2.1　ステータス変数」の「SHOW STATUSコマンドによるステータス変数の確認」を参照

4. mysqladminのextended-statusコマンドを使って、5秒ごとのステータス変数の変化を確認する

ヒント：レッスン13「13.2.1　ステータス変数」の「mysqladminのextended-statusコマンドによるステータス変数の確認」を参照

解説

1．SHOW VARIABLESによるシステム変数の確認方法は以下の通りです。ここでは、グローバル単位で設定されている設定値を確認しています。

```
mysql> SHOW GLOBAL VARIABLES LIKE 'sort_buffer_size';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| sort_buffer_size | 262144 |
+------------------+--------+
1 row in set, 1 warning (0.01 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'thread_cache_size';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| thread_cache_size | 9     |
+-------------------+-------+
1 row in set, 1 warning (0.01 sec)

mysql> SHOW GLOBAL VARIABLES LIKE
'innodb_buffer_pool_size';
+-------------------------+-----------+
| Variable_name           | Value     |
+-------------------------+-----------+
| innodb_buffer_pool_size | 134217728 |
+-------------------------+-----------+
```

```
1 row in set, 1 warning (0.01 sec)
```

2. パフォーマンス・スキーマによるシステム変数の確認方法は以下の通りです。ここでは、グローバル単位で設定されている設定値を確認しています。

```
mysql> SELECT * FROM performance_schema.global_variables WHERE VARIABLE_NAME IN ('sort_buffer_size','thread_cache_size','innodb_buffer_pool_size');
+-------------------------+----------------+
| VARIABLE_NAME           | VARIABLE_VALUE |
+-------------------------+----------------+
| innodb_buffer_pool_size | 134217728      |
| sort_buffer_size        | 262144         |
| thread_cache_size       | 9              |
+-------------------------+----------------+
3 rows in set, 1 warning (0.00 sec)
```

3. クエリー実行時のステータス変数の変化を確認する方法は以下の通りです。

```
mysql> FLUSH STATUS;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM world.City;
```

```
+------+----------------------------------+-------------+-----------
-----------+------------+
| ID    | Name                                | CountryCode |
District                | Population |
+------+----------------------------------+-------------+-----------
-----------+------------+
|      1 | Kabul                                | AFG           |
Kabol                   |      1780000 |
|      2 | Qandahar                             | AFG           |
Qandahar                |       237500 |
<<中略>>
| 4079 | Rafah                                | PSE           |
Rafah                   |        92020 |
+------+----------------------------------+-------------+-----------
-----------+------------+
4079 rows in set (0.02 sec)

mysql> SHOW STATUS;
+-------------------------------------------+---------------------
--------------------------+
|  Variable_name                                                    |
Value                                          |
+-------------------------------------------+---------------------
--------------------------+
|  Aborted_clients                                                  |
0                                              |
|  Aborted_connects                                                 |
0                                              |
```

```
|  Binlog_cache_disk_use                                    |
0                                                           |
|  Binlog_cache_use                                         |
0                                                           |
|  Binlog_stmt_cache_disk_use                               |
0                                                           |
<<中略>>
|  validate_password_dictionary_file_last_parsed  |  2016-10-
05 22:03:19                                          |
|        validate_password_dictionary_file_words_count        |
0                                                           |
+-------------------------------------------------+----------------------
----------------------------+
358 rows in set (0.00 sec)
```

4. ヒント部分の解説を参照してください。

レッスン

14

パフォーマンスチューニングに役立つ機能やコマンド

このレッスンではパフォーマンスチューニングに役立つ機能やコマンドについて学習します。

14.1 パフォーマンス・スキーマとsysスキーマ

「パフォーマンス・スキーマ」は、性能統計情報を分析するための仕組みです。MySQLサーバー内部の処理時間等を記録し、performance_schemaデータベース内のテーブルに蓄積しています。この機能はデフォルトで有効になっていますが、無効化したり、収集する情報の粒度をカスタマイズすることも可能です。

さらにMySQL 5.7では、パフォーマンス・スキーマをより便利に活用するためのビュー、プロシージャ、ファンクションがセットになった「sysスキーマ」が標準搭載されています。MySQL 5.5、5.6でもDB作成後にsysスキーマを追加でセットアップすることが可能ですが、MySQL 5.7ではDB作成時にsysスキーマが自動的に作成されるように仕様変更されています。

パフォーマンス・スキーマを参照すると、詳細に性能統計情報を確認することが可能です。ただし、情報量が非常に多いため、まずはsysスキーマのビューを参照して、性能統計情報を確認することをおすすめします。sysスキーマには、パフォーマンス・スキーマの情報を目的にあわせて見やすく集計したビューが多数あります。例えば、以下のビューがチューニングに役立ちます。

- **sys.statement_analysis**

  SQLごとに実行回数や実行時間（最大、平均、トータル）などを集計したビュー。デフォルトではトータル実行時間が長い順にソートされているが、ORDER BY句で他の列を指定することで、任意の項目でソートも可能

- **sys.innodb_lock_waits**

ロック待ちしているSQLの調査に役立つビュー。どのオブジェクトに対するロックが原因でどのSQLがどれくらいの時間待っているか、ロック待ちを発生させているコネクション／ロック待ちしているコネクションのコネクションID、ロック待ちを発生させているSQL／セッションを強制終了するためにはどんなコマンドを打てばいいか、などの情報が確認できる

- **sys.schema_unused_indexes**

  使用していないインデックスの調査に役立つビュー。レッスン13で解説したように、インデックスは検索処理のレスポンスタイムを向上できる可能性があるが、更新処理に対しては追加のオーバーヘッドが発生するため、使用していないインデックスは削除することが望ましい。このビューを参照すると使用していないインデックスの一覧が確認できるため、削除候補のインデックスの選定に役立つ。このビューを参照する時は、以下の点に注意する

  - DBを起動してから1回も使用していないインデックスが出力されるため、DB起動後の稼働期間を十分確保してからビューを参照する
  - DB起動後にアクセスがあったテーブルに付けられているインデックスが対象であるため、テーブル自体にアクセスがなければそのテーブルに付けられているインデックスは検出されない

パフォーマンス・スキーマ、sysスキーマから確認できる情報の詳細は、以下リファレンスマニュアルを参照してください。

- **MySQL 5.7 Reference Manual : 23.9 Performance Schema Table Descriptions**
  http://dev.mysql.com/doc/refman/5.7/en/performance-schema-table-descriptions.html
- **MySQL 5.7 Reference Manual : 24.4.3 sys Schema Views**
  http://dev.mysql.com/doc/refman/5.7/en/sys-schema-views.html

14.2 SQLチューニングに役立つ機能

SQLチューニングに役立つ機能を解説します。

14.2.1 スロークエリーログ

「スロークエリーログ」は、実行時間が指定した時間以上かかったクエリーを出力するログです。デフォルトでは出力されないため、システム変数を設定して出力します。スロークエリーログに関するシステム変数についてはレッスン3の「3.1.6　ログ」で解説しているので、そちらを参照してください。

スロークエリーログにクエリーが出力されるタイミングは、クエリーの実行が完了した後です。そのため、現在実行中の時間がかかっているクエリーについてはスロークエリーログでは調査できないことに注意してください。現在実行中の時間がかかっているクエリーを調査する場合は、後述するSHOW FULL PROCESSLISTを使用します。

14.2.2 mysqldumpslow

mysqldumpslowは、スロークエリーログの集計ツールです。スロークエリーログは新しいスロークエリーが発見されるたびに追記されます。mysqldumpslowを使えば、たくさんのクエリーが含まれているスロークエリーログを集計し、同じパターンのクエリーをまとめて実行回数、累積実行時間、合計実行時間を集計できるため、優先してチューニングすべきクエリーの特定などに役立ちます。

同じパターンのクエリーとは、WHERE句で指定した条件値のみが異なるクエリーのことです。例えば、以下のように実行するとスロークエリーログを集計し、累積実行時間が長いクエリーが上位に表示されます（リスト1）。

リスト1　mysqldumpslow使用方法

```
$ mysqldumpslow -s at ￥
```

mysqldumpslowのオプションの詳細は、以下のリファレンスマニュアルを参照してください。

- **URL : MySQL 5.7 Reference Manual : 5.6.8 mysqldumpslow – Summarize Slow Query Log Files**
  https://dev.mysql.com/doc/refman/5.7/en/mysqldumpslow.html

14.2.3 SHOW FULL PROCESSLIST

SHOW FULL PROCESSLISTは、mysqlコマンドラインクライアントで使用できるコマンドで、現在接続しているコネクションの状態を一覧表示できます。現在実行中のクエリーで時間がかかっているものがある場合はSHOW FULL PROCESSLISTを実行し、「Command: Query」と出力されているものから「Time」の値が大きなものを探すことで特定可能です。

14.2.4 EXPLAIN

EXPLAINは、SQLの実行計画を確認するためのコマンドです。MySQLコマンドラインクライアントでSQL文の前にEXPLAINというキーワードを付けることで、SQLの実行計画を確認できます（リスト2）。

リスト2　EXPLAINによる実行計画の確認例

```
mysql> EXPLAIN SELECT * FROM world.City WHERE ID=1;
+----+-------------+-------+------------+-------+---------------+---------
-+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key
| key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+---------
-+---------+-------+------+----------+-------+
|  1 | SIMPLE      | City  | NULL       | const | PRIMARY
| PRIMARY | 4       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+---------
-+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

実行計画とは、SQLを処理する時の内部的な処理手順です。SQLは「どのテーブルからどの列の情報を取得する」など処理の目的は定義していますが、RDBMS内部での処理手順については定義していません。MySQLでは、オプティマイザというモジュールが最も少ないコストで処理できると思われる処理手順を選択して、実行計画を作成します。ここでいうコストとは、IOリソース、CPUリソースの消

費量です。つまり、IOリソース、CPUリソースの消費量が最も少ないと思われる処理手順が実行計画として選択されます。

　実行計画の違いでわかりやすい例は、テーブルスキャンとインデックススキャンの違いです。リスト3のSQLを実行した場合、テーブルに対してインデックスを経由して「CountryCode='JPN'」の行にだけアクセスするか、インデックスを使わずにテーブルの全行にアクセスしてから後で「CountryCode='JPN'」のフィルタリングをするかは、オプティマイザが判断した実行計画に依存します。

リスト3　WHERE句を使ったSQLの例

