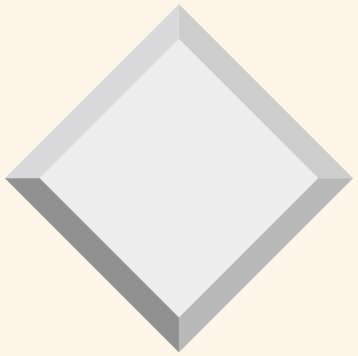


Introduction to Database Systems

Chapter 1

What Is a DBMS?

비트 Bit(b)	바이트 Byte(B)	킬로바이트 Kilobyte(KB)	메가바이트 Megabyte(MB)
기가바이트 Gigabyte(GB)	테라바이트 Terabyte(TB)	페타바이트 Petabyte(PB)	엑사바이트 Exabyte(EB)
제타바이트 Zettabyte(ZB)	요타바이트 Yottabyte(YB)	론나바이트 Ronnabyte(RB)	퀘타바이트 Quettabyte(QB)

1 KByte = 10^3 byte

- Data vs Information vs Knowledge
  - Data: raw facts that are described, observed, or measured
  - Information: data that have been prepared or organized
  - Knowledge: data/information/rules that are used for actual decision making

  (rules = 패턴)
- A (large, integrated) collection of (related) data. ⇒ DB
- Models real-world *enterprise.*
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS564)
- A *Database Management System (DBMS)* is a software package designed to store and manage databases.

cf) MS Access

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

cf) 모델링 : 현실세계를 묘사하기위한 가상의 것을 만드는것

Data Models

schema와 instance 구분하기!
ex) schema : DB의 구조설명
ex) DB 'A'는 relational data model이고 'SID' row가있다
instance : DB의 스냅샷
ex) 9/12일자로 보니 2016~ 학생데이터가있다

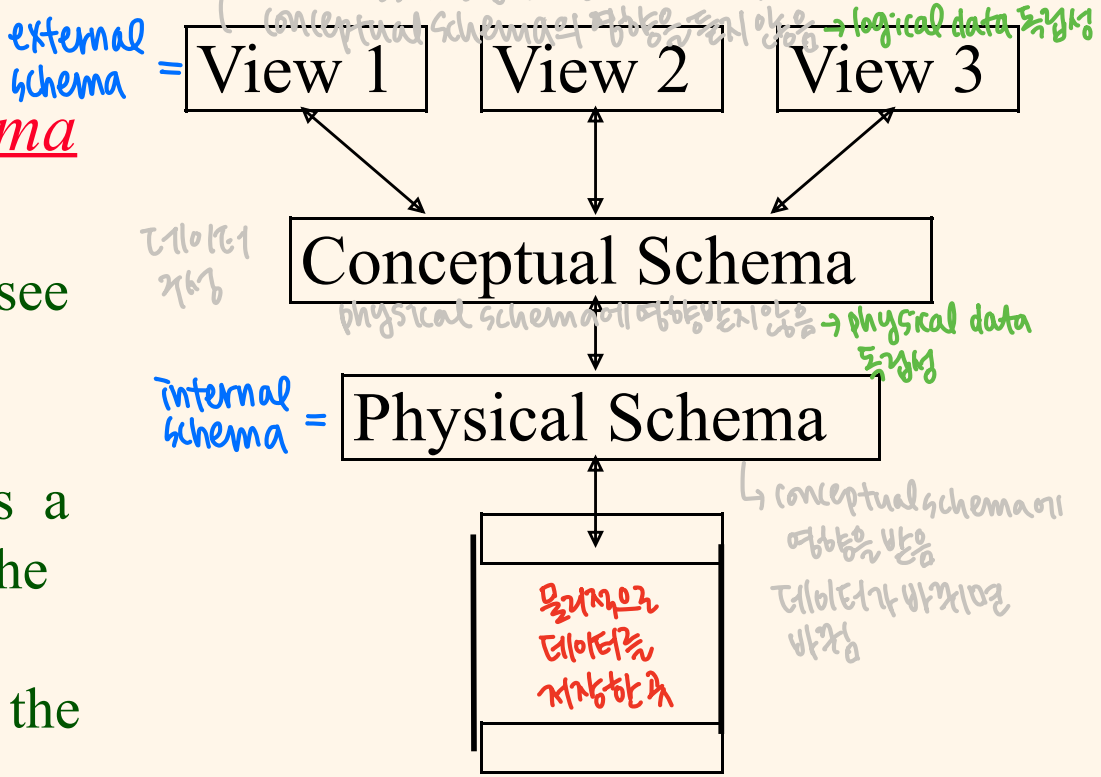
- A *data model* is a collection of concepts for describing data. → DB를만들기위함 (DB : collection of data)
- A *schema* is a description of a particular collection of data, using the given data model. (DB의 구조를 묘사 (칼럼, …))
- The *relational model of data* is the most widely used model today. ⇒ chap2,3에서계속
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.

