22. RDB に関する知識 II

1. 科目の概要

関係データベース(RDB)に関して、インデックスの取り扱いや SQL によるデータベースの操作など、実際の活用に際して必須の知識について解説する。またオープンソース RDBMS の紹介や企業 DB 設計、DB アプリケーションの例など、RDB に関する応用的な知識を紹介する。

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-22-1. インデックスの概念	データベース検索の高速化に必須の技術であるインデックスの概念について解説する。探索のアルゴリズムなどインデックスの原理について説明し、インデックスの物理的な構造や性能、リカバリ特性など、インデックスに関する基本的な考え方を示す。	9
II-22-2. インデックスの種類と特徴	バイナリツリー(B木)、索引構成表、ビットマップ・インデックス、逆キー・インデックス、ハッシュ・クラスタ(ハッシュ・インデックス)など、様々なインデックスの種類を紹介し、それぞれの特徴と利点・欠点について解説する。	9
II-22-3. データベースの物理設計	関係データベースを実装する際に必要なデータベースの物理構造について説明する。データベース物理設計の目的や、物理設計時に留意すべきトレードオフ、データとインデックス、ログをどのように配置するか、データベース記憶領域の確保など、物理設計に必要な具体的な手順を示す。	10
II-22-4. リレーショナルデータベースとSQL	関係データベースをアクセスするための基本的な技術である Structured Query Language (SQL)の構造、原理、動作仕様、種類、特徴などについて解説する。またSQLの標準化といった関連情報も紹介する。	11
II-22-5. SQLによる表の作成	SQLを用いて表を作成する手順を示す。データ型の指定やNULLの取り扱い、主キーと外部キーの指定方法といった表の作成に関する基本的な項目を具体的に説明する。	11, 12
II-22-6. SQLを用いた問い合わせ処理	SQLを用いたデータの検索、問合せ処理の手順を示す。検索の基本となるSELECT文について説明し、式や数値、文字列、日付といったデータの処理や、WHERE句による演算、集合関数などについて解説する。	11, 12
II-22-7. 表の結合、複雑な処理、テーブルの更新	外部結合や自己結合、副問合せといった表の結合(ジョイン)に関するSQLによる操作方法や、SQLを用いた複雑な検索・照会の方法、テーブルの更新、削除など、SQLによる様々なテーブル操作について説明する。	11, 12
II-22-8. オープンソースRDBMSの種類と特徴	PostgreSQL、Firebird、MySQLといった代表的なオープンソース RDBMSについて、開発の歴史や主たる機能、特徴やライセンスなどについて解説する。さらに商用RDBMSとの比較やオープンソース RDBMSを利用する理由についても説明する。	13
II-22-9. RDBを用いた企業DB設計の例	業務分析としてエンティティを洗い出しER図を作成したのち、スキーマとインデクスの設計、分散データベースの設計という企業基幹データベースの設計手順を示す。また物理設計として性能設計や処理効率化検討といった項目についても解説する。	14
II-22-10. RDBを用いたDBアプリケーションの例	テーブルの作成とデータのロードといった具体的な手順を示し、さらに検索結果の表示やビューの設定といった業務帳票や画面設計の手順、パフォーマンスの検討など、実際のDBアプリケーション開発に必要な項目を解説する。	15

【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

「22. RDB 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
22. RDBに関する基礎スキル	<データベースの基礎理論>	<RDBMSの基本知識>	<トランザクションの基本概念>	<データベースの構成要素>	<DOAの内容概要>	<データベース設計の基本理論>	<ERモデル>	<正規化の手順と方法>	<データベースインデックス>	<データベースの物理構造>	<SQLによるデータベースアクセス>	<SQL実践演習ワークショップ>	<代表的なオープンソースRDBMS製品>	<データベース設計構築の実践>	<データベース構築>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_22.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
組織間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 [22-1]	IT-IM2.データベース問合わせ言語 [22-1] [22-11,12,15]	IT-IM3.データアーキテクチャ [22-2,4,8] [22-14]	IT-IM4.データモデリングとデータベース設計 [22-5,6,7] [22-14]	IT-IM5.データと情報の管理 [22-3] [22-10]	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.ソフトウェアの標準化	
	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

OSS モデルカリキュラム固有の知識はデータベースの OSS 実装に関する知識であり、他の内容は IT 知識体系と共通した RDB に関する内容を扱う。

科目名	第9回	第10回	第11回	第12回	第13回	第14回	第15回
22.RDB に関する基礎知識Ⅱ	(1)インデックスの原理	(1)データベースの物理設計	(1)リレーショナルデータベースとSQL	(1)DDL 系	(1) PostgreSQL	(1)業務分析	(1)テーブルの作成
	(2)インデックスの構造	(2)更新系と検索系処理の物理的特性	(2)表の作成	(2)DML 系	(2)Firebird	(2)企業基幹データベースの設計	(2)業務帳票・画面の作成
	(3)インデックスの種類		(3)問合せ処理	(3)整合性の実装	(3)MySQL	(3)物理設計	(3)パフォーマンスの検討
			(4)複雑な検索・照会	(4)セキュリティ	(4)Oracle 10gとMySQL 4.1.7の機能比較		
			(5)表の結合(ジョイン)		(5)オープンソースRDB が用いられる理由		
			(6)テーブルの更新				

(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	22 RDB に関する知識 II	応用
習得ポイント	II-22-1. インデックスの概念	
対応する コースウェア	第 9 回 データベース構築	

II-22-1. インデックスの概念

データベース検索の高速化に必須の技術であるインデックスの概念について解説する。探索のアルゴリズムなどインデックスの原理について説明し、インデックスの物理的な構造や性能、リカバリ特性など、インデックスに関する基本的な考え方を示す。

【学習の要点】

* データベースに保存されたデータは、二分検索、B 木検索、B+木検索などの検索アルゴリズムを利用したインデックスを作成して検索が行われる。
* インデックスは物理的な配置方法の違いにより、クラスタードインデックスと非クラスタードインデックスがある。

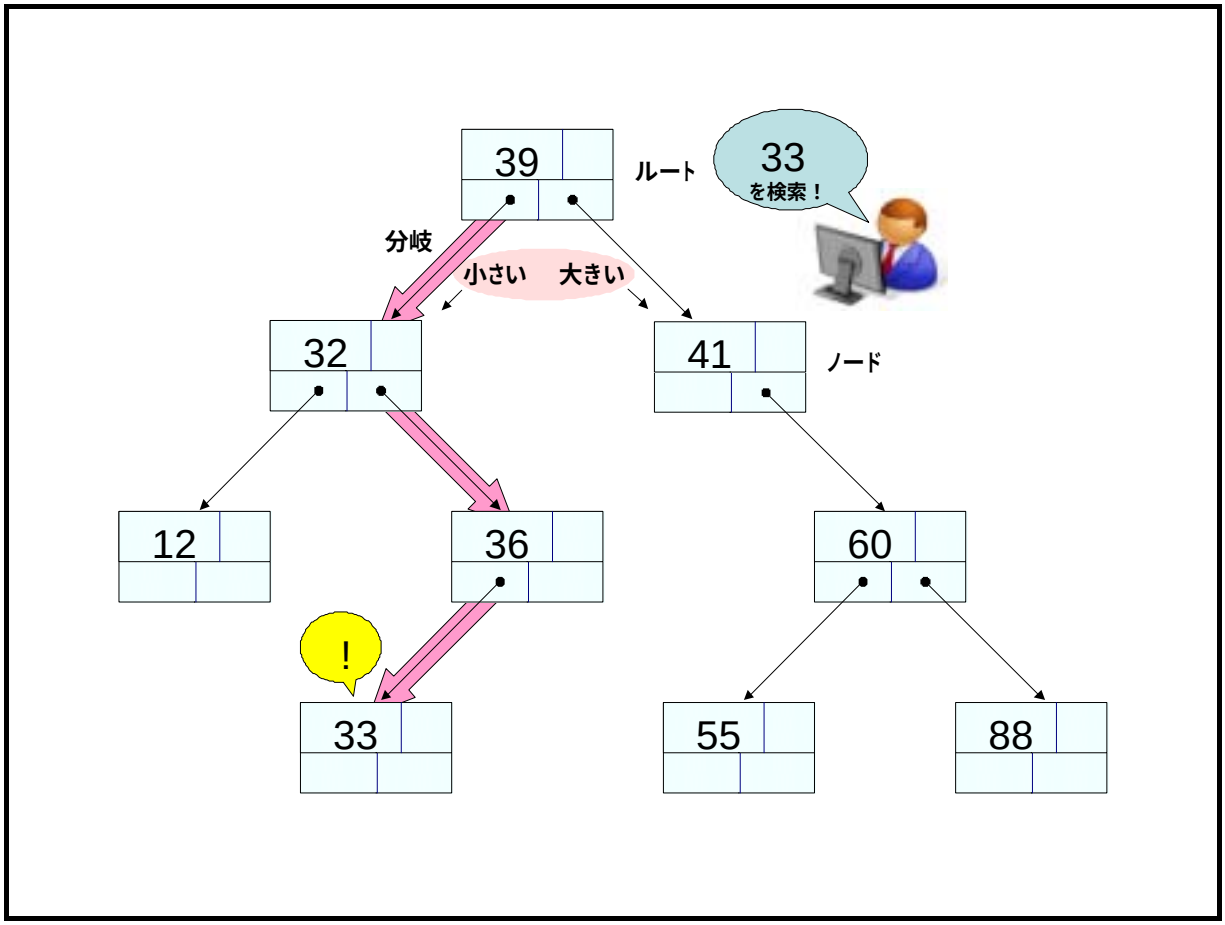

図 II-22-1.　木構造の例

【解説】

1) インデックスとは

インデックスはデータベースのレコードを効率よくアクセスするために用いられる。
インデックスは本の巻末に記載されている索引と同様の役割を持つ。例えば、本に索引が存在しない場合は、目的の語を見つけるためには、全文を最初から読んで探していかなければならない。本にアルファベット順に並べられた索引があれば、語句の一覧に示されたページを見て、そのページから目的の語句をすばやく探し出すことができる。その反面、本の索引を作成するには相応の労力とページ数が必要でコストがかかる。本の内容によっては索引の有用性が低いこともある。

