23. RDB システム管理に関する知識 II

1. 科目の概要

関係データベースの運用管理機能と管理方法について、最適化やチューニング、トラブルシューティング方法といった応用知識を説明する。また実際の DB アプリケーション構築手順やインデックス導入による検索高速化、日本語処理環境の諸問題、ツールによる操作やバックアップとリカバリなど、具体的な管理手順を解説する。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説 明	シラバスの対応コマ
II-23-1. データベースの環境構築	データベースの構築からアプリケーションの実装までの手順をMySQLを用いて具体的に説明する。MySQLのインストールから初期設定までを示し、テーブルの設計やデータの投入といった具体的な手順を示すことによってアプリケーション構築の事例を提示する。	9
II-23-2. 統計情報の取得	データベースを運用管理するための統計情報取得について解説する。取得すべき情報として、実行計画の情報、データベースのステータス情報、運用中の資源利用状況に関する情報、同時接続数やアプリケーションサーバとの連携におけるトラフィック情報などを挙げる。	7
II-23-3. クライアント側での最適化	利用環境に対するチューニングの一環として、クライアント側での最適化手法を説明する。具体的には、SQL文のチューニング方法、ストアドプロシージャの利用、ビューを用いた最適化などの方法を解説する。	6,8
II-23-4. サーバ側での最適化	利用環境に対するチューニングのうち、サーバ側での最適化手法を説明する。スキーマ設計における最適化や、インデックスの利用、サーバのパラメータチューニングなどサーバ側で対処すべきチューニング方法の具体例を解説する。	6,10
II-23-5. 障害解析方法と対策	データベースのトラブルシューティングについて、各種の対処方法を紹介する。エラーログの解析方法やログからのデータベースの復旧可能性判定、故障発生時のリストア、ロールフォワードなど、具体的な対策手順を説明する。	7,12
II-23-6. 運用管理の実際	オンラインバックアップやリカバリなど、データベースを止めずに運用管理処理を行う現実的な手法について解説する。またテーブルメンテナンスやリアサイメントといったシステムの運用効率を保ち障害を未然に防ぐための作業も紹介する。	14
II-23-7. レプリケーションによる信頼性改善	レプリケーションの考え方や仕組み、レプリケーションを設定する方法について説明する。また関連情報として分散データベースにおけるテーブル位置と結合仕様の決定手順、分散データベースにおけるストアドプロシージャやトリガの利用方法も説明する。	13
II-23-8. 運用時の制約条件と高可用性運用	データベースに障害が発生した際にバックアップシステムへ切り替える時間や、トラブル時にバックアップを取る時間など、実際の運用において考慮すべき制約条件について解説する。また信頼性を高めたシステムによる高可用性運用について述べる。	7
II-23-9. 各種ツールによるデータベース操作	GUIツールを利用したデータベースの操作方法について説明する。MySQLを題材に用いて説明し、モデリングツール「MySQL Workbench」や管理ツール「MySQL Administrator」、「PhpMyAdmin」などを用いたデータベース操作の手順を示す。	13
II-23-10. 陥りがちなトラブル対策	実際のアプリケーション構築やマイグレーションで陥りがちなトラブル対策について説明する。日本語処理特有の問題として、日本語キャラクタの取り扱いや固有の問題、BLOBの取り扱い方法など、関連する問題点と対策について解説する。	13

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「23. RDB システム管理に関する知識Ⅱ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）					応用レベル（Ⅱ）									
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
23. RDBシステム管理に関するスキル	<データベース運用管理の目的と項目>	<データベースの運用作業と障害回復>	<データベース運用設計>	<データベースセキュリティ>	<データベースリカバリ設計>	<データベースの最適化>	<データベースのトラブル>	<データベースチューニング>	<データベース構築>	<データベースインデックスを用いた性能改善>	<MySQLの導入と運用>	<データベースストラブルシューティング>	<データベース運用環境構築>	<データベース運用>	<データベースチューニング>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_23.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎 [23-5]	IT-IM2. データベース問合わせ言語 [23-9]	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計 [23-3] [23-8]	IT-IM5. データと情報の管理 [23-1, 2, 4, 5] [23-8]	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトエアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータとアーキテクチャ	13	CE-CA0 コンピュータのアーキテクチャと構成	CE-CA00. 歴史と概要	CE-CA01. コンピュータアーキテクチャの基礎	CE-CA02. メモリシステムの構成とアーキテクチャ	CE-CA03. インタフェースと通信	CE-CA04. デバイスサブシステム	CE-CA05. CPUアーキテクチャ	CE-CA06. 性能・コスト評価	CE-CA07. 分散・並列処理	CE-CA08. コンピュータによる計算	CE-CA09. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

IT 知識体系と共通した RDB の実運用に関する内容を、MySQL、PostgreSQL という 2 つの OSS 実装を通して習得する。なお、ここでの解説では MySQL を例に表現しているが、実習の際には両方を比較するのが望ましい。

科目名	第6回	第7回	第8回	第9回	第10回	第11回	第12回	第13回	第14回	第15回
23.RDB システム管理に関する基礎知識Ⅱ	(1)アプリケーション性能向上施策 (2)オプティマイザの動作	(1)トラブルから見たRDBMSの運用ポイン (2)トラブル対応の内容	(1)RDBMS 性能設計のポイント (2)設計とその実装時の留意点	(1)MySQL の概要と環境構築 (2)アプリケーションの実装	(1)データベース要件の検討 (2)改善プランの作成	(1)mysql 上での運用コマンドとSQL (2)ユーザの管理 (3)バックアップ (4)バッチ処理 (5)my.cnf ファイル	(1)障害の発生と原因の調査 (2)アプリケーション要件の調査 (3)インデックスの設定状況と十分性の検証 (4)原因の解明と対処	(1)日本語処理環境の実装 (2)GUI ツールによるデータベース操作 (3)分散データベース設計	(1)バックアップリカバリ (2)バックアップリカバリの実施手順	(1)パフォーマンスチューニング (2)システムチューニング

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-1. データベースの環境構築	
対応する コースウェア	第 9 回 データベース構築	

II-23-1. データベースの環境構築

データベースの構築からアプリケーションの実装までの手順を MySQL を用いて具体的に説明する。MySQL のインストールから初期設定までを示し、テーブルの設計やデータの投入といった具体的な手順を示すことによってアプリケーション構築の事例を提示する。

【学習の要点】

* MySQL はソースコードからのインストールの他、RPM パッケージによりインストールすることができる。
* mysql コマンド等を用いてアプリケーションで利用するデータベースの作成及びデータの登録方法を確認する。

