

MySQL

2012년 웹프로그래밍
권동섭

Reference

- Official site:
 - <http://www.mysql.com/>
 - <http://dev.mysql.com/doc/>
- W3school
 - http://w3schools.com/php/php_mysql_intro.asp
 - <http://w3schools.com/sql/default.asp>

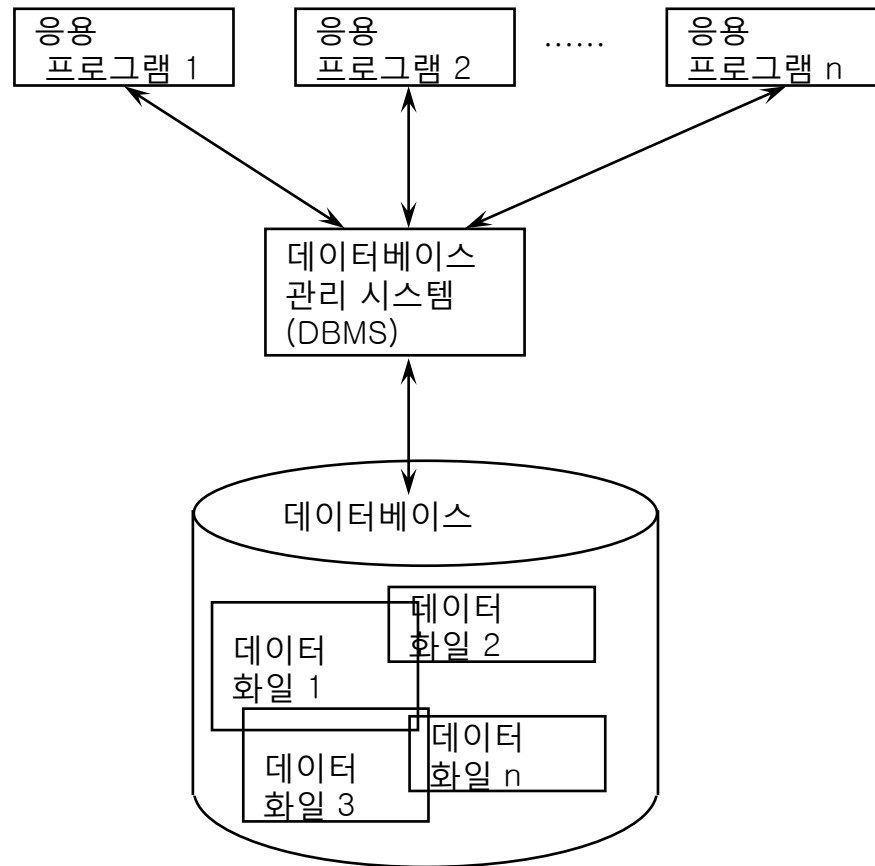
DATABASE

Database & DBMS

- Database
 - 한 조직의 여러 응용 시스템들이 공용(Shared)하기 위해 통합(Integrated), 저장(Stored)한 운영 데이터(Operational data)의 집합
- Database Management System (DBMS)
 - DB관리를 위한 시스템
 - 통합된 데이터 관리
- 장점
 - 데이터 중복 제거
 - 데이터 공용
 - 일관성/무결성/보안 유지
 - 표준화 용이

DBMS

- 데이터의 종속성과 중복성의 문제 해결
- 데이터베이스를 공유할 수 있도록 관리하는 시스템



DBMS 제품

- Oracle: RDBMS 최초 상용화, RDBMS 시장 점유율 가장 높음 (국내 점유율 특히 높음)
- IBM DB2: RDBMS 최초개발, 메인프레임등에서 점유율 높음
- MS-SQL Server: Sybase 코드에 기반
- 기타: Teradata, Informix, Sybase ...

