



## 01. 데이터베이스 개요

1. DATA
2. DATABASE
3. DBMS (Database Management System)
4. SQL 개요

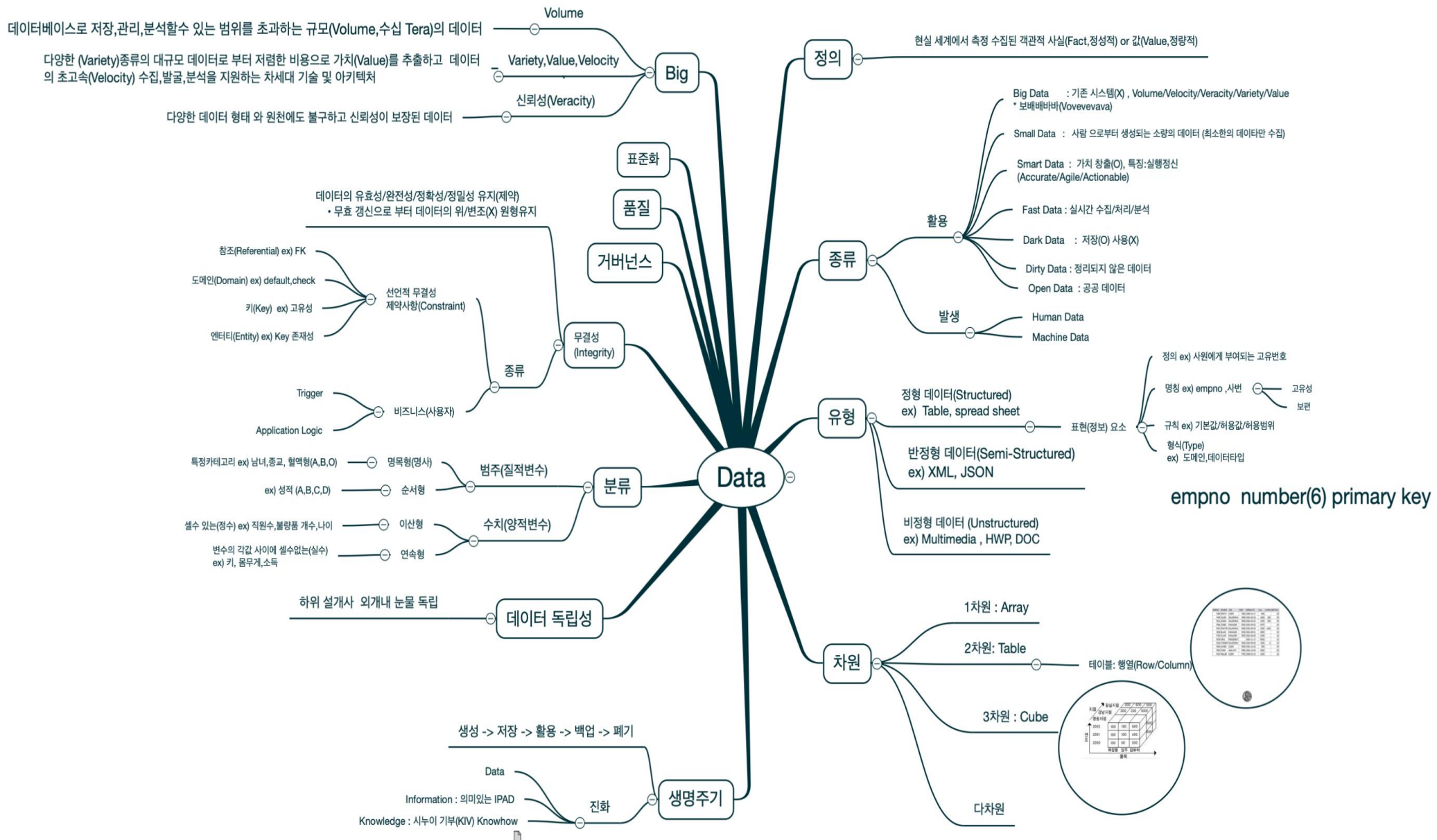
# ● 1. DATA

---

## DATA 는?

- 현실세계로 부터 단순한 관찰이나 측정을 통해서 수집된 **사실(Fact)**이나 **값(Value)**
  - 가공(처리)되지 않은 원본 데이터
  - 자료
- 
- ☐ 비즈니스 데이터 (Business Data , Transaction Data)  
ex) 계좌정보, 대출 내역, 자산관리 내역 , 통화내역, 과금정보
  - ☐ 머신 데이터 (Machine Data)  
ex) Application log, CDR(Call Detail Record),센서 데이터, Access Log, Alert Log
  - ☐ 빅데이터(Big Data)  
ex) 통신사 통화 품질 ,SNS ,VOC(Voice Of Customer) , 스마트팩토리 생산공정 데이터

# 1. DATA



## ● 2. DATABASE

정의	<ul style="list-style-type: none"> <li>- Data + <b>Base</b>(토대, 기초, 기지, 근거지, 저장)</li> <li>- <b>데이터 집합</b></li> <li>- (논리적인 연관성을 통해 <b>구조적으로 통합된</b>) <b>데이터 집합</b></li> <li>- 다수의 사용자가 공유(<b>Sharable</b>)하기 위해 통합(<b>Integrated</b>), 저장(<b>Stored</b>)한 운영(<b>Operational</b>) 데이터의 집합</li> <li>* <b>DB IS OS</b></li> </ul>
특징	<ol style="list-style-type: none"> <li>1) <b>동시 공유 (Concurrent sharing)</b> <ul style="list-style-type: none"> <li>- 다수의 사용자(App)가 동시에 사용</li> </ul> </li> <li>2) <b>계속적인 변화 (Continuous evolution)</b> <ul style="list-style-type: none"> <li>- 저장된 데이터는 사용자의 요청(삽입, 수정, 삭제)에 따라 동적으로 변화</li> </ul> </li> <li>3) <b>실시간 접근성 (Real-time accessibilitiy)</b> <ul style="list-style-type: none"> <li>- 사용자의 요청(질의:Query)에 실시간 처리 및 응답</li> </ul> </li> <li>4) <b>내용에 의한 참조 (Content reference)</b> <ul style="list-style-type: none"> <li>- 물리적인 위치나 주소가 아닌 데이터 내용(값)에 의해 참조</li> </ul> </li> </ol>
활용	고객 DB, 인사/회계 DB, 학사 행정 DB, 금융정보 DB
분류 (모델기반)	<ol style="list-style-type: none"> <li>1) 계층형 데이터베이스 (Hierarchical Data Model - Hierarchical DB)</li> <li>2) 네트워크형 데이터베이스 (Network Data Model - Network DB)</li> <li>3) <b>관계형 데이터 베이스 (Relational Data Model - Relational DB)</b></li> <li>4) 객체지향형 데이터베이스 (Object Oriented Data Model - Object Oriented DB)</li> </ol>
*구조	5) <b>객체-관계형 데이터 베이스 (Object-Relational Data Model- Object-Relational DB)</b>

## 2. DATABASE

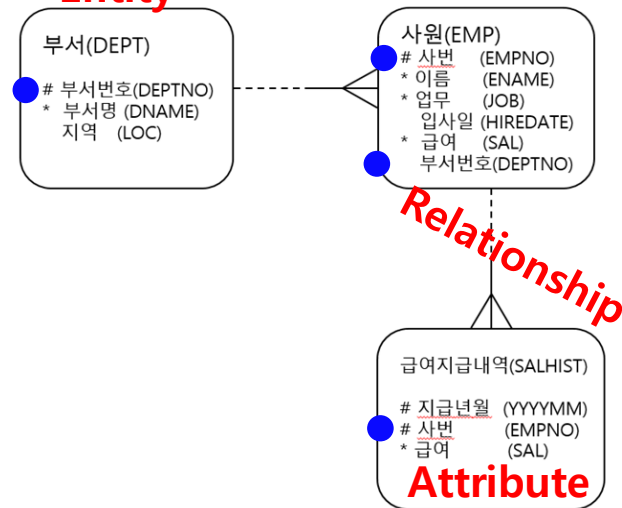
### Relational Data Model

EMPNO	NUMBER(4)
ENAME	VARCHAR2(10)
JOB	VARCHAR2(9)
MGR	NUMBER(4)
HIREDATE	DATE
SAL	NUMBER(7,2)
COMM	NUMBER(7,2)
DEPTNO	NUMBER(2)