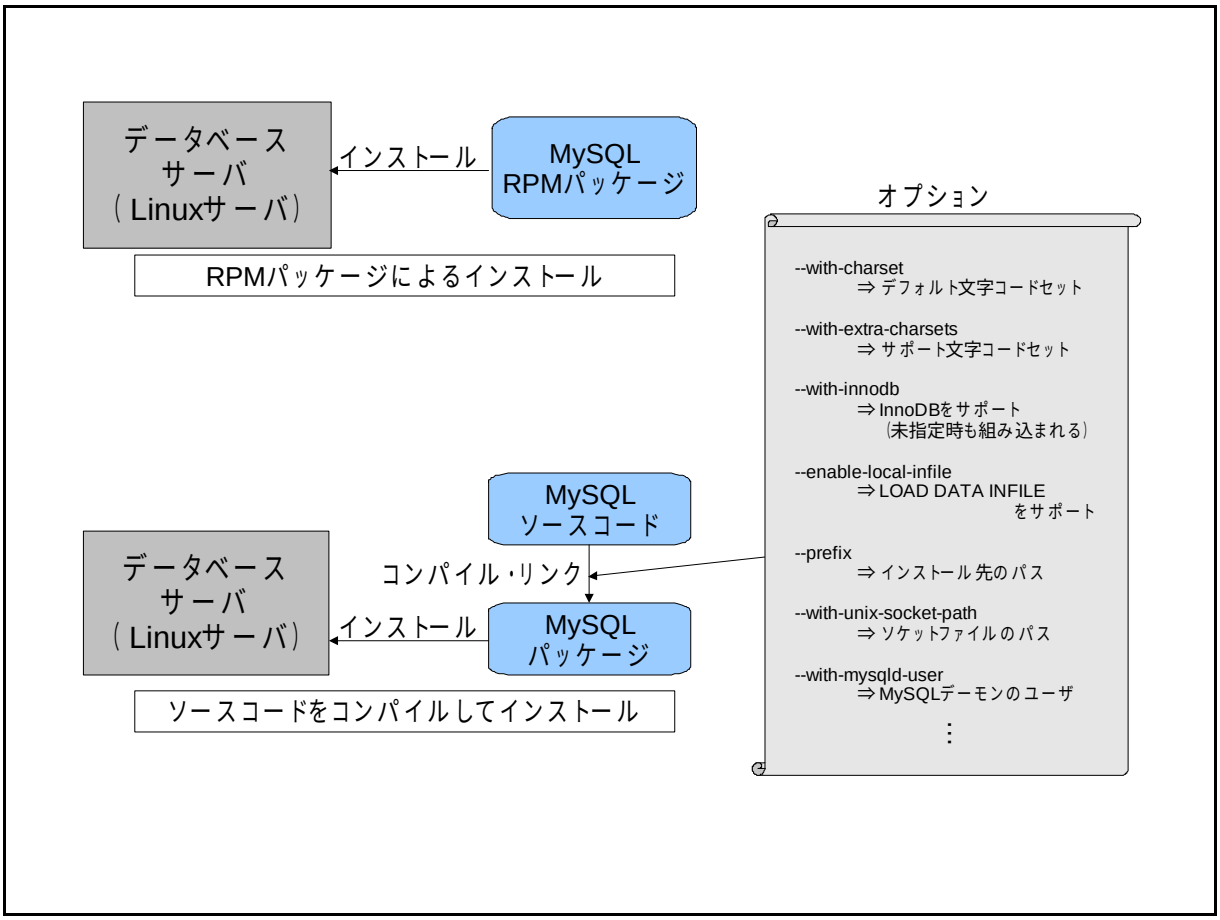


図 II-23-1.　MySQL のインストール

【解説】

MySQLはソースコードからのインストールの他、RPMパッケージなどによりインストールすることができる。RPMパッケージからのインストールはインストールが容易であること、ソースコードからのインストールはコンパイル時のオプションの指定ができることなど、長所と短所があるため、利用目的によって最適なものを選択する。ここでは、Linux 上に MySQL をソースコードからインストールする例、およびローカル接続で MySQL を起動してデータベースを作成する例を示す。

1) ソースコードの入手

配布先（http://dev.mysql.com/downloads/）からソースコードのファイル（**mysql-xxx.tar.gz**）を入手する。（「xxx」は MySQL の任意のバージョンを示す。）

2) MySQL 用のユーザとグループの登録

```
# groupadd mysql
# useradd -g mysql -s /noexists -d /usr/local/mysql mysql
```

3) ソースファイルの解凍

```
$ gunzip -c mysql-xxx.tar.gz | tar xvf
```

4) コンパイル

任意のオプションを指定して、コンパイルを実行し、実行モジュールを作成する。

```
$ ./configure --with-charset=eucjpms --with-extra-charsets=all --with-innodb
--enable-local-infile --prefix=/usr/local --with-unix-socket-path=/tmp/mysql.sock
--with-mysqld-user=mysql
$ make
# make install
```

5) 権限データベースの作成

MySQL のデーモンを起動するためには、権限データベースの作成のスクリプトを実行する必要があるため、あらかじめ作成する。

```
# /usr/local/bin/mysql_install_db --user=mysql
```

6) 起動の準備

```
# cp /usr/local/share/mysql/mysql.server /etc/init.d/mysql
```

7) MySQL デーモンの起動

/etc/init.d/mysql start

8) MySQL コマンドの起動

```
# mysql -u root
```

9) データベースの作成

```
mysql> CREATE DATABASE db1;
```

10) データベースの表示

```
mysql> SHOW DATABASES;
```

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-2. 統計情報の取得	
対応する コースウェア	第 7 回 データベースのトラブル	

II-23-2. 統計情報の取得

データベースを運用管理するための統計情報取得について解説する。取得すべき情報として、実行計画の情報、データベースのステータス情報、運用中の資源利用状況に関する情報、同時接続数やアプリケーションサーバとの連携におけるトラフィック情報などを挙げる。

【学習の要点】

* データベースのチューニングを行うためには統計情報を採取し、現在のデータベースにおいてどこに問題があるのか把握する必要がある。
* データベースの統計情報としてはデータベースの利用状況に関するものや SQL の実行に関するものがある。
* データベースの利用に関する統計情報としては、資源の利用状況や同時接続数などがあり、SQL の実行に関するものとしては SQL の実行計画、サーバクライアント間のトラフィック情報などがある。

SHOW STATUS

Open_tables	サーバが稼動してから開いたテーブルの総数
Max_used_connections	サーバが稼動してから同時使用した接続の最大数
Threads_connected	現在の接続数
Bytes_sent	サーバからの送信の転送量
Bytes_received	サーバからの受信の転送量

EXPLAIN

type	実行計画の結合型 （「ALL」は全表検索となったことを示し、要改善の対象となる）
key	使用するインデックス
ref	参照先としてマッチする結合項目
rows	検索が必要となる件数の概算

図 II-23-2.　MySQL での統計情報と実行計画の主な項目

【解説】

1) 統計情報の採取

統計情報で、サーバの動作状況の統計を確認することができる。これによりデータベースが非効率な動作をしていないか、異常な状態がないかなどを確認できる。
MySQL では、「SHOW STATUS」コマンドを使用する。
チューニングに役立つ統計情報の項目の例として以下のものがある。

- Max_used_connections ：サーバが稼動してから同時使用した接続の最大数
- Bytes_sent ：サーバからの送信の転送量
- Open_tables ：サーバが稼動してから開いたテーブルの総数

「mysqladmin」コマンドの「extended-status」オプションを用いても SHOW STATUS コマンドと同様の内容を確認できる。mysqladmin コマンドを使うと定期的に画面を更新して表示することもできる。
GUI ツールの MySQL Administrator を利用すると接続数やトラフィックなどをサーバの負荷状態を視覚的に確認することができる。

2) システム変数の採取

RDBMS に対して設定した内容が利用目的に即しているか確認する。
MySQL では、「SHOW VARIABLES」コマンドで、最大接続数、キャッシュサイズ、文字コードなどの、システム変数の内容を確認できる。mysqladmin コマンドの「variables」オプションでも同様の内容を確認できる。

3) 実行計画（クエリプラン）の採取

RDBMS のオプティマイザは、作成した SQL に対して、結合順序、インデックスの使用有無などの実行計画を決定する。
通常、作成された実行計画は、効率的で最適なものとなるが、RBMBS が推定したテーブルの取得件数に見込み違いがあった場合などでは、非効率になってしまうこともある。

* 実行計画の採取方法
アプリケーション操作上の待ち時間（レスポンスタイム）や統計情報、スローログの確認などで、期待通りのパフォーマンスが得られていない SQL が発見された場合に、実行計画が不適切なものとなっている可能性がある。
SQL の作成中に作成している SQL が効率的に動作しているかを確認する場合に「EXPLAIN」命令を用いて実行計画を採取する。該当の SQL の先頭に EXLPAIN をつけて SQL を実行すると、問い合わせ文の実行計画が表示され、インデックスの効果などを確認できる。これにより、SQL の見直しが必要か、インデックスの追加が必要かなどを検討できる。

* 実行計画の確認に基づく改善
実行計画の「type」（実行計画の結合型）が「ALL」（全表検索）となった場合は該当のテーブルのレコードが全件検索されてしまい、レコード件数によってはパフォーマンスが著しく低下する恐れがあるため、インデックスの追加等の改善を検討する。
更に SQL の記述で改善できる箇所があるか検討する。SQL の見直しとしては、結合順序やインデックスの使用の指定を定義するなどの、実行計画を指定できる文がある。これには、テーブル結合の順番を指定する「STRAIGHT_JOIN」や、どのインデックスを使用するか指定する「USE INDEX」などがある。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-3. クライアント側での最適化	
対応する コースウェア	第 6 回 データベースの最適化 第 8 回 データベースチューニング	

II-23-3. クライアント側での最適化

利用環境に対するチューニングの一環として、クライアント側での最適化手法を説明する。具体的には、SQL 文のチューニング方法、ストアドプロシージャの利用、ビューを用いた最適化などの方法を解説する。