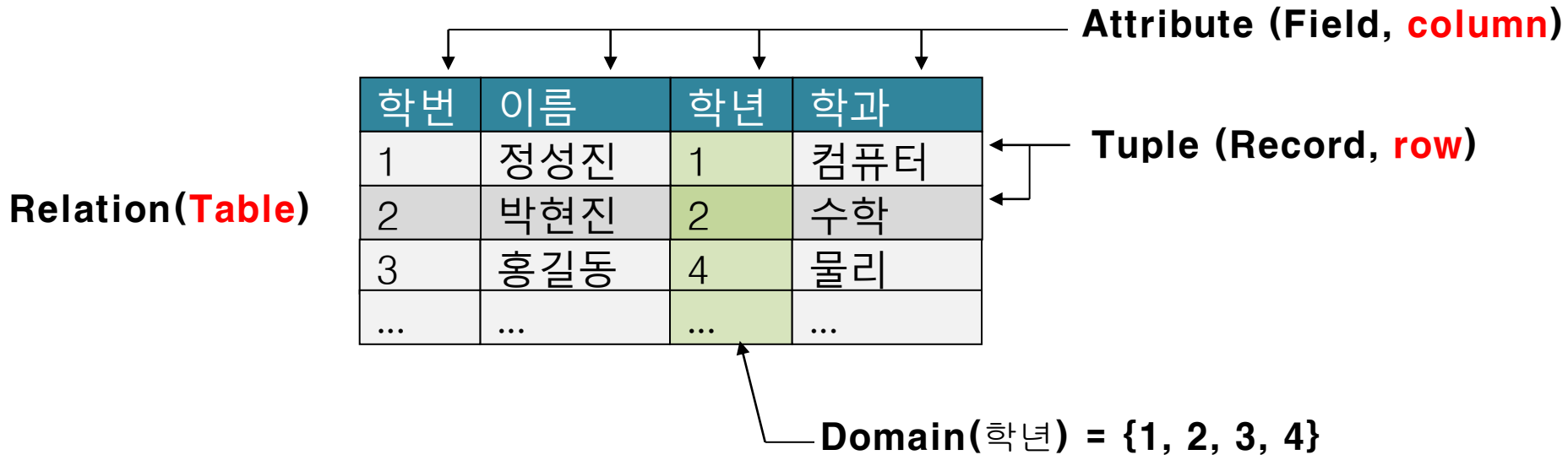
Relational Database Management Systems (RDBMS) Vendors					
Total Software Revenue, Worldwide, 2010-2011 (Millions of U.S. Dollars)					
Vendor	(Ctrl) 2010	2011	Share of 2010	Share of 2011	Growth 2011
Oracle	9,990.5	11,787.0	48.2%	48.8%	18.0%
IBM	4,300.4	4,870.4	20.7%	20.2%	13.3%
Microsoft	3,641.2	4,098.9	17.6%	17.0%	12.6%
SAP/Sybase	744.4	1,101.1	3.6%	4.6%	47.9%
Teradata	754.7	882.3	3.6%	3.7%	16.9%
Other Vendors	1,315.3	1,389.7	6.3%	5.8%	5.7%
Grand Total	20,746.6	24,129.5	100.0%	100.0%	16.3%
Source: Gartner (March 2012)					

기타 DBMS

- Open Source: **MySQL**, PostgreSQL, Firebird, Cubrid
- Main-Memory(Real-time) DB : Altibase, TimesTen
- Embedded DB: SQLite, BerkeleyDB
- NoSQL: MongoDB, CouchDB, HBase, Cassandra, ...