```
mysql> SELECT * FROM world.City WHERE CountryCode='JPN';
```

　以下は、この実行計画を確認するためにEXPLAINを使用した例です。

リスト4　EXPLAINによる実行計画の確認例

```
mysql> EXPLAIN SELECT * FROM world.City WHERE CountryCode='JPN';
+----+-------------+-------+------------+------+---------------+---------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+---------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | City  | NULL       | ref  | CountryCode   | CountryCode | 3       | const |  248 |   100.00 | NULL  |
```

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

実行計画を確認すると、実際に使用するインデックスを表示するkey列にCountryCodeと表示されていることから、このクエリーはインデックスCountryCodeを使用してテーブルデータにアクセスすることがわかります。

また、テーブルへのアクセスタイプ（type列）にrefと表示されていることから、ユニークではないインデックスを使用した等価検索であることもわかります。

EXPLAINによって出力される情報の詳細は、次のリファレンスマニュアルを参照してください。

- **URL : MySQL 5.7 Reference Manual : 9.8.2 EXPLAIN Output Format**
  https://dev.mysql.com/doc/refman/5.7/en/explain-output.html

14.2.5 オプティマイザ・トレース

オプティマイザが実行計画を選択する過程で、どのように各処理手順のコストを見積もったかなど、オプティマイザの動きの詳細を確認できます。適切な実行計画が選択されなかった場合に、オプティマイザ・トレースを取得することで、なぜその実行計画が選択されたのかというオプティマイザの判断を詳細に調査できます。以下の手順でオプティマイザ・トレースが取得できます（リスト5）。

リスト5　オプティマイザ・トレース取得方法

```
mysql> SET optimizer_trace="enabled=on";
mysql> <<トレースを取得したいSQL文を実行>>
mysql>                    SELECT                    *                    FROM
INFORMATION_SCHEMA.OPTIMIZER_TRACE￥G
mysql> SET optimizer_trace="enabled=off";
```

14.2.6 ヒント

オプティマイザが選択する実行計画が適切ではない場合、ヒントを使用してオプティマイザに対して指示を出すことで、実行計画を適切なものに変更できる場合があります。MySQL 5.7では、以下の3種類のヒントが使用できます。

- ●インデックスヒント
- ●STRAIGHT_JOINヒント
- ●オプティマイザヒント（MySQL 5.7から追加されたヒント）

インデックスヒント

インデックスを適切に使用することはSQLチューニングにおいて非常に重要です。オプティマイザが適切なインデックスを選択しない場合、インデックスヒントを使うことでオプティマイザに対して特定のインデックスを使用する／使用しない、という指示を出せます。表1の通り、3種類のインデックスヒントが存在します。

表1 インデックスヒントの種類

種類	効果
USE INDEX	特定のインデックスを使用するように、オプティマイザに指示を出す（指定されたインデックスが使用できても、テーブルスキャンの方が効率的と判断すればテーブルスキャンを選択する）
FORCE INDEX	特定のインデックスを使用するように、オプティマイザに指示を出す（指定されたインデックスが使用できる場合は、必ずインデックススキャンを選択する）
IGNORE INDEX	特定のインデックスを使用しないように、オプティマイザに指示を出す

リスト6は、インデックスヒントの使用例です。EXPLAINによりインデックスヒントを使用することで実行計画が変化していることを確認しています。この例では、USE INDEXを使用してオプティマイザにインデックスContを使用するように指示を出しましたが、オプ

ティマイザはテーブルスキャンの方が効率的と判断したため、USE INDEXでは実行計画は変化しませんでした。しかし、FORCE INDEXを使用することでpossible_keysがContに変化し、インデックスContを使用する実行計画に変更できたことが確認できます。

リスト6　インデックスヒントの使用例

```
mysql> EXPLAIN SELECT * FROM world.Country WHERE Continent='Africa' OR Continent='Asia';
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | Country | NULL       | ALL  | Cont          | NULL | NULL    | NULL |  239 |    45.61 | Using where |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql> EXPLAIN SELECT * FROM world.Country USE INDEX(Cont) WHERE Continent='Africa' OR Continent='Asia';
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
```

```
+----+-------------+---------+------------+------+---------------+------
+---------+------+------+----------+-------------+
|    1  |  SIMPLE           |  Country  |  NULL             |  ALL     |
Cont             | NULL | NULL      | NULL | 239   | 45.61      | Using
where |
+----+-------------+---------+------------+------+---------------+------
+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)

mysql>  EXPLAIN  SELECT  *  FROM  world.Country  FORCE
INDEX(Cont) WHERE Continent='Africa' OR Continent='Asia';
+----+-------------+---------+------------+-------+---------------+------
+---------+------+------+----------+-----------------------+
| id | select_type | table     | partitions | type    | possible_keys |
key    | key_len | ref    | rows | filtered | Extra                      |
+----+-------------+---------+------------+-------+---------------+------
+---------+------+------+----------+-----------------------+
|    1  |  SIMPLE           |  Country  |  NULL             |  range   |
Cont             | Cont |         1 | NULL | 109   | 100.00     | Using
index condition |
+----+-------------+---------+------------+-------+---------------+------
+---------+------+------+----------+-----------------------+
1 row in set, 1 warning (0.00 sec)
```

　また、それぞれのインデックスヒントは、FOR句を使うことでWHERE句以外のJOIN、ORDER BY、GROUP BYでインデックスを使用

するように指定することもできます。リスト7はインデックスヒントの構文です。

リスト7　インデックスヒントの構文