【学習の要点】

* クライアント側での最適化としては主に SQL の実行に関するチューニングを行い、SQL の見直しやクライアント・サーバ間のトラフィックの改善などがある。
* SQL の見直しは実行計画を採取し SQL 実行にてボトルネックとなる部分を抽出し SQL の実行時間が最小になるように SQL の改善を行う。
* クライアントから複数の SQL を実行する処理を行う場合、実行する SQL をストアドプロシージャやビューとして作成することにより、クライアント・サーバ間のトラフィックを低減し処理時間の改善を行う。

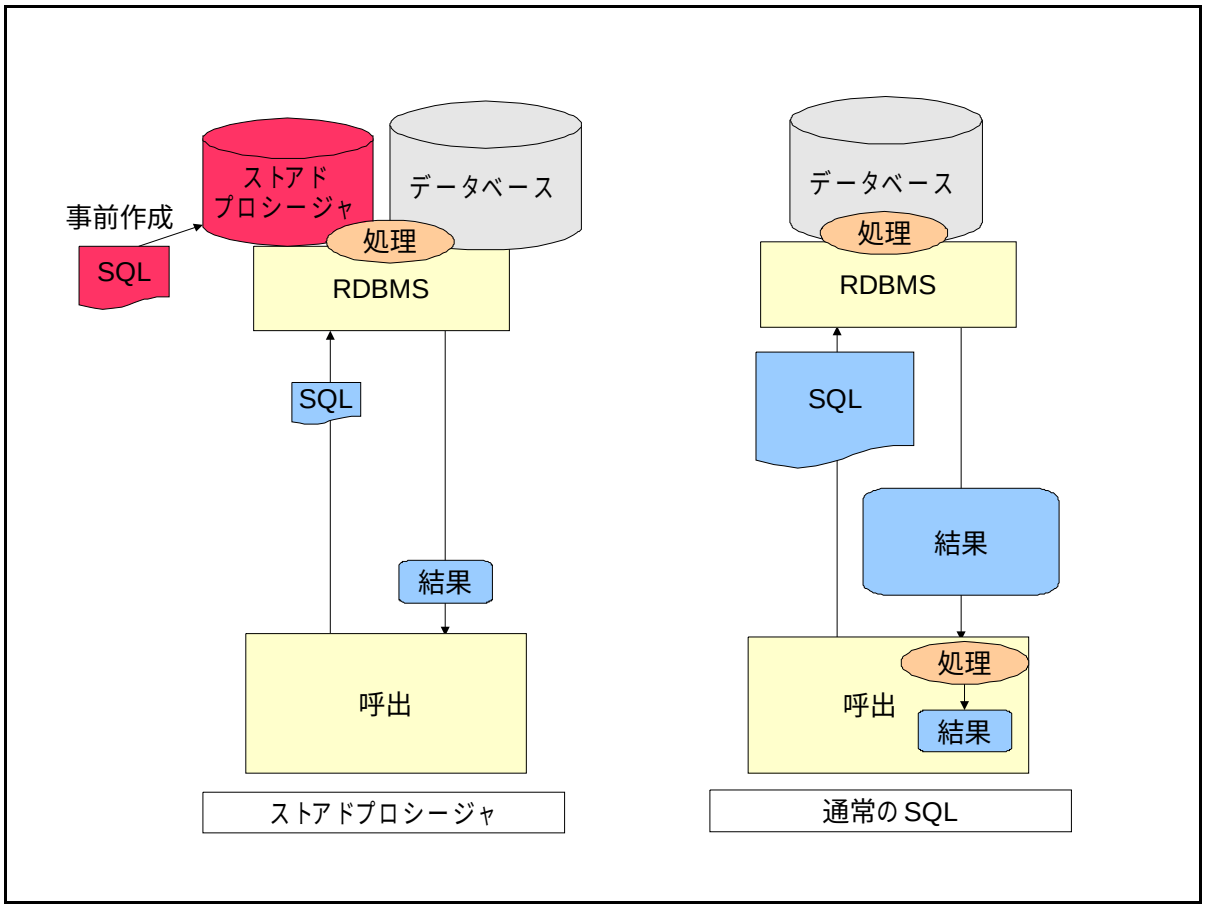


図 II-23-3.　ストアドプロシージャのパフォーマンス向上効果

【解説】

1) SQL の見直し

高いパフォーマンスを得るためには、不必要なデータの取得を避けるべきである。これにより、データベースの負荷が軽減され、呼出側のプログラムのサーバ間とのトラフィックも軽減される。

* データ取得件数を絞り込む
  データを取得する場合に、必ずしも全データが必要ではないことも多い。このような場合には、LIMIT 句を用いて、不必要なデータを取得しないようにすると良い。全件必要な場合においても、ページネーションなどアプリケーションとの連携も含めて転送量を抑える工夫を検討する。

* 必要なフィールドのみ取得する
  データを取得する場合に、「SELECT *」のようにすべてのフィールドを取得するよりも、「SELECT c1, c2 」のように必要なフィールドのみを取得すると、DB サーバの処理時間は一概に高速になるとは限らないが、呼び出し側と DB サーバとの間でのデータの転送量が低減されることにより、高速な結果が得られる。また、一般に保守性も高くなるとされる。

* 結合方法を見直す
  実行計画を確認した結果、期待通りの実行計画になっていない場合には、SQL で結合順序やインデックスの使用の指定を定義する。

2) 問題箇所の特定

データベースのパフォーマンスが問題になった際に、チューニングを行うためには、ボトルネックとなっている SQL がどれなのかを認識することが大切である。このような場合は、ログファイルで、SQL の実行状態を確認すると効果的である。

* ログファイルの記録方法
  MySQL では、「my.cnf」設定ファイルに「log_slow_queries」(スローログの記録) 設定を追記することで、SQL の実行に長い時間を要した場合に「slow.log」ファイルにログが書き込まれる。これを確認することにより、長時間を要した SQL の洗い出しが可能となる。「long_query_time」に任意の秒数を指定することで、その秒数以上に時間がかかった SQL が記録される。

* ログファイル確認上の注意点
  実行時間が長いものを記録する性質上、サーバが全体的に負荷が高い状態になっている場合や、抽出条件の関係上、取得件数が膨大だった場合など、SQLとしては改善の必要がないものも記録対象になってしまうため、その点を加味して、チューニング対象の SQL を検討する。

3) ストアドプロシージャ

ストアドプロシージャはデータベースに複数の SQL やプログラム (関数) を組み込むことができる機能である。

MySQL では、ストアドプロシージャは、5.0 以降のバージョンから備わった機能であり、それ以前のバージョンでは利用できない。

* ストアドプロシージャの利点
  - ストアドプロシージャは RDBMS に事前登録しておくため、実行時は最小限のパラメータを渡すだけで済み、トラフィックが軽減される
  - 必要な情報のみを RDBMS から返すようにすることで、呼出側のプログラムで取得結果に対して複雑な処理を行う場合と比べて、トラフィックの軽減や実行効率の向上に寄与する
  - 効果的に使用することにより、プログラムの開発効率向上ないし保守性向上に役立つ

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-4. サーバ側での最適化	
対応する コースウェア	第 6 回 データベースの最適化 第 10 回 データベースインデックスを用いた性能改善	

II-23-4. サーバ側での最適化

利用環境に対するチューニングのうち、サーバ側での最適化手法を説明する。スキーマ設計における最適化や、インデックスの利用、サーバのパラメータチューニングなどサーバ側で対処すべきチューニング方法の具体例を解説する。

【学習の要点】

* サーバ側の最適化としては主に SQL の処理時間に関するチューニングを行い、データベースの構成の見直しや利用状況に合わせたリソースの最適化などがある。
* データベースを利用する業務から列の構成やキーなどの見直しを行い、表の結合や分割を行うことにより最適なスキーマを設計することで処理時間の改善を行う。
* 使用する表の抽出条件からインデックスを設計・適用し、データ検索時の処理時間の改善を行う。
* データベースへの同時アクセス数やリソースの利用状況を取得し、データベースの実行パラメータを最適化することによりデータベースシステム全体の処理時間の改善を行う。

SQLチューニング

データ件数	データ件数を確認しボトルネックになりそうなものがないか
索引付加状況	検索項目でインデックスが付加されていないものがないか
データ型	データ型が適切に指定されているか、 結合項目で型が不一致となっている項目がないか
結合順序	結合順序はデータ量が多い方から少ない方へ向かい抽出されているか

