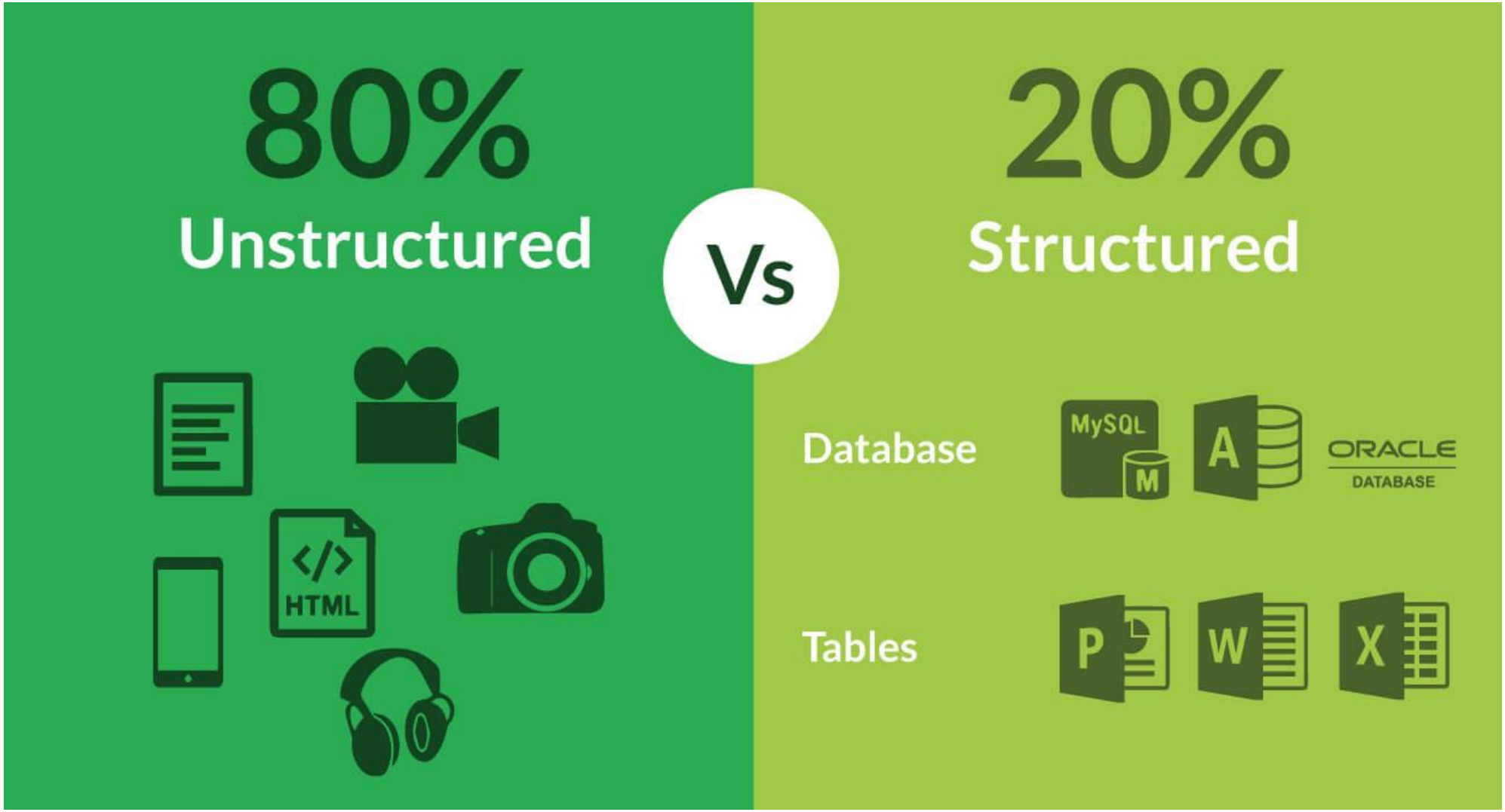
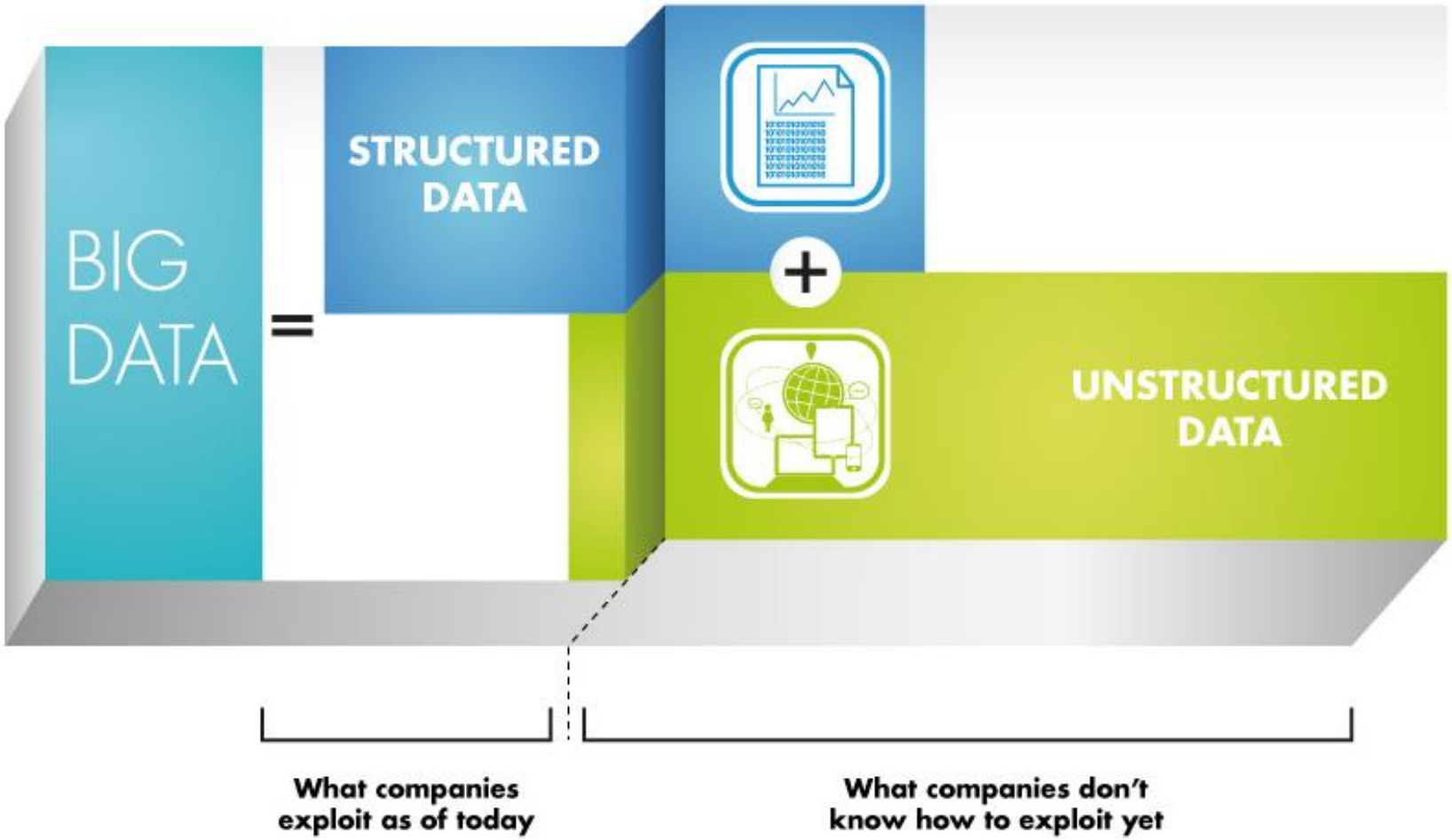
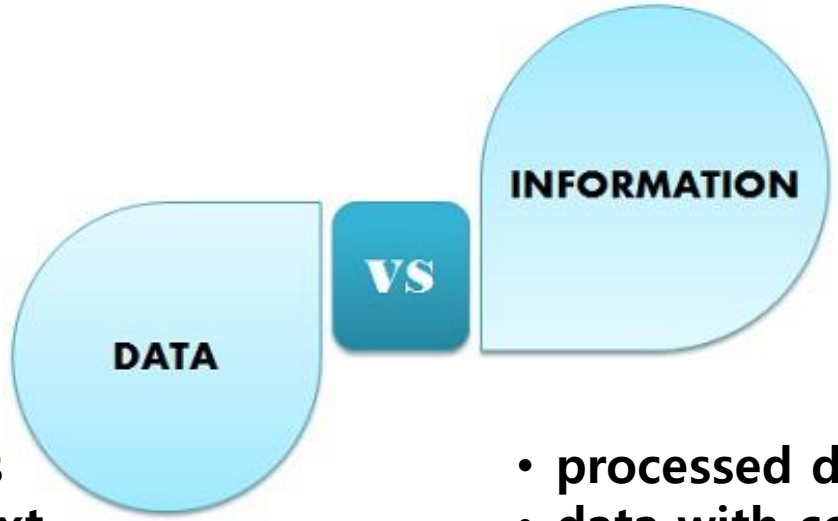
1. DBMS

