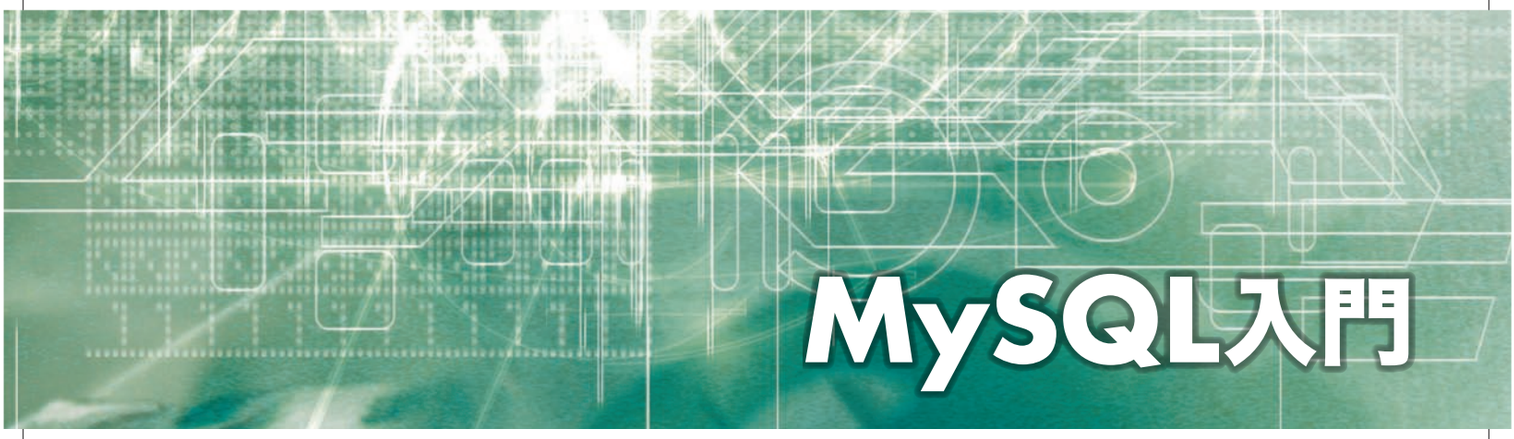

MySQL入門

第1章　データベースシステムの基礎と役割　1

1.1 データベース　2

1.1.1 データベース利用のメリット　2
1.1.2 データベースの種類　3

1.2 データベース管理システム　4

1.2.1 DBMSの機能　4
1.2.2 アプリケーションからのデータベースの利用　6

1.3 リレーショナル・データベース管理システムとOSSの実装　7

1.3.1 MySQL　7
1.3.2 PostgreSQL　9
1.3.3 Firebird　10
1.3.4 SQLite　11
1.3.5 商用RDBMSとオープンソースのRDBMS　12

第2章　RDBMSの仕組みと構造　15

2.1 リレーショナルモデル　16

2.1.1 リレーショナルモデルとは　16
2.1.2 主キー　17
2.1.3 関係演算　18
2.1.4 集合演算　19

2.2 SQL　20

2.2.1 SQLの概要　20
2.2.2 データ定義言語（DDL）　21
2.2.3 データ操作言語（DML）　22
2.2.4 データ制御言語（DCL）　23

第3章　MySQLのインストールと基本操作(1)　25

3.1 MySQLのインストール　26

3.1.2 MySQLの概要　26
3.1.3 RPMパッケージでのインストール（CentOS）　27

3.2 MySQLの基本操作　29

3.2.1 MySQLサーバの起動と終了　29
3.2.2 MySQLクライアント　31
3.2.3 データベースへの接続　33

第4章　MySQLのインストールと基本操作(2)　37

4.1 データベースの作成と削除　38

4.1.1 データベースの作成　38
4.1.3 データベースの削除　39

4.2 MySQLの設定 40
4.2.1 新規データベースユーザーの作成 40
4.2.2 データベースユーザーへの権限設定 42
4.2.3 データベースユーザーパスワードの変更 43
4.2.4 データベースユーザーの削除 44
4.3 MySQLとプログラミング言語の連携 45
4.3.1 MySQLとPHPの連携 45
4.3.2 MySQLとRubyの連携 46

第5章 テーブルの更新、照会、結合 49
5.1 テーブルの作成と更新 50
5.1.1 テーブルの作成 50
5.1.2 主キー 52
5.1.3 NOT NULL 53
5.1.4 レコードの追加 54
5.2 データの検索 55
5.2.1 SELECT文の基本 55
5.2.2 検索条件の指定 57
5.2.3 グループ化 59
5.2.4 集合関数 60
5.2.5 出力レコード数の制限 62
5.2.6 ソート 63
5.2.7 ファイルへの出力 64

第6章 表の作成、更新、照会、結合(2) 67
6.1 テーブルの結合 68
6.1.1 UNIONによる結合 68
6.1.2 内部結合 70
6.1.3 外部結合 72
6.1.4 副問い合わせ 74
6.1.5 自己結合 75
6.2 テーブルの操作 76
6.2.1 列の追加 76
6.2.2 列の削除 77
6.2.3 レコードの更新 78
6.2.4 テーブルのコピー 79
6.2.5 レコードとテーブルの削除 80
6.2.6 テーブル名の変更 81

第7章　トランザクション、参照整合性　83

7.1 トランザクション　84
7.1.1 トランザクションとは　84
7.1.2 ロックと排他制御　85
7.1.3 MySQLにおけるトランザクション　86

7.2 参照整合性　89
7.2.1 外部キーとは　89
7.2.2 参照整合性制約　90

第8章　データベース設計の基礎(1)　93

8.1 データベースの設計　94
8.1.1 データモデリング　94
8.1.2 概念データモデルと論理データモデルの作成　95
8.1.3 3層スキーマ　96

8.2 ER図　97
8.2.1 ERモデル　97
8.2.2 ER図の記法　98
8.2.3 カーディナリティ　99

第9章　データベース設計の基礎(2)　101

9.1 正規化　102
9.1.1 正規化とは　102
9.1.2 関数従属とは　103
9.1.3 第1正規形　104
9.1.4 第2正規形　105
9.1.5 第3正規形　106

第10章　MySQLでのRDBシステム管理(1)　109

10.1 MySQLサーバの起動と停止　110
10.1.1 MySQLサーバの起動　110
10.1.2 MySQLサーバの動作確認　111

10.2 mysqladminコマンド　112
10.2.1 mysqladminコマンドの基本　112
10.2.2 MySQLサーバの情報確認　113
10.2.3 データベースの作成と削除　114
10.2.4 MySQLサーバのプロセス処理　115
10.2.5 その他操作　116
10.2.6 システム変数　117

10.3 データベースユーザーの権限設定 119
10.3.1 データベースユーザーの権限 119
10.3.2 データベースユーザーの登録 120
10.3.3 データベースユーザーの権限確認 121
10.3.4 権限とデータベースユーザーの削除 122
10.3.5 パスワードの変更 123

第11章 MySQLでのRDBシステム管理(2) 125

11.1 MySQLの管理 126
11.1.1 MySQLサーバの情報確認 126
11.1.2 バックアップとリストア 128
11.2 MySQLの構造 129
11.2.1 ストレージエンジン 129
11.2.2 ストレージエンジンの変更 130
11.2.3 設定ファイル 132
11.2.4 関連ファイル 133
11.2.5 ログファイル 134

第12章 Webを使ったRDBシステム管理 137

phpMyAdmin 138
phpMyAdmin 138
phpMyAdminのインストール（ソース） 139
phpMyAdminのインストール（RPM） 140
phpMyAdminの利用 141
phpMyAdminの機能 142
テーブル内容の表示 145
レコードの検索 146
データベースとテーブルの作成 147

MySQL

第1章

データベースシステムの基礎と役割

1.1 データベース

データベースとは、構造化された電子データの集合です。単なるデータの集まりではなく、相互に関連するデータを整理・統合し、検索性を考慮して構造化しています。

1.1.1 データベース利用のメリット

さまざまな業務システムやWebアプリケーションにおいては、データベースを利用しないものよりも利用するものの方がずっと多いでしょう。現在のアプリケーション開発では、データベースは不可欠なものとなってきています。

アプリケーションでデータを扱うには、たとえばファイルに独自フォーマットでデータを記録する処理をアプリケーションごとに用意する方法もありますが、そういった方法と比較して、データベースを利用するメリットには、次のようなものがあります。

・データの一元管理
複数のファイルやホストにデータが分散せず、一カ所で一元管理することができます。

・データの共有
複数のアプリケーション､複数のユーザーでデータを安全に共有できます。

・処理とデータの分離
データへアクセスするアプリケーションと、データそのものとを分離することで、保守が容易になります。

・データを関連づけて格納
複数のデータを相互に関連づけて格納し、検索することができます。

・登録するデータへの制約付与
ある項目には数値のみ、ある項目には20文字以内の文字列しか登録できないなど、制約を付けることができます。

・データの整合性確保
データの一貫性を確保することができます。

・複数のデータアクセス同時処理
複数のアプリケーションから複数のデータへ同時にアクセスしても、同時に処理をさばくことができます。

・データの安全な格納
アクセス権限を設定し、機密性を確保しやすくなっています。

1.1.2 データベースの種類

データベースの論理的なデータ構造を論理データモデルといいます。論理データモデルには、階層型データベース、ネットワーク型データベース、リレーショナル・データベース、オブジェクト指向データベースなどがあります。

◆階層型データベース

データをツリー構造として格納するモデルです。親となるデータが複数の子データを持つことができ、子データには必ず親データが存在します。

◆ネットワーク型データベース

データをネットワーク構造として格納するモデルです。

◆リレーショナル・データベース

データ構造を二次元の表を使って表します。表と表との間には関連づけ（リレーションシップ）を行うことで、複雑なデータ構造も表現できます。現在主流となっている形式のデータベースです。

◆オブジェクト指向データベース

オブジェクト指向の概念（カプセル化、クラス、継承など）を取り入れており、データをオブジェクトとして格納できます。そのため、オブジェクト指向プログラミング言語での開発がスムーズになります。

1.2 データベース管理システム

データベースを運用・管理し、データへのアクセス要求に応答するソフトウェアをデータベース管理システム（DBMS：DataBase Management System）といいます。データベースへデータを格納したり、データを検索したりする機能は、DBMSによって提供されます。

1.2.1 DBMSの機能

DBMSの主な機能として、トランザクション管理、排他制御、障害回復があります。

◆トランザクション管理

銀行の口座送金処理を例に取りましょう。Aさんの口座からBさんの口座へ1万円を送金するとします。この処理は、

①Aさんの口座残高を1万円減らす
②Bさんの口座残高を1万円増やす

という2つの処理に分けることができます。しかし、㈰の処理が完了した時点でシステムエラーが起きたとするとどうなるでしょう。Aさんの口座は1万円引かれ、Bさんの口座はそのままなので、1万円が消えてしまいます。つまり、これら2つの処理は、どちらかが欠けると整合性が取れなくなってしまうわけです。そのような分割不可の処理単位をトランザクションといいます。正常にトランザクションが完了したときはデータベースが更新されますが、途中で失敗した場合はデータベースに反映されません。そのようにしてデータベースの整合性を維持するのがトランザクション管理です。

◆排他制御

複数のユーザーが同時に一つのデータにアクセスし、別々の更新処理をかけるとデータに矛盾が発生してしまいます。そのようなことを防ぐのが排他制御機能です。あるユーザーがデータにアクセスしたときにデータベースにロックをかけ、他のユーザーが利用できないようにします。ロックには、データの参照と更新をともにロックする「占有ロック」と、データの更新だけをロックする「共有ロック」があります。

◆障害回復

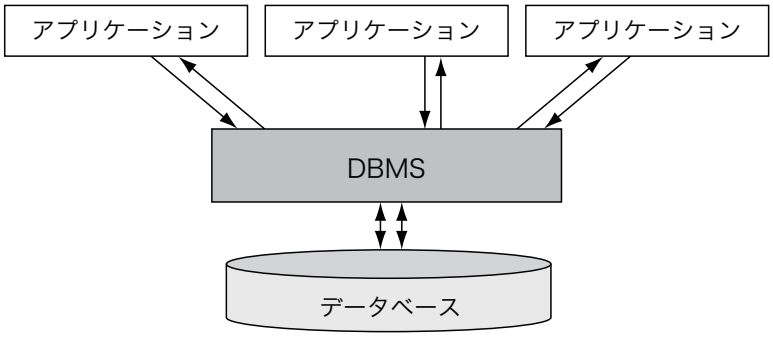
ハードウェアやソフトウェアに障害が発生しても、データベースの内容を復旧する機能が障害回復機能です。障害回復処理には、ロールフォワードとロールバックがあります。

DBMSは、データベースに対して更新処理が行われると、その内容をログファイル（ジャーナル）に保存します。障害が発生したとき、データベースのバックアップに加えて、ログファイルに記録されている処理を再現し、障害が発生する直前の状態まで復旧させるのがロールフォワードです。一方、障害が発生したとき、バックアップ時点までデータを巻き戻してから改めて処理を開始することをロールバックといいます。

1.2.2 アプリケーションからのデータベースの利用

アプリケーションでデータベースを利用する際の処理の流れを図に示します。

［図1-1］アプリケーションからのデータベースの利用



アプリケーションからはDBMSに対して指示を与えます。DBMSは指示を解釈し、OS上にあるデータベースのファイルにアクセスし、結果をアプリケーションに返します。このように、DBMSはアプリケーションとデータベースとの仲介役を果たしています。

1.3　リレーショナル・データベース管理システムとOSSの実装

オープンソースとして実装されたリレーショナル・データベース管理システムには、MySQL、PostgreSQL、Firebird、SQLiteなどがあります。SQLiteをのぞき、クライアント/サーバ型の構成をとります。

1.3.1 MySQL

MySQLは、スウェーデンのMySQL AB社で開発され、現在はSun Microsystemsによって開発されているリレーショナル・データベース管理システムです。1995年にバージョン1.0が登場し、現在はバージョン5.1系（安定版）と5.4系（開発版）、最新機能を盛り込んだ6.0系（開発版）が提供されています。機能よりも高速性と堅牢性を重視して発展してきましたが、現在では機能においても他のRDBMSに劣らないものとなっています。オープンソースのRDBMSとしては世界でもっとも普及しています。

［図1-2］www-jp.mysql.com

MySQLはオープンソースとして開発されていますが、無償で利用できるGPLライセンスと、有料のコマーシャルライセンスのデュアルライセンスを採用しており、用途に合わせて選択できます。無償版は「MySQL Community Server」、有償版は、一般的なDBサーバとして社内やASP等で使われる「MySQL Enterprise」、組込み用途やアプライアンスとして使われる「MySQL エンベデッドデータベース」、高い可用性とスケーラビリティ、パフォーマンスを備えた「MySQL Cluster」があります。有償版は開発元によってテストが行われたバイナリが採用されており、サポートも付属します。

日本では、MySQLのユーザーコミュニティとして日本MySQLユーザ会（http://www.mysql.gr.jp）があり、メーリングリストを中心に活動しています。

◆MySQL公式サイト

http://www.mysql.com

◆MySQL日本語サイト

http://www-jp.mysql.com

◆MySQLユーザ会

http://www.mysql.gr.jp

1.3.2 PostgreSQL

PostgreSQLは、PostgreSQL Global Development Groupによって開発されているリレーショナル・データベース管理システムです。商用データベース並みの充実した機能を持っており、特に日本国内では商用システムにおいて広く利用されています。

［図1-3］www.postgresql.jp

日本では、PostgreSQLのユーザーコミュニティとしてPostgreSQLユーザー会（http://www.postgresql.jp/）があります。

◆PostgreSQL公式サイト

http://www.postgresql.org/

◆PostgreSQLユーザー会

http://www.postgresql.jp/

1.3.3 Firebird

Firebirdは、商用RDBMSのInterBaseから派生したリレーショナル・データベース管理システムです。Mozilla Public Licenseに近いInterBase Public Licenseに基づいてオープンソースで開発されており、MySQL、PostgreSQLに次ぐオープンソースRDBMSとして注目されています。

◆公式サイト

http://www.firebirdsql.org/

◆Firebird日本ユーザー会

http://firebird.gr.jp/

1.3.4 SQLite

SQLiteは、アプリケーションに組み込んで利用される形態の簡易RDBMSで、クライアント/サーバ型ではなくライブラリとして提供されます。PHPやPythonでは標準ライブラリに組み込まれており、プログラムから簡単に利用することができます。

◆SQLite公式サイト

http://www.sqlite.org/

1.3.5 商用RDBMSとオープンソースのRDBMS

よく知られた商用RDBMS製品としては、Oracle社のOracle Database、Microsoft社のSQL Server、IBM社のDB2などがあります。いずれも有償の製品であり、ベンダーによるサポートが付属しています。

※無償で利用できる版が提供されている場合もあります。

これらの商用RDBMSと比較して、オープンソースのRDBMSは何が違うのでしょうか。おそらく一般的な用途では、機能面でオープンソースのRDBMSが大きく劣るということはなくなってきています。サポートに関しても、オープンソースのRDBMSをサポートしているサードパーティベンダー*がありますから、有償でサポートを受けることは可能です。比較的大規模なデータベースにも、オープンソースRDBMSを適用することができます。