パラメータチューニング

メモリの割り当て	データベースに対して最適なメモリが割り当てられているか
キャッシュの割り当て	インデックスキャッシュの量が必要十分か、 クエリキャッシュが効果的に働く利用目的か
最大接続可能数	サーバの性能に対して過剰な接続数にしていないか

図 II-23-4.　データベースの最適化要素

【解説】

1) スキーマ設計における最適化

パフォーマンス向上を目的としたスキーマ設計の例を挙げる。

* インデックスの利用

スキーマの最適化の検討の際には、検索パフォーマンス向上のために、どのようなインデキシングを行うべきかを第一に検討するべきである。インデックスは、検索、結合、ソートなどの処理の高速化に役立つ。

実行計画の採取（EXPLAIN）によって、インデックスが適切に利用されているか確認できる。

* 列の構成の見直し

結合におけるパフォーマンス低下を回避するためのデータベース設計方法として非正規化がある。しかし、データの不整合によるトラブルの発生原因となるため、適用するかどうかは慎重に検討するべきである。

非正規化以外の方法として、パフォーマンス向上のために、別なテーブルを用いる方法もある。例えば、主となる正規化したデータを通常のテーブルとして利用して、検索用に非正規化テーブルを別途用意する、または、過去年度などのデータをバッチ処理で別テーブルに移し、これらのデータをオンライントランザクション処理の対象としない、などの方法が採られる場合もある。

2) サーバ側のパラメータチューニング

MySQL では設定ファイル（my.cnf）を編集してパラメータをチューニングできる。設定量は CPU の性能やディスクの性能、メモリ搭載量などのリソースに合わせて調整する。

* キャッシュの設定

データベースシステムは大量のデータを蓄積して扱う性質上、ディスクアクセスが特にボトルネックになりやすい。そのため、メモリをキャッシュに用いて、ディスクアクセスの処理の遅さを緩和する方法が採られている。近年、64bitOS の普及により、メモリを大量に搭載してパフォーマンスを向上させるチューニング手法も可能となっている。

以下にキャッシュの種類の例を挙げる。

- クエリキャッシュ

過去のクエリ内容と結果をキャッシュに記録して、同一の抽出が頻繁にある場合にキャッシュから出力する。CMS など同一の内容を頻繁に表示することが多い用途では、大幅なパフォーマンス改善が期待できる。

- インデックスキャッシュ

インデックス項目をキャッシュに記録して、インデックス項目の検索パフォーマンスを向上させる。

* 最大接続数

無尽蔵に接続を許可した場合に、サーバの負荷が増大し、サーバがサービス停止してしまうことを防止するために、接続可能数に制限値を設ける。

* 断片化の解消（OPTIMIZE TABLE）

データの断片化を解消しパフォーマンスの改善とディスク容量を節約する。レコードの削除が多く発生した場合や、可変サイズのフィールドがある場合に有効である。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-5. 障害解析方法と対策	
対応する コースウェア	第 7 回 データベースのトラブル 第 12 回 データベーストラブルシューティング	

II-23-5. 障害解析方法と対策

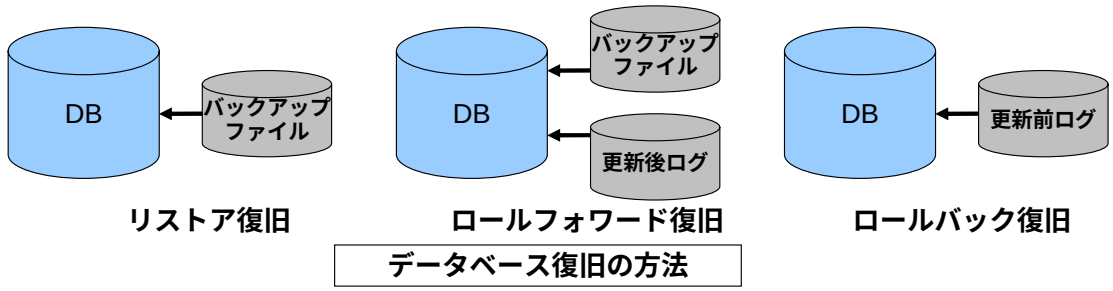
データベースのトラブルシューティングについて、各種の対処方法を紹介する。エラーログの解析方法やログからのデータベースの復旧可能性判定、故障発生時のリストア、ロールフォワードなど、具体的な対策手順を説明する。

【学習の要点】

* データベースの障害としては、SQL の実行によるトランザクション障害、データベースシステムや OS のバグなどによるシステム障害およびハードディスクの故障による媒体障害がある。
* エラーログや SQL の実行ログから障害の発生原因や喪失したデータの範囲を確認し、どこまで復旧が可能か検証を行う。
* システム障害や媒体障害によりデータが喪失した場合、バックアップデータのリストアおよびトランザクションログを利用したロールフォワードを行うことにより、データの復旧を行う。

レベル	障害概要	原因の例
軽度	トランザクション障害	デッドロック検出 プログラムによる異常検出 プログラムの強制終了 など
中度	システム障害	データベース管理システムの障害 OSの障害 CPUやメモリなどのハードウェアの障害によるシステム停止
重度	媒体障害	ディスク装置の故障 ディスク媒体の故障によるデータ全体または一部の破壊、損失

データベース障害のレベルと原因



データベース復旧の方法

図 II-23-5.　データベース障害の原因と復旧方法

【解説】

データベースで発生する障害としては、障害の度合いによりいくつかのレベルがある。
エラーログや SQL の実行ログから、エラーの発生場所、発生時の SQL の実行状況を確認し、障害レベルの判定やデータ復旧の必要性などの判定を行う。
判定に当たっては、事前に各障害のレベルの発見方法や障害発生時のリカバリの方法を定めた運用ポリシーを作成しておく。
それぞれの原因と対応策を以下に示す。

1) トランザクション障害（軽度）

* 原因
  SQL 実行時におけるデッドロック検出、プログラムのバグなどにより異常終了したなど。
* 対策
  トランザクション障害は事前に想定されるものであり、自動的にロールバックによりデータ復旧が行われるように設定を行う。

2) システム障害（中度）

* 原因
  データベース管理システムや OS の障害、CPU やメモリなどのハードウェアの障害によりシステムが停止したなど。
* 対策
  障害の原因を改善しシステムを再起動することにより、データベースシステムが自動的に起動され、実行中であったトランザクションも再起動するように設定を行う。
  データの破壊や喪失が発見された場合は、データベースのリストアを行いデータの普及を行う。

3) 媒体障害（重度）

* 原因
  ディスク装置やデバイス媒体の故障により、データの全体又は一部が破壊又は喪失するなど。
* 対策
  ディスク装置やデバイス媒体を交換後、データベースのリストアを行いデータの復旧を行う。

4) データベースの復旧

データベースの復旧方法としては大きく以下の 3 つに分けられる。

* リストア復旧
  バックアップ機能によって保存されたバックアップファイルを用いて、障害が発生する直前にバックアップした状態へ復旧を行う。
* ロールフォワード復旧
  リストア復旧に加え、トランザクションログを用いて更新順に処理を再現し、障害が発生する直前の状態へ普及を行う。
* ロールバック復旧
  トランザクションログを用いてデータベースへの更新された部分を元に戻して行くことによりデータベースの更新を無効にする。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-6. 運用管理の実際	
対応する コースウェア	第 14 回 データベース運用	

II-23-6. 運用管理の実際

オンラインバックアップやリカバリなど、データベースを止めずに運用管理処理を行う現実的な手法について解説する。またテーブルメンテナンスやリアサイメントといったシステムの運用効率を保ち障害を未然に防ぐための作業も紹介する。

【学習の要点】

* データベースの実際の運用ではデータベースを停止してメンテナンスを行うことができない場合が多く、オンラインバックアップによるデータのバックアップや、トランザクションログを作成し、障害発生時の復旧作業に備える。
* トランザクション障害やシステム障害が発生した場合は、トランザクションログを利用しオンラインでリストアすることも可能である。
* レコードの削除や更新からデータベース領域が断片化するなどに無駄なスペースが生じパフォーマンスが低下するため、定期的にテーブルの再編成や再配置を行う必要がある。