1.1. 데이터베이스 개요
1.2. 데이터베이스 설계
1.3. SQL
1.4. SQLite

1.1. 데이터베이스 개요


80%
Unstructured
Vs
20%
Structured
HTML
Database
MySQL
ORACLE
DATABASE
Tables


BIG DATA
=
STRUCTURED DATA
+
UNSTRUCTURED DATA
What companies exploit as of today
What companies don't know how to exploit yet



- **raw facts**
- **no context**
- **just numbers and text**

business transaction, sales, payments, etc.

- **processed data**
- **data with context**
- **value-added data**

customer balance, sales trends, etc.

➡ ***make decisions***

Database

- Collection of data organized in a manner that allows access, retrieval, and use of that data

Data

- Collection of unprocessed items
  - Text
  - Numbers
  - Images
  - Audio
  - Video

Information

- Processed data
  - Documents
  - Audio
  - Images
  - Video

structured

■ What is Database?

○ 데이터베이스

- a collection of interrelated data
- a set of integrated, stored, shared and operational data

■ 데이터베이스 조건

○ 통합(Integrated)

- 동일한 데이터가 중복되지 않도록 구성
- 최소한의 중복 또는 통제된 중복 만 허용

○ 저장(Stored)

- 컴퓨터로 접근 가능한 물리적 저장 매체 저장

○ 공유(Shared)

- 공동으로 소유하고 유지하며 이용하는 데이터

○ 운영(Operational)

- 존재 목적이나 기능 수행에 꼭 필요한 데이터 집합

■ 데이터베이스 특징

○ 실시간 접근성(Real-time Accessibility)
- ➢ 데이터들 간의 밀접한 관계로 연결
- ➢ 중복 데이터를 배제하도록 지양
- ➢ 사용자의 어떤 요구에도 즉각 응답

○ 계속적인 변화(Continuous evolution)
- ➢ 현실 세계의 상태를 정확히 반영
- ➢ 동적으로 삽입/삭제/수정하여 현재의 데이터 유지

○ 동시 공유 가능(Concurrent sharing)
- ➢ 여러 사용자들이 동시에 이용
- ➢ 같은 시간에 같은 데이터에 접근하여 이용

○ 내용에 의한 참조 가능(Content reference)
- ➢ 저장된 주소나 위치에 의해서 참조하지 않고
- ➢ 사용자가 요구하는 데이터의 내용/값에 따라 참조

■ What is DBMS?

○ DBMS

- ➢ a collection of programs
- ➢ manage the database structure
- ➢ controls access to the data stored in the database