2) インデックスの構造

データベースにおいても本の索引と同様に、インデックスによって、検索時にレコードに対してより効率よくアクセスすることができる。
本の索引と同様の考え方で、コンピュータ上でファイルに索引を持たせる方法を、ISAM（索引表付き順編成ファイル）と呼ぶ。このような構造を持たせたインデックスを利用しても、検索効率は向上するが、データの更新を繰り返すと、性能が低下していく問題がある。
そのため、よりも高いパフォーマンスを得るために、木構造と呼ばれるインデックスを採用する場合も多い。

3) インデックスの特徴

インデックスは DB の性能のみに関係する。インデックスの設定によって、検索やソートのパフォーマンスを上げることができる。言い換えれば、インデックスを設定しなければ、目的のレコードを抽出するために、全レコードを読み込んで検索することが必要になることがある。
主キーやユニークキーはデータベースの行を一意に特定するもので、インデックスを必ずしも内包するわけではなく、異なる概念であるが、キーは行の追加時に重複の確認をする必要があるので、パフォーマンス向上のためにキーを設定するとインデックスも自動的に設定される RDMBS も存在する。重複がないインデックスをユニークインデックスという。

4) インデックスの設定

インデックスは全体的に張り巡らせれば良いというわけではない。インデックスの設定が効果的な場合と、逆効果になりがちな場合がある。効果的な場合とそうでない場合の例を以下に挙げる。
インデックスは、インデックス用にデータ量が増大するためディスク必要量が増大することにも注意が必要である。
また、データの更新が多いテーブルでは、レコードの登録、削除と同時にインデックスを更新する処理が必要になるため慢性的に負荷が増大しパフォーマンスが低下することがある。

* 効果的な場合
  - 検索条件に使われることが多い
  - 結合に使われることが多い
  - ソートやグループ化で使われることが多い
* 非効果的な場合
  - レコード数が少ない
  - カーディナリティが低い（重複した値が多く値の分布が少ない）

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-2. インデックスの種類と特徴	
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-22-2. インデックスの種類と特徴

バイナリツリー(B 木)、索引構成表、ビットマップ・インデックス、逆キー・インデックス、ハッシュ・クラスタ（ハッシュ・インデックス）など、様々なインデックスの種類を紹介し、それぞれの特徴と利点・欠点について解説する。

【学習の要点】

* B 木インデックスは、木構造のページ群にアドレス情報を格納し、インデックス値を比較しながら二分検索法で該当する行の検索を行う。
* ハッシュ・インデックスはハッシュ関数を使って直接アドレス計算を行って該当する行検索を行う。
* ビットマップ・インデックスは 1 行に 1 ビットを割り当てたビットマップデータの相対位置から該当する行検索を行う。

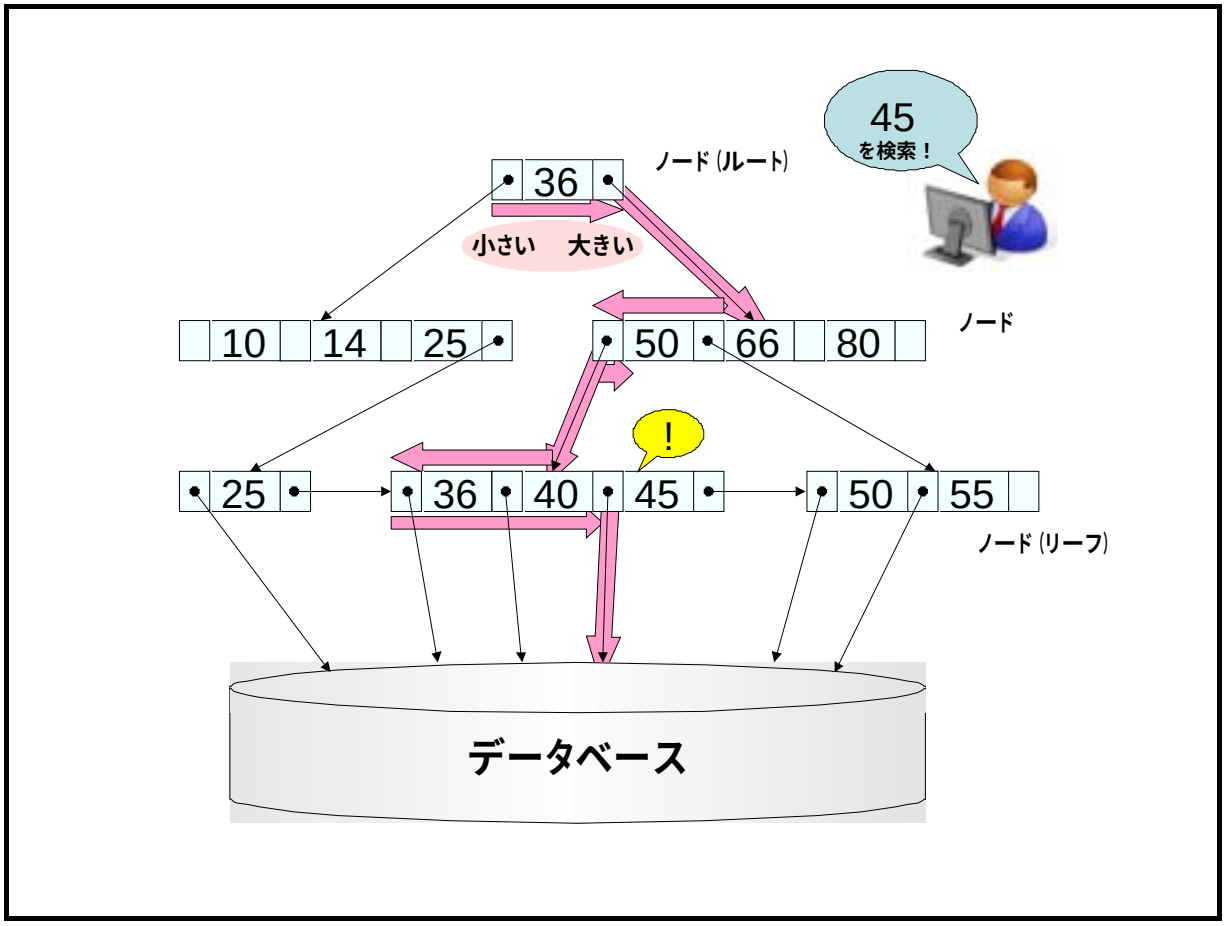

図 II-22-2.　B+木インデックス

【解説】

1) インデックスの種類

インデックスには様々な種類があり、それぞれ利点と欠点を持っている。B+木インデックスが、RDBMS のインデックスで最も多く使われる木構造といわれている。

* 二分木インデックス

木構造は木のように広がっていく構造をもつ。二分木は木構造のなかで最も基本的なもので、ノード（節）の下にある子のノードが最高で2つであるという構造を持つ。

検索方法は、検索したい値をルート（根）から分岐（枝）へたどっていき、目的のキー値がノードのキー値よりも小さければその前の枝へ向かい、大きければその次の枝へ向かう。その次のノードでもその動作を繰り返す。等しいときに完了となる。

データの追加により二分木のバランスが崩れ片側だけに集中するようになると、パフォーマンスが低下するという欠点がある。

* B 木インデックス

B 木（バランス木）は、二分木インデックスの欠点を解消するために考えられたもので、バランス木の名前の通り、データの分岐の先がすべて同一の階層に属した多分木の構造を持つ。ノード内にもソートされた複数のキーが含まれる。

検索方法は、ノード内のキーを検索し、目的のキーが、ノードのキー値よりも小さければその前の枝へ向かい、大きければその次の枝へ向かい、その次のノードでもその動作を繰り返す。リーフノードにキーがあったときに完了となる。