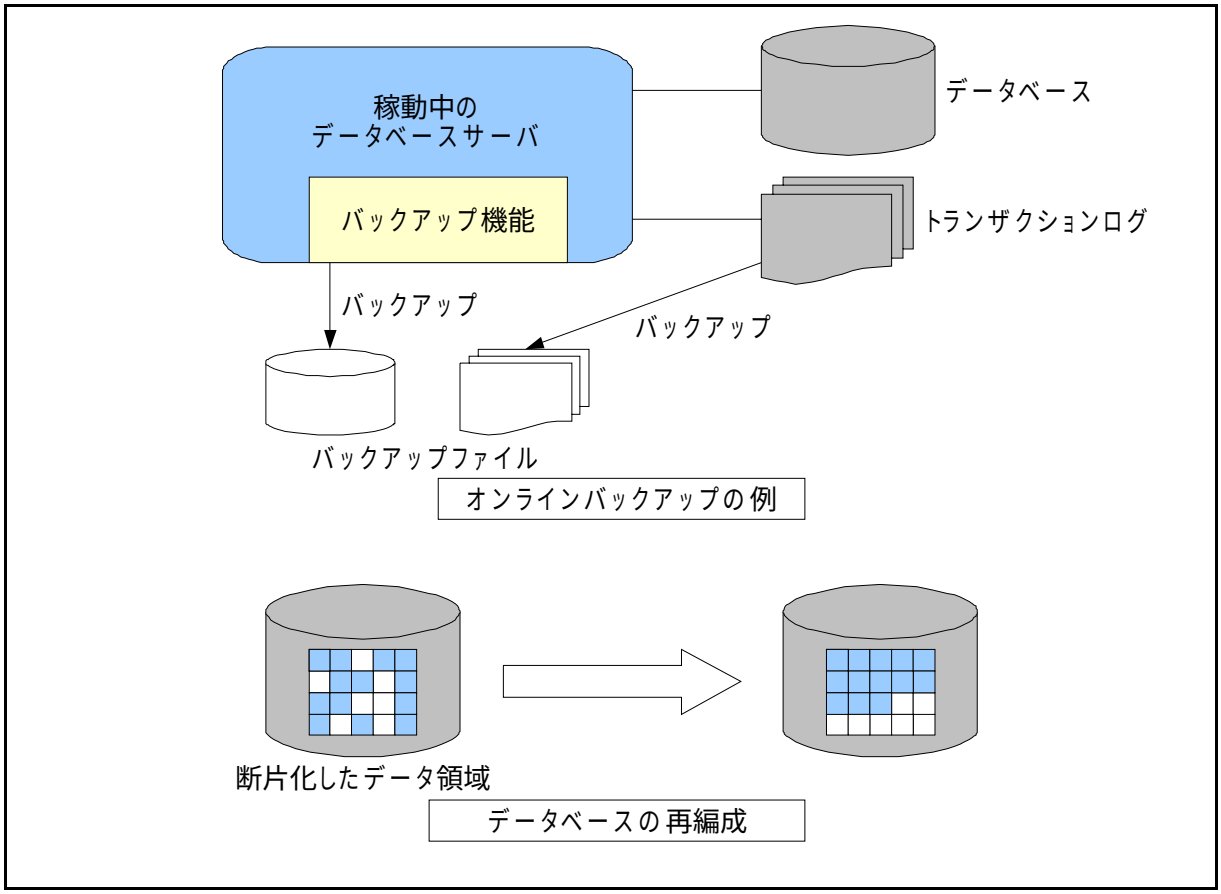


図 II-23-6.　オンラインバックアップとリストア

【解説】

1) 運用管理におけるバックアップ

バックアップはデータベース障害が発生した時に備えて取得するデータのコピーである。
データ更新の頻度、サービスの提供時間などから、バックアップの対象となるデータ、バックアップ時のシステムの状態、バックアップを保存する媒体といったバックアップ方法を設計する。
ネットワークサービスなどデータベースサーバを停止できない場合、オンラインバックアップにてバックアップを行う。

2) オンラインバックアップ

データベースシステムを動作している状態でバックアップを行うことをオンラインバックアップ（またはホットバックアップ）と呼ぶ。
オンラインバックアップは通常、データベースシステムが提供するバックアップ機能を使用して行い、MySQL では mysqldump コマンドなどがある。
オンラインバックアップでは、バックアップ中でもデータの更新が行われることがあるため、データベースのトランザクションログを作成するように設定を行う必要がある。トランザクションログには、データベースの更新を行う前の状態を保存する更新前ログと、更新された後の状態を保存する更新後ログがある。このログも合わせてバックアップを行う。

3) オンラインでのデータベース復旧

データベース障害によりデータベースの復旧を行うには、データベースシステムを停止した状態で行う場合が多いが、トランザクション障害や軽度のシステム障害でデータに不整合が発生した場合は、システムを停止することなくオンラインでデータの復旧を行うことができる。
オンラインでのデータの復旧はトランザクションログの更新後ログを用いて、データを更新前の状態に戻すことにより行う。これをロールバック復旧と呼ぶ。

4) データベースの領域管理

データベースの運用管理ではデータベースのバックアップにより障害発生に備えるだけではなく、障害が発生することを未然に防ぐための作業を行う必要がある。
データベースはディスク上に作成された複数のファイルで構成されており、通常はデータを保存する領域の大きさがきめられている。データベースの領域の管理としては以下のような項目がある。

* データベースの空き容量の管理
  データを保存する領域がゼロになってしまうと、データベースシステムにてエラーが発生し新たにデータを保存することが出来なくなり、データベースが正常に動作できなくなる可能性がある。
  空き容量が少なくなった場合はファイルの追加やハードディスクを追加しデータベース領域の作成場所を変更するなどを行いデータベース領域の拡張を行う。

* 連続空き領域の管理
  データベースにてレコードの削除や更新を繰り返すと、データベースの空き領域が断片化する。
  データベース領域の断片化が進むとデータの格納効率や検索効率が悪くなるばかりではなく、空き領域があるにもかかわらず、データを保存することが出来なくなる可能性がある。
  連続したデータベースの空き領域が少なくなった場合、データベースやテーブルを削除し、バックアップファイルをリストアすることでデータベースの再編成を行う。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-7. レプリケーションによる信頼性改善	
対応する コースウェア	第 13 回 データベース運用環境構築	

II-23-7. レプリケーションによる信頼性改善

レプリケーションの考え方や仕組み、レプリケーションを設定する方法について説明する。また関連情報として分散データベースにおけるテーブル位置と結合仕様の決定手順、分散データベースにおけるストアドプロシージャやトリガの利用方法も説明する。

【学習の要点】

* データベースのレプリケーションは分散データベースにより、負荷分散や障害発生時の復旧時間の短縮することを目的として行う。
* レプリケーションの方法としては、全てのデータベースが同時に更新されるマルチマスタ方式と、1 つのマスタへの更新を複数のスレーブへ順次更新していくマスタスレーブ方式がある。
* 分散データベースは表を縦方向および横方向に分散する方法があり、データの機密性やデータベース利用者の組織、地域などにより分散方法を決定する。
* 分散データベースではトリガやストアドプロシージャを作成することより参照整合性を保つことができる。