### Table

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	17/09/10	800		20
2	7499	ALLEN	SALESMAN	7698	17/09/10	1600	300	30
3	7521	WARD	SALESMAN	7698	17/09/10	1250	500	30
4	7566	JONES	MANAGER	7839	17/09/10	2975		20
5	7654	MARTIN	SALESMAN	7698	17/09/10	1250	1400	30
6	7698	박보검	MANAGER	7839	17/09/10	2850		30
7	7782	CLARK	MANAGER	7839	17/09/10	2450		10
8	7788	SCOTT	ANALYST	7566	17/09/10	3000		20
9	7839	KING	PRESIDENT		17/09/10	5000		10
10	7844	TURNER	SALESMAN	7698	17/09/10	1500	0	30
11	7876	ADAMS	CLERK	7788	17/09/10	1100		20
12	7900	JAMES	CLERK	7698	17/09/10	950		30
13	7902	FORD	ANALYST	7566	17/09/10	3000		20
14	7934	MILLER	CLERK	7782	17/09/10	1300		10

### Entity



\* 테이블(Table) : 관계형 데이터베이스에서 데이터를 저장하는 기본적인 구조 (그릇,틀)  
행 과 열로 구성된 2차원 구조

가로	행	Row	인스턴스(Instance)	레코드(Record)
세로	열	Column	속성(Attribute)	필드(Field)

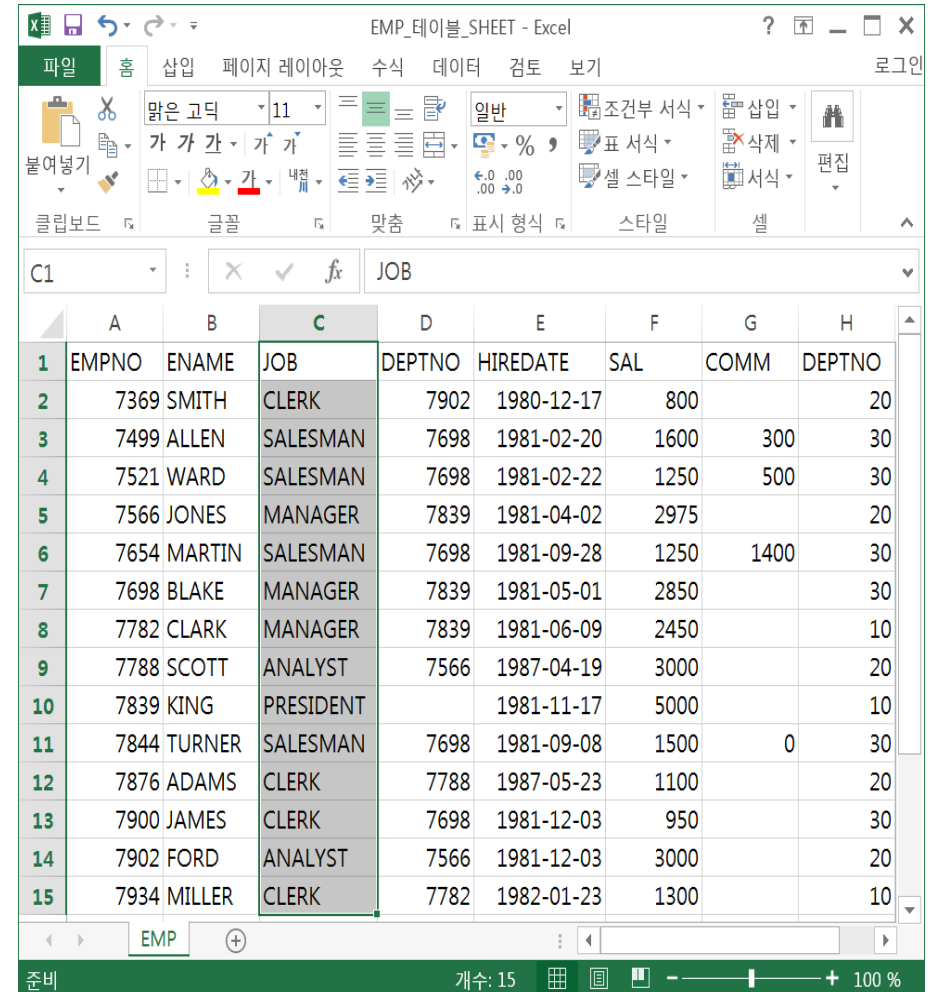
#: identifier , \*: mandatory

## ● 2. DATABASE

Table

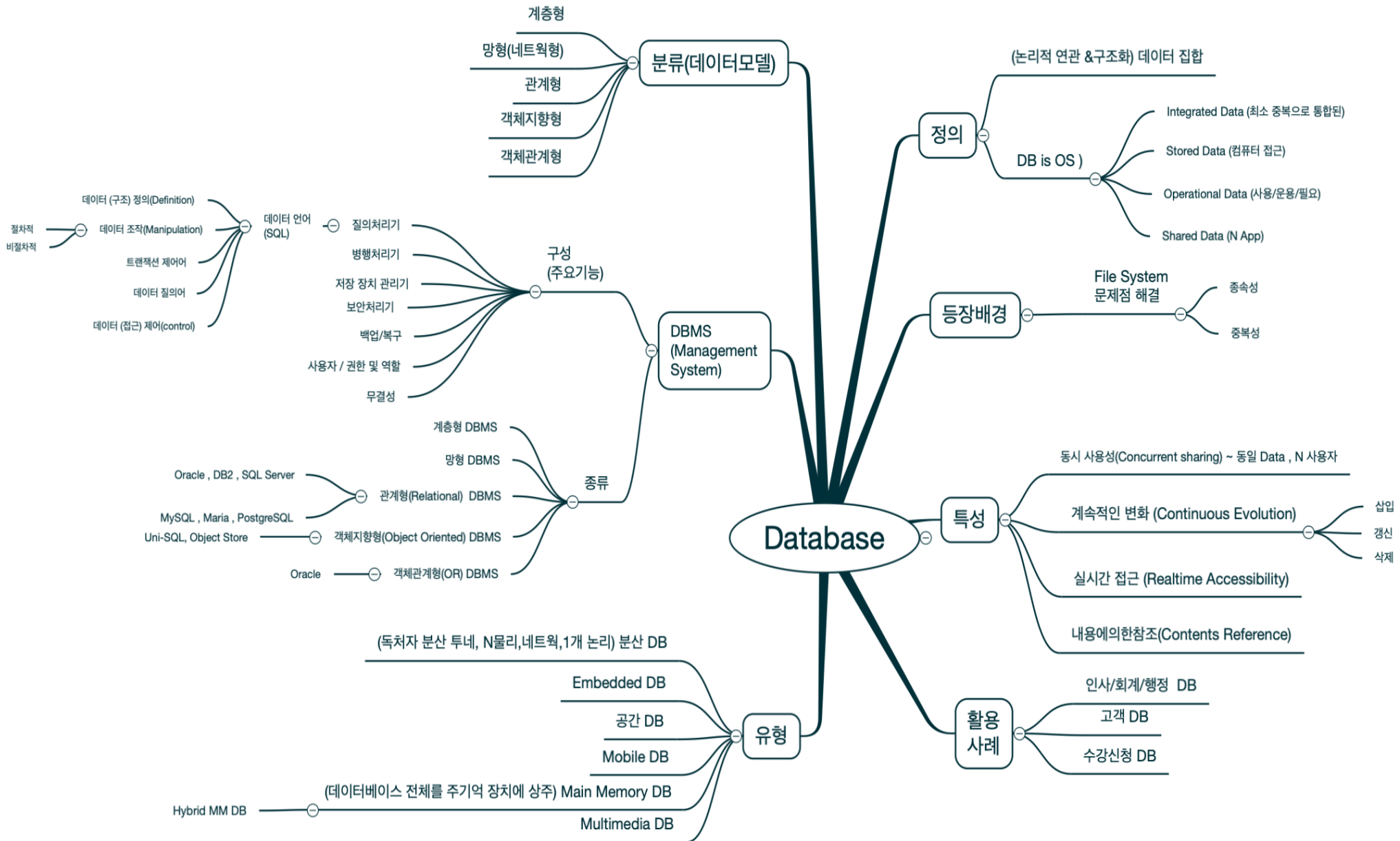
EMPNO	ENAME	JOB	DEPTNO	HIREDATE	SAL
7369	SMITH	CLERK	20	1980-12-17	800
7499	ALLEN	SALESMAN	30	1981-02-20	1600
7521	WARD	SALESMAN	30	1981-02-22	1250
7566	JONES	MANAGER	20	1981-04-02	2975
7654	MARTIN	SALESMAN	30	1981-09-28	1250
7698	BLAKE	MANAGER	30	1981-05-01	2850
7782	CLARK	MANAGER	10	1981-06-09	2450
7788	SCOTT	ANALYST	20	1987-04-19	3000
7839	KING	PRESIDENT	10	1981-11-17	5000
7844	TURNER	SALESMAN	30	1981-09-08	1500
7876	ADAMS	CLERK	20	1987-05-23	1100
7900	JAMES	CLERK	30	1981-12-03	950
7902	FORD	ANALYST	20	1981-12-03	3000
7934	MILLER	CLERK	10	1982-01-23	1300

