==========

9-1. 日付と時刻：日付関数と操作

はじめに

データベースでは、日付や時刻の情報を扱うことが非常に多くあります。学校データベースでも、授業の開催日、成績の提出日、教師の休暇期間など、様々な場面で日付や時刻の情報が使われています。

SQLでは、日付や時刻を効率的に操作するための専用の関数が多数用意されています。これらの関数を使うことで、以下のような作業を簡単に行うことができます：

- 「今月開催される授業」を検索する
- 「提出期限まで残り何日か」を計算する
- 「曜日別の授業数」を集計する
- 「過去1年間の成績データ」を分析する

この章では、MySQLにおける日付と時刻の基本的な扱い方と、よく使われる日付関数について学びます。

日付と時刻のデータ型

MySQLでは、日付と時刻を扱うための複数のデータ型が用意されています：

> **用語解説**：
>
> - **データ型**：データベースがデータをどのような形式で保存するかを定義するもの。日付専用のデータ型を使うことで、日付の計算や比較が正確に行えます。

主要な日付・時刻データ型

データ型	説明	形式例
DATE	日付のみを格納	'2025-05-27'
TIME	時刻のみを格納	'14:30:00'
DATETIME	日付と時刻の両方を格納	'2025-05-27 14:30:00'
TIMESTAMP	日付と時刻（タイムゾーン考慮）	'2025-05-27 14:30:00'
YEAR	年のみを格納	2025

学校データベースでは、以下のような使い分けがされています：

- `course_schedule.schedule_date`：DATE型（授業の開催日）
- `class_periods.start_time`、`end_time`：TIME型（授業時間）
- `grades.submission_date`：DATE型（成績の提出日）
- `teacher_unavailability.start_date`、`end_date`：DATE型（教師の休暇期間）

現在の日付と時刻を取得する関数

NOW()関数：現在の日付と時刻

現在の日付と時刻を取得するには`NOW()`関数を使います。

> **用語解説**：
>
> - **NOW()関数**：「今」という意味で、実行した瞬間の日付と時刻を返す関数です。

```
SELECT NOW() AS 現在の日時;
```

実行結果：

現在の日時
2025-05-27 14:30:00

CURDATE()関数：現在の日付のみ

現在の日付だけが必要な場合は`CURDATE()`関数を使います。

> **用語解説**：
>
> - **CURDATE()関数**：「Current Date」の略で、現在の日付のみを返す関数です。

```
SELECT CURDATE() AS 今日の日付;
```

実行結果：

今日の日付
2025-05-27

CURTIME()関数：現在の時刻のみ

現在の時刻だけが必要な場合は`CURTIME()`関数を使います。

> **用語解説**：
>
> - **CURTIME()関数**：「Current Time」の略で、現在の時刻のみを返す関数です。

```
SELECT CURTIME() AS 現在の時刻;
```

実行結果：

現在の時刻
14:30:00

日付の各部分を取得する関数

日付から年、月、日などの特定の部分だけを取り出すことができます。

YEAR()、MONTH()、DAY()関数

```
SELECT schedule_date,
       YEAR(schedule_date) AS 年,
       MONTH(schedule_date) AS 月,
       DAY(schedule_date) AS 日
FROM course_schedule
WHERE schedule_id <= 5;
```

実行結果：

schedule_date	年	月	日
2025-05-14	2025	5	14
2025-05-15	2025	5	15
2025-05-16	2025	5	16
2025-05-20	2025	5	20
2025-05-21	2025	5	21

DAYNAME()、MONTHNAME()関数：曜日名と月名

```
SELECT schedule_date,
       DAYNAME(schedule_date) AS 曜日,
       MONTHNAME(schedule_date) AS 月名
FROM course_schedule
WHERE schedule_id <= 5;
```

実行結果：

schedule_date	曜日	月名
2025-05-14	Wednesday	May
2025-05-15	Thursday	May
2025-05-16	Friday	May
2025-05-20	Tuesday	May
2025-05-21	Wednesday	May

DAYOFWEEK()、WEEKDAY()関数：曜日を数値で取得

> **用語解説：**
>
> - **DAYOFWEEK()関数**：曜日を1（日曜日）から7（土曜日）の数値で返します。
> - **WEEKDAY()関数**：曜日を0（月曜日）から6（日曜日）の数値で返します。

```
SELECT schedule_date,
       DAYOFWEEK(schedule_date) AS 曜日番号_日曜始まり,
       WEEKDAY(schedule_date) AS 曜日番号_月曜始まり
FROM course_schedule
WHERE schedule_id <= 3;
```

実行結果：

schedule_date	曜日番号_日曜始まり	曜日番号_月曜始まり
2025-05-14	4	2
2025-05-15	5	3
2025-05-16	6	4

日付の計算と操作

DATE_ADD()、DATE_SUB()関数：日付の加算と減算

日付に特定の期間を足したり引いたりするには、`DATE_ADD()`と`DATE_SUB()`関数を使います。

> **用語解説：**
>
> - **DATE_ADD()関数**：指定した日付に期間を加算する関数です。
> - **DATE_SUB()関数**：指定した日付から期間を減算する関数です。
> - **INTERVAL**：期間を表すキーワードで、「1 DAY」「2 WEEK」「3 MONTH」のように使います。

基本構文

```
DATE_ADD(日付, INTERVAL 数値 単位)
DATE_SUB(日付, INTERVAL 数値 単位)
```

主な単位：

- `DAY`：日
- `WEEK`：週
- `MONTH`：月
- `YEAR`：年
- `HOUR`：時間
- `MINUTE`：分

例：日付の加算・減算

```
SELECT schedule_date,
       DATE_ADD(schedule_date, INTERVAL 1 WEEK) AS 1週間後,
       DATE_SUB(schedule_date, INTERVAL 3 DAY) AS 3日前,
       DATE_ADD(schedule_date, INTERVAL 2 MONTH) AS 2ヶ月後
FROM course_schedule
WHERE schedule_id <= 3;
```

実行結果：

schedule_date	1週間後	3日前	2ヶ月後
2025-05-14	2025-05-21	2025-05-11	2025-07-14
2025-05-15	2025-05-22	2025-05-12	2025-07-15
2025-05-16	2025-05-23	2025-05-13	2025-07-16

DATEDIFF()関数：日付の差を計算

2つの日付の間の日数を計算するには`DATEDIFF()`関数を使います。

> **用語解説**：
>
> - **DATEDIFF()関数**：2つの日付の差を日数で返す関数です。「Date Difference」の略です。

```
SELECT submission_date,
       DATEDIFF(CURDATE(), submission_date) AS 提出からの経過日数
FROM grades
WHERE submission_date IS NOT NULL
LIMIT 5;
```

実行結果：

submission_date	提出からの経過日数
2025-05-20	7
2025-05-20	7
2025-05-10	17
2025-05-10	17
2025-05-18	9

日付の書式設定

DATE_FORMAT()関数：日付の表示形式を変更

日付を様々な形式で表示するには`DATE_FORMAT()`関数を使います。

> **用語解説：**
>
> - **DATE_FORMAT()関数**：日付を任意の形式で表示するための関数です。
> - **書式指定子**：日付の表示形式を指定するための特殊な文字（%Y、%m、%dなど）です。

主要な書式指定子

指定子	説明	例
%Y	4桁の年	2025
%y	2桁の年	25
%m	2桁の月（01-12）	05
%c	月（1-12）	5
%d	2桁の日（01-31）	14
%e	日（1-31）	14
%W	曜日名（英語）	Wednesday
%a	曜日名（短縮）	Wed

例：様々な日付形式

```
SELECT schedule_date,
       DATE_FORMAT(schedule_date, '%Y年%m月%d日') AS 日本形式,
       DATE_FORMAT(schedule_date, '%m/%d/%Y') AS 米国形式,
       DATE_FORMAT(schedule_date, '%Y-%m-%d (%W)') AS 曜日付き
FROM course_schedule
WHERE schedule_id <= 3;
```

実行結果：

schedule_date	日本形式	米国形式	曜日付き
2025-05-14	2025年05月14日	05/14/2025	2025-05-14 (Wednesday)
2025-05-15	2025年05月15日	05/15/2025	2025-05-15 (Thursday)
2025-05-16	2025年05月16日	05/16/2025	2025-05-16 (Friday)

実践例：日付関数を活用したクエリ

例1：今月の授業スケジュールを取得

```
SELECT cs.schedule_date,
       DATE_FORMAT(cs.schedule_date, '%m月%d日(%a)') AS 開催日,
       c.course_name,
       t.teacher_name
FROM course_schedule cs
JOIN courses c ON cs.course_id = c.course_id
JOIN teachers t ON cs.teacher_id = t.teacher_id
WHERE YEAR(cs.schedule_date) = YEAR(CURDATE())
  AND MONTH(cs.schedule_date) = MONTH(CURDATE())
ORDER BY cs.schedule_date
LIMIT 5;
```

例2：教師の休暇期間の残り日数を計算

```
SELECT teacher_id,
       start_date,
       end_date,
       DATEDIFF(end_date, CURDATE()) AS 残り日数,
       CASE
         WHEN DATEDIFF(end_date, CURDATE()) < 0 THEN '終了済み'
         WHEN DATEDIFF(end_date, CURDATE()) = 0 THEN '本日終了'
         ELSE CONCAT(DATEDIFF(end_date, CURDATE()), '日後終了')
       END AS 状況
FROM teacher_unavailability
LIMIT 5;
```

例3：曜日別の授業数を集計

```
SELECT DAYNAME(schedule_date) AS 曜日,
       COUNT(*) AS 授業数
FROM course_schedule
GROUP BY DAYNAME(schedule_date), DAYOFWEEK(schedule_date)
ORDER BY DAYOFWEEK(schedule_date);
```

練習問題

問題9-1-1

現在の日付と時刻、現在の日付のみ、現在の時刻のみを一つのクエリで取得するSQLを書いてください。

問題9-1-2

course_schedule（授業カレンダー）テーブルから、2025年5月に開催される授業の日付、年、月、日、曜日名を取得するSQLを書いてください。

問題9-1-3

grades（成績）テーブルから、提出日（submission_date）が今日から30日以内の成績レコードを取得するSQLを書いてください。

問題9-1-4

course_schedule（授業カレンダー）テーブルから、各授業日について「○年○月○日（○曜日）」の形式で表示するSQLを書いてください。

問題9-1-5

teacher_unavailability（講師スケジュール管理）テーブルから、休暇期間の日数（end_date - start_date + 1）を計算して表示するSQLを書いてください。

問題9-1-6

course_schedule（授業カレンダー）テーブルから、月曜日に開催される授業の数を取得するSQLを書いてください。

問題9-1-7

grades（成績）テーブルから、提出日が2025年5月15日から1週間以内（2025年5月21日まで）の成績を取得するSQLを書いてください。

問題9-1-8

course_schedule（授業カレンダー）テーブルから、今日から1ヶ月後に開催される授業スケジュールを取得するSQLを書いてください。

問題9-1-9

grades（成績）テーブルから、各月ごとの成績提出件数を「○月：○件」の形式で表示するSQLを書いてください。

問題9-1-10

course_schedule（授業カレンダー）テーブルから、金曜日に開催される授業について、授業日の2日前の日付も合わせて表示するSQLを書いてください。

解答と詳細な解説

解答9-1-1

```
SELECT NOW() AS 現在の日付と時刻,
       CURDATE() AS 現在の日付,
       CURTIME() AS 現在の時刻;
```

解説：

- `NOW()`：実行時点の日付と時刻を「YYYY-MM-DD HH:MM:SS」形式で返します

- `CURDATE()`：実行時点の日付を「YYYY-MM-DD」形式で返します
- `CURTIME()`：実行時点の時刻を「HH:MM:SS」形式で返します

解答9-1-2

```
SELECT schedule_date,
       YEAR(schedule_date) AS 年,
       MONTH(schedule_date) AS 月,
       DAY(schedule_date) AS 日,
       DAYNAME(schedule_date) AS 曜日名
FROM course_schedule
WHERE YEAR(schedule_date) = 2025
  AND MONTH(schedule_date) = 5;
```

解説：

- `YEAR()`、`MONTH()`、`DAY()`関数で日付から各要素を抽出
- `DAYNAME()`関数で英語の曜日名を取得
- WHERE句で2025年5月の条件を指定

解答9-1-3

```
SELECT *
FROM grades
WHERE submission_date IS NOT NULL
  AND submission_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);
```

解説：

- `DATE_SUB(CURDATE(), INTERVAL 30 DAY)`で今日から30日前の日付を計算
- `submission_date IS NOT NULL`でNULL値を除外
- 提出日が30日前以降（つまり30日以内）の条件を指定

解答9-1-4

```
SELECT schedule_date,
       DATE_FORMAT(schedule_date, '%Y年%m月%d日（%W）') AS 開催日
FROM course_schedule;
```

解説：

- `DATE_FORMAT()`関数で日付の表示形式をカスタマイズ
- `%Y`：4桁年、`%m`：2桁月、`%d`：2桁日、`%W`：英語曜日名
- 日本語の「年月日」と括弧付きの曜日で表示

解答9-1-5

```
SELECT teacher_id,
       start_date,
       end_date,
       DATEDIFF(end_date, start_date) + 1 AS 休暇日数
FROM teacher_unavailability;
```

解説：

- `DATEDIFF(end_date, start_date)`で終了日と開始日の差を日数で計算
- 開始日と終了日の両方を含めるため、結果に1を加算
- 例：5月1日～5月3日の場合、DATEDIFF結果は2だが、実際は3日間なので+1

解答9-1-6

```
SELECT COUNT(*) AS 月曜日の授業数
FROM course_schedule
WHERE DAYOFWEEK(schedule_date) = 2;
```

解説：

- `DAYOFWEEK()`関数は1（日曜）～7（土曜）の数字を返す
- 月曜日は2に対応するため、`DAYOFWEEK(schedule_date) = 2`で条件指定
- `COUNT(*)`で該当する授業の数を集計

解答9-1-7

```
SELECT *
FROM grades
WHERE submission_date BETWEEN '2025-05-15' AND '2025-05-21';
```

または

```
SELECT *
FROM grades
WHERE submission_date >= '2025-05-15'
  AND submission_date <= DATE_ADD('2025-05-15', INTERVAL 1 WEEK);
```

解説：

- 1つ目の解答では`BETWEEN`演算子を使用してシンプルに期間指定
- 2つ目の解答では`DATE_ADD()`関数で1週間後を動的に計算
- どちらも同じ結果を返すが、2つ目の方が「1週間以内」という条件をより明確に表現

解答9-1-8

```
SELECT *
FROM course_schedule
WHERE schedule_date = DATE_ADD(CURDATE(), INTERVAL 1 MONTH);
```

解説：

- `DATE_ADD(CURDATE(), INTERVAL 1 MONTH)`で今日から1ヶ月後の日付を計算
- 完全一致（=）で該当する日の授業のみを取得
- もし「1ヶ月後以降」の意味なら、`>=`を使用

解答9-1-9

```
SELECT CONCAT(MONTH(submission_date), '月：', COUNT(*), '件') AS 月別提出件数
FROM grades
WHERE submission_date IS NOT NULL
GROUP BY MONTH(submission_date)
ORDER BY MONTH(submission_date);
```

解説：

- `MONTH()`関数で提出日から月を抽出
- `GROUP BY MONTH(submission_date)`で月ごとにグループ化
- `CONCAT()`関数で「○月：○件」の形式に文字列結合
- `submission_date IS NOT NULL`でNULL値を除外

解答9-1-10

```
SELECT schedule_date,
       DAYNAME(schedule_date) AS 曜日,
       DATE_SUB(schedule_date, INTERVAL 2 DAY) AS 2日前の日付
FROM course_schedule
WHERE DAYOFWEEK(schedule_date) = 6;
```

解説：

- `DAYOFWEEK(schedule_date) = 6`で金曜日を指定（6が金曜日）
- `DATE_SUB(schedule_date, INTERVAL 2 DAY)`で授業日の2日前を計算
- 金曜日の授業なら水曜日が2日前として表示される

まとめ

この章では、MySQLにおける日付と時刻の操作について学びました：

1. **基本的な日付関数**：NOW()、CURDATE()、CURTIME()で現在の日時を取得
2. **日付要素の抽出**：YEAR()、MONTH()、DAY()、DAYNAME()で日付の各部分を取得

3. **日付の計算**：DATE_ADD()、DATE_SUB()、DATEDIFF()で日付の加減算と差の計算
4. **日付の書式設定**：DATE_FORMAT()で任意の形式での日付表示
5. **実践的な活用**：複数の関数を組み合わせた実用的なクエリの作成

日付関数は、データ分析やレポート作成において非常に重要な機能です。特に学校データベースのような時系列データを多く扱うシステムでは、これらの関数を効果的に使うことで、より深い洞察を得ることができます。

次のセクションでは、「文字列操作：テキスト関数」について学び、テキストデータの加工や検索の技術を習得します。

9-2. 文字列操作：テキスト関数

はじめに

データベースでは、テキスト（文字列）データを扱うことが非常に多くあります。学校データベースでも、学生名、教師名、講座名、コメントなど、様々な文字列データが保存されています。

これらの文字列データを効果的に操作するためのSQL関数を「文字列関数」または「テキスト関数」と呼びます。文字列関数を使うことで、以下のような作業を効率的に行うことができます：

- 「田中」という姓の学生を「田中さん」として表示する
- 講座名の最初の10文字だけを表示する
- 学生名から姓と名を分離する
- 大文字・小文字を統一して検索の精度を上げる
- 余分な空白を取り除いてデータを整理する

この章では、MySQLで使用できる主要な文字列関数について学び、実際の業務で役立つテキスト処理のテクニックを習得します。

> **用語解説**：
>
> - **文字列関数**：テキストデータを加工、検索、操作するための専用関数群です。
> - **テキスト処理**：文字列データを目的に応じて変換、抽出、結合などを行う作業のことです。

文字列の基本情報を取得する関数

LENGTH()関数：文字列の長さを取得

文字列の文字数（バイト数）を取得するには`LENGTH()`関数を使います。

> **用語解説**：
>
> - **LENGTH()関数**：文字列の長さをバイト単位で返す関数です。日本語の場合、1文字が複数バイトになることがあります。

```
SELECT student_name,
       LENGTH(student_name) AS バイト数
```

```
FROM students
LIMIT 5;
```

実行結果：

student_name	バイト数
黒沢春馬	12
新垣愛留	12
柴崎春花	12
森下風凛	12
河口菜恵子	15

CHAR_LENGTH()関数：文字数を取得

実際の文字数を取得するには`CHAR_LENGTH()`関数を使います。

> **用語解説**：
>
> - **CHAR_LENGTH()関数**：文字列の実際の文字数を返す関数です。日本語でも1文字は1としてカウントされます。

```
SELECT student_name,
       LENGTH(student_name) AS バイト数,
       CHAR_LENGTH(student_name) AS 文字数
FROM students
LIMIT 5;
```

実行結果：

student_name	バイト数	文字数
黒沢春馬	12	4
新垣愛留	12	4
柴崎春花	12	4
森下風凛	12	4
河口菜恵子	15	5

文字列の抽出と部分取得

SUBSTRING()関数：文字列の一部を抽出

文字列の指定した位置から指定した長さの部分を取り出すには`SUBSTRING()`関数を使います。

> **用語解説：**
>
> - **SUBSTRING()関数**：文字列の一部分を抽出する関数です。開始位置と長さを指定できます。

基本構文

```
SUBSTRING(文字列, 開始位置, 長さ)
SUBSTRING(文字列, 開始位置)  -- 開始位置から最後まで
```

※ MySQLでは位置は1から始まります（0ではありません）

例：文字列の部分抽出

```
SELECT course_name,
       SUBSTRING(course_name, 1, 10) AS 短縮名,
       SUBSTRING(course_name, 1, 5) AS 超短縮名
FROM courses
LIMIT 5;
```

実行結果：

course_name	短縮名	超短縮名
ITのための基礎知識	ITのための基礎知識	ITのた
UNIX入門	UNIX入門	UNIX入
Cプログラミング演習	Cプログラミング演習	Cプロ
Webアプリケーション開発	Webアプリケーション開	Webア
データベース設計と実装	データベース設計と実装	データ