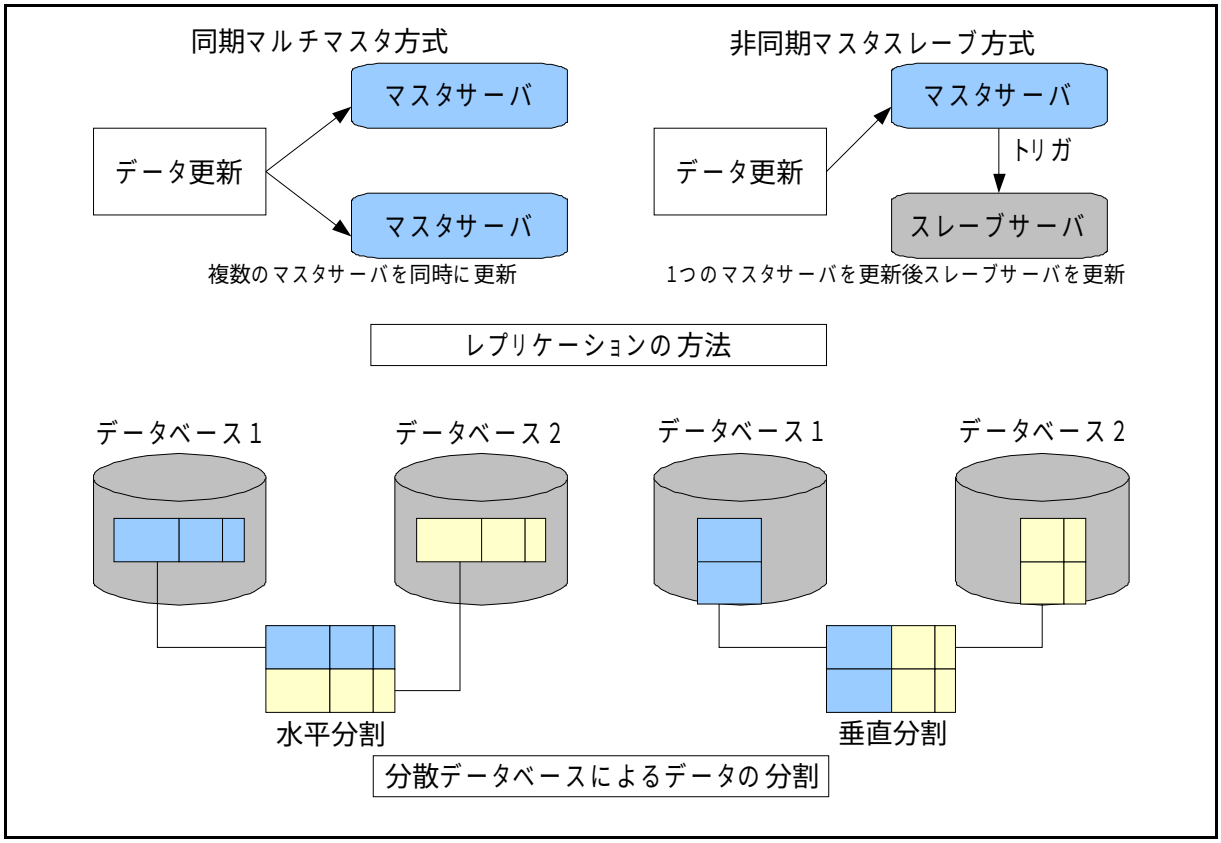


図 II-23-7.　分散データベースとレプリケーション

【解説】

1) データベースのレプリケーション

負荷分散による性能の向上や、障害発生時にシステム停止時間を短縮する為に、分散データベースが使用される。分散データベースではネットワーク上に配置される複数のデータベースで構成されるためデータベース間のデータの同期が必要となる。
データの同期を行う方法の1つとしてレプリケーションがある。
レプリケーションはデータベース間で SQL を用いてお互いのデータをコピーしあうサービスで、以下の 2 つの方法に大別される。

* 同期マルチマスタ方式
  データの更新処理を行うマスタサーバが複数あり、それぞれのマスタサーバがデータベースの複製を持っている。マスタサーバにてデータの更新が行われると、即時に他のマスタサーバに反映される。各マスタサーバにおけるデータ更新の時差は少ないが、他のデータベースのロック処理を行う必要があり、更新処理が複雑になり処理時間が長くなる。
* 非同期マスタスレーブ方式
  データの更新処理を行うマスタサーバは 1 つで、他のサーバはマスタサーバのコピーを持つスレーブサーバとなる。マスタサーバでデータの更新が行われると、各スレーブサーバへ変更内容が通知されデータのコピーを行う。マスタサーバでのデータ更新処理は短い時間で行われるが、全てのサーバでデータの同期が行われるまで時差が発生する。

2) 分散データベースによるデータ分割

分散データベースでは各サーバで管理するデータの同期を行って利用するだけではなく、データの内容によりデータを分散し、それぞれを各サーバにて管理することができる。データを分散する方法としては以下の 2 つに分類される。

* 垂直分割
  データを項目単位で分割するもので、各サーバにて管理を行うデータの範囲を分散しアクセス権限を設定することによりセキュリティを確保することができる。
  各サーバのデータを JOIN で結合することによりデータの照会を行う。
* 水平分割
  データをレコード単位で分割するもので、組織や地域など各サーバにて必要なデータのみ管理を行うことで、データベース領域の増大や検索性能の低下を防ぐことができる。
  各サーバのデータを UNION で結合することによりデータの照会を行う。

3) 分散データベースの更新処理

分散データベースのレプリケーションでは、データの更新や参照を行った時に他のサーバへ通知を行う必要がある。
他のサーバへの通知は「トリガ」を作成することにより行う。
トリガは各テーブルに作成するストアドプロシージャで、トリガを作成したテーブルに対して INSERT、UPDATE、DELETE などの更新が行われると自動的にトリガが実行される。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-8. 運用時の制約条件と高可用性運用	
対応する コースウェア	第 7 回 データベースのトラブル	

II-23-8. 運用時の制約条件と高可用性運用

データベースに障害が発生した際にバックアップシステムへ切り替える時間や、トラブル時にバックアップを取る時間など、実際の運用において考慮すべき制約条件について解説する。また信頼性を高めたシステムによる高可用性運用について述べる。

【学習の要点】

* データベースを停止するケースとしては定期的にメンテナンス作業を行う計画停止と障害の発生により行う計画外停止があり、共に停止回数の減少や停止時間の短縮により可用性を高める必要がある。
* バックアップ方法の最適化によるバックアップ・リストア時間の短縮を図ると共に、データベースの起動パラメータを調整や使用するリソースの変更により障害発生を低減する。
* データベースサーバをクラスタ構成することにより耐障害性を向上すると共に、計画停止によるサービスの停止を行う必要がなくなり、可用性が向上する。