利点は、レコード追加後の検索性能が保たれること、欠点は、レコード削除での検索性能が低下することである。

* B+木インデックス

B 木はノードにもデータを記録するが、B+木ではデータは木の最下層にあるノード（葉ノード、リーフレベルのノード）に格納され、内部ノードにはキーのみが記録される。

また隣のノード同士をポインタで結合してシーケンシャルアクセスの性能向上が図られている。

B 木インデックスと B+木インデックスともに区別なく B 木インデックスと呼ぶこともある。

* ハッシュインデックス

B 木インデックスを使った場合には、目的のレコードのアドレスを得るためには、数回のアクセスが必要になることがある。これをハッシュ関数と呼ばれるものを使って、一回で目的のレコードのアドレスを取得できるようにしたものである。

B 木とは異なり、BETWEEN などの範囲検索には利用できない欠点がある。

* ビットマップインデックス

キーとなる値をビット列で用意するものである。ビット列はキーの値の数だけ用意される。

利点は、通常ではインデックスの設定に不向きとされる、カーディナリティが低い場合に検索性能が高く、インデックスの容量を少なくできる。欠点は、キーの値の数が多い場合にはビット列が増えるため不向きであることと、レコード追加がある場合に性能の低下が大きいことである。

* 逆キーインデックス

キーの値を逆にしてインデックスを格納する方式で、連番（シーケンス番号）などに付加する索引では同じノードに更新が集中するのを避け更新を分散させることができる。範囲検索ができない欠点がある。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-3. データベースの物理設計	
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-22-3. データベースの物理設計

関係データベースを実装する際に必要なデータベースの物理構造について説明する。データベース物理設計の目的や、物理設計時に留意すべきトレードオフ、データとインデックス、ログをどのように配置するか、データベース記憶領域の確保など、物理設計に必要な具体的な手順を示す。

【学習の要点】

* RDB におけるデータベース物理設計の目的は、表内と表間の整合性制約の定義、性能向上のための構造定義、セキュリティの設定を含んでいる。
* データベースの物理設計では、データの非正規化、インデックスの利用、問合せ処理の最適化などを行う。
* システムで必要となる容量や性能を想定して、I/O の分散化や容量設計を実施する。

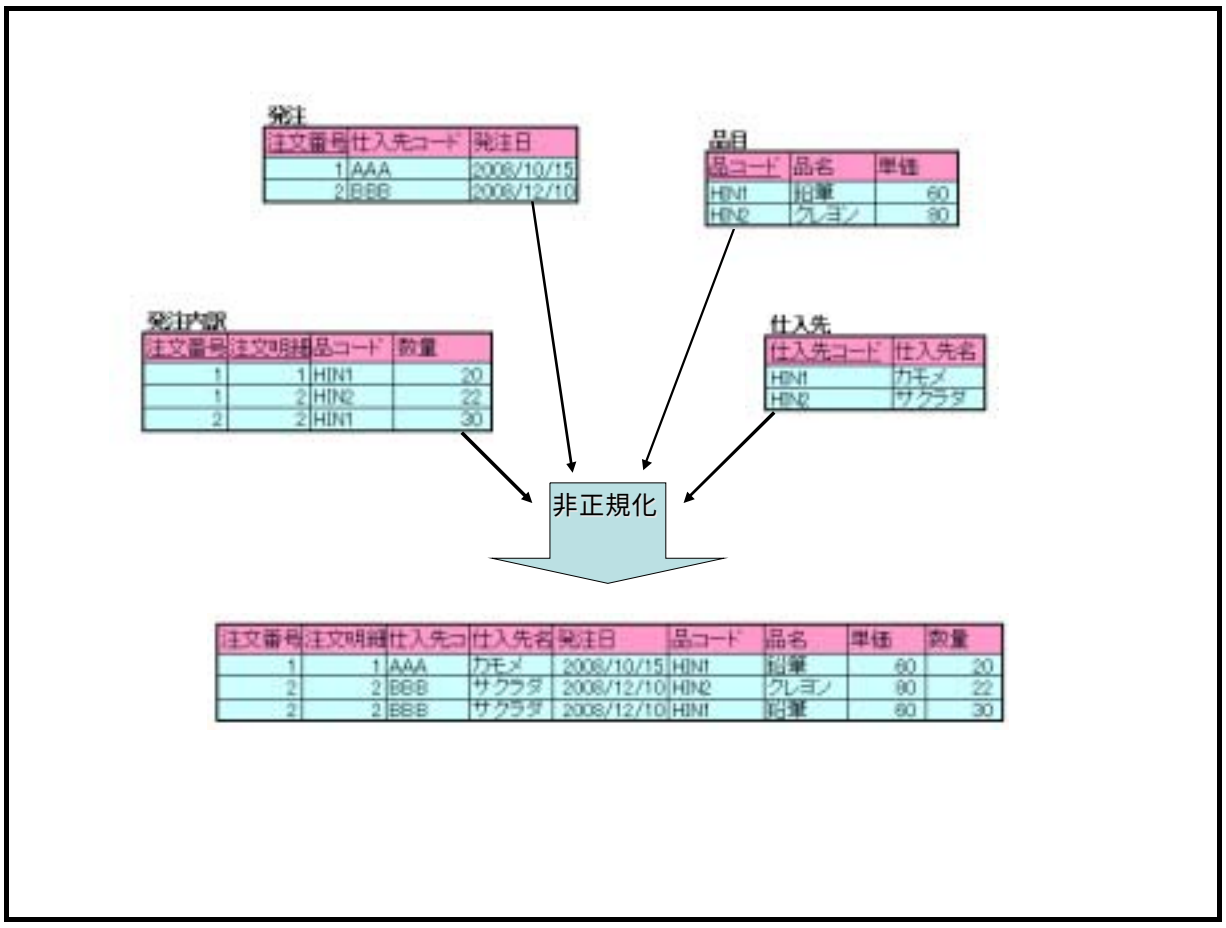

図 II-22-3.　非正規化による性能向上

【解説】

1) 整合性制約

関係データベースでは、整合性制約を定義することにより、整合性制約と異なるデータの更新が禁止され、矛盾しない状態を保つことができる。制約の種類として、以下の項目がある。

* 主キー制約（一意性制約）
  レコードを一意に特定するために、テーブル内の項目での値の重複を禁止する。
* 参照制約
  外部キーによって、関連テーブル間での依存関係を定義する。
* ドメイン制約
  値の制限値の範囲を事前に設定することで、それ以外の値の入力を禁止する。
* NOT NULL 制約（非ナル制約）
  NULL 値の入力を禁止する。NULL 値とは0（ゼロ）や文字列の″″とは異なり、値が入っていない空の状態のことを言う。

2) 性能向上のための構造定義

* インデックス
  インデックスの設定によって、検索やソートのパフォーマンスを上げることができる。
* 非正規化
  正規化を行うと、データの整合性が高く保たれるが、数多いテーブルに分解され、結合条件が多くなりパフォーマンスが低下する。そのため、データベースのパフォーマンス向上のために、非正規化という、正規化後に故意に正規化を冗長化する構造定義の方法が用いられる。
  非正規化は、重複してデータを持つことになるため、整合性が低くなる欠点がある。これは、正規化とはトレードオフの関係になるため、物理設計時に留意すべきである。

3) セキュリティ

データベースは共有して利用することが多い性質上、不正利用を防止するための機能がある。

* アクセス制限管理
  多くの RDBMS では、特定の表の参照のみ可能なユーザを設けるなど、ユーザの属性毎に参照や更新の許可設定をすることにより、きめ細かいセキュリティ管理をすることができる。また、全てのユーザにパスワードを設定することで、なりすましでのログインを防止できる。更に、外部の IP アドレスからの接続を拒否できる設定を持つものもある。
* ログ
  ログは、データベースに対して発行したクエリの監視結果を保存する機能である。不正アクセスの監視やデータベースの障害時復旧の情報として使われる。また、長時間を要したクエリを記録する機能を持つ RDBMS もあり、性能向上のための情報としても役立つ。

4) 容量設計と I/O 分散

運用中に容量不足が発生した場合には、データベースに障害が発生し、停止してしまう場合がある。それを避けるために、あらかじめシステムで必要となるデータ量を想定した上で、十分なものとなるように、容量設計を実施する。
処理性能が必要となる用途では、ディスクの性能を向上させるべく、ファイルを複数のディスクに分散配置することや、ディスクアレイシステムを構築するなど、I/O 分散の工夫を図る。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-4. リレーショナルデータベースと SQL	
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-22-4. リレーショナルデータベースと SQL

関係データベースをアクセスするための基本的な技術である Structured Query Language (SQL)の構造、原理、動作仕様、種類、特徴などについて解説する。また SQL の標準化といった関連情報も紹介する。

【学習の要点】

* SQL は RDB のデータ定義や操作を行うために開発された非手続き型のデータ言語である。
* SQL にはデータ定義、問合せ、データ更新、埋め込み型 SQL などがある。
* ANSI 及び ISO で言語仕様の標準化が行われており、制定年により SQL92、SQL99、SQL2003 などがある。