LEFT()、RIGHT()関数：左端・右端から文字を取得

文字列の左端または右端から指定した文字数を取得するには`LEFT()`と`RIGHT()`関数を使います。

> **用語解説：**
>
> - **LEFT()関数**：文字列の左端（最初）から指定した文字数を取得する関数です。
> - **RIGHT()関数**：文字列の右端（最後）から指定した文字数を取得する関数です。

```
SELECT student_name,
       LEFT(student_name, 2) AS 姓,
       RIGHT(student_name, 2) AS 名
FROM students
WHERE CHAR_LENGTH(student_name) = 4
LIMIT 5;
```

実行結果：

student_name	姓	名
黒沢春馬	黒沢	春馬
新垣愛留	新垣	愛留
柴崎春花	柴崎	春花
森下風凛	森下	風凛

文字列の検索と位置特定

LOCATE()関数：文字列の位置を検索

特定の文字列が何文字目にあるかを調べるには`LOCATE()`関数を使います。

> **用語解説**：
>
> - **LOCATE()関数**：指定した文字列が対象の文字列の何文字目にあるかを返す関数です。見つからない場合は0を返します。

基本構文

```
LOCATE(検索文字列, 対象文字列)
LOCATE(検索文字列, 対象文字列, 開始位置)
```

例：文字列の位置検索

```
SELECT course_name,
       LOCATE('プログラミング', course_name) AS プログラミングの位置,
       LOCATE('入門', course_name) AS 入門の位置
FROM courses
WHERE course_name LIKE '%プログラミング%' OR course_name LIKE '%入門%'
LIMIT 5;
```

実行結果：

course_name	プログラミングの位置	入門の位置
UNIX入門	0	5
Cプログラミング演習	2	0
JavaScript入門とDOM操作	0	11
モバイルアプリ開発入門	0	11

course_name	プログラミングの位置	入門の位置
IoTデバイスプログラミング実践	8	0

文字列の変換と加工

UPPER()、LOWER()関数：大文字・小文字変換

英字の大文字・小文字を変換するには`UPPER()`と`LOWER()`関数を使います。

> **用語解説：**
>
> - **UPPER()関数**：英字を大文字に変換する関数です。
> - **LOWER()関数**：英字を小文字に変換する関数です。

```
SELECT classroom_id,
       UPPER(classroom_id) AS 大文字,
       LOWER(classroom_id) AS 小文字
FROM classrooms
LIMIT 5;
```

実行結果：

classroom_id	大文字	小文字
101A	101A	101a
102B	102B	102b
201C	201C	201c
202D	202D	202d
301E	301E	301e

TRIM()、LTRIM()、RTRIM()関数：空白の除去

文字列の前後や片側の空白を除去するには`TRIM()`、`LTRIM()`、`RTRIM()`関数を使います。

> **用語解説：**
>
> - **TRIM()関数**：文字列の前後の空白を除去する関数です。
> - **LTRIM()関数**：文字列の左側（前）の空白を除去する関数です。
> - **RTRIM()関数**：文字列の右側（後）の空白を除去する関数です。

```
-- 例：空白を含むデータの処理（実際のデータでテスト）
SELECT CONCAT('「', '  テスト文字列  ', '」') AS 元の文字列,
       CONCAT('「', TRIM('  テスト文字列  '), '」') AS TRIM後,
       CONCAT('「', LTRIM('  テスト文字列  '), '」') AS LTRIM後,
       CONCAT('「', RTRIM('  テスト文字列  '), '」') AS RTRIM後;
```

実行結果：

元の文字列	TRIM後	LTRIM後	RTRIM後
「 テスト文字列 」	「テスト文字列」	「テスト文字列 」	「 テスト文字列」

文字列の結合と分割

CONCAT()関数：文字列の結合

複数の文字列を結合するには`CONCAT()`関数を使います。

> **用語解説**：
>
> - **CONCAT()関数**：複数の文字列を1つにつなげる関数です。「Concatenate（連結）」の略です。

```
SELECT teacher_name,
       CONCAT('講師：', teacher_name, '先生') AS 敬称付き名前,
       CONCAT('ID-', teacher_id, '：', teacher_name) AS ID付き名前
FROM teachers
LIMIT 5;
```

実行結果：

teacher_name	敬称付き名前	ID付き名前
寺内鞍	講師：寺内鞍先生	ID-101：寺内鞍
田尻朋美	講師：田尻朋美先生	ID-102：田尻朋美
内村海凪	講師：内村海凪先生	ID-103：内村海凪
藤本理恵	講師：藤本理恵先生	ID-104：藤本理恵
黒木大介	講師：黒木大介先生	ID-105：黒木大介

CONCAT_WS()関数：区切り文字付き結合

区切り文字を指定して文字列を結合するには`CONCAT_WS()`関数を使います。

> **用語解説**：
>
> - **CONCAT_WS()関数**：「CONCAT With Separator」の略で、指定した区切り文字で複数の文字列を結合する関数です。

```
SELECT student_id,
       student_name,
       CONCAT_WS(' | ', student_id, student_name, '在籍中') AS 学生情報
```

```
FROM students
LIMIT 5;
```

実行結果：

student_id	student_name	学生情報
301	黒沢春馬	301
302	新垣愛留	302
303	柴崎春花	303
304	森下風凛	304
305	河口菜恵子	305

SUBSTRING_INDEX()関数：区切り文字による分割

区切り文字を基準に文字列を分割するには`SUBSTRING_INDEX()`関数を使います。

> **用語解説**：
>
> - **SUBSTRING_INDEX()関数**：指定した区切り文字で文字列を分割し、指定した番目までの部分を返す関数です。

基本構文

```
SUBSTRING_INDEX(文字列, 区切り文字, 番目)
```

- 正の数：左から数えて指定番目まで
- 負の数：右から数えて指定番目まで

例：文字列の分割

```
-- 施設情報から最初の設備だけを取得
SELECT classroom_name,
       facilities,
       SUBSTRING_INDEX(facilities, '、', 1) AS 主要設備
FROM classrooms
WHERE facilities IS NOT NULL
LIMIT 5;
```

実行結果：

classroom_name	facilities	主要設備
1号館コンピュータ実習室A	パソコン30台、プロジェクター	パソコン30台

classroom_name	facilities	主要設備
2号館コンピュータ実習室B	パソコン25台、プロジェクター、大型モニター	パソコン25台
2号館コンピュータ実習室C	パソコン25台、プロジェクター	パソコン25台
2号館コンピュータ実習室D	パソコン25台、プロジェクター、3Dプリンター	パソコン25台
3号館講義室E	プロジェクター、マイク設備、録画設備	プロジェクター

文字列の置換と操作

REPLACE()関数：文字列の置換

文字列の一部を別の文字列に置き換えるには`REPLACE()`関数を使います。

> **用語解説**：
>
> - **REPLACE()関数**：文字列内の指定した部分を別の文字列に置き換える関数です。

基本構文

```
REPLACE(対象文字列, 検索文字列, 置換文字列)
```

例：文字列の置換

```
SELECT course_name,
       REPLACE(course_name, '入門', 'ベーシック') AS 置換後の講座名
FROM courses
WHERE course_name LIKE '%入門%'
LIMIT 5;
```

実行結果：

course_name	置換後の講座名
UNIX入門	UNIXベーシック
JavaScript入門とDOM操作	JavaScriptベーシックとDOM操作
モバイルアプリ開発入門	モバイルアプリ開発ベーシック
クラウドサービス入門	クラウドサービスベーシック
UI/UXデザイン入門	UI/UXデザインベーシック

REVERSE()関数：文字列の反転

文字列を逆順にするには`REVERSE()`関数を使います。

> **用語解説：**
>
> - **REVERSE()関数**：文字列の文字順序を逆にする関数です。

```
SELECT classroom_id,
       REVERSE(classroom_id) AS 逆順ID
FROM classrooms
LIMIT 5;
```

実行結果：

classroom_id	逆順ID
101A	A101
102B	B201
201C	C102
202D	D202
301E	E103

文字列の比較とパターンマッチング

STRCMP()関数：文字列の比較

2つの文字列を辞書順で比較するには`STRCMP()`関数を使います。

> **用語解説：**
>
> - **STRCMP()関数**：「String Compare」の略で、2つの文字列を比較する関数です。結果は-1、0、1のいずれかを返します。

戻り値：

- 0：両方の文字列が同じ
- -1：最初の文字列が2番目より小さい（辞書順で前）
- 1：最初の文字列が2番目より大きい（辞書順で後）

```
SELECT student_name,
       STRCMP(student_name, '田中太郎') AS 田中太郎との比較
FROM students
WHERE student_name IN ('田中太郎', '佐藤花子', '山田翔太')
LIMIT 3;
```

実践例：文字列関数を活用したクエリ

例1：学生名の姓と名を分離して表示

```
SELECT student_name,
       CASE
         WHEN CHAR_LENGTH(student_name) = 4 THEN
           CONCAT(LEFT(student_name, 2), '・', RIGHT(student_name, 2))
         WHEN CHAR_LENGTH(student_name) = 5 THEN
           CONCAT(LEFT(student_name, 3), '・', RIGHT(student_name, 2))
         ELSE student_name
       END AS 姓名分離
FROM students
WHERE CHAR_LENGTH(student_name) IN (4, 5)
LIMIT 8;
```

例2：講座名の長さによる分類とタグ付け

```
SELECT course_name,
       CHAR_LENGTH(course_name) AS 文字数,
       CASE
         WHEN CHAR_LENGTH(course_name) <= 10 THEN '[短]'
         WHEN CHAR_LENGTH(course_name) <= 15 THEN '[中]'
         ELSE '[長]'
       END AS 長さタグ,
       CONCAT(
         CASE
           WHEN CHAR_LENGTH(course_name) <= 10 THEN '[短] '
           WHEN CHAR_LENGTH(course_name) <= 15 THEN '[中] '
           ELSE '[長] '
         END,
         course_name
       ) AS タグ付き講座名
FROM courses
ORDER BY CHAR_LENGTH(course_name)
LIMIT 5;
```

例3：コメントの要約作成

```
SELECT student_id,
       comment,
       CASE
         WHEN comment IS NULL THEN '(コメントなし)'
         WHEN CHAR_LENGTH(comment) <= 10 THEN comment
         ELSE CONCAT(LEFT(comment, 10), '...')
       END AS コメント要約
FROM attendance
WHERE schedule_id = 1
LIMIT 5;
```

練習問題

問題9-2-1

students（学生）テーブルから、学生名の文字数とバイト数を取得し、文字数が4文字の学生のみを表示するSQLを書いてください。

問題9-2-2

courses（講座）テーブルから、講座名の最初の8文字と最後の3文字を表示するSQLを書いてください。

問題9-2-3

teachers（教師）テーブルから、教師名に「田」という文字が含まれる位置を表示するSQLを書いてください。

問題9-2-4

classrooms（教室）テーブルから、教室IDを大文字と小文字で表示し、「教室：」という文字を前に付けて表示するSQLを書いてください。

問題9-2-5

courses（講座）テーブルから、講座名に「プログラミング」という単語を「コーディング」に置き換えて表示するSQLを書いてください。

問題9-2-6

students（学生）テーブルから、学生IDと学生名を「ID：○○○、名前：○○」の形式で表示するSQLを書いてください。

問題9-2-7

classrooms（教室）テーブルから、施設情報（facilities）の最初の設備のみを抽出して表示するSQLを書いてください。（「、」で区切られている）

問題9-2-8

courses（講座）テーブルから、講座名の文字数が15文字を超える場合は最初の12文字に「...」を付けて表示し、15文字以下の場合はそのまま表示するSQLを書いてください。

問題9-2-9

students（学生）テーブルから、学生名を逆順にした文字列と、元の学生名を比較して同じかどうかを判定するSQLを書いてください。（回文の検出）

問題9-2-10

teachers（教師）テーブルとcourses（講座）テーブルを結合し、「○○先生が担当する『○○』講座」という形式で表示するSQLを書いてください。

解答と詳細な解説

解答9-2-1

```
SELECT student_name,
       CHAR_LENGTH(student_name) AS 文字数,
       LENGTH(student_name) AS バイト数
FROM students
WHERE CHAR_LENGTH(student_name) = 4;
```

解説：

- CHAR_LENGTH()で実際の文字数を取得
- LENGTH()でバイト数を取得（日本語は1文字3-4バイト）
- WHERE句で文字数が4文字の条件を指定
- 日本語の姓名は通常4文字（姓2文字+名2文字）が多いため、この条件で多くの学生が抽出される

解答9-2-2

```
SELECT course_name,
       LEFT(course_name, 8) AS 最初の8文字,
       RIGHT(course_name, 3) AS 最後の3文字
FROM courses;
```

解説：

- LEFT(course_name, 8)で文字列の左端から8文字を取得
- RIGHT(course_name, 3)で文字列の右端から3文字を取得
- 講座名が8文字未満の場合は、そのまま全体が表示される
- 講座名が3文字未満の場合も、そのまま全体が表示される

解答9-2-3

```
SELECT teacher_name,
       LOCATE('田', teacher_name) AS 田の位置,
       CASE
         WHEN LOCATE('田', teacher_name) > 0 THEN
           CONCAT(LOCATE('田', teacher_name), '文字目に「田」があります')
         ELSE '「田」は含まれていません'
       END AS 結果
FROM teachers;
```

解説：

- LOCATE('田', teacher_name)で「田」の位置を検索
- 戻り値が0の場合は該当文字が見つからない
- CASE文で見つかった場合とそうでない場合の表示を分岐
- 位置は1から始まる（0ベースではない）

解答9-2-4

```
SELECT classroom_id,
       UPPER(classroom_id) AS 大文字,
       LOWER(classroom_id) AS 小文字,
       CONCAT('教室：', classroom_id) AS 教室表示
FROM classrooms;
```

解説：

- UPPER()で英字部分を大文字に変換
- LOWER()で英字部分を小文字に変換
- CONCAT()で「教室：」の文字列を前に結合
- 数字部分は大文字・小文字変換の影響を受けない

解答9-2-5

```
SELECT course_name,
       REPLACE(course_name, 'プログラミング', 'コーディング') AS 置換後の講座名
FROM courses;
```

解説：

- REPLACE()関数で「プログラミング」を「コーディング」に置換
- 該当する文字列がない場合は元の文字列がそのまま表示される
- 文字列内に複数の「プログラミング」がある場合は、すべて置換される
- 大文字・小文字を区別する（MySQLのデフォルト設定による）

解答9-2-6

```
SELECT student_id,
       student_name,
       CONCAT('ID：', student_id, '、名前：', student_name) AS 学生情報
FROM students;
```

解説：

- CONCAT()で複数の文字列と値を結合
- 数値であるstudent_idも自動的に文字列として結合される
- 日本語の句読点（、）も正常に表示される
- より読みやすい形式でデータを表示できる

解答9-2-7

```
SELECT classroom_name,
       facilities,
```

```
       SUBSTRING_INDEX(facilities, '、', 1) AS 最初の設備
FROM classrooms
WHERE facilities IS NOT NULL;
```

解説：

- SUBSTRING_INDEX()で「、」を区切り文字として1番目までを取得
- つまり最初の「、」より前の部分を抽出
- WHERE facilities IS NOT NULLでNULL値を除外
- 施設情報がない教室は結果に含まれない

解答9-2-8

```
SELECT course_name,
       CHAR_LENGTH(course_name) AS 文字数,
       CASE
         WHEN CHAR_LENGTH(course_name) > 15 THEN
           CONCAT(LEFT(course_name, 12), '...')
         ELSE course_name
       END AS 表示用講座名
FROM courses;
```

解説：

- CHAR_LENGTH(course_name) > 15で15文字を超えるかを判定
- 超える場合はLEFT(course_name, 12)で最初の12文字を取得し「...」を追加
- 15文字以下の場合は元の講座名をそのまま表示
- CASE文で条件分岐を明確に表現

解答9-2-9

```
SELECT student_name,
       REVERSE(student_name) AS 逆順名前,
       CASE
         WHEN student_name = REVERSE(student_name) THEN '回文です'
         ELSE '回文ではありません'
       END AS 回文判定
FROM students;
```

解説：

- REVERSE()で文字列を逆順にする
- 元の文字列と逆順の文字列を比較
- 同じ場合は回文（前から読んでも後ろから読んでも同じ）
- 日本語の名前で回文になることは稀だが、データ処理の例として有用

解答9-2-10

```
SELECT t.teacher_name,
       c.course_name,
       CONCAT(t.teacher_name, '先生が担当する『', c.course_name, '』講座') AS 担当講座
情報
FROM teachers t
JOIN courses c ON t.teacher_id = c.teacher_id;
```

解説：

- JOINで教師テーブルと講座テーブルを結合
- CONCAT()で複数の要素を組み合わせて読みやすい文章を作成
- 『』で講座名を囲んで視認性を向上
- 実際の業務でよく使われる形式でデータを表示

まとめ

この章では、MySQLにおける文字列操作について学びました：

1. **文字列の基本情報取得**：LENGTH()、CHAR_LENGTH()で長さを測定
2. **文字列の抽出**：SUBSTRING()、LEFT()、RIGHT()で部分文字列を取得
3. **文字列の検索**：LOCATE()で特定の文字列の位置を特定
4. **文字列の変換**：UPPER()、LOWER()で大文字・小文字変換、TRIM()で空白除去
5. **文字列の結合**：CONCAT()、CONCAT_WS()で複数の文字列を結合
6. **文字列の分割**：SUBSTRING_INDEX()で区切り文字による分割
7. **文字列の置換**：REPLACE()で特定の文字列を別の文字列に置換
8. **実践的な活用**：複数の関数を組み合わせた複雑なテキスト処理

文字列関数は、データの表示形式を整えたり、データクレンジング（データの清浄化）を行ったり、レポート作成時に読みやすい形式に変換したりする際に非常に重要な機能です。特に学校データベースのような多様なテキストデータを扱うシステムでは、これらの関数を効果的に使うことで、より使いやすく、理解しやすいデータ表示が可能になります。

次のセクションでは、「JSON：構造化データの格納と操作」について学び、より複雑なデータ構造を扱う技術を習得します。

9-3. JSON：構造化データの格納と操作

はじめに

現代のWebアプリケーションやシステムでは、複雑な構造を持つデータを扱うことが増えています。従来のリレーショナルデータベースでは、このような構造化データを複数のテーブルに分けて保存する必要がありましたが、MySQLではJSON（JavaScript Object Notation）データ型を使って、構造化データを1つのカラムに格納できるようになりました。

学校データベースでも、以下のような場面でJSONが活用できます：

- 学生の詳細情報（趣味、スキル、連絡先情報など）

- 講座の設定情報（評価基準、カリキュラム詳細など）
- 教室の設備詳細（機器の仕様、配置情報など）
- 授業のメタデータ（使用教材、課題設定など）

この章では、MySQLでのJSON データ型の基本的な使い方から、JSON データの作成、検索、更新、操作方法について学びます。

> **用語解説**：
>
> - **JSON（JavaScript Object Notation）**：軽量なデータ交換フォーマットで、人間が読みやすく、機械が解析しやすい構造化データの表現方法です。
> - **構造化データ**：階層構造や複雑な関係を持つデータのことで、配列やオブジェクトなどの形式で表現されます。

JSONデータ型とは

MySQLのJSONデータ型は、有効なJSONドキュメントを格納するために設計された専用のデータ型です。通常のTEXT型と比べて、以下の利点があります：

JSONデータ型の利点

1. **自動検証**：無効なJSONが挿入されるとエラーになる
2. **最適化された格納**：JSON構造に最適化された内部形式で保存
3. **専用関数**：JSON操作のための豊富な関数群
4. **インデックス対応**：JSON内の特定の要素にインデックスを作成可能

JSONの基本構造

```
{
  "name": "田中太郎",
  "age": 20,
  "skills": ["Java", "Python", "SQL"],
  "contact": {
    "email": "tanaka@example.com",
    "phone": "090-1234-5678"
  },
  "active": true
}
```

> **用語解説**：
>
> - **オブジェクト**：{}で囲まれた、キーと値のペアの集合です。
> - **配列**：[]で囲まれた、値のリストです。
> - **キー**：JSONオブジェクト内でデータを識別するための文字列です。
> - **値**：文字列、数値、真偽値、null、オブジェクト、配列のいずれかです。