Excel - Sheet



EMPNO	ENAME	JOB	DEPTNO	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	20	1980-12-17	800		20
7499	ALLEN	SALESMAN	30	1981-02-20	1600	300	30
7521	WARD	SALESMAN	30	1981-02-22	1250	500	30
7566	JONES	MANAGER	20	1981-04-02	2975		20
7654	MARTIN	SALESMAN	30	1981-09-28	1250	1400	30
7698	BLAKE	MANAGER	30	1981-05-01	2850		30
7782	CLARK	MANAGER	10	1981-06-09	2450		10
7788	SCOTT	ANALYST	20	1987-04-19	3000		20
7839	KING	PRESIDENT	10	1981-11-17	5000		10
7844	TURNER	SALESMAN	30	1981-09-08	1500	0	30
7876	ADAMS	CLERK	20	1987-05-23	1100		20
7900	JAMES	CLERK	30	1981-12-03	950		30
7902	FORD	ANALYST	20	1981-12-03	3000		20
7934	MILLER	CLERK	10	1982-01-23	1300		10

## 2. DATABASE



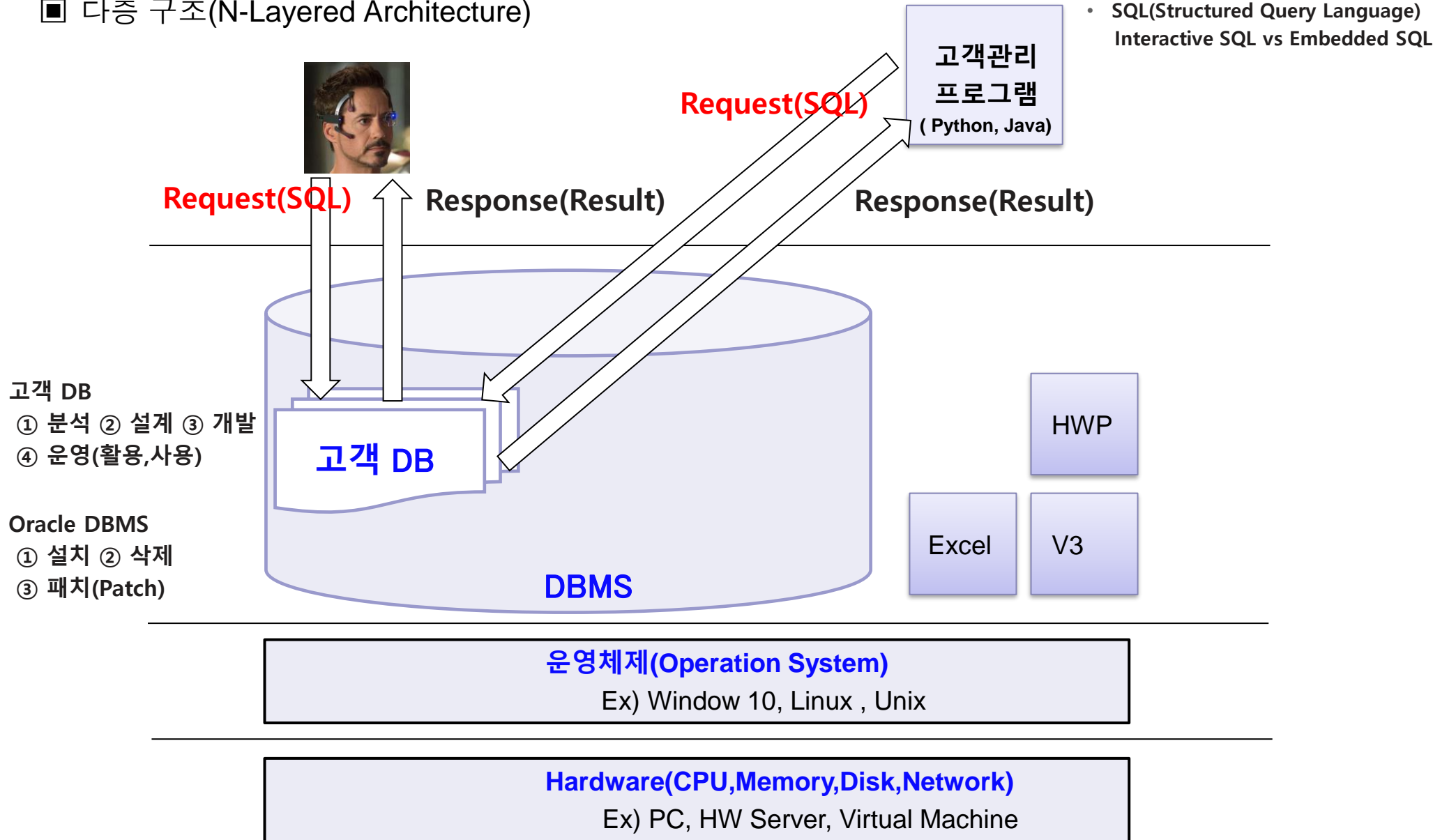


### ● 3. DBMS(DataBase Management System)

정의	<ul style="list-style-type: none"> <li>- 데이터베이스를 <b>관리(Management)</b>하는 <b>소프트웨어(System)</b></li> <li>- 데이터베이스를 생성하여 운영하는데 필요한 기능을 제공하는 소프트웨어(DBMS)</li> </ul>
주요기능	<ol style="list-style-type: none"> <li>1) <b>데이터 구조 정의(Definition) 기능,</b> <b>데이터 조회(SELECT) 및 조작(Manipulation) 기능,</b> <b>트랜잭션 제어(Transaction Control) 기능</b> <b>데이터 접근 제어(Access Control) 기능</b></li> <li>2) 백업(Backup) 및 복구(Recovery) 기능</li> <li>3) 데이터 무결성 및 동시성 제어 기능</li> <li>4) 사용자 및 권한 관리 기능</li> <li>5) 시스템 자원 및 저장 장치 관리 기능</li> <li>6) 사용자 인터페이스 기능</li> </ol>
유형	RDBMS, ORDBMS, OODBMS, Main Memory DBMS, Distributed DBMS, Mobile DBMS
제품군	<ul style="list-style-type: none"> <li>- Oracle DBMS, IBM DB2 DBMS, MS SQL-Server, 알티베이스 DBMS, 티베로(Tibero) DBMS</li> <li>- MySQL DBMS, Maria DBMS</li> <li>- Amazon RDS(Relational Database Service)</li> <li>- MongoDB ?? DB vs DBMS</li> </ul>

