

SQL

DBMS (Database Management System)



- **데이터베이스란?**

: 대규모의 정보를 사용하고 관리할 수 있도록 체계적으로 모아놓은 것. 필요에 의해 일정한 형식으로 저장해놓은 것

- **DBMS 란?**

: 이러한 데이터 집합과 데이터를 사용할 수 있는 프로그램으로 구성. 이런 프로그램을 통해 사용자가 편리하고 효과적으로 데이터를 추가/삭제/조회를 할 수 있고, 에러 핸들링, 보안 등의 기능도 제공한다.

- **데이터베이스 응용분야**

1. 은행 계좌 이체
2. 공항 예약, 스케줄
3. 대학 수강 신청, 학적 관리

DBMS가 없다면? - 파일 시스템을 통해 데이터를 저장하면?



1. Data Redundancy, Inconsistency

: 시간이 지나며 App 이 변경되거나 서로 다른 App 이 같은 데이터에 접근할 경우 파일 포맷이 다르다던지 등의 이유로 불필요한 카피가 생기고, App간 데이터 불일치 문제 발생 가능

2. Data 요구사항 변경에 대응하기 어렵다

: 기존의 App 시스템이 모든 데이터를 불러오도록 구현되어 있는데 특정 데이터만 뽑는 새로운 요구사항이 들어왔을 때 매번 새로운 App을 짚어내야 한다.

3. Data Isolation

: 서로 관련 있는 데이터가 서로 다른 파일/ 포맷에 존재하는 것. 이 또한, App마다 데이터 관리 구현이 달라 발생할 수 있는데, 이 경우 데이터 관리가 매우 복잡해진다.

4. Atomic한 업데이트가 어렵다.

: 데이터 업데이트는 완전히 성공하거나 아예 건드리지 않아야 일관성 문제를 피하고, 대처할 수 있다. 파일 시스템은 부분적인 업데이트를 막지 못한다.

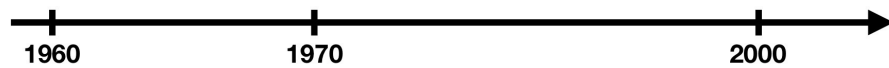
5. 여러 사용자의 동시 접속 관리 불가

: 파일 시스템은 자체적으로 동시 접속 관리 매커니즘이 없으므로 불일치 문제를 야기한다.

6. 보안 문제

: 파일 시스템은 데이터를 일부만 허용하는게 어렵다. 유저 access control 관리가 어렵다.

File → SQL → NoSQL



SQL (Structured Query Language)

1. DDL (Data definition Language) - CREATE / ALTER / DROP / RENAME / TRUNCATE



- **DDL 이란?**

: 스키마를 정의/ 수정/ 삭제할 때 사용

- 각 Relation의 스키마
- 각 Attribute의 도메인
- Integrity Constraints (데이터 제약 조건)
- 기타
 - Relation별 index 집합
 - Relation별 보안/권한 정보
 - Relation별 물리적 storage 구조

- **SQL의 도메인 타입**

- char(n) : 고정 길이 문자열 (n : 길이)
- varchar(n) : 가변 길이 문자열 (n : max 길이)
- int : 정수
- smallint : 작은 정수
- numeric(a, b) : 고정 소수점 실수 (a : 정수부 길이, b : 소수부 길이)
- real : 가변 길이 부동 소수점 실수
- float : 고정 길이 부동 소수점 실수

- **ex)**

- ID VARCHAR(20) NOT NULL
: ID는 최대 길이 20인 문자열이며, NULL일 수 없다.
- Deposit INT DEFAULT 0
: Deposit은 정수이며 값을 주지 않으면 0이다.

- **테이블 생성**

```
create table student {  
    ID int,  
    LastName varchar(255),  
    FirstName varchar(255) not null,  
    Address varchar(255),  
}
```

```
City varchar(255),
    primary key(ID),
    foreign key(dept_name) references department(dept_name)
);
```

- **not null**

: 값을 NULL로 설정하는 것을 막음

- **primary key**

: 기본키로 지정. 값이 NULL이 될 수 없으므로 자동으로 not null 처리된다. 또한, 중복된 값을 가진 튜플이 삽입되는 것을 막음

- **foreign key**

: 외래키로 지정, reference Relation의 Attribute는 해당 Relation의 기본키여야 한다.