JSON データの作成と挿入

テーブルの作成

まず、JSON カラムを含むテーブルを作成してみましょう。学校データベースに学生の詳細情報を格納するテーブルを追加します：

```
CREATE TABLE student_profiles (
    student_id BIGINT PRIMARY KEY,
    profile_data JSON,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);
```

JSON データの挿入

例1：基本的なJSON データの挿入

```
INSERT INTO student_profiles (student_id, profile_data) VALUES
(301, JSON_OBJECT(
    'name', '黒沢春馬',
    'birth_year', 2003,
    'skills', JSON_ARRAY('Java', 'HTML', 'CSS'),
    'contact', JSON_OBJECT(
        'email', 'kurosawa@example.com',
        'emergency_phone', '090-1111-2222'
    ),
    'interests', JSON_ARRAY('プログラミング', 'ゲーム開発', '映画鑑賞'),
    'active', true
));
```

例2：文字列形式のJSON の挿入

```
INSERT INTO student_profiles (student_id, profile_data) VALUES
(302, '{
    "name": "新垣愛留",
    "birth_year": 2004,
    "skills": ["Python", "データ分析", "機械学習"],
    "contact": {
        "email": "aragaki@example.com",
        "emergency_phone": "090-2222-3333"
    },
    "interests": ["AI", "データサイエンス", "読書"],
    "active": true
}'),
(303, '{
    "name": "柴崎春花",
    "birth_year": 2003,
    "skills": ["JavaScript", "React", "Node.js"],
    "contact": {
        "email": "shibasaki@example.com",
```

```
            "emergency_phone": "090-3333-4444"
        },
        "interests": ["Webデザイン", "UI/UX", "音楽"],
        "active": true
    }');
```

> **用語解説：**
>
> - **JSON_OBJECT()関数**：キーと値のペアからJSONオブジェクトを作成する関数です。
> - **JSON_ARRAY()関数**：複数の値からJSON配列を作成する関数です。

JSON データの検索と抽出

JSON_EXTRACT() 関数：値の抽出

JSON データから特定の値を抽出するには `JSON_EXTRACT()` 関数または `->` 演算子を使用します。

> **用語解説：**
>
> - **JSON_EXTRACT()関数**：JSONドキュメントから指定したパスの値を抽出する関数です。
> - **JSONパス**：JSON内の特定の要素を指定するための記法（例：$.name、$.contact.email）です。

基本構文

```
JSON_EXTRACT(json_doc, path)
-- または短縮形
json_doc -> path
-- 引用符を除去する場合
json_doc ->> path
```

例1：基本的な値の抽出

```
SELECT student_id,
       profile_data -> '$.name' AS 名前,
       profile_data -> '$.birth_year' AS 生年,
       profile_data ->> '$.contact.email' AS メールアドレス
FROM student_profiles;
```

実行結果：

student_id	名前	生年	メールアドレス
301	"黒沢春馬"	2003	kurosawa@example.com
302	"新垣愛留"	2004	aragaki@example.com

student_id	名前	生年	メールアドレス
303	"柴崎春花"	2003	shibasaki@example.com

例2：配列要素の抽出

```
SELECT student_id,
       profile_data ->> '$.name' AS 名前,
       profile_data -> '$.skills[0]' AS 主要スキル1,
       profile_data -> '$.skills[1]' AS 主要スキル2,
       JSON_LENGTH(profile_data -> '$.skills') AS スキル数
FROM student_profiles;
```

実行結果：

student_id	名前	主要スキル1	主要スキル2	スキル数
301	黒沢春馬	"Java"	"HTML"	3
302	新垣愛留	"Python"	"データ分析"	3
303	柴崎春花	"JavaScript"	"React"	3

JSON検索関数

JSON_CONTAINS()関数：値の存在確認

JSON内に特定の値が含まれているかを確認します。

```
-- 特定のスキルを持つ学生を検索
SELECT student_id,
       profile_data ->> '$.name' AS 名前,
       profile_data -> '$.skills' AS スキル一覧
FROM student_profiles
WHERE JSON_CONTAINS(profile_data -> '$.skills', '"Java"');
```

JSON_SEARCH()関数：値の位置検索

JSON内で特定の値の位置（パス）を検索します。

```
-- メールアドレスに'kurosawa'を含む学生を検索
SELECT student_id,
       profile_data ->> '$.name' AS 名前,
       JSON_SEARCH(profile_data, 'one', '%kurosawa%') AS 検索結果パス
FROM student_profiles
WHERE JSON_SEARCH(profile_data, 'one', '%kurosawa%') IS NOT NULL;
```

WHERE句でのJSON検索

例1：JSON値での条件絞り込み

```
-- 2003年生まれの学生を検索
SELECT student_id,
       profile_data ->> '$.name' AS 名前,
       profile_data -> '$.birth_year' AS 生年
FROM student_profiles
WHERE profile_data -> '$.birth_year' = 2003;
```

例2：JSON配列での検索

```
-- 'プログラミング'に興味がある学生を検索
SELECT student_id,
       profile_data ->> '$.name' AS 名前,
       profile_data -> '$.interests' AS 興味分野
FROM student_profiles
WHERE JSON_CONTAINS(profile_data -> '$.interests', '"プログラミング"');
```

JSON データの更新

JSON_SET()関数：値の設定・更新

既存のJSONデータに新しい値を設定したり、既存の値を更新したりします。

> **用語解説**：
>
> - **JSON_SET()関数**：JSONドキュメント内の指定したパスに値を設定する関数です。パスが存在しない場合は新規作成し、存在する場合は上書きします。

例1：既存の値を更新

```
-- 学生の生年を更新
UPDATE student_profiles
SET profile_data = JSON_SET(profile_data, '$.birth_year', 2004)
WHERE student_id = 301;
```

例2：新しい要素を追加

```
-- 学生に新しい情報（GPAスコア）を追加
UPDATE student_profiles
SET profile_data = JSON_SET(
    profile_data,
```

```
    '$.gpa', 3.8,
    '$.last_updated', NOW()
)
WHERE student_id = 301;
```

JSON_INSERT()関数：新規追加のみ

値が存在しない場合のみ新しい値を追加します。

```
-- 存在しない場合のみ、学習時間の情報を追加
UPDATE student_profiles
SET profile_data = JSON_INSERT(profile_data, '$.study_hours_per_week', 25)
WHERE student_id = 302;
```

JSON_REPLACE()関数：既存値の置換のみ

既存の値のみを置換し、新しい要素は追加しません。

```
-- 既存のメールアドレスのみを更新
UPDATE student_profiles
SET profile_data = JSON_REPLACE(
    profile_data,
    '$.contact.email', 'new.kurosawa@example.com'
)
WHERE student_id = 301;
```

JSON配列の操作

JSON_ARRAY_APPEND()関数：配列要素の追加

既存の配列に新しい要素を追加します。

```
-- スキルに新しい項目を追加
UPDATE student_profiles
SET profile_data = JSON_ARRAY_APPEND(
    profile_data,
    '$.skills', 'Docker'
)
WHERE student_id = 301;
```

JSON_ARRAY_INSERT()関数：配列の特定位置に挿入

配列の指定した位置に新しい要素を挿入します。

```
-- スキルの最初に新しい項目を挿入
UPDATE student_profiles
SET profile_data = JSON_ARRAY_INSERT(
    profile_data,
    '$.skills[0]', 'Git'
)
WHERE student_id = 302;
```

JSON_REMOVE()関数：要素の削除

JSON から特定の要素を削除します。

```
-- 特定のスキルを削除
UPDATE student_profiles
SET profile_data = JSON_REMOVE(profile_data, '$.skills[2]')
WHERE student_id = 303;
```

JSON データの集計と分析

例1：スキル別の学生数集計

```
SELECT skill,
       COUNT(*) AS 学生数
FROM student_profiles,
     JSON_TABLE(profile_data, '$.skills[*]'
                COLUMNS (skill VARCHAR(50) PATH '$')) AS skills_table
GROUP BY skill
ORDER BY 学生数 DESC;
```

例2：生年別の分析

```
SELECT profile_data -> '$.birth_year' AS 生年,
       COUNT(*) AS 人数,
       GROUP_CONCAT(profile_data ->> '$.name') AS 学生名一覧
FROM student_profiles
GROUP BY profile_data -> '$.birth_year';
```

例3：アクティブな学生の興味分野分析

```
SELECT interest,
       COUNT(*) AS 興味を持つ学生数
FROM student_profiles,
     JSON_TABLE(profile_data, '$.interests[*]'
```

```
                COLUMNS (interest VARCHAR(100) PATH '$')) AS interests_table
WHERE profile_data -> '$.active' = true
GROUP BY interest
ORDER BY 興味を持つ学生数 DESC;
```

実践例：学校データベースでのJSON活用

例1：講座設定情報の管理

```
-- 講座の詳細設定を格納するテーブル
CREATE TABLE course_settings (
    course_id VARCHAR(16) PRIMARY KEY,
    settings JSON,
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);

-- 講座設定の挿入
INSERT INTO course_settings (course_id, settings) VALUES
('1', '{
    "grading": {
        "midterm_weight": 0.3,
        "final_weight": 0.4,
        "assignments_weight": 0.2,
        "participation_weight": 0.1
    },
    "requirements": ["基礎数学", "コンピュータリテラシー"],
    "tools": ["Eclipse", "MySQL Workbench"],
    "difficulty_level": "初級",
    "estimated_hours": 60
}');
```

例2：教室の詳細設備情報

```
-- 教室の設備詳細を管理
CREATE TABLE classroom_details (
    classroom_id VARCHAR(16) PRIMARY KEY,
    equipment JSON,
    FOREIGN KEY (classroom_id) REFERENCES classrooms(classroom_id)
);

-- 設備詳細の挿入
INSERT INTO classroom_details (classroom_id, equipment) VALUES
('101A', '{
    "computers": {
        "count": 30,
        "specs": {
            "cpu": "Intel Core i5",
            "ram": "8GB",
            "storage": "256GB SSD"
```

```
        },
        "software": ["Windows 11", "Office 2021", "Visual Studio"]
    },
    "projector": {
        "model": "EPSON EB-X41",
        "resolution": "1024x768"
    },
    "network": {
        "wifi": true,
        "ethernet_ports": 32
    },
    "accessibility": ["車椅子対応", "視覚支援機器"]
}');
```

練習問題

問題9-3-1

student_profiles テーブルを作成し、学生ID 304 の学生について、名前「森下風凛」、生年2003、スキル「["HTML", "CSS", "Photoshop"]」、メールアドレス「morishita@example.com」の情報を挿入するSQLを書いてください。

問題9-3-2

student_profiles テーブルから、すべての学生の名前とスキルの数を取得するSQLを書いてください。

問題9-3-3

student_profiles テーブルから、「HTML」スキルを持つ学生の名前とメールアドレスを取得するSQLを書いてください。

問題9-3-4

student_profiles テーブルで、学生ID 301 のスキル配列に「Kubernetes」を追加するSQLを書いてください。

問題9-3-5

student_profiles テーブルから、2003年生まれの学生の名前と興味分野を取得するSQLを書いてください。

問題9-3-6

student_profiles テーブルで、学生ID 302 の緊急連絡先電話番号を「090-5555-6666」に更新するSQLを書いてください。

問題9-3-7

student_profiles テーブルから、興味分野に「AI」を含む学生数を取得するSQLを書いてください。

問題9-3-8

student_profiles テーブルで、学生ID 303 のプロフィールから2番目のスキルを削除するSQLを書いてください。

問題9-3-9

student_profiles テーブルから、各学生について名前と「スキル：〇〇」という形式でスキルを1つの文字列として表示するSQLを書いてください。

問題9-3-10

student_profiles テーブルから、アクティブな学生（active = true）の中で、メールアドレスのドメインが「example.com」の学生の名前を取得するSQLを書いてください。

解答と詳細な解説

解答9-3-1

```
-- テーブル作成
CREATE TABLE student_profiles (
    student_id BIGINT PRIMARY KEY,
    profile_data JSON,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (student_id) REFERENCES students(student_id)
);

-- データ挿入
INSERT INTO student_profiles (student_id, profile_data) VALUES
(304, '{
    "name": "森下風凛",
    "birth_year": 2003,
    "skills": ["HTML", "CSS", "Photoshop"],
    "contact": {
        "email": "morishita@example.com"
    }
}');
```

解説：

- JSONデータ型のカラムを持つテーブルを作成
- JSON文字列形式でデータを挿入
- 外部キー制約により、既存の学生IDとの整合性を保つ
- 配列形式でスキル情報を格納し、オブジェクト形式で連絡先情報を構造化

解答9-3-2

```
SELECT profile_data ->> '$.name' AS 名前,
       JSON_LENGTH(profile_data -> '$.skills') AS スキル数
FROM student_profiles;
```

解説：

- `->>` 演算子で引用符なしの文字列として名前を取得
- `JSON_LENGTH()` 関数で配列の要素数を取得
- `->` 演算子でJSONパスを指定してskills配列にアクセス

解答9-3-3

```
SELECT profile_data ->> '$.name' AS 名前,
       profile_data ->> '$.contact.email' AS メールアドレス
FROM student_profiles
WHERE JSON_CONTAINS(profile_data -> '$.skills', '"HTML"');
```

解説：

- `JSON_CONTAINS()` 関数で配列内の特定の値の存在を確認
- 検索する値は引用符で囲む必要がある（`"HTML"`）
- ネストしたオブジェクトのプロパティは`.`で連結してアクセス

解答9-3-4

```
UPDATE student_profiles
SET profile_data = JSON_ARRAY_APPEND(profile_data, '$.skills', 'Kubernetes')
WHERE student_id = 301;
```

解説：

- `JSON_ARRAY_APPEND()` 関数で既存の配列に新しい要素を追加
- 配列の末尾に要素が追加される
- WHERE句で特定の学生のみを対象に更新

解答9-3-5

```
SELECT profile_data ->> '$.name' AS 名前,
       profile_data -> '$.interests' AS 興味分野
FROM student_profiles
WHERE profile_data -> '$.birth_year' = 2003;
```

解説：

- JSON内の数値フィールドでの条件指定
- 興味分野は配列形式で格納されているため、そのまま表示
- 等価比較演算子（=）でJSON内の値を直接比較

解答9-3-6

```
UPDATE student_profiles
SET profile_data = JSON_SET(profile_data, '$.contact.emergency_phone', '090-5555-
6666')
WHERE student_id = 302;
```

解説：

- `JSON_SET()` 関数で既存の値を更新または新規追加
- ネストしたオブジェクト内のプロパティを指定
- 値が存在しない場合は新規作成、存在する場合は上書き

解答9-3-7

```
SELECT COUNT(*) AS AI興味学生数
FROM student_profiles
WHERE JSON_CONTAINS(profile_data -> '$.interests', '"AI"');
```

解説：

- `COUNT(*)` で条件に合致する行数を集計
- `JSON_CONTAINS()` で配列内の特定の値の存在を確認
- 文字列値の検索時は引用符で囲む必要がある

解答9-3-8

```
UPDATE student_profiles
SET profile_data = JSON_REMOVE(profile_data, '$.skills[1]')
WHERE student_id = 303;
```

解説：

- `JSON_REMOVE()` 関数で指定したパスの要素を削除
- 配列のインデックスは0から始まるため、`[1]`は2番目の要素
- 削除後は残りの要素が自動的に詰められる

解答9-3-9

```
SELECT profile_data ->> '$.name' AS 名前,
       CONCAT('スキル：',
              REPLACE(
                  REPLACE(
                      JSON_EXTRACT(profile_data, '$.skills'),
                      '[', ''
                  ),
                  ']', ''
```

```
                )
        ) AS スキル一覧
FROM student_profiles;
```

解説：

- `JSON_EXTRACT()` で配列全体を取得
- `REPLACE()` 関数で配列の角括弧を除去
- `CONCAT()` で「スキル：」プレフィックスを追加
- 複数の文字列関数を組み合わせてデータを整形

解答9-3-10

```
SELECT profile_data ->> '$.name' AS 名前
FROM student_profiles
WHERE profile_data -> '$.active' = true
  AND profile_data ->> '$.contact.email' LIKE '%@example.com';
```

解説：

- 複数のJSON条件を組み合わせ
- `profile_data -> '$.active' = true` でアクティブ状態を確認
- `LIKE` 演算子でメールアドレスのドメイン部分をパターンマッチング
- `->>` 演算子で文字列として値を取得してLIKE検索に使用

まとめ

この章では、MySQLにおけるJSONデータの格納と操作について学びました：

1. **JSONデータ型の基本**：従来のテキスト型との違いとJSONの利点
2. **JSON データの作成**：JSON_OBJECT()、JSON_ARRAY()による構造化データの作成
3. **JSON データの検索**：JSON_EXTRACT()、->、->>演算子による値の抽出
4. **JSON データの更新**：JSON_SET()、JSON_INSERT()、JSON_REPLACE()による値の変更
5. **JSON配列の操作**：要素の追加、挿入、削除
6. **JSON データの集計**：JSON_TABLE()を使った集計分析
7. **実践的な活用**：学校データベースでの具体的なJSON活用例

JSONデータ型は、複雑な構造を持つデータを効率的に格納し、柔軟に操作できる強力な機能です。特に設定情報、メタデータ、ユーザープロファイルなどの用途で威力を発揮します。ただし、リレーショナルデータベースの正規化の原則とのバランスを考慮して、適切な場面で使用することが重要です。

次のセクションでは、「地理空間データ：位置情報の取り扱い」について学び、位置データの格納と空間検索の技術を習得します。

9-4. 地理空間データ：位置情報の取り扱い

はじめに

現代のアプリケーションでは、位置情報を扱うことが非常に多くなっています。GPSデータ、地図サービス、配送管理、店舗検索など、様々な場面で地理空間データが活用されています。

学校データベースでも、以下のような場面で地理空間データが役立ちます：

- **キャンパスマップ**：各建物や教室の正確な位置情報
- **通学路管理**：学生の通学経路や最寄り駅からの距離
- **災害対策**：避難経路や集合場所の位置データ
- **施設管理**：駐車場、食堂、図書館などの施設位置
- **イベント会場**：屋外授業や学園祭の会場位置
- **近隣情報**：最寄りの病院、コンビニ、公共交通機関

この章では、MySQLの地理空間データ型と空間関数について学び、位置情報を効率的に格納・検索・分析する方法を習得します。

> **用語解説**：
>
> - **地理空間データ（Spatial Data）**：地球上の位置や形状を表現するデータで、点、線、面などの幾何学的情報を含みます。
> - **GIS（地理情報システム）**：地理空間データを収集、格納、分析、表示するためのシステムです。
> - **座標系**：地球上の位置を数値で表現するための参照系統です。

地理空間データ型の基本

MySQLでは、地理空間データを格納するための専用データ型が用意されています。これらは**OpenGIS標準**に準拠しており、様々なGISソフトウェアとの互換性があります。

主要な地理空間データ型

データ型	説明	例
POINT	単一の点（座標）	建物の位置、教室の場所
LINESTRING	線（複数の点を結んだ線）	通学路、道路
POLYGON	多角形（閉じられた領域）	建物の敷地、キャンパスエリア
MULTIPOINT	複数の点の集合	複数の出入口
MULTILINESTRING	複数の線の集合	複数の通学路
MULTIPOLYGON	複数の多角形の集合	複数の建物群
GEOMETRY	上記すべてを格納可能	混在する地理データ

> **用語解説**：
>
> - **POINT**：経度・緯度で表される地球上の一点です。
> - **LINESTRING**：複数の点を順番に結んだ線分です。
> - **POLYGON**：閉じられた領域を表す多角形です。

座標系とSRID

地理空間データを扱う際は、**座標系（Coordinate Reference System）**の理解が重要です。

主要な座標系

1. **WGS84（SRID: 4326）**
    - 世界測地系、GPS で使用される標準座標系
    - 経度・緯度で表現（例：東京駅 139.7673, 35.6809）
2. **Web メルカトル（SRID: 3857）**
    - Google Maps、OpenStreetMap で使用
    - メートル単位での距離計算に適している

> **用語解説：**
>
> - **SRID（Spatial Reference System Identifier）**：座標系を識別するための番号です。
> - **WGS84**：世界測地系の標準で、GPSシステムで使用される座標系です。