```
USE INDEX [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
FORCE INDEX [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
IGNORE INDEX [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
index_name [, index_name] ...
```

　インデックスヒントの詳細は、次のマニュアルを参照してください。

- **URL：MySQL 5.7 Reference Manual：9.9.4 Index Hints**
  https://dev.mysql.com/doc/refman/5.7/en/index-hints.html

STRAIGHT_JOINヒント

　STRAIGHT_JOINヒントは、テーブルをJOINする時にJOINの順番を指定できるヒントです。MySQLがJOINに使用するアルゴリズムはNested Loop JOINとその改良系ですが、Nested Loop JOINでは、基本的に先に読み取る表（駆動表）が後に読み取る表（内部表）よりも小さい方が効率的に結合できます。また、より絞り込みができるテーブルからJOINした方が効率が良いため、3テーブル以上JOINする

時は、基本的に結果セットが少量のテーブルからJOINする方が効率が良くなります。

　オプティマイザが選択したJOINの順番が適切ではない場合、STRAIGHT_JOINヒントを使用することでJOINの順番を明示的に指定できます。STRAIGHT_JOINヒントの詳細は、次のマニュアルを参照してください。

- **URL：MySQL 5.7 Reference Manual：14.2.9 SELECT Syntax**
  https://dev.mysql.com/doc/refman/5.7/en/select.html

　オプティマイザが選択したJOINの順番は、後述のVisual EXPLAINを使用すると簡単に判断できます。また、MySQL 5.7ではオプティマイザが改良され、JOINの順番を変えることによるコストの違いを見積もる精度が向上しているため、以前のバージョンよりもJOIN順番を適切に判断できるようになっています。

オプティマイザヒント

　オプティマイザヒントは、MySQL 5.7で追加された新しいヒント構文です。オプティマイザの詳細な動作を制御するために、システム変数optimizer_switchを変更するチューニング手法がありますが、オプティマイザヒントを使用すると、optimizer_switchと同様の制御をoptimizer_switchよりも細かい粒度で指定できます。例えば、サブクエリー部分だけが影響を受けるように変更を反映するなどです。

　オプティマイザヒントを使ったチューニングは、インデックスヒント、STRAIGHT_JOINヒントを使ったチューニングよりも高度なチ

ューニングとなります。ヒントを使ってSQLをチューニングする際は、まずインデックスヒント、STRAIGHT_JOINヒントを使用したチューニングを行い、それ以上にチューニングする必要がある場合にオプティマイザヒントの使用を検討することをおすすめします。

表2は、オプティマイザヒントの一覧です。MAX_EXECUTION_TIMEとQB_NAMEを除き、オプティマイザが選択する最適化アルゴリズムを指定したり、最適化アルゴリズムの有効／無効を切り替えたりするヒントになっています。QB_NAMEは、ほかのヒントの有効範囲にサブクエリーを指定するためのヒントです。サブクエリーに任意の名前を付けることで、他のヒントからそのサブクエリーを指定できます。また、MAX_EXECUTION_TIMEは、クエリーの実行時間にタイムアウト時間を設定するためのヒントです。

表2 オプティマイザヒント一覧

ヒント名	説明	スコープ
BKA, NO_BKA	BKA(Batched Key Access) join の有効／無効	クエリーブロック , テーブル
BNL, NO_BNL	BNL(Block Nested-Loop) join の有効／無効	クエリーブロック , テーブル
MAX_EXECUTION_TIME	クエリーの実行時間制限	グローバル
MRR, NO_MRR	MRR(Multi-Range Read) の有効／無効	テーブル , インデックス
NO_ICP	ICP(Index Condition Pushdown) の無効	テーブル , インデックス
NO_RANGE_OPTIMIZATION	インデックスレンジスキャン、インデックスマージ、ルースインデックススキャン (Using index for group-by) の無効	テーブル , インデックス
QB_NAME	クエリーブロック (1 つ 1 つのサブクエリー) に名前を付ける。QB_NAME ヒントで名付けた名前を、他のヒント (BKA など) で指定できる	クエリーブロック
SEMIJOIN, NO_SEMIJOIN	純結合変換による最適化の有効／無効	クエリーブロック