또한 예제에서 instructor의 dept_name은 반드시 department에 존재하는 튜플의 dept_name만 들어갈 수 있다.

- 테이블 삭제/ 수정

1. **DROP TABLE Shippers;**

: 테이블과 내용 삭제

2. **TRUNCATE TABLE Student;**

: 테이블에서 행을 모두 제거

3. **ALTER TABLE Student ADD birth DATETIME;**

: 테이블의 속성 추가, 추가된 attribute에 대한 값들은 일단 NULL값으로 초기화된다.

4. **ALTER TABLE Student DROP Address;**

: 테이블 속성 삭제. 단, 외래키로 사용되는 속성을 삭제해버리면 참조 무결성을 해칠 우려가 있어 지원되지 않는 경우가 많다.

	DELETE	TRUNCATE	DROP
명령어 종류	DML	DDL	DDL
처리속도	느림	빠름	빠름
commit	사용자가 직접	자동	자동
Rollback 가능 여부	commit이전 가능	불가	불가
삭제되는 정도	데이터만 삭제	테이블을 CREATE 상태로 되돌림	테이블까지 완전히 제거

2. DML (Data Manipulation Language) - SELECT / INSERT / UPDATE / DELETE



- **select / select all**

: 중복을 허용하는 일반적인 select

- **select distinct**

: 중복을 제거하는 select

- **select ***

: 모든 Attribute. 즉. 테이블을 그대로 원하는 경우

- **select 사칙 연산**

: attribute 값에 산술연산을 가미한 결과를 얻고 싶은 경우. 원본 Relation에는 영향을 주지 않음

```
select all dept_name from instructor;
select distinct dept_name from instructor;
select * from instructor;
select ID, name, salary/12 from instructor;
```

- **where 절**

: 튜플 필터링 조건을 명시. and, or, not 과 같은 논리 연산자를 사용할 수 있다.

또한, 날짜, 시간, 문자열 등에 비교 연산자를 사용할 수 있다.

```
// 소속이 K02 인 선수 중에서 포지션이 MF가 아니고, 175 이상 185 이하가 아닌 선수를 찾0
SELECT      PLAYER_NAME 선수이름, POSITION 포지션, BACK_NO 백넘버, HEIGHT 키
FROM        PLAYER
WHERE       TEAM_ID = 'K02'
AND NOT     POSITION = 'MF'
AND NOT     HEIGHT BETWEEN 175 AND 185 ;
```

```
// 국적이 NULL이 아닌 선수를 찾아라
SELECT      PLAYER_NAME 선수이름, NATION 국적
FROM        PLAYER
WHERE       NATION IS NOT NULL ;
```

- **from 절**

: 대상 Relation 명시, 여러 Relation을 대상으로 할 수 있음

```
select name, course_id from instructor, teaches
```

Cartesian Product: *instructor X teaches*

instructor

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000

teaches

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

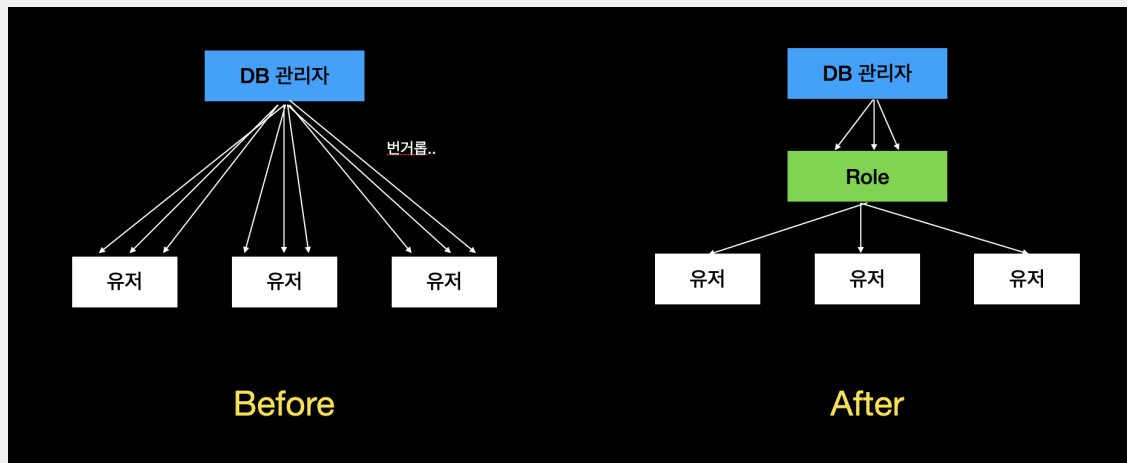
inst.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...
...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010