学校データベースへの地理空間データの追加

学校の位置情報を管理するためのテーブルを作成しましょう。

建物位置テーブルの作成

```
-- 建物の位置情報テーブル
CREATE TABLE building_locations (
    building_id VARCHAR(16) PRIMARY KEY,
    building_name VARCHAR(100) NOT NULL,
    location POINT NOT NULL SRID 4326,
    entrance_points MULTIPOINT SRID 4326,  -- 複数の入口
    building_area POLYGON SRID 4326,       -- 建物の敷地
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    -- 空間インデックスの作成
    SPATIAL INDEX idx_location (location),
    SPATIAL INDEX idx_area (building_area)
);
```

キャンパス施設テーブルの作成

```
-- キャンパス内施設の位置情報テーブル
CREATE TABLE campus_facilities (
    facility_id VARCHAR(16) PRIMARY KEY,
    facility_name VARCHAR(100) NOT NULL,
    facility_type VARCHAR(50),  -- 'parking', 'restaurant', 'library', etc.
    location POINT NOT NULL SRID 4326,
    service_area POLYGON SRID 4326,  -- サービス提供エリア
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    SPATIAL INDEX idx_facility_location (location)
);
```

地理空間データの挿入

POINT データの挿入

```
-- 建物の位置データを挿入（東京都内の架空の学校）
INSERT INTO building_locations (building_id, building_name, location,
building_area) VALUES
('BLDG001', '1号館（本館）',
 ST_GeomFromText('POINT(139.7000 35.6500)', 4326),
 ST_GeomFromText('POLYGON((139.6995 35.6495, 139.7005 35.6495, 139.7005 35.6505,
139.6995 35.6505, 139.6995 35.6495))', 4326)
),
('BLDG002', '2号館（実習棟）',
 ST_GeomFromText('POINT(139.7010 35.6510)', 4326),
 ST_GeomFromText('POLYGON((139.7005 35.6505, 139.7015 35.6505, 139.7015 35.6515,
139.7005 35.6515, 139.7005 35.6505))', 4326)
),
('BLDG003', '3号館（講義棟）',
 ST_GeomFromText('POINT(139.6990 35.6520)', 4326),
 ST_GeomFromText('POLYGON((139.6985 35.6515, 139.6995 35.6515, 139.6995 35.6525,
139.6985 35.6525, 139.6985 35.6515))', 4326)
);
```

施設データの挿入

```
-- キャンパス施設の位置データを挿入
INSERT INTO campus_facilities (facility_id, facility_name, facility_type,
location) VALUES
('FAC001', '学生食堂', 'restaurant', ST_GeomFromText('POINT(139.7005 35.6508)',
4326)),
('FAC002', '図書館', 'library', ST_GeomFromText('POINT(139.6995 35.6515)', 4326)),
('FAC003', '駐車場A', 'parking', ST_GeomFromText('POINT(139.6985 35.6490)',
4326)),
('FAC004', '駐車場B', 'parking', ST_GeomFromText('POINT(139.7020 35.6525)',
4326)),
('FAC005', '保健室', 'medical', ST_GeomFromText('POINT(139.7002 35.6502)', 4326));
```

> **用語解説：**
>
> - **ST_GeomFromText()**：WKT（Well-Known Text）形式の文字列から地理空間オブジェクトを作成する関数です。
> - **WKT（Well-Known Text）**：地理空間データをテキストで表現するための標準形式です。

地理空間データの検索と表示

基本的な位置データの取得

```
-- 建物の位置情報を表示
SELECT building_id,
       building_name,
       ST_AsText(location) AS 位置座標,
       ST_X(location) AS 経度,
       ST_Y(location) AS 緯度
FROM building_locations;
```

実行結果：

building_id	building_name	位置座標	経度	緯度
BLDG001	1号館（本館）	POINT(139.7 35.65)	139.7	35.65
BLDG002	2号館（実習棟）	POINT(139.701 35.651)	139.701	35.651
BLDG003	3号館（講義棟）	POINT(139.699 35.652)	139.699	35.652

距離の計算

```
-- 1号館から各施設までの距離を計算（メートル単位）
SELECT f.facility_name,
       f.facility_type,
       ROUND(ST_Distance_Sphere(
           b.location,
           f.location
       ), 2) AS 距離_メートル
FROM building_locations b, campus_facilities f
WHERE b.building_id = 'BLDG001'
ORDER BY 距離_メートル;
```

実行結果：

facility_name	facility_type	距離_メートル
保健室	medical	35.18
学生食堂	restaurant	89.34
図書館	library	167.23
駐車場A	parking	223.89
駐車場B	parking	278.45

空間検索と空間関数

範囲内検索（バッファ検索）

特定の地点から一定距離内にある施設を検索します。

```
-- 1号館から200メートル以内の施設を検索
SELECT f.facility_name,
       f.facility_type,
       ROUND(ST_Distance_Sphere(
           (SELECT location FROM building_locations WHERE building_id =
'BLDG001'),
           f.location
       ), 2) AS 距離_メートル
FROM campus_facilities f
WHERE ST_Distance_Sphere(
    (SELECT location FROM building_locations WHERE building_id = 'BLDG001'),
    f.location
) <= 200
ORDER BY 距離_メートル;
```

最寄り施設の検索

```
-- 各建物から最も近い駐車場を検索
SELECT b.building_name,
       f.facility_name AS 最寄り駐車場,
       ROUND(ST_Distance_Sphere(b.location, f.location), 2) AS 距離_メートル
FROM building_locations b
JOIN campus_facilities f ON f.facility_type = 'parking'
WHERE (b.building_id, ST_Distance_Sphere(b.location, f.location)) IN (
    SELECT b2.building_id, MIN(ST_Distance_Sphere(b2.location, f2.location))
    FROM building_locations b2, campus_facilities f2
    WHERE f2.facility_type = 'parking'
    GROUP BY b2.building_id
);
```

エリア内検索（ポリゴン内検索）

```
-- キャンパスエリア内の施設を検索（キャンパス全体のポリゴンを仮定）
SET @campus_area = ST_GeomFromText('
    POLYGON((
        139.6980 35.6485, 139.7025 35.6485,
        139.7025 35.6530, 139.6980 35.6530,
        139.6980 35.6485
    ))', 4326);

SELECT facility_name,
       facility_type,
       ST_AsText(location) AS 位置
```

```
FROM campus_facilities
WHERE ST_Contains(@campus_area, location);
```

複雑な空間分析

通学路データの管理

```
-- 通学路テーブルの作成
CREATE TABLE commute_routes (
    route_id VARCHAR(16) PRIMARY KEY,
    route_name VARCHAR(100),
    start_point POINT NOT NULL SRID 4326,
    end_point POINT NOT NULL SRID 4326,
    route_path LINESTRING NOT NULL SRID 4326,
    distance_meters DECIMAL(10,2),
    estimated_time_minutes INT,
    route_type VARCHAR(20), -- 'walking', 'bicycle', 'bus'

    SPATIAL INDEX idx_start (start_point),
    SPATIAL INDEX idx_end (end_point),
    SPATIAL INDEX idx_path (route_path)
);
```

通学路データの挿入

```
-- 最寄り駅からのルートを挿入
INSERT INTO commute_routes (route_id, route_name, start_point, end_point,
route_path, distance_meters, estimated_time_minutes, route_type) VALUES
('ROUTE001', 'oo駅から1号館',
 ST_GeomFromText('POINT(139.6950 35.6450)', 4326),
 ST_GeomFromText('POINT(139.7000 35.6500)', 4326),
 ST_GeomFromText('LINESTRING(139.6950 35.6450, 139.6975 35.6475, 139.7000
35.6500)', 4326),
 623.50, 8, 'walking'
),
('ROUTE002', 'バス停から2号館',
 ST_GeomFromText('POINT(139.7030 35.6480)', 4326),
 ST_GeomFromText('POINT(139.7010 35.6510)', 4326),
 ST_GeomFromText('LINESTRING(139.7030 35.6480, 139.7020 35.6495, 139.7010
35.6510)', 4326),
 387.20, 5, 'walking'
);
```

ルート分析

```
-- 各ルートの詳細分析
SELECT route_name,
       route_type,
       distance_meters,
       estimated_time_minutes,
       ROUND(distance_meters / estimated_time_minutes * 60 / 1000, 2) AS 平均時速
_kmh,
       ST_AsText(start_point) AS 出発地点,
       ST_AsText(end_point) AS 到着地点
FROM commute_routes;
```

地理空間インデックスとパフォーマンス

空間インデックスの効果確認

```
-- インデックスを使用した高速な近傍検索
EXPLAIN SELECT facility_name, facility_type
FROM campus_facilities
WHERE ST_Distance_Sphere(
    location,
    ST_GeomFromText('POINT(139.7000 35.6500)', 4326)
) <= 100;
```

大量データでのパフォーマンス最適化

```
-- 学生の居住地データテーブル（大量データを想定）
CREATE TABLE student_addresses (
    student_id BIGINT PRIMARY KEY,
    address_text VARCHAR(200),
    location POINT SRID 4326,
    distance_to_school DECIMAL(8,2), -- 事前計算した距離

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    SPATIAL INDEX idx_student_location (location),
    INDEX idx_distance (distance_to_school)
);
```

実践例：学校データベースでの地理空間活用

例1：教室予約システムとの連携

```
-- 教室の位置情報を既存のclassroomsテーブルに追加
ALTER TABLE classrooms
ADD COLUMN location POINT SRID 4326,
ADD COLUMN building_floor INT,
```

```
ADD SPATIAL INDEX idx_classroom_location (location);

-- 教室の位置データを更新
UPDATE classrooms
SET location = ST_GeomFromText('POINT(139.7000 35.6500)', 4326),
    building_floor = 1
WHERE classroom_id = '101A';

UPDATE classrooms
SET location = ST_GeomFromText('POINT(139.7000 35.6502)', 4326),
    building_floor = 1
WHERE classroom_id = '102B';
```

例2：最適な教室配置の分析

```
-- 学生の移動距離を最小化する教室配置分析
SELECT
    cs1.course_id AS 前の授業,
    cs2.course_id AS 次の授業,
    c1.classroom_name AS 前の教室,
    c2.classroom_name AS 次の教室,
    ROUND(ST_Distance_Sphere(c1.location, c2.location), 2) AS 移動距離_メートル,
    CASE
        WHEN ST_Distance_Sphere(c1.location, c2.location) > 200 THEN '長距離移動'
        WHEN ST_Distance_Sphere(c1.location, c2.location) > 100 THEN '中距離移動'
        ELSE '短距離移動'
    END AS 移動分類
FROM course_schedule cs1
JOIN course_schedule cs2 ON DATE(cs1.schedule_date) = DATE(cs2.schedule_date)
    AND cs1.period_id = cs2.period_id - 1
JOIN classrooms c1 ON cs1.classroom_id = c1.classroom_id
JOIN classrooms c2 ON cs2.classroom_id = c2.classroom_id
WHERE c1.location IS NOT NULL AND c2.location IS NOT NULL
ORDER BY 移動距離_メートル DESC;
```

例3：災害時避難計画

```
-- 避難場所テーブル
CREATE TABLE evacuation_points (
    point_id VARCHAR(16) PRIMARY KEY,
    point_name VARCHAR(100),
    capacity INT,
    location POINT NOT NULL SRID 4326,
    evacuation_area POLYGON SRID 4326,

    SPATIAL INDEX idx_evac_location (location)
);

-- 各建物から最寄りの避難場所を計算
```

```
SELECT b.building_name,
       ep.point_name AS 最寄り避難場所,
       ROUND(ST_Distance_Sphere(b.location, ep.location), 2) AS 距離_メートル,
       ROUND(ST_Distance_Sphere(b.location, ep.location) / 80 * 60, 1) AS 避難時間
_秒
FROM building_locations b
CROSS JOIN evacuation_points ep
WHERE (b.building_id, ST_Distance_Sphere(b.location, ep.location)) IN (
    SELECT b2.building_id, MIN(ST_Distance_Sphere(b2.location, ep2.location))
    FROM building_locations b2, evacuation_points ep2
    GROUP BY b2.building_id
);
```

練習問題

問題9-4-1

building_locations テーブルから、すべての建物の位置を経度・緯度形式で表示し、建物名と共に取得するSQLを書いてください。

問題9-4-2

campus_facilities テーブルに新しい施設「コンビニ」（facility_id: 'FAC006'、位置: 139.7008, 35.6512）を挿入するSQLを書いてください。

問題9-4-3

2号館（BLDG002）から最も近い施設を1つ見つけ、その施設名、タイプ、距離を表示するSQLを書いてください。

問題9-4-4

すべての建物から150メートル以内にある施設を検索し、建物名、施設名、距離を表示するSQLを書いてください。

問題9-4-5

各施設タイプ（facility_type）ごとに、施設数と平均的な他施設からの距離を計算するSQLを書いてください。

問題9-4-6

3号館（BLDG003）を中心とした半径300メートルの円形エリア内にある施設を検索するSQLを書いてください。

問題9-4-7

各建物から最も遠い施設を見つけ、建物名、最も遠い施設名、その距離を表示するSQLを書いてください。

問題9-4-8

'restaurant'タイプまたは'library'タイプの施設で、1号館から200メートル以内にあるものを検索するSQLを書いてください。

問題9-4-9

建物の敷地面積（building_area）を平方メートル単位で計算して表示するSQLを書いてください。

問題9-4-10

キャンパスの中心点（すべての建物の重心）を計算し、そこから各施設までの距離を表示するSQLを書いてください。

解答と詳細な解説

解答9-4-1

```
SELECT building_id,
       building_name,
       ST_X(location) AS 経度,
       ST_Y(location) AS 緯度,
       ST_AsText(location) AS 位置座標
FROM building_locations;
```

解説：

- `ST_X()`関数でPOINTの経度（X座標）を取得
- `ST_Y()`関数でPOINTの緯度（Y座標）を取得
- `ST_AsText()`関数で地理空間データをWKT形式のテキストとして表示
- 地理空間データの基本的な表示方法

解答9-4-2

```
INSERT INTO campus_facilities (facility_id, facility_name, facility_type,
location)
VALUES ('FAC006', 'コンビニ', 'convenience_store',
        ST_GeomFromText('POINT(139.7008 35.6512)', 4326));
```

解説：

- `ST_GeomFromText()`関数でWKT形式の文字列から地理空間オブジェクトを作成
- SRID 4326（WGS84座標系）を指定
- POINTデータの標準的な挿入方法

解答9-4-3

```
SELECT f.facility_name,
       f.facility_type,
```

```
       ROUND(ST_Distance_Sphere(b.location, f.location), 2) AS 距離_メートル
FROM building_locations b, campus_facilities f
WHERE b.building_id = 'BLDG002'
ORDER BY ST_Distance_Sphere(b.location, f.location)
LIMIT 1;
```

解説：

- ST_Distance_Sphere()関数で球面距離を計算（メートル単位）
- WHERE句で2号館を指定
- ORDER BYで距離の昇順に並べてLIMIT 1で最短距離の施設を取得
- ROUND()で距離を小数点第2位まで表示

解答9-4-4

```
SELECT b.building_name,
       f.facility_name,
       f.facility_type,
       ROUND(ST_Distance_Sphere(b.location, f.location), 2) AS 距離_メートル
FROM building_locations b, campus_facilities f
WHERE ST_Distance_Sphere(b.location, f.location) <= 150
ORDER BY b.building_name, 距離_メートル;
```

解説：

- クロス結合で全建物と全施設の組み合わせを生成
- WHERE句で150メートル以内の条件を指定
- 建物名と距離で並び替えて見やすく表示

解答9-4-5

```
SELECT f1.facility_type,
       COUNT(*) AS 施設数,
       ROUND(AVG(min_distances.最短距離), 2) AS 平均最短距離_メートル
FROM campus_facilities f1
JOIN (
    SELECT f.facility_id,
           MIN(ST_Distance_Sphere(f.location, f2.location)) AS 最短距離
    FROM campus_facilities f, campus_facilities f2
    WHERE f.facility_id != f2.facility_id
    GROUP BY f.facility_id
) min_distances ON f1.facility_id = min_distances.facility_id
GROUP BY f1.facility_type;
```

解説：

- サブクエリで各施設から他の施設への最短距離を計算

- 自分自身を除外するため`f.facility_id != f2.facility_id`条件を追加
- 施設タイプごとにグループ化して平均を計算

解答9-4-6

```
SELECT facility_name,
       facility_type,
       ROUND(ST_Distance_Sphere(
           (SELECT location FROM building_locations WHERE building_id =
'BLDG003'),
           location
       ), 2) AS 距離_メートル
FROM campus_facilities
WHERE ST_Distance_Sphere(
    (SELECT location FROM building_locations WHERE building_id = 'BLDG003'),
    location
) <= 300
ORDER BY 距離_メートル;
```

解説：

- サブクエリで3号館の位置を取得
- `ST_Distance_Sphere() <= 300`で300メートル以内の条件を指定
- 円形範囲内の施設検索の典型的なパターン

解答9-4-7

```
SELECT b.building_name,
       f.facility_name,
       f.facility_type,
       ROUND(ST_Distance_Sphere(b.location, f.location), 2) AS 最長距離_メートル
FROM building_locations b, campus_facilities f
WHERE (b.building_id, ST_Distance_Sphere(b.location, f.location)) IN (
    SELECT b2.building_id, MAX(ST_Distance_Sphere(b2.location, f2.location))
    FROM building_locations b2, campus_facilities f2
    GROUP BY b2.building_id
);
```

解説：

- サブクエリで各建物から施設への最大距離を計算
- WHERE句のIN条件で最大距離を持つ組み合わせのみを抽出
- 複合条件による最大値検索のパターン

解答9-4-8

```
SELECT facility_name,
       facility_type,
       ROUND(ST_Distance_Sphere(
           (SELECT location FROM building_locations WHERE building_id =
'BLDG001'),
           location
       ), 2) AS 距離_メートル
FROM campus_facilities
WHERE facility_type IN ('restaurant', 'library')
  AND ST_Distance_Sphere(
      (SELECT location FROM building_locations WHERE building_id = 'BLDG001'),
      location
  ) <= 200
ORDER BY 距離_メートル;
```

解説：

- IN演算子で複数の施設タイプを指定
- AND条件で距離制限と施設タイプの両方の条件を満たす施設を検索
- 複合条件での空間検索の例

解答9-4-9

```
SELECT building_id,
       building_name,
       ROUND(ST_Area(ST_Transform(building_area, 3857)), 2) AS 敷地面積_平方メートル
FROM building_locations
WHERE building_area IS NOT NULL;
```

解説：

- `ST_Area()`関数でポリゴンの面積を計算
- `ST_Transform()`で座標系をメートル単位の投影座標系（3857）に変換
- WGS84（4326）は角度単位のため、面積計算には投影座標系への変換が必要

解答9-4-10

```
SELECT f.facility_name,
       f.facility_type,
       ROUND(ST_Distance_Sphere(
           ST_Centroid(ST_Collect(
               (SELECT ST_Collect(location) FROM building_locations)
           )),
           f.location
       ), 2) AS 中心点からの距離_メートル
FROM campus_facilities f
ORDER BY 中心点からの距離_メートル;
```

解説：

- `ST_Collect()`で複数のPOINTを集約してMULTIPOINTを作成
- `ST_Centroid()`で重心（中心点）を計算
- サブクエリで全建物の位置を集約し、その重心から各施設までの距離を計算
- 複数の空間関数を組み合わせた高度な計算例

まとめ

この章では、MySQLにおける地理空間データの格納と操作について学びました：

1. **地理空間データ型の基本**：POINT、LINESTRING、POLYGONなどの基本データ型
2. **座標系の理解**：WGS84（SRID 4326）とWeb メルカトル（SRID 3857）の使い分け
3. **地理空間データの作成**：ST_GeomFromText()による地理空間オブジェクトの作成
4. **空間検索**：ST_Distance_Sphere()による距離計算と範囲検索
5. **空間分析**：最寄り施設検索、エリア内検索、ルート分析
6. **パフォーマンス最適化**：空間インデックスの活用
7. **実践的な活用**：学校データベースでの位置情報管理と分析

地理空間データは、現代のアプリケーションにおいて重要な役割を果たしています。特に学校のような物理的な施設を多く持つ組織では、効率的な施設管理、災害対策、学生サービスの向上などに大きく貢献します。

空間インデックスを適切に設定し、座標系を正しく理解して使用することで、大量の位置データでも高速な検索と分析が可能になります。また、GISソフトウェアとの連携により、より高度な地理空間分析も実現できます。