＊MySQLの場合は開発元であるSun Microsystemsがサポートしています。

ただ、パフォーマンスやスケーラビリティについては商用製品に利があることが多いでしょう。もちろん、パフォーマンスチューニングを施すことによってオープンソースRDBMSの性能を高めることはできますが、そのためのノウハウが十分に提供されているとは言えない状況があります。パフォーマンスチューニングをせずにオープンソースRDBMSを導入し、「やはり無料のソフトウェアは使えない」と判断されるケースも耳にします。オープンソースRDBMSを導入する場合は、その特徴を理解し、性能を十分に発揮できるように工夫しなければならないこともあります。

また、実務で利用する場合は、協力会社が必要なスキルを持っているか、教育・研修体制が十分か、開発ツールが対応しているか、他のシステムとの親和性はあるか、技術情報が豊富に存在するか、といったことも重要になってきます。

RDBMSを導入する場合は、そのような点を考慮し、オープンソースを採用するのか、商用製品を採用するのかを決定する必要があるでしょう。

第1章 テスト

問　題　1

代表的なオープンソースのRDBMSの名前を3つ以上挙げてください。

問　題　2

商用RDBMSとオープンソースのRDBMSとの違いをいくつか挙げてください。

MySQL

第2章

RDBMSの仕組みと構造

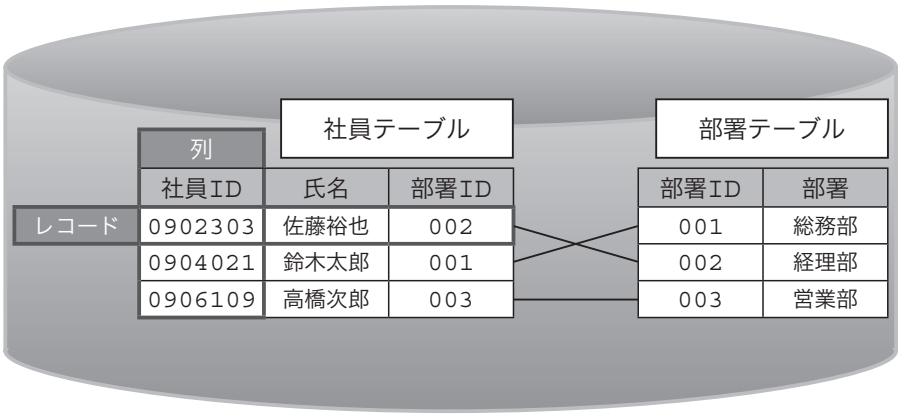
2.1 リレーショナルモデル

データベースにおけるデータの構造をモデル化したものをデータベースモデルといいます。現在では、関係モデル（リレーショナルモデル）が主流となっています。

2.1.1 リレーショナルモデルとは

リレーショナルモデルでは、データ構造は表形式として見ることができます。

［図2-1］リレーショナルモデル



テーブルは列（フィールド、カラム、属性）と行（レコード）から構成されます。本書では、「列」「レコード」と呼ぶことにします。

RDBでは、複数のテーブルのデータを関連づけて表現します。たとえば、図2-1では、社員テーブルの部署IDと、部署テーブルの部署IDが対応しています。このような対応関係をリレーションシップといいます。

2.1.2 主キー

テーブル内で任意のレコードを一意に識別できる列もしくは列の組のことを主キーといいます。主キーに必要な条件は次のとおりです。

・一意である（重複がない）
・必ず値が格納されている（NULL値がない）
・更新されない（値が不変）

たとえば、次の図を見てください。

[図2-2] 主キー

社員番号	氏名	電話番号	携帯番号
91001	佐藤　太郎	030-0000-0000	090-0000-0000
91002	鈴木　次郎	030-1111-1111	090-1111-1111
91003	鈴木　花子	030-1111-1111	
80011	佐藤　太郎	030-2222-2222	090-2222-2222

このテーブルには、4つの列（社員番号、氏名、電話番号、携帯番号）があります。これらの列のうち、氏名欄は同姓同名がいるので、主キーにはできません。電話番号も、同一世帯の社員では重複するので不適切です。携帯番号は、携帯電話を持っていない人もいるので不適切です。社員番号が重複なしに割り当てられているとすれば、主キーとして適切と考えられます。

主キーは複数の列を組み合わせてもかまいません。次の図では、出席番号がクラス内で一意であるとするなら、クラス番号と出席番号を組み合わせて主キーとすることができます。複数の列を組み合わせたものを複合キーといいます。

[図2-3] 複合キー

クラス番号	出席番号	氏名
200901	0001	佐藤　太郎
200901	0002	鈴木　花子
200902	0001	高橋　次郎
200902	0002	田中　優子

一般的に、「○○番号」「○○ID」「○○コード」といった列が主キーの候補となるでしょう。

2.1.3 関係演算

データベースを検索して必要なデータを取り出すことを演算といいます。演算には、関係演算と集合演算があります。

関係演算は、目的とするデータをテーブルから取り出す作業です。関係演算には、射影、選択、結合があります。

テーブルから、指定した列だけを取り出すのが射影、指定した行だけを取り出すのが選択です。次の例は、顧客名による射影と、顧客番号が2034の行による選択を表しています。

［図2-4］射影、選択

顧客番号	顧客名	店舗コード
1001	佐藤　太郎	0008
2003	鈴木　花子	0021
2034	高橋　次郎	0012
2972	田中　優子	0003

選択

顧客番号	顧客名	店舗コード
2034	高橋　次郎	0012

射影

顧客名
佐藤　太郎
鈴木　花子
高橋　次郎
田中　優子

結合は、ある項目に基づいて2つ以上のテーブルを連結させる作業です。次の例では、店舗コードに基づいて2つのテーブルを結合しています。

［図2-5］結合

顧客番号	顧客名	店舗コード
1001	佐藤　太郎	0008
2003	鈴木　花子	0021
2034	高橋　次郎	0012
2972	田中　優子	0003

店舗コード	店舗名
0003	新宿店
0008	調布店
0012	三鷹店
0021	新横浜店

結合

顧客番号	顧客名	店舗コード	店舗名
1001	佐藤　太郎	0008	調布店
2003	鈴木　花子	0021	新横浜店
2034	高橋　次郎	0012	三鷹店
2972	田中　優子	0003	新宿店

2.1.4 集合演算

集合演算は、2つのテーブルからデータを取り出す演算です。すべてのデータを取り出す和演算、共通するデータを取り出す積演算、どちらか一方のテーブルのみにあるデータを取り出す差演算があります。

例として、次のようなテーブルA、テーブルBがあるとしましょう。

［図2-6］集合演算

テーブルA

店舗コード	店舗名
0003	新宿店
0008	調布店
0012	三鷹店
0021	新横浜店

テーブルB

店舗コード	店舗名
0003	新宿店
0009	八王子店
0018	川崎店
0021	新横浜店

和演算では、テーブルAとテーブルBにあるデータがすべて取り出せます。2つのテーブルで重複する行は1行にまとめられます。

［図2-7］和

店舗コード	店舗名
0003	新宿店
0008	調布店
0009	八王子店
0012	三鷹店
0018	川崎店
0021	新横浜店

積演算では、テーブルAとテーブルBの双方に共通するデータが取り出せます。

［図2-8］積

店舗コード	店舗名
0003	新宿店
0021	新横浜店

差演算では、テーブルAとテーブルBの差分、たとえば以下の例では、テーブルAからテーブルBにあるものを引いたデータが取り出せます。

［図2-9］差

店舗コード	店舗名
0008	調布店
0012	三鷹店

2.2 SQL

SQLは、リレーショナル・データベースにアクセスするためのデータベース言語です。SQLを使って、テーブルの作成・削除や、データの検索・更新・削除といった作業ができます。

2.2.1 SQLの概要

SQLは、関係モデルに基づくデータベース言語で、ISOやANSI、JISによって標準化されています。制定年度によって、SQL92（1992年）、SQL99（1999年）、SQL:2003（2003年）などの規格があります。どのRDBMSでもSQLをサポートしていますが、どのSQL規格を採用しているかはRDBMSによって異なります。また、それぞれのRDBMSで独自の拡張を加えている場合もありますので、注意が必要です。

SQLのコマンドは、機能によって「データ定義言語」「データ操作言語」「データ制御言語」の3つに分類することができます。

2.2.2 データ定義言語（DDL）

データ定義言語（DDL：Data Definition Language）は、データベースに格納するデータの構造を定義します。DDLには次表のようなものがあります。

［図2-10］データ定義言語（DDL）

命令	説明
CREATE	テーブルなどを作成する
ALTER	テーブルなどの定義を変更する
DROP	テーブルなどを削除する

2.2.3 データ操作言語（DML）

データ操作言語（DML：Data Manipuration Language）は、テーブル内のデータの参照、更新・追加、削除などを行うための命令です。DMLには次表のようなものがあります。

［図2-11］データ操作言語（DML）

命令	説明
SELECT	レコードを検索する
INSERT	レコードを挿入する
UPDATE	レコードを更新する
DELETE	レコードを削除する

2.2.4 データ制御言語（DCL）

データ制御言語（DCL：Data Control Language）は、データベースのアクセス権などを設定するための命令です。DCLには次表のようなものがあります。

［図2-12］データ制御言語（DCL）

命令	説明
COMMIT	トランザクションを確定する
ROLLBACK	トランザクションを破棄する
GRANT	特定のユーザーに権限を与える
REVOKE	ユーザーに与えた権限を削除する

第2章 テスト

問　題　1

テーブルを構成する2つの要素を挙げてください。

問　題　2

テーブル内で任意のレコードを一位に識別できる列もしくは列の組のことを何と呼びますか？

問　題　3

関係演算を3つ挙げてください。

問　題　4

SQLとは何ですか？

問　題　5

SQLの代表的な規格を挙げてください。

MySQL

第3章

MySQLのインストールと基本操作(1)

3.1 MySQLのインストール

MySQLは、オープンソースのDBMSとしては、世界でもっとも広く使われているソフトウェアです。

3.1.2 MySQLの概要

MySQLは、スウェーデンのMySQL AB社で開発されたリレーショナル・データベース管理システムです。1995年にバージョン1.0が登場し、現在はバージョン5.1系（安定版）と5.4系（開発版）、最新機能を盛り込んだ6.0系（開発版）が提供されています。

MySQL AB社は2007年にSun Microsystems社に買収され、以後はSun MicrosystemsがMySQLの開発と提供を行ってきました。

MySQLの最大の特徴は「軽快さ」です。大規模なデータベースでも高速なアクセスが可能ですが、その反面、古いバージョンでは他のRDBMSにあるような機能が含まれていませんでした。しかし現在では標準的な機能を実装し、かつ高速性も備えています。

Webアプリケーションの標準的なオープンソースプラットフォームとして「LAMP」（Linux、Apache、MySQL、PHP/Perl/Python）という言葉がありますが、そこにMySQLが含まれているとおり、Webアプリケーションの土台としても広く利用されてきています。

MySQLは、Linux、UNIX、Windowsをはじめ、20種類以上のOS上で実行することができます。Linuxディストリビューションでは標準パッケージを用意しているものも多く、手間をかけずに使い始めることができます。

3.1.3 RPMパッケージでのインストール（CentOS）

CentOSでは、MySQLのRPMパッケージが用意されていますので、yumコマンドを使ってインストールできます。どのようなMySQL関連のパッケージがあるのかは、次のようにして調べることができます。

```
# yum search mysql
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.riken.jp
 * updates: ftp.riken.jp
 * addons: ftp.riken.jp
 * extras: ftp.riken.jp
========================= Matched: mysql ==========================
mod _ auth _ mysql.i386 : Basic authentication for the Apache web
server using a
                        : MySQL database.
qt-MySQL.i386 : MySQL drivers for Qt's SQL classes.
MySQL-python.i386 : An interface to MySQL
bytefx-data-mysql.i386 : MySQL database connectivity for Mono
freeradius-mysql.i386 : MySQL bindings for freeradius
libdbi-dbd-mysql.i386 : MySQL plugin for libdbi
mysql.i386 : MySQL client programs and shared libraries.
mysql-bench.i386 : MySQL benchmark scripts and data.
mysql-connector-odbc.i386 : ODBC driver for MySQL
mysql-devel.i386 : Files for development of MySQL applications.
mysql-server.i386 : The MySQL server and related files.
mysql-test.i386 : The test suite distributed with MySQL.
pdns-backend-mysql.i386 : MySQL backend for pdns
perl-DBD-MySQL.i386 : A MySQL interface for perl
php-mysql.i386 : A module for PHP applications that use MySQL
databases.
php-pdo.i386 : A database access abstraction module for PHP
applications
php-pear-MDB2-Driver-mysql.noarch : MySQL MDB2 driver
qt4-mysql.i386 : MySQL drivers for Qt's SQL classes
rsyslog.i386 : Enhanced system logging and kernel message
trapping daemons
rsyslog-mysql.i386 : MySQL support for rsyslog
unixODBC.i386 : A complete ODBC driver manager for Linux.
```

主なMySQL関連パッケージを表にまとめます。

[表3-1] MySQL関連パッケージ

パッケージ名	説明
mysql-server	MySQLサーバ
mysql	MySQLクライアントプログラムと共有ライブラリ
mysql-bench	MySQLベンチマークプログラム
mysql-server	開発用パッケージ
mysql-connector-odbc	ODBCドライバ
php-mysql	PHPからMySQLを利用するためのモジュール
perl-DBD-MySQL	PerlからMySQLを利用するためのインターフェース

MySQLサーバとMySQLクライアントをインストールするには、次のコマンドを実行します。

```
# yum install mysql mysql-server -y
```

MySQLがインストールされたかどうかは、次のコマンドで確認できます。

```
# rpm -qa | grep mysql
mysql-5.0.45-7.el5
mysql-server-5.0.45-7.el5
```

3.2 MySQLの基本操作

MySQLサーバに対する操作には、コマンドラインで行うものとGUIで行うものがあります。ここではコマンドラインによる操作を取り上げます。

3.2.1 MySQLサーバの起動と終了

MySQLサーバを起動するには、起動スクリプトを使います。

```
# /etc/init.d/mysqld start
Starting MySQL:                            [  OK  ]
```

デフォルトでは、MySQLサーバの管理者ユーザーにはパスワードが設定されていないため、初回起動時には警告メッセージが表示されます。MySQLの管理コマンドであるmysqladminコマンドを使って、パスワードを設定しておきましょう。「new-password」の部分は、任意のパスワードに置き換えてください。

```
# mysqladmin -u root password 'new-password'
```

このrootユーザーは、システムのrootユーザーとは別ですので注意してください。

MySQLサーバは3306番ポートを利用します。netstatコマンドを実行して、3306番ポートが開かれているか確認できます。

```
# netstat -atn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address    Foreign Address  State
tcp        0      0 0.0.0.0:3306     0.0.0.0:*        LISTEN
tcp        0      0 127.0.0.1:25     0.0.0.0:*        LISTEN
tcp        0      0 :::22            :::*             LISTEN
```

MySQLサーバの動作は、mysqladminコマンドを使っても確認することができます。パスワードには、先ほど設定したパスワードを入力します。

```
# mysqladmin -u root -p ping
Enter password:
mysqld is alive
```

「mysqld is alive」と表示されれば、MySQLサーバは稼働しています。MySQLサーバが稼働していない場合は、次のようなエラーメッセージが表示されます。

```
# mysqladmin -u root -p ping
Enter password:
mysqladmin: connect to server at 'localhost' failed
error: 'Can't connect to local MySQL server through socket '/var/
lib/mysql/mysql.sock' (2)'
Check that mysqld is running and that the socket: '/var/lib/mysql/
mysql.sock' exists!
```

起動スクリプトを使ってMySQLサーバの稼働状態を調べることもできます。MySQLサーバが起動しているときは、PID（プロセスID）番号と「running...」の文字が表示されます。

```
# service mysqld status
mysqld (pid 12116) is running...
```

MySQLサーバを停止するには、次のコマンドを実行します。

```
# /etc/init.d/mysqld stop
Stopping MySQL:                            [  OK  ]
```

また、設定変更をするなどしてMySQLサーバの再起動が必要な場合は、次のコマンドを実行します。

```
# /etc/init.d/mysqld restart
Stopping MySQL:                            [  OK  ]
Starting MySQL:                            [  OK  ]
```


3.2.2 MySQLクライアント

MySQLサーバに接続してコマンドラインで操作をするには、MySQLクライアントコマンドを使います。MySQLクライアントコマンドはmysqlコマンドによって起動します。

```
# mysql -u root -p
Enter password:
```

-uオプションの引数には接続するユーザー名を指定します。-pオプションを指定すると、対話的にパスワードを尋ねられます。パスワードを正しく入力すると、プロンプトが「mysql>」に変わります。

※-pオプションの直後にパスワードを指定することもできますが、コマンド履歴に残ってしまうので、対話的に入力した方がよいでしょう。

```
# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or ¥g.
Your MySQL connection id is 2
Server version: 5.0.45 Source distribution

Type 'help;' or '¥h' for help. Type '¥c' to clear the buffer.

mysql>
```

