第3章

データベースの設計

必要性／ERモデル／正規形／ERモデルから関係モデルへの変換／
ボトムアップ設計／トップダウン設計／
データベースの論理設計／関係モデルの変換／
リレーションの設計／属性の設計／インデックスの設計／
データベースの物理設計／ディスク容量の見積もり

ここでは，データベース，とくに，関係データベースの設計方法について述べます．まず，データベース設計の必要性について述べ，次に，データベース設計の概要を示します．そして，データベース設計で使用するデータモデルや理論について述べます．その後，データベース設計の各フェーズについて述べます．

3.1 データベース設計の必要性

最適でないリレーションスキーマでは，更新不整合（update anomaly）と呼ばれる不整合が生じます．更新不整合には，①修正不整合（modification anomaly），②挿入不整合（insertion anomaly），ならびに，③削除不整合（deletion anomaly），の3種類があります．

ここでは，**図3.1**のリレーションを例にしてこれらの不整合について説明します．**図3.1**は，学生とその受講科目を管理するリレーションの例です．属性として，「学籍番号」，「学生名」，「学年」，「受講科目名」，ならびに，「担当教員名」を持ちます．主キーは「学籍番号」と「受講科目名」の組です．

図3.1　学生リレーション（不整合を生じる例）

学生

学籍番号	学生名	学年	受講科目名	担当教員名
2001001	青山　一郎	1	データベース入門	宮本　武蔵
2001001	青山　一郎	1	ソフトウェア工学	佐々木　小次郎
2000023	鈴木　花子	2	データベース入門	宮本　武蔵
2000023	鈴木　花子	2	コンパイラ	武蔵坊　弁慶
1999065	山田　太郎	3	メディア工学	源　義経

(1) 修正不整合（modification anomaly）

タプルの修正時に生じる不整合です．同じ情報を表すフィールドが多数ある場合，その情報を修正するにはそれらのすべてのフィールドを修正しなければなりません．もし一つでも修正を見逃すと，存在しないはずの情報が残ってしまいます．例えば，**図3.1**のリレーションにおいて，「データベース入門」の担当が「平　将門」に変わる場合，該当するすべての担当（「宮本　武蔵」）を「平　将門」に修正しなければなりません．

(2) 挿入不整合（insertion anomaly）

タプルの挿入ができないという不整合です．例えば，「情報工学」という授業を新設する場合，受講する学生がいない場合はその科目に相当するタプルを挿入することはできません．なぜならば，主キーの一部である学籍番号に値を格納できないからです．つまり，例えば，（NULL, NULL, NULL,“情報工学”,“佐々木　小次郎”）というタプルを挿入することができないのです．

(3) 削除不整合（deletion anomaly）

タプルの削除に伴い，削除したい情報ではない情報が削除されてしまうという不整合です．例えば，学籍番号が「1999065」の学生が退学した場合，この学生だけが「メディア工学」という授業を受講していたならば，「メディア工学」の担当教員は「源　義経」であるという「メディア工学」の科目に関する情報も同時に削除されてしまうのです．

図3.1に示したリレーションの場合，学生に関する情報，履習に関する情報と科目に関する情報が一つのリレーション中に存在しているのが不整合の原因です．したがって，不整合を回避するためには，**図3.1**のリレーションは**図3.2**に示すような三つのリレーション（「学生」,「受講」と「科目」）に分解すべきです．このリレーションでは，上記の3種類の不整合は生じません．

このような分解を適確に行うためにデータベース設計は不可欠です．

図3.2　学生リレーション，受講リレーションと科目リレーション

学生

学籍番号	学生名	学年
2001001	青山　一郎	1
2000023	鈴木　花子	2
1999065	山田　太郎	3

受講

学籍番号	受講科目名
2001001	データベース入門
2001001	ソフトウエア工学
2000023	データベース入門
2000023	コンパイラ
1999065	メディア工学

科目

科目名	担当教員名
データベース入門	宮本 武蔵
ソフトウエア工学	佐々木 小次郎
コンパイラ	武蔵坊 弁慶
メディア工学	源 義経

以上述べた点は関係データベースの論理的な面についての問題ですが，このほかにも，データを格納する際の表現形式の違いの問題や，異なる属性名であるが同じ情報を格納している場合や，同じ属性名で異なる情報を格納している場合など，システム全体として整合性のとれたデータベースとする必要があります．

また，データベース設計は，それ単独で行われることはあまりなく，通常，情報システム設計の一部として行われます．データベース設計により，情報システムで扱う情報が正確かつ詳細に洗い出されます．したがって，概念的な面でもデータベース設計は重要です．

さらに，データベースの物理的な面から，データベース処理からみたリレーションの構造の妥当性の検討，検索の高速化を図るためのインデックスの付与，データ量の見積もり，ログ情報の取得頻度等の検討が必要です．実際にデータベースを運用するためにはさまざまな点からの検討が必要なのです．

データベースは一度構築し運用を始めると，後から構造等を変更するのは非常に困難です．また，システムが不安定になる可能性が高くなります．さらに，データベースは二次記憶装置に格納されて長年にわたり存在し続けるので，他のアプリケーションのように，バージョンアップで今までのものを廃棄して新しいものにするということが困難です．データベースは，計算機ハードウエアよりも長く存在し続けることが多いので，初期設計を誤ると取り返しのつかないことになる可能性が高いのです．また，データの更新に伴って，データベースの内部構造自体も変化してゆきます．これは，どのような更新がどのような順序で生起するかに依存するため予測は困難です．このようなデータベースを長年にわたり安定させて運用するには，最初に十分な設計を行っておく必要があります．

3.2 データベース設計の概要

データベース設計は，通常，概念設計，論理設計，ならびに，物理設計の3段階から構成されます．データベース設計の手順を**図3.3**に示します．

データベースの概念設計では，概念スキーマを決定します．概念スキーマは，対象とする業務におけるデータとデータ間の関連を記述したものであり，使用するデータベースやデータベース管理システムに関係なく，対象とする業務におけるデータが本来どうあるべきかを記述したものです．概念スキーマの記述には，実体・関連モデル（ERモデル）に代表されるデータモデルが使用されます．

データベースの論理設計では，概念スキーマをもとにして論理スキーマを決定します．論理スキーマは，概念スキーマで表現されたデータとデータ間の関連を，実際に利用するデータベースの世界の表現として記述したものです．論理スキーマは使用するデータベースによって異なります．例えば，関係データベースとするのか，オブジェクト・リレーショナルデータベースとするのかで，設計結果の論理スキーマは全く