コンピュータ概論B
ー ソフトウェアを中心に ー

#09 データベース / RDBMS

京都産業大学
安田豊

データベースとは

- 教科書 pp.103-
- 外見
  - データを決まった形式（フォーマット）で整理し蓄積したもの
  - レコード (Record) の存在
  - オブジェクト指向データベースのように決まったデータ型を用意しないタイプもある（例外は常にある）

データベースの目的

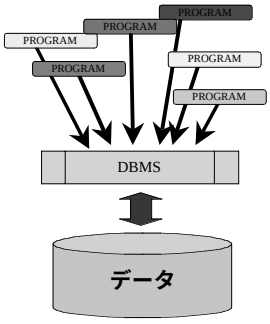
- 目的
  - 入力・更新
  - 高速な検索、再利用
- 一元管理
  - あちこちにあるデータを一元管理したい
  - 多数ユーザに最新で正しいデータを提供する
- 共有
  - 多くのユーザで参照、更新したい
  - 一元管理と常にセットで現れる問題

一元管理と共有の間で

- データと処理プログラムの独立性の確保
  - 項目が一つ増えたらプログラム全部修正？
- 整合性の確保
  - 複数箇所に同じ値がある
  - 学費データと履修データの両方に在籍情報がある
  - 多数利用者の同時更新から生まれる矛盾抑制
- データの保全
  - プログラムの中断やシステムダウンからの保護
- アクセス制限
  - 複数利用者が前提
- 簡易な問い合わせ言語機能の利用

DBMS

- 前出の機能をどうやって実現するか？
  - データをプログラムが直接扱えないようにする
  - DBMS の登場
  - Database Management System
  - 全ての作業はDBMSというプログラムを経由する
  - 独立性、整合性、保全、アクセス制限



独立性・整合性

- 独立性
  - データのフォーマットはDBMSに定義・管理
  - 処理プログラムはその定義を引用して動作する
- 整合性
  - 処理プログラムの手続きはすべてDBMSに対する指示として実行される
  - DBMS は実行時に整合性管理、アクセス制限管理
  - 順序処理、排他制御も行う

排他制御

- あるデータをカウントアップする
  - 「読んで」から「足し」て「書く」
  - 複数の処理リクエストが来た場合、正しくカウントアップできない
  - 「読・足・書・読・足・書」なら良いが
  - 「読・読・足・足・書・書」なら？
- 排他制御
  - 「この処理が終わるまで、この資源はロック」
  - デッドロックに注意
  - DBMSではロールバックの必要性につながる
  - DBMSに限らず多用されている
  - 計算機科学・技術の重要な概念の一つ

データ保全

- 処理プログラムの中断
  - バグ、オペレーションミス、システムダウン
- 一貫性の保持
  - 更新処理途中での停止
  - 会員資格更新時に会員番号 100 までで止まった
  - 会員マスターは更新したが支払いデータは未更新
  - 作業しなかったか、完了したかのどちらかに確定しないといけない
- トランザクションとロールバック

トランザクション

- データの整合性を保つために必要な最小の一連処理
  - その途中で終了した場合、データに矛盾が生じる
  - 大量データの削除処理などもそう
  - プログラマにしかトランザクションの存在が分からないケースもある
  - 明示的なトランザクションもある
- ロールバック
  - トランザクションを完了できなかった場合、トランザクション前の状態に巻き戻す

バックアップ・レストア

- DBMS自体の不意の中断
  - バグ、オペレーションミス、システムダウン
  - それでも一貫性を保持しなければならない
  - あるポイントでバックアップを取る
  - そこからは記録された更新情報を元に再現
- ログ管理
  - 更新記録(Log)をトランザクション単位で記録
  - レストアでは最終バックアップからログを頼りに更新作業を再現
  - ログがいっぱいになったら通常はDBMS自体が自動停止してデータを保護する。

DBMS のまとめ

- データベースが守るべき要件
  - データ独立、整合性管理、データ保全、アクセス管理
  - 多くはマルチプログラミングからの保護
  - 排他制御
  - バックアップ・レストア、ログ管理
- とにかくデータの一貫性を保持すること
  - そのためにDBMSという「仲介人」を入れる