### ● 3. DBMS(DataBase Management System)

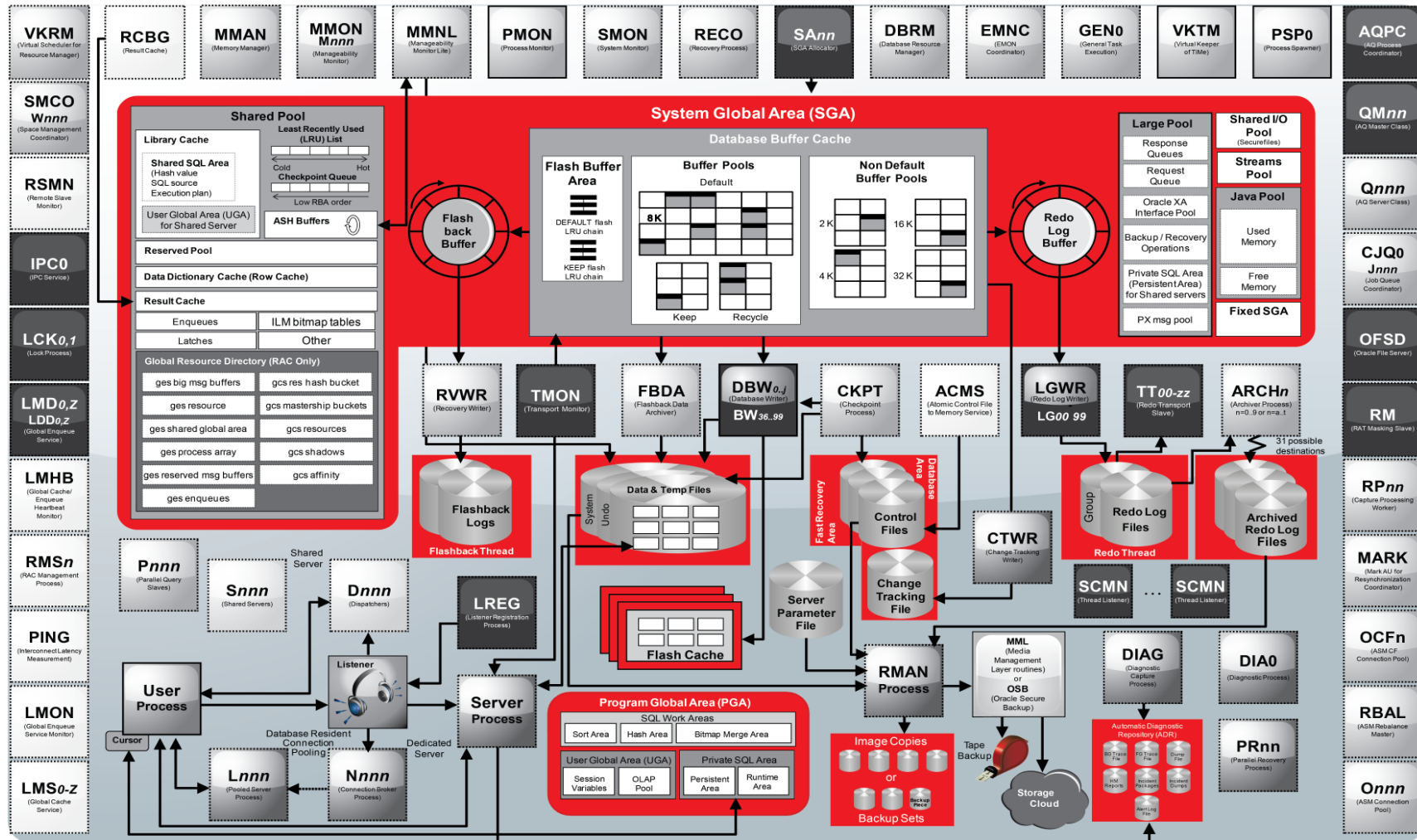
#### ■ 다층 구조(N-Layered Architecture)



# 3. DBMS(DataBase Management System)

## Oracle 12C DBMS 아키텍처

### ORACLE<sup>®</sup> 12<sup>c</sup> DATABASE Architecture Diagram



ORACLE<sup>®</sup>

## ● 4. SQL(Structured Query Language) 개요

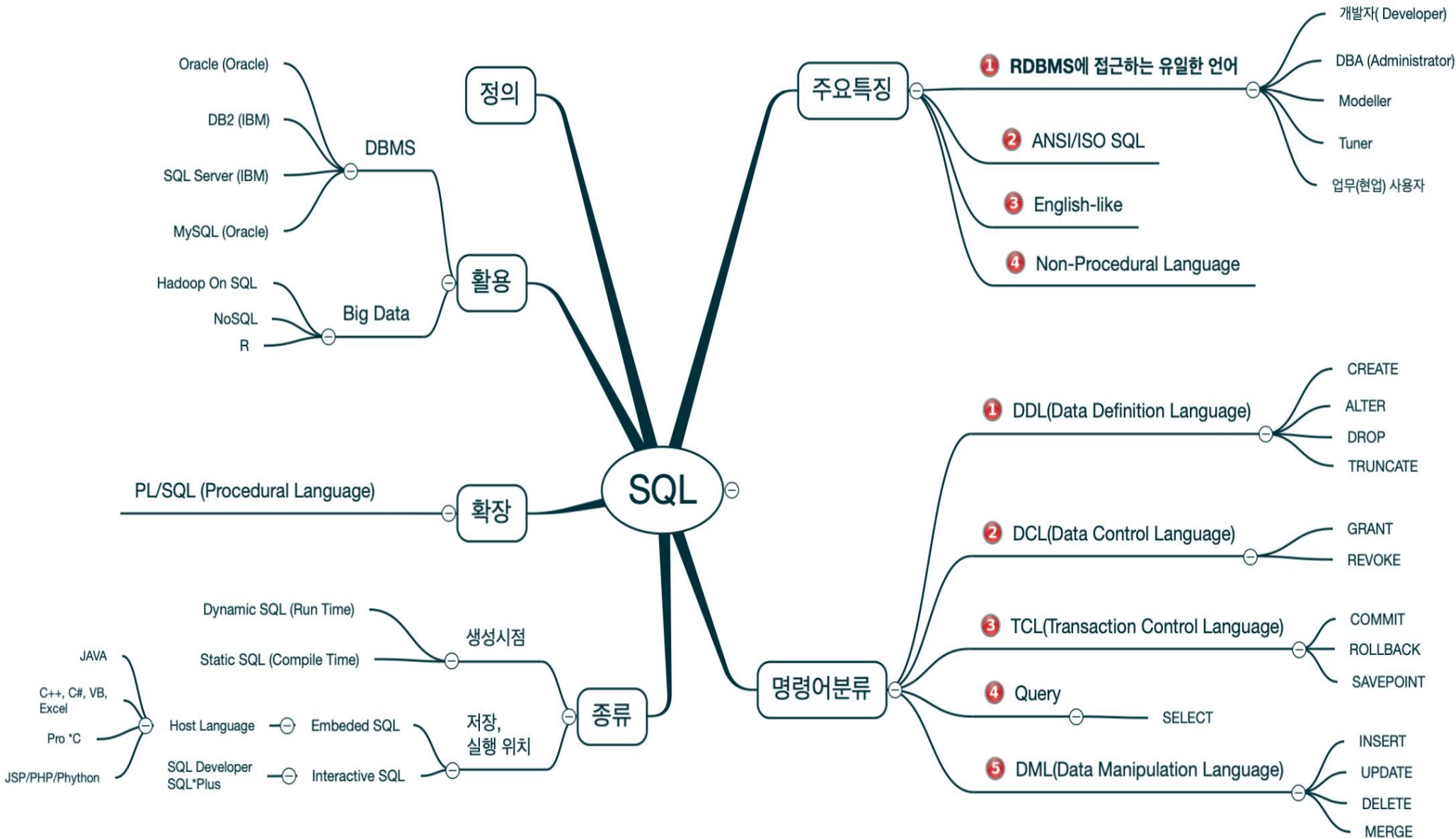
정의	- 데이터베이스에 저장된 데이터에 대한 처리(Manipulation)를 하거나 데이터 베이스 객체(Object)를 정의(Definition) 하고 권한(Privilege)을 제어(Control) 하는 언어
주요특징	<div>① 관계형 <b>DBMS</b>에 접근하는 유일한 언어</div> <div>② <b>ANSI-SQL</b><ul style="list-style-type: none"><li>* 미국 산업 표준화 기구 (ANSI: American National Standards Institute)</li></ul></div> <div>③ <b>English-Like</b><ul style="list-style-type: none"><li>* SQL 명령어의 문법적인 구조 및 의미</li><li>* SQL 명령어는 대-소문자를 구분하지 않는다 (<b>Case-Insensitive</b>)</li></ul></div> <div>④ 비절차적 언어 (<b>Non-Procedural Language</b>)</div>