관계형 데이터베이스

- 모든 데이터를 테이블에 기반하여 저장



- 데이터 조작/검색 등은 **SQL언어**로 수행

PK & FK

- Primary Key: 테이블 내에서 각 튜플을 구별하는 유일키
- Foreign Key: 테이블 간의 관계를 표현

PK FK EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80/12/17	880		20
7499	ALLEN	SALESMAN	7698	81/02/20	1760	300	30
7521	WARD	SALESMAN	7698	81/02/22	1375	500	30
7566	JONES	MANAGER	7839	81/04/02	2975		20
7654	MARTIN	SALESMAN	7698	81/09/28	1375	1400	30
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7788	SCOTT	ANALYST	7566	87/04/19	3000		20
7839	KING	PRESIDENT		81/11/17	5000		10
7844	TURNER	SALESMAN	7698	81/09/08	1650	0	30
7876	ADAMS	CLERK	7788	87/05/23	1210		20
7900	JAMES	CLERK	7698	81/12/03	1045		30
7902	FORD	ANALYST	7566	81/12/03	3000		20
7934	MILLER	CLERK	7782	82/01/23	1430		10

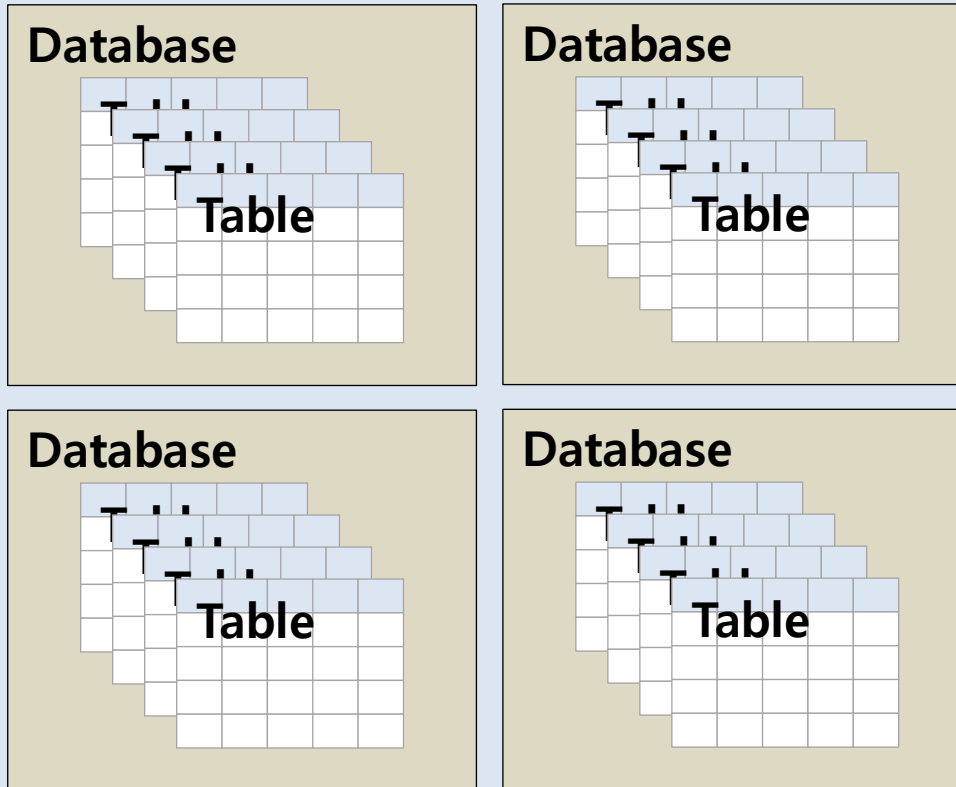
PK DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