プロンプト「mysql>」に続けて、mysqlコマンド用のコマンドを実行できます。また、任意のSQLコマンドを実行したり、MySQL独自のSQLコマンドを実行することもできます。主なコマンドを表3-2にまとめます。

[表3-2] 主なmysqlコマンド

mysqlコマンド	説明
\?、\h	MySQLクライアントコマンドのコマンドを一覧を表示する
\q	MySQLクライアントコマンドを終了する
\s	MySQLサーバのステータスを表示する
\C 文字コード	文字コードを変更する
USE データベース名;	指定したデータベースに接続する
SHOW DATABASES;	データベース一覧を表示する
SHOW TABLES;	テーブル一覧を表示する
DESCRIBE テーブル名;	テーブルの定義内容を表示する

MySQLクライアント用のコマンドは、「\」＋1文字のアルファベットで指定されます。たとえば、MySQLモニタを終了するには「\q」を入力します。

```
mysql> \q
Bye
```

実は、「\q」は「quit」コマンドの省略形です。次のように、quitコマンドを指定することもできます。その場合は、行末に「;」を付けてください。

```
mysql> quit;
Bye
```

すべてのMySQLクライアントコマンドは「\h」もしくは「\?」を実行すると表示されます。また「help;」と入力しても同じです。

```
mysql> \h

For information about MySQL products and services, visit:
   http://www.mysql.com/
For developer information, including the MySQL Reference Manual,
visit:
   http://dev.mysql.com/
To buy MySQL Network Support, training, or other products, visit:
   https://shop.mysql.com/

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db
and host.
delimiter (\d) Set statement delimiter. NOTE: Takes the rest of
the line as new delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to _ pager]. Print the query results via
PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an
argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to _ outfile]. Append everything into
given outfile.
use       (\u) Use another database. Takes database name as argument.
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'
```


3.2.3 データベースへの接続

MySQLサーバは、複数のデータベースを同時に扱うことができます。どのようなデータベースが存在するのかは、MySQLモニタ上で次のようにして確認することができます。

```
mysql> SHOW DATABASES;
+------------------------+
| Database               |
+------------------------+
| information _ schema   |
| mysql                  |
| test                   |
+------------------------+
3 rows in set (0.01 sec)
```

「SHOW DATABASESコマンド」というMySQLコマンドを実行すると、結果が表示されます。「information_schema」と「mysql」は、MySQLが内部的に利用するデータベースです。「test」はデフォルトで用意されているテスト用のデータベースです。

MySQLクライアントを起動した状態では、どのデータベースにも接続していません。データベースへ接続するには、USEコマンドを使います。次の例では、testデータベースに接続しています。

```
mysql> USE test;
Database changed
```

現在使用中のデータベースを知るには、SELECT DATABASEコマンドを実行します。

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| test       |
+------------+
1 row in set (0.01 sec)
```

もし、どのデータベースにも接続していないのであれば、次のように表示されます。

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| NULL       |
+------------+
1 row in set (0.00 sec)
```

「NULL」というデータベースに接続しているわけではないので注意してください。

第3章 テスト

問　題　1

正しい説明には○を、不適切な説明には×を記述してください。

（　　）MySQLはWebアプリケーションでよく利用される
（　　）MySQLは多くのOS上で動作する
（　　）MySQLサーバは通常、5432番ポートで待ち受ける

問　題　2

MySQLサーバへrootユーザーとして接続するためのコマンドを選択してください。

A. mysql -p root -u
B. mysql -u root -p
C. mysqladmin -p root -u
D. mysqladmin -u root -p

問　題　3

MySQLクライアントを終了するために入力する2文字を記述してください。

MySQL

第4章

MySQLのインストールと基本操作(2)

4.1 データベースの作成と削除

MySQLでは複数のデータベースを同時に管理することができます。

4.1.1 データベースの作成

データベースを作成するには、SQLコマンドのCREATE DATABASEを使います。

```
CREATE DATABASE データベース名;
```

次の例では、testdbという名前のデータベースを作成しています。

```
mysql> CREATE DATABASE testdb;
```

なお、データベースで日本語を扱う際には、キャラクタセットを指定してデータベースを作成します。日本語のキャラクタセットには、utf8（UTF-8）、eucjpms（EUC-JP）、cp932（シフトJIS）が指定できます。たとえば、キャラクタセットをシフトJISにしてデータベースtestdbを作成するには、次のように指定します。

```
mysql> CREATE DATABASE testdb CHARACTER SET
cp932;
```

なお、デフォルトのキャラクタセットはLatin-1です。

4.1.3 データベースの削除

作成済みのデータベースを削除するには、SQLコマンドのDROP DATABASEを使います。

```
DROP DATABASE データベース名;
```

次の例では、testdbデータベースを削除しています。

```
mysql> DROP DATABASE testdb;
```

データベースを削除できるのは、削除できる権限を持っているユーザーだけです。削除時には確認メッセージもなく、ただちにデータベースは削除されてしまいますので、誤って別のデータベースを削除してしまわないよう注意してください。

4.2 MySQLの設定

MySQLに接続して利用するには、あらかじめデータベースユーザーを作成しておく必要があります。デフォルトでは、管理者権限を持つrootユーザーが用意されています。

4.2.1 新規データベースユーザーの作成

MySQLをインストールした直後の状態では、データベースユーザーはrootユーザーのみが登録されています。このrootユーザーは、システムアカウントのrootユーザーとは関係ないので注意してください。一般的に、MySQLを利用する場合は、専用のユーザーを作成して利用します。管理者権限のあるrootユーザーを利用し続けることは、セキュリティ上好ましくないからです。

ユーザーの作成は、以下の書式で行います。

```
CREATE USER 'ユーザー名'@'ホスト名' IDENTIFIED BY 'パスワード';
```

次の例では、studentという名前のユーザーを作成しています。パスワードは「himitu」を指定しています。

```
mysql> CREATE USER 'student'@'localhost' IDENTIFIED BY 'himitu';
```

ただし、CREATE USERコマンドで作成されたユーザーには権限が設定されていないので、別途権限を設定する必要があります。そのため、GRANTコマンドを使って、ユーザーの作成と権限の設定を同時に行うのが一般的です。

```
GRANT 権限 ON データベース名.テーブル名 TO 'ユーザー名'@'ホスト名' IDENTIFIED BY 'パスワード';
```

次の例では、testdbデータベースのすべてのテーブルに対し、すべての権限を与えてstudentユーザーを作成します。

```
mysql> GRANT ALL ON testdb.* TO 'student'@'localhost' IDENTIFIED BY 'himitu';
```

データベースを指定しない場合は、ワイルドカード「*」を使ってください。次の例では、すべてのデータベースに対して、SELECTコマンドとUPDATEコマンドが利用できるユーザーstudentを作成します。

```
mysql> GRANT SELECT,UPDATE ON *.* TO 'student'@
'localhost' IDENTIFIED BY 'himitu';
```

なお、他のホストからネットワーク経由で利用したい場合は、ホスト名欄に「%」を指定してください。

```
mysql> GRANT ALL ON testdb.* TO 'student'@'%'
IDENTIFIED BY 'himitu';
```

この場合、localhostからは接続できなくなるので、student@localhostでの設定も別途必要です。

4.2.2 データベースユーザーへの権限設定

※ユーザーが存在しないと、そのユーザーを作成します。

GRANTコマンドは、データベースユーザーの権限を設定するコマンドです。

```
GRANT 権限 ON データベース名.テーブル名 TO ユーザー名@ホスト名 [IDENTIFIED BY 'パスワード'];
```

権限には、表に示すような権限を指定することができます。

```
mysql> CREATE USER 'student'@'localhost' IDENTIFIED BY 'himitu';
```

4.2.3 データベースユーザーパスワードの変更

データベースユーザーのパスワードは、SET PASSWORDコマンドを使って変更できます。

```
SET PASSWORD FOR ユーザー名@ホスト名=password('新しいパスワード');
```

次の例では、student@localhostユーザーのパスワードを「newpass」に変更しています。

```
mysql> SET PASSWORD FOR student@localhost=password('newpass');
```

4.2.4 データベースユーザーの削除

データベースユーザーを削除するには、DROP USERコマンドを使います。

```
DROP USER ユーザー名@ホスト名;
```

次の例では、student@localhostユーザーを削除しています。

```
mysql> DROP USER student@localhost
```

なお、MySQL 5.0.2未満では、DROP USERコマンドを実行する前に、REVOKEコマンドを使ってユーザーの権限を無効化する必要があります。次の例では、testdbデータベースですべての権限が与えられているstudentユーザーの権限を無効化しています。

```
mysql> REVOKE ALL ON testdb.* from 'student'@
'localhost';
```

4.3 MySQLとプログラミング言語の連携

プログラミング言語からMySQLに接続して利用するためには、それぞれの言語に対応したライブラリが必要になります。

4.3.1 MySQLとPHPの連携

PHPは、Webアプリケーションの開発によく利用されるスクリプト言語です。各種データベースとのインターフェースが用意されているので、MySQLをはじめとする各種データベースと簡単に連携できます。

CentOSをはじめとするLinuxディストリビューションでは、php-mysqlパッケージをインストールすることでPHPとMySQLを連携させることができます。

```
# yum install php-mysql -y
```

4.3.2 MySQLとRubyの連携

RubyはWebアプリケーションをはじめ、さまざまな分野で利用できるスクリプト言語です。RubyプログラムからMySQLを利用するためには、MySQL/RubyかRuby/MySQLが必要です。MySQL/RubyはC言語で作られたライブラリで、MySQL標準のC APIとRubyとの仲介役を果たします。また、Ruby/MySQLはRubyで作られたライブラリですが、最新のMySQLに対応していないなど、MySQL/Rubyと比較して機能不足が目立ちます。

MySQL/Rubyは、RubyGemsを使ってインストールするのが一般的です。あらかじめRubyGemsをインストール後、以下のコマンドでMySQL/Rubyをインストールできます。

```
# gem install --remote mysql
```

第4章 テスト

問　題　1

データベース「testdb」を作成するためのSQLを記述してください。

問　題　2

データベース「testdb」を削除するためのSQLを記述してください。

問　題　3

正しい説明には○を、不適切な説明には×を記述してください。

（　　）MySQLサーバを利用する場合、データベースユーザーは必ずrootを使用する
（　　）データベースユーザーに権限を与えるには、SQLのGRANT文を使用する
（　　）データベースユーザーのパスワードはOSのユーザーパスワードと同じである

MySQL

第5章

テーブルの更新、照会、結合(1)

5.1 テーブルの作成と更新

データベースにデータを登録するには、まずテーブルを作成する必要があります。

5.1.1 テーブルの作成

テーブルを作成するには、SQLコマンドのCREATE TABLEを使います。

```
CREATE TABLE テーブル名 (列名 データ型, 列名 データ型, ...);
```

各列には、挿入するデータの種類をデータ型で指定します。データ型から外れたデータは挿入できません。適切なデータ型を使用することで、データの整合性を高く保つことができます。MySQLでは、以下のデータ型を利用することができます。

※利用できるデータ型はRDBMSごとに若干異なります。

［表5-1］MySQLのデータ型

カテゴリ	データ型	説明
数値	TINYINT	1バイトの整数
	SMALLINT	2バイトの整数
	MEDIUMINT	3バイトの整数
	INT	4バイトの整数
	BIGINT	8バイトの整数
	FLOAT	4バイトの浮動小数点数
	DOUBLE	8バイトの浮動小数点数
	BIT	ビット列
日付・時刻	DATE	YYYY-MM-DD（1000-01-01～9999-12-31）
	DATETIME	YYYY-MM-DD HH:MM:SS（1000-01-01 00:00:00～9999-12-31 23:59:59）
	TIME	HH:MM:SS
	TIMESTAMP	YYYY-MM-DD HH:MM:SS（1970-01-01 00:00:01～2037-12-31 23:59:59）
	YEAR	YYYY（1901～2155）、YY（01～69、70～99）
文字列	CHAR(L)	0～255文字の固定長文字列
	VARCHAR(L)	0～65535文字の可変長文字列
	TINYTEXT	255バイトまでの文字列（大文字小文字の区別なし）
	TINYBLOB	255バイトまでの文字列（大文字小文字の区別あり）
	TEXT	65535バイトまでの文字列（大文字小文字の区別なし）
	BLOB	65535バイトまでの文字列（大文字小文字の区別あり）
	MIDIUMTEXT	16777215バイトまでの文字列（大文字小文字の区別なし）
	MIDIUMBLOB	16777215バイトまでの文字列（大文字小文字の区別あり）
	LONGTEXT	4294967295バイトまでの文字列（大文字小文字の区別なし）
	LONGBLOB	4294967295バイトまでの文字列（大文字小文字の区別あり）

```
mysql> CREATE TABLE staff (id VARCHAR(4), name
VARCHAR(20), age INT, bid VARCHAR(4));
```

SQLは「;」を行末としますので、以下のように複数行に分けて見やすくすることもできます。行末に「;」を入力するまで、続けて何行も入力することができます。その間、プロンプトは「-> 」に変更されます。

※途中で入力を取りやめたい場合は「¥c」と入力します。

```
mysql> CREATE TABLE staff (
    -> id VARCHAR(4),
    -> name VARCHAR(20),
    -> age INT, bid VARCHAR(4)
    -> );
```

5.1.2 主キー

主キーを設定するには、CREATE TABLEコマンドを実行する際、データ型の直後に「PRIMARY KEY」と指定します。次の例では、id列を主キーとして設定しています。

```
mysql> CREATE TABLE staff (id SMALLINT PRIMARY
KEY, name VARCHAR(20), age TINYINT, bid
SMALLINT);
```

どの列に主キーが設定されているかは、以下のようにして確認できます。

```
mysql> DESC staff;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | smallint(6) | NO   | PRI |         |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| age   | tinyint(4)  | YES  |     | NULL    |       |
| bid   | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

key欄に「PRI」とあるフィールドが主キーです。

5.1.3 NOT NULL

レコードを識別するためのキーとして使いたい列にデータが入っていなければ、困ったことになります。データの入力を必須とする列を指定するには、NOT NULLを指定します。次の例では、id列への入力を必須としています。

```
mysql> CREATE TABLE staff (id SMALLINT NOT
NULL, name VARCHAR(20), age TINYINT, bid
SMALLINT);
```

DESCコマンドを使ってテーブルの情報を調べてみます。

```
mysql> DESC staff;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | smallint(6) | NO   |     |         |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| age   | tinyint(4)  | YES  |     | NULL    |       |
| bid   | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

id列のNull欄が「NO」になっているのが確認できます。「YES」となっている列は、値が格納されていなくてもかまわない、ということです。

5.1.4 レコードの追加

テーブルにレコードを追加するには、SQLコマンドのINSERTを使います。

```
INSERT INTO テーブル名 [(列名1[, 列名2, ...])] VALUES
(値1[, 値2, ...]);
```

※文字列は引用符"もしくは""で区切ります。

次の例では、staffテーブルにレコードを追加しています。

```
mysql> INSERT INTO staff VALUES ('1', 'Sato',
'23', '1');
```

以下のように、複数のレコードをまとめて追加することもできます。

```
mysql> INSERT INTO staff VALUES ('2', 'Suzuki',
'24', '2'), ('3', 'Takahashi', '28', '1');
```

もちろん、見やすいように、複数行に分けてもかまいません。

```
mysql> INSERT INTO staff VALUES
> ('2', 'Suzuki', '24', '2'),
> ('3', 'Takahashi', '28', '1');
```

次のようにすると、指定した列にのみ値を追加することができます。

```
mysql> INSERT INTO staff (id, name) VALUES
('9', 'Kobayashi');

mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    5 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
|  9 | Kobayashi | NULL | NULL |
+----+-----------+------+------+
```

値を指定しなかった列はNULLとなります。

5.2 データの検索

データの検索もSQL文で行います。

5.2.1 SELECT文の基本

データベース内のレコードを検索するには、SQLコマンドのSELECTを使います。基本的な書式は次のとおりです。

```
SELECT 列名 FROM テーブル名;
```

次の例では、staffテーブルのname列とage列を出力します。

```
mysql> SELECT name,age FROM staff;
+-----------+------+
| name      | age  |
+-----------+------+
| Sato      |   23 |
| Suzuki    |   24 |
| Takahashi |   28 |
| Tanaka    |   26 |
| Watanabe  |   30 |
+-----------+------+
```

列名に「*」を指定すると、すべての列が出力されます。

```
mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+
```

列名を表示するときに、ASを使って別名を付けることもできます。

```
mysql> SELECT name as '名前' ,age as '年齢'  FROM
staff;
+-----------+--------+
| 名前        | 年齢     |
+-----------+--------+
| Sato      |     23 |
| Suzuki    |     24 |
| Takahashi |     28 |
| Tanaka    |     26 |
| Watanabe  |     30 |
+-----------+--------+
```