1974년생 40대 꽃중년

```
SELECT  DEPTNO, EMPNO, ENAME, JOB, SAL
FROM    EMP
WHERE   ENAME = '박보검';
```

## 6. SQL



## ● 6. SQL

분 류	종 류
QUERY	SELECT(데이터 조회)
DML(Data Manipulation Language)	INSERT(데이터 입력), UPDATE(데이터 수정), DELETE(데이터 삭제)
TCL(Transaction Control Language)	COMMIT(트랜잭션 저장), ROLLBACK(트랜잭션 취소), SAVEPOINT(트랜잭션 임시 저장점)
DDL(Data Definition Language)	CREATE(데이터베이스 Object 생성), ALTER(데이터베이스 Object 변경), DROP(데이터베이스 Object 삭제)
DCL(Data Control Language)	GRANT(권한 부여), REVOKE(권한 취소)

주요한 12개의 SQL 명령어들은 용도에 따라 5개의 그룹으로 분류 한다.

DDL , DCL은 데이터베이스의 구조 와 보안을 제어 할수 명령어 이기 때문에 산업현장 에서는 데이터베이스 관리자(DBA)만 사용하고 개발자는 **QUERY,DML,TCL**을 사용한다. 학습과정에서는 5개 그룹의 명령어 대한 이해가 필요하다.

### 가. QUERY(질의어)

데이터베이스에 저장된 데이터를 조회 하는 명령어

### 나. DML(Data Manipulation Language:데이터 처리어)

데이터베이스에 저장된 데이터를 삽입/수정/삭제 하는 명령어

### 다. TCL(Transaction Control Language:트랜잭션 제어어)

데이터베이스에서 발생하는 트랜잭션을 저장/취소하는 명령어

### 라. DDL(Data Definition Language:데이터 정의어)

데이터베이스의 논리적 구조를 정의/변경/삭제 하기 위한 명령어

### 마. DCL(Data Control Language:데이터 제어어)

데이터베이스에 저장된 데이터에 대한 접근 권한을 부여/회수하는 명령어

[참고] 인터넷 게시판을 예로 SQL 명령어 사용 사례를 봅시다

RDBMS에서 모든 데이터는 테이블 구조로 저장된다..

게시판 데이터를 저장하는 테이블을 생성하는 명령어는

DDL 계열의 **CREATE**를 사용한다.

게시글 쓰기는 **INSERT**(데이터 입력) 명령어 사용,

게시글 수정은 **UPDATE**(데이터 수정) 명령어 사용,

게시글 삭제는 **DELETE**(데이터 삭제) 명령어 사용,

게시글 조회는 **SELECT**(데이터 조회) 명령어 사용

여러분이 자주 사용하는 인터넷상의 게시판의

기능은 위의 4개 SQL명령어로 수행된다.

## ● 참고 - RDBMS 이해

### DBMS 발전동향

시 기	데이터 모델	데이터 관리 시스템	제품군
1960년대	파일	파일 시스템	ISAM, VSAM
1960년대	계층형 모델	계층형 DBMS	IMS, System 2000
1970년대	망형(네트워크형) 모델	망형 DBMS	IDS, IDMS
1980년대	관계형 모델	<b>RDBMS (Relational DBMS)</b>	Oracle 7.3, Informix
1990년대	객체지향형 모델	OODBMS (Object-Oriented DBMS)	UniSQL, ObjectStore
2000년대	객체-관계형 모델	<b>ORDBMS (Object-Relational DBMS)</b>	Oracle 8.0 이후버전

소규모 기업 환경에서 데이터를 저장 관리하는 기본적이고 보편적인 방법은 파일을 사용 하는 것이지만 응용 프로그램 과 데이터의 종류가 많아짐에 따라 파일을 사용하게 되는 경우 종속성(응용프로그램에 데이터가 종속적) 과 중복성(데이터간)이 발생하게 되고 이로 인해 데이터 보안성, 데이터 일관성, 경제성, 데이터 무결성의 문제가 발생하게 된다.

파일 시스템의 단점을 해결 하고자 **1970**년대에 **DBMS**가 등장 하게 되었지만 계층형 과 망형 모델의 구조적인 한계로 인해 제한적인 영역에서만 사용되었다.



## ● 참고 - RDBMS 이해

---

1980년대에 이전 단계 **DBMS**의 구조적 문제점을 개선한 관계형 **DBMS**가 개발되었고 관계 모델의 유연성 과 단순성으로 인해 90년대에는 데이터를 관리하는 거의 모든 영역에서 관계형 **DBMS**를 채택 하여 사용하게 되었다

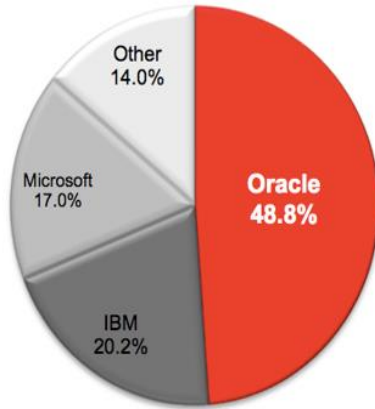
1990년대 객체지향 개념,모델,객체지향 언어를 지원하기 위한 객체지향형(**Object Oriented**) **DBMS**가 개발되었지만 기업, 기관의 대부분 데이터가 **RDBMS**로 구축 되었고 객체지향의 복잡성으로 인해 제한적인 영역에서만 사용되었다.

2000년대에는 2가지 유형의 객체-관계형 **DBMS(Object-Relational DBMS)**가 등장하게 되었는데 객체지향형 **DBMS**를 기반으로 관계형 모델을 추가한 유형과 관계형 **DBMS**를 기반으로 객체 모델을 추가한 유형이다.

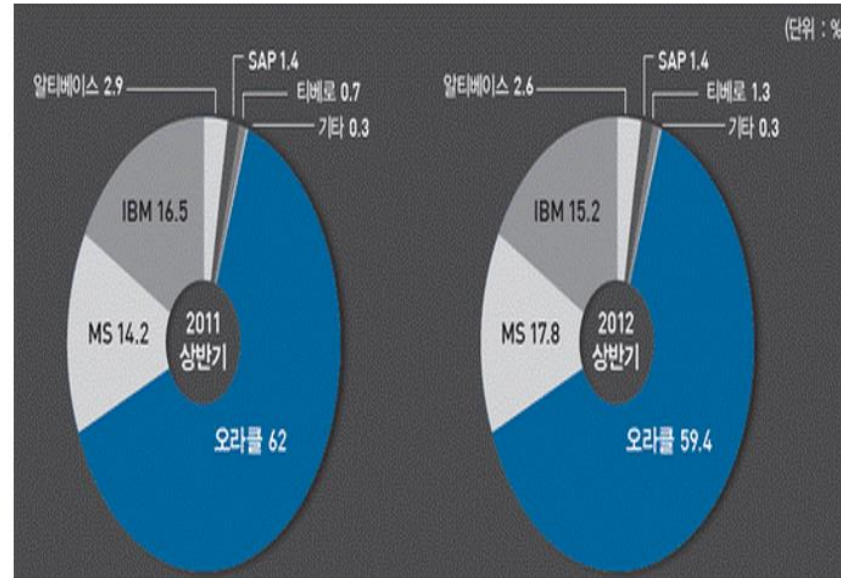
대부분의 기업과 기관에서는 관계형 **DBMS**를 기반으로 객체형 모델을 추가한 객체-관계형 **DBMS**를 사용하고 있다.