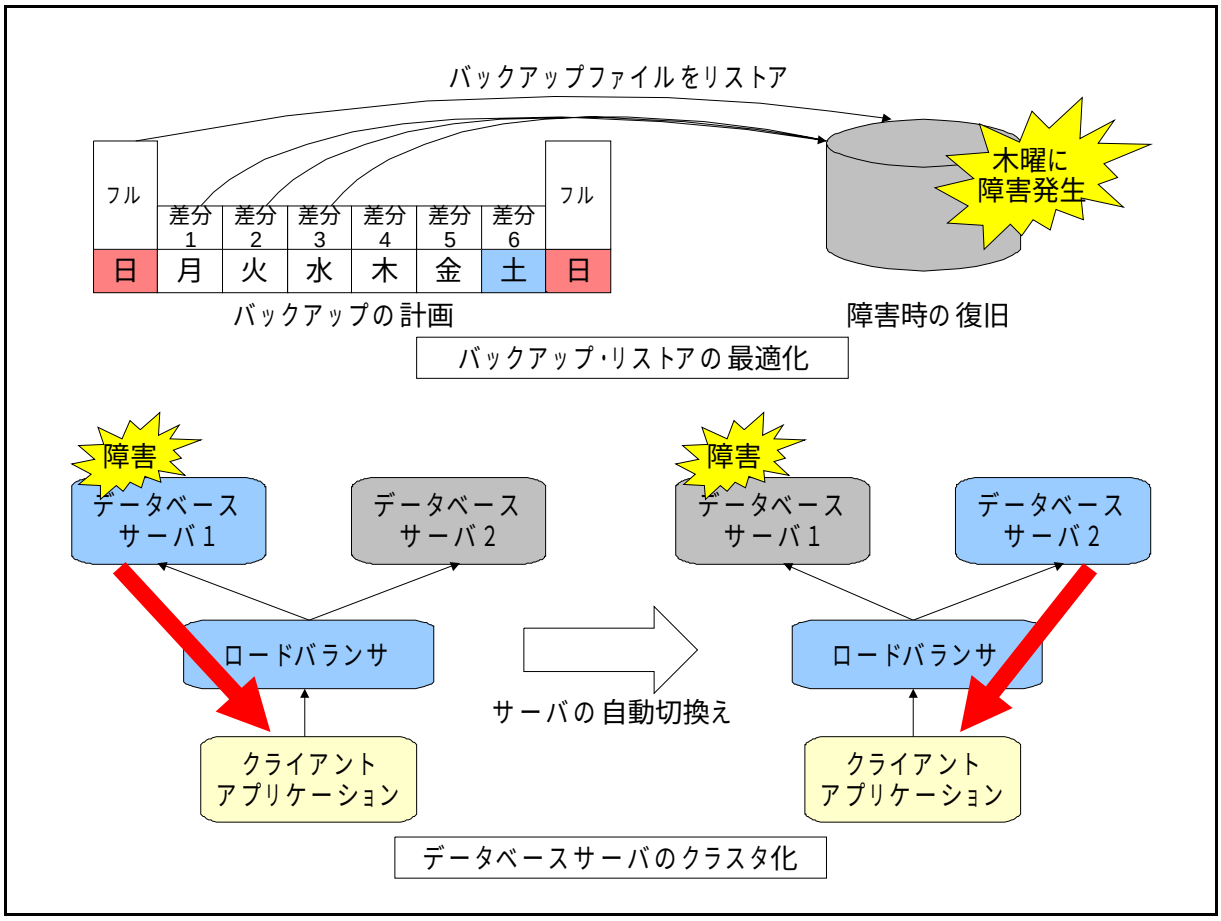


図 II-23-8.　データベースの高可用性運用

【解説】

1) 運用管理におけるデータースの停止

データベースを停止するケースとしては定期的にメンテナンス作業を行うために行う計画停止と、障害の発生及び復旧作業のために行う計画外停止がある。
勤務時間に使用する社内システムや、公開サーバとして 24 時間サービスを提供するシステムなど、用途によってサーバ停止が可能な時間を決めておく必要がある。
また、計画外停止の場合は、障害のレベル毎に復旧までの時間を決めておき、時間内で行う作業を明確にしておく必要がある。

2) バックアップ･リストアの最適化

計画停止の際のデータベースのバックアップは、障害発生時のデータベース復旧時間をできるだけ短縮するようにバックアップ方法やスケジュールを決定する。
バックアップの方法としては、フルバックアップと差分バックアップがあり、この方法を組み合わせてバックアップのスケジュールを決定する。フルバックアップの間隔を長く取るとバックアップに必要な時間を短縮できるが、復旧の際には多くのバックアップファイルをリストアする必要があり復旧までの時間が長くなる。また差分バックアップの方法としては以下の方法があり、実際の運用方法に合わせて最適な方法を選択する。

* フルバックアップからの差分を対象とする
  バックアップのデータ量が大きくなりバックアップ時間も長くなるが、バックアップデータの管理が容易となる。
* 前回の差分バックアップからの差分を対象とする
  バックアップのデータ量が小さくないバックアップ時間も短縮できるが、管理するバックアップデータが多くなる。

3) 障害発生の防止

障害の発生に備えてバックアップを行うと共に、障害が発生しないようにメンテナンスを行うことも重要である。定期的なメンテナンスで行う作業としては、データベースの起動パラメータを調整や使用するリソースの変更などがある。
データベースのログデータを解析し、CPU、メモリやディスクなどのリソースが不足している場合、データベースの起動パラメータを調整し使用するリソースの改善を行うことにより、障害が発生する可能性を低減させる。

4) データベースの高可用性化

データベースの可用性を向上する方法として、データベースサーバのクラスタ化がある。
データベースサーバをクラスタ構成にすると以下のようなことが可能となる。

* 障害発生による停止時間の短縮
  障害が発生した場合、他のサーバへ自動的に処理を引き継ぐことによりデータベースの停止時間を短縮、または停止することなくサービスを提供できる。
* 定期メンテナンスでの計画停止が不要
  サービスを提供するデータベースサーバを切り替えながら順番に定期メンテナンスを行うことにより、データベースを停止することなく作業が行える。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-9. 各種ツールによるデータベース操作	
対応する コースウェア	第 13 回 データベース運用環境構築	

II-23-9. 各種ツールによるデータベース操作

GUI ツールを利用したデータベースの操作方法について説明する。MySQL を題材に用いて説明し、モデリングツール「MySQL Workbench」や管理ツール「MySQL Administrator」、「phpMyAdmin」などを用いたデータベース操作の手順を示す。

【学習の要点】

* データベースの設計、管理を行うため多くの GUI ツールが提供されている。
* MySQL Workbench はデータベースを設計するツールで ER 図などを視覚的に設計、作成、管理することができる。
* MySQL Administrator はデータベースサーバを管理するツールでサーバの状態監視やデータベースのバックアップ・リストアなどを行うことができる。
* phpMyAdmin は PHP で作成されたデータベースを管理するツールで、データベースの作成など SQL の実行を行うことができる。

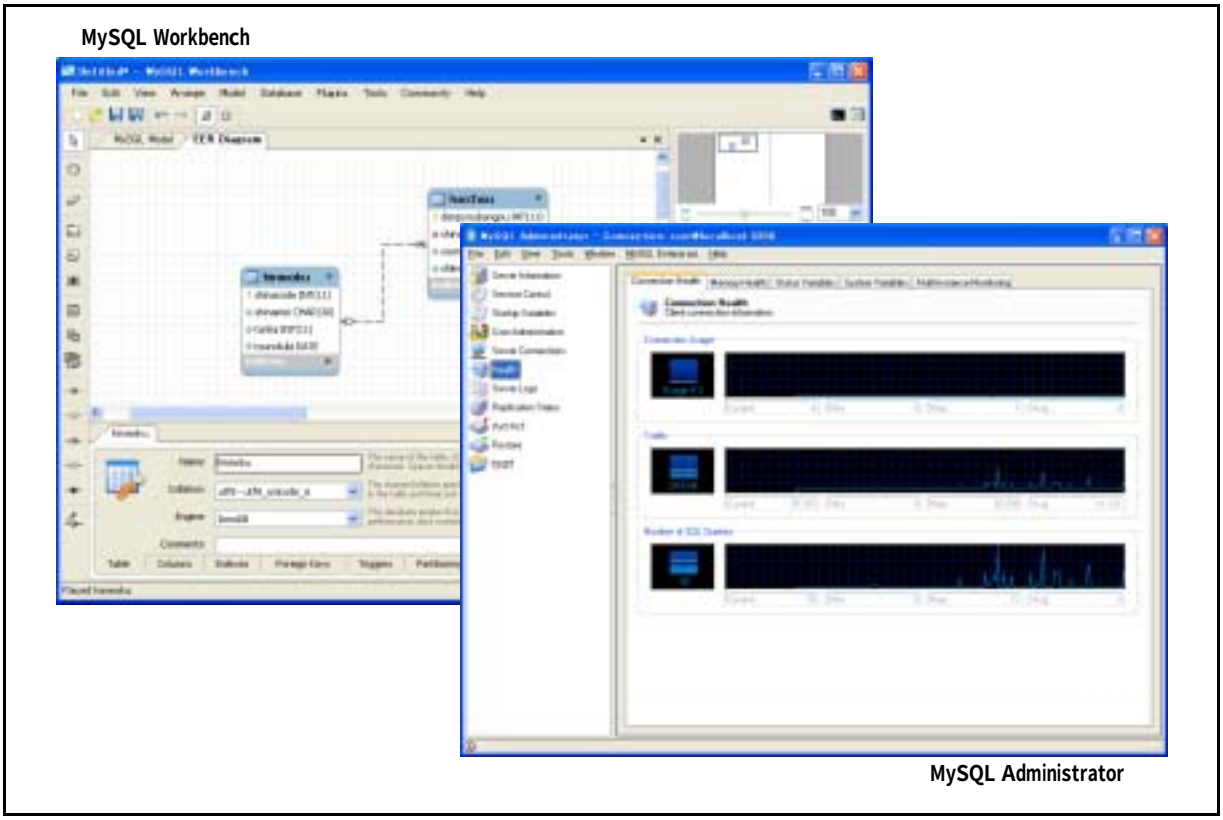


図 II-23-9.　MySQL Workbench と MySQL Administrator

【解説】

データベースの設計･管理を容易に行えるように、多くの GUI ツールが提供されている。
MySQL ではデータベースの設計を行うツールとして「MySQL Workbench」、データベースの管理を行うツールとして、クライアント／サーバアプリケーションでは「MySQL Administrator」、Web アプリケーションでは「phpMyAdmin」などがある。