5.2.2 検索条件の指定

検索条件を指定するには、SELECTに続けてWHERE句を使います。

```
SELECT 列名 FROM テーブル名 WHERE 検索条件;
```

次の例では、age列が25未満のレコードを出力します。

```
mysql> SELECT * FROM staff WHERE age <= 25;
+----+--------+------+------+
| id | name   | age  | bid  |
+----+--------+------+------+
|  1 | Sato   |   23 |    1 |
|  2 | Suzuki |   24 |    2 |
+----+--------+------+------+
```

条件には、以下の表のような記号を利用することができます。

[表5-2] 条件に指定できる記号

演算子	説明
=	等しい
!=	等しくない
IN	いずれかが含まれる

ある文字列を含むレコードを検索したい場合は、LIKEで条件を指定します。

```
SELECT 列名 FROM テーブル名 WHERE 列名 LIKE 条件;
```

次の例では、name列が「T」で始まるレコードを出力します。

```
mysql> SELECT * FROM staff WHERE name LIKE
'T%';
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
+----+-----------+------+------+
```

「%」は任意の文字列を表すワイルドカード文字です。「_」は任意の1文字を表します。SQLでは「*」や「?」ではありませんので注意してください。

なお、「〜以外」で検索したいときは、LIKEの代わりに「NOT LIKE」を使います。

```
mysql> SELECT * FROM staff WHERE name NOT LIKE
'T%';
+----+----------+------+------+
| id | name     | age  | bid  |
+----+----------+------+------+
|  1 | Sato     |   23 |    1 |
|  2 | Suzuki   |   24 |    2 |
|  5 | Watanabe |   30 |    2 |
+----+----------+------+------+
```

5.2.3 グループ化

列のデータが同じレコードをまとめることをグループ化といいます。グループ化はGROUP BY句を使います。

```
SELECT 列名 FROM テーブル名 GROUP BY グループ化する列名;
```

次の例では、bid列が同じレコードをグループ化して表示しています。

```
mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+

mysql> SELECT * FROM staff GROUP BY bid;
+----+--------+------+------+
| id | name   | age  | bid  |
+----+--------+------+------+
|  1 | Sato   |   23 |    1 |
|  2 | Suzuki |   24 |    2 |
|  4 | Tanaka |   26 |    3 |
+----+--------+------+------+
```

この例では、bid列が共通しているレコードの中から1レコードだけが抽出されています。グループ化は、次に述べる集合関数を使って、合計や平均を計算する場合に使います。

5.2.4 集合関数

集合関数を使うと、抽出した結果を集計したり、検索条件にマッチしたレコードの中から平均値や最大値を計算したりすることができます。集合関数には、表のようなものがあります。

［表5-3］主な集合関数

関数	説明
AVG(列名)	平均値を計算する
COUNT(列名)	レコード数をカウントする
COUNT(DISTINCT 列名)	重複なしでレコード数をカウントする
MAX(列名)	最大値を調べる
MIN(列名)	最小値を調べる
SUM(列名)	合計を計算する

例を見てみましょう。次のようなテーブルを例に取ります。

```
mysql> SELECT * FROM members;
+-----------+----------+------+
| name      | address  | age  |
+-----------+----------+------+
| Sato      | Tokyo    |   23 |
| Suzuki    | Tokyo    |   24 |
| Takahashi | Kanagawa |   28 |
| Tanaka    | Kanagawa |   26 |
| Watanabe  | Tokyo    |   30 |
+-----------+----------+------+
```

addressごとにage列の合計を表示するには、SUM()関数を使います。

```
mysql> SELECT address,SUM(age) FROM members
GROUP BY address;
+----------+----------+
| address  | SUM(age) |
+----------+----------+
| Kanagawa |       54 |
| Tokyo    |       77 |
+----------+----------+
```

今度は、AVG()関数を使って平均を計算してみます。

```
mysql> SELECT address,AVG(age) FROM members
GROUP BY address;
+----------+----------+
| address  | AVG(age) |
+----------+----------+
| Kanagawa |  27.0000 |
| Tokyo    |  25.6667 |
+----------+----------+
```

COUNT()関数を使うとレコード数をカウントできます。

```
mysql> SELECT address,COUNT(name) FROM members
GROUP BY address;
+----------+-------------+
| address  | COUNT(name) |
+----------+-------------+
| Kanagawa |           2 |
| Tokyo    |           3 |
+----------+-------------+
```

5.2.5 出力レコード数の制限

表示するレコード数は、LIMITで指定できます。

```
SELECT 列名 FROM テーブル名 LIMIT 出力レコード数;
```

次の例では、出力レコード数を3に指定しています。

```
mysql> SELECT * FROM staff LIMIT 3;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    1 |
+----+-----------+------+------+
```

5.2.6 ソート

ORDER BY句を使うと、任意の項目でソートして出力することができます。デフォルトは昇順です。

```
SELECT 列名 FROM テーブル名 ORDER BY 順序指定;
```

次の例では、age列を昇順にソートして出力します。

```
mysql> SELECT * FROM staff ORDER BY age;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  4 | Tanaka    |   26 |    3 |
|  3 | Takahashi |   28 |    1 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+
```

DESCを指定すると、昇順ではなく降順でソートされます。

```
mysql> SELECT * FROM staff ORDER BY age DESC;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  5 | Watanabe  |   30 |    2 |
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
|  2 | Suzuki    |   24 |    2 |
|  1 | Sato      |   23 |    1 |
+----+-----------+------+------+
```

5.2.7 ファイルへの出力

SELECT文の実行結果をファイルに出力することもできます。

```
SELECT 列名 FROM テーブル名 INTO OUTFILE 出力ファイル名;
```

次の例では、ファイルtest.txtに検索結果が出力されます。

```
mysql> SELECT * FROM staff INTO OUTFILE 'test.
txt';
```

デフォルトでは、データベースファイルのあるディレクトリ（/var/lib/mysql以下など）に出力されます。

```
# cat /var/lib/mysql/testdb/test.txt
1       Sato    23      1
2       Suzuki  24      2
3       Takahashi       28      5
4       Tanaka  26      3
5       Watanabe        30      2
```

第5章 テスト

問　題　1

正しい説明には○を、不適切な説明には×を記述してください。

（　　）テーブルを作成するにはCREATE TABLE文を使う

（　　）DESCコマンドを使うと、どの列が主キーに設定されているか確認できる

（　　）データの入力を必須とする列にはNO NULLを設定する

問　題　2

レコードを追加するためのSQLコマンドを記述してください。

問　題　3

staffテーブルのすべての列を出力するSQL文を記述してください。

問　題　4

staffテーブルから、age列が30以上のレコードを出力するSQL文を記述してください。その際、name列とage列のみ出力するものとします。

MySQL

第6章

表の作成、更新、照会、結合(2)

6.1 テーブルの結合

複数のテーブルを結合して、1つの表として扱うことができます。結合にはいくつかの種類があります。

6.1.1 UNIONによる結合

UNIONを使うと、複数のSELECT文を繋いで、複数テーブルのレコードを表示することができます。

```
SELECT 列名1 FROM テーブル名1 UNION SELECT 列名2
FROM テーブル名2;
```

たとえば、次のような2つのテーブルがあるとします。

```
mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+

mysql> SELECT * FROM staff2;
+----+----------+------+------+
| id | name     | age  | bid  |
+----+----------+------+------+
|  1 | Ito      |   27 |    2 |
|  2 | Yamamoto |   31 |    1 |
|  3 | Nakamura |   24 |    4 |
+----+----------+------+------+
```

2つのテーブルのレコードを全部まとめて出力できます。

```
mysql> SELECT * FROM staff UNION SELECT * FROM
staff2;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
|  1 | Ito       |   27 |    2 |
|  2 | Yamamoto  |   31 |    1 |
|  3 | Nakamura  |   24 |    4 |
+----+-----------+------+------+
```

なお、それぞれのテーブルでキャラクタセットが異なると、UNIONを使うことができませんので注意してください。

6.1.2 内部結合

UNIONによる結合は、2つのテーブルを縦に結合しましたが、横に結合するには内部結合や外部結合を使います。まず、以下のような2つのテーブル（staffとbranch）があるとします。

```
mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    1 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+
mysql> SELECT * FROM branch;
+----+------------+
| id | branchname |
+----+------------+
|  1 | Tokyo      |
|  2 | Yokohama   |
|  3 | Osaka      |
|  4 | Nagoya     |
+----+------------+
```

staffテーブルのbid列とbranchテーブルのid列を関連づけて表示することを内部結合と呼んでいます。

```
SELECT 列名1 FROM テーブル名1 [INNER] JOIN テーブル名2
ON テーブル名1.列名1 = テーブル名2.列名2;
```

次の例を見てください。「staff.bid = branch.id」により、それぞれの列が関連づけられています。

```
mysql> SELECT * FROM staff JOIN branch ON staff.bid = branch.id;
+----+-----------+------+------+----+------------+
| id | name      | age  | bid  | id | branchname |
+----+-----------+------+------+----+------------+
|  1 | Sato      |   23 |    1 |  1 | Tokyo      |
|  2 | Suzuki    |   24 |    2 |  2 | Yokohama   |
|  3 | Takahashi |   28 |    1 |  1 | Tokyo      |
|  4 | Tanaka    |   26 |    3 |  3 | Osaka      |
|  5 | Watanabe  |   30 |    2 |  2 | Yokohama   |
+----+-----------+------+------+----+------------+

mysql> SELECT name,age,branchname FROM staff JOIN branch ON
staff.bid = branch.id;
+-----------+------+------------+
| name      | age  | branchname |
+-----------+------+------------+
| Sato      |   23 | Tokyo      |
| Takahashi |   28 | Tokyo      |
| Suzuki    |   24 | Yokohama   |
| Watanabe  |   30 | Yokohama   |
| Tanaka    |   26 | Osaka      |
+-----------+------+------------+
```

なお、branchテーブルにある4つめのレコード（Nagoya）は、staffテーブルにはないので表示されません。

6.1.3 外部結合

内部結合では、結合に使う列のデータが両方のテーブルにあるレコードのみ出力されますが、外部結合では、どちらかのテーブルにしかデータがないレコードも出力されます。外部結合には、左外部結合と右外部結合があります。

```
SELECT 列名1 FROM テーブル名1 LEFT|RIGHT [OUTER]
JOIN テーブル名2 ON テーブル名1.列名1 = テーブル名2.列名2;
```

先ほどのテーブルとは少し異なるデータを使います。staffテーブルには、bidが5のメンバーがいるとします。一方、branchテーブルには、idは4までしかありません。

```
mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    5 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+

mysql> SELECT * FROM branch;
+----+------------+
| id | branchname |
+----+------------+
|  1 | Tokyo      |
|  2 | Yokohama   |
|  3 | Osaka      |
|  4 | Nagoya     |
+----+------------+
```

まず、左外部結合から見ていきましょう。左外部結合では、最初に（左側で）指定した方のテーブルにあるレコードすべてを出力します。右側のテーブルのレコードは、場合によってはNULLとなります。次の例では、左側のテーブルにあるbid = 5に対応するデータが右側のテーブルにないので、その部分はNULLとなっています。

※「LEFT JOIN」は「LEFT OUTER JOIN」と指定してもかまいません。

```
mysql> SELECT * FROM staff LEFT JOIN branch ON staff.bid =
branch.id;
+----+-----------+------+------+------+------------+
| id | name      | age  | bid  | id   | branchname |
+----+-----------+------+------+------+------------+
|  1 | Sato      |   23 |    1 |    1 | Tokyo      |
|  2 | Suzuki    |   24 |    2 |    2 | Yokohama   |
|  3 | Takahashi |   28 |    5 | NULL | NULL       |
|  4 | Tanaka    |   26 |    3 |    3 | Osaka      |
|  5 | Watanabe  |   30 |    2 |    2 | Yokohama   |
```

右外部結合はその逆です。後に（右側で）指定した方のテーブルにあるレコードをすべて出力します。

※「RIGHT JOIN」は「RIGHT OUTER JOIN」と指定してもかまいません。

```
mysql> SELECT * FROM staff RIGHT JOIN branch ON staff.bid =
branch.id;
+------+----------+------+------+----+------------+
| id   | name     | age  | bid  | id | branchname |
+------+----------+------+------+----+------------+
|    1 | Sato     |   23 |    1 |  1 | Tokyo      |
|    2 | Suzuki   |   24 |    2 |  2 | Yokohama   |
|    5 | Watanabe |   30 |    2 |  2 | Yokohama   |
|    4 | Tanaka   |   26 |    3 |  3 | Osaka      |
| NULL | NULL     | NULL | NULL |  4 | Nagoya     |
+------+----------+------+------+----+------------+
```

6.1.4 副問い合わせ

ある条件で検索したデータを使ってさらに検索を行いたい場合があります。そのような、SELECT文の中で使われるSELECT文を副問い合わせ（サブクエリー）といいます。

```
SELECT 列名 FROM テーブル名 WHERE 列名 IN (SELECT ～);
```

次の例では、age列が最大のレコードのname列を表示しています。

```
mysql> SELECT name FROM staff WHERE age IN
(SELECT MAX(age) FROM staff);
+----------+
| name     |
+----------+
| Watanabe |
+----------+
```

処理を分析してみましょう。

```
mysql> SELECT MAX(age) FROM staff;
+----------+
| MAX(age) |
+----------+
|       30 |
+----------+

mysql> SELECT * FROM staff WHERE age = 30;
+----+----------+------+------+
| id | name     | age  | bid  |
+----+----------+------+------+
|  5 | Watanabe |   30 |    2 |
+----+----------+------+------+
```

6.1.5 自己結合

同一のテーブルに別名を付けて結合することを自己結合といいます。

```
SELECT 列名 FROM テーブル名 AS 別名1 JOIN テーブル名 AS
別名2;
```

自己結合の例を見てみましょう。

```
mysql> SELECT * FROM staff AS a JOIN staff AS b;
+----+-----------+------+------+----+-----------+------+------+
| id | name      | age  | bid  | id | name      | age  | bid  |
+----+-----------+------+------+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |  1 | Sato      |   23 |    1 |
|  3 | Takahashi |   28 |    5 |  1 | Sato      |   23 |    1 |
|  4 | Tanaka    |   26 |    3 |  1 | Sato      |   23 |    1 |
|  5 | Watanabe  |   30 |    2 |  1 | Sato      |   23 |    1 |
|  1 | Sato      |   23 |    1 |  2 | Suzuki    |   24 |    2 |
|  2 | Suzuki    |   24 |    2 |  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    5 |  2 | Suzuki    |   24 |    2 |
|  4 | Tanaka    |   26 |    3 |  2 | Suzuki    |   24 |    2 |
|  5 | Watanabe  |   30 |    2 |  2 | Suzuki    |   24 |    2 |
|  1 | Sato      |   23 |    1 |  3 | Takahashi |   28 |    5 |
|  2 | Suzuki    |   24 |    2 |  3 | Takahashi |   28 |    5 |
|  3 | Takahashi |   28 |    5 |  3 | Takahashi |   28 |    5 |
|  4 | Tanaka    |   26 |    3 |  3 | Takahashi |   28 |    5 |
|  5 | Watanabe  |   30 |    2 |  3 | Takahashi |   28 |    5 |
|  1 | Sato      |   23 |    1 |  4 | Tanaka    |   26 |    3 |
|  2 | Suzuki    |   24 |    2 |  4 | Tanaka    |   26 |    3 |
|  3 | Takahashi |   28 |    5 |  4 | Tanaka    |   26 |    3 |
|  4 | Tanaka    |   26 |    3 |  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |  4 | Tanaka    |   26 |    3 |
|  1 | Sato      |   23 |    1 |  5 | Watanabe  |   30 |    2 |
|  2 | Suzuki    |   24 |    2 |  5 | Watanabe  |   30 |    2 |
|  3 | Takahashi |   28 |    5 |  5 | Watanabe  |   30 |    2 |
|  4 | Tanaka    |   26 |    3 |  5 | Watanabe  |   30 |    2 |
|  5 | Watanabe  |   30 |    2 |  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+----+-----------+------+------+
```

このように、最初のテーブル（別名1）に対してすべての組み合わせが出力されます。このテーブルは5つのレコードがあったので、5×5＝25の結果が出力されています。

6.2 テーブルの操作

ここでは、テーブルや列、レコードの追加や削除、更新といった作業について取り上げます。

6.2.1 列の追加

テーブルを作成後に、列を新たに追加することができます。

```
ALTER TABLE テーブル名 ADD 新規列名 データ型 [FIRST];
```

この書式を使うと、テーブルの右端に列を追加することができます。次の例では、branchテーブルにtel欄を追加しています。

```
mysql> ALTER TABLE branch ADD tel VARCHAR(16);

mysql> DESC branch;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | smallint(6) | NO   | PRI |         |       |
| branchname | varchar(20) | YES  |     | NULL    |       |
| tel        | varchar(16) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

なお、次のように「FIRST」を指定すると、テーブルの右端ではなく左端に列が追加されます。

```
mysql> ALTER TABLE branch ADD tel VARCHAR(16)
FIRST;
```

6.2.2 列の削除

列を削除するには、以下の書式を使います。

```
ALTER TABLE テーブル名 DROP 列名;
```

次の例では、branchテーブルからtel列を削除します。削除の確認メッセージなどは表示されませんので注意して操作してください。