History of Data Models

- Early 60's : Network Model
  - By Charles Bachman at GE
  - CODASYL (Conference On Data Systems Language)
- Late 60's : Hierarchical Model
  - IMS by IBM
  - SABRE airline reservation system
- 70's: Relational Model
  - By Edgar Codd at IBM
  - SQL from IBM System R
- Late 80's & early 90's: Extended(Object) Relational Model or Object-Oriented Model
- Late 90's: Data Warehousing, Data Mining, Distributed DB, Web Database, ERP(Enterprise Resource Planning),
- How about ER(Entity-Relationship Model)
  - Semantic Data Model vs Record-based Model

Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines a entire logical structure of the data
  - Physical schema describes the files and indexes used.



* *Schemas are defined using DDL; data is modified/queried using DML.*

Example: University Database

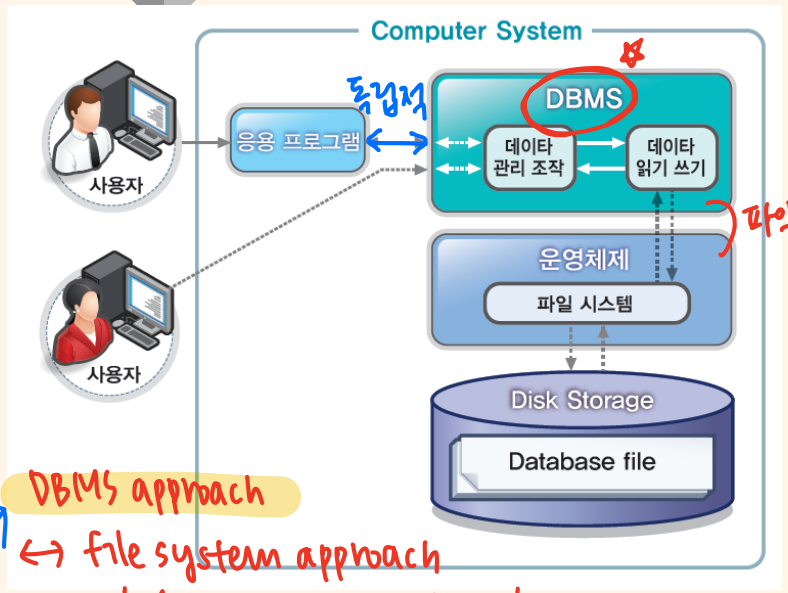
- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema: (internal schema)
  - Relations stored as unordered files.
  - Index on first column of Students.

  파일구조

  많아지면 오버헤드↑

  별도의 structure → Index
- External Schema (View):
  - *Course_info(cid:string, enrollment:integer)*

  ↳ 필요한 일부 column만 추출 (schema를 보는 방법의 일종)



DBMS approach
↔ file system approach
(직접접근, DBMS는 거치지 않음)

schema가 바뀌어도
다들어고치지 않아도됨

```
typedef struct
{
  char sid[10];
  char name[300];
  char login[20];
  int age;
  float gpa;
  char address[300];
} students;


int main()
{
 int counter;
 FILE *ptr_myfile;
 students my_record;

 ptr_myfile=fopen("c:\MyDB\test.db","rb");
if (!ptr_myfile)
{
  printf("Unable to open file!");
 return 1;
}

for ( counter=1; counter <= 10; counter++)
{
 fread(&my_record,sizeof(struct rec),1,ptr_myfile);
 printf("%d\n",my_record.x);
}
fclose(ptr_myfile);
return 0;
}
```

가장 중요한 이유!