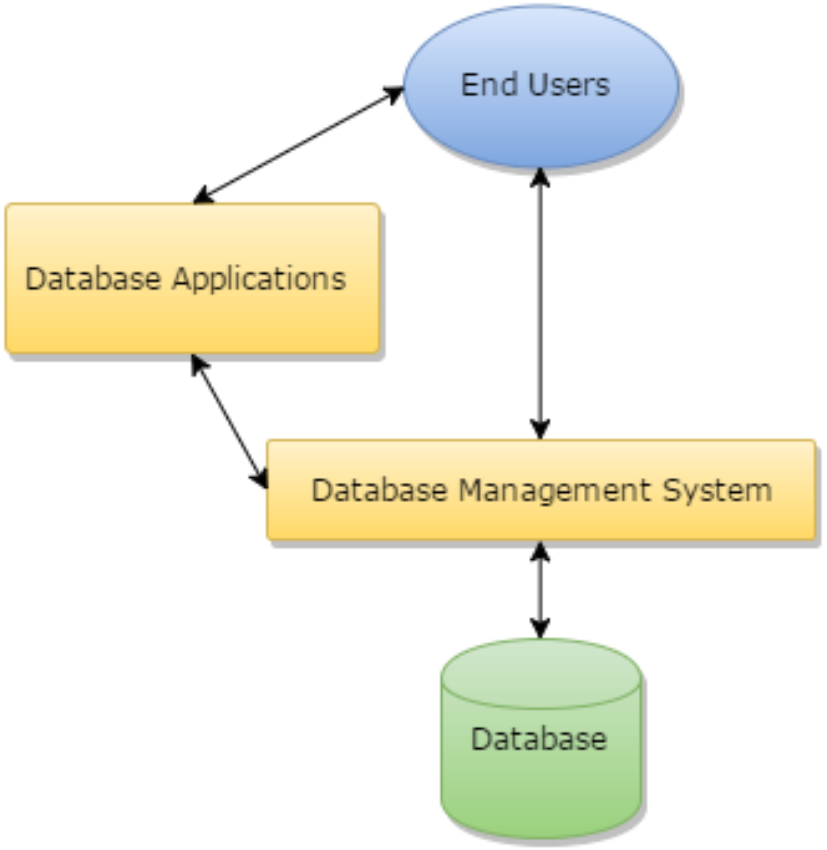
■ DBMS의 역할

○ create databases

○ insert, update and delete data

○ sort and query data

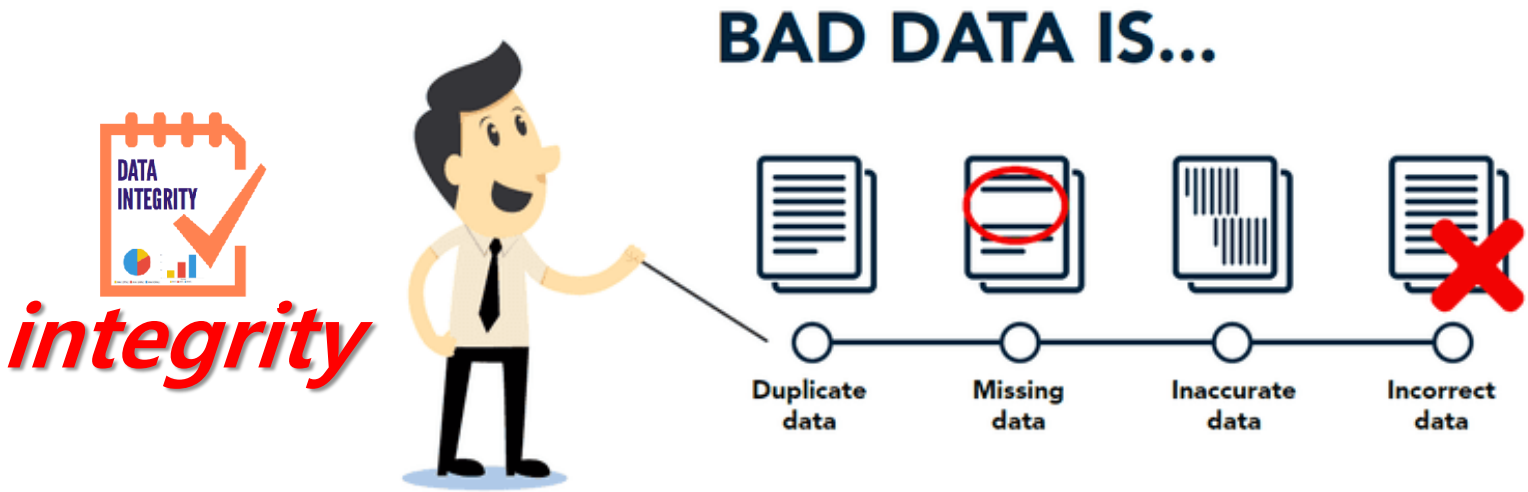
○ create form and report



Components of a Database Management System

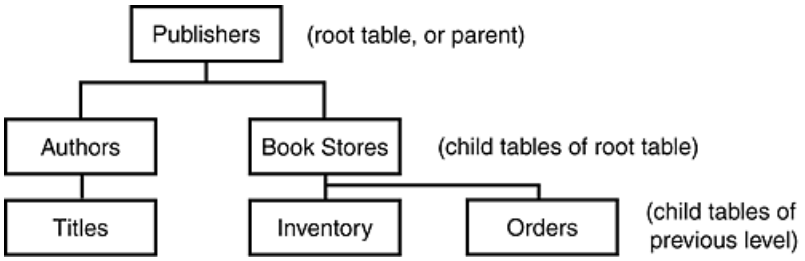
■ DBMS 역할과 장점

- ○ Improved data sharing
- ○ Improved data security
- ○ Minimized data inconsistency
- ○ Improved data access
- ○ Improved decision making
- ○ Increased end-user productivity
- ○ Reduce application development time

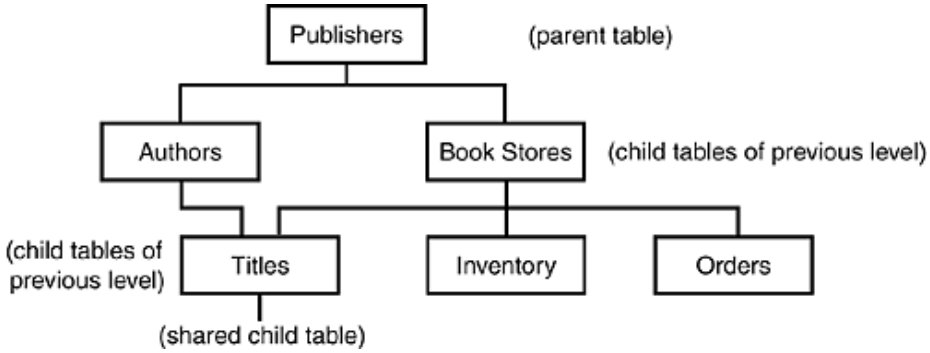


■ DBMS 종류

○ 계층형 모델(Hierarchical Database Model)
- ➢ 트리 형태로 표현 (1:N)
- ➢ 개체를 노드, 개체 집합들 사이의 관계를 링크로 표현

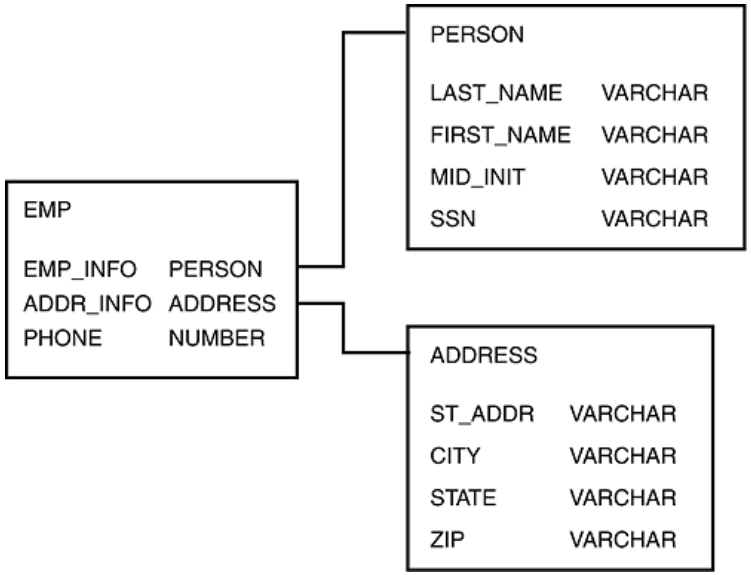


○ 네트워크형 모델(Network Database Model)
- ➢ 그래프 형태로 표현 (N:M)
- ➢ 각 개체가 여러 관계를 가질 수 있음



○ 관계 데이터 모델(Relational Database Model)
- ➢ 개체를 테이블, 개체 집합들 사이의 관계를 공통 속성으로 연결

○ 객체 관계 모델(Object Relational Database Model)
- ➢ 속성-관계
- ➢ 개체-관계



1.2. RDBMS

■ Why use an RDBMS?