年代	製品	標準規格	発展内容
1970	System R Oracle		RDBMSの誕生
1980	DB2 InterBase POSTGRES 1	 SQL86 SQL89	商用RDBMSの普及
1990	 MySQL	SQL92 SQL99	SQL標準規格の制定 オープンソースRDBMSの普及
2000	SQLite Firebird	 SQL2003	 XML・オブジェクト指向などへの対応

図 II-22-4.SQL 製品と標準化規格の発展

【解説】

1) SQL とは

SQL とは、Structured Query Language の頭文字を取ったもので、RDB のデータ定義や操作を行うために開発された非手続き型のデータ言語である。RDBMS のほとんどが、データ操作言語として SQL を採用している。

SQL は、E.F.Codd 氏の論文をもとに、IBM の SanJose 研究所で開発されたデータベースシステム System R に実装された SEQUEL という言語を発展させたものといわれている。SQL は、使いやすさに優れていたため、それに倣った RDMBS 製品が多数リリースされるようになった。

SQL は各 RDBMS 間で記述方法の共通点は多いものの、独自拡張などの異なる点も多くあるため、互換性向上を目的として SQL92、SQL99 などの標準化規格が制定されている。オープンソース RDBMS でもこれらの仕様の一部が取り入れられている。

最新の規格は SQL2003 で、デーブル XML のデータに変換する XML 連携等の仕様が追加されている。この仕様の一部を取り入れたオープンソース RDBMS も存在する。

2) SQL の命令の種類

主な SQL 命令にはデータ定義命令（DDL）、データ操作命令（DML）、データ制御命令（DCL）の三種類に分類できる。

* データ定義命令（DDL）

  データベースやテーブルの削除、登録、変更等、データベースの構造や整合性制約を定義する。

  - CREATE　データベースやテーブルを作成する
  - DROP　データベースやテーブルを削除する
  - ALTER　データベースやテーブルを変更する

* データ操作命令（DML）

  データ行の削除、登録、変更等、データの操作を行う。

  - SELECT　テーブルからデータ行を取得する
  - INSERT　テーブルにデータ行を追加する
  - DELETE　テーブルからデータ行を削除する
  - UPDATE　テーブルの既存データ行を更新する

* データ制御命令（DCL）

  データベースのユーザ権限の管理やデータのトランザクション処理を行う。

  - GRANT　権限を付加する
  - REVOKE　権限を剥奪する
  - COMMIT　変更内容を確定する
  - ROLLBACK　変更内容を取消する

3) 埋め込み SQL

プログラミング言語で SQL を用いるための方法の一つとして、親となるプログラム中に SQL（DML）を直接記述して使用できる、埋め込み型 SQL 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	22 RDB に関する知識 II	応用
習得ポイント	II-22-5. SQL による表の作成	
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-22-5. SQL による表の作成

SQL を用いて表を作成する手順を示す。データ型の指定や NULL の取り扱い、主キーと外部キーの指定方法といった表の作成に関する基本的な項目を具体的に説明する。

【学習の要点】

* RDB に SQL で表（テーブル）を作成には「CREATE TABLE」文を使用する。
* 列のデータ型には大きく分けて文字列型、数値型、日時型が指定できる。
* 表の行の一意性制約のために NULL 値を持たない主キーとなる列を作成する必要がある。

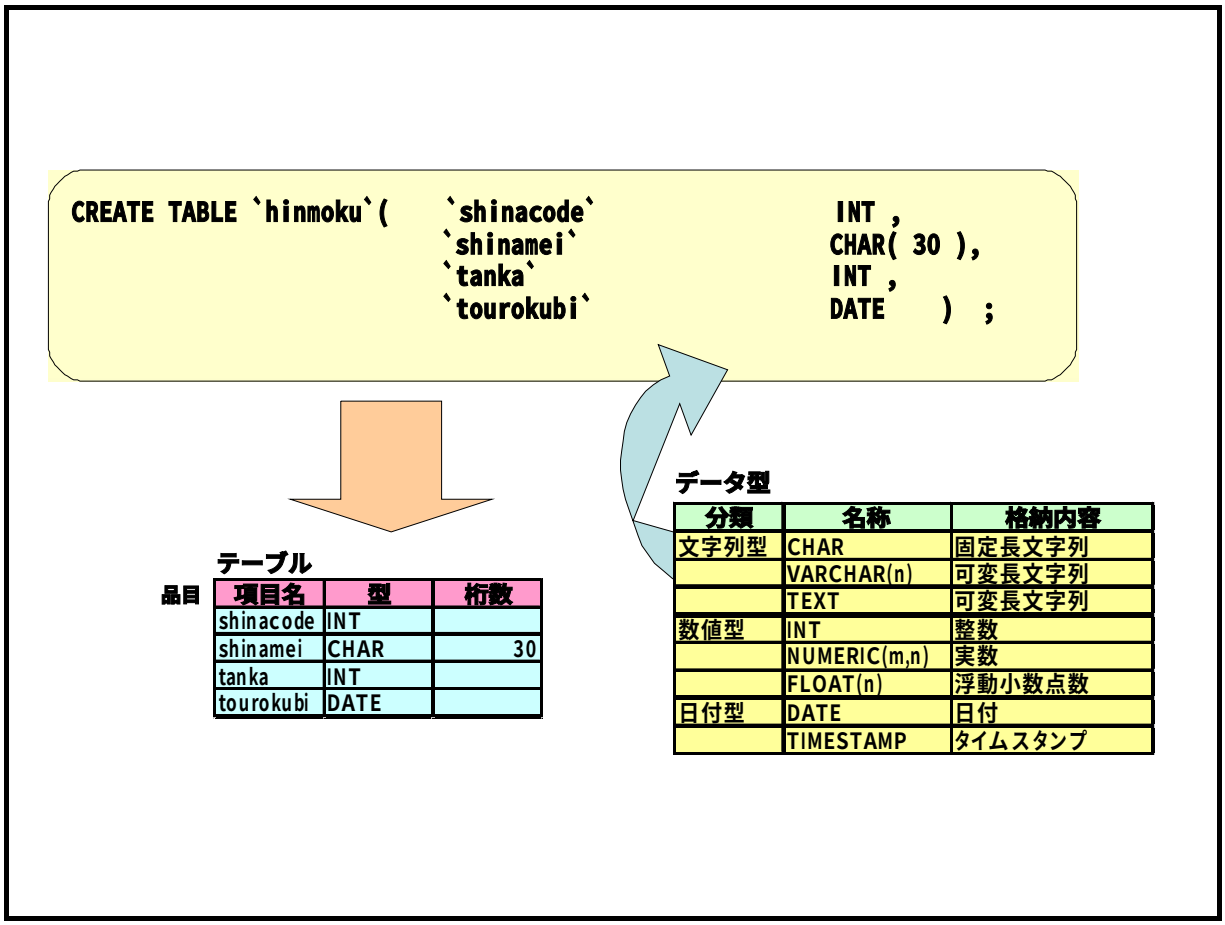

図 II-22-5.　データ型とテーブルの作成

【解説】

1) 表の作成

データベースに格納されるデータは「テーブル」で管理される。SQLを用いてRDBに表（テーブル）を作成するには「CREATE TABLE」文を使用し、以下の構文で記述する。

```
CREATE TABLE テーブル名 ( フィールド名 データ型 [ , フィールド名 データ型] … ) ;
```

記述例：
```
CREATE TABLE `hinmoku` ( `shinacode` INT ,
                         `shinamei` CHAR( 30 ) ,
                         `tanka` INT ,
                         `tourokubi` DATE) ;
```

2) データ型

列のデータ型には大きく分けて文字列型（char、varchar、text）、数値型（int、float、double）、日時型（date、datetime、timestamp）等があり、適切な定義により、データの整合性を高く保つことができる。使用できるデータ型や記述内容はRDMBSによって一部異なる箇所がある。ここではMySQLを例とする。

3) NOT NULL

必須の項目を定義するには、NOT NULL制約のオプションを指定する。テーブルを変更する場合は「ALTER TABLE」を使用する。

記述例：
```
ALTER TABLE `hinmoku` MODIFY COLUMN `shinamei` CHAR( 30 ) NOT NULL ;
```

4) 主キーと外部キー

* 主キー（プライマリキー）

  主キーを作成するには、CREATE TABLEまたはALTER TABLEで「PRIMARY KEY」を指定する。

  記述例：
  ```
  ALTER TABLE `hinmoku` ADD PRIMARY KEY (`shinacode`) ;
  ```

* 外部キー

  外部キーを作成するには、CREATE TABLEまたはALTER TABLEで「FOREIGN KEY」を指定する。なお、MySQLでは外部キーの機能を使用するには、ストレージエンジンにInnoDBを指定する必要がある。外部キーを使用する場合は、あらかじめ既に作成した参照先テーブルのストレージエンジンもInnoDBに変更する。