## ● 참고 - RDBMS 시장 점유율



2011년 전세계 DBMS 시장 점유율



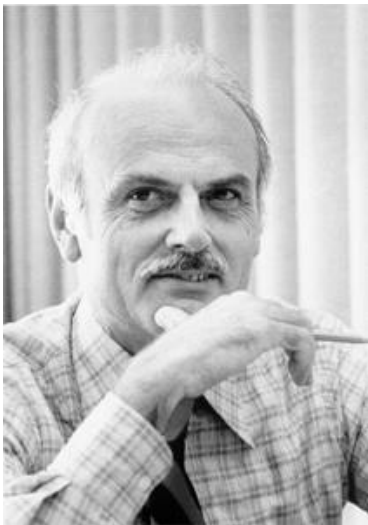
2011/2012년 국내 DBMS 시장점유율 -한국IDC

## ● 참고 - RDBMS 발전

1970년에 IBM연구소에 근무하는 수학자 출신의 E.F Codd 박사가 “A Relational Model of Data for Large Shared Data Banks”라는 논문에서 관계형 모델을 최초로 정의 하였다. 관계형 모델을 기반으로 하여 IBM 연구소가 최초의 RDBMS(관계형 데이터베이스 관리 시스템)을 개발하였고 이때 SEQUEL (Structured English Query Language)이라는 언어를 개발 하였다. 1979년에 Relation Software社(현Oracle社)가 SQL언어를 사용하여 최초의 상업용 RDBMS를 개발했다. 그이 후 다양한 RDBMS가 개발됨에 따라 1986년에 ANSI 위원회와 ISO 위원회가 관계형 데이터베이스의 표준 언어로 SQL(Structured Query Language)을 제정 했다. 현재 사용되는 대부분의 데이터베이스는 관계형 데이터베이스 모델을 기반으로 한다.

[참고] ANSI(American National Standard Institute:미국 표준 협회)

ISO(International Organization for Standardization:국제 표준화 기구)



## ● 참고 - RDBMS 이해

1980년대초 관계형 **DBMS**가 점차 인기를 끌게 되자 망형 또는 계층형 **DBMS**의 일부 기능을 수정하여 관계형 **DBMS** 라고 판매를 하게되자 관계형 데이터베이스 본질이 훼손되는 것을 막기 위해 **E.F Codd**가 제정한 규칙이다. 규칙을 통해 관계형 데이터베이스를 이해해보자.

No	Rule
1	The Information Rule (정보 규칙)
2	Guaranteed Access Rule (보장된 접근 규칙)
3	Systematic Treatment of NULL Values (널 값의 체계적인 처리)
4	Dynamic Online Catalog Based on the Relational Model (관계형 모델 기반 동적 온라인 카탈로그)
5	Comprehensive Data Sublanguage Rule(종합적인 데이터 보조언어 규칙)
6	View Updating Rule(뷰 갱신 규칙)
7	High-level Insert, Update and Delete (고수준 삽입, 갱신, 제거)
8	Physical Data Independenc (물리적 데이터 독립성)
9	Logical Data Independence (논리적 데이터 독립성)
10	Integrity Independence (무결성 독립성)
11	Distribution Independence (분산 독립성)
12	Non-Subversion Rule (비전복 규칙)

[표 3.1] E.F Codd의 관계형 데이터베이스 12가지 규칙

## ● 참고 - RDBMS 이해

---

E.F Codd박사의 12가지 규칙중 아래의 5가지 규칙을 관심있게 읽어 두자.

Rule 1 The Information Rule (정보 규칙)

데이터베이스내의 모든 정보는 한가지 방법(테이블)으로만 표현되어야 한다.

즉 행(Row)과 열(Column)로 구성된 테이블에 저장되는 값으로 표현 되어야 한다.

Rule 2 Guaranteed Access Rule (보장된 접근 규칙)

데이터베이스내의 모든 데이터는 테이블명, 컬럼명, 기본키(Primary Key)를 통해 접근 가능해야 한다.

Rule 3 Systematic Treatment of NULL Values(체계적인 Null값 처리)

Null 값을 다루기 위한 체계적인 지원을 해야한다.

Rule 5 Comprehensive Data Sublanguage Rule(종합적인 데이터 지원 언어 규칙)

다음의 특징을 가지는 관계형 언어를 지원해야 한다.

- ① 대화식으로 사용 될수 있고, 어플리케이션 프로그램 안에서 사용될수 있다.
- ② 데이터 정의(Definition)를 할수 있어야 한다
- ③ 데이터 처리(Manipulation)를 할수 있어야 한다.
- ④ 트랜잭션 제어(control)를 할수 있어야 한다.
- ⑤ 보안(Access) 과 무결성(Integrity) 제약 조건을 지원해야 한다.

Rule 10 Integrity Independence (무결성 독립성)

무결성 제약 조건의 정의(definition)는 어플리케이션 프로그램들과는 별도로 수행할수 있고 데이터베이스 카탈로그에 저장 되어야 한다. 무결성 제약 조건 변경은 기존 어플리케이션에 영향을 주지 않고 수행되어야 한다.

## ● 참고 - RDBMS 장점

계층형,망형 데이터베이스에 비해 관계형 데이터베이스는 장점은

① 모델 단순성 ② 독립성 과 유연성 ③ 비절차적 **SQL** ④ 수학적집합론 이다.

### ● 모델 단순성

관계형 모델은 데이터의 구조적인 표현이

행(Row) 과 열(Column)로 이루어진 2차원 테이블

(Rule 1)형태로 구성됨으로 구조적으로 단순하며

데이터의 접근이 매우 편리함에 따라

데이터베이스 분석가,설계자, 개발자,일반 사용자가  
쉽게 이해하고 활용 할수 있는 장점을 제공한다.

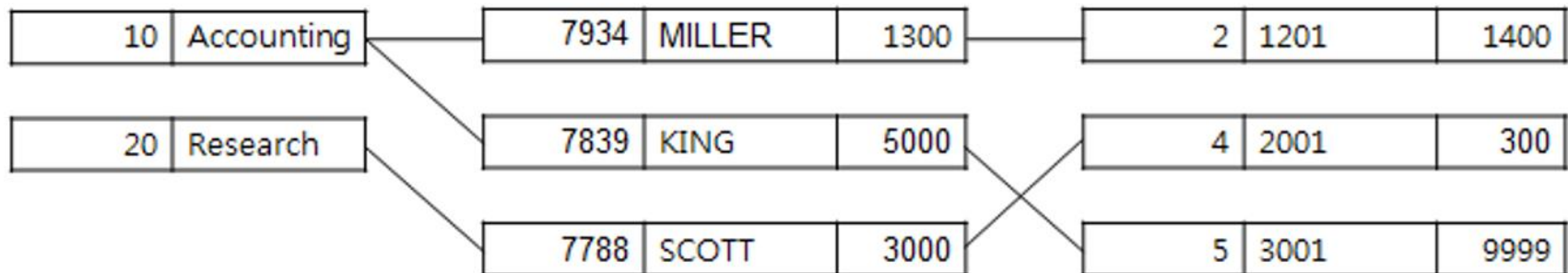
EMPNO	ENAME	JOB	DEPTNO	HIREDATE	SAL
7369	SMITH	CLERK	20	1980-12-17	800
7499	ALLEN	SALESMAN	30	1981-02-20	1600
7521	WARD	SALESMAN	30	1981-02-22	1250
7566	JONES	MANAGER	20	1981-04-02	2975
7654	MARTIN	SALESMAN	30	1981-09-28	1250
7698	BLAKE	MANAGER	30	1981-05-01	2850
7782	CLARK	MANAGER	10	1981-06-09	2450
7788	SCOTT	ANALYST	20	1987-04-19	3000
7839	KING	PRESIDENT	10	1981-11-17	5000
7844	TURNER	SALESMAN	30	1981-09-08	1500
7876	ADAMS	CLERK	20	1987-05-23	1100
7900	JAMES	CLERK	30	1981-12-03	950
7902	FORD	ANALYST	20	1981-12-03	3000
7934	MILLER	CLERK	10	1982-01-23	1300

## ● 참고 - RDBMS 장점

### ● 모델 독립성 과 유연성

기존의 계층형 또는 망형 모델의 데이터 구조는 레코드간에 직접 연결하는 방식으로 되어 있는 종속성으로 인해 논리적 저장 구조나 물리적 구조가 변경되면 어플리케이션 프로그램을 변경 해야 하지만 관계형 모델은 논리적 독립성(**Rule 8**), 물리적 독립성(**Rule 9**)을 제공 함에 따라 논리적 물리적 구조 변경시 독립성으로 인해 어플리케이션 프로그램 수정을 최소화한다. 개발 진행중, 유지보수시 빈번히 발생하는 데이터 구조 변경에 유연하게 대처할수 있어 개발 생산성 및 유지보수 효율성을 가져온다.