○ 데이터 안정성(Data Safety)

- ➢ data is immune to program crashes

○ 동시 접근성(Concurrent Access)

- ➢ atomic updates via transactions

○ 장애 허용성(Fault Tolerance)

- ➢ replicated DBS for instant failover on machine/disk crashes

○ 데이터 무결성(Data Integrity)

- ➢ aids to keep data meaningful

○ 데이터 확장성(Scalability)

- ➢ can handle small/large quantities of data in a uniform manner

○ 데이터 보고서(Reporting)

- ➢ easy to write SQL programs to generate arbitrary reports

■ RDBMS Concepts

○ Relation / Table

○ Tuple / Row or Record

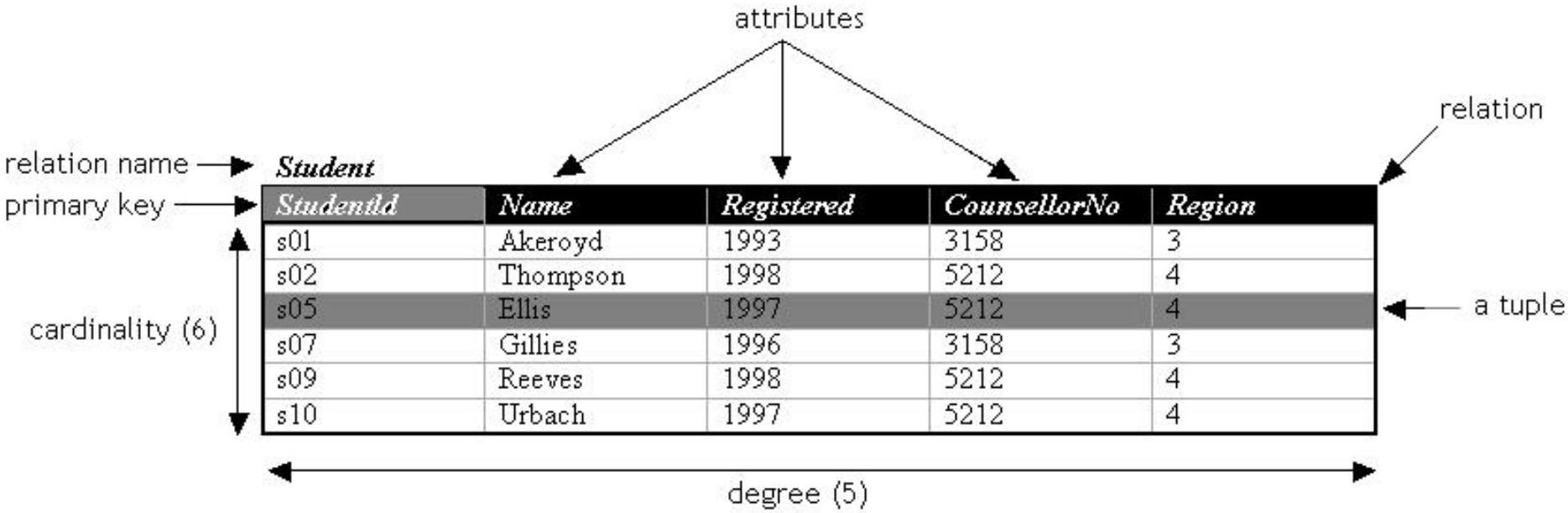
○ Attribute / Column or Field

○ Cardinality / Number of Rows

○ Degree / Number of Columns

○ Domain / Pool or legal values

○ Primary Key / Unique identifier



Student

stu_no	stu_name	stu_ename	tel
20001001	김유신	Kim Yoo-Shin	061)685-7818
20001015	박도준	Park Do-Jun	061)744-6126
20001021	이상길	Lee Sang-Gil	031)691-5423
20041002	김유미	Kim Yoo-Mi	061)763-1439
20041007	정인정	Jeung Yin-Jeung	061)723-1078
20041033	연개소문	Yean Gae-So-Moon	061)642-9304
20061011	박정인	Park Jung-In	02)652-2439
20061014	고혜진	Ko Hea-Jin	061)781-5135
20061048	김영호	Kim Young-Ho	062)548-8881
20071001	장수인	Jang Soo-In	061)791-1236
20071010	홍길동	Hong Gil-Dong	061)642-4034
20071022	이순신	Lee Sun-Shin	061)745-7667
20071300	유하나	Yoo Ha-Na	061)651-5992
20071307	김문영	Kim Moon-Young	061)745-5485

■ 개체 관계 모델(Entity-Relationship Model)

○ 개체(Entity)

- ➢ 실 세계에 존재하는 분리된 실체 하나를 표현, 일반적으로 명사 하나에 해당

○ 관계(Relationship)

- ➢ 개체들 사이에 존재하는 연관이나 연결, 일반적으로 동사에 해당
- ➢ 최소 대응수(minimum cardinality)와 최대 대응수(maximum cardinality)로 구성

○ 속성(Attribute)

- ➢ 개체의 성질, 분류, 식별, 수량, 상태 등을 나타내는 세부 항목
- ➢ 관계 또한 속성을 보유할 수 있음

○ 기본키(Primary Key)

- ➢ 모든 개체를 고유하게 식별할 수 있는 속성

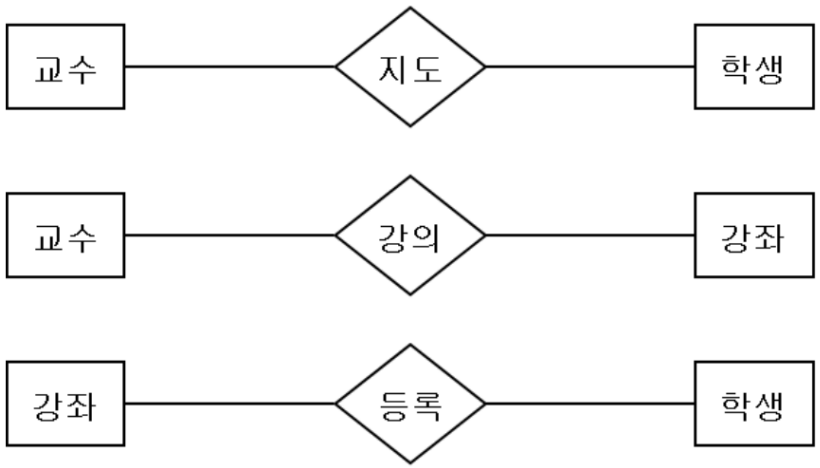
■ 예제: 교수-학생 ER모델

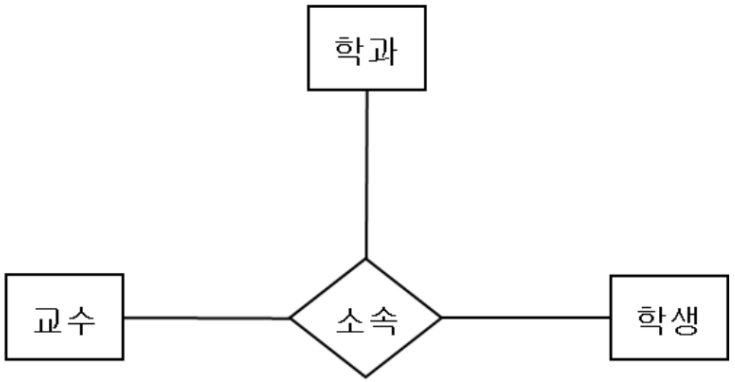
○ 이항 관계

- 교수는 학생을 지도한다.
- 교수는 강좌를 강의한다.
- 강좌는 학생들이 등록한다.