SUBQUERY	サブクエリーの最適化方法を指定 (INTOEXISTS or MATERIALIZATION)	クエリーブロック

オプティマイザヒントの詳細は、次のマニュアルを参照してください。

- **URL : MySQL 5.7 Reference Manual : 9.9.3 Optimizer Hints**
  https://dev.mysql.com/doc/refman/5.7/en/optimizer-hints.html

14.2.7 MySQL Workbenchでのチューニング

レッスン5で紹介したMySQL Workbenchには、チューニングに役立つ機能も搭載されています。

クライアントコネクションの一覧を確認

MySQL Workbenchの管理メニューには、クライアントコネクションの一覧を確認できるメニューが含まれています。図1のように「MANAGEMENT」の「Client Connections」をクリックすることで、クライアントコネクションの一覧を確認できます。

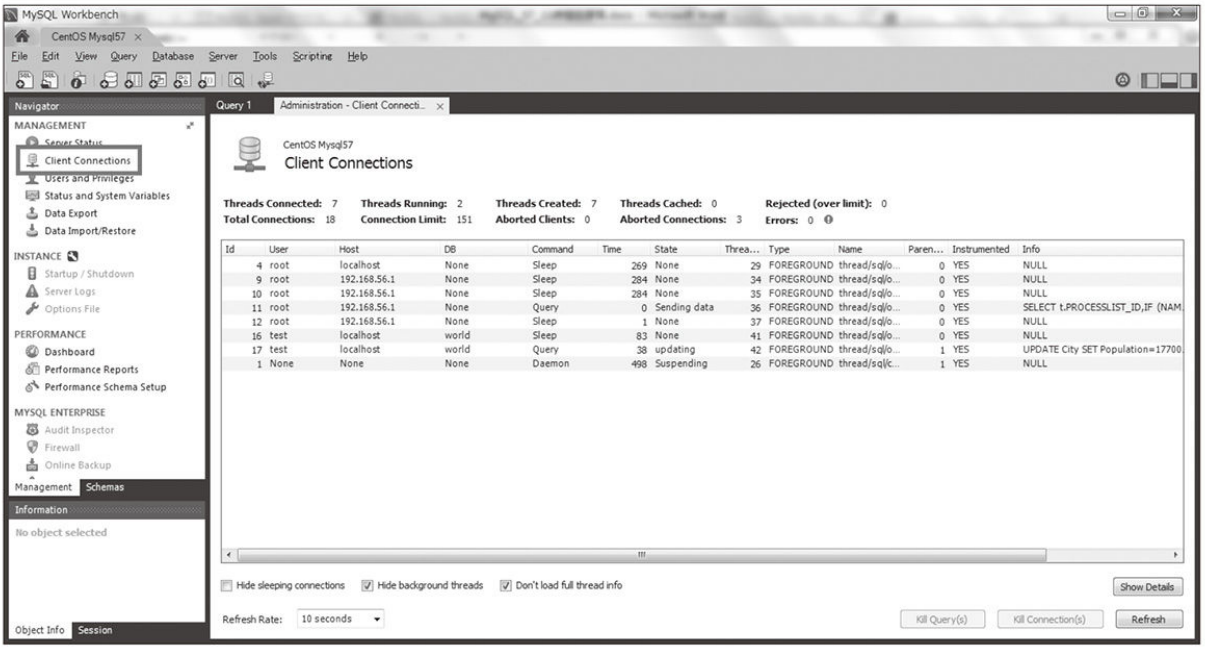


図1 MySQL Workbenchの Client Connections

確認できる情報はSHOW FULL PROCESSLISTと同等ですが、出力を任意の項目でソートできる、スリープしているコネクションの情報を非表示にできる、「Show Details」ボタンをクリックしてコネクシ

ョンの詳細情報を表示できるなど、MySQL Workbenchならではの利点もあります。

システム変数、ステータス変数の確認

図2のように「MANAGEMENT」の「Status and System Variables」をクリックすることで、システム変数、ステータス変数を確認できます。

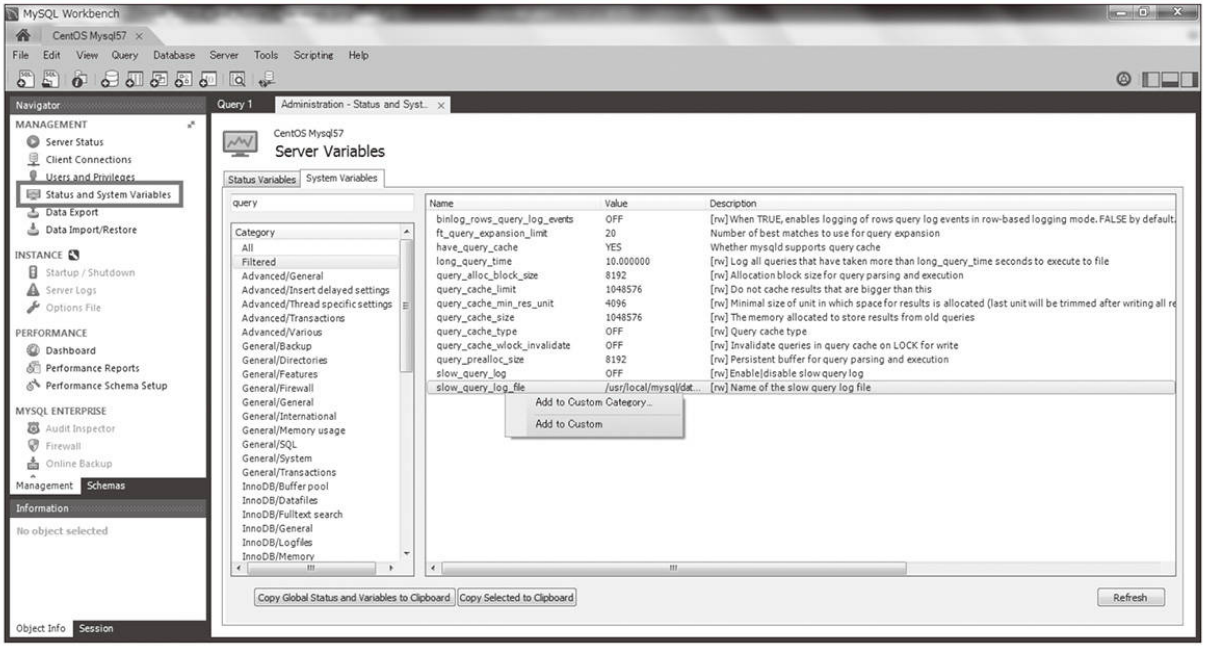


図2 MySQL Workbenchの Status and System Variables

ウィンドウに変数名の一部を入力することで、中間一致での絞り込みが可能です。また、変数名を右クリックして「Add to Custom」をクリックすると、任意の変数をCustomカテゴリに集約して表示することも可能です。

Visual EXPLAIN

MySQL WorkbenchのSQLエディタからSQLを実行すると、EXPLAINの情報も続けて確認できます。SQLを実行せずに、EXPLAINの情報のみを確認することも可能です。

MySQL WorkbenchでEXPLAINを確認すると、デフォルトでは実行計画を図示化したVisual EXPLAINが参照できますが、MySQLコマンドラインクライアントと同様のテキスト形式のEXPLAINも確認可能です。また、Visual EXPLAINはJSON形式のEXPLAINを元に作成されていますが、JSON形式のEXPLAINそのものを確認することも可能です。

SQL実行後にEXPLAINを確認する場合は、SQL実行後（図3）に「Execution Plan」のアイコン（図4右下）をクリックします。

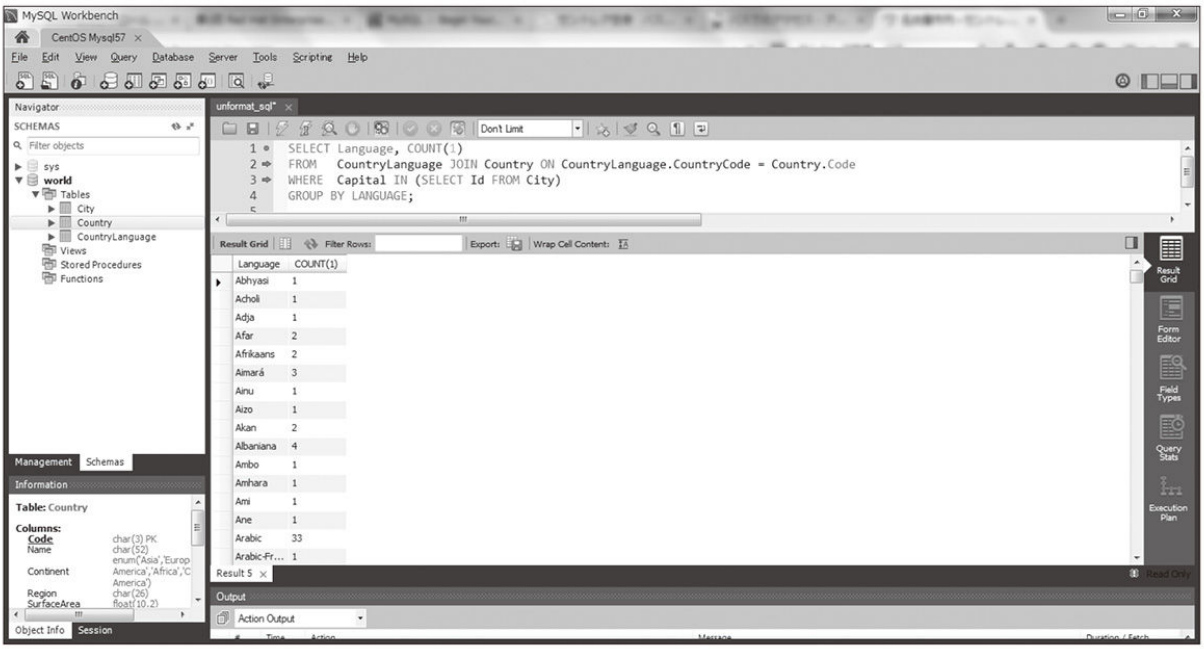


図 3 MySQL Workbench で SQL を実行した画面

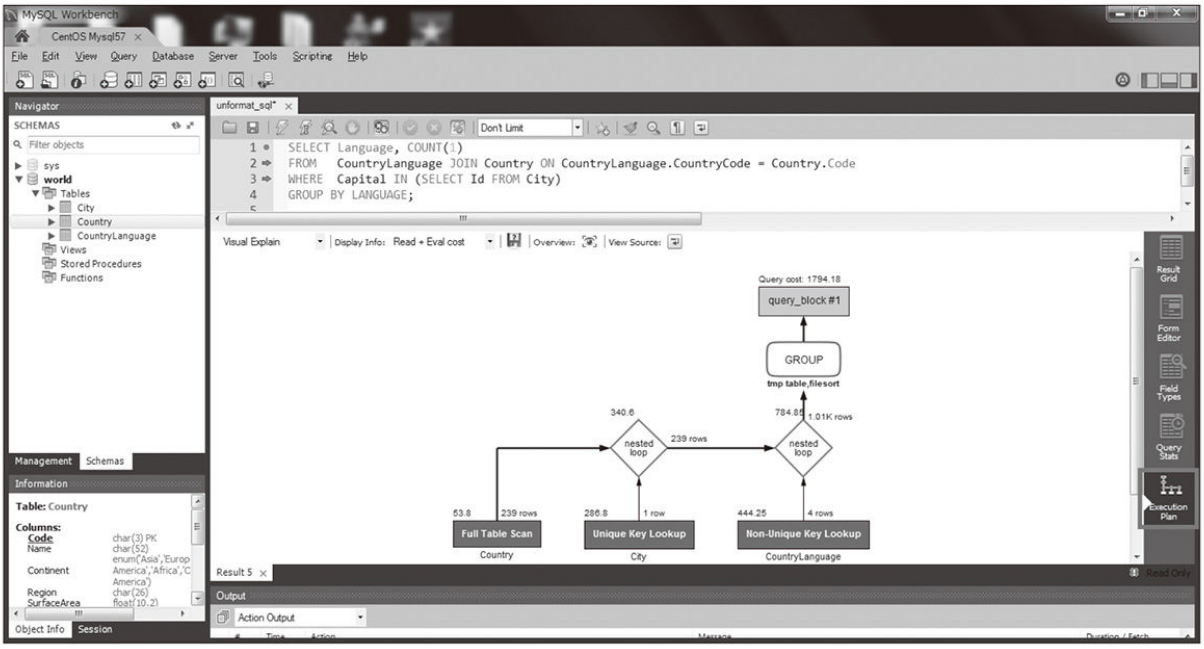


図 4　MySQL Workbench で確認できる Visual EXPLAIN

Visual EXPLAINでは、通常のEXPLAINに比べて以下の利点があります。

- ●オブジェクトへのアクセスパターンが色で識別できる[※ 1]
- ●JOINの順番がフローチャートのような図で確認できる
- ●コストの情報など、JSON形式のEXPLAINに含まれる追加情報も確認できる

SQLを実行せずにEXPLAINのみを確認する場合は、図5の中央上部にある虫眼鏡のボタンをクリックします。

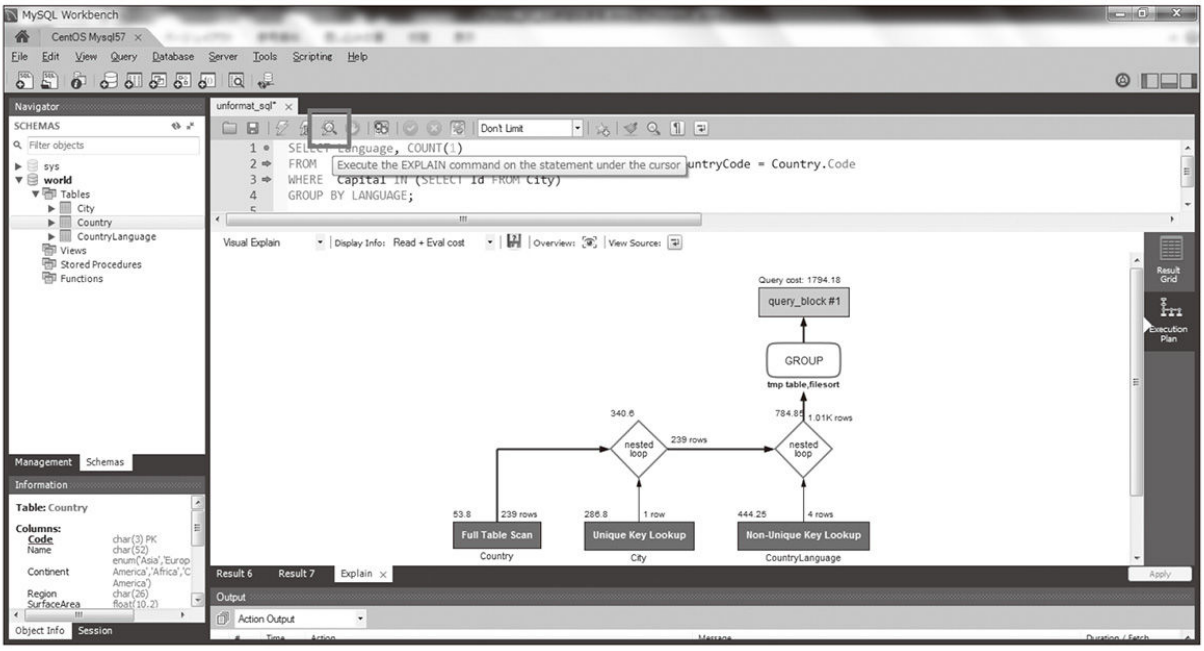


図 5　MySQL Workbench で EXPLAIN のみを確認する場合

Query Statistics

MySQL WorkbenchのSQLエディタからSQLを実行すると、SQL実行時の各種統計情報もあわせて確認できます。

SQL実行時の各種統計情報を確認する場合は、SQL実行後に「Query Stats」のアイコン（図6右）をクリックします。

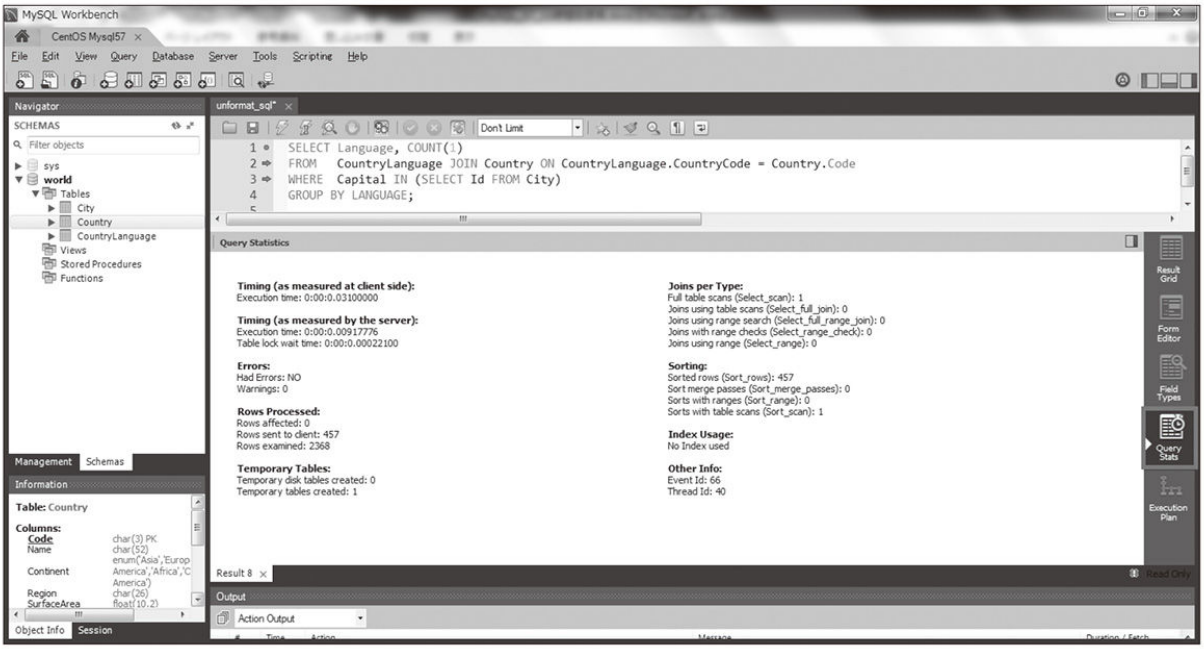


図6 MySQL Workbench で確認できる Query Statistics

Query Statisticsでは、SQLの実行時間だけでなく、SQLチューニングにおいて重要なステータス変数の変化、エラー／警告の有無などを一覧で確認できます。

パフォーマンス・ダッシュボード、パフォーマンス・レポート

MySQL Workbenchの管理メニューには、パフォーマンス・スキーマの情報をベースにしたパフォーマンス・ダッシュボードや、sysスキーマの情報をベースにしたパフォーマンス・レポート、パフォーマンス・スキーマの設定を変更するためのGUIが含まれています。