Data Independence

- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data. → conceptual schema가 바뀐다해 view에 영향X
- *Physical data independence*: Protection from changes in *physical* structure of data.
  → physical schema가 바뀐다해 logical schema(+view)에 영향X
  단지 이것으로 인덱스를 새로 만들어 → physical schema의 변화

* *One of the most important benefits of using a DBMS!*

transaction : 거래
cf) ATM - deposit. withdraw

DB의 처리단위 (operation)

Transaction: An Execution of a DB Program

- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes). → operation
  쪼갤수없는 logical unit
  SELECT INSERT. DELETE. UPDATE
- Each transaction, executed completely, must leave the DB in a *consistent state* if DB is consistent when the transaction begins.
  transaction 실행중일때의 의미가 아님!
  시작과 끝에 consistent하게 유지하면됨

DBMS가 도와줌
commit. rollback
→ consistant state를 유지
DB

OS 내용!

↳ DBMS 가 알아서

Concurrency Control

동시성 제어

throughput을 높이기 위해 동시작업을 돌리면 동시성제어 필요

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.

(멀티프로그래밍) 교차수행 병렬처리 X

동시제어 관리 문제

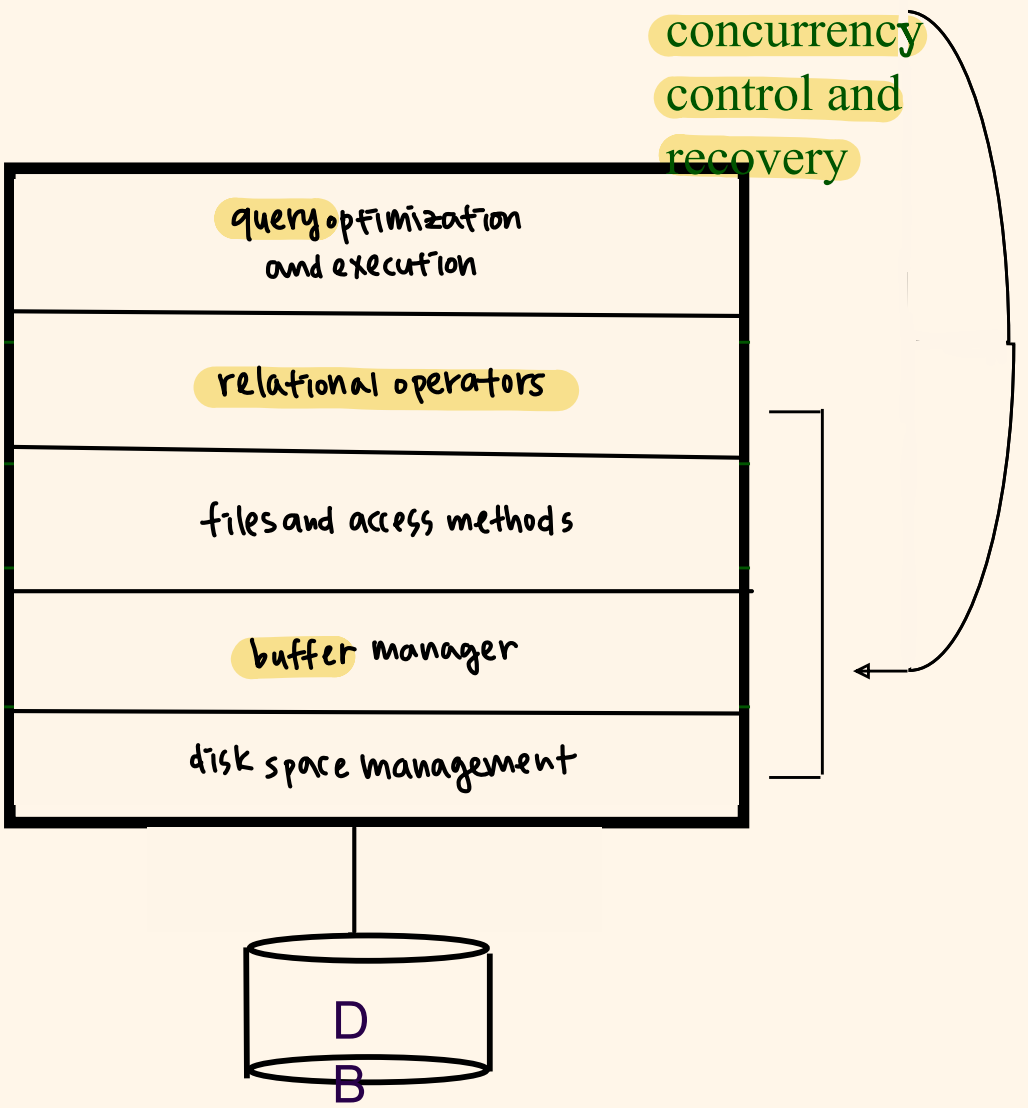
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.
- *Scheduling, Locking,,,,,,,,* +) semaphore