아래의 그림처럼 계층형이나 망형 데이터베이스에서는 데이터끼리 실제 포인터(주소 참조)를 통해 물리적으로 연결되어 구조화 된다. 포인터로 데이터간 연결되어 데이터간 참조시 빠른 검색이 가능한 반면 논리적/물리적 구조 변경시 어플리케이션 프로그램 수정, 관련된 모든 데이터 연결을 변경 해야하기에 유연하지 않다.





## ● 참고 - RDBMS 장점

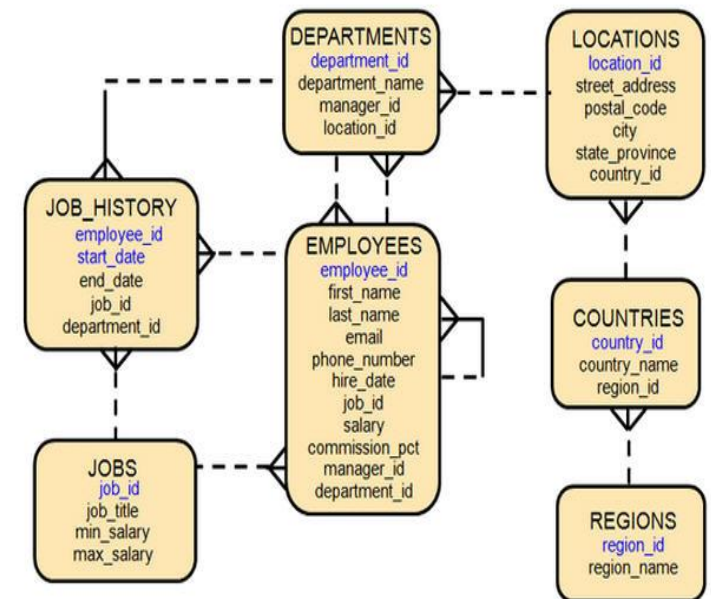
관계형 데이터베이스는 데이터간 물리적 연결 없이 독립적으로 존재 하다가 데이터간 연결이 필요할때 내용에 의한 참조(**Content Reference**)를 한다.아래의 그림은 관계형 **DBMS**의 테이블간 관계도(**Relationship Diagram**)이다. 각각의 테이블은 독립적으로 존재하다가 데이터간 연결이 필요할 때 테이블간 연결성에 따라 실시간으로 관계를 형성하여 결과를 처리한다.

테이블간에 연결된 점선,실선은 데이터의 물리적 연결을 의미 하는것이 아니라 테이블간 관계성을 표현 한것이다.

[주의] 테이블간의 관계성에 대한 정확한 표현은 각각의 테이블내에 존재하는 데이터간 필수적,선택적 관계성이다.

실시간으로 테이블간 관계를 형성하는 것을 **JOIN** 연산이라고 한다. **JOIN**은 관계형 데이터베이스의 가장 중요한 연산중 하나이다. 기억해두자.

[참고] 계층형,망형 데이터베이스는 연관된 데이터간에 물리적으로 연결되는 종속성으로 인해 유연성이 낮은 반면 연관된 데이터 검색시 빠르다. 관계형 데이터베이스는 데이터간 물리적 연결 없는 독립성으로 인해 사용의 편의성,개발 생산성, 유지보수 용이성이 높은 반면 연관된 데이터 검색시 실시간으로 **JOIN** 연산 수행을 통해 데이터간을 연결하기에 상대적으로 느리고 이는 관계형 **DBMS**의 최대 단점이 되었다. **JOIN** 연산은 관계형 데이터베이스 구조에서는 필수적으로 사용되는 가장 중요한 연산이지만 이런 단점으로 인해 관계형 **DBMS**는 성공 할수 없다는 예측도 있었고 초기에는 소규모 환경에서만 사용되었다. 관계형 데이터베이스가 국내에 도입된 초기에는 **JOIN** 연산을 금지시키는 IT 조직도 있었다. H/W 성능 향상, **RDBMS** 엔진 최적화(**Optimizing**),SQL 성능 향상 기법의 발달로 인해 **JOIN**의 구조적인 성능 문제가 해결되었고 현재 사용되는 대부분의 **DBMS**는 **RDBMS**이다. **JOIN**의 구조적인 성능 문제는 해결되었지만 효율적인 **JOIN** 연산 사용은 여전히 개발자의 몫이다.



## ● 참고 - RDBMS 장점

---

### ● 비절차적 SQL(Structured Query Language)

계층형,망형 데이터베이스는 절차적인 어플리케이션 프로그래밍을 통해서 데이터 조회 및 처리를 수행한다.이로 인해 기업내의 다양한 데이터 사용자는 IT 조직의 지원을 받아서 개발을 해야지만 데이터를 활용 할수 있었다.

관계형 데이터베이스는 업무 담당자, 비즈니스 담당자등 비IT 조직의 구성원들도 쉽게 데이터를 활용 할수 있도록 비절차적인 **SQL**을 제공하여 누구나 쉽게 데이터를 활용 할수 있도록 되었고 이로 인해 관계형 데이터베이스가 보편적으로 널리 사용 되었다

(Rule 5 ①)

### ● 수학적 집합론

관계형 데이터베이스 모델 창시자인 **E.F Codd** 박사는 수학자로써 집합 이론과 1차 술어 논리라는 수학을 기반으로 관계형 모델 이론을 만들었다. 관계형은 수학의 관계형 이론(**relational theory**)으로 부터 유래한다. 관계형 이론은 이차원의 행렬(**matrix**) 구조의 정보에 가해지는 다양한 연산들의 특성을 수학적으로 정의하고 있다. 수학적인 기반은 개발할 시스템의 성능을 수학적으로 미리 예측할 수 있거나 여러 연산을 수학적으로 최적화할 수 있는 장점이 있다. 현재 관계형 데이터베이스가 제공하는 질의 기능, 질의 최적화 기능, 인덱싱 기능 등은 수학적 기반을 통해 제공하는 것이다.



## ● 참고 - SQL 개요

---

SQL을 통해 데이터베이스에 저장된 데이터에 대한 처리(**Manipulation**)를 하거나 데이터베이스 객체(**Object**)를 정의(**Definition**) 하고 권한(**Privilege**) 제어(**Control**)를 한다.

### SQL 주요 특징

#### ① 관계형 **DBMS**에 접근하는 유일한 언어

데이터베이스와 관련된 IT 직군은

데이터베이스를 분석,설계하는 모델러(**Modeller**)

개발 업무를 수행하는 개발자(**Developer**),

**DBMS**를 관리하는 관리자(**Database Administrator**),

느려진 데이터베이스 성능을 개선하는 튜너(**Tuner**)등이 있습니다.

개발자는 **SQL**을 사용하여 개발을 합니다.

DBA는 **SQL**을 사용하여 **DBMS**를 관리합니다.

모델러는 **SQL**을 사용하여 물리적 모델을 **DBMS**내에 생성 합니다.