図7～9のように「PERFORMANCE」の下にある各ボタンをクリックすることで、これらを利用できます。

図 7　MySQL Workbench のパフォーマンス・ダッシュボード

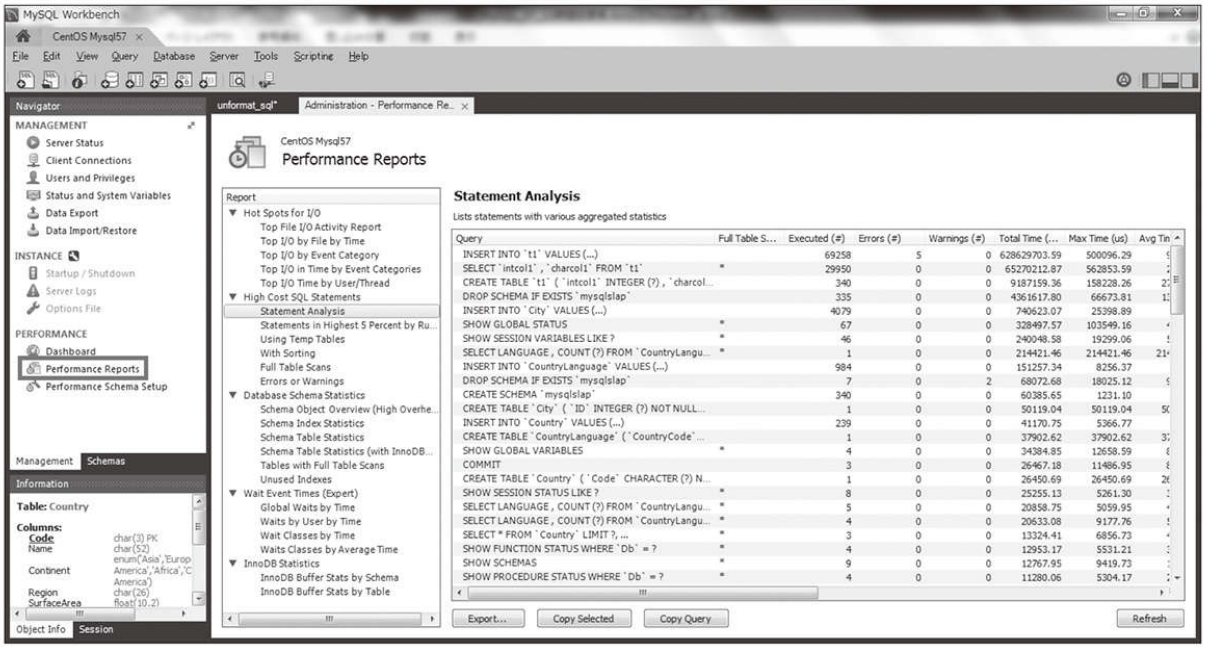


図 8　MySQL Workbench のパフォーマンス・レポート

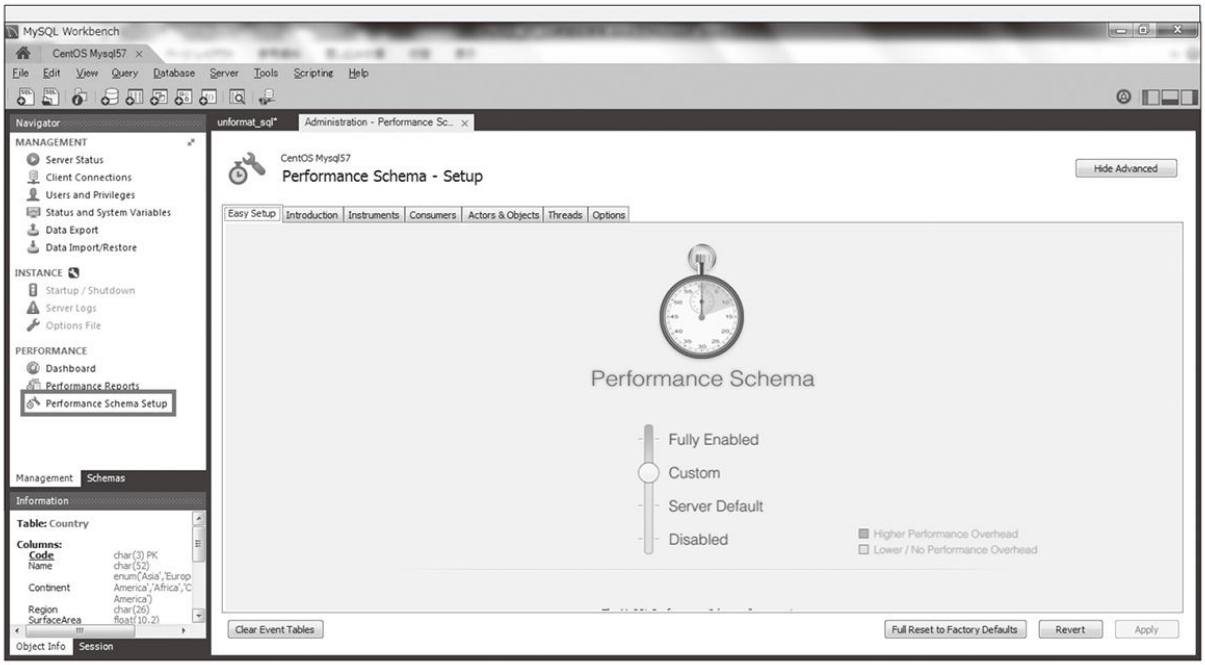


図 9　MySQL Workbench から制御できるパフォーマンス・スキーマの設定

14.2.8 MySQL Enterprise Monitor の Query Analyzer

MySQL Enterprise Editionでは、MySQL Enterprise Monitorという監視機能が使えます。その中のQuery Analyzerは、パフォーマンスが悪いSQLを簡単に調査できる機能です。Query Analyzerでは、MySQLサーバーの稼働状況の各種グラフを確認しながら、気になった時間帯をグラフ上で選択することで、その時間帯に実行されたSQLのみを絞り込んで確認できます。実行計画、実行時間など、特定されたSQLの詳細情報も確認できるため、チューニングすべきSQLの特定と稼働状況の把握を短時間で行えます（図10）。

MySQL Enterprise Editionの契約にはコンサルティング・サポートというサポートも含まれており、MySQLサーバー全体のチューニングや、SQLチューニングに関する問合せも対応可能となっています。そのため、Query Analyzerで特定したSQLをうまくチューニングできない場合は、サポートに問合せてチューニングに関するアドバイスを受けることもできます。

コンサルティング・サポートの詳細は、以下のサイトを参照ください。

- **URL：MySQL コンサルティング・サポート**
  https://www-jp.mysql.com/support/consultative.html

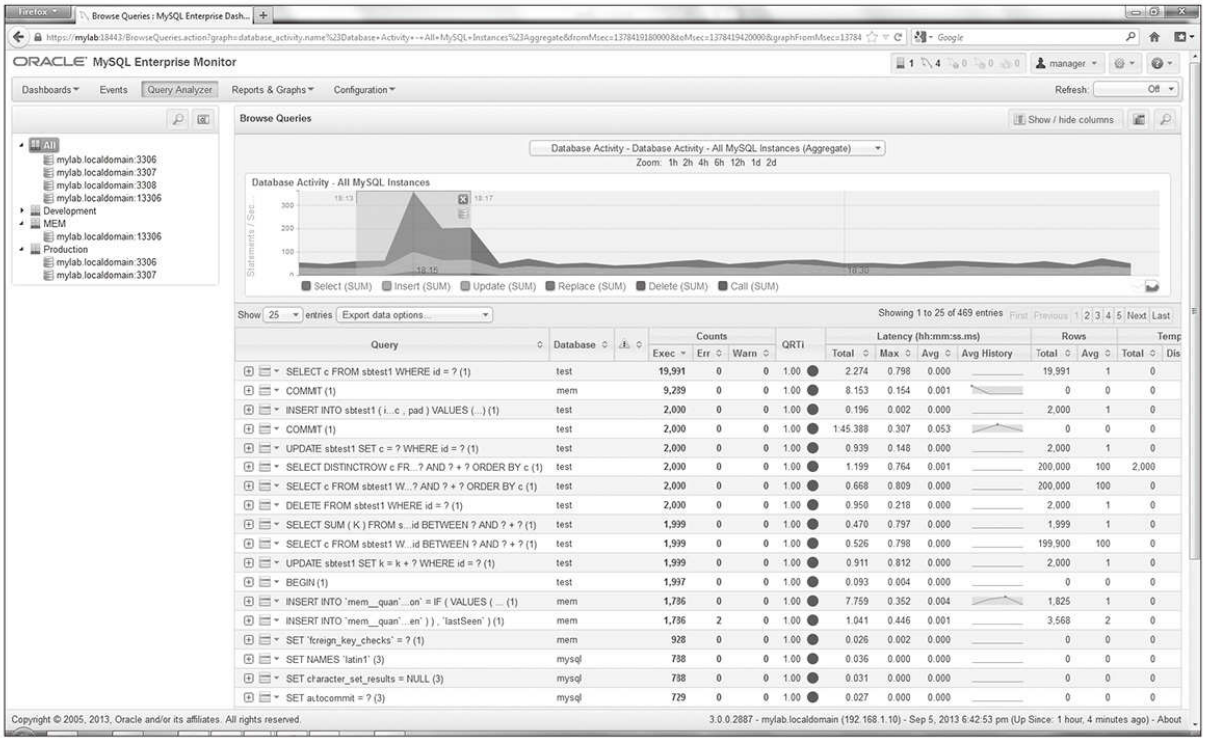


図 10 MySQL Query Analyzer

Column　不可視索引（INVISIBLE INDEX）

現在開発中の次バージョンMySQL 8.0では、不可視索引（INVISIBLE INDEX）という機能が実装される予定です。この機能もロールと同様に、原稿執筆時点で公開されているMySQL 8.0.0 DMR（開発途上版）でも、既に実装されています。

不可視索引とは、オプティマイザから見えない（インビジブルな）索引（インデックス）のことです。既存のインデックスをインビジブルな状態に変更したり、インビジブルな状態から通常の状態（ビジブル）に戻したりできます。また、最初にインデック

スを作成する時から、インビジブルな状態でインデックスを作成することもできます。

　このレッスンでは、sysスキーマのsys.schema_unused_indexesから使用していないインデックスが確認できることを解説しましたが、既存のインデックスの削除にはパフォーマンス劣化のリスクもともないます。また、一旦削除したインデックスを再作成する場合、テーブルのデータ量が多ければインデックス作成にも時間がかかってしまいます。

　そこで、不可視索引が役立ちます。不可視索引を使用すれば、いきなりインデックスを削除するのではなく、まずはインデックスをインビジブルにすることで、インデックスを削除した場合のパフォーマンス影響を確認できます。そして、パフォーマンスに悪影響を及ぼさないことを確認してからインデックスを削除できます。また、インデックスをインビジブルにしたことで万が一パフォーマンスが劣化した場合は、インデックスを通常の状態（ビジブル）に戻せば、素早く元の状態に戻せます。

　このように、不可視索引を使えばより安全にインデックスを削除できるのです。

14.3 演習

このレッスンでは、MySQLのパフォーマンスチューニングに役立つ機能やコマンドについて学習しました。ここではsysスキーマ、スロークエリーログ、MySQL Workbenchの活用方法について確認しましょう。

1. sysスキーマのsys.statement_analysisビューを参照し、現在の環境でトータル実行時間が一番長いSQLを特定する

ヒント：レッスン14の「14.1　パフォーマンス・スキーマとsysスキーマ」を参照

2. スロークエリーログの出力を有効にした状態でlong_query_time 以上に時間がかかるSQLを実行し、そのSQLがスロークエリーログに出力されることを確認する

ヒント：レッスン14の「14.2.1　スロークエリーログ」およびレッスン3「3.1.6　ログ」の「スロークエリーログ」を参照

3. MySQL Workbenchを使用して、クライアントコネクションの一覧を表示する

ヒント：レッスン14「14.2.7　MySQL Workbench でのチューニング」の「クライアントコネクションの一覧を確認」を参照

4. MySQL Workbenchを使用して、以下SQLのVisual Explainを確認する