```
mysql> ALTER TABLE branch DROP tel;
```

6.2.3 レコードの更新

既存のレコードを更新するには、SQLコマンドのUPDATEを使います。

```
UPDATE テーブル名 SET 列名 = 値 [WHERE 条件]
```

次の例では、name列の値が「Sato」であるレコードのage列の値を「24」に変更します。

```
mysql> UPDATE staff SET age='24' WHERE name =
'Sato';

mysql> SELECT * FROM staff WHERE name =
'Sato';
+----+------+------+------+
| id | name | age  | bid  |
+----+------+------+------+
|  1 | Sato |   24 |    1 |
+----+------+------+------+
```

なお、WHERE句を指定しないと、すべてのレコードの列に同じ値が格納されてしまいますので注意してください。

```
mysql> UPDATE staff SET age = '20';

mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   20 |    1 |
|  2 | Suzuki    |   20 |    2 |
|  3 | Takahashi |   20 |    5 |
|  4 | Tanaka    |   20 |    3 |
|  5 | Watanabe  |   20 |    2 |
+----+-----------+------+------+
```

6.2.4 テーブルのコピー

テーブルの構造をコピーして新たなテーブルを作ることができます。

```
CREATE TABLE 新規テーブル名 LIKE コピー元テーブル名;
```

次の例では、staffテーブルと同じ構造である空のテーブルstaff3を作成します。

```
mysql> CREATE TABLE staff3 LIKE staff;
```

構造を比較してみます。

```
mysql> DESC staff;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | smallint(6) | NO   | PRI |         |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| age   | tinyint(4)  | YES  |     | NULL    |       |
| bid   | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+

mysql> DESC staff3;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | smallint(6) | NO   | PRI |         |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| age   | tinyint(4)  | YES  |     | NULL    |       |
| bid   | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

なお、空のテーブルではなく、レコードも含めてテーブルをコピーしたい場合は、次の書式を使います。

```
CREATE TABLE 新規テーブル名 SELECT * FROM コピー元テーブル名;
```

6.2.5 レコードとテーブルの削除

テーブルを削除するには、SQLコマンドのDROP TABLEを使います。

```
DROP TABLE テーブル名;
```

次の例では、staff3テーブルを削除しています。

```
mysql> DROP TABLE staff3;
```

レコードを削除するには、SQLコマンドのDELETEを使います。

```
DELETE FROM テーブル名 [WHERE 条件];
```

次の例では、id列の値が「5」のレコードを削除しています。

```
mysql> DELETE FROM staff WHERE id='5';
```

WHERE句を省略すると、すべてのレコードを削除します。次の例では、staffテーブルのレコードをすべて削除します。ただし、テーブルの構造はそのまま残ります。

```
mysql> DELETE FROM staff;
```

全レコードを削除する場合は、DELETE FROMの代わりにTRUNCATEコマンドを使った方が高速です。

```
mysql> TRUNCATE staff;
```

6.2.6 テーブル名の変更

テーブルを作成後にテーブル名を変更することができます。

```
ALTER TABLE テーブル名 RENAME 新規テーブル名;
```

次の例では、staff3テーブルの名前を「newstaff」に変更しています。

```
mysql> ALTER TABLE staff3 RENAME newstaff;
```

第6章 テスト

問　題　1

2つのテーブルを横方向に結合する方法は2つあります。2つの結合はそれぞれ何と呼びますか？

問　題　2

ある条件で検索したデータを使ってさらに検索を行うとき、SELECT文の中でSELECT文を使います。これを何と呼びますか？

問　題　3

レコードを更新するためのSQLコマンドを記述してください。

MySQL

第7章

トランザクション、参照整合性

7.1 トランザクション

データベースの特性である「データの整合性」を保持するために導入されている仕組みがトランザクション処理です。

7.1.1 トランザクションとは

第1章でも触れましたが、分割不可能なデータベース処理の単位をトランザクションといいます。銀行の口座Aから口座Bへ送金する処理を例に取ると、その処理は「口座Aからの出金」「口座Bへの入金」という2つの処理から構成されていると見ることができます。しかし、これらの処理は、片方だけ行われると、つじつまが合わなくなってしまいます。そのような不整合を防ぐために、複数の処理をまとめて扱い、いずれかが失敗すれば全体をなかったことにするのがトランザクション処理です。

DBMSがトランザクション処理をするための要素として、4つの特性が挙げられます。頭文字を取ってACID特性と呼ばれます。

◆原子性（Atomicity）

トランザクション内の処理がすべて完全に実行されるか、もしくは一つも実行されないか、いずれかの状態になることを原子性といいます。トランザクションを確定することをコミット、トランザクション中の処理を取り消すことをロールバックといいます。

◆一貫性（Consistency）

トランザクションの前後でデータベースの整合性が保たれ、データが首尾一貫としていることを一貫性といいます。

◆独立性（Isolation）

トランザクション内の処理が外部から隠蔽されていて、他の処理に影響を与えないことを独立性といいます。

◆耐久性（Durability）

一度確定（コミット）されたトランザクションは、システム障害が発生しても失われることがないことを耐久性といいます。

7.1.2 ロックと排他制御

トランザクション内でのデータの一貫性を実現するために使われているのがロックと排他制御です。ロックとは、データベース内の特定のデータへのアクセスを一時的に制限することです。たとえば、あるデータへの書き換えを行っているときに、同じデータが別の処理によって書き換えられてしまうと、矛盾が生じてしまいます。そこで、データへの書き換えを行っているときにはそのデータをロックし、他の処理が書き込みできないようにしておきます。これを排他制御といいます。

MySQLの場合、テーブルレベルのロックと行レベルのロックが利用できます。

◆テーブルレベルのロック

対象データを含むテーブルすべてをロックします。

◆行レベルのロック

対象データを含む行をロックします。

7.1.3 MySQLにおけるトランザクション

MySQLでは複数のストレージエンジンを使うことができます（第11章参照）。デフォルトのストレージエンジンであるMyISAMでは、トランザクション機能を利用することができません。InnoDBなど、トランザクションに対応したストレージエンジンを使用する必要があります。

ストレージエンジンはテーブルごとに設定できます。たとえば、staffテーブルをInnoDBに変更したい場合は、次のコマンドを実行します。

```
mysql> ALTER TABLE staff ENGINE=InnoDB;
```

トランザクションでは複数のSQL文を1セットとして扱えますが、デフォルトではSQL文を実行する度にコミットが行われます。これをAUTO COMMITモードといいます。このモードでは、BEGINコマンドもしくはSTART TRANSACTIONコマンドを実行すると、トランザクション機能が有効になります。トランザクションはCOMMITコマンドを実行するまで有効です。

```
mysql> START TRANSACTION;
```

何かSQL文を実行してみます。

```
mysql> INSERT INTO staff VALUES ('6',
'Transaction', '99', '1');

mysql> SELECT * FROM staff;
+----+-------------+------+------+
| id | name        | age  | bid  |
+----+-------------+------+------+
|  1 | Sato        |   23 |    1 |
|  2 | Suzuki      |   24 |    2 |
|  3 | Takahashi   |   28 |    5 |
|  4 | Tanaka      |   26 |    3 |
|  5 | Watanabe    |   30 |    2 |
|  6 | Transaction |   99 |    1 |
+----+-------------+------+------+
```

ここで、確定（コミット）をせず、操作の取り消し（ロールバック）をしてみます。ロールバックはROLLBACKコマンドで行います。

```
mysql> ROLLBACK;

mysql> SELECT * FROM staff;
+----+-----------+------+------+
| id | name      | age  | bid  |
+----+-----------+------+------+
|  1 | Sato      |   23 |    1 |
|  2 | Suzuki    |   24 |    2 |
|  3 | Takahashi |   28 |    5 |
|  4 | Tanaka    |   26 |    3 |
|  5 | Watanabe  |   30 |    2 |
+----+-----------+------+------+
```

追加されたレコードが取り消されているのが分かります。次に、コミット後にロールバックを実施してみます。コミットするにはCOMMITコマンドを実行します。

```
mysql> START TRANSACTION;

mysql> INSERT INTO staff VALUES ('6',
'Transaction', '99', '1');

mysql> SELECT * FROM staff;
+----+-------------+------+------+
| id | name        | age  | bid  |
+----+-------------+------+------+
|  1 | Sato        |   23 |    1 |
|  2 | Suzuki      |   24 |    2 |
|  3 | Takahashi   |   28 |    5 |
|  4 | Tanaka      |   26 |    3 |
|  5 | Watanabe    |   30 |    2 |
|  6 | Transaction |   99 |    1 |
+----+-------------+------+------+

mysql> COMMIT;

mysql> ROLLBACK;

mysql> SELECT * FROM staff;
+----+-------------+------+------+
| id | name        | age  | bid  |
+----+-------------+------+------+
|  1 | Sato        |   23 |    1 |
|  2 | Suzuki      |   24 |    2 |
|  3 | Takahashi   |   28 |    5 |
|  4 | Tanaka      |   26 |    3 |
|  5 | Watanabe    |   30 |    2 |
|  6 | Transaction |   99 |    1 |
+----+-------------+------+------+
```

ロールバックを行っても、コミット時点でトランザクションが確定しているので、テーブルの情報は何も更新されていないのが分かります。

なお、AUTO COMMITモードを無効にすると、常時トランザクションが行われるようになるので、BEGINコマンドもしくはSTART TRANSACTIONコマンドでトランザクションの開始を宣言する必要がなくなります。AUTO COMMITモードを無効にするには、次のコマンドを実行します。

```
mysql> SET AUTOCOMMIT=0;
```

AUTO COMMITモードを有効にするには、次のコマンドを実行します。

```
mysql> SET AUTOCOMMIT=1;
```

7.2 参照整合性

あるテーブルの列が別のテーブルの列を参照しているとき、参照関係が正しく維持できている必要があります。そのための仕組みが参照整合性制約です。

7.2.1 外部キーとは

別のテーブルで主キーとなっている列のことを外部キーといいます。たとえば次の例を見てください。

```
mysql> DESC staff;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | smallint(6) | NO   | PRI |         |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| age   | tinyint(4)  | YES  |     | NULL    |       |
| bid   | smallint(6) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+

mysql> DESC branch;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | smallint(6) | NO   | PRI |         |       |
| branchname | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+

mysql> SELECT * FROM staff JOIN branch ON staff.bid = branch.id;
+----+-----------+------+------+----+------------+
| id | name      | age  | bid  | id | branchname |
+----+-----------+------+------+----+------------+
|  1 | Sato      |   23 |    1 |  1 | Tokyo      |
|  2 | Suzuki    |   24 |    2 |  2 | Yokohama   |
|  3 | Takahashi |   28 |    1 |  1 | Tokyo      |
|  4 | Tanaka    |   26 |    3 |  3 | Osaka      |
|  5 | Watanabe  |   30 |    2 |  2 | Yokohama   |
+----+-----------+------+------+----+------------+
```

staffテーブルのbid列とbranchテーブルのid列を関連づけていますが、branchテーブルのid列は主キーとなっています。staffテーブルのbid列は外部キーとなっています。

7.2.2 参照整合性制約

先の例では、branchテーブルのid列にある値が、staffテーブルのbid列にないと、その部分はNULLとなってしまいます。

```
mysql> SELECT * FROM staff RIGHT JOIN branch ON staff.bid =
branch.id;
+------+----------+------+------+----+------------+
| id   | name     | age  | bid  | id | branchname |
+------+----------+------+------+----+------------+
|    1 | Sato     |   23 |    1 |  1 | Tokyo      |
|    2 | Suzuki   |   24 |    2 |  2 | Yokohama   |
|    5 | Watanabe |   30 |    2 |  2 | Yokohama   |
|    4 | Tanaka   |   26 |    3 |  3 | Osaka      |
| NULL | NULL     | NULL | NULL |  4 | Nagoya     |
+------+----------+------+------+----+------------+
```

そのようなことがないように、外部キーとなるbranchテーブルのid列の値が、staffテーブルのbid列から参照可能である状態を保持することを参照整合性制約といいます。

参照整合性制約を設定するには、テーブル作成時に、外部キーを「FOREIGN KEY」と指定し、「REFERENCES」に続いて参照先のテーブルと列を指定します。

```
mysql> CREATE TABLE staff (
    -> id VARCHAR(4),
    -> name VARCHAR(20),
    -> age INT,
    -> bid VARCHAR(4),
    -> FOREIGN KEY bid REFERENCES branch (id)
    -> ON DELETE RESTRICT ON UPDATE RESTRICT;
    -> )
    -> ENGINE=InnoDB
    -> ;
```

「ON DELETE」や「ON UPDATE」は、参照元テーブルのデータが削除もしくはアップデートされた際に、参照先データをどうするかを指定します。「RESTRICT」を指定するとエラーを返します。「SET NULL」を指定するとNULL値を挿入します。「CASCADE」を指定すると、変更の場合は参照元と同様に変更、削除の場合は同じように削除、となります。

第7章 テスト

問　題　1

トランザクションの4つの特性を挙げてください。

問　題　2

トランザクション処理を確定することを何と呼びますか？

問　題　3

トランザクション処理中に処理を取り消すことを何と呼びますか？

問　題　4

別のテーブルで主キーとなっている列のことを何と呼びますか？

MySQL

第8章

データベース設計の基礎(1)

8.1 データベースの設計

データベースの設計は、適切な手順と手法を取ることで、品質を高めることができます。

8.1.1 データモデリング

データベースで管理される情報は、現実世界の情報をコンピュータ上に移したものといえます。その作業をデータモデリングといいます。抽象度の高いものから、概念データモデル、論理データモデル、物理データモデルに分類されます。

［図8-1］データモデル

論理データモデル
概念データモデル
物理データモデル

◆概念データモデル

システムの要件に基づいて業務を分析し、データベースで扱う対象となる情報やデータ構造を表したものです。代表的な概念データモデルには、ERモデルやオブジェクト指向モデルがあります。システム開発工程では、基本計画で扱います。

◆論理データモデル

概念データモデルから、実際にコンピュータ上で実現できるモデルを表したものです。代表的な論理データモデルには、階層型、関係型、ネットワーク型があります。もっとも一般的なのは関係型（リレーショナル型）です。システム開発工程では、外部設計で扱います。

◆物理データモデル

実際に利用するコンピュータ、磁気ディスク、データベースソフトウェアの仕様を考慮したモデルです。

8.1.2 概念データモデルと論理データモデルの作成

データモデルの作成には、トップダウン型アプローチとボトムアップ型アプローチがあります。それぞれに一長一短がありますので、うまく組み合わせて使うことが大切です。

◆トップダウン型アプローチ

トップダウン型アプローチでは、概念データモデルを作成してから、徐々に詳細化していくデータモデリング手法です。現行システムがない状態において理想的な設計ができるかもしれませんが、現在のユーザーニーズが的確に反映されない可能性があります。具体的な手順は次のとおりです。

①データの理想型を元にER図を使って概念データモデルを表す
②ER図からテーブル設計を行う
③テーブルの正規化を行う

◆ボトムアップ型アプローチ

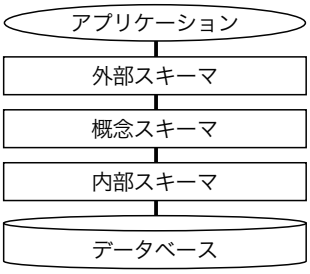
既存システムの画面や帳票に基づいて必要な情報を抽象化し、その結果に基づいて概念データモデルを作成するデータモデリング手法です。現行の業務システムに基づくため、ユーザーニーズを反映させやすいですが、将来的な拡張性や柔軟性が不十分になることがあります。具体的な手順は次のとおりです。

①現行システムからテーブル設計を行う
②テーブルの正規化を行う
③既存システムの概念データモデルをER図で表し、変更を反映させる

8.1.3 3層スキーマ

データベースを利用する側（ユーザーやアプリケーション）からデータを独立させるため、データベース構造の定義であるスキーマを3層に分けて考えます。この考え方はANSI/SPARC（米国規格協会標準化計画委員会）で提案されたもので、概念スキーマ、外部スキーマ、内部スキーマから構成されます。

［図8-2］3層スキーマ



◆概念スキーマ

業務対象の論理構造を表現したものが概念スキーマです。リレーショナルデータベースではテーブル（表）の定義が概念スキーマに相当します。

◆外部スキーマ

データベースにアクセスする利用者やアプリケーションプログラム側からデータベースの構造を表現したものを外部スキーマといいます。リレーショナルデータベースではビューの定義が外部スキーマに相当します。

※表の演算結果を仮想的な表として表したものをビューといいます。

◆内部スキーマ

データの物理的な格納方法や媒体を表現したものを内部スキーマといいます。内部スキーマはデータベースソフトウェアやハードウェアによって異なります。

スキーマを3層に分けることにより、概念スキーマは変更されにくい、安定したものとなることができます。

8.2 ER図

ERモデルはリレーショナルデータベースにおいて重要な表現法です。ERモデルはER図で表されます。

8.2.1 ERモデル

データ構造をエンティティ（Entity）とリレーションシップ（Relationship）で表現するデータモデリングがERモデルです。リレーショナルデータベースの設計に広く利用されています。ERモデルは、エンティティ、リレーションシップ、アトリビュートの3つの要素があります。