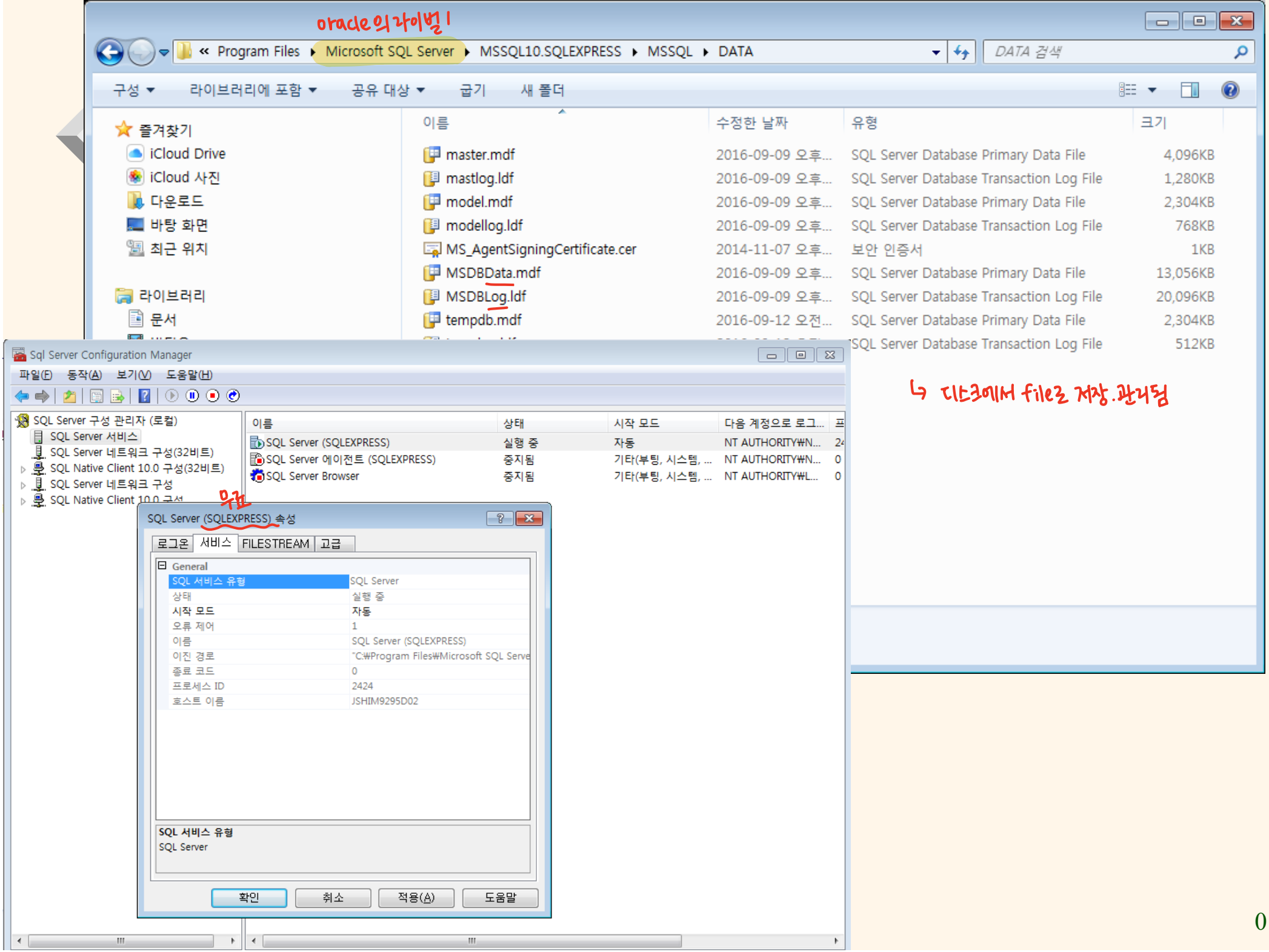
transaction fail 난경우? (transaction 중도에) ⇒ 없던것으로 처리해야함!
read/write 후 commit을 call해야 하나의 transaction이 끝남
log에 commit이 없는경우 transaction fail → recovery 작업필요