  記述例：
  ```
  CREATE TABLE `hacchuu` ( `denpyoubangou` INT ,
                           `shinacode` INT ,
                           `suuryou` INT ,
                           `shiiresakimei` CHAR( 30 ) NOT NULL,
                            PRIMARY KEY (`denpyoubangou`) ,
                            FOREIGN KEY (`shinacode`)
                            REFERENCES `hinmoku`(`shinacode`)
                          ) ENGINE = InnoDB;
  ```

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-6. SQL を用いた問い合わせ処理	
対応する コースウェア	第 14 回 データベース運用	

II-22-6. SQL を用いた問い合わせ処理

SQL を用いたデータの検索、問合せ処理の手順を示す。検索の基本となる SELECT 文について説明し、式や数値、文字列、日付といったデータの処理や、WHERE 句による演算、集合関数などについて解説する。

【学習の要点】

* 表から指定した条件を満たす行の集合列を取り出すには「SELECT」文を使用する。
* FROM 句で指定した表から SELECT 句で指定した列名の列だけを取り出し、WHERE 句で指定した検索条件を満たす行を取り出す。
* 指定した列で同じ値を持つ行の集計結果を取り出すには、GROUP BY 句と COUNT、MAX などの集計関数を使用する。

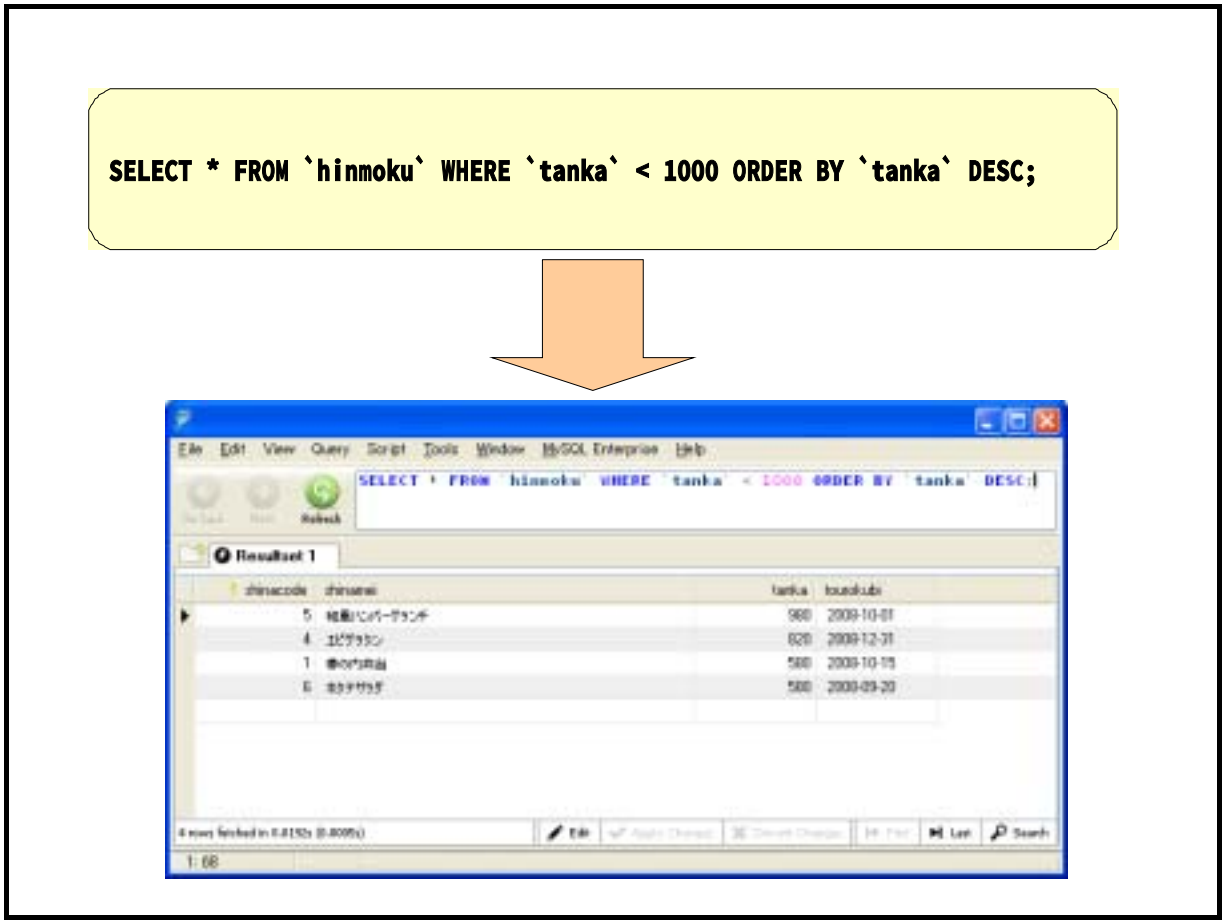

図 II-22-6.　SQL を用いた問い合わせ結果

【解説】

1) データ取得（SELECT）

テーブルのデータを取得するには、「SELECT」文を使用する。SELECT の次に抽出したい参照フィールド名を記述する。ここに参照したいフィールドを列挙して必要なフィールドを取得することができる。「*」は指定したテーブルに存在する全てのフィールドを取得することができる。「FROM」句の次に抽出対象のテーブルを記述する。

記述例：

```
SELECT `shinamei`, `tanka` FROM `hinmoku`;
```

2) 検索条件（WHERE）

検索条件を指定し特定のレコードを取得する場合には、SELECT に続いて「WHERE」句を用いる。条件に用いる演算子として「=」（等しい）、「!=」（等しくない）、「IN」（いずれかが含まれる）などを使用できる。

記述例：

```
SELECT * FROM `hinmoku` WHERE `tanka` > 890;
```

3) 集合関数（集計関数）

集合関数を用いることにより、抽出した結果を集計したり検索条件に該当するものの中から、最大値や平均値等を取得することができる。集合関数として、AVG（平均値）、SUM（合計値）、MAX（最大値）、MIN（最小値）等がある。

記述例：

```
SELECT MAX(`tanka`) FROM `hinmoku` WHERE `tanka` < 1000;
```

4) 文字列関数 日付関数

文字列関数や日付関数を用いることにより日付や文字列を操作することができる。例えば、「DAYOFMONTH」関数は日を得る関数である。RDBMS の種類によって関数の有無や名称が異なる場合が多い点に留意するべきである。

記述例：

```
SELECT DAYOFMONTH(`tourokubi`) FROM `hinmoku`;
```

5) グループ化

「GROUP BY」句を使うと、同フィールド内の値が同一である場合に複数のレコードをグループ化して取得することができる。また、集合関数をあわせて使うことにより、グループ化された項目のグループ内を集計することができる。

記述例：

```
SELECT `tourokubi` , COUNT(`tourokubi`) FROM `hinmoku` GROUP BY `tourokubi`;
```

6) 並べ替え

「ORDER BY」句を使うとレコードを指定したフィールドでソートして取得できる。「ASC」句は昇順でソートし、「DESC」句は降順でソートする。

記述例：

```
SELECT * FROM `hinmoku` WHERE `tanka` < 1000 ORDER BY `tanka` DESC;
```

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-7. 表の結合、複雑な処理、テーブルの更新	
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-22-7. 表の結合、複雑な処理、テーブルの更新

外部結合や自己結合、副問合せといった表の結合(ジョイン)に関する SQL による操作方法や、SQL を用いた複雑な検索･照会の方法、テーブルの更新、削除など、SQL による様々なテーブル操作について説明する。

【学習の要点】

* 表の結合により正規化で分割した表を一つの表として扱う事ができる。
* 表の結合を行う方法としては、他の表と結合する外部結合、同じ表で結合する自己結合、他の SQL 文の実行結果を使用する副問合せがある。
* 表の更新処理を行う SQL 文としては、行の挿入を行う INSERT 文、行の更新を行う UPDATE 文、行の削除を行う DELETE 文がある。