次のセクションでは、「全文検索：テキスト検索の最適化」について学び、大量のテキストデータから効率的に情報を検索する技術を習得します。

9-5. 全文検索：テキスト検索の最適化

はじめに

現代の情報システムでは、大量のテキストデータから必要な情報を素早く見つけることが重要です。従来のLIKE演算子による部分一致検索では、大量のデータに対して十分なパフォーマンスを発揮できません。

学校データベースでも、以下のような場面で高度なテキスト検索が必要になります：

- **講座検索**：キーワードから関連する講座を素早く見つける
- **教材検索**：シラバスや教材の内容から関連情報を検索
- **学生レポート検索**：提出されたレポートの内容検索
- **FAQ検索**：よくある質問から適切な回答を検索
- **図書館システム**：書籍の内容や概要からの検索
- **学習記録検索**：学習ログやコメントからの情報抽出

MySQLの全文検索機能を使うことで、これらの要求に効率的に対応できます。この章では、全文検索の基本概念から実践的な活用方法まで学びます。

> **用語解説：**
>
> - **全文検索（Full-Text Search）**：テキスト全体を対象として、キーワードや語句を効率的に検索する技術です。
> - **インデックス**：検索速度を向上させるためのデータ構造で、全文検索では特別な全文インデックスを使用します。
> - **関連度スコア**：検索結果の各行がクエリにどの程度関連しているかを数値で表したものです。

MySQLの全文検索の基本

MySQLでは、**MyISAM**と**InnoDB**の両方のストレージエンジンで全文検索がサポートされています。全文検索は、通常のインデックスとは異なる**全文インデックス（Full-Text Index）**を使用します。

全文検索の利点

1. **高速検索**：大量のテキストデータでも高速に検索可能
2. **関連度ランキング**：検索結果を関連度順に並び替え
3. **柔軟な検索**：単語の組み合わせや除外条件の指定
4. **自然言語検索**：日常言語による検索クエリ
5. **最小語長制御**：短すぎる単語の除外

対応データ型

全文検索は以下のデータ型で使用できます：

- CHAR
- VARCHAR
- TEXT（TEXT、MEDIUMTEXT、LONGTEXT）

> **用語解説：**
>
> - **ストレージエンジン**：データの格納と取得を行うMySQLの内部コンポーネントです。
> - **MyISAM**：高速読み取りに特化したストレージエンジンです。
> - **InnoDB**：トランザクション処理をサポートする汎用ストレージエンジンです。

全文検索用テーブルの準備

学校データベースで全文検索を活用するため、教材やレポートのテーブルを作成しましょう。

教材テーブルの作成

```
-- 教材・資料テーブル
CREATE TABLE course_materials (
    material_id VARCHAR(16) PRIMARY KEY,
    course_id VARCHAR(16),
    title VARCHAR(200) NOT NULL,
    description TEXT,
    content LONGTEXT,
    material_type VARCHAR(50), -- 'textbook', 'handout', 'slides', 'video_script'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (course_id) REFERENCES courses(course_id),

    -- 全文インデックスの作成
    FULLTEXT INDEX ft_title (title),
    FULLTEXT INDEX ft_description (description),
    FULLTEXT INDEX ft_content (content),
    FULLTEXT INDEX ft_all (title, description, content)
) ENGINE=InnoDB;
```

学生レポートテーブルの作成

```
-- 学生レポート・課題テーブル
CREATE TABLE student_reports (
    report_id VARCHAR(16) PRIMARY KEY,
    student_id BIGINT,
    course_id VARCHAR(16),
    title VARCHAR(200) NOT NULL,
    abstract TEXT,
    content LONGTEXT,
    keywords VARCHAR(500),
    submission_date DATE,

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id),

    -- 全文インデックスの作成
    FULLTEXT INDEX ft_report_title (title),
    FULLTEXT INDEX ft_report_abstract (abstract),
    FULLTEXT INDEX ft_report_content (content),
    FULLTEXT INDEX ft_report_keywords (keywords),
    FULLTEXT INDEX ft_report_all (title, abstract, content, keywords)
) ENGINE=InnoDB;
```

サンプルデータの挿入

全文検索のテストのため、サンプルデータを挿入します。

教材データの挿入

```
INSERT INTO course_materials (material_id, course_id, title, description, content,
material_type) VALUES
('MAT001', '1', 'ITの基礎知識 第1章', 'コンピュータとインターネットの基本概念について学習します',
 'コンピュータは現代社会において欠かせない道具となっています。デジタル技術の発展により、私たちの生活は大きく変化しました。インターネットの普及によって、世界中の情報にアクセスできるようになり、コミュニケーションの方法も革新されました。本章では、コンピュータの基本構成、CPU、メモリ、ストレージについて詳しく解説し、インターネットの仕組みやプロトコルについても学習します。', 'textbook'),

('MAT002', '3', 'Cプログラミング入門', 'C言語の基本文法とプログラミングの基礎を学びます',
```

```
 'C言語は1972年にデニス・リッチーによって開発されたプログラミング言語です。多くの現代的なプログラミング言語の基礎となっており、システムプログラミングやアプリケーション開発に広く使用されています。本教材では、変数、データ型、制御構造、関数、配列、ポインタなどのC言語の基本概念を学習します。実際のプログラム例を通して、アルゴリズムの実装方法やデバッグ技術についても習得します。', 'textbook'),

('MAT003', '5', 'データベース設計の基礎', 'リレーショナルデータベースの設計原理と正規化について',
 'データベースは現代の情報システムにおいて中核的な役割を果たします。効率的なデータベース設計により、データの整合性を保ち、パフォーマンスを最適化できます。本章では、エンティティ関係図（ERD）の作成方法、正規化の理論と実践、インデックスの設計、SQLクエリの最適化について学習します。実際のビジネス要件から論理設計、物理設計までの一連のプロセスを通して、実践的なスキルを身につけます。', 'textbook'),

('MAT004', '4', 'Webアプリケーション開発実践', 'HTMLからJavaScriptまでのWeb技術総合',
 'Webアプリケーション開発では、フロントエンドとバックエンドの両方の技術を理解する必要があります。HTML5によるマークアップ、CSSによるスタイリング、JavaScriptによる動的な機能実装について学習します。また、Node.jsを使用したサーバーサイド開発、データベース連携、RESTfulAPIの設計と実装についても解説します。実際のプロジェクトを通して、モダンなWeb開発のベストプラクティスを身につけます。', 'textbook');
```

学生レポートデータの挿入

```
INSERT INTO student_reports (report_id, student_id, course_id, title, abstract,
content, keywords, submission_date) VALUES
('RPT001', 301, '1', 'AIの社会的影響について', 'AI技術が社会に与える影響と課題について考察',
 '人工知能（AI）技術の急速な発展により、私たちの社会は大きな変革期を迎えています。機械学習、深層学習、自然言語処理などのAI技術は、医療、教育、交通、製造業など様々な分野で活用され始めています。しかし、AI技術の普及には多くの課題も存在します。雇用への影響、プライバシーの問題、アルゴリズムのバイアス、倫理的な判断などについて社会全体で議論する必要があります。本レポートでは、AI技術の現状と将来の可能性について分析し、社会がどのように対応すべきかについて提言します。',
 '人工知能,AI,機械学習,社会影響,倫理,プライバシー', '2025-05-20'),

('RPT002', 302, '3', 'アルゴリズムの効率性比較', 'ソートアルゴリズムの計算量比較と実装',
 'コンピュータサイエンスにおいて、アルゴリズムの効率性は重要な要素です。本レポートでは、代表的なソートアルゴリズムの時間計算量と空間計算量を理論的に分析し、実際の実装による性能測定を行いました。バブルソート、選択ソート、挿入ソート、マージソート、クイックソートについて、異なるデータサイズでの実行時間を測定し、Big-O記法による理論値との比較を行いました。結果として、データサイズが大きくなるにつれて、効率的なアルゴリズムの重要性が増すことが確認できました。',
 'アルゴリズム,ソート,計算量,効率性,プログラミング,C言語', '2025-05-18'),

('RPT003', 303, '5', 'データベース正規化の実践', '学校管理システムの正規化プロセス',
 'データベース設計において正規化は重要なプロセスです。本レポートでは、学校管理システムを例として、第1正規形から第3正規形までの正規化プロセスを実践しました。非正規化されたデータから開始し、主キーの識別、部分関数従属性の排除、推移関数従属性の排除を段階的に実施しました。正規化により、データの冗長性が削減され、更新異常、挿入異常、削除異常が解決されることを確認しました。また、パフォーマンスとのトレードオフについても考察し、実際の運用における正規化レベルの選択指針を提示します。',
 'データベース,正規化,関数従属,主キー,冗長性,SQL', '2025-05-22'),

('RPT004', 304, '4', 'レスポンシブWebデザインの実装', 'モバイルファーストアプローチによるWebサイト構築',
 'モバイルデバイスの普及により、レスポンシブWebデザインは現代のWeb開発において必須の技術となっています。本レポートでは、モバイルファーストアプローチによるレスポンシブWebサイトの設計と実装について報告します。CSS3のメディアクエリ、フレキシブルグリッドレイアウト、可変画像などの技術を組み合わせ、デスクトップ、タブレット、スマートフォンに対応した学校紹介Webサイトを制作しました。パフォーマンス最適化、アクセシビリティ、SEO対策についても考慮し、実
```

```
用的なWebサイトの構築プロセスを実践しました。',
 'Web開発,レスポンシブデザイン,モバイルファースト,CSS,HTML,JavaScript', '2025-05-25');
```

基本的な全文検索

MATCH ... AGAINST 構文

全文検索は `MATCH ... AGAINST` 構文を使用します。

基本構文

```
SELECT カラム名 FROM テーブル名
WHERE MATCH(検索対象カラム) AGAINST('検索語句');
```

例1：基本的な全文検索

```
-- 教材のタイトルから「プログラミング」を検索
SELECT material_id, title, material_type
FROM course_materials
WHERE MATCH(title) AGAINST('プログラミング');
```

実行結果：

material_id	title	material_type
MAT002	Cプログラミング入門	textbook

例2：複数カラムでの検索

```
-- タイトルと説明から「データベース」を検索
SELECT material_id, title, description
FROM course_materials
WHERE MATCH(title, description) AGAINST('データベース');
```

実行結果：

material_id	title	description
MAT003	データベース設計の基礎	リレーショナルデータベースの設計原理と正規化について

例3：関連度スコア付きの検索

```

```

```
-- 関連度スコアを含む検索結果
SELECT material_id,
       title,
       MATCH(title, description, content) AGAINST('AI 人工知能') AS 関連度スコア
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('AI 人工知能')
ORDER BY 関連度スコア DESC;
```

全文検索のモード

MySQLの全文検索には複数のモードがあります。

1. 自然言語モード（Natural Language Mode）

デフォルトのモードで、自然言語での検索クエリを処理します。

```
-- 自然言語モードでの検索（デフォルト）
SELECT title, abstract
FROM student_reports
WHERE MATCH(title, abstract, content) AGAINST('アルゴリズム効率性' IN NATURAL LANGUAGE
MODE)
ORDER BY MATCH(title, abstract, content) AGAINST('アルゴリズム効率性') DESC;
```

2. ブール言語モード（Boolean Mode）

論理演算子を使用してより詳細な検索条件を指定できます。

> **用語解説：**
>
> - **ブール言語モード**：論理演算子（+、-、*など）を使用して複雑な検索条件を指定できるモードです。

ブール演算子

演算子	意味	例
+	必須単語	+プログラミング +C言語
-	除外単語	+データベース -MySQL
*	ワイルドカード	プログラミング*
""	完全一致フレーズ	"Web開発"
()	グループ化	+(AI 人工知能) -倫理

例：ブール言語モードの活用

```
-- 「プログラミング」を含み、「Java」を含まない教材を検索
SELECT material_id, title
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('+プログラミング -Java' IN BOOLEAN
MODE);
```

```
-- 「データベース」と「設計」の両方を含む教材を検索
SELECT material_id, title
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('+データベース +設計' IN BOOLEAN
MODE);
```

```
-- 「Web」で始まる単語を含む教材を検索
SELECT material_id, title
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('Web*' IN BOOLEAN MODE);
```

3. クエリ拡張モード（Query Expansion Mode）

初回検索結果を基に自動的にクエリを拡張して、より関連性の高い結果を取得します。

```
-- クエリ拡張モードでの検索
SELECT material_id, title,
       MATCH(title, description, content) AGAINST('プログラミング' WITH QUERY
EXPANSION) AS スコア
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('プログラミング' WITH QUERY
EXPANSION)
ORDER BY スコア DESC;
```

高度な全文検索クエリ

例1：複合条件での検索

```
-- AI関連のレポートで、特定の期間に提出されたものを検索
SELECT r.title,
       r.abstract,
       s.student_name,
       r.submission_date,
       MATCH(r.title, r.abstract, r.content) AGAINST('AI 人工知能 機械学習') AS 関
連度
FROM student_reports r
JOIN students s ON r.student_id = s.student_id
```

```
WHERE MATCH(r.title, r.abstract, r.content) AGAINST('AI 人工知能 機械学習')
  AND r.submission_date >= '2025-05-01'
ORDER BY 関連度 DESC;
```

例2：カテゴリ別の検索

```
-- 講座別の教材検索
SELECT c.course_name,
       m.title,
       m.material_type,
       MATCH(m.title, m.description, m.content) AGAINST('実践 応用') AS 関連度
FROM course_materials m
JOIN courses c ON m.course_id = c.course_id
WHERE MATCH(m.title, m.description, m.content) AGAINST('実践 応用')
ORDER BY c.course_name, 関連度 DESC;
```

例3：キーワード頻度分析

```
-- 学生レポートでよく使用されるキーワードの分析
SELECT keywords,
       COUNT(*) AS 使用回数,
       AVG(MATCH(title, abstract, content) AGAINST('技術 開発')) AS 平均関連度
FROM student_reports
WHERE keywords IS NOT NULL
  AND MATCH(title, abstract, content) AGAINST('技術 開発')
GROUP BY keywords
ORDER BY 使用回数 DESC;
```

全文検索のパフォーマンス最適化

1. 全文インデックスの管理

```
-- 既存テーブルに全文インデックスを追加
ALTER TABLE course_materials
ADD FULLTEXT INDEX ft_combined (title, description);

-- 全文インデックスの削除
ALTER TABLE course_materials
DROP INDEX ft_combined;
```

2. 最小語長の設定

MySQLの設定変数で検索対象となる最小語長を調整できます。

```
-- 現在の最小語長設定を確認
SHOW VARIABLES LIKE 'ft_min_word_len';

-- InnoDB全文検索の最小語長を確認
SHOW VARIABLES LIKE 'innodb_ft_min_token_size';
```

3. ストップワードの管理

一般的すぎて検索に有用でない単語（ストップワード）の管理も重要です。

```
-- 現在のストップワード設定を確認
SHOW VARIABLES LIKE 'ft_stopword_file';
```

実践例：学校データベースでの全文検索活用

例1：統合検索システム

```
-- 教材とレポートを統合した検索システム
(SELECT 'material' AS type,
        material_id AS id,
        title,
        description AS summary,
        MATCH(title, description, content) AGAINST('データベース 設計') AS 関連度
 FROM course_materials
 WHERE MATCH(title, description, content) AGAINST('データベース 設計'))
UNION ALL
(SELECT 'report' AS type,
        report_id AS id,
        title,
        abstract AS summary,
        MATCH(title, abstract, content) AGAINST('データベース 設計') AS 関連度
 FROM student_reports
 WHERE MATCH(title, abstract, content) AGAINST('データベース 設計'))
ORDER BY 関連度 DESC
LIMIT 10;
```

例2：推薦システムの基盤

```
-- 学生の提出レポートに基づく関連教材の推薦
SELECT DISTINCT m.material_id,
       m.title AS 推薦教材,
       c.course_name,
       AVG(MATCH(m.title, m.description, m.content)
           AGAINST(r.keywords)) AS 関連度
FROM student_reports r
JOIN course_materials m ON r.course_id = m.course_id
```

```
JOIN courses c ON m.course_id = c.course_id
WHERE r.student_id = 301
  AND r.keywords IS NOT NULL
  AND MATCH(m.title, m.description, m.content) AGAINST(r.keywords) > 0
GROUP BY m.material_id, m.title, c.course_name
ORDER BY 関連度 DESC
LIMIT 5;
```

例3：学習進度分析

```
-- 特定トピックに関する学習進度の分析
SELECT s.student_name,
       COUNT(r.report_id) AS レポート数,
       AVG(MATCH(r.title, r.abstract, r.content)
           AGAINST('プログラミング アルゴリズム')) AS 平均関連度,
       MAX(r.submission_date) AS 最新提出日
FROM students s
LEFT JOIN student_reports r ON s.student_id = r.student_id
WHERE MATCH(r.title, r.abstract, r.content)
      AGAINST('プログラミング アルゴリズム') > 0
GROUP BY s.student_id, s.student_name
ORDER BY 平均関連度 DESC;
```

練習問題

問題9-5-1

course_materials テーブルから、タイトルに「Web」を含む教材を全文検索で取得し、関連度スコアと共に表示するSQLを書いてください。

問題9-5-2

student_reports テーブルから、「AI」と「機械学習」の両方を含むレポートをブール言語モードで検索するSQLを書いてください。

問題9-5-3

course_materials テーブルから、「データベース」を含むが「MySQL」を含まない教材をブール言語モードで検索するSQLを書いてください。

問題9-5-4

student_reports テーブルから、「プログラミング」で始まる単語を含むレポートを検索し、学生名と提出日も表示するSQLを書いてください。

問題9-5-5

course_materials と student_reports の両方から「設計」に関連する内容を検索し、タイプ（教材/レポート）別に結果を表示するSQLを書いてください。

問題9-5-6

「アルゴリズム」をキーワードとして、student_reports テーブルをクエリ拡張モードで検索し、関連度の高い順に上位3件を取得するSQLを書いてください。

問題9-5-7

course_materials テーブルから、完全一致フレーズ「プログラミング言語」を含む教材をブール言語モードで検索するSQLを書いてください。

問題9-5-8

student_reports テーブルから、2025年5月に提出されたレポートの中で「技術」に関連する内容を検索し、講座名も表示するSQLを書いてください。

問題9-5-9

各講座について、その講座の教材で最も多く使用されているキーワード（全文検索スコアが最も高い語句）を「システム」として分析するSQLを書いてください。

問題9-5-10

student_reports テーブルから、「効率」または「最適化」を含むレポートをブール言語モードで検索し、学生別にレポート数と平均関連度を計算するSQLを書いてください。

解答と詳細な解説

解答9-5-1

```
SELECT material_id,
       title,
       MATCH(title) AGAINST('Web') AS 関連度スコア
FROM course_materials
WHERE MATCH(title) AGAINST('Web')
ORDER BY 関連度スコア DESC;
```

解説：

- `MATCH(title) AGAINST('Web')`で全文検索を実行
- SELECT句にも同じ式を記述して関連度スコアを取得
- WHERE句で全文検索条件を指定し、ORDER BYで関連度順に並び替え
- 関連度スコアは検索語句とのマッチ度を数値で表現

解答9-5-2

```
SELECT report_id, title, abstract
FROM student_reports
WHERE MATCH(title, abstract, content) AGAINST('+AI +機械学習' IN BOOLEAN MODE);
```

解説：

- `IN BOOLEAN MODE`でブール言語モードを指定
- `+`演算子で両方の単語が必須であることを指定
- 「AI」と「機械学習」の両方を含む文書のみが検索される
- 複数カラム（title, abstract, content）を検索対象に指定

解答9-5-3

```
SELECT material_id, title, description
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('+データベース -MySQL' IN BOOLEAN
MODE);
```

解説：

- `+データベース`で「データベース」を必須条件として指定
- `-MySQL`で「MySQL」を含む文書を除外
- ブール言語モードの除外機能を活用した検索条件
- 特定の単語を含む文書を除外したい場合に有効

解答9-5-4

```
SELECT r.report_id,
       r.title,
       s.student_name,
       r.submission_date
FROM student_reports r
JOIN students s ON r.student_id = s.student_id
WHERE MATCH(r.title, r.abstract, r.content) AGAINST('プログラミング*' IN BOOLEAN MODE)
ORDER BY MATCH(r.title, r.abstract, r.content) AGAINST('プログラミング*') DESC;
```