◆エンティティ

データベースで管理すべきデータのまとまりをエンティティ（Entity：実体）といいます。リレーショナルデータベースでは表やレコードに相当します。具体的には、「社員」「銀行」「支店」といったものが挙げられます。エンティティを具体化したものをインスタンスといいます。

◆リレーションシップ

エンティティ間の関連を表すものをリレーションシップ（Relationship：関連）といいます。エンティティが名詞とすると、リレーションシップは動詞に相当します。具体的には、社員は会社に「所属する」といったものが挙げられます。

◆アトリビュート

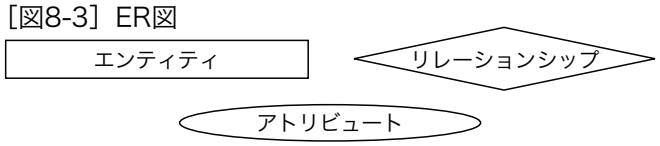
エンティティやリレーションシップの属性や状態をアトリビュートといいます。リレーショナルデータベースでは、列に相当します。具体的には、社員エンティティであれば「社員番号」「氏名」「住所」「生年月日」などが挙げられます。

8.2.2 ER図の記法

※ER図にはいくつかの表記法がありますので、本書とは異なる表記が使われている場合もあります。

ERモデルはER図（Entity-Relationship Diagram）で表記することができます。

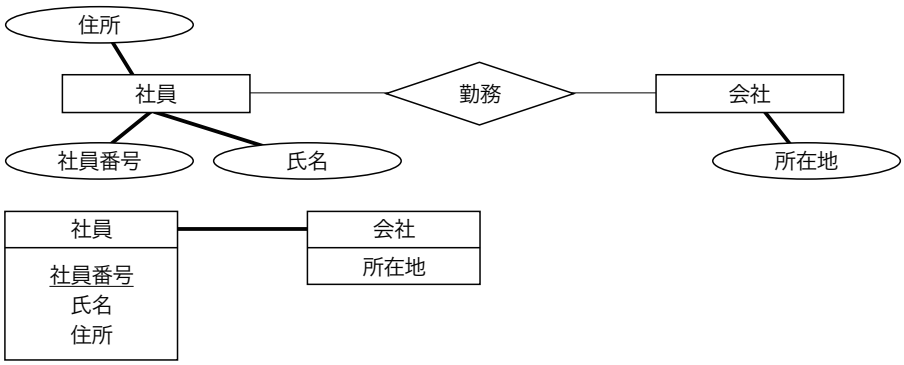
［図8-3］ER図



エンティティは長方形を描き、その中にエンティティ名を記述します。リレーションシップは菱形を描き、その中にリレーションシップ名を記述します。菱形の各頂点をエンティティと直線で繋ぎます。アトリビュートは楕円を描き、その中にアトリビュート名を記述します。また、エンティティやリレーションシップと直線で繋ぎます。

次に、ER図の具体例を示します。

［図8-4］ER図の具体例



8.2.3 カーディナリティ

リレーションシップには、エンティティの対応関係によって、次の3種類があります。

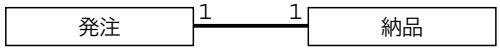
・1対1
・1対多
・多対多

このような対応関係をカーディナリティといいます。

◆1対1

エンティティAに対してエンティティBが1つだけしか存在せず、かつエンティティBに対してエンティティAも1つだけしか存在しないリレーションシップが「1対1」です。いずれかを指定すれば、他方も特定することができます。たとえば、「発注」と「納品」のカーディナリティは1対1となります。

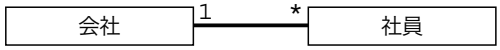
［図8-5］1対1



◆1対多

エンティティAに対してエンティティBは多数存在し、反対にエンティティBに対してエンティティAは一つだけしか存在しないリレーションシップが「1対多」です。多数ある側を指定すれば1つの方を特定できますが、その逆はできません。たとえば、「会社」と「社員」、「発注」と「商品」のカーディナリティは1対多となります。

［図8-6］1対多



◆多対多

エンティティAに対してエンティティBが多数存在し、エンティティBに対してエンティティAも多数存在するリレーションシップが「多対多」です。たとえば、「商品」と「顧客」、「顧客」と「注文」のカーディナリティは多対多となります。

［図8-7］多対多

第8章 テスト

問　題　1

3層スキーマを構成する3つのスキーマを挙げてください。

問　題　2

ERモデルの3つの要素を挙げてください。

問　題　3

エンティティどうしの対応関係を何と呼びますか？

MySQL

第9章

データベース設計の基礎(2)

9.1 正規化

データベースで適切に利用できるような形のテーブルを作るために必須となる作業が正規化です。

9.1.1 正規化とは

同じデータが複数箇所に登録されていると、すべての箇所を修正しなければ不整合が発生してしまいます。そのようなデータの冗長性をなくし、関連性の強い属性を集めることを正規化といいます。正規化を進めることで、データの追加や削除をしたときに不整合が発生してしまうことを避け、それによってデータの一貫性や整合性を保てるようになります。

ただし、正規化にはメリットばかりというわけではありません。正規化を進めるにしたがってデータが多くのテーブルに分散するため、データの検索には時間がかかってしまいます。そのため、あえて正規化をせず、冗長なままの状態を保つこともあります。これを非正規化といいます。

9.1.2 関数従属とは

正規化で大切となる概念、関数従属について説明しておきます。たとえば、次のような列で構成されるテーブルがあるとします。伝票番号と商品番号は主キーです（複合キー）。

伝票番号　注文日　顧客番号　顧客名　商品番号　数量　単価

ここで、伝票番号が決まれば、注文日や顧客番号が一意に決まるとき、「注文日（顧客番号）は伝票番号に関数従属している」といいます。関数従属は「伝票番号→注文日」「伝票番号→顧客番号」のように矢印で表します。

◆部分関数従属

関数従属の中で、主キーの一部に対して関数従属であることを部分関数従属といいます。この例では、注文日と顧客番号は伝票番号に部分関数従属です。また、商品番号が決まれば単価も決まるとすれば、単価は商品番号に部分関数従属です。

◆完全関数従属

商品の数量は、伝票番号と商品番号の両方が決まらなければ特定できません。このように、主キー全体に対して関数従属であり、部分関数従属でないことを完全関数従属といいます。

◆推移関数従属

顧客番号と顧客名が1対1で対応しているならば、わざわざレコードごとに顧客名を入れる必要はありません。伝票番号が決まれば顧客番号が決まり、顧客番号が決まれば顧客名が決まります。そのような場合、顧客名は伝票番号に推移関数従属しているといいます。

［図9-1］関数従属

9.1.3 第1正規形

まったく正規化されていない状態のテーブルを非正規形といいます。

[図9-2] 非正規形

銀行コード	支店コード	名称	銀行名	支店名	都道府県コード	都道府県名	電話番号
0021	001	牛久保銀行本店	牛久保銀行	本店	25	神奈川県	000-001-0001,000-001-0002
0021	002	牛久保銀行都筑支店	牛久保銀行	都筑支店	25	神奈川県	000-002-0001
0021	003	牛久保銀行中川支店	牛久保銀行	中川支店	24	東京都	001-001-0001

このテーブルを正規化していきましょう。まず、繰り返しの属性が存在しないフラットな状態にします。これを第1正規形といいます。第1正規形にすることを第1正規化といいます。図9-2では、1つめのレコードに電話番号が繰り返されていますので、これを分割します。第1正規形は図9-3のようになります。

[図9-3] 第1正規形

銀行コード	支店コード	名称	銀行名	支店名	都道府県コード	都道府県名	電話番号
0021	001	牛久保銀行本店	牛久保銀行	本店	25	神奈川県	000-001-0001
0021	001	牛久保銀行本店	牛久保銀行	本店	25	神奈川県	000-001-0002
0021	002	牛久保銀行都筑支店	牛久保銀行	都筑支店	25	神奈川県	000-002-0001
0021	003	牛久保銀行中川支店	牛久保銀行	中川支店	24	東京都	001-001-0001

9.1.4 第2正規形

第2正規形は、第1正規形であり、かつキー以外の列は主キーに対して完全関数従属になっている状態です。具体的には、主キーに完全関数従属している列を残し、部分関数従属する列を別のテーブルに分割します。

［図9-4］第2正規形

図9-4a

銀行コード	支店コード	支店名	都道府県コード	都道府県名
0021	001	本店	25	神奈川県
0021	002	都筑支店	25	神奈川県
0021	003	中川支店	24	東京都

図9-4b

銀行コード	支店コード	電話番号
0021	001	000-001-0001
0021	001	000-001-0002
0021	002	000-002-0001
0021	003	001-001-0001

図9-4c

銀行コード	銀行名
0021	牛久保銀行

この例では、主キーは銀行コードと支店コードです。これらに対して完全関数従属になっている「支店名」「都道府県コード」「都道府県名」を残して、図9-4bと図9-4cに分割します。

9.1.5 第3正規形

第3正規形は、第2正規形であり、かつキー以外の列が推移関数従属になっていない状態です。

［図9-5］第3正規形

図9-5a

銀行コード	支店コード	支店名	都道府県コード
0021	001	本店	25
0021	002	都筑支店	25
0021	003	中川支店	24

図9-5b

都道府県コード	都道府県名
25	神奈川県
24	東京都

都道府県名は、主キー（「銀行コード」「支店コード」）と都道府県コードが分かれば特定できますので、その部分を分割します。

以上で、図9-5a、図9-5b、図9-4b、図9-4cいずれもが第3正規形となりました。

第9章 テスト

問　題　1

正規化のメリットを2つ挙げてください。

問　題　2

正規化のデメリットを挙げてください。

問　題　3

まったく正規化されていないテーブルの状態を何と呼びますか？

MySQL

第10章

MySQLでのRDBシステム管理(1)

10.1 MySQLサーバの起動と停止

第3章では起動スクリプトを使ったMySQLサーバの起動と終了を紹介しましたが、ここでは更に詳しく取り上げます。

10.1.1 MySQLサーバの起動

MySQLサーバを起動する方法は3とおりあります。

◆起動スクリプトを利用する

他のサービスと同様、起動スクリプトを使って操作します。引数には「start」「stop」「restart」「status」が利用できます。

```
# /etc/init.d/mysqld start
```

◆mysqldを直接実行する

MySQLサーバプログラムであるmysqldをコマンドライン上で直接実行する方法もあります。実行ユーザーは--userオプションで指定します。バックグラウンドで動作するよう、末尾に「&」を付けて実行してください。

```
# /usr/libexec/mysqld --user=mysql &
```

mysqldでは多くのオプションが利用できます。オプションを確認するには、次のように--verboseオプションと--helpオプションを付けて実行します。

```
# /usr/libexec/mysqld --verbose --help
/usr/libexec/mysqld  Ver 5.0.45-log for redhat-linux-gnu on i686 (Source
distribution)
Copyright (C) 2000 MySQL AB, by Monty and others
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Starts the MySQL database server

Usage: /usr/libexec/mysqld [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf /etc/my.cnf
The following groups are read: mysqld server mysqld-5.0
The following options may be given as the first argument:
(以下省略)
```

オプションはコマンドラインで指定しても良いですが、設定ファイル（my.cnf）内の[mysqld]グループ内に記述する方がよいでしょう。

10.1.2 MySQLサーバの動作確認

MySQLサーバが動作しているかどうかは、次のようにして確認できます。

```
# /etc/init.d/mysqld status
mysqld (pid 2992) is running...
```

上の例ではMySQLサーバは動作しています。動作していないときは次のような表示になります。

```
# /etc/init.d/mysqld status
mysqld is stopped
```

psコマンドを使って、mysqldプロセスが動作していることを確認してもよいでしょう。

```
# ps aux | grep mysqld
root       2932  0.0  0.2   4528  1240 ?        S    06:47   0:00
/bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --socket=/
var/lib/mysql/mysql.sock --log-error=/var/log/mysqld.log --pid-
file=/var/run/mysqld/mysqld.pid
mysql      2992  0.0  3.3 136684 17080 ?        Sl   06:47   0:01
/usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql
--user=mysql --pid-file=/var/run/mysqld/mysqld.pid --skip-external-
locking --socket=/var/lib/mysql/mysql.sock
root       8929  3.0  0.1   3916   688 pts/0    S+   09:50   0:00
grep mysqld
```

10.2 mysqladminコマンド

mysqladminコマンドは、コマンドライン上でMySQLサーバの管理作業をするためのコマンドです。

10.2.1 mysqladminコマンドの基本

mysqladminコマンドの書式は次のとおりです。さまざまな命令が用意されています。

```
mysqladmin ［オプション］ ［命令］
```

主なオプションと命令を表に示します。

［表10-1］mysqladminコマンドの主なオプション

オプション	説明
-h ホスト名	サーバのホスト名を指定する
--host=ホスト名	
-p パスワード	パスワードを指定する
--password=パスワード	
-u ユーザー	ユーザーを指定する
--user=ユーザー	

［表10-2］mysqladminコマンドの主な命令

コマンド	説明
ping	MySQLサーバの死活確認をする
status	MySQLサーバの状態を表示する
extended-status	MySQLサーバの状態を詳細に表示する
variables	システム変数を表示する
version	MySQLサーバのバージョンを表示する
create データベース名	データベースを作成する
drop データベース名	データベースを削除する
shutdown	MySQLサーバをシャットダウンする
processlist	スレッド一覧を表示する
kill <id>	指定したidのスレッドをkillする
password パスワード	パスワードを変更する
refresh	すべてのテーブルとログファイルを開き直す
flush-tables	テーブルのデータをディスクにフラッシュする

10.2.2 MySQLサーバの情報確認

MySQLサーバの死活確認をするには、ping命令を指定します。動作していれば「mysqld is alive」と表示されます。

```
# mysqladmin -u root -p ping
Enter password:
mysqld is alive
```

status命令を指定すると、MySQLサーバの状態を表示できます。

```
# mysqladmin -u root -p status
Enter password:
Uptime: 117061  Threads: 1  Questions: 46  Slow
queries: 0  Opens: 19
Flush tables: 1  Open tables: 13  Queries per
second avg: 0.000
```

10.2.3 データベースの作成と削除

create命令を使うと、データベースを作成します。これは、MySQLモニタ上でCREATE DATABASE文を実行するのと同じです。次の例では、testdb2データベースを作成しています。

```
# mysqladmin -u root -p create testdb2
```

drop命令を使うと、データベースを削除します。これは、MySQLモニタ上でDROP DATABASE文を実行するのと同じです。次の例では、testdb2データベースを削除しています。

```
# mysqladmin -u root -p drop testdb2
```

10.2.4 MySQLサーバのプロセス処理

shutdown命令を指定すると、MySQLサーバをシャットダウンできます。特にメッセージは表示されません。

```
# mysqladmin -u root -p shutdown
```

MySQLのスレッドを一覧表示するには、processlist命令を指定します。

```
# mysqladmin -u root -p processlist
Enter password:
+----+------+-----------+----+---------+------+-------+------------------+
| Id | User | Host      | db | Command | Time | State | Info             |
+----+------+-----------+----+---------+------+-------+------------------+
| 2  | root | localhost |    | Query   | 0    |       | show processlist |
+----+------+-----------+----+---------+------+-------+------------------+
```

任意のスレッドをkill命令で終了させることもできます。次の例では、3番のスレッドを終了させています。

```
# mysqladmin -u root -p kill 3
```

10.2.5 その他操作

※コマンド履歴に残る場合は、history -cコマンドを実行するなどして履歴から消去しておいた方がよいでしょう。

ユーザーのパスワードを再度設定するには、password命令とともに新しいパスワードを指定します。次の例では、rootユーザーのパスワードにsecretを指定しています。

```
# mysqladmin -u root -p password secret
```

refresh命令を使うと、すべてのテーブルとログファイルをいったん閉じた後で新たに開きます。

```
# mysqladmin -u root -p refresh
```

flush-tables命令を使うと、開いているテーブルを閉じます。その際、メモリ上にキャッシュされているデータがディスクにフラッシュされます。

```
# mysqladmin -u root -p flush-tables
```

10.2.6 システム変数

MySQLサーバの動作を規定するさまざまなシステム変数の一覧を確認するには、variables命令を実行します。

※グローバルパラメータということもあります。

```
# mysqladmin -u root -p variables
Enter password:
+---------------------------------+-----------------------------------+
| Variable_name                   | Value                             |
+---------------------------------+-----------------------------------+
| auto_increment_increment        | 1                                 |
| auto_increment_offset           | 1                                 |
| automatic_sp_privileges         | ON                                |
| back_log                        | 50                                |
| basedir                         | /usr/                             |
| bdb_cache_size                  | 8388600                           |
| bdb_home                        | /var/lib/mysql/                   |
| bdb_log_buffer_size             | 32768                             |
| bdb_logdir                      |                                   |
| bdb_max_lock                    | 10000                             |
| bdb_shared_data                 | OFF                               |
| bdb_tmpdir                      | /tmp/                             |
| binlog_cache_size               | 32768                             |
| bulk_insert_buffer_size         | 8388608                           |
| character_set_client            | latin1                            |
| character_set_connection        | latin1                            |
| character_set_database          | sjis                              |
| character_set_filesystem        | binary                            |
| character_set_results           | latin1                            |
| character_set_server            | latin1                            |
| character_set_system            | utf8                              |
| character_sets_dir              | /usr/share/mysql/charsets/        |
| collation_connection            | latin1_swedish_ci                 |
| collation_database              | sjis_japanese_ci                  |
| collation_server                | latin1_swedish_ci                 |
(以下省略)
```