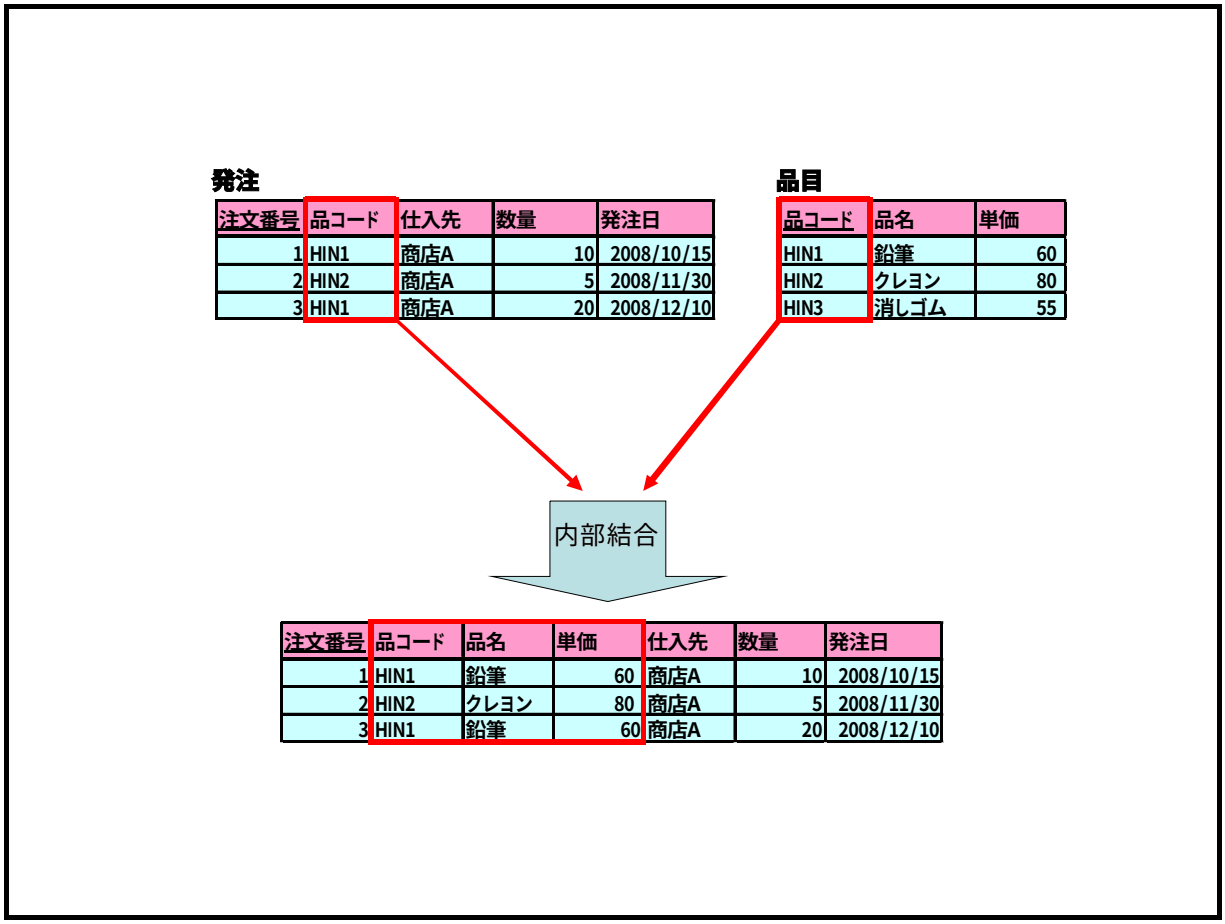

図 II-22-7.　表の結合

【解説】

1) 表の結合

* 外部結合と内部結合

表の結合（JOIN）により、正規化で分割したいくつかの表を一つの表として扱う事ができる。「ON」句の後ろに結合フィールドを指定する。

結合方法には内部結合と外部結合がある。内部結合（INNER JOIN）は結合したフィールドの両方に合致したレコードを抽出する方法である。外部結合（OUTER JOIN）は結合したフィールドの両方に合致するレコードが存在しなかった場合でも抽出する方法で、左外部結合（LEFT OUTER JOIN）は、結合条件の左側のテーブルのレコードを全て抽出する方法である。

記述例：
```
SELECT * FROM `hinmoku` AS A LEFT OUTER JOIN `hacchuu` AS B
    ON A.`shinacode` = B.`shinacode`
     WHERE A.`tanka` < 800 ;
```

* 自己結合

自己結合とはテーブルに別名をつけて、同じテーブル同士を結合することである。下記の例では「,」で結合し、結合条件を WHERE 句で記述しているが、INNER JOIN で記述しても同じ結果になる。同一要素の組み合わせを排除する条件を入れることで、重複行の抽出が防止される。

記述例：
```
SELECT A.`shinamei`,A.`tanka` FROM `hinmoku` AS A, `hinmoku` AS B
    WHERE A.`tanka` = B.`tanka`
      AND A.`shinacode` <> B.`shinacode` ;
```

* 副問い合わせ

副問い合わせ（サブクエリー）とは、SELECT 文中に更に囲うことのできる SELECT 文である。問い合わせ結果の一部として使用できる他、WHERE 句などでの条件式としても使用できる。

記述例：
```
SELECT `shinamei` , (SELECT MAX(`suuryou`) FROM `hacchuu`
    WHERE `shinacode` = A.`shinacode`) AS `suuryou` FROM `hinmoku` AS A;
```

2) データ操作命令

* INSERT（追加）

「INSERT」はテーブルにレコードを追加する。INTO の後ろにテーブル名と項目名を、VALUES の後ろに追加する値を記述する。

記述例：
```
INSERT INTO `hacchuu`
    (`denpyoubangou` ,`shinacode` ,`suuryou` ,`shiiresakimei` )
    VALUES ('1', '1', '100', '仕入商店');
```

* UPDATE（更新）

「UPDATE」はテーブルの既存レコードを更新する。SET の後ろに更新する項目と値を「=」で続けて記述する。WHERE 句の後ろは SELECT 文と同様に更新対象となる条件を指定する。

記述例：
```
UPDATE `hacchuu` SET `shiiresakimei` = '山田商店'
    WHERE `denpyoubangou` = 1 ;
```

* DELETE（削除）

「DELETE」はテーブルのレコードを削除する。FROM の後ろにテーブル名を記述する。

記述例：
```
DELETE FROM `hacchuu`
    WHERE `denpyoubangou` = 2 ;
```

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-8. オープンソース RDBMS の種類と特徴	
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-22-8. オープンソース RDBMS の種類と特徴

PostgreSQL、Firebird、MySQL といった代表的なオープンソース RDBMS について、開発の歴史や主たる機能、特徴やライセンスなどについて解説する。さらに商用 RDBMS との比較やオープンソース RDBMS を利用する理由についても説明する。

【学習の要点】

* 代表的なオープンソース RDBMS としては、企業が開発しオープンソース化した MySQL、大学の研究プロジェクトで開発された PostgreSQL、商用 RDBMS から分岐した Firebird 等がある。
* オープンソース RDBMS は機能、性能共に商用 RDBMS と遜色ないものとなってきており、企業システムで使えるレベルになっている。