○ 삼항 관계

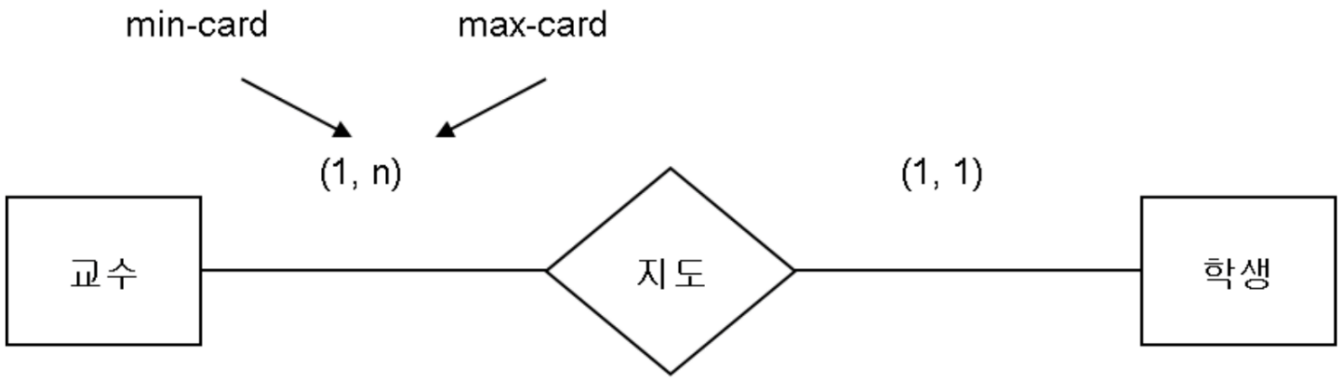
- 교수와 학생은 특정 학과에 소속된다.