MYSQL

MySQL 구조

MySQL 서버

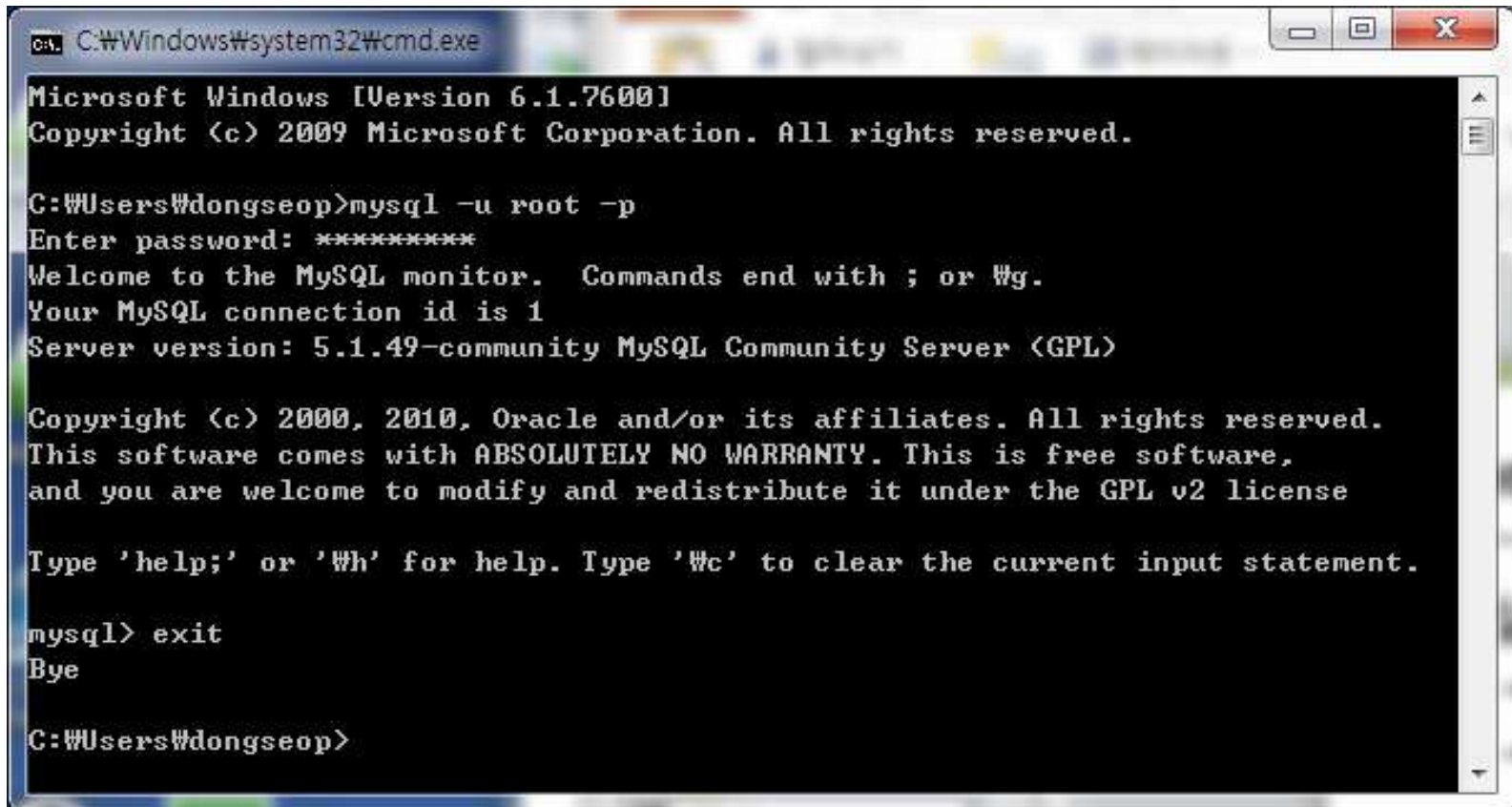


User



Command-line tool

- 접속
 - 명령행(cmd)에서 mysql 실행
 - `mysql -u {사용자명} -p{비밀번호} {데이터베이스이름}`



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\dongseop>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.49-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye

C:\Users\dongseop>
```

Database 생성 & 권한 부여

- root로 접속: **mysql -u root -p**
- 데이터베이스 생성: **CREATE DATABASE {DB이름};**
- 데이터베이스 사용: **USE {DB이름};**
- 데이터베이스 목록보기: **SHOW DATABASES;**
- 사용자 생성/권한 부여:
GRANT ALL ON {DB이름}.* TO '{아이디}'
IDENTIFIED BY '{비밀번호}';

실습 1: DB/사용자 생성