```
SELECT City.name as capital, CountryLanguage.Language
    FROM City JOIN Country ON City.CountryCode = Country.Code
        JOIN CountryLanguage ON Country.Code = CountryLanguage.CountryCode
    WHERE City.ID = Country.capital;
```

ヒント：レッスン14「14.2.7　MySQL Workbench でのチューニング」の「Visual EXPLAIN」を参照

5. MySQL Workbenchを使用して、パフォーマンス・レポートからトータル実行時間が一番長いSQLを特定する

ヒント：レッスン14「14.2.7　MySQL Workbench でのチューニング」の「パフォーマンス・ダッシュボード、パフォーマンス・レポート」を参照

解説

1. 以下のSQLを実行することで、トータル実行時間が一番長いSQLを特定できます。

```
mysql> SELECT * FROM sys.statement_analysis ORDER BY total_latency DESC LIMIT 1￥G
```

2. ヒント部分の解説を参照してください。

3. ヒント部分の解説を参照してください。

4. ヒント部分の解説を参照してください。

5. 図8のパフォーマンス・レポートから「High Cost SQL Statements」の「Statement Analysis」を選択します。その後、出力の「Total Time」列をクリックしてソートすることで、トータル実行時間が一番長いSQLを特定できます。

※1 詳細はMySQL Workbench Manual：7.3 Visual Explain Planの「Table 7.1 Visual Explain Diagram Information」を参照してください。
http://dev.mysql.com/doc/workbench/en/wb-performance-explain.html

レッスン

15

Oracle MySQL Cloud Service

このレッスンでは、Oracle MySQL Cloud Service
について学習します。

15.1 Oracle MySQL Cloud Serviceとは

Oracle MySQL Cloud Serviceは、オラクルが提供するクラウドサービスである「Oracle Public Cloud」のDBaaSの1つです。Oracle MySQL Cloud Serviceは、MySQL Enterprise Editionをベースとしています。数クリックで迅速にMySQLサーバーを起動でき、管理ツールやOracle Public Cloudの各種サービスとも連携ができるようになっています。

図1 Oracle MySQL Cloud Service 紹介ページ

- **URL：Oracle MySQL Cloud Service**
  http://cloud.oracle.com/ja_JP/mysql

MySQLをベースとした他のDBaaSとの大きな違いは、MySQL Enterprise Editionが提供する監視ツール「MySQL Enterprise Monitor」をはじめ、セキュリティ拡張機能の「MySQL Enterprise Firewall」や「MySQL Enterprise Audit」などが利用できることに加えて、MySQLの開発元からサポートサービスを受けることが可能となっていることが

挙げられます。MySQL Enterprise Editionに含まれるセキュリティ機能もすべて利用できるので、ビジネスにとって重要なデータの格納先としての活用が期待されます。

　また、MySQLサーバーの機能に制限を加えていないため、オンプレミスとクラウドで同じソフトウェアが利用できます。レプリケーションを含めたすべての機能が利用可能となっており、同時にOracle Cloud Serviceが提供する拡張可能なノードやストレージを利用できます。またMySQL Enterprise Backupによってバックアップしたデータをクラウドとオンプレミスの間で移動できるため、開発フェーズと本番フェーズで最適な環境を選択することも可能となっています。Oracle MySQL Cloud Serviceでは、MySQLサーバーが稼働するサーバーへSSHでのアクセスが可能となっており、MySQL Enterprise Backupによるバックアップファイルの取得、GUIツールのMySQL WorkbenchからのSSH経由での利用も可能です。Oracle MySQL Cloud ServiceのMySQLインスタンスとオンプレミス環境のMySQLサーバーとの間でのレプリケーション構成も可能で、物理バックアップファイルへのアクセスとあわせて、クラウドとオンプレミスの相互の移行やハイブリッド型の構成が簡単に利用できる点が大きなメリットとなります。

15.2 Oracle MySQL Cloud Serviceの概要