解説：

- `プログラミング*`のワイルドカード検索で「プログラミング」で始まる単語をマッチ
- JOIN句で学生テーブルと結合して学生名を取得
- ワイルドカード（*）は語幹マッチングに使用
- 関連度順に並び替えて結果を表示

解答9-5-5

```
(SELECT 'material' AS タイプ,
        material_id AS ID,
        title AS タイトル,
        MATCH(title, description, content) AGAINST('設計') AS 関連度
 FROM course_materials
```

```
 WHERE MATCH(title, description, content) AGAINST('設計'))
UNION ALL
(SELECT 'report' AS タイプ,
        report_id AS ID,
        title AS タイトル,
        MATCH(title, abstract, content) AGAINST('設計') AS 関連度
 FROM student_reports
 WHERE MATCH(title, abstract, content) AGAINST('設計'))
ORDER BY 関連度 DESC;
```

解説：

- UNION ALLで教材テーブルとレポートテーブルの検索結果を結合
- 各サブクエリで同じ構造の結果セットを作成
- タイプ列で教材とレポートを区別
- 統合検索システムの基本的なパターン

解答9-5-6

```
SELECT report_id,
       title,
       abstract,
       MATCH(title, abstract, content) AGAINST('アルゴリズム' WITH QUERY EXPANSION)
AS 関連度
FROM student_reports
WHERE MATCH(title, abstract, content) AGAINST('アルゴリズム' WITH QUERY EXPANSION)
ORDER BY 関連度 DESC
LIMIT 3;
```

解説：

- WITH QUERY EXPANSIONでクエリ拡張モードを使用
- 初回検索結果から関連語を自動抽出してクエリを拡張
- より幅広い関連文書を取得できる
- LIMIT 3で上位3件のみを取得

解答9-5-7

```
SELECT material_id, title
FROM course_materials
WHERE MATCH(title, description, content) AGAINST('"プログラミング言語"' IN BOOLEAN
MODE);
```

解説：

- ダブルクォート（""）で完全一致フレーズを指定
- 「プログラミング言語」という連続した語句のみがマッチ

- 単語の順序と隣接性が重要な検索に使用
- ブール言語モードでのフレーズ検索機能

解答9-5-8

```
SELECT r.report_id,
       r.title,
       c.course_name,
       r.submission_date,
       MATCH(r.title, r.abstract, r.content) AGAINST('技術') AS 関連度
FROM student_reports r
JOIN courses c ON r.course_id = c.course_id
WHERE r.submission_date >= '2025-05-01'
  AND r.submission_date < '2025-06-01'
  AND MATCH(r.title, r.abstract, r.content) AGAINST('技術')
ORDER BY 関連度 DESC;
```

解説：

- WHERE句で日付範囲と全文検索条件を組み合わせ
- JOIN句で講座テーブルと結合して講座名を取得
- 時間軸とテキスト内容の両方で絞り込み
- 複合条件での実用的な検索例

解答9-5-9

```
SELECT c.course_name,
       COUNT(*) AS システム関連教材数,
       AVG(MATCH(m.title, m.description, m.content) AGAINST('システム')) AS 平均関連
度
FROM courses c
JOIN course_materials m ON c.course_id = m.course_id
WHERE MATCH(m.title, m.description, m.content) AGAINST('システム')
GROUP BY c.course_id, c.course_name
ORDER BY 平均関連度 DESC;
```

解説：

- GROUP BYで講座別に集計
- COUNT(*)で該当する教材数を計算
- AVG()で平均関連度を算出
- 講座別のキーワード使用傾向分析の例

解答9-5-10

```
SELECT s.student_name,
       COUNT(r.report_id) AS レポート数,
```

```
        AVG(MATCH(r.title, r.abstract, r.content) AGAINST('効率 最適化')) AS 平均関
連度
FROM students s
JOIN student_reports r ON s.student_id = r.student_id
WHERE MATCH(r.title, r.abstract, r.content) AGAINST('効率 OR 最適化' IN BOOLEAN
MODE)
GROUP BY s.student_id, s.student_name
ORDER BY 平均関連度 DESC;
```

解説：

- OR演算子で「効率」または「最適化」のいずれかを含む条件
- GROUP BYで学生別に集計
- JOIN句で学生テーブルと結合
- 学生の関心分野分析に活用できるクエリパターン

まとめ

この章では、MySQLにおける全文検索機能について学びました：

1. **全文検索の基本概念**：従来のLIKE検索との違いと利点
2. **全文インデックスの作成**：FULLTEXT INDEXによる検索性能の向上
3. **MATCH...AGAINST構文**：基本的な全文検索の実行方法
4. **検索モード**：自然言語モード、ブール言語モード、クエリ拡張モード
5. **ブール演算子**：+、-、*、""、()を使った高度な検索条件指定
6. **関連度スコア**：検索結果の品質評価と並び替え
7. **パフォーマンス最適化**：インデックス管理とシステム設定
8. **実践的な活用**：統合検索、推薦システム、学習分析への応用

全文検索は、大量のテキストデータを扱う現代のアプリケーションにおいて必須の技術です。特に学校データベースのような教育コンテンツを多く含むシステムでは、学習者が必要な情報を素早く見つけるために重要な役割を果たします。

適切な全文インデックスの設計と、用途に応じた検索モードの選択により、ユーザーエクスペリエンスを大幅に向上させることができます。また、関連度スコアを活用することで、検索結果の品質を定量的に評価し、推薦システムや学習分析などの高度な機能も実現できます。

次のセクションでは、「XML：構造化マークアップデータの操作」について学び、階層構造を持つデータの格納と操作技術を習得します。

9-6. XML：構造化マークアップデータの操作

はじめに

XML（eXtensible Markup Language）は、データの構造と内容を記述するためのマークアップ言語です。HTMLと似た記法を使いながら、より柔軟で拡張性の高いデータ表現が可能です。

学校データベースでは、以下のような場面でXMLが活用されます：

- **成績表のエクスポート**：他システムとの成績データ交換
- **シラバス管理**：階層構造を持つカリキュラム情報の保存
- **設定ファイル**：システム設定やユーザー設定の保存
- **レポート形式**：構造化されたレポートテンプレートの管理
- **外部システム連携**：他の教育システムとのデータ交換
- **メタデータ管理**：教材やコンテンツの詳細情報

MySQLでは、XMLデータを格納し、XPath式を使って要素を抽出・更新する機能が提供されています。この章では、XMLデータの基本概念から実践的な操作方法まで学びます。

> **用語解説**：
>
> - **XML（eXtensible Markup Language）**：拡張可能なマークアップ言語で、データの構造化と交換のために設計されています。
> - **XPath**：XML文書内の要素や属性を指定するための言語です。
> - **マークアップ**：テキストに構造や意味を付加するためのタグ記法です。

XMLの基本構造

XMLは階層構造を持つマークアップ言語で、以下の要素から構成されます：

XML文書の基本構成

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <element attribute="value">
        <child>Content</child>
        <child>Another content</child>
    </element>
</root>
```

主要な構成要素

1. **XML宣言**：文書の最初に記述される宣言
2. **要素（Element）**：開始タグと終了タグで囲まれた内容
3. **属性（Attribute）**：要素に付加される名前と値のペア
4. **テキストコンテンツ**：要素内のテキスト内容
5. **CDATA**：特殊文字を含むテキストデータ

> **用語解説**：
>
> - **要素（Element）**：XMLの基本構成単位で、`<tag>content</tag>`の形式で表現されます。
> - **属性（Attribute）**：要素に追加情報を付与するための`name="value"`形式のデータです。
> - **CDATA**：「Character Data」の略で、XMLパーサーが解析しない生データを示します。

XMLデータ用テーブルの作成

学校データベースでXMLデータを活用するためのテーブルを作成しましょう。

シラバステーブルの作成

```
-- シラバス情報をXML形式で保存するテーブル
CREATE TABLE course_syllabi (
    syllabus_id VARCHAR(16) PRIMARY KEY,
    course_id VARCHAR(16),
    academic_year INT,
    syllabus_data XML,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

成績記録テーブルの作成

```
-- 詳細な成績情報をXML形式で保存するテーブル
CREATE TABLE detailed_grades (
    record_id VARCHAR(16) PRIMARY KEY,
    student_id BIGINT,
    course_id VARCHAR(16),
    academic_term VARCHAR(10),
    grade_data XML,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

システム設定テーブルの作成

```
-- システム設定をXML形式で保存するテーブル
CREATE TABLE system_configurations (
    config_id VARCHAR(16) PRIMARY KEY,
    config_name VARCHAR(100),
    config_data XML,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

XMLデータの挿入

シラバスデータの挿入

```
INSERT INTO course_syllabi (syllabus_id, course_id, academic_year, syllabus_data)
VALUES
('SYL001', '1', 2025, '<?xml version="1.0" encoding="UTF-8"?>
<syllabus>
    <course_info>
        <title>ITのための基礎知識</title>
        <credits>2</credits>
        <instructor>寺内鞍</instructor>
        <description>現代社会で必要とされるIT基礎知識を学習します</description>
    </course_info>
    <schedule>
        <weeks total="15">
            <week number="1">
                <topic>コンピュータの基本概念</topic>
                <objectives>
                    <objective>ハードウェアとソフトウェアの違いを理解する</objective>
                    <objective>基本的なコンピュータの構成要素を説明できる</objective>
                </objectives>
            </week>
            <week number="2">
                <topic>インターネットとネットワーク</topic>
                <objectives>
                    <objective>インターネットの基本的な仕組みを理解する</objective>
                    <objective>IPアドレスとドメイン名について説明できる</objective>
                </objectives>
            </week>
            <week number="3">
                <topic>データとファイルシステム</topic>
                <objectives>
                    <objective>デジタルデータの概念を理解する</objective>
                    <objective>ファイルとフォルダの関係を説明できる</objective>
                </objectives>
            </week>
        </weeks>
    </schedule>
    <evaluation>
        <method type="exam" weight="60">期末試験</method>
        <method type="assignment" weight="30">課題提出</method>
        <method type="participation" weight="10">授業参加</method>
    </evaluation>
</syllabus>'),

('SYL002', '3', 2025, '<?xml version="1.0" encoding="UTF-8"?>
<syllabus>
    <course_info>
        <title>Cプログラミング演習</title>
        <credits>3</credits>
        <instructor>寺内鞍</instructor>
        <description>C言語の基本文法とプログラミング技法を実習を通して学習します</description>
    </course_info>
    <schedule>
        <weeks total="15">
            <week number="1">
```

```
                <topic>C言語の基礎とプログラム構造</topic>
                <objectives>
                    <objective>C言語の歴史と特徴を理解する</objective>
                    <objective>基本的なプログラム構造を理解する</objective>
                </objectives>
            </week>
            <week number="2">
                <topic>変数とデータ型</topic>
                <objectives>
                    <objective>基本データ型を理解し使い分けできる</objective>
                    <objective>変数の宣言と初期化を正しく行える</objective>
                </objectives>
            </week>
        </weeks>
    </schedule>
    <evaluation>
        <method type="practical" weight="50">実習課題</method>
        <method type="exam" weight="40">期末試験</method>
        <method type="participation" weight="10">授業参加</method>
    </evaluation>
</syllabus>');
```

成績データの挿入

```
INSERT INTO detailed_grades (record_id, student_id, course_id, academic_term,
grade_data) VALUES
('GRD001', 301, '1', '2025Q2', '<?xml version="1.0" encoding="UTF-8"?>
<grade_record>
    <student_info>
        <student_id>301</student_id>
        <name>黒沢春馬</name>
    </student_info>
    <course_info>
        <course_id>1</course_id>
        <title>ITのための基礎知識</title>
    </course_info>
    <assessments>
        <assessment type="midterm">
            <date>2025-05-20</date>
            <score>85</score>
            <max_score>100</max_score>
            <comments>基本概念の理解は良好。応用問題でやや苦戦。</comments>
        </assessment>
        <assessment type="assignment">
            <date>2025-05-10</date>
            <score>45</score>
            <max_score>50</max_score>
            <comments>期限内提出。内容も充実している。</comments>
        </assessment>
        <assessment type="participation">
            <score>8</score>
```

```
            <max_score>10</max_score>
            <comments>積極的な授業参加が見られる。</comments>
        </assessment>
    </assessments>
    <final_grade>
        <letter_grade>B</letter_grade>
        <numerical_grade>82</numerical_grade>
        <gpa_points>3.0</gpa_points>
    </final_grade>
</grade_record>'),

('GRD002', 302, '3', '2025Q2', '<?xml version="1.0" encoding="UTF-8"?>
<grade_record>
    <student_info>
        <student_id>302</student_id>
        <name>新垣愛留</name>
    </student_info>
    <course_info>
        <course_id>3</course_id>
        <title>Cプログラミング演習</title>
    </course_info>
    <assessments>
        <assessment type="practical">
            <date>2025-05-25</date>
            <score>92</score>
            <max_score>100</max_score>
            <comments>優秀なプログラミングスキル。コードの可読性も高い。</comments>
        </assessment>
        <assessment type="exam">
            <date>2025-06-15</date>
            <score>88</score>
            <max_score>100</max_score>
            <comments>理論的理解も十分。実践的な問題解決能力も高い。</comments>
        </assessment>
        <assessment type="participation">
            <score>9</score>
            <max_score>10</max_score>
            <comments>他の学生への協力も積極的。</comments>
        </assessment>
    </assessments>
    <final_grade>
        <letter_grade>A</letter_grade>
        <numerical_grade>90</numerical_grade>
        <gpa_points>4.0</gpa_points>
    </final_grade>
</grade_record>');
```

XML関数による基本操作

MySQLでは、XML データを操作するための関数が提供されています。

ExtractValue()関数：XML要素の抽出

> **用語解説：**
>
> - **ExtractValue()関数**：XML文書からXPath式を使用して特定の要素の値を抽出する関数です。

基本構文

```
ExtractValue(xml_document, xpath_expression)
```

例1：基本的な要素の抽出

```
-- シラバスから講座タイトルを抽出
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/course_info/title') AS 講座名,
       ExtractValue(syllabus_data, '/syllabus/course_info/credits') AS 単位数,
       ExtractValue(syllabus_data, '/syllabus/course_info/instructor') AS 担当教師
FROM course_syllabi;
```

実行結果：

syllabus_id	講座名	単位数	担当教師
SYL001	ITのための基礎知識	2	寺内鞍
SYL002	Cプログラミング演習	3	寺内鞍

例2：属性値の抽出

```
-- 評価方法と重みを抽出
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/evaluation/method[1]') AS 評価方法1,
       ExtractValue(syllabus_data, '/syllabus/evaluation/method[1]/@weight') AS 重み1,
       ExtractValue(syllabus_data, '/syllabus/evaluation/method[2]') AS 評価方法2,
       ExtractValue(syllabus_data, '/syllabus/evaluation/method[2]/@weight') AS 重み2
FROM course_syllabi;
```

実行結果：

syllabus_id	評価方法1	重み1	評価方法2	重み2
SYL001	期末試験	60	課題提出	30
SYL002	実習課題	50	期末試験	40

UpdateXML()関数：XML要素の更新

> **用語解説：**
>
> - **UpdateXML()関数**：XML文書内の特定の要素を新しい値で更新する関数です。

基本構文

```
UpdateXML(xml_document, xpath_expression, new_value)
```

例1：要素値の更新

```
-- 講座の単位数を更新
UPDATE course_syllabi
SET syllabus_data = UpdateXML(
    syllabus_data,
    '/syllabus/course_info/credits',
    '4'
)
WHERE syllabus_id = 'SYL002';
```

例2：複数要素の更新

```
-- 担当教師名を更新
UPDATE course_syllabi
SET syllabus_data = UpdateXML(
    UpdateXML(
        syllabus_data,
        '/syllabus/course_info/instructor',
        '田尻朋美'
    ),
    '/syllabus/course_info/description',
    'プログラミングの基礎からアルゴリズムまでを実践的に学習します'
)
WHERE syllabus_id = 'SYL002';
```

XPath式の基本

XPathは XML 文書内の要素を指定するための言語です。

基本的なXPath記法

記法	説明	例
`/`	絶対パス	`/syllabus/course_info/title`
`//`	任意の階層	`//title`

記法	説明	例
`.`	現在のノード	`./title`
`..`	親ノード	`../title`
`@`	属性	`/@weight`
`[]`	述語（条件）	`/method[@type="exam"]`
`[n]`	n番目の要素	`/method[1]`

高度なXPath式の例

```
-- 特定の評価タイプを持つ方法を検索
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/evaluation/method[@type="exam"]') AS
試験評価,
       ExtractValue(syllabus_data,
'/syllabus/evaluation/method[@type="exam"]/@weight') AS 試験重み
FROM course_syllabi
WHERE ExtractValue(syllabus_data,
'/syllabus/evaluation/method[@type="exam"]/@weight') > 40;
```

```
-- 週番号を条件とした学習目標の抽出
SELECT syllabus_id,
       ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[@number="1"]/topic') AS 第1週トピック,
       ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[@number="1"]/objectives/objective[1]') AS 第1週目標
1
FROM course_syllabi;
```

複雑なXMLクエリの例

例1：成績データの分析

```
-- 各学生の最終成績と数値評価を抽出
SELECT record_id,
       ExtractValue(grade_data, '/grade_record/student_info/name') AS 学生名,
       ExtractValue(grade_data, '/grade_record/course_info/title') AS 講座名,
       ExtractValue(grade_data, '/grade_record/final_grade/letter_grade') AS 評価,
       ExtractValue(grade_data, '/grade_record/final_grade/numerical_grade') AS 数
値評価,
       ExtractValue(grade_data, '/grade_record/final_grade/gpa_points') AS GPA
FROM detailed_grades;
```

例2：評価別の成績統計

```
-- 中間試験の成績を分析
SELECT
    ExtractValue(grade_data, '/grade_record/student_info/name') AS 学生名,
    ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/score') AS 中間試験得点,
    ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/max_score') AS 満点,
    ROUND(
        ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/score') /
        ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/max_score') * 100,
        1
    ) AS 達成率
FROM detailed_grades
WHERE ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/score') IS NOT NULL;
```

例3：条件付きデータ抽出

```
-- 優秀な成績（A評価）の学生を検索
SELECT
    ExtractValue(grade_data, '/grade_record/student_info/name') AS 学生名,
    ExtractValue(grade_data, '/grade_record/course_info/title') AS 講座名,
    ExtractValue(grade_data, '/grade_record/final_grade/numerical_grade') AS 数値
評価
FROM detailed_grades
WHERE ExtractValue(grade_data, '/grade_record/final_grade/letter_grade') = 'A'
ORDER BY ExtractValue(grade_data, '/grade_record/final_grade/numerical_grade')
DESC;
```

XML データの検索と集計

例1：シラバスの内容検索

```
-- 特定のキーワードを含むシラバスを検索
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/course_info/title') AS 講座名
FROM course_syllabi
WHERE ExtractValue(syllabus_data, '/syllabus/course_info/description') LIKE '%プロ
グラミング%'
   OR ExtractValue(syllabus_data, '//topic') LIKE '%プログラミング%';
```

例2：成績統計の計算

```
-- 講座別の平均成績を計算
SELECT
    ExtractValue(grade_data, '/grade_record/course_info/title') AS 講座名,
    COUNT(*) AS 受講者数,
    AVG(CAST(ExtractValue(grade_data, '/grade_record/final_grade/numerical_grade')
AS DECIMAL(5,2))) AS 平均点,
    AVG(CAST(ExtractValue(grade_data, '/grade_record/final_grade/gpa_points') AS
DECIMAL(3,2))) AS 平均GPA
FROM detailed_grades
GROUP BY ExtractValue(grade_data, '/grade_record/course_info/title');
```

例3：学習目標の分析

```
-- 各講座の学習目標数を集計
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/course_info/title') AS 講座名,
       ExtractValue(syllabus_data, '/syllabus/schedule/weeks/@total') AS 総週数,
       (SELECT COUNT(*)
        FROM (SELECT ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[1]/objectives/objective[1]') AS obj1
              UNION ALL
              SELECT ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[1]/objectives/objective[2]') AS obj2
              UNION ALL
              SELECT ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[2]/objectives/objective[1]') AS obj3
              UNION ALL
              SELECT ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[2]/objectives/objective[2]') AS obj4
             ) objectives_table
        WHERE objectives_table.obj1 IS NOT NULL AND objectives_table.obj1 != '')
AS 目標数
FROM course_syllabi;
```