```
D:\Projects>mysql -u root -p
Enter password: *****
```

root로 접속

```
...
mysql> show databases;
+-----+
| Database |
+-----+
...
+-----+
3 rows in set (0.02 sec)
```

```
mysql> create database wp_test;
Query OK, 1 row affected (0.00 sec)
```

wp_test DB 생성

```
mysql> grant all on wp_test.* to 'id001' identified by 'pwd001';
Query OK, 0 rows affected (0.01 sec)
```

id001 / pwd001 생성, 권한부여

```
mysql> exit;
Bye
```

```
D:\Projects>
```

실습 1: (계속)

```
D:\Projects>mysql -u id001 -ppwd001 wp_test
```

```
...
```

```
mysql> show databases;
```

```
+-----+
```

```
| Database |
```

```
+-----+
```

```
| information_schema |
```

```
| wp_test |
```

```
+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> show tables;
```

```
Empty set (0.00 sec)
```

```
mysql>
```

생성한 ID/DB로 접속

생성된 DB 확인

현재 DB내의 테이블 확인

테이블 생성

- 생성:
 - CREATE TABLE {테이블이름} ({컬럼이름} {데이터타입}, ...);
- 데이터 타입:
 - 숫자형: TINYINT(1), SMALLINT(2), INT(4), FLOAT(4), DOUBLE(8)
 - 문자형: CHAR(n), VARCHAR(n) / TEXT
 - BINARY: BINARY(n), VARBINARY(n) / BLOB
 - 날짜/시간: DATETIME, DATE, TIME, TIMESTAMP
- 제약조건
 - PRIMARY KEY: PK설정(NOT NULL + UNIQUE)
 - UNIQUE: 중복 안됨
 - DEFAULT: 기본값 설정
 - NOT NULL: NULL안됨
- 기타:
 - 자동 증가: AUTO_INCREMENT

테이블 기타

- 테이블 삭제: DROP TABLE {테이블 이름}
- 테이블 변경: ALTER TABLE {테이블 이름} ...
- 테이블 목록: SHOW TABLES;
- 테이블 정의 확인(describe): DESC {테이블 이름}
- 테이블 이름 변경: RENAME TABLE *xxx* TO *yyy*;

실습 2: 테이블 생성

```
CREATE TABLE users (  
  id VARCHAR(30) PRIMARY KEY,  
  password VARCHAR(80) NOT NULL,  
  name VARCHAR(30) NOT NULL,  
  email VARCHAR(80) NOT NULL,  
  birthday DATE,  
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

users 테이블 생성

```
CREATE TABLE articles (  
  id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,  
  title VARCHAR(200) NOT NULL,  
  content TEXT,  
  user_id VARCHAR(30) NOT NULL,  
  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

articles 테이블 생성

```
SHOW TABLES;  
DESC users;  
DESC articles;
```

테이블 생성 내용 확인

데이터 저장/수정/삭제

- 저장: **INSERT INTO 테이블명(컬럼명...) VALUES (값...);**
 - 컬럼명의 순서에 따라 VALUES에 값을 열거
 - 컬럼 정보가 없으면 create 순서대로
 - 정의되지 않은 값은 DEFAULT/NULL이 저장됨
 - 자동 증가 컬럼의 경우 자동 증가됨
- 수정: **UPDATE 테이블명 SET 이름=값 {WHERE 조건}**
 - 조건에 해당하는 모든 레코드 변경
 - 조건은 SELECT 문 참고
- 삭제: **DELETE FROM 테이블명 {WHERE 조건}**
 - 조건에 해당하는 모든 레코드 삭제

실습 3: 데이터 삽입

```
INSERT INTO users(id, password, name, email, birthday)
VALUES('jane', 'abc', 'Patrick Jane', 'patrick@gmail.org', '1970-
04-16');
```