関係データベース

- 特徴
  - 数学的に完成したモデルがあった
  - Codd (1970, IBM) が理論的モデルを提唱
  - データを表組みで表現
  - 表と表の関係処理を集合演算モデルで定義
- RDB の完成
  - 1973 の SystemR (IBM), Ingress (UCB バークレー校)
  - 1979 Oracle
  - SQL の発明 (1986, ANSI 標準となる)
  - 現在もっとも市場で多く使われているタイプ

RDBMS

- RDB の DBMS
  - テーブル、項目の管理
  - ログ管理
  - アクセス制限
- 特徴
  - SQL 問い合わせ言語によるアクセス
  - 簡易問い合わせシステムがついている
  - プログラムからもSQLでアクセス

RDB における表

- データは表形式
  - 行と列による表現
  - 多様なデータを表と項目の関係で記述
- 学生情報で一人分
  - 学生レコード一行
  - 学費レコード一行
  - 履修登録レコード複数行

GNO	NAME	GAKUBU	GAKUNEN
473088	榎田裕一郎	E	2
859674	明日田勇作	B	1

GNO	GAKUHI	SIHARAI
473088	1223000	643000
859674	1200000	1200000

GNO	KAMOKU	UNIT
473088	科学と哲学	4
473088	基礎演習	2
473088	人生航路	4
859674	科学と哲学	4

RDBにおける演算

- 集合と見なして演算
- 部分集合
  - GNOが473088の行を抜く
  - GNOとGAKUBUだけを取り出す

GNO	NAME	GAKUBU	GAKUNEN
473088	榎田裕一郎	E	2
859674	明日田勇作	B	1

GNO	NAME	GAKUBU	GAKUNEN
473088	榎田裕一郎	E	2

GNO	GAKUBU
473088	E
859674	B

RDBにおける演算

- 足す（集合和）
- 同じ項目名の列をそのままくわえる

GNO	NAME	GAKUBU	GAKUNEN
473088	榎田裕一郎	E	2
859674	明日田勇作	B	1

＋

GNO	NAME	GAKUBU	GAKUNEN
785412	暁三四郎	E	1
325698	空手一大	J	3

GNO	NAME	GAKUBU	GAKUNEN
473088	榎田裕一郎	E	2
859674	明日田勇作	B	1
785412	暁三四郎	E	1
325698	空手一大	J	3

RDBにおける演算

- 表どうしを結ぶ
  - 共通の項目(key)で付き合わせ
  - JOIN
  - キーによる突き合わせ

GNO	NAME	GAKUBU	GAKUNEN
473088	榎田裕一郎	E	2
859674	明日田勇作	B	1

GNO	GAKUHI	SIHARAI
473088	1223000	643000
859674	1200000	1200000

GNO	NAME	GAKUBU	GAKUNEN	GAKUHI	SIHARAI
473088	榎田裕一郎	E	2	1223000	643000
859674	明日田勇作	B	1	1200000	1200000

SQL

- Full Spec 無し(略語ではない)
  - 元はあったが今は SQL として仕様化
- 集合演算をプログラミング言語風に簡略化
  - SELECT 一つで殆どの処理を行う
- 選択
  - SELECT * FROM GAKUSEI
    WHERE GAKUBU=“E”
  - SELECT * FROM GAKUHI
    WHERE SIHARAI > 600000

SQL

- 選択（項目抜きだし）
  - SELECT GNO, GAKUBU FROM GAKUSEI
- 突き合わせ
  - SELECT * FROM GAKUSEI, GAKUHI
    WHERE GAKUSEI.GNO = GAKUHI.GNO
- カウント他
  - SELECT COUNT(*) FROM GAKUSEI
    WHERE GAKUBU=“E”
  - SELECT GNO, GAKUHI-SIHARAI
    FROM GAKUSEI, GAKUHI
    WHERE GAKUSEI.GNO = GAKUHI.GNO

関係データベース

- 利点
  - 柔軟、プログラムとデータが独立
  - SQL という問い合わせ言語の便利さ
  - 数学的完全性
- 欠点
  - 概して低速
  - データ格納効率が高くならない
- 動かしながら開発したり将来変更が多そうなシステムに向く
- 現在もっとも多く市場で使われているタイプである
  - 欠点をマシン能力でカバーするという考え方
  - 現代的なソフトとハードの関係の典型例