主なシステム変数の意味は次表のとおりです。

[表10-3] 主なシステム変数

システム変数	説明
key_buffer_size	MyISAM用のインデックス用に使用されるメモリ上のバッファサイズ
max_connections	同時に接続できるクライアントの最大数
query_cache_limit	問い合わせをキャッシュする最大サイズ（デフォルトは1M）
query_cache_size	問い合わせをキャッシュするサイズ（デフォルトは0＝無効）
thread_cache_size	再利用のためにキャッシュに保持するスレッド数

システム変数を一時的に変更するには、SET GLOBALコマンドを使います。たとえば、最大同時接続数を120に設定するには、次のコマンドを実行します。

```
mysql> SET GLOBAL max _ connections=120;
```

永続的に変更するには、設定ファイルmy.cnfに記述しておきます。

10.3 データベースユーザーの権限設定

システムのユーザーとは別に、MySQL用のユーザー（以下、データベースユーザー）があります。MySQLサーバに接続するには、あらかじめユーザーの登録と権限の設定が必要です。

10.3.1 データベースユーザーの権限

MySQLでは、システムのユーザーアカウントとは別に、MySQLサーバへ接続する際のデータベースユーザーが登録されています。データベースユーザーはユーザー名とパスワード、接続元ホスト（IPアドレス）によって識別されます。MySQLサーバに接続すると、あらかじめ設定された権限がデータベースユーザーに付与され、与えられた権限内でさまざまな操作をすることができます。権限には、次表のものがあります。

※ユーザー名が同じでも、ホストが異なれば別のデータベースユーザーとして認証されます。

［表10-4］MySQLの権限の例

権限	許可される権限
ALL	すべての権限
ALL PRIVILEGES	ALLと同じ
SELECT	SELECT文の実行
UPDATE	UPDATE文の実行
INSERT	INSERT文の実行
DELETE	DELETE文の実行
CREATE	データベースとテーブルの作成
DROP	データベースとテーブルの削除
ALTER	テーブルや列の変更
GRANT	他のユーザーの権限変更
CREATE USER	新規ユーザーの作成
USAGE	一切の権限なし

10.3.2 データベースユーザーの登録

デフォルトでは、MySQLのデータベースユーザーはrootユーザーのみが登録されています。rootユーザーは管理者として使われるユーザーですが、インストール直後の状態ではパスワードが設定されていませんので、セキュリティ上、パスワードは必ず設定してください。

データベースユーザーを登録するには、GRANT文を使います。

```
GRANT 権限 ON データベース名.テーブル名 TO ユーザー名[@ホスト名] IDENTIFIED BY 'パスワード';
```

次の例では、testdbデータベースのすべてのテーブルに対し、すべての権限があるdbuserユーザーを作成します。

```
mysql> GRANT ALL ON testdb.* TO 'dbuser'@'localhost' IDENTIFIED BY 'secret';
```

ここでは接続元ホスト名（@localhost）を指定しています。ホスト名を指定しない場合は、どのホストからでも接続できてしまうので、ホスト名も指定するようにしてください。

次の例では、172.16.0.3から接続できるdbtestユーザーに、testdbデータベースのstaffテーブルでSELECT文およびUPDATE文が実行できる権限を設定しています。

```
mysql> GRANT SELECT,UPDATE ON testdb.staff TO 'dbtest'@'localhost' IDENTIFIED BY 'secret';
```

10.3.3 データベースユーザーの権限確認

データベースユーザーの権限を確認するには、SHOW GRANTS文を使います。次の例では、dbtestユーザーの権限を確認しています。

```
mysql> SHOW GRANTS FOR 'dbtest'@'localhost';
+-----------------------------------------------------------------------------------------+
| Grants for dbtest@localhost                                                             |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'dbtest'@'localhost' IDENTIFIED BY PASSWORD '428567f408994404' |
| GRANT SELECT, UPDATE ON `testdb`.`staff` TO 'dbtest'@'localhost'                        |
+-----------------------------------------------------------------------------------------+
```

「USAGE」は「何も権限がない」状態を表します。つまり、すべてのデータベースに対して、何も権限を持っていない、というのが1行目の意味です。パスワードは暗号化されています。2行目に、先に設定した情報が表示されているのが確認できます。

10.3.4 権限とデータベースユーザーの削除

データベースユーザーに設定した権限を削除するにはREVOKE文を使います。

```
REVOKE 権限 ON データベース名.テーブル名 FROM ユーザー名[@ホスト名];
```

次の例では、dbtestユーザーからUPDATEの権限を削除します。

```
mysql> REVOKE UPDATE ON testdb.staff FROM 'dbtest'@'localhost';
```

データベースユーザー自体を削除するには、DROP USER文を使います。

```
DROP USER ユーザー名;
```

次の例では、dbtestユーザーを削除しています。

```
mysql> DROP USER 'dbtest'@'localhost';
```

10.3.5 パスワードの変更

データベースユーザー作成時に設定したパスワードを変更するには、SET PASSWORDコマンドを使います。

```
SET PASSWORD FOR ユーザー名[@ホスト名]=PASSWORD('新パスワード');
```

次の例では、dbtestユーザーのパスワードを「secret」に変更しています。

```
mysql> SET PASSWORD FOR 'dbtest'@'localhost'=PASSWORD('secret');
```

第10章 テスト

問　題　1

mysqladminコマンドを使って実現できるものに○をつけてください。

（　　）データベースの作成
（　　）システム変数の表示
（　　）MySQLサーバの死活監視
（　　）MySQLサーバの状態表示
（　　）MySQLサーバのシャットダウン

問　題　2

データベースユーザーに権限を設定するSQLコマンドを記述してください。

問　題　3

データベースユーザーの権限を削除するSQLコマンドを記述してください。

MySQL

第11章

MySQLでのRDBシステム管理(2)

11.1 MySQLの管理

前章に引き続き、MySQLサーバの管理を取り上げます。

11.1.1 MySQLサーバの情報確認

MySQLサーバのステータス情報を確認するには、STATUSコマンドを使います。バージョンやキャラクタセットを確認することができます。

```
mysql> STATUS
--------------
mysql  Ver 14.12 Distrib 5.0.45, for redhat-linux-gnu (i686) using
readline 5.0

Connection id:          3
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.0.45 Source distribution
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 3 days 16 hours 37 min 26 sec

Threads: 1  Questions: 39  Slow queries: 0  Opens: 17  Flush
tables: 1  Open tables: 8  Queries per second avg: 0.000
--------------
```

SHOW STATUSを使うと、利用中のクライアントの統計情報が確認できます。これは、mysqladminコマンドでextended-status命令を指定するのと同じです。また、サーバ全体の統計情報はSHOW GLOBAL STATUSで確認できます。

```
mysql> SHOW STATUS;
+-----------------------------------+----------+
| Variable_name                     | Value    |
+-----------------------------------+----------+
| Aborted_clients                   | 0        |
| Aborted_connects                  | 1        |
| Binlog_cache_disk_use             | 0        |
| Binlog_cache_use                  | 0        |
| Bytes_received                    | 128      |
| Bytes_sent                        | 162      |
| Com_admin_commands                | 0        |
| Com_alter_db                      | 0        |
| Com_alter_table                   | 0        |
| Com_analyze                       | 0        |
| Com_backup_table                  | 0        |
(以下省略)
```

11.1.2 バックアップとリストア

データベースのバックアップは、データベース単位で行います。mysqldumpコマンドを実行すると、データベースの内容をSQLとしてテキストファイルで出力します。

```
mysqldump -u ユーザー名 -p[パスワード] データベース名 > 出力ファイル名
```

次の例では、testdbデータベースの内容を、testdb.txtとして出力します。パスワードをコマンドライン上で指定する場合は、-pの直後にスペースを空けずに指定してください。-pオプションのみを指定すると、対話的にパスワードを入力することになります。

```
# mysqldump -u root -p testdb > testdb.txt
```

出力されるファイルには、SQL文が記述されています。以下は出力ファイルの例（一部）です。

```
--
-- Table structure for table `branch`
--

DROP TABLE IF EXISTS `branch`;
CREATE TABLE `branch` (
  `id` smallint(6) NOT NULL,
  `branchname` varchar(20) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=sjis;
```

バックアップしたデータベースをリストアするには、最初に空のデータベースを作成し、次にバックアップしたファイルを実行します。

```
mysql -u ユーザー名 -p[パスワード] データベース名 < バックアップファイル名
```

次の例では、空のtestdbデータベースを作成し、リストアしています。

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

mysql> quit
Bye

# mysql -u root -p testdb < testdb.txt
```


11.2 MySQLの構造

ここでは、MySQLの大きな特徴であるストレージエンジンと、設定ファイル、ログファイルについて取り上げます。

11.2.1 ストレージエンジン

MySQLの構造は、大きく分けると次の2つがあります。

・SQLの受け付けやデータベース接続を司る部分
・データの格納や検索を処理する部分

後者をストレージエンジンといいます。MySQLでは複数のストレージエンジンが用意されており、用途に合わせて使い分けることができます。

[表11-1] ストレージエンジン

ストレージエンジン	説明
MyISAM	デフォルトで使用される。高速だがトランザクションに対応していない。
InnoDB	トランザクションに対応している。
MEMORY	データをファイルではなくメモリ上に格納するので高速。
MERGE	複数のテーブルを1つのテーブルのように利用できる。
CSV	カンマ区切りのCSV形式でデータを格納する。
ARCHIVE	INSERTとSELECTのみ利用できる。
FEDERATED	他のMySQLサーバにデータを格納する。
NDB	クラスタ構成で利用する。

デフォルトはMyISAMです。高速ですが、トランザクションに対応していません。トランザクションを利用したい場合は通常、InnoDBを利用します。

11.2.2 ストレージエンジンの変更

ストレージエンジンは、テーブルごとに設定することができます。

```
ALTER TABLE テーブル名 ENGINE=ストレージエンジン名;
```

たとえば、staffテーブルをInnoDBに変更したい場合は、次のコマンドを実行します。

```
mysql> ALTER TABLE staff ENGINE=InnoDB;
```

現在使われているストレージエンジンを確認するには、次のコマンドを実行し、「ENGINE=」の部分を確認します。

```
mysql> SHOW CREATE TABLE staff;

+-------+-----------------------------------------
----------------------------------------------------
----------------------------------------------------
----------------------------------------------------
-------------------+
| Table | Create Table

                                                  |
+-------+-----------------------------------------
----------------------------------------------------
----------------------------------------------------
----------------------------------------------------
-------------------+
| staff | CREATE TABLE `staff` (
  `id` smallint(6) NOT NULL,
  `name` varchar(20) default NULL,
  `age` tinyint(4) default NULL,
  `bid` smallint(6) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=sjis |
+-------+-----------------------------------------
----------------------------------------------------
----------------------------------------------------
----------------------------------------------------
-------------------+
```

この例では「ENGINE=MyISAM」となっています。以下、主なストレージエンジンを紹介します。

◆MyISAM

MySQLのデフォルトとして使われるストレージエンジンです。トランザクション機能はありませんが、読み出しが高速です。

◆InnoDB

トランザクション機能が利用できるストレージエンジンです。また、行ロックや外部キーもサポートしています。トランザクション機能が必要な場合に利用します。

◆MEMORY

データをすべてメモリ上に保持するので高速ですが、MySQLサーバが停止するとデータは消えてしまいます（テーブル定義は残ります)。

11.2.3 設定ファイル

※Windowsではmy.iniとなります。

※/usr/share/mysql/*cnfとして設定ファイルのひな形が用意されています。

MySQLサーバの設定ファイルは、/etc/my.cnfです。

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Default to using old password format for
compatibility with mysql 3.x
# clients (those using the mysqlclient10
compatibility package).
old _ passwords=1

[mysqld _ safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

もしサーバ側のデフォルトのキャラクタセットを指定したい場合は、「[mysqld]」セクション内に以下のように記述してください。

```
default-character-set=utf8
```

11.2.4 関連ファイル

データベースの物理的なファイルは、/etc/my.cnfの「datadir」に設定したディレクトリ内に、データベースごとにサブディレクトリに分かれて保存されます。たとえば、testdbデータベースの場合、デフォルトでは/var/lib/mysql/testdbディレクトリが使われます。

```
# ls /var/lib/mysql/testdb/
branch.frm   branch.MYD   branch.MYI   db.opt
staff.frm   staff.MYD   staff.MYI
```

～.frmにはフィールド定義などの情報が格納されます。それ以外のファイルは、ストレージエンジンごとに異なります。MyISAMの場合、～.MYDファイルに実際のデータが、～.MYIファイルにはテーブルのインデックスが格納されます。

11.2.5 ログファイル

MySQLのログにはいくつかの種類があります。

[表11-2] ログファイル

ログの種類	説明
エラーログ	MySQLサーバの起動や停止などのメッセージが記録される
スロークエリログ	処理に時間のかかった問い合わせのみ記録される
一般ログ	すべての操作が記録される
バイナリログ	更新SQL文だけが記録される
debugログ	開発者向けの詳細なログ

以下、主なログファイルを紹介します。

◆エラーログ

MySQLサーバの起動､終了メッセージなどが記録されます｡ログファイルは､/etc/my.cnfの「log-error」で設定するか、mysqld起動時に「--log-error=ログファイル名」で指定します。CentOSでは、/var/log/mysqld.logファイルにログが記録されます。

```
090501 23:53:25  mysqld started
090501 23:53:25  InnoDB: Started; log sequence number 0 43655
090501 23:53:25 [Note] /usr/libexec/mysqld: ready for connections.
Version: '5.0.45'   socket: '/var/lib/mysql/mysql.sock'   port: 3306
Source distribution
090502  0:13:46 [Note] /usr/libexec/mysqld: Normal shutdown

090502  0:13:46  InnoDB: Starting shutdown...
090502  0:13:49  InnoDB: Shutdown completed; log sequence number 0
43655
090502  0:13:49 [Note] /usr/libexec/mysqld: Shutdown complete

090502 00:13:49  mysqld ended
```

◆スロークエリログ

処理に一定以上の時間（デフォルトでは10秒以上）がかかった問い合わせのみを記録します。Slow Queryログを出力するには、mysqld起動時に「--log-slow-queries=ログファイル名」と指定するか、設定ファイルで以下のように記述します。

```
log-slow-queries = slow.log
```

ファイル名を指定しなかった場合は、datadirで指定したディレクトリ以下に、「ホスト名-slow.log」という名前でログファイルが作成されます。

閾値となる時間は、--log-query-timeオプションで指定できます。設定ファイルに記述する場合は、次のようにします。

```
log-query-time = 5
```

◆一般ログ

詳細ログには、接続元のホストやユーザー、発行された問い合わせなどが記録されます。詳細ログを出力するには、mysqld起動時に「-l」もしくは「--log=ログファイル名」と指定するか、設定ファイルで以下のように記述します。出力量が多いので、一般的には開発環境で利用されます。ログファイル名は、デフォルトでは「ホスト名.log」となります。