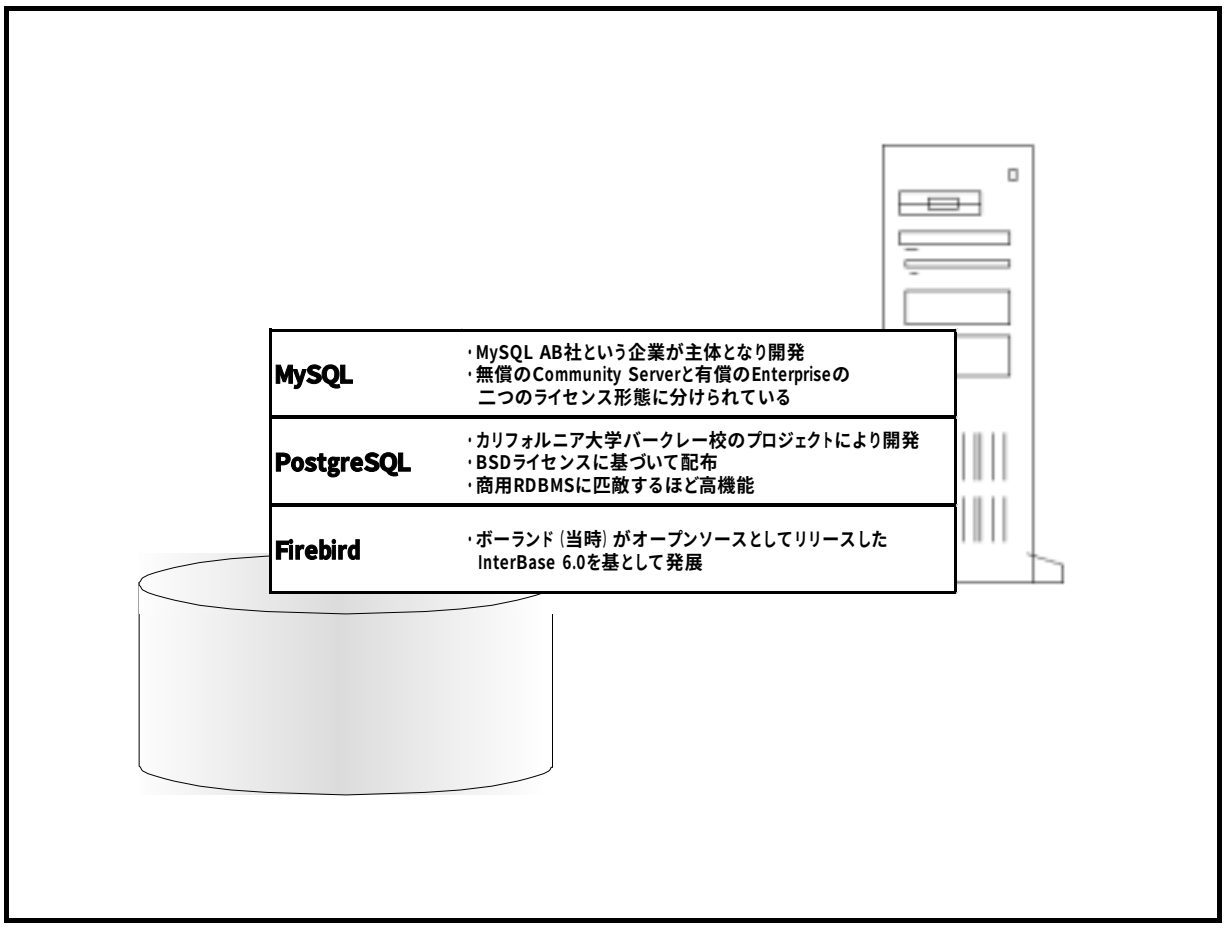

図 II-22-8.　オープンソース RDBMS の種類と特徴

【解説】

1) オープンソース RDBMS の普及

昨今のオープンソース RDBMS は、商用 RDBMS と比較しても機能、性能共に遜色ないものとなっており、オープンソース OS や Web サーバの普及に伴って広く利用されている。
オープンソース RDBMS を利用する理由となるのは、通常のオープンソースソフトウェアと同様に改変可能、ライセンス料金不要という点である。特に注目されるのは、ライセンス料金不要の側面である。Web システムでは利用ユーザ数が予測しづらく、コストも抑制されがちである。かかる利用目的において、商用 RDBMS はコストが高く、ユーザ数でライセンス料金が変動する形式のものもあり、採用しにくい面があった。そこで、代用としてオープンソース RDBMS が急速に普及していった。

2) オープンソース RDBMS の種類

オープンソース RDBMS は様々な製品が登場している。主たるものとして以下の 3 製品がある。

* PostgreSQL
PostgreSQL は、カリフォルニア大学バークレー校のプロジェクトにより開発され、その後インターネットコミュニティに引き継がれ開発が継続されている RDBMS である。
PostgreSQL は商用 RDBMS に匹敵するほど高機能な RDBMS で、企業システム用途での利用を意識して機能強化が進められており、比較的早期にストアドファンクションやサブクエリーをサポートするなど機能が充実している。
また、プログラム改変後のソースコードの公開の義務がなく、商用利用でも制限なく使用可とする、BSD ライセンスに基づいて配布されており、ライセンス上の懸念が少なく利用できる。そのため、基幹業務システムで採用される場合も多い。

* MySQL
MySQL はコミュニティ主体ではなく MySQL AB 社という企業が主体となり開発した RDBMS である。2008 年にサン・マイクロシステムズが買収した。
MySQL は、無償の Community Server と有償の Enterprise Server という二つのライセンス形態に分けられている。両者ともオープンソースソフトウェアであるが、Enterprise Server のバイナリプログラムは再配布はできない。Enterprise Server は安定性を重視し、Community Server は機能実装を重視する方針で開発が進められている。
MySQL は従来から処理速度が高速であることで知られていたが、機能面でも他の RDBMS と遜色がなくなってきている。
PostgreSQL と同様に、各種開発言語用の API が用意されており、Web システムのバックエンド RDBMS として多くの実績がある。また、ブログなどのオープンソース CMS では、対応 DB として MySQL が採用されている場合が多く、個人向けレンタルサーバで搭載される DB としての普及も進んでいるため、個人ユースとしても広く利用されている。

* Firebird
Firebird は、商用の RDBMS「InterBase」を起源として約 20 年の歴史を持つ RDBMS で、ボーランド（当時）がオープンソースとしてリリースした InterBase 6.0 を基として、開発が続けられている。
Firebird プロジェクトには過去に InterBase の開発を行っていたメンバーが多く参加しており、商用製品に遜色ないプロジェクト体制を有している。
ストアドファンクション、オンラインバックアップなど企業システムを意識した機能を備え、また、商用の InterBase から派生した経緯から、InterBase ユーザの代替としても広く利用されている。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-9. RDB を用いた企業 DB 設計の例	
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-22-9. RDB を用いた企業 DB 設計の例

業務分析としてエンティティを洗い出し ER 図を作成したのち、スキーマとインデクスの設計、分散データベースの設計という企業基幹データベースの設計手順を示す。また物理設計として性能設計や処理効率化検討といった項目についても解説する。

【学習の要点】

* データベース作成を行う業務のエンティティの洗い出しを行い、ER 図の作成を行い概念データモデルの設計方法を習得する。
* 概念データモデルからスキーマやインデックスの設計を行い、論理データモデルの設計方法を習得する。
* 論理設計の結果からレコードの構造、順番、アクセスパスなどの設計を行い、物理設計の方法を習得する。

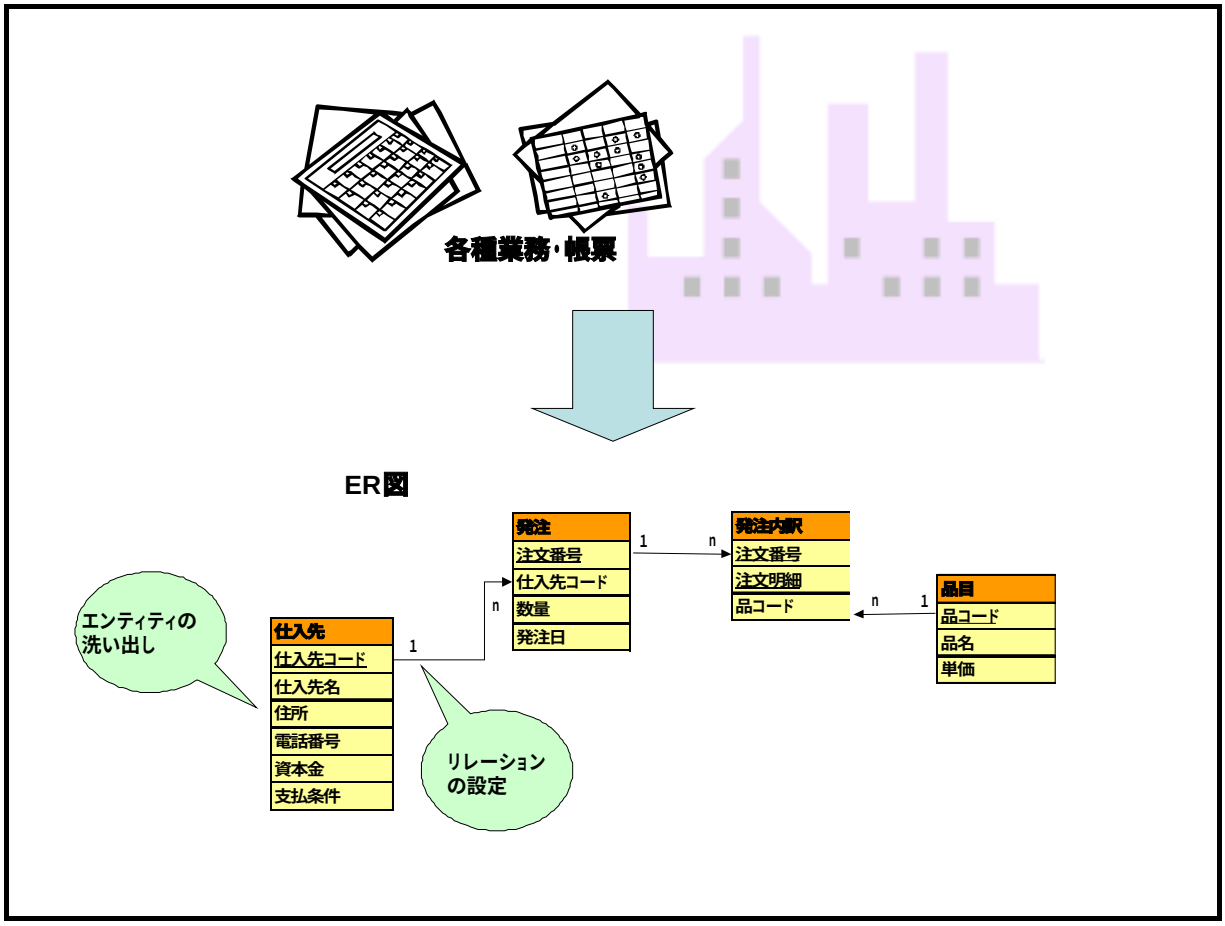

図 II-22-9.　ER 図

【解説】

1) データモデリング

データベースシステムを構築する場合には、現実にある様々な情報から、システムの要件に基づき、構築に必要なデータ項目を探し出して、項目間の関係性をまとめ上げ、より無駄なく集合化して、論理的かつ意味的に正しいデータモデルを組み立てることが必要になる。データモデルを組み立てる行為のことをデータモデリングという。

データモデルの持つ機能としては、データ定義、データ操作、整合性維持の3つがある。

データモデルは、一般に次の3段階を経て完成し、段階を経る毎により詳細化していく手順となる。

* 概念データモデル

  システムの要件をコンピュータでの処理を前提に分析し識別し、ER 図などで定義した、特定の DBMS に依存しない大まかなデータモデルのことを「概念データモデル」という。