```
INSERT INTO users(id, password, name, email)
VALUES('lisbon', 'def', 'Teresa Lisbon', 'teresa@gmail.org');
```

```
INSERT INTO users(id, password, name, email, birthday)
VALUES('cho', 'ghi', 'Kimball Cho', 'kimball@gmail.org', NULL);
```

```
INSERT INTO articles(title, content, user_id)
VALUES('Love you', 'No secrets, Jane. No lies. No tricks. No
surprises. The truth.', 'lisbone');
```

```
INSERT INTO articles(title, content, user_id)
VALUES('Hello', 'Hello\nMy name is Patrick Jane\n', 'jane');
```

```
INSERT INTO articles(title, content, user_id)
VALUES('Test Article', 'This is a test article.\nShow me the
money\n', 'jane');
```

데이터 검색 (기본)

- **SELECT** 컬럼리스트
FROM 테이블
{**WHERE** 조건}
{**ORDER BY** 정렬 순서}
- 컬럼 리스트: *는 모든 컬럼을 의미
- 조건: 컬럼 값 필터링, 참(TRUE)인 튜플만 결과로 나옴
 - AND, OR, () 등 사용 가능
- 정렬 순서: 오름차순(ASC), 내림차순(DESC)
 - score DESC, name ASC → 앞 컬럼 부터 우선 순위가 높음

실습 4: 데이터 검색

```
SELECT * FROM users;
SELECT * FROM users WHERE id='jane';
SELECT id, password FROM users WHERE id='jane';
SELECT id, birthday FROM users WHERE birthday < '1975-10-10';
SELECT id, password FROM users WHERE birthday > '1975-10-10';

SELECT * FROM articles;
SELECT title, user_id FROM articles;
SELECT title, user_id FROM articles ORDER BY title;
SELECT title, user_id FROM articles
    WHERE user_id = 'jane' ORDER BY title;
SELECT title, user_id FROM articles ORDER BY user_id, title;
SELECT title, user_id FROM articles ORDER BY user_id DESC, title;

SELECT title, user_id FROM articles
    WHERE content LIKE '%the%';
SELECT title, user_id FROM articles
    WHERE content LIKE '%the%' AND user_id='jane';
SELECT title, user_id FROM articles
    WHERE content LIKE '%the%' OR user_id='jane';
```

실습 5: 데이터 수정 / 삭제

데이터 수정

```
UPDATE users SET birthday = '1980-05-01' WHERE id='lisbon';
UPDATE articles SET title = 'Hello Everyone', content = 'My name
    is Patrick Jane.\nHello.' WHERE id = 2;
```

```
INSERT INTO users(id, password, name, email, birthday)
VALUES('kim', '123', 'James Kim', 'kim@gmail.org', '1984-01-01');
```