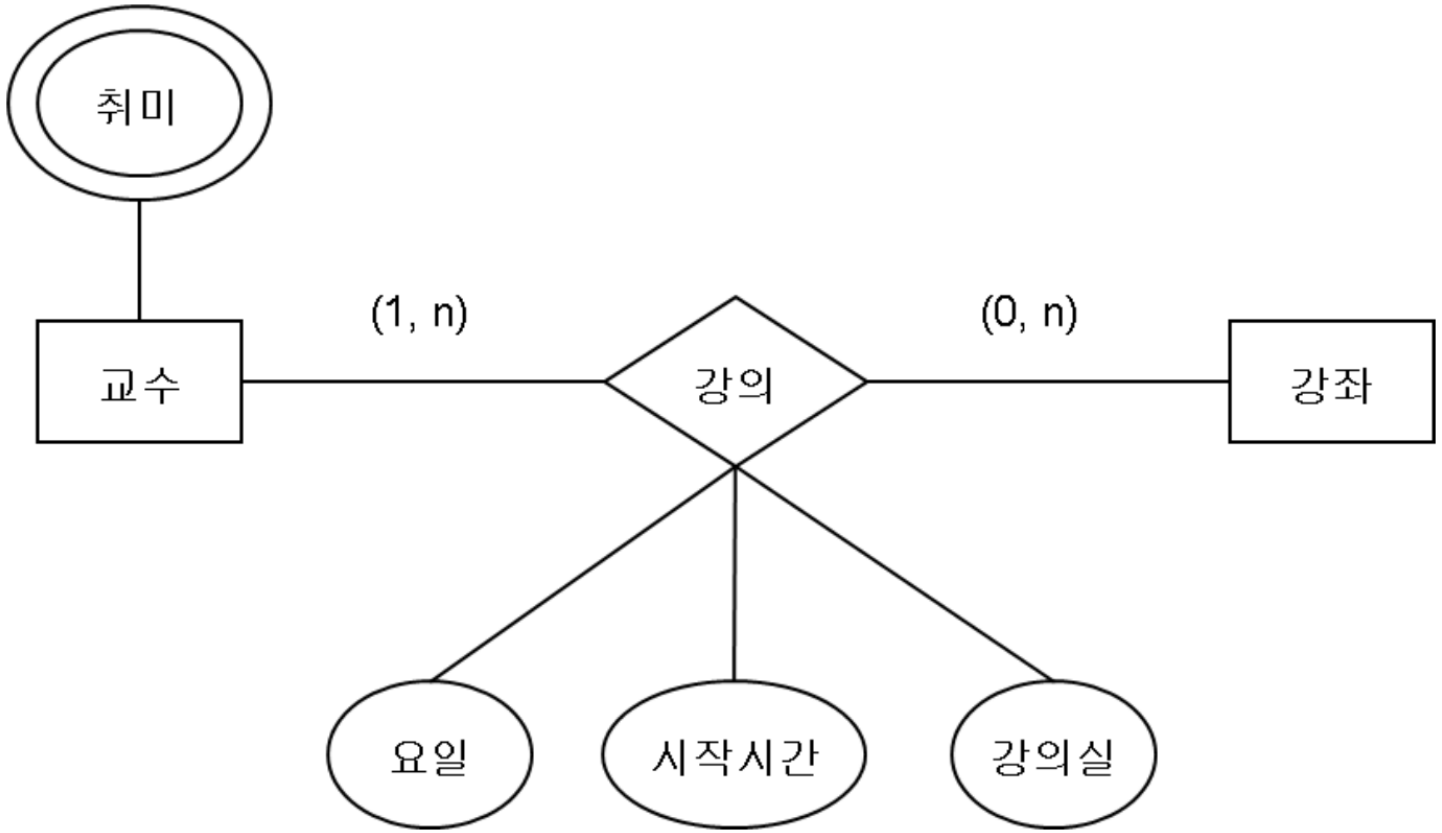
○ 관계 – Mapping Cardinality

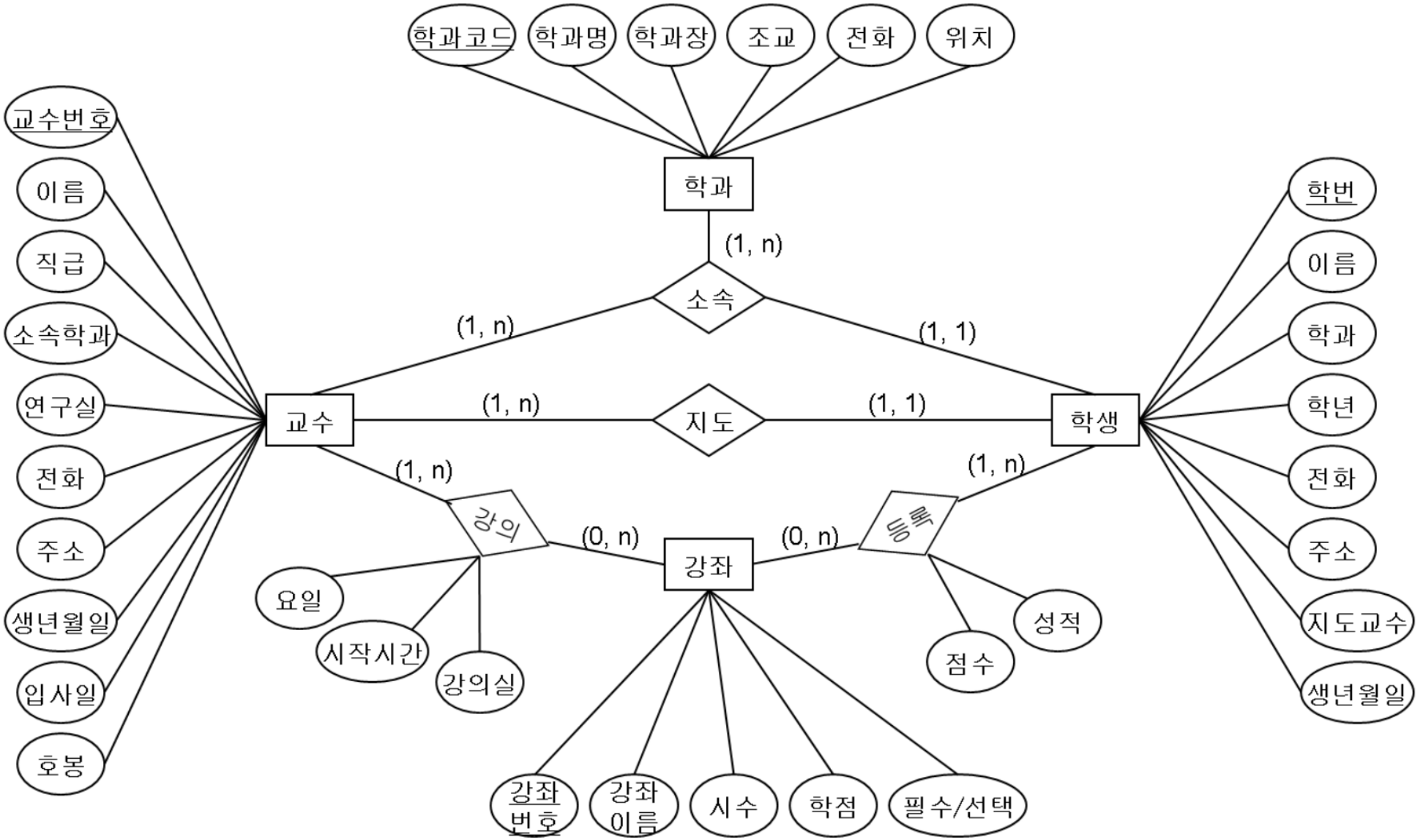
- 조건 1 : 교수는 꼭 학생에 대한 지도를 해야 한다.
  - min-card(교수, 지도) = 1
- 조건 2 : 교수는 여러 명의 학생을 지도할 수 있다.
  - max-card(교수, 지도) = n
- 조건 3 : 학생은 꼭 교수에게 지도를 받아야 한다.
  - min-card(교수, 지도) = 1
- 조건 4 : 학생은 여러 명의 교수에게 지도를 받을 수 없다.
  - max-card(교수, 지도) = 1



○ 속성

- 교수는 어려 개의 취미를 가질 수 있다.
- 강의는 요일, 시작시간, 강의실의 속성을 갖는다.




학과코드
학과명
학과장
조교
전화
위치
학과
(1, n)
소속
(1, n)
(1, 1)
교수번호
이름
직급
소속학과
연구실
전화
주소
생년월일
입사일
호봉
교수
(1, n)
지도
(1, 1)
학생
학번
이름
학과
학년
전화
주소
지도교수
생년월일
(1, n)
강의
(0, n)
강좌
(0, n)
듣다
(1, n)
요일
시작시간
강의실
성적
점수
강좌 번호
강좌 이름
시수
학점
필수/선택

■ 예제: 공급자와 부품

○ A DB with four tables (*SUPPLIER*, *CITY*, *PART*, *SELLS*)

○ *SUPPLIER* has attributes: number (SNO), the name (SNAME) and the city number (CNO)

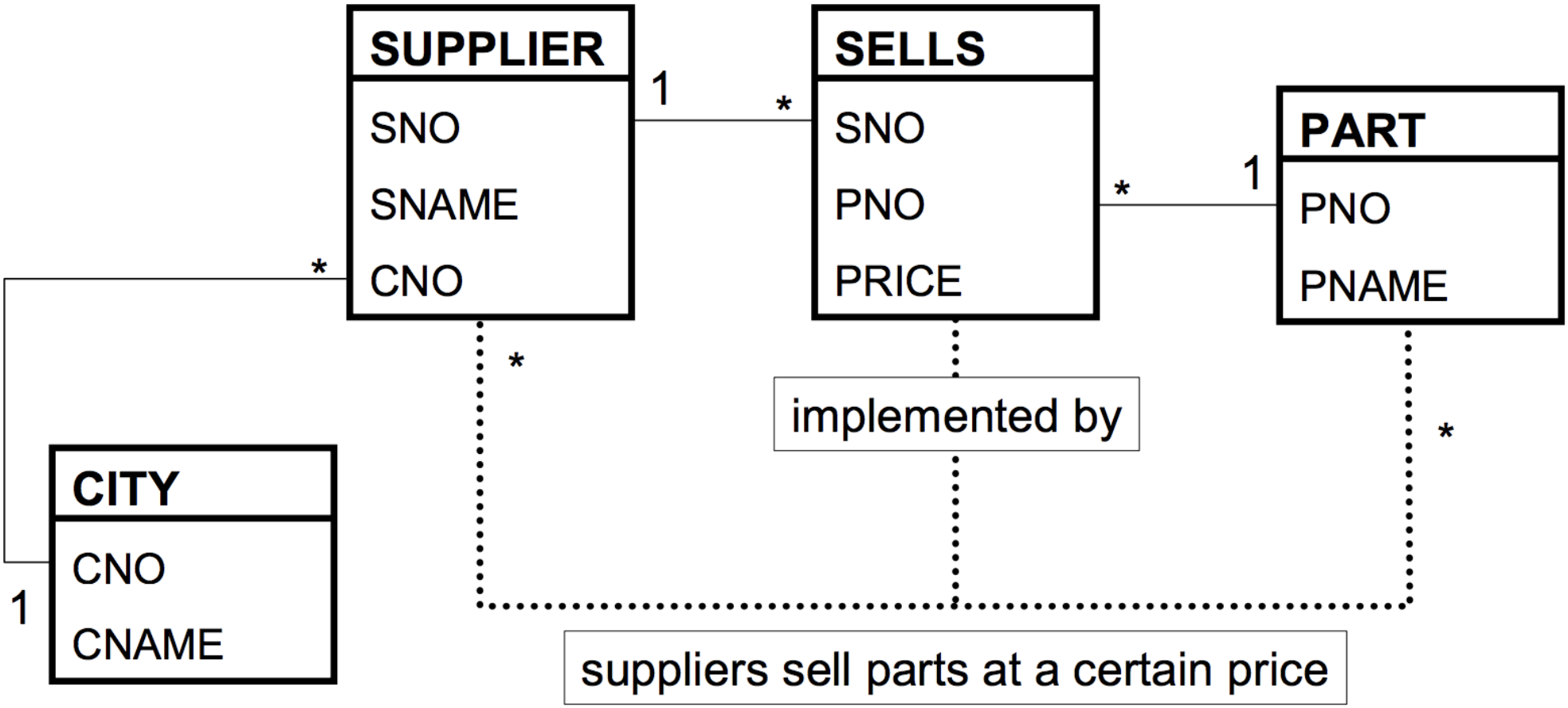
○ *CITY* has attributes number (CNO) and the city name (CNAME)