* 論理データモデル

  概念データモデルに、システムで必要となるスキーマを作成して、コンピュータに実装可能な形に変換したデータモデルのことを「論理データモデル」という。

  スキーマは、表と表内のフィールド、フィールドや表の関係などのデータベースの構造の定義のことである。

  論理データモデルは、より詳細な ER 図などで作成され、DBMS の特性を加味して、それに依存したデータモデルになることが多い。

* 物理データモデル

  実装を意識して、データ型や索引までを定義し、CREATE TABLE 文などの、DBMS でサポートされている形式で記述可能となるレベルのデータモデルを「物理データモデル」という。

  物理データモデルでのデータ定義は、実テーブルを設計することであり、整合性制約の定義、性能向上のための構造定義（インデックス、パラメータ）など、RDB の物理的な内部構造を決定する。

  アクセスパスは、データの取得ルートのことをいう。インデックスや抽出条件、結合条件などを見直すことで、よりパフォーマンスの高いアクセスパスとなることを目指す。

2) 分散データベースの設計

異なる場所にデータベースを配置しネットワークで接続されたデータベースシステムのことを分散データベースという。この方法は、ネットワークの負荷低減のために有用である。分散データベースには、非同期型更新や2相コミットなどの方法がある。

* 非同期型更新

  分散先のサイトに更新データを適宜送信してデータの同期を取る方法である。同じデータを複数のサイトで同時に更新する必要がある場合は、更新内容に競合が生じる可能性があるため、利用には注意が必要である。

* 2 相コミット（2 フェーズコミット）

  主サイトから分散した処理先のサイトの更新処理がコミット可能か確認し、コミット処理待ちの指示を出すフェーズ（第1フェーズ）と、それぞれのサイトが更新をコミットする指示を出すフェーズ（第2フェーズ）に分けて処理がおこなわれ、第1フェーズの結果を確認し、コミットするかロールバックするかを決定して、処理先のサイトへ指示を出す。比較的競合を防止できるが、ネットワークの負荷は増大しがちになる。

スキル区分	OSS モデルカリキュラムの科目	レベル
RDB 分野	22 RDB に関する知識 II	応用
習得ポイント	II-22-10. RDB を用いた DB アプリケーションの例	
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-22-10. RDB を用いた DB アプリケーションの例

テーブルの作成とデータのロードといった具体的な手順を示し、さらに検索結果の表示やビューの設定といった業務帳票や画面設計の手順、パフォーマンスの検討など、実際の DB アプリケーション開発に必要な項目を解説する。

【学習の要点】

* 企業で利用するために設計されたデータベースを SQL にて作成し、データの登録を行う。
* 必要なデータの検索を行う SQL を作成し、結果を表示させる。
* インデックスの作成等により作成したデータベースのパフォーマンスの改善を行う。

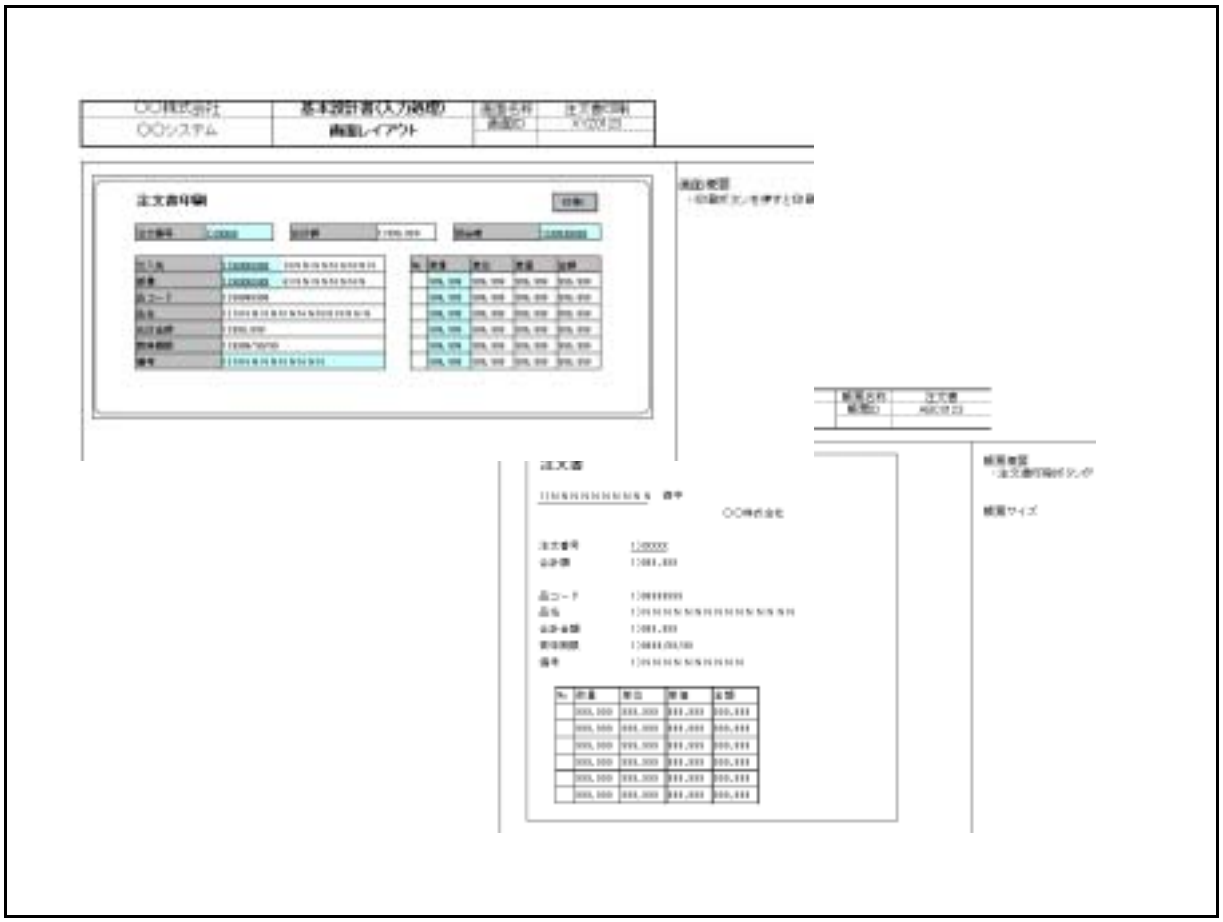

図 II-22-10.　画面･帳票設計の例

【解説】

1) データのロード

企業でデータベースを利用するための移行データなどの投入には、CSV データを利用することが広く行われている。

作成したテーブルにデータをロードするには、プログラムを開発して、データを INSERT 文に変換する方法の他、RDBMS にロード用のユーティリティが用意されている場合がある。例えば、MySQL では、以下の構文で CSV ファイルをテーブルにロードすることができる。

```
LOAD DATA INFILE ‘[ファイル名]’ INTO TABLE ` [テーブル名] `
FIELDS TERMINATED BY ",(セパレーター)"
OPTIONALLY ENCLOSED BY '"（囲い文字）';
```

2) ビューの設定

ビューは、既存の一つまたは複数のテーブルから抽出した結果を、仮想的な名称をつけて表示できるテーブルである。

ビューは利用側からは単一のテーブルのように見えるが、ビュー内で結合取得するなど複雑なクエリとなっている場合は、パフォーマンスが低下する。

* ビューの利用目的
  - テーブルをカプセル化することで、同じ検索結果を得たいプログラムが複数存在した場合に、同じようなSQLが乱立しプログラム開発上の不具合を誘発することを防止して、開発効率を向上させる。
  - テーブル全体を取得させないなどの管理の厳格化により、データのセキュリティを向上させる。
* 記述例

```
CREATE VIEW hacchuu_shinamei AS
 ( SELECT B.suuryou , B.shinacode , A.shinamei FROM `hinmoku` AS A INNER JOIN
`hacchuu` AS B ON A.`shinacode` = B.`shinacode` WHERE A.`tanka` < 800 )
```

3) 業務帳票や画面設計の手順

帳票や画面の設計は、システム作成依頼先から要件確認を行い、業務フローの作成によって、システム化対象となった機能の洗い出し（画面一覧、帳票一覧）をする。

そして、各機能から必要となるエンティティを確認しながら、データモデルを作成していく。

データモデルの作成においては、複数の画面帳票が相互に関連しているものもあるので、機能別にテーブル関連図や CRUD 図（各々の機能が、どのテーブルに対して、データの参照、作成、更新、削除を行うかを示した表）などを作成する。

この前後に、必要と判断された機能について、「帳票レイアウト」と「画面レイアウト」の作成を実施する。これらに機能関連表、項目一覧、機能概要説明などを付記したものが一般に「基本設計書」となる。

4) パフォーマンスの検討

要件確認においては、登録データの件数もできるだけヒアリングし、パフォーマンス上、ボトルネックとなりそうな箇所を検討しておく。これらの箇所について、SQL のチューニングやデータベースの非正規化、インデックスの設定などのパフォーマンス対策の実施を検討する。