튜너는 **SQL**을 사용하여 튜닝 업무를 수행 합니다.

관계형 **DBMS**에 접근하는 유일한 언어가 **SQL** 이기 때문에

**DBMS** 관련 업무를 수행하는 모든 사람들은 **SQL**을 통해서만 업무를 수행할수 있습니다.

**SQL**은 **DBMS**관련 모든 분야에서 사용된다.

### ② ANSI-SQL

ANSI(American National Standards Institute)는 단어 의미 그대로 미국 산업 표준화 기구 이다.

RDBMS 관련 연구소/기업들이 **SQL**을 채택하고 각기 자신만의 고유한 **SQL** 기능을 추가하여 다양화 되자 표준 **SQL** 제정의 필요에 따라 **1986**년에 **ANSI-SQL**을 최초로 제정했다.

ANSI-SQL은 개발 입장에서는 어떤 장점이 있을까요 ?

(1) SQL 한번만 배우면 모든 **RDBMS**에서 사용할수 있다.

(2) Oracle RDBMS에서 개발된 솔루션(프로그램)을 **IBM RDBMS**로 쉽게 옮길수 있다

(1) Oracle RDBMS의 **SQL**에 익숙한 개발자는**IBM RDBMS**로 프로젝트를 진행 할 IBM RDBMS의 **SQL** 문법을 참고하면 쉽게 전향 할수 있습니다. SQL을 깊이 있게 잘 익혀두고 **ANSI-SQL**에 대한 개념을 가지고 있다면 개발자로서의 가치가 높아 지겠죠.

(2) 모든 **RDBMS**에서 **ANSI-SQL**을 지원하지만 각 업체만의 기능을 추가 하게 됩니다.      **ANSI-SQL** 만을 사용하여 구현을 한다면 타 **RDBMS**로 쉽게 이행이 되지만,실제 각 **RDBMS** 업체의 고유 기능들에 각기의 장점이 있고 **DBMS**별로 최적화 방안이 달라 특정 **RDBMS**에 종속된 **SQL**을 사용하게 되어 **RDBMS**간의 이전은 생각보다 어렵게 됩니다.

### ② ANSI-SQL

ANSI-SQL 2003(SQL4)는 2003년도에 표준 제정을 했으며 SQL4 또는 ANSI-SQL 2003으로 표기한다는 의미입니다. 시대적으로 필요한 기술이 늘어나면서 주기적으로 표준 제정을 통해 새로운 기능을 추가해 가고 있습니다. 현재까지 8차 ANSI-SQL 표준화 재정이 진행 되었으며 차수별 특징은 다음과 같다

년도	명칭	별칭	설명
1986	SQL-86	SQL-87	ANSI / ISO 에 의해 최초 표준화 정의
1989	SQL-89	FIPS127-1	마이너 개정, integrity constraints가 추가, FIPS 127-1에서 채택.
1992	SQL-92	SQL2	메이저 개정 (ISO 9075), Entry Level SQL-92은 FIPS 127-2로 채택.
1999	SQL : 1999	SQL3	정규 표현식 매칭 추가, 재귀 쿼리 (예, 이행적 폐쇄), 데이터베이스 트리거, 절차적, 통제흐름 구문 지원, 비규격 타입 그리고 객체지향형 특징 지원(예, 구조화 타입). 자바에서 내장 SQL 지원(SQL/OLB) 그리고 (SQL/JRT).
2003	SQL : 2003	SQL4	XML 관련 특징 도입 (SQL/XML), window functions, 자동 생성값에 대한 표준화된 시퀀스와 컬럼(아이덴티티 컬럼 포함).
2006	SQL : 2006	SQL5	ISO/IEC 9075-14:2006은 XML과 결합되어 SQL이 사용되는 방법을 정의하고 있다. 여기에는 SQL 데이터베이스 내의 XML 데이터의 불러오기와 저장하는 방법을 정의하고 있으며, 데이터베이스 내에서 조작하여 XML 형식의 전통적 SQL 자료와 XML 형식 모두로 출력하는 방법을 제시한다. 게다가, 여기에서는 W3C에 의해 제안된 XML 쿼리 언어, Xquery를 이용하여 SQL 코드로 애플리케이션을 통합할 수 있도록 하여, 보통의 SQL 데이터와 XML 문서에 접근할 수 있게 한다.
2008	SQL : 2008	SQL6	커서 정의 외부의 ORDER BY를 합법화, INSTEAD OF 트리거 추가, TRUNCATE 구문 추가.
2011	SQL : 2011	SQL7	임시 데이터베이스에 대한 지원 향상

-인용 <http://ko.wikipedia.org/wiki/SQL>

### ③ English-Like

SQL의 원래 명칭은 **SEQUEL(Structured English QUery Language)** 이고 이중 **English** 라는 단어에 주목 하시기 바랍니다. 영어권 사람들이 쉽게 데이터에 접근하고 관리 하도록 만들다 보니 **SQL**에 영어적인 특징이 들어가 있다.

English-Like의 2가지 특징입니다.

- SQL 명령어의 문법적인 구조나 의미가 영어 문맥과 유사 하다
- SQL 명령어는 대-소문자를 구분하지 않는다.(**Case-Insensitive**)

① **SQL, PL/SQL** 명령어는 대소문자를 구분하지 않는다.

Ex)           SELECT \* FROM emp where job = 'PRESIDENT';  
                SELECT \* FROM emP where job = 'PRESIDENT';  
                SELECT \* FROM EMP WHERE JOB = 'PRESIDENT';

② 데이터는 대-소문자를 구분한다. 아래의 예제는

SQL 관점에서 동일한 명령어 이지만 다른 데이터를 처리 합니다.

Ex)           SELECT \* FROM emp where job = 'PRESIDENT' ;  
                SELECT \* FROM emp where job = 'president' ;

### ④ 비절차적 언어

비절차적(**Non-Procedural**)이란 단어적인 해석을 해보면 글자 그대로 처리 절차가 없다는 의미 이다.  
절차적 언어란?

개발자(사용자)가 처리절차(처리 순서/방법)를 처음부터 끝까지 정해 주어야하는 언어로  
**COBOL,C,JAVA**가 대표적 언어 이다.

비절차적 언어란?

개발자(사용자)가 처리 절차(처리 순서/방법)를 지정하지 않고 원하는 결과(**Whay**)를  
정의하여 요청하는 언어로 **SQL**이 대표적 언어이다.

SQL의 가장 중요한 특징중 하나인 비절차적 언어라는 특징 이해할 필요가 있다. SQL을 깊게 이해하고 효율적인  
**SQL**을 구사하는데 필요한 중요한 개념이기 때문이다. 비절차적(**Non-Procedural**)은 처리방법,순서를 명기하지  
않고 원하는 조건을 정의 하는 것이다. 아래의 **SQL**예제를 통해 이해 해보자.

2000만원 이상의 급여를 받는 사원의 이름,입사일,급여를 조회 하는 **SQL**이다.

```
SELECT 이름,입사일,급여 FROM 사원테이블 WHERE 급여 >= 2000;
```

① 사원 테이블로 부터(**FROM**)

② 급여가 **2000**만원 이상의 조건(**WHERE**)을 만족하는

③ 이름,입사일,급여를 조회(**SELECT**)

C 언어 또는 **Java**언어로 위의 기능을 개발한다면 어떻게(**How**) 처리 할것인지를 프로그래밍  
해야 하지만 **SQL**로 개발 할때는 원하는 것이 무엇(**What**)인지를 명기한다.

## ● 과제

---

- 1) DB vs DBMS 개념 및 차이점 문서로 정리(개인폴더에)
- 2) Oracle SE(Standard Edition) 과 EE(Enterprise Edition) 차이점 문서 정리
- 3) Client/Server 구조도 및 최신 사례 문서 정리
- 4) SQL 주요 특징 4가지 암기 및 구두 설명
- 5) SQL 명령어 분류표 암기 및 구두 설명
- 6) SELECT 문법구조를 Oracle 메뉴얼에서 검색후 문서 정리