○ *PART* has attributes: number (PNO) the name (PNAME) and the price (PRICE)

○ *SELLS* has attributes: part (PNO) and supplier (SNO). SELLS connects SUPPLIER and PART

■ Find Entity / Relationship

■ ER 모델


SUPPLIER
SNO
SNAME
CNO
SELLS
SNO
PNO
PRICE
PART
PNO
PNAME
CITY
CNO
CNAME
1
*
1
*
1
*
*
*
*
1
implemented by
suppliers sell parts at a certain price

■ Entity – City

CITY:

CNO	CNAME
1	London
2	Paris
3	Rome
4	Vienna

○ Row (1, London) in CITY represents a distinct city, London, with city number 1

■ Entity – Supplier

SUPPLIER:

SNO	SNAME	CNO
1	Smith	1
2	Jones	2
3	Adams	1
4	Blake	3

○ Row (1,Smith, 1) in SUPPLIER represents a supplier with supplier number 1 whose name is Smith and who is based in the city numbered 1 (London)

■ Entity – Part

PART:

PNO	PNAME
1	Screw
2	Nut
3	Bolt
4	Cam

○ Row (2,Nut) in PART represents a part with part number 2, and part name Nut

■ Entity – Sells

SELLS:

SNO	PNO	PRICE
1	1	10
1	2	8
2	4	38
3	1	11
3	3	6
4	2	7
4	3	4
4	4	45

○ Row (1,2,8) in SELLS represents the relationship of supplier 1 (Smith) selling part 2 (Nut) for 8 cents

1.3. SQL

■ SQL

○ Structured Query Language

- ➢ RDBMS의 데이터를 관리하기 만들어진 언어 (대부분 ISO 표준을 따름)
- ➢ 자료의 검색과 관리, 데이터베이스 스키마 생성과 수정, 데이터베이스 객체 접근 조정 관리 등을 고안

■ SQL 명령

○ 데이터 정의 언어 DDL(Data Definition Language)

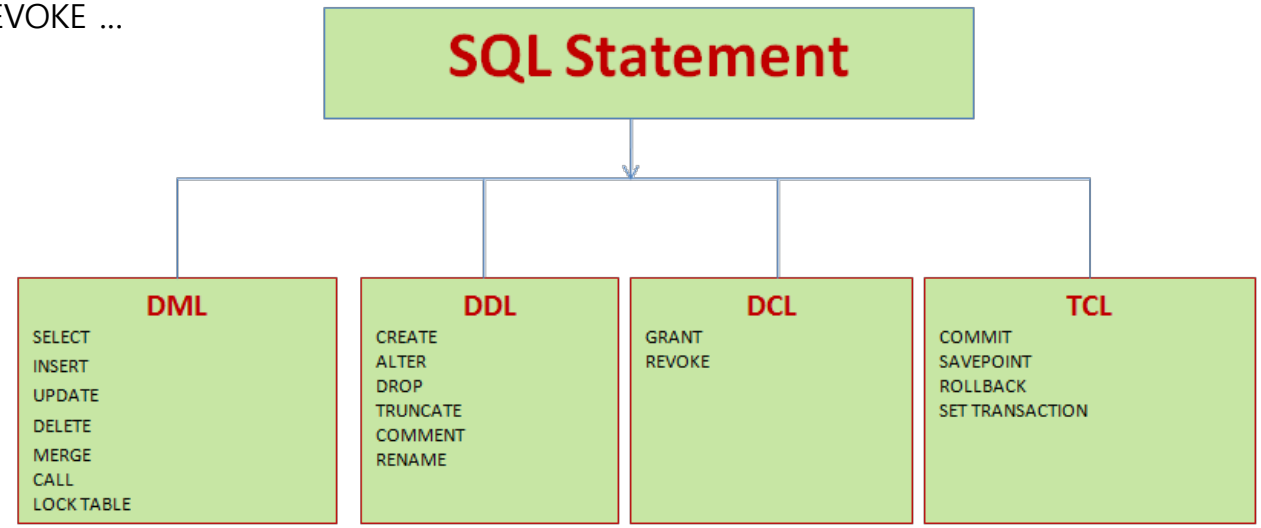
- ➢ CREATE, DROP, ALTER, TRUNCATE ...

○ 데이터 조작 언어 DML(Data Manipulation Language)

- ➢ INSERT, UPDATE, DELETE, SELECT ...

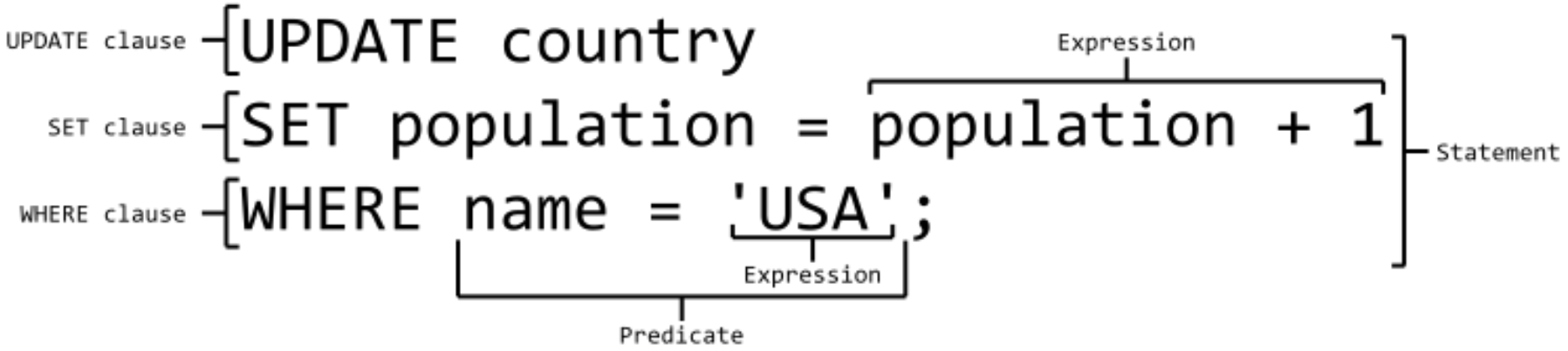
○ 데이터 제어 언어 DCL(Data Control Language)

- ➢ GRANT, REVOKE ...