Oracle MySQL Cloud Serviceは、基本30日間の無料トライアルが用意されています。ここではトライアルで機能の概要を確認していきましょう。

まず、Oracle MySQL Cloud Service紹介ページ（図1）の右上にある「無料トライアル」からトライアル一覧のページに移動します。Oracle MySQL Cloud Serviceは、Oracle Cloud PaaSの1つなので、トライアル一覧ページの「Oracle Cloud PaaSおよびIaaS」の表示の下に表示される「試してみる >」のリンクを選びます（図2）。

図2 Oracle Cloud トライアル

Oracle Cloudトライアルのサインアップページから、連絡先や携帯電話のSMSを利用した確認コードの入力などを行い、トライアルへの申込を済ませます（図3）。申込後、サービスの準備が完了するとアクセス用のアイデンティティ・ドメイン、ユーザ名と初期パスワードがメールで送付されます。

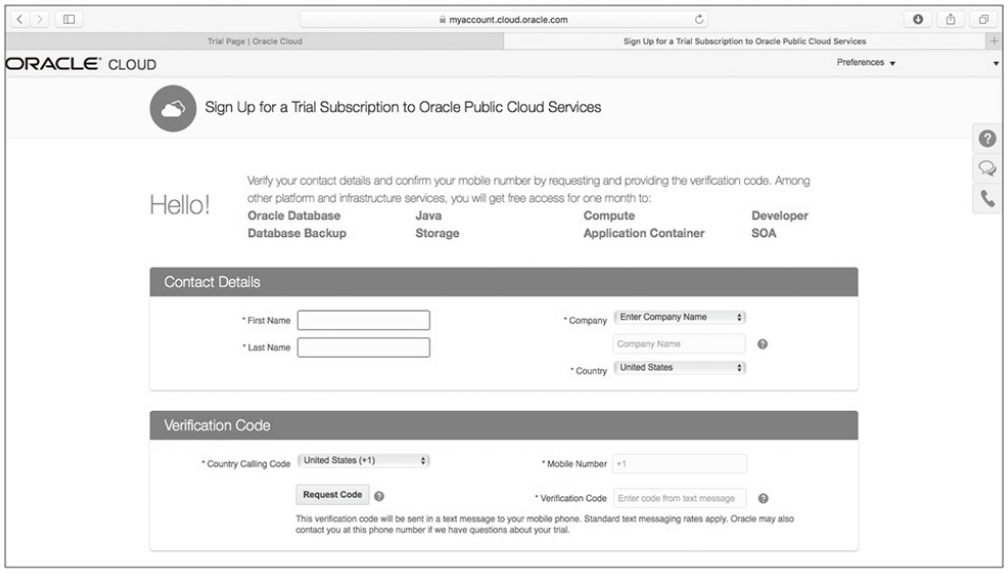


図3　Oracle Cloud トライアルのサインアップページ

メールに記載されたマイサービスURLまたはサインインページから、「マイ・サービス」に進みます（図4）。

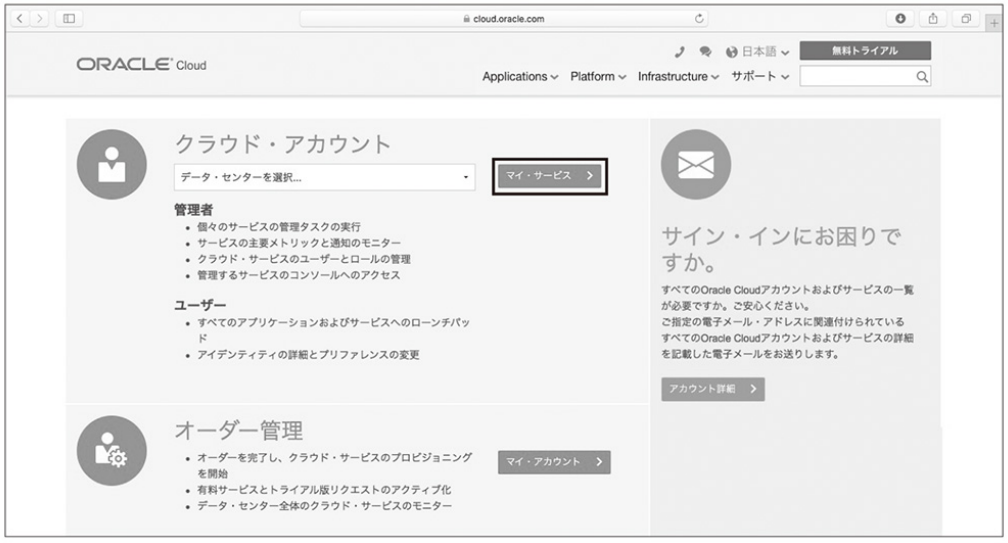


図4　Oracle Cloud サインインページ (https://cloud.oracle.com/ja_JP/sign-in)

ここでMySQL Cloud Serviceを選択すると（図5）、サービスの作成や既存のサービスのサマリー（図6）、さらに個別のサービスを管理

できます（図7）。

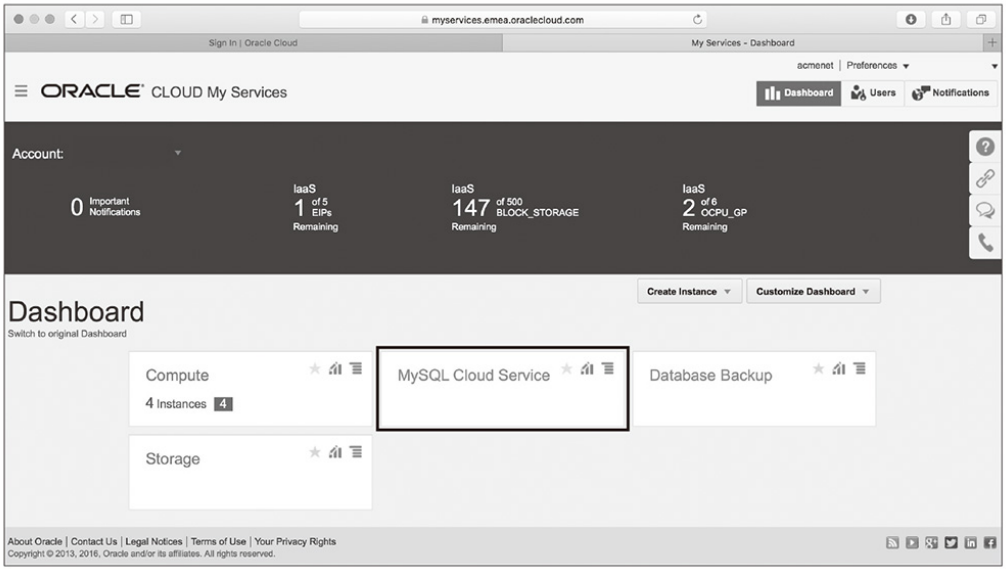


図5　Oracle Cloud マイ・サービス ダッシュボード

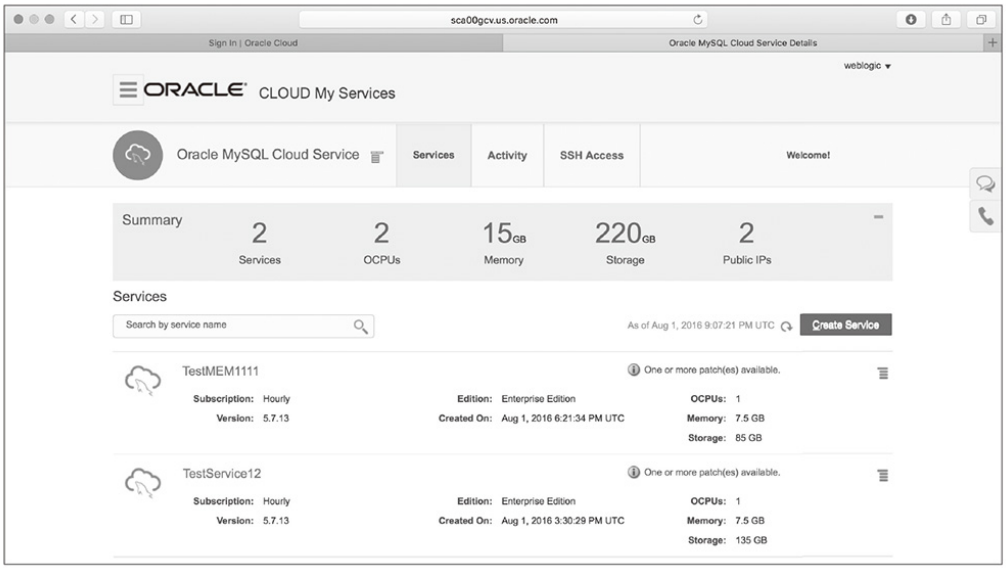


図6　サービスのサマリー

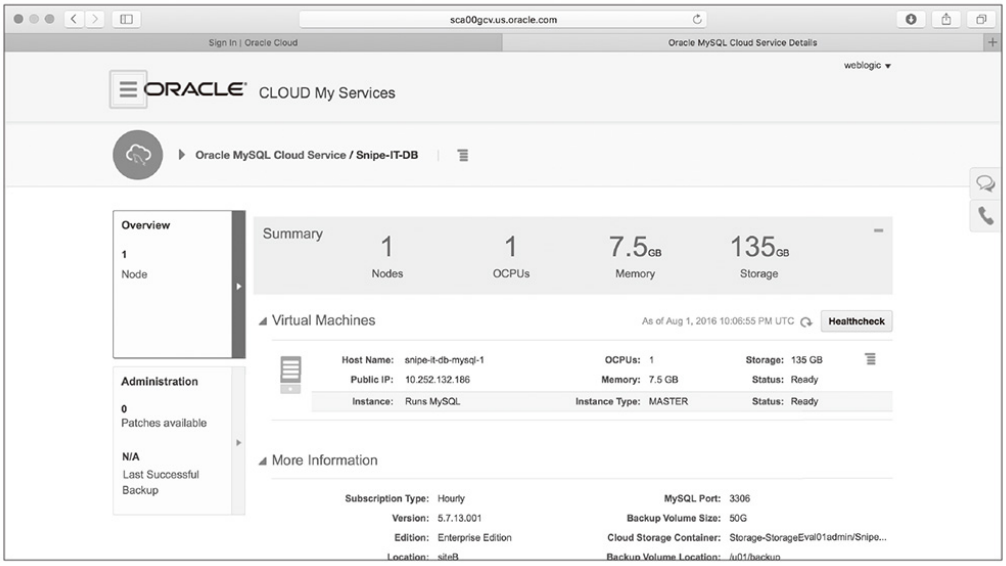


図 7　個別サービス監視管理画面

Oracle MySQL Cloud ServiceはSSHでのアクセスが可能です。以下の例ではSSHで接続し（図8）、MySQLクライアントシェルプログラムでMySQLサーバに接続しています（図9）。

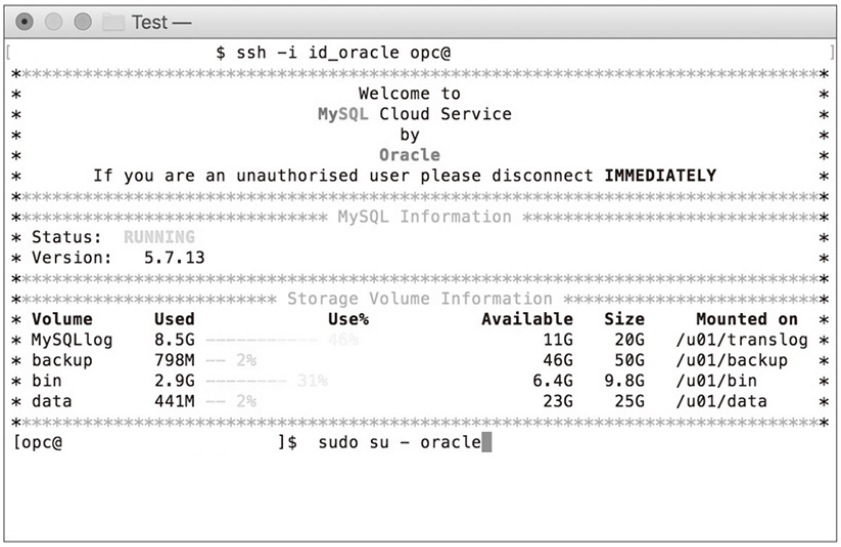


図 8　SSH 接続後の画面

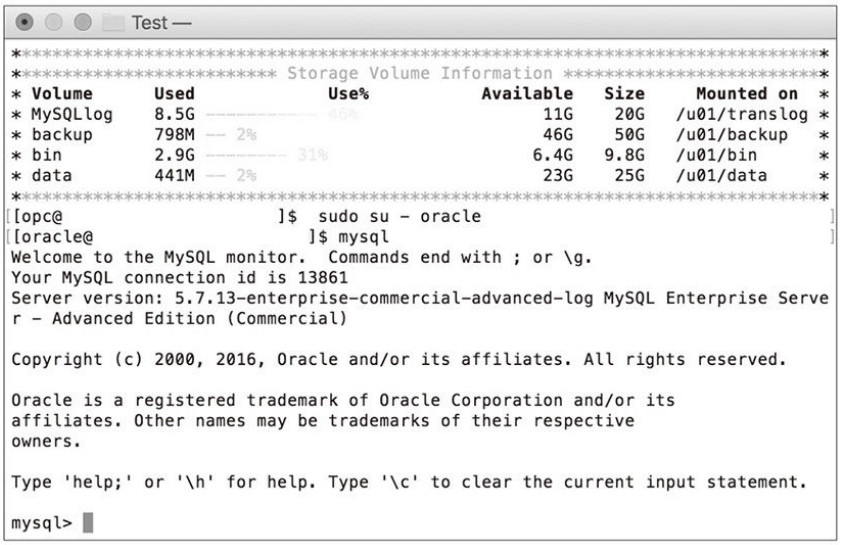


図 9　SSH 接続後に MySQL クライアントシェルプログラムにて MySQL サーバに接続

Oracle MySQL Cloud Serviceでは、MySQL Enterprise Backupで自動的にMySQLサーバーのオンラインバックアップが可能です（図10）。管理コンソールからバックアップの即時実行、スケジュール設定や進捗監視ができるようになっています。バックアップデータの保存期間もあわせて設定できます。指定した日時へのポイントインタイムリカバリも可能です。

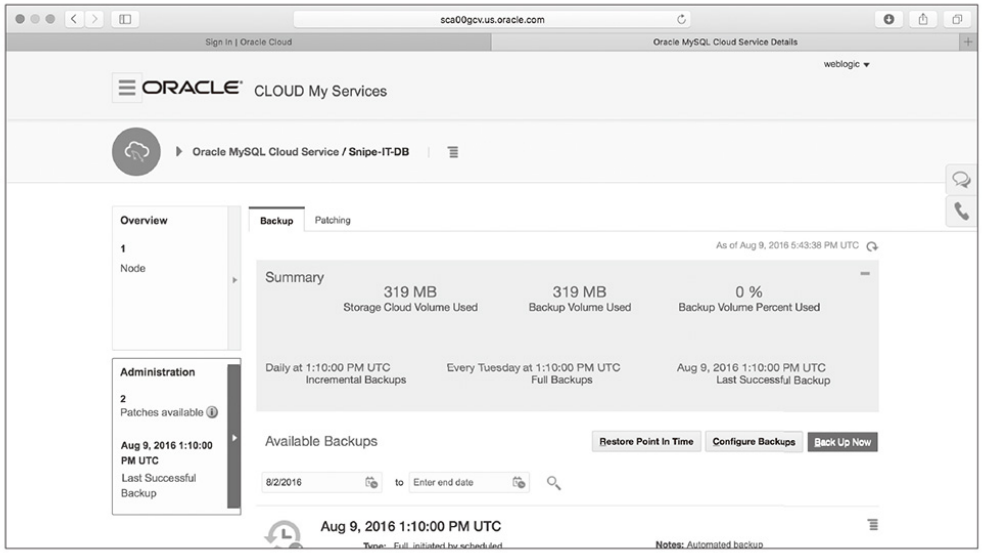


図 10　バックアップ設定画面

MySQLサーバーへのパッチ適用、また以前のバージョンへのロールバックも管理コンソールから実行できます（図11）。

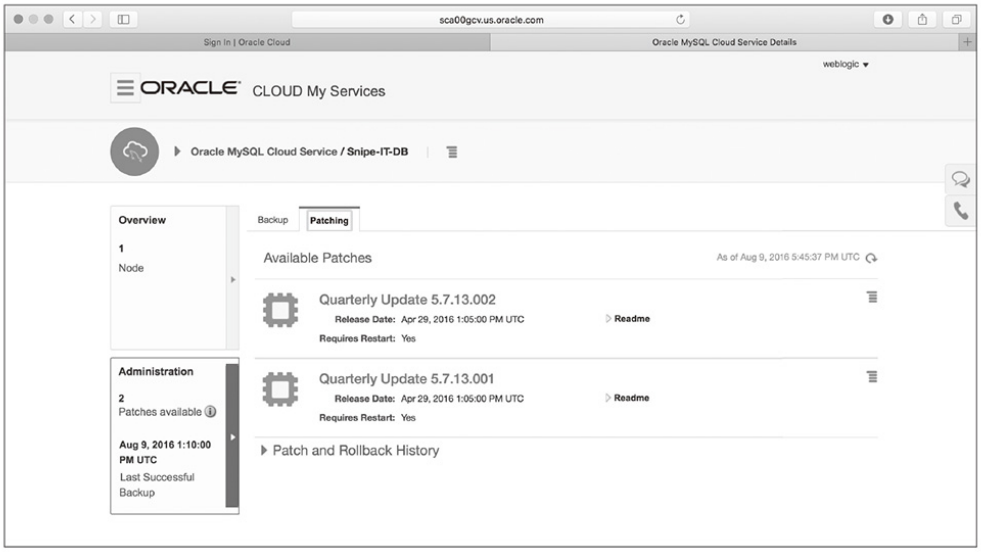


図 11　パッチ適用画面

クラウドに求められる"エラスティック"なリソースの拡張も、もちろんOracle MySQL Cloud Serviceに用意されています。CPUのスケールアップやスケールダウン、ストレージの拡張や縮小が可能です（図12）。MySQL Enterprise Thread Poolも標準で構成されているため、同時アクセス数が大きい環境への対応も考慮されています。

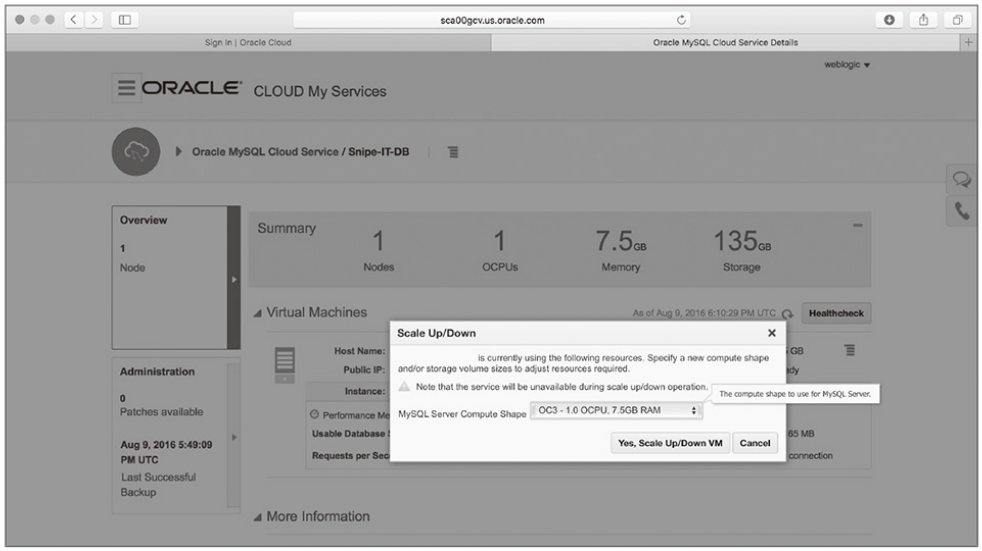


図 12　CPU リソース設定変更画面

15.3 Oracle MySQL Cloud Serviceの独自性

運用管理については、MySQL EnterpriseによるMySQLサーバーの詳細な監視から、Oracle Enterprise Managerによる単一ツールによるOracle Cloudの各種サービスの一括管理、さらにはOracle Cloud ConsoleとCommand Line Interface（CLI）による包括的な管理といった、多彩な方法がそろっています。DevOps支援にもつながるREST APIによるOracle MySQL Cloud Service運用管理も現在準備中となっています。

Oracle MySQL Cloud Serviceは、監視機能、バックアップ機能、そしてセキュリティ拡張機能といったMySQL Enterprise Editionのすべての機能が簡単に利用でき、さらにMySQLの開発元からのサポートサービスを受けることができます。