XMLデータのインポートとエクスポート

例1：成績データのXMLエクスポート

```
-- 成績データをXML形式でエクスポート用に整形
SELECT CONCAT(
    '<?xml version="1.0" encoding="UTF-8"?>',
    '<grade_export>',
    '<export_date>', CURDATE(), '</export_date>',
    '<records>',
    GROUP_CONCAT(
        '<record>',
        '<student_id>', student_id, '</student_id>',
```

```
        '<course_id>', course_id, '</course_id>',
        '<term>', academic_term, '</term>',
        '<grade>', ExtractValue(grade_data,
'/grade_record/final_grade/letter_grade'), '</grade>',
        '</record>'
        SEPARATOR ''
    ),
    '</records>',
    '</grade_export>'
) AS xml_export
FROM detailed_grades;
```

例2：システム設定のXML管理

```
-- システム設定をXML形式で挿入
INSERT INTO system_configurations (config_id, config_name, config_data) VALUES
('CFG001', 'grade_calculation_settings', '<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <grade_scale>
        <grade letter="A" min="90" max="100" gpa="4.0"/>
        <grade letter="B" min="80" max="89" gpa="3.0"/>
        <grade letter="C" min="70" max="79" gpa="2.0"/>
        <grade letter="D" min="60" max="69" gpa="1.0"/>
        <grade letter="F" min="0" max="59" gpa="0.0"/>
    </grade_scale>
    <calculation_rules>
        <rule type="rounding">round_half_up</rule>
        <rule type="decimal_places">1</rule>
        <rule type="minimum_score">0</rule>
    </calculation_rules>
    <notification_settings>
        <notify_students>true</notify_students>
        <notify_parents>false</notify_parents>
        <email_template>grade_notification</email_template>
    </notification_settings>
</configuration>');
```

実践例：学校データベースでのXML活用

例1：統合成績管理システム

```
-- 成績データとシラバス情報を統合した詳細レポート
SELECT
    g.record_id,
    ExtractValue(g.grade_data, '/grade_record/student_info/name') AS 学生名,
    ExtractValue(s.syllabus_data, '/syllabus/course_info/title') AS 講座名,
    ExtractValue(s.syllabus_data, '/syllabus/course_info/credits') AS 単位数,
    ExtractValue(g.grade_data, '/grade_record/final_grade/letter_grade') AS 最終評
価,
```

```
    ExtractValue(g.grade_data, '/grade_record/final_grade/gpa_points') AS GPA,
    ExtractValue(s.syllabus_data,
'/syllabus/evaluation/method[@type="exam"]/@weight') AS 試験重み
FROM detailed_grades g
JOIN course_syllabi s ON g.course_id = s.course_id
WHERE ExtractValue(g.grade_data, '/grade_record/final_grade/letter_grade') IN
('A', 'B');
```

例2：学習進度の追跡

```
-- 週次の学習トピックと関連する評価を照合
SELECT
    ExtractValue(s.syllabus_data, '/syllabus/course_info/title') AS 講座名,
    ExtractValue(s.syllabus_data, '/syllabus/schedule/weeks/week[1]/topic') AS 第1
週トピック,
    ExtractValue(s.syllabus_data, '/syllabus/schedule/weeks/week[2]/topic') AS 第2
週トピック,
    COUNT(g.record_id) AS 評価済み学生数,
    AVG(CAST(ExtractValue(g.grade_data,
'/grade_record/final_grade/numerical_grade') AS DECIMAL(5,2))) AS 平均点
FROM course_syllabi s
LEFT JOIN detailed_grades g ON s.course_id = g.course_id
GROUP BY s.syllabus_id, 講座名, 第1週トピック, 第2週トピック;
```

練習問題

問題9-6-1

course_syllabi テーブルから、すべてのシラバスの講座名と担当教師名をXMLから抽出して表示するSQLを書いてください。

問題9-6-2

detailed_grades テーブルから、最終評価が「A」の学生の名前と数値評価をXMLから抽出するSQLを書いてください。

問題9-6-3

course_syllabi テーブルで、講座ID「1」のシラバスの講座説明を「ITと情報処理の基礎を学ぶ総合的な講座です」に更新するSQLを書いてください。

問題9-6-4

detailed_grades テーブルから、中間試験（midterm）の得点が80点以上の学生を検索するSQLを書いてください。

問題9-6-5

course_syllabi テーブルから、評価方法で「exam」タイプの重みが50以上の講座を検索するSQLを書いてください。

問題9-6-6

detailed_grades テーブルから、各講座の平均GPA値を計算するSQLを書いてください。

問題9-6-7

course_syllabi テーブルから、第1週の学習トピックと第1番目の学習目標を抽出するSQLを書いてください。

問題9-6-8

detailed_grades テーブルから、実習課題（practical）の評価がある学生の名前と得点を抽出するSQLを書いてください。

問題9-6-9

course_syllabi テーブルから、単位数が3単位の講座の総週数を抽出するSQLを書いてください。

問題9-6-10

detailed_grades と course_syllabi を結合して、A評価を取った学生の名前、講座名、担当教師を表示するSQLを書いてください。

解答と詳細な解説

解答9-6-1

```
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/course_info/title') AS 講座名,
       ExtractValue(syllabus_data, '/syllabus/course_info/instructor') AS 担当教師
FROM course_syllabi;
```

解答9-6-2

```
SELECT ExtractValue(grade_data, '/grade_record/student_info/name') AS 学生名,
       ExtractValue(grade_data, '/grade_record/final_grade/numerical_grade') AS 数値評価
FROM detailed_grades
WHERE ExtractValue(grade_data, '/grade_record/final_grade/letter_grade') = 'A';
```

解説：

- WHERE句でXMLの内容による条件指定
- `letter_grade`要素の値が「A」の記録のみを抽出
- 複数の要素を同時に抽出してSELECT句で表示

解答9-6-3

```
UPDATE course_syllabi
SET syllabus_data = UpdateXML(
    syllabus_data,
    '/syllabus/course_info/description',
    'ITと情報処理の基礎を学ぶ総合的な講座です'
)
WHERE syllabus_id = 'SYL001';
```

解説：

- `UpdateXML()`関数でXML要素の値を更新
- 第1引数：更新対象のXML文書
- 第2引数：更新する要素のXPath
- 第3引数：新しい値
- WHERE句で特定のレコードのみを対象に更新

解答9-6-4

```
SELECT ExtractValue(grade_data, '/grade_record/student_info/name') AS 学生名,
       ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/score') AS 中間試験得点
FROM detailed_grades
WHERE CAST(ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="midterm"]/score') AS UNSIGNED) >= 80;
```

解説：

- XPath式で属性による条件指定：`[@type="midterm"]`
- `CAST()`関数でXMLから抽出した文字列を数値に変換
- 数値比較のための型変換が重要
- 属性フィルタリングによる特定の評価タイプの抽出

解答9-6-5

```
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/course_info/title') AS 講座名,
       ExtractValue(syllabus_data,
'/syllabus/evaluation/method[@type="exam"]/@weight') AS 試験重み
FROM course_syllabi
WHERE CAST(ExtractValue(syllabus_data,
'/syllabus/evaluation/method[@type="exam"]/@weight') AS UNSIGNED) >= 50;
```

解説：

- @weightで属性値を抽出
- 属性と要素内容の両方を条件に使用
- [@type="exam"]で特定タイプの評価方法をフィルタリング
- 属性値の数値比較のためのCAST変換

解答9-6-6

```
SELECT ExtractValue(grade_data, '/grade_record/course_info/title') AS 講座名,
       COUNT(*) AS 学生数,
       AVG(CAST(ExtractValue(grade_data, '/grade_record/final_grade/gpa_points')
AS DECIMAL(3,2))) AS 平均GPA
FROM detailed_grades
GROUP BY ExtractValue(grade_data, '/grade_record/course_info/title');
```

解説：

- GROUP BYでXML要素の値によるグループ化
- AVG()関数と組み合わせて平均値を計算
- DECIMAL型への変換で精密な数値計算
- XMLデータを使った統計分析の例

解答9-6-7

```
SELECT syllabus_id,
       ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[@number="1"]/topic') AS 第1週トピック,
       ExtractValue(syllabus_data,
'/syllabus/schedule/weeks/week[@number="1"]/objectives/objective[1]') AS 第1学習目
標
FROM course_syllabi;
```

解説：

- 属性による要素の特定：[@number="1"]
- 配列インデックスによる要素選択：[1]
- 深い階層構造でのXPath記法
- 属性値とインデックスを組み合わせた高度な要素抽出

解答9-6-8

```
SELECT ExtractValue(grade_data, '/grade_record/student_info/name') AS 学生名,
       ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="practical"]/score') AS 実習得点,
       ExtractValue(grade_data,
'/grade_record/assessments/assessment[@type="practical"]/comments') AS コメント
FROM detailed_grades
WHERE ExtractValue(grade_data,
```

```
  '/grade_record/assessments/assessment[@type="practical"]/score') IS NOT NULL
    AND ExtractValue(grade_data,
  '/grade_record/assessments/assessment[@type="practical"]/score') != '';
```

解説：

- 複合条件でNULL値と空文字の両方をチェック
- XMLの存在チェックでデータの完全性を確認
- 特定の評価タイプの存在確認と値の抽出
- XMLデータの品質チェックを含む検索

解答9-6-9

```
SELECT syllabus_id,
       ExtractValue(syllabus_data, '/syllabus/course_info/title') AS 講座名,
       ExtractValue(syllabus_data, '/syllabus/schedule/weeks/@total') AS 総週数
FROM course_syllabi
WHERE ExtractValue(syllabus_data, '/syllabus/course_info/credits') = '3';
```

解説：

- 属性値の抽出：`/@total`
- WHERE句でXML要素による条件指定
- 数値条件でも文字列として比較（XMLでは文字列として格納）
- 条件付きの属性値抽出

解答9-6-10

```
SELECT ExtractValue(g.grade_data, '/grade_record/student_info/name') AS 学生名,
       ExtractValue(s.syllabus_data, '/syllabus/course_info/title') AS 講座名,
       ExtractValue(s.syllabus_data, '/syllabus/course_info/instructor') AS 担当教
師
FROM detailed_grades g
JOIN course_syllabi s ON g.course_id = s.course_id
WHERE ExtractValue(g.grade_data, '/grade_record/final_grade/letter_grade') = 'A';
```

解説：

- JOINとXML抽出の組み合わせ
- 複数テーブルのXMLデータを統合した検索
- 関連データの結合によるより詳細な情報の取得
- 実践的な統合クエリの例

まとめ

この章では、MySQLにおけるXMLデータの格納と操作について学びました：

1. **XMLの基本概念**：階層構造、要素、属性、XPath記法の理解
2. **XMLデータの格納**：XMLデータ型を使用したテーブル設計
3. **ExtractValue()関数**：XPath式を使用したXML要素の抽出
4. **UpdateXML()関数**：XML文書内の要素値の更新
5. **XPath式の活用**：属性フィルタリング、インデックス指定、階層ナビゲーション
6. **複雑なクエリ**：XMLデータを使った統計分析、条件検索、テーブル結合
7. **実践的な活用**：シラバス管理、成績記録、システム設定での具体的な使用例

XMLデータ型は、階層構造を持つ複雑なデータを効率的に格納し、柔軟に操作できる強力な機能です。特に教育分野では、シラバス、成績記録、学習進度など、構造化された情報を扱うことが多いため、XMLの活用により、より柔軟で拡張性の高いシステムを構築できます。

ただし、XMLデータの操作は通常のリレーショナルデータに比べて複雑になる場合があるため、適切な設計とパフォーマンスの考慮が重要です。また、XPath式の理解と適切な活用により、XMLデータの真価を発揮できます。

次のセクションでは、「大きなオブジェクト（BLOB/CLOB）：バイナリデータの格納」について学び、画像、動画、文書ファイルなどのバイナリデータの効率的な管理方法を習得します。

- `ExtractValue()`関数でXML文書から特定の要素を抽出
- XPath式`/syllabus/course_info/title`で階層構造をたどって目的の要素を指定
- ルート要素から順番にパスを指定する絶対パス記法を使用

9-7. 大きなオブジェクト（BLOB/CLOB）：バイナリデータの格納

はじめに

現代の情報システムでは、テキストや数値データだけでなく、画像、動画、音声、文書ファイルなどの大きなバイナリデータを扱うことが一般的になっています。

学校データベースでは、以下のような場面で大きなオブジェクトデータが必要になります：

- **学生証明写真**：入学時や在学証明書用の写真データ
- **教材ファイル**：PDF資料、PowerPointスライド、動画教材
- **課題提出**：学生が提出するレポートファイル、プログラムファイル
- **授業録画**：オンライン授業の動画ファイル
- **音声データ**：語学学習用の音声ファイル、講義録音
- **スキャンデータ**：手書きノートや試験答案のスキャン画像
- **システムログ**：大容量のログファイルやデバッグ情報

MySQLでは、このような大きなデータを効率的に格納するために、BLOB（Binary Large Object）とTEXT（Character Large Object）という専用のデータ型が提供されています。この章では、これらのデータ型の特徴と使い方を学びます。

> **用語解説**：

> - **BLOB（Binary Large Object）**：画像、動画、音声などのバイナリデータを格納するためのデータ型です。
> - **CLOB（Character Large Object）**：大量のテキストデータを格納するためのデータ型で、MySQLではTEXT型がこれに相当します。
> - **バイナリデータ**：文字として解釈されない生のデータのことで、画像ファイルや実行ファイルなどが該当します。

BLOBとTEXTデータ型の種類

MySQLでは、データのサイズに応じて複数のBLOB/TEXTデータ型が用意されています。

BLOBデータ型（バイナリデータ用）

データ型	最大サイズ	用途例
TINYBLOB	255バイト	小さなアイコン、短いバイナリデータ
BLOB	65KB (64KB)	一般的な画像ファイル、小さな文書
MEDIUMBLOB	16MB	高解像度画像、短い動画、音声ファイル
LONGBLOB	4GB	長時間動画、大きな文書ファイル

TEXTデータ型（テキストデータ用）

データ型	最大サイズ	用途例
TINYTEXT	255文字	短いコメント、概要
TEXT	65KB	一般的な文書、レポート
MEDIUMTEXT	16MB	長い文書、書籍テキスト
LONGTEXT	4GB	大量のログデータ、全文データ

> **用語解説**：
>
> - **KB（キロバイト）**：約1,000バイトの単位で、1KB = 1,024バイトです。
> - **MB（メガバイト）**：約100万バイトの単位で、1MB = 1,024KBです。
> - **GB（ギガバイト）**：約10億バイトの単位で、1GB = 1,024MBです。

大きなオブジェクト用テーブルの作成

学校データベースで大きなオブジェクトを活用するためのテーブルを作成しましょう。

学生写真テーブルの作成

```
-- 学生の証明写真を保存するテーブル
CREATE TABLE student_photos (
    photo_id VARCHAR(16) PRIMARY KEY,
    student_id BIGINT NOT NULL,
    photo_type VARCHAR(20), -- 'profile', 'id_card', 'graduation'
```

```
    image_data MEDIUMBLOB,
    image_format VARCHAR(10), -- 'JPEG', 'PNG', 'GIF'
    image_size INT, -- バイト数
    image_width INT, -- ピクセル幅
    image_height INT, -- ピクセル高さ
    upload_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    notes TEXT,

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    INDEX idx_student_type (student_id, photo_type),
    INDEX idx_upload_date (upload_date)
);
```

教材ファイルテーブルの作成

```
-- 教材ファイルを保存するテーブル
CREATE TABLE course_files (
    file_id VARCHAR(16) PRIMARY KEY,
    course_id VARCHAR(16),
    teacher_id BIGINT,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50), -- 'pdf', 'pptx', 'docx', 'mp4', 'mp3'
    file_data LONGBLOB,
    file_size BIGINT, -- バイト数
    mime_type VARCHAR(100), -- 'application/pdf', 'video/mp4', など
    description TEXT,
    upload_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_accessed TIMESTAMP,
    download_count INT DEFAULT 0,
    is_public BOOLEAN DEFAULT FALSE,

    FOREIGN KEY (course_id) REFERENCES courses(course_id),
    FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id),
    INDEX idx_course_type (course_id, file_type),
    INDEX idx_public_files (is_public, file_type),
    INDEX idx_upload_date (upload_date)
);
```

学生提出物テーブルの作成

```
-- 学生の課題提出ファイルを保存するテーブル
CREATE TABLE student_submissions (
    submission_id VARCHAR(16) PRIMARY KEY,
    student_id BIGINT NOT NULL,
    course_id VARCHAR(16) NOT NULL,
    assignment_name VARCHAR(200),
    submitted_file LONGBLOB,
    file_name VARCHAR(255),
    file_type VARCHAR(50),
```

```
    file_size BIGINT,
    submission_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    submission_text LONGTEXT, -- テキスト形式の提出内容
    teacher_comments MEDIUMTEXT, -- 教師のフィードバック
    grade_score DECIMAL(5,2),

    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id),
    INDEX idx_student_course (student_id, course_id),
    INDEX idx_submission_date (submission_date),
    INDEX idx_assignment (assignment_name)
);
```

システムログテーブルの作成

```
-- システムの詳細ログを保存するテーブル
CREATE TABLE system_logs (
    log_id BIGINT AUTO_INCREMENT PRIMARY KEY,
    log_level VARCHAR(20), -- 'INFO', 'WARNING', 'ERROR', 'DEBUG'
    log_source VARCHAR(100), -- ログの出力元
    log_message TEXT,
    detailed_log LONGTEXT, -- 詳細なログ情報
    log_data MEDIUMBLOB, -- バイナリ形式のログデータ
    user_id BIGINT,
    session_id VARCHAR(64),
    ip_address VARCHAR(45),
    user_agent TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    INDEX idx_level_date (log_level, created_at),
    INDEX idx_source_date (log_source, created_at),
    INDEX idx_user_date (user_id, created_at)
);
```

BLOBデータの挿入と基本操作

バイナリデータの挿入

バイナリデータをBLOBカラムに挿入する方法は複数あります。

例1：HEX形式でのデータ挿入

```
-- 小さな画像データ（サンプル）をHEX形式で挿入
INSERT INTO student_photos (
    photo_id, student_id, photo_type, image_data,
    image_format, image_size, image_width, image_height, notes
) VALUES (
    'PHT001', 301, 'profile',
```

```
UNHEX('FFD8FFE000104A4649460001010100600060000FFDB00430008060607060508070707090908
0A0C140D0C0B0B0C1912130F141D1A1F1E1D1A1C1C20242E2720222C231C1C2837292C30313434341
F27393D38323C2E333432FFDB0043010909090C0B0C180D0D1832211C213232323232323232323232323
232323232323232323232323232323232323232323232323232323232323232323232323232FFC00
011080001000103012200021101031101FFC4001F000001050101010101010000000000000000010203
040506070809 0A0BFFC400B5100002010303020403050504040000017D01020300041105122131410
613516107227114328191A1082342B1C11552D1F02433627282090A161718191A25262728292A34353
63738393A434445464748494A535455565758595A636465666768696A737475767778797A838485868
788898A92939495969798999AA2A3A4A5A6A7A8A9AAB2B3B4B5B6B7B8B9BAC2C3C4C5C6C7C8C9CAD2D
3D4D5D6D7D8D9DAE1E2E3E4E5E6E7E8E9EAF1F2F3F4F5F6F7F8F9FAFFC4001F0100030101010101010
101010000000000000102030405060708090A0BFFC400B5110002010204040304070504040001027700
0010203110405213106124151076171132281913214227381A1B14232C152D1F016242533430A17181
9E125356491A2643443456537283726F1C120E1F25564374738485946 5A6B6C6D6E6F7485969748595
A65A666768696A737475767778797A828384858687888 98A92939495969798999AA2A3A4A5A6A7A8A9
AAB2B3B4B5B6B7B8B9BAC2C3C4C5C6C7C8C9CAD2D3D4D5D6D7D8D9DAE2E3E4E5E6E7E8E9EAF2F3F4F5
F6F7F8F9FAFFDA000C03010002110311003F00'),
    'JPEG', 2048, 100, 100, '入学時に撮影された証明写真'
);
```