위와 같이 여러 Relation을 나열할 경우, **Cartesian Product** 한 것을 결과 Relation으로 반환한다. 이로 인해, 불필요한 튜플이 많이 끼게 되므로 여러 Product를 사용하는 경우 where 절을 함께 사용하는 것이 좋다. (**join** 용을 추천)

3. DCL (Data Control Language) - GRANT / REVOKE



- **ROLE**



- **SQL 표준 권한**

- **select** : relation을 읽거나, view를 만드는 query를 작성할 권한
- **insert** : 튜플을 삽입할 권한, 일부 Attribute만 허용할 수 있다.
- **update** : 튜플을 수정할 권한, 일부 Attribute만 허용할 수 있다.
ex) `grant update ({Attribute}) on {Relation} to {user_id})`
- **delete** : 튜플을 삭제할 권한, 일부 Attribute만 허용할 수 있다.
- **all privileges** : 모든 권한, 사용자는 자신이 만든 relation에 대해서는 all privileges를 자동으로 갖는다.

- **권한 부여**

```
create role instructor;
// instructor라는 role 생성

grant instructor to Amit;
// 유저 Amit에게 instructor의 권한 부여

grant select, delete on Student to instructor;
// Student Relation의 select 권한을 instructor에게 부여

create role assistant;
grant assistant to instructor;
```

```
// assistant 라는 role을 생성한 후, instructor에 연결하면, assistant에 부여하는  
// 권한은 자동으로 instructor도 갖게 된다.
```

- 권한 취소

```
revoke insert on Student from manager;  
// manager에 부여된 insert 권한 회수  
  
revoke all on Student from manager;  
// manager에 부여된 모든 권한 회수  
  
revoke manager from Amit;  
// 사용자 Amit을 manager로부터 배제
```

4. TCL (Transaction Control Language) - COMMIT / ROLLBACK / SAVEPOINT

: 논리적인 작업의 단위를 묶어서 **DML**에 의해 조작된 결과를 작업단위(트랜잭션) 별로 제어하는 명령어를 말함.



자동 COMMIT 시점

- DDL 문장 실행, DB 정상 종료, 애플리케이션 종료시


COMMIT, ROLLBACK - 데이터 무결성을 보장하기 위함

```
ROLLBACK;  
ROLLBACK TO SV1;  
  
SAVEPOINT SV1;
```

< 참고 자료 >

[CS] 데이터베이스 - 1. DBMS와 관계형 모델


데이터베이스란 대규모의 정보를 사용하고 관리할 수 있도록 체계적으로 모아놓은 것. 필요에 의해 일정한 형식으로 저장해놓은 것 DBMS란 무엇인가 이러한 데이터집합과 데이터를 사용할 수 있는 프로그램으로 구성. 이런 프로그램을 통해 데이터를 추가/삭제/조회를 할 수 있음. 사

 <https://velog.io/@yohanblessyou/CS-데이터베이스-1.-Introduction>

velog

데이터베이스 SQL & NoSQL

SQL & NoSQL의 등장 데이터를 컴퓨터 내에 보관하기 시작한 이후부터 불가피하게 발생한 여러 문제점들을 보완하다보니 파일시스템이 점차 발전하게 된다. 각각의 문제점들과 보완된 DB방식의 가장 큰 축에 1970년대에 만들어진 SQL과 2000년 이후의 NoSQL 방식이 있다. File → SQL.

 <https://velog.io/@yejinh/데이터베이스-SQL-NoSQL-pkk4zdy3au>

File → SQL → NoSQL