Structure of a DBMS

하나의 예시!

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.




oracle의 라이벌!
« Program Files ▸ Microsoft SQL Server ▸ MSSQL10.SQLEXPRESS ▸ MSSQL ▸ DATA
DATA 검색
구성 ▾ 라이브러리에 포함 ▾ 공유 대상 ▾ 굽기 새 폴더
즐겨찾기
iCloud Drive
iCloud 사진
다운로드
바탕 화면
최근 위치
라이브러리
문서
이름 수정한 날짜 유형 크기
master.mdf 2016-09-09 오후... SQL Server Database Primary Data File 4,096KB
mastlog.ldf 2016-09-09 오후... SQL Server Database Transaction Log File 1,280KB
model.mdf 2016-09-09 오후... SQL Server Database Primary Data File 2,304KB
modellog.ldf 2016-09-09 오후... SQL Server Database Transaction Log File 768KB
MS_AgentSigningCertificate.cer 2014-11-07 오후... 보안 인증서 1KB
MSDBData.mdf 2016-09-09 오후... SQL Server Database Primary Data File 13,056KB
MSDBLog.ldf 2016-09-09 오후... SQL Server Database Transaction Log File 20,096KB
tempdb.mdf 2016-09-12 오전... SQL Server Database Primary Data File 2,304KB
SQL Server Database Transaction Log File 512KB
↳ 디스크에서 file로 저장.관리됨
Sql Server Configuration Manager
파일(F) 동작(A) 보기(V) 도움말(H)
SQL Server 구성 관리자 (로컬)
SQL Server 서비스
SQL Server 네트워크 구성(32비트)
SQL Native Client 10.0 구성(32비트)
SQL Server 네트워크 구성
SQL Native Client 10.0 구성
이름 상태 시작 모드 다음 계정으로 로그...
SQL Server (SQLEXPRESS) 실행 중 자동 NT AUTHORITY₩N...
SQL Server 에이전트 (SQLEXPRESS) 중지됨 기타(부팅, 시스템, ... NT AUTHORITY₩N...
SQL Server Browser 중지됨 기타(부팅, 시스템, ... NT AUTHORITY₩L...
우클
SQL Server (SQLEXPRESS) 속성
로그온 서비스 FILESTREAM 고급
General
SQL 서비스 유형 SQL Server
상태 실행 중
시작 모드 자동
오류 제어 1
이름 SQL Server (SQLEXPRESS)
이진 경로 "C:₩Program Files₩Microsoft SQL Serve
종료 코드 0
프로세스 ID 2424
호스트 이름 JSHIM9295D02
SQL 서비스 유형
SQL Server
확인 취소 적용(A) 도움말

Summary

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence. → schema
- A DBMS typically has a layered architecture. + component 기반
- DBAs hold responsible jobs and are well-paid!
- DBMS R&D is one of the broadest, most exciting areas in CS.