```
--log = mysqld-detail.log
```

以下は詳細ログの出力例です。

```
/usr/libexec/mysqld, Version: 5.0.45-log (Source distribution).
started with:
Tcp port: 0  Unix socket: /var/lib/mysql/mysql.sock
Time                 Id Command    Argument
090502 21:00:52       1 Connect     Access denied for user 'UNKN
OWN _ MYSQL _ US'@'localhost' (using password: NO)
090502 21:01:08       2 Connect     root@localhost on
                      2 Query     select @@version _ comment limit 1
090502 21:01:18       2 Query       SELECT * FROM staff
090502 21:01:24       2 Query       SELECT DATABASE()
                        2 Init DB     testdb
                        2 Query       show databases
                        2 Query       show tables
                        2 Field List  branch
                        2 Field List  city
                        2 Field List  members
                        2 Field List  staff
                        2 Field List  staff2
                        2 Field List  staff3
090502 21:01:26       2 Query       SELECT * FROM staff
090502 21:01:27       2 Quit
```

第11章 テスト

問　題　1

MySQLクライアントのコマンドとして適切な説明に○をつけてください。

（　　）STATUSコマンドで、MySQLサーバのステータスを表示する

（　　）SHOW STATUSコマンドで、MySQLサーバへ接続中のクライアントの統計情報が確認できる

問　題　2

MySQLデータベースをバックアップするコマンドを記述してください。

問　題　3

MySQLのストレージエンジンの説明として正しいものを選択してください。

A. SQLの受付やデータベース接続を司る

B. データの格納や検索を処理する

C. クラスタを構成する

D. データをメモリ上に展開する

問　題　4

Linuxにおける、MySQLサーバの設定ファイル名を記述してください。

問　題　5

MySQLのログファイル名をいくつか挙げてください。

MySQL

第12章

Webを使った RDBシステム管理

phpMyAdmin

GUIベースのMySQL管理ツールphpMyAdminを使ってみます。

phpMyAdmin

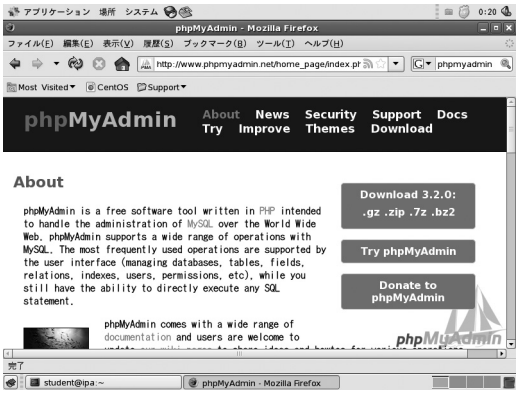
phpMyAdminは、Webブラウザ経由で利用する、オープンソースのMySQL管理ツールです。主な特徴は次のとおりです。

・データベースの作成、削除、コピー、リネーム等の管理
・テーブルの作成、削除、コピー、リネーム等の管理
・フィールドの追加、削除、修正
・SQL文の実行
・テキストファイルをテーブルにロード
・テーブルのダンプ出力と読み込み
・データをCSV、XML、PDF、OpenDocument形式で出力
・複数サーバの管理
・MySQLユーザーおよび権限の管理
・データベースレイアウトのPDFイメージ作成
・55言語に対応

phpMyAdminのインストール（ソース）

phpMyAdminは、公式サイト（http://www.phpmyadmin.net/）から最新版をダウンロードできます。本稿執筆時点では、バージョン3.2.0です。

［図］phpMyAdmin公式サイト



一般的なインストール手順は次のとおりです。
①ソースパッケージのダウンロード
②ソースパッケージの展開
③ドキュメントルート以下にディレクトリをコピー
④Webブラウザからインストール先ディレクトリにアクセス

phpMyAdminのインストール（RPM）

CentOSでは、サードパーティのリポジトリを追加することで、phpMyAdminをRPMパッケージでインストールすることができます。phpMyAdminのバージョン3.2.0では、バージョン5.2以降のPHPが要求されますが、CentOS 5.3ではその要件を満たしていないので、RPMパッケージでインストールする場合は、PHP関連のパッケージもあわせてアップデートすることになります。

まずは古いバージョンのPHPとMySQLをアンインストールしておきます。

```
# yum remove "php*" "mysql*"
```

次に、YUMのリポジトリを追加します。

```
# wget http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
# wget http://rpms.famillecollet.com/el5.i386/remi-release-5-6.el5.remi.noarch.rpm
# wget http://dag.wieers.com/rpm/packages/rpmforge-release/rpmforge-release-0.3.6-1.el5.rf.i386.rpm
# rpm -Uvh epel-release-5-3.noarch.rpm remi-release-5-6.el5.remi.noarch.rpm rpmforge-release-0.3.6-1.el5.rf.i386.rpm
```

必要なパッケージをインストールします。

```
# yum --enablerepo=remi,epel,rpmforge update "php*" "mysql*" phpMyAdmin -y
```

phpMyAdminの利用

phpMyAdminにはWebサーバが必要です。あらかじめApacheを起動しておきます。

```
# /etc/init.d/httpd start
```

次に、ブラウザで「http://127.0.0.1/phpmyadmin/」にアクセスすると、ユーザー認証ウィンドウが開かれます。

［図］ユーザー認証

MySQLユーザー名とパスワードを入力すると、トップ画面が表示されます。

［図］トップ画面

デフォルトでは、ローカルホスト以外からはアクセス禁止となっています。ローカルホスト以外からのアクセスを許可するには、/etc/httpd/conf.d/phpMyAdmin.confを設定し、Apacheを再起動してください。

/etc/httpd/conf.d/phpMyAdmin.conf（抜粋）

```
<Directory /usr/share/phpMyAdmin/>
   order deny,allow
   deny from all
   #allow from 127.0.0.1 ← この行をコメントアウト
   allow from 172.16.0.100     ← 許可したいIPアドレスを追加
</Directory>
```

phpMyAdminの機能

phpMyAdminのトップ画面はいくつかのタブから構成されています。代表的な画面を見ておきます。

◆データベース

データベースの一覧が表示されます。データベース名をクリックすると、詳細情報が表示されます。新規データベースの作成もこの画面から行えます。

［図］データベース

◆SQL

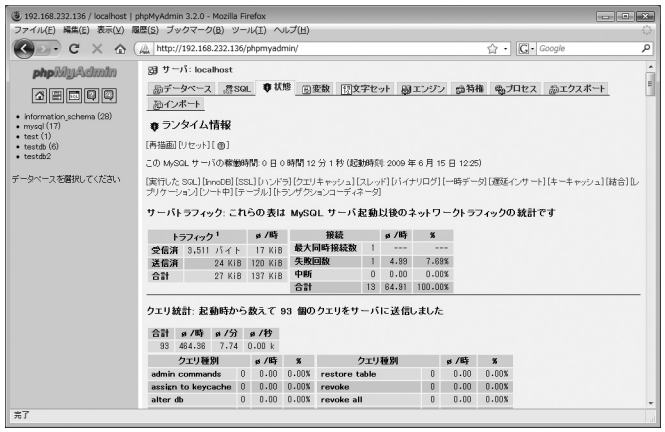
任意のSQLを実行できます。

［図］SQL

◆状態

サーバの状態と統計情報が表示されます。

［図］状態



◆変数

サーバ変数と設定値が表示されます。

［図］変数

◆エクスポート

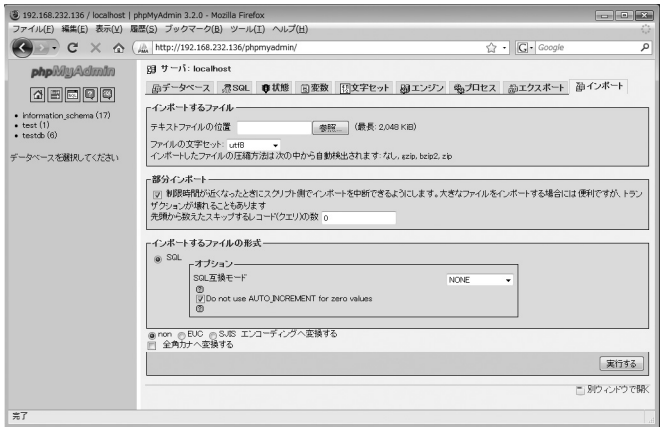
任意のデータベースをダンプします。デフォルトはSQLですが、さまざまな形式でエクスポートできます。

［図］エクスポート

◆インポート

ファイルからインポートします。

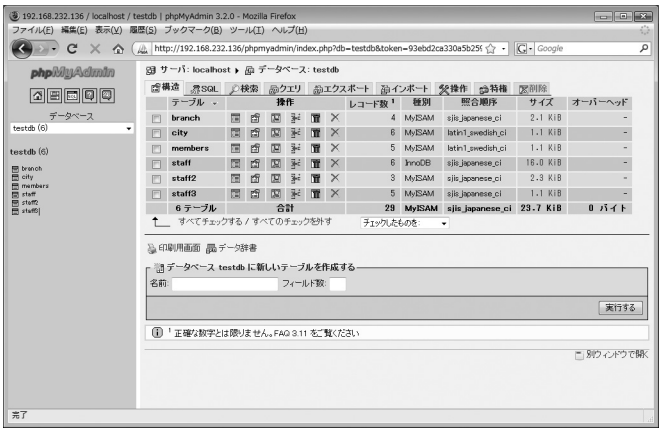
［図］インポート



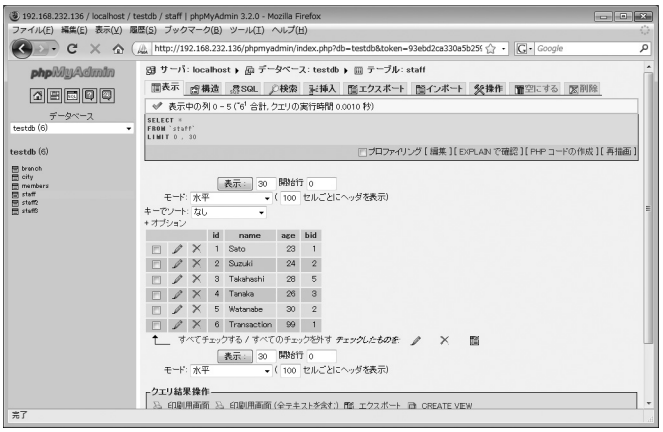
テーブル内容の表示

テーブル内容を表示するには、左側のペインからデータベースを選択し、表示したいテーブルの「表示」アイコンをクリックするだけです。

［図］テーブル一覧

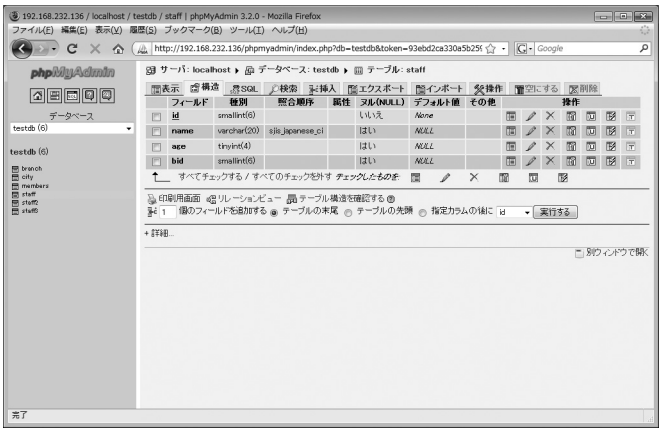


［図］テーブル内容



「構造」アイコンをクリックすると、テーブルの構造が表示されます。

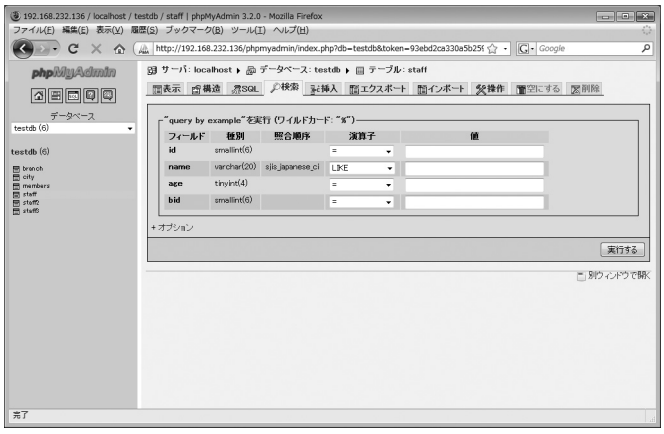
［図］テーブル構造の表示



レコードの検索

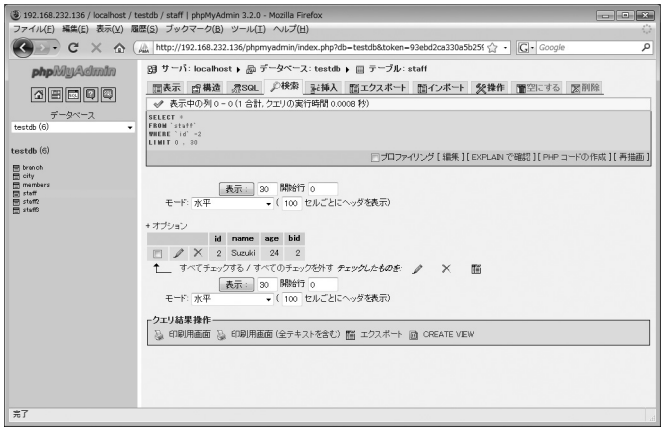
レコードを検索するには、テーブル一覧画面で「検索」タブをクリックします。

［図］レコード検索



検索結果と、検索に利用したSQLが表示されます。

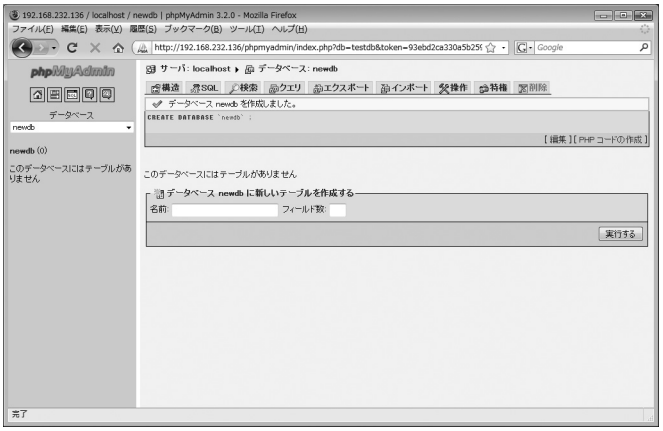
［図］レコード検索結果



データベースとテーブルの作成

データベースを作成するには、トップ画面で「新規データベースを作成する」にデータベース名を入力し、作成ボタンを押します。

［図］データベースの作成



テーブルの作成も続けて行うことができます。たとえば、テーブル名を「newtable」、フィールド数を「4」とすると、次のようなテーブル定義画面が表示されます。

［図］テーブルの作成

必要な事項を入力してください。ストレージエンジンも変更することができます。

索　引

記　号

/etc/my.cnf……132

数　字

3層スキーマ …… 96

アルファベット

A

ACID特性 …… 84
ALTER …… 76
AUTO COMMITモード …… 88
AVG …… 60

B

BEGINコマンド …… 86

C

COMMITコマンド …… 86
COUNT …… 60
CREATE DATABASE…… 38
CREATE TABLE …… 50
CREATE USERコマンド …… 40

D

DBMS …… 4
DCL …… 23
DDL …… 21
DELETE…… 80
DESC …… 53
DML …… 22
DROP DATABASE …… 39
DROP TABLE …… 80
DROP USERコマンド …… 44
DROP USER文 ……122

E

ERモデル …… 97
ER図 …… 97・98

F

Firebird …… 10

G

GRANTコマンド …… 40・42
GRANT文 ……120
GROUP BY …… 59

I

InnoDB ……129・131
INSERT …… 54
InterBase …… 10

L

LAMP…… 26
LIKE …… 57
LIMIT …… 62

M

MAX …… 60
MIN…… 60
my.ini ……132
MyISAM ……129
MyISAM ……131
MySQL …… 26
MySQL Cluster …… 7
MySQL Community Server …… 7
MySQL Enterprise…… 7
MySQL エンベデッドデータベース…… 7
mysqladminコマンド…… 29・112
mysqld ……110
MySQLクライアント …… 31
mysqlコマンド…… 31

N

NOT LIKE …… 57
NOT NULL …… 53

O

ORDER BY …… 63

P

phpMyAdmin ……138
PostgreSQL…… 9

R

REVOKE文 ……122
ROLLBACKコマンド …… 87

S

SELECT…… 55
SELECT DATABASEコマンド …… 33
SET PASSWORDコマンド …… 43・123
SHOW DATABASESコマンド …… 33
SHOW GRANTS文……121
SQL …… 20
SQLite …… 11
START TRANSACTIONコマンド …… 86
SUM …… 60

T

TRUNCATE …… 80

U

UNION …… 68
UPDATE …… 78
USEコマンド …… 33

W

WHERE …… 57

カ　ナ

ア

アトリビュート…… 97
エンティティ…… 97

カ

カーディナリティ…… 99
概念スキーマ…… 96
概念データモデル…… 94
外部キー…… 89
外部結合…… 72
外部スキーマ…… 96
関係演算…… 18
関係モデル…… 16
関数従属……103
共有ロック…… 4
行…… 16
グループ化…… 59
コミット…… 84

サ

サブクエリー…… 74
参照整合性…… 89・90
自己結合…… 75
システム変数……117
集合演算…… 18・19
主キー…… 17・52
ストレージエンジン……129
正規化……102
占有ロック…… 4

タ

第1正規形 ……104
第2正規形 ……105
第3正規形 ……106
データ型…… 50
データ制御言語…… 23
データ操作言語…… 22
データ定義言語…… 21
データベース管理システム…… 4
データモデリング…… 94
テーブル…… 16
トランザクション……4・84
トランザクション管理…… 4

ナ

内部結合…… 70
内部スキーマ…… 96
日本MySQLユーザ会 …… 7

ハ

排他制御……4・85
バックアップ……128
副問い合わせ…… 74
物理データモデル…… 94

ラ

リストア……128
リレーショナルモデル…… 16
リレーションシップ…… 16・97
レコード…… 16
列…… 16
ロールバック……5・84
ロールフォワード…… 5
ログファイル……134
ロック…… 85
論理データモデル…… 94

MySQL入門

Ver 1.0.0

2009年7月1日　　初 版　　第1刷　発行

著　者　　株式会社 リナックスアカデミー
監　修　　サン・マイクロシステムズ 株式会社
発　行　　株式会社 リナックスアカデミー
〒160-0023
東京都新宿区西新宿7-4-3 升本ビル
TEL：03-3365-2072　FAX：03-3365-2076
URL：http://www.linuxacademy.ne.jp/
http://www.linuxacademy.ne.jp/biz/

090519JLB01