SSHでの接続と制限のないソフトウェア構成のため、要件やアクセス状況に応じてクラウドとオンプレミスの間で相互の移行や連動ができる仕組みとなっており、クラウドにロックインされる心配が不要な点は独自の大きなメリットといえます。

Oracle MySQL Cloud Serviceの活用が想定される場面としては、開発環境やテスト環境を迅速に用意するケースをはじめ、オンプレミスやほかのクラウドサービスから機能やサービスがより充実した環境への移行、さらにはアクセスのピーク時にクラウドのリソースを利用することや逆に負荷が低いデータベースを集約することなどが考えられます。はじめからクラウド環境での運用を想定したアプリケーションを迅速に開発し、かつ機密性の高いデータの保護を低コストで実現する環境にもなります。

Column　アプリケーションからの接続部品

各プログラミング言語で開発したアプリケーションからMySQLサーバーに接続するための部品の中で、MySQL開発チームが開発したモジュール群については「Connectors」と総称しています。

Connectorsは現在、以下のものが開発され公開されています。

名称	対象言語等
Connector/ODBC	ODBC ドライバー
Connector/Net	.Net データ・プロバイダー
Connector/J	JDBC ドライバー
Connector/Node.js	Node.js 用ドライバー
Connector/Python	Python 用ドライバー
Connector/C++	C++ インターフェース
Connector/C	C クライアントライブラリ

名称対象言語等Connector/ODBCODBCドライバーConnector/Net.Netデータ・プロバイダーConnector/JJDBCドライバーConnector/Node.jsNode.js用ドライバーConnector/PythonPython用ドライバーConnector/C++C++インターフェースConnector/CCクライアントライブラリ

PHP用のドライバーであるMySQL Native Driver（mysqlnd）がMySQLのサイトからダウンロードできるようになっています。なお名称にはConnectorは含まれていません。mysqlndには、レプリケーションのマスタースレーブ構成において、更新処理をマスターに、参照処理をスレーブのいずれかに振り分ける処理が可能なmysqlnd_msや、クエリーの実行結果をクライアントサイドでキャ

ッシュするmysqlnd-qcなどのプラグインによって機能拡張が可能です。

JDBCドライバーであるConnector/Jは、mysqlnd_msに相当する機能を持つほか、MBeanインターフェースに対してJMXクライアントから接続先サーバーの追加や削除、変更などが動的に行えるなど、Connectorsの中でも機能が豊富な製品となっています。

上記以外の開発言語向けの接続部品は、MySQLの開発チームでは開発を行っていません。Rubyについては日本MySQLユーザ会の代表である、とみたまさひろさんが開発したMySQL/RubyやRuby/MySQLが使われてきましたが、現在はmysql2を利用することが多いようです。

著者プロフィール

梶山 隆輔（かじやま りゅうすけ）

日本オラクル株式会社 MySQL Global Business Unit Asia Pacific & Japan担当
MySQL Sales Consulting Senior Manager

日本オラクル株式会社において、MySQLのお客様環境への導入支援や製品の技術解説を担当するセールスコンサルタントチームのアジア太平洋地域リーダー。多国籍なMySQL部門にてアジア太平洋地域に在籍するチームメンバーを束ね、25以上の国や地域でのMySQL普及やビジネスの拡大をミッションとする。

山﨑 由章（やまさき よしあき）

MySQLのセールスコンサルタント。元々はOracleデータベースのコンサルティング、サポート等に従事していたが、オープンソースとフリーソフトウェア（自由なソフトウェア）の世界に興味を持ち、MySQLの仕事を始める。趣味は旅行と美味しいものを食べること。

● STAFF LIST

カバー・本文デザイン　株式会社トップスタジオ デザイン室
DTP制作・編集協力　株式会社トップスタジオ
編集　伊藤 隆司

■ 読者の窓口
インプレスカスタマーセンター
〒101-0051 東京都千代田区神田神保町一丁目105番地
TEL 03-6837-5016 ／ FAX 03-6837-5023
info@impress.co.jp

■ 書店／販売店のご注文窓口
株式会社インプレス 受注センター
TEL 048-449-8040
FAX 048-449-8041

やさしく学べるMySQL運用・管理入門【5.7対応】

2016年12月21日 初版発行

著　者　梶山 隆輔、山﨑 由章

発行人　土田米一

編集人　高橋隆志

発行所　株式会社インプレス
〒101-0051 東京都千代田区神田神保町一丁目105番地
TEL 03-6837-4635（出版営業統括部）
ホームページ　http://book.impress.co.jp/

印刷所　株式会社 廣済堂

ISBN978-4-295-00019-8 C3055

Printed in Japan