1) MySQL Workbench

MySQL にてデータベースの設計を行う為のツールで ER 図などを視覚的に設計、作成、管理することができる。以下に主な機能を示す。

* ビジュアルなデータベース設計
  テーブル、インデックス、ビューなどデータベースを構築する為に必要な情報をモデル化してメンテナンスを行う。また ER 図などデータベースのモデリングを行うことができる。
* フォワード/リバースエンジニアリング
  データベースモデルから物理データベースの作成または既存の物理データベースからデータベースモデルの作成を行う。
* 変更管理
  データベーススキーマの変更履歴の保存及び比較を行う。
* データベースのドキュメント出力
  データベースモデルから HTML 形式またはテキスト形式などでドキュメントを作成する。

2) MySQL Administrator

クライアント／サーバ型のアプリケーションでサーバの状態監視やデータベースのバックアップ･リストアなど、MySQL の管理を行うことができる。以下に主な機能を示す。

* Server Information　サーバ情報の表示を行う。
* Service Control　サービスの起動･停止を行う。
* User Administration　ユーザ管理（作成･変更･削除）を行う。
* Health　サーバの稼動状況をグラフで表示する。
* Server Logs　各種ログの表示を行う。
* Backup／Restore　データベースのバックアップの作成、リストアの実行
* Catalogs　テーブルやインデックスの設定内容の表示やメンテナンスを行う。

3) phpMyAdmin

PHP で作成された Web アプリケーションで、データベースの作成や SQL の実行など MySQL の管理を行うことができる。以下に主な機能を示す。

* 複数サーバの管理
* データベースの作成と削除
* SQL ステートメントの実行。
* テーブルのダンプの作成と読み込み、データのエクスポート（CSV、XML、LaTeX 形式など）
* MySQL ユーザ･特権の管理
* データベースレイアウトの PDF 画像の作成

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	23 RDB システム管理に関する知識 II	応用
習得ポイント	II-23-10. 陥りがちなトラブル対策	
対応する コースウェア	第 13 回 データベース運用環境構築	

II-23-10. 陥りがちなトラブル対策

実際のアプリケーション構築やマイグレーションで陥りがちなトラブル対策について説明する。日本語処理特有の問題として、日本語キャラクタの取り扱いや固有の問題、BLOB の取り扱い方法など、関連する問題点と対策について解説する。

【学習の要点】

* 日本語を扱う場合、データベースシステムの内部文字コードの違いにより文字化けや、文字コードの変換において、文字コード「0x5C」などによるラウンドトリップが発生する。
* データベースにて画像などバイナリデータを扱う場合、BLOB(Binary Large Object)を利用するが、データを扱うアプリケーションにより内部形式が異なるためクライアントで正しく表示できない場合がある。
* BLOB などを使用しているデータベースを他の DBMS へ移行を行う場合、サポートされている型の違いや、各型おけるサイズの違いにより型変換が必要となる場合がある。

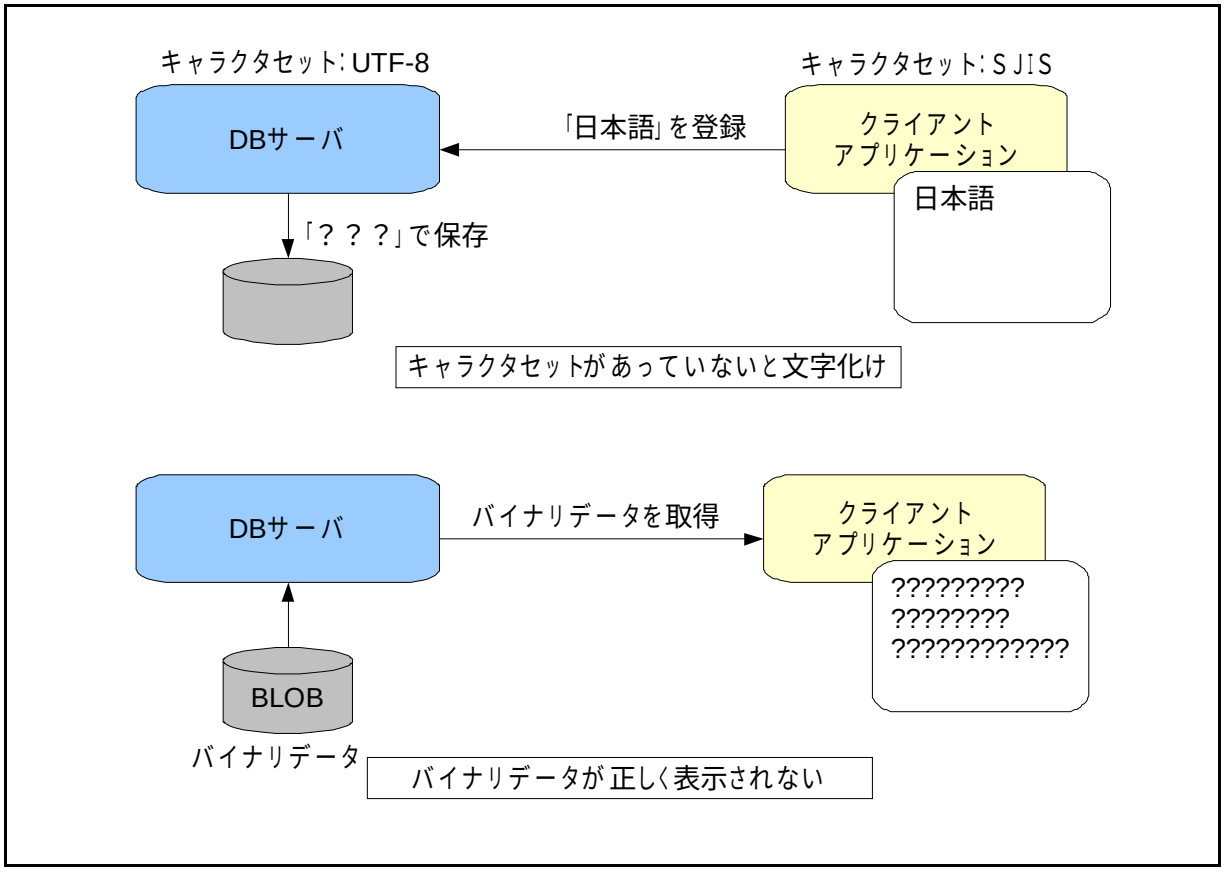


図 II-23-10.　日本語、バイナリデータの障害

【解説】

クライアント／サーバアプリケーションの構築や、データベースが動作する OS またはデータベースシステムを変更する場合など、システム環境の違いによって発生する問題としては以下のようなものがある。

1) キャラクタセットの違いによる障害

データベースにて日本語を扱う場合、アプリケーションの構築ではクライアントとサーバ、OS やデータベースシステムの変更では新･旧のシステムにて扱うキャラクタセットの違いにより障害が発生する場合がある。

キャラクタセットの違いにより発生する障害として以下のような現象が発生する。

- 文字が他の文字に化ける。
- 入力した文字が消える。
- 検索結果に合致しない結果が返ってくる。
- 文字コードの変換にてラウンドトリップが発生する（0x5c など）

これらの問題を解決する為には以下のような対応を行う必要がある。

- サーバのキャラクタセットを日本語のキャラクタセットに変更する。
- クライアントのキャラクタセットをサーバと同じにする。
- OS やデータベースシステムの変更では変更前のデータベースからデータをエクスポートする時に変更後のキャラクタセットに変換を行う。

2) バイナリデータの扱いによる障害

データベースにて画像などバイナリデータを扱う場合、BLOB(Binary Large Object)などバイナリデータを扱うデータの型を利用して保存を行う。

* 保存されたデータを使用する際の障害

保存されたバイナリデータの内部形式は使用するアプリケーションにより決定されるが、データベースでは内部形式まで解釈を行わない為、BLOB へ保存されたデータは検索の対象となりないシステムが多い。また、クライアント側で BLOB を使用する場合、データの内部形式がわからない為、正しく表示できない場合がある。

BLOB を使用する場合は、同一レコード中に検索を行う為のキーや、データの内部形式を識別するデータを保存することが必要となる。

* データ移行時の障害

また、BLOB などを使用しているデータベースを他のデータベースシステムへ移行する場合、サポートされている型の違いや、各型おけるサイズの違いにより型変換が必要となる。

Oracle から MySQL への変更を例にすると、MySQL では Oracle のラージオブジェクト（LOB）に相当するデータ型がなく、BLOB 型などへ変換する必要がある。

ただし、Oracle の LBO はデータの実体が保存されておらず、実体へのロケータのみが取得されるため、実体を読み込んで MySQL へ格納する処理が別途必要となる。