Chap 1 The Worlds of DataBase Systems

◉ DB System의 진화

* DB : DBMS에서 관리하려는 데이터 수집, DBMS를 이용하여 데이터를 관리 및 저장

* DBMS (DB Management System) (= database system)
  : 대용량 데이터를 효율적으로 생성 및 관리하고 데이터를 장시간에 걸쳐 안전하게 유지할수 있도록 하는 전용 SW
  - 데이터 모델 : DDL을 이용하여 데이터 (스키마)의 논리적 구조를 지정
  - 고수준의 질의언어 (High level query language)
  - 영구적인 스토리지 (저장소) 시스템 : 대용량 데이터 (저장된 데이터가 부족한 경우 x), 효율적인 접근 (돈을 빨리 얻을)
  - 고립성 : 많은 사용자로 부터 동시에 데이터에 접근할때 이를 제어
  - 원자성 : 모두 실행 되거나 전혀 실행되지 않아야 함
  - 내구성 : 데이터의 복구가 유용하다.

* ex) DBMS가 없다면?
  ⇒ 데이터를 하나 또는 여러개 파일에 일일히 저장하여 관리할 것이다.
  이때 생기는 문제점
  - 부적절한 튜플레이아웃 : 갱신의 문제
  - 검색 비용 문제
  - 질의처리와 전체탐색문제
  - 동시성 제어 없음 : 여러 사용자가 한파일을 동시에 수정하는 경우 원치하지 않은 결과가 나올수 있다.
  - 메인메모리에는 잘 쓰는 정보를 두고 사용해야 부담이 적다.
  - 복구없음 : 컴퓨터가 고장 나거나 시스템이 망가지면 복구정책이 없다.
  - 스키마 (논리적 구조)의 부재
  - 고수준 언어부재 : C 언어를 해야지만 책을 빌릴수 있는경우
  - 무결성 문제 : 데이터가 있어야 하는데 모종이 이유로 없거나 잘못 데이터 기입하는 경우
  - 보안문제
  - 파일은 응용프로그램에 따라 다르다 : 호환의 문제로 엑셀, 한글 등이 열리지 않거나 구분기호가 Tab, # 등으로 다르거나, 수정을 사람마다 다르게 하는 경우 등의 문제

* 초기 DBMS

1960년대 쯤 개발되어 파일시스템이 진화하였다.

고수준의 언어가 없었다.

* 관계형 모델 (Relational database systems) ⇒ 관계 DBMS

관계 모델 : table 이나 관계로 데이터가 보인다. ⇒ 릴레이션

고수준의 언어 사용 ⇒ 질의를 쉽게 위해서

* 점점 더 작은 시스템

- 원래 DBMS 는 대형 컴퓨터에서 실행되는 크고 비싼 SW 시스템
- 오늘날 DBMS은 많은 곳에서 사용가능하다. (기가 바이트 이상의 용량, 무료소프트웨어 제공, 개인컴퓨터 등)

* 점점 더 커지는 시스템.

- 테라바이트를 초과하지 않는 경우 ⇒ 페타, 엑타, 제타 바이트 사용.

(ex) 위성데이터, 사진 데이터 사이트, 동영상 데이터 사이트 등

* 정보 통합

- 여러 DB를 보유한 대규모조직 독립적으로 구축됨. ⇒ 다른 DBMS 사용하기 편하게 만드는 것도 중요
- 안전하게 보관

● DB System 연구 개요 (마지막)

교과서를 중심으로

Part 1 : 관계형 데이터베이스 모델링

Part 2 : 관계형 데이터베이스 프로그래밍 - SQL (Java , PHP, JDBC 등)

Part 3 : 반정형 데이터 모델링 및 프로그래밍 - XML , XQuery

Part 4 : 데이터 베이스 시스템 구현

Part 5 : 최신 DB System 문제

◎ DBMS 개요.

* DBMS의 주요기능 구성요소

- 저장 : 효율적으로
- 질의 검색처리
- 트랜젝션 처리

파일수준 저장과정

유저 → 검색, 갱신 → 컴파일러 → 검색엔진 → 튜플 → 관계 → 저장소

DB관리자 → DDL 컴파일러 → 검색엔진
= DBA

* DBMS에 대한 명령

- DB 관리자 (DBA) : DDL을 사용하여 DB 스키마를 관리하는 특별한 권한.
  - ↳ 데이터 정의 언어 (Data Definition Language)

DB에서 하나의 논리적기능을 수행하기위한 작업의 단위 / 작업의 논리적단위

* Transaction (처리) 과정 : 쿼리 및 기타 DML 작업은 트랜젝션으로 그룹화됩니다.

→ Transaction : ACID 특성을 만족하는 연산들의 그룹.

* Transaction 의 ACID 특성

- A : 원자성 : 전부 또는 전무의 실행
- C : 항상성 : DB 분리에 충실성 (일관성) 유지, 트랜젝션의 실행이 완료된 후에는 데이터베이스가 모든 일관성 조건을 만족해야함
- I : 고립성 : 다른 처리가 동시에 실행되지 않는것처럼
- D : 내구성 : 일이 완료가 되고 DB에 미치는 영향은 절대 손실이 되면 안된다.

* 트랜젝션 관리를 위한 Tasks (작업)

- 로깅 (기록하는) : 원자성과 내구성을 위해, 로그관리자와 회복관리자.
  - (로그관리자 → 변화를 기록 / 회복관리자 → DB를 완전한 상태로 재저장)
- 동시성 제어 : 동시에 실행되는 여러 Transaction (처리)의 분리된 실행을 위해, 스케쥴러가 고립성 보장
  - DBMS내에있는 쿼 데이터블에 저장
- 교착상태 해결 : 거래가 교착상태에 빠질수 있다. (ex) 콘서트 예약 → 대기열 → 예약선택
  - 스케쥴러가 교착상태해결에 대한 책임

* 쿼리 프로세서

- 쿼리 컴파일러 : 무언가를 형식에 맞게 바꿔주는 것, 질의 파서, 질의 전처리기, 질의 최적화기 로 구성
  - 질의를 질의 계획으로 바꿔주는 것
  - 질의 파서 → 파서트리생성
  - 질의 전처리기 → 질의의 오류 (어) 부착명
  - 질의 최적화기 → 가장 효율적인 질의 계획으로 변환
- 실행 엔진 : 앞에서 쿼리를 짜 놓으면 여기서 실제로 탐색하는 부분

⇒ Query Processor

Query Processor
Query Compiler
Execution engine