```
SELECT * FROM users;
```

```
DELETE FROM users WHERE id = 'kim';
SELECT * FROM users;
```

```
DELETE FROM users;
SELECT * FROM users;
```

레코드 삭제

전체 데이터 삭제

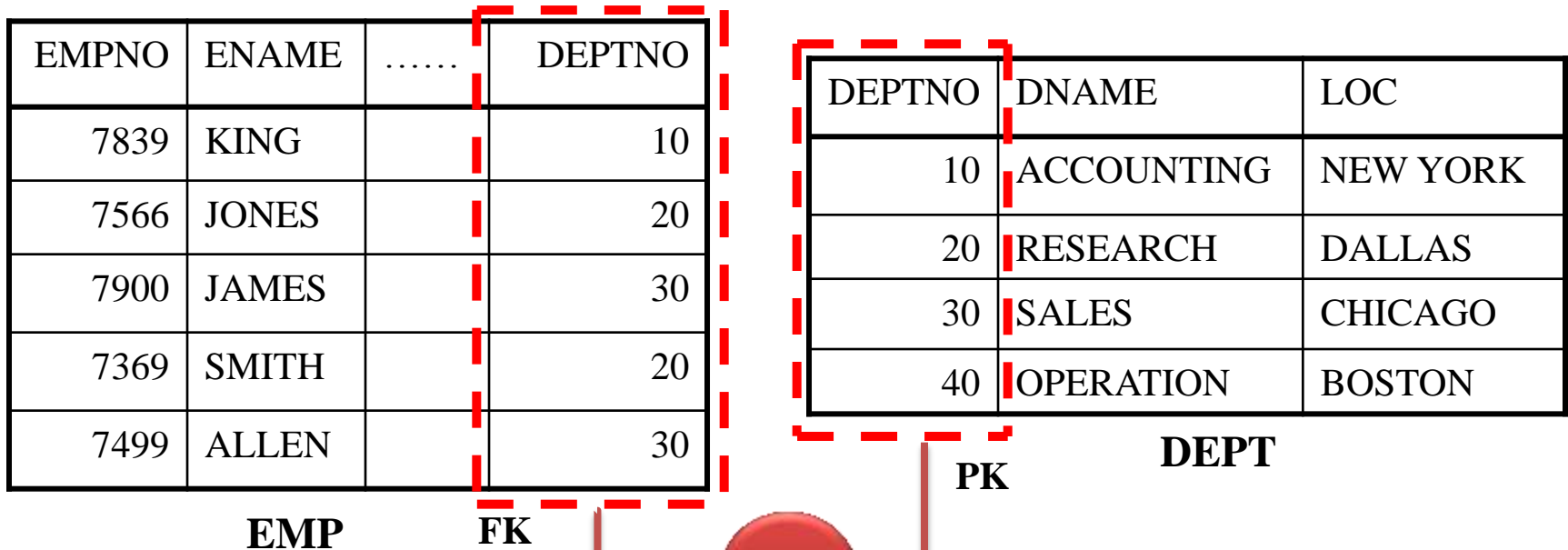
다음 실습을 위하여 실습 3 중 users 테이블의 INSERT를 다시 실행 하자.

Join

- 둘 이상의 테이블을 합쳐서 하나의 큰 테이블로 만드는 방법
- 예: User name이 'Patrick Jane'인 사용자가 작성한 Article은?
 - Articles에는 user_id 밖에 없음. Users 테이블에서 'Patrick Jane'을 찾고 다시 SQL을 작성하여 Article 검색???
 - 2번의 SQL문을 수행하므로 비효율적
 - Article과 User를 Join(합쳐서) 하여 검색 (1번의 SQL)
 - SELECT *
FROM **articles, users**
WHERE **articles.user_id = users.id**
AND users.name = 'Patrick Jane'

Join

```
SELECT * FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO
```



EMPNO	ENAME	DEPTNO	DEPTNO	DNAME	LOC
7839	KING		10	10	ACCOUNTING	NEW YORK
7566	JONES		20	20	RESEARCH	DALLAS
7900	JAMES		30	30	SALES	CHICAGO
7369	SMITH		20	20	RESEARCH	DALLAS
7499	ALLEN		30	30	SALES	CHICAGO

기타 SQL Features

- Outer Join
- GROUP BY / HAVING
- Aggregate Function
- Sub-query

실습 6: Join & 기타

// 두 가지 Join 표현 방법

```
SELECT * FROM users, articles WHERE users.id = articles.user_id;  
SELECT * FROM users JOIN articles ON users.id = articles.user_id;
```

// Aggregate Function

```
SELECT count(*) FROM articles WHERE user_id = 'jane';
```

```
SELECT users.id, count(articles.id)  
FROM users JOIN articles ON users.id = articles.user_id  
GROUP BY users.id;
```

// OUTER JOIN

```
SELECT * FROM users LEFT JOIN articles ON  
                                users.id = articles.user_id;  
SELECT users.id, count(articles.id)  
FROM users LEFT JOIN articles ON users.id = articles.user_id  
GROUP BY users.id;
```