例2：LOAD_FILE()関数での外部ファイル読み込み

```
-- 外部ファイルからデータを読み込んで挿入
INSERT INTO course_files (
    file_id, course_id, teacher_id, file_name,
    file_type, file_data, file_size, mime_type, description
) VALUES (
    'FILE001', '1', 101, 'IT基礎知識_第1章.pdf',
    'pdf', LOAD_FILE('/path/to/textbook_chapter1.pdf'),
    (SELECT OCTET_LENGTH(LOAD_FILE('/path/to/textbook_chapter1.pdf'))),
    'application/pdf', 'IT基礎知識の第1章テキスト'
);
```

> **注意**：LOAD_FILE()関数を使用するには、MySQLサーバーに適切なファイルアクセス権限が必要です。また、セキュリティ上の理由から、多くの本番環境では制限されています。

大きなテキストデータの挿入

```
-- 長いテキストデータの挿入
INSERT INTO student_submissions (
    submission_id, student_id, course_id, assignment_name,
    file_name, file_type, submission_text, teacher_comments
) VALUES (
    'SUB001', 301, '1', 'ITの社会的影響レポート',
    'report_kurosawa.txt', 'text',
    'ITの社会的影響について

現代社会において、情報技術（IT）は私たちの生活のあらゆる面に深く浸透し、社会構造そのものを変革しています。本レポートでは、ITが社会に与える多面的な影響について詳細に分析し、その光と影の両面を考察します。
```

```
第1章：デジタル変革の概要
ITの普及により、従来のアナログベースの社会システムが急速にデジタル化されています。この変化は、効率性の向上、情報アクセスの民主化、新たなビジネスモデルの創出など、多くの利益をもたらしています。一方で、デジタルデバイドの拡大、プライバシーの侵害、サイバーセキュリティの脅威など、新たな課題も生み出しています。

第2章：経済への影響
IT産業の成長は、世界経済の重要な牽引力となっています。電子商取引の拡大により、従来の小売業界が大きく変化し、新しい雇用機会が創出される一方で、既存の職業が自動化により脅威にさらされています。また、フィンテック革命により、金融サービスの在り方も根本的に変化しています。

第3章：教育への影響
オンライン教育プラットフォームの普及により、教育へのアクセスが大幅に向上しました。地理的制約を超えて質の高い教育を受けることが可能になり、生涯学習の概念も変化しています。しかし、対面教育の重要性や、デジタルリテラシーの格差など、解決すべき課題も多く存在します。

第4章：社会インフラへの影響
スマートシティの概念により、都市インフラの効率性と持続可能性が向上しています。IoT技術の活用により、交通システム、エネルギー管理、廃棄物処理などが最適化されています。しかし、システムの複雑化により、障害時の影響範囲も拡大しています。

第5章：コミュニケーションへの影響
ソーシャルメディアの普及により、人々のコミュニケーション方法が大きく変化しました。距離を超えたつながりが可能になった一方で、フェイクニュースの拡散、エコーチェンバー効果、サイバーいじめなどの問題も深刻化しています。

結論：
ITの社会的影響は多岐にわたり、その恩恵を最大化しながら負の側面を最小化するためには、技術開発者、政策立案者、市民が協力して取り組む必要があります。デジタルリテラシーの向上、適切な規制の整備、倫理的なガイドラインの策定が急務です。

今後のIT発展においては、単なる技術的進歩だけでなく、社会全体の福祉と持続可能性を考慮したアプローチが求められます。私たち一人一人が、ITの恩恵を享受しながらも、その責任ある使用について考え続けることが重要です。',
    '内容が充実しており、多角的な視点からITの影響を分析できています。特に具体例を用いた説明が効果的です。今後は、より最新の技術動向（AI、ブロックチェーンなど）についても触れると、さらに良いレポートになるでしょう。'
);
```

BLOBデータの検索と取得

データサイズによる検索

```sql
-- ファイルサイズが1MB以上の教材を検索
SELECT file_id, file_name, file_type,
       ROUND(file_size / 1024 / 1024, 2) AS サイズMB,
       upload_date
FROM course_files
WHERE file_size > 1024 * 1024
ORDER BY file_size DESC;
```

メタデータによる検索

```
-- 特定の形式の画像ファイルを検索
SELECT photo_id, student_id, photo_type,
       image_format, image_width, image_height,
       ROUND(image_size / 1024, 2) AS サイズKB
FROM student_photos
WHERE image_format = 'JPEG'
  AND image_width >= 200
  AND image_height >= 200;
```

データの部分取得

```
-- BLOBデータの最初の100バイトのみを取得
SELECT file_id, file_name,
       LEFT(file_data, 100) AS ファイルヘッダー,
       HEX(LEFT(file_data, 20)) AS ヘッダーHEX
FROM course_files
WHERE file_type = 'pdf'
LIMIT 3;
```

BLOBデータの操作関数

LENGTH()とOCTET_LENGTH()関数

```
-- データサイズの取得
SELECT file_id, file_name,
       LENGTH(file_data) AS データ長,
       OCTET_LENGTH(file_data) AS バイト数,
       file_size AS 記録サイズ
FROM course_files
WHERE file_data IS NOT NULL
LIMIT 5;
```

HEX()とUNHEX()関数

```
-- バイナリデータを16進数文字列に変換
SELECT photo_id,
       HEX(LEFT(image_data, 10)) AS 画像ヘッダーHEX,
       image_format
FROM student_photos
WHERE image_data IS NOT NULL;
```

SUBSTRING()関数でのバイナリデータ操作

```
-- バイナリデータの特定位置からの抽出
SELECT file_id, file_name,
       HEX(SUBSTRING(file_data, 1, 4)) AS ファイルシグネチャ,
       CASE
         WHEN HEX(SUBSTRING(file_data, 1, 4)) = '25504446' THEN 'PDF'
         WHEN HEX(SUBSTRING(file_data, 1, 2)) = 'FFD8' THEN 'JPEG'
         WHEN HEX(SUBSTRING(file_data, 1, 8)) = '89504E470D0A1A0A' THEN 'PNG'
         ELSE 'Unknown'
       END AS ファイル種別判定
FROM course_files
WHERE file_data IS NOT NULL;
```

パフォーマンスとストレージの考慮事項

インデックス戦略

```
-- BLOBカラムを除いたインデックス作成
CREATE INDEX idx_file_metadata ON course_files (file_type, file_size,
upload_date);
CREATE INDEX idx_photo_metadata ON student_photos (student_id, photo_type,
image_format);

-- 部分インデックスの作成（TEXT型の場合）
CREATE INDEX idx_submission_text_partial ON student_submissions
(submission_text(100));
```

ストレージサイズの監視

```
-- テーブルごとのストレージ使用量を確認
SELECT
    table_name,
    ROUND(((data_length + index_length) / 1024 / 1024), 2) AS テーブルサイズMB,
    ROUND((data_length / 1024 / 1024), 2) AS データサイズMB,
    ROUND((index_length / 1024 / 1024), 2) AS インデックスサイズMB,
    table_rows AS 行数
FROM information_schema.tables
WHERE table_schema = DATABASE()
  AND table_name IN ('student_photos', 'course_files', 'student_submissions')
ORDER BY (data_length + index_length) DESC;
```

データ圧縮と最適化

```
-- 大きなデータの圧縮状況確認
SELECT file_id, file_name, file_type,
       file_size AS 元サイズ,
```

```
        LENGTH(file_data) AS 保存サイズ,
        ROUND((file_size - LENGTH(file_data)) / file_size * 100, 2) AS 圧縮率パーセン
ト
FROM course_files
WHERE file_size > 0 AND file_data IS NOT NULL;
```

実践例：ファイル管理システム

例1：ファイルアップロード管理

```
-- ファイルアップロード処理のシミュレーション
DELIMITER $$

CREATE PROCEDURE UploadCourseFile(
    IN p_file_id VARCHAR(16),
    IN p_course_id VARCHAR(16),
    IN p_teacher_id BIGINT,
    IN p_file_name VARCHAR(255),
    IN p_file_type VARCHAR(50),
    IN p_file_data LONGBLOB,
    IN p_mime_type VARCHAR(100),
    IN p_description TEXT
)
BEGIN
    DECLARE file_size_bytes BIGINT;

    -- ファイルサイズを計算
    SET file_size_bytes = IFNULL(LENGTH(p_file_data), 0);

    -- ファイル情報を挿入
    INSERT INTO course_files (
        file_id, course_id, teacher_id, file_name, file_type,
        file_data, file_size, mime_type, description,
        upload_date
    ) VALUES (
        p_file_id, p_course_id, p_teacher_id, p_file_name, p_file_type,
        p_file_data, file_size_bytes, p_mime_type, p_description,
        NOW()
    );

    -- アップロード成功のログ記録（簡略化）
    INSERT INTO system_logs (log_level, log_source, log_message, user_id)
    VALUES ('INFO', 'FILE_UPLOAD',
            CONCAT('File uploaded: ', p_file_name, ' (', file_size_bytes, '
bytes)'),
            p_teacher_id);

END$$

DELIMITER ;
```

例2：ファイルダウンロード追跡

```
-- ファイルダウンロード追跡の処理
CREATE VIEW popular_files AS
SELECT cf.file_id,
       cf.file_name,
       cf.file_type,
       c.course_name,
       t.teacher_name,
       cf.download_count,
       ROUND(cf.file_size / 1024 / 1024, 2) AS サイズMB,
       cf.upload_date,
       cf.last_accessed
FROM course_files cf
JOIN courses c ON cf.course_id = c.course_id
JOIN teachers t ON cf.teacher_id = t.teacher_id
WHERE cf.is_public = TRUE
ORDER BY cf.download_count DESC, cf.upload_date DESC;
```

例3：ストレージクリーンアップ

```
-- 古くて使用されていないファイルの特定
SELECT file_id, file_name, file_type,
       ROUND(file_size / 1024 / 1024, 2) AS サイズMB,
       upload_date,
       IFNULL(last_accessed, upload_date) AS 最終アクセス,
       DATEDIFF(NOW(), IFNULL(last_accessed, upload_date)) AS 未使用日数
FROM course_files
WHERE DATEDIFF(NOW(), IFNULL(last_accessed, upload_date)) > 365
  AND download_count = 0
ORDER BY file_size DESC;
```

セキュリティとアクセス制御

ファイルアクセス権限の管理

```
-- ファイルアクセス権限テーブル
CREATE TABLE file_permissions (
    permission_id VARCHAR(16) PRIMARY KEY,
    file_id VARCHAR(16),
    user_type VARCHAR(20), -- 'student', 'teacher', 'admin'
    user_id BIGINT,
    permission_level VARCHAR(20), -- 'read', 'write', 'admin'
    granted_by BIGINT,
    granted_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_date TIMESTAMP NULL,
```

```
    FOREIGN KEY (file_id) REFERENCES course_files(file_id),
    INDEX idx_file_user (file_id, user_type, user_id),
    INDEX idx_expiry (expires_date)
);
```

安全なファイル取得

```
-- 権限チェック付きファイル取得
SELECT cf.file_id, cf.file_name, cf.mime_type,
       cf.file_data
FROM course_files cf
JOIN file_permissions fp ON cf.file_id = fp.file_id
WHERE cf.file_id = 'FILE001'
  AND fp.user_id = 301
  AND fp.permission_level IN ('read', 'admin')
  AND (fp.expires_date IS NULL OR fp.expires_date > NOW());
```

練習問題

問題9-7-1

student_photos テーブルから、すべての学生の写真情報（写真ID、学生ID、写真タイプ、画像形式、ファイルサイズ）を取得するSQLを書いてください。

問題9-7-2

course_files テーブルから、ファイルサイズが500KB以上のPDFファイルを検索し、ファイル名、サイズ（MB単位）、アップロード日を表示するSQLを書いてください。

問題9-7-3

student_submissions テーブルから、提出テキストの文字数が1000文字以上のレポートを検索し、学生ID、課題名、文字数を表示するSQLを書いてください。

問題9-7-4

student_photos テーブルから、各学生の写真枚数を集計するSQLを書いてください。

問題9-7-5

course_files テーブルから、各講座の教材ファイル数と総ファイルサイズ（MB単位）を計算するSQLを書いてください。

問題9-7-6

system_logs テーブルから、エラーレベルのログで詳細ログ（detailed_log）がNULLでないレコードを検索するSQLを書いてください。

問題9-7-7

student_submissions テーブルから、ファイル提出がない（submitted_file が NULL）学生のレポートを検索し、テキスト提出の文字数も表示するSQLを書いてください。

問題9-7-8

course_files テーブルから、各ファイルタイプ別の平均ファイルサイズと最大ファイルサイズを計算するSQLを書いてください。

問題9-7-9

student_photos テーブルから、画像の幅と高さが同じ（正方形）の写真を検索するSQLを書いてください。

問題9-7-10

course_files テーブルから、最近30日以内にアップロードされたファイルの中で、ダウンロード数が0回のファイルを検索し、ファイル名、サイズ、アップロード日を表示するSQLを書いてください。

解答と詳細な解説

解答9-7-1

```
SELECT photo_id,
       student_id,
       photo_type,
       image_format,
       ROUND(image_size / 1024, 2) AS ファイルサイズKB
FROM student_photos
ORDER BY student_id, photo_type;
```

解説：

- ROUND()関数でバイト単位のサイズをKB単位に変換
- 1024で割ることでバイトをキロバイトに変換
- ORDER BYで学生IDと写真タイプ順に並び替え

解答9-7-2

```
SELECT file_name,
       ROUND(file_size / 1024 / 1024, 2) AS サイズMB,
       upload_date
FROM course_files
WHERE file_type = 'pdf'
  AND file_size >= 500 * 1024
ORDER BY file_size DESC;
```

解説：

- WHERE句で複数条件を指定（ファイルタイプとサイズ）

- 500 * 1024で500KBをバイト単位に変換
- ファイルサイズの大きい順に並び替え

解答9-7-3

```
SELECT student_id,
       assignment_name,
       CHAR_LENGTH(submission_text) AS 文字数
FROM student_submissions
WHERE CHAR_LENGTH(submission_text) >= 1000
ORDER BY 文字数 DESC;
```

解説：

- CHAR_LENGTH()関数でテキストの文字数を取得
- WHERE句で1000文字以上の条件を指定
- 文字数の多い順に並び替えて表示

解答9-7-4

```
SELECT student_id,
       COUNT(*) AS 写真枚数
FROM student_photos
GROUP BY student_id
ORDER BY 写真枚数 DESC;
```

解説：

- GROUP BYで学生IDごとにグループ化
- COUNT(*)で各グループの行数（写真枚数）を集計
- 写真枚数の多い順に並び替え

解答9-7-5

```
SELECT course_id,
       COUNT(*) AS ファイル数,
       ROUND(SUM(file_size) / 1024 / 1024, 2) AS 総サイズMB
FROM course_files
WHERE file_data IS NOT NULL
GROUP BY course_id
ORDER BY 総サイズMB DESC;
```

解説：

- GROUP BYで講座IDごとにグループ化
- COUNT(*)でファイル数、SUM(file_size)で総サイズを計算

- WHERE句でNULLデータを除外
- MB単位に変換して表示

解答9-7-6

```
SELECT log_id,
       log_source,
       log_message,
       created_at,
       CHAR_LENGTH(detailed_log) AS 詳細ログ文字数
FROM system_logs
WHERE log_level = 'ERROR'
  AND detailed_log IS NOT NULL
ORDER BY created_at DESC;
```

解説：

- WHERE句で複数条件（エラーレベルかつ詳細ログが存在）
- IS NOT NULLでNULL値を除外
- CHAR_LENGTH()で詳細ログの文字数も表示
- 作成日時の新しい順に並び替え

解答9-7-7

```
SELECT student_id,
       assignment_name,
       CHAR_LENGTH(submission_text) AS テキスト文字数,
       submission_date
FROM student_submissions
WHERE submitted_file IS NULL
  AND submission_text IS NOT NULL
ORDER BY テキスト文字数 DESC;
```

解説：

- WHERE句でファイル提出がなく、テキスト提出がある条件
- IS NULLとIS NOT NULLを組み合わせた条件指定
- テキストのみで提出された課題を特定

解答9-7-8

```
SELECT file_type,
       COUNT(*) AS ファイル数,
       ROUND(AVG(file_size) / 1024 / 1024, 2) AS 平均サイズMB,
       ROUND(MAX(file_size) / 1024 / 1024, 2) AS 最大サイズMB
FROM course_files
WHERE file_size > 0
```

```
GROUP BY file_type
ORDER BY 平均サイズMB DESC;
```

解説：

- GROUP BYでファイルタイプごとに集計
- AVG()で平均、MAX()で最大値を計算
- WHERE句でサイズが0より大きいファイルのみを対象
- ファイルタイプ別の統計情報を取得

解答9-7-9

```
SELECT photo_id,
       student_id,
       photo_type,
       image_width,
       image_height,
       image_format
FROM student_photos
WHERE image_width = image_height
  AND image_width IS NOT NULL
  AND image_height IS NOT NULL
ORDER BY image_width DESC;
```

解説：

- WHERE句で幅と高さが等しい条件
- IS NOT NULLでNULL値を除外
- 正方形の画像を特定する条件
- 画像サイズの大きい順に並び替え

解答9-7-10

```
SELECT file_name,
       ROUND(file_size / 1024 / 1024, 2) AS サイズMB,
       upload_date,
       download_count
FROM course_files
WHERE upload_date >= DATE_SUB(NOW(), INTERVAL 30 DAY)
  AND download_count = 0
ORDER BY upload_date DESC;
```

解説：

- DATE_SUB()関数で30日前の日付を計算
- WHERE句で期間条件とダウンロード数条件を組み合わせ
- アップロードされたが使用されていないファイルを特定

- 新しいアップロード順に並び替え

まとめ

この章では、MySQLにおける大きなオブジェクト（BLOB/CLOB）データの格納と操作について学びました：

1. **BLOBとTEXTデータ型の理解**：各サイズ別データ型の特徴と用途
2. **大きなオブジェクト用テーブル設計**：写真、ファイル、提出物、ログデータの効率的な格納
3. **バイナリデータの挿入**：HEX形式、LOAD_FILE()関数の使用方法
4. **BLOBデータの検索**：メタデータによる検索、サイズ条件での絞り込み
5. **データ操作関数**：LENGTH()、HEX()、SUBSTRING()などの専用関数
6. **パフォーマンス考慮事項**：インデックス戦略、ストレージ監視、データ圧縮
7. **実践的な活用**：ファイル管理システム、アクセス制御、セキュリティ対策

大きなオブジェクトデータの管理は、現代の情報システムにおいて重要な要素です。特に学校データベースのような教育コンテンツを多く扱うシステムでは、画像、動画、文書ファイルなどの効率的な管理が学習体験の質に直結します。

ただし、BLOBデータは通常のリレーショナルデータと比べてストレージ容量やパフォーマンスに大きな影響を与えるため、適切な設計と運用が重要です。特に以下の点に注意が必要です：

- **ストレージ容量の計画**：大容量データの増加を見込んだ容量設計
- **バックアップ戦略**：大きなファイルを含むバックアップの時間とストレージ考慮
- **ネットワーク負荷**：大きなデータの転送によるネットワーク負荷
- **セキュリティ**：機密性の高いファイルのアクセス制御

これらの考慮事項を適切に管理することで、BLOBデータの利点を最大限に活用できます。

第9章全体のまとめ

第9章「特殊なデータ型と操作」では、以下の7つのセクションを通じて、現代のデータベースシステムで重要な特殊データ型について学習しました：

1. **日付と時刻**：時間データの効率的な管理と分析
2. **文字列操作**：テキストデータの加工と検索最適化
3. **JSON**：構造化データの柔軟な格納と操作
4. **地理空間データ**：位置情報の管理と空間分析
5. **全文検索**：大量テキストからの高速情報検索
6. **XML**：階層構造データの標準的な操作方法
7. **BLOB/CLOB**：バイナリデータと大容量テキストの管理

これらの技術を組み合わせることで、従来のリレーショナルデータベースの枠を超えた、より柔軟で強力なデータ管理システムを構築できます。特に学校データベースのような多様なデータ形式を扱うシステムでは、これらの技術の適切な活用が、システムの価値と使いやすさを大幅に向上させることができます。