■ SQL 구문

○ 문(Statement), 절(Clause), 식(Expression), 술어(Predicate)



■ Data Type

○ Boolean (BOOLEAN)

- ➢ to represent value TRUE or FALSE

○ Character (CHAR, VARCHAR)

- ➢ to represent character data for columns such as names of persons or cities

○ Exact numeric (NUMERIC, DECIMAL, INTEGER, SMALLINT, BIGINT)

- ➢ to represent number such as 1, 2, 3, ... and so on
- ➢ used for numbers with fractions such as 123.45

○ Approximate numeric (REAL, FLOAT, DOUBLE)

- ➢ to represent numbers with fractions such as 123.45
- ➢ precision of the fraction may not be preserved across manipulation of such column values

○ Datetime (DATE, TIME, TIMESTAMP) / Large Object (CLOB, BLOB)

Data Type Family	Data Types
Character String	CHARACTER, VARCHAR, CLOB
Boolean	BOOLEAN
Binary String	BLOB
Date Time	DATE, TIME, TIMESTAMP
Number	SMALLINT, INTEGER, DECIMAL, NUMERIC, REAL, FLOAT, DOUBLE

■ BOOLEAN

○ values of BOOLEAN column : **TRUE** or **FALSE**

○ Syntax : **BOOLEAN**

○ Example : *definition of a column **is_manager** in a table*

is_manager ***BOOLEAN***

■ CHARACTER

○ a **sequence of characters** such as names of persons, cities, etc

○ Syntax : **{ CHARACTER | CHAR } [(length)]**

○ Length : length of the string, 1 if not specified / a fixed-length data type

○ Example :

book_title *CHARACTER(50)*

book_category *CHAR*

door_type *CHAR(3)*

Inserted Value	Actual Value Inserted
'IN'	'IN '
'OUT'	'OUT'
'INOUT'	Nothing is inserted due to error.

■ VARCHAR

○ varying character data type

○ size of data stored in such a column in each row can vary according to the actual size of the data

○ Syntax : **{ VARCHAR | CHARACTER VARYING | CHAR VARYING } [(length)]**

○ Length : if the data size is less than the size specified, data is not padded with blanks

○ Example :

book_synopsis VARCHAR(500)

■ INTEGER

○ integers in the range of -2147483648 to 2147483647

○ Syntax : **INTEGER**

○ Length : number of bytes occupied in a column by integer is **4bytes**(32bits)

○ Example :

book_sno *INTEGER*

order_no *INTEGER*

■ SMALLINT

○ integers in the range of -32768 to 32767

○ Syntax : **SMALLINT**

○ Length : number of bytes occupied in a column by smallint is **2bytes**(16bits)

○ Example :

roll_no	***SMALLINT***
quantity	***SMALLINT***

■ NUMERIC

○ numbers that can have fractions

○ Syntax : **{ NUMERIC } [(precision [, scale])]**

○ Precision : total number of digits including number of decimal places, default is RDBMS specific

○ Scale : number of decimal places, default is 0

○ Length : one byte per digit

○ Example :

price	***NUMERIC(8,2)***
discount	***NUMERIC(4,1)***
interest	***NUMERIC(4,2)***

■ REAL

○ approximate numerical data type

○ Syntax : **REAL**

○ Length : 4bytes(32bits)

○ Example :

salary	***REAL***
width	***REAL***

■ FLOAT

○ approximate numerical data type

○ Syntax : **FLOAT [precision]**

○ Precision : precision of mantissa

○ Length : 4bytes(32bits)

○ Example :

distance *FLOAT*

■ DOUBLE

○ approximate numerical data type

○ Syntax : **DOUBLE**

○ Length : 8bytes(64bits)

○ Example :

distance ***DOUBLE***

■ DATE

○ calendar date

○ consists of datetime fields : **YEAR**, **MONTH** and **DAY**

○ Syntax : **DATE**

○ Example :

date_of_birth ***DATE***

join_data ***DATE***

■ TIME

○ time of a day

○ consists of datetime fields : **HOUR**, **MINUTE** and **SECOND**

○ Syntax : **TIME [(precision)]**

○ Precision : seconds value, default is 0 / 3 means milliseconds / 6 means microseconds

○ Example :

stat_titme	***TIME***
event_time	***TIME(6)***

■ CLOB

○ large amount of character data

○ Syntax : **{ CHARACTER LARGE OBJECT | CHAR LARGE OBJECT | CLOB } [(large-object-length)]**

○ Large object length : length **[K | M | G]**

○ Example :

part_description CLOB(5M)

emp_resume CLOB(10K)

■ BLOB

○ large amount of binary data

○ Syntax : **{ BINARY LARGE OBJECT | BLOB } [(large-object-length)]**

○ Large object length : length **[K | M | G]**

○ Example :

part_image ***BLOB(2M)***

emp_photograph ***CLOB(150K)***

Data type	Access	SQLServer	Oracle	MySQL	PostgreSQL
boolean	Yes/No	Bit	Byte	N/A	Boolean
integer	Number (integer)	Int	Number	Int Integer	Int Integer
float	Number (single)	Float Real	Number	Float	Numeric
currency	Currency	Money	N/A	N/A	Money
string (fixed)	N/A	Char	Char	Char	Char
string (variable)	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
binary object	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary

■ DDL

○ CREATE

```
CREATE TABLE [table name] ( [column definitions] ) [table parameters]
```

➢ column definitions

A comma-separated list consisting of any of the following

Column definition: *[column name] [data type] {NULL | NOT NULL} {column options}*

Primary key definition: *PRIMARY KEY ([comma separated column list])*

Constraints: *{CONSTRAINT} [constraint definition]*

RDBMS specific functionality

➢ constraints

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY - Uniquely identifies a row/record in another table
CHECK - Ensures that all values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column when no value is specified
INDEX - Used to create and retrieve data from the database very quickly

➢ 예제

```
CREATE TABLE employees (
    id              INTEGER         PRIMARY KEY,
    first_name      VARCHAR(50)     not null,
    last_name       VARCHAR(75)     not null,
    fname           VARCHAR(50)     not null,
    dateofbirth     DATE            not null
);
```

➢ 예제

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

PersonID	LastName	FirstName	Address	City

○ DROP

```
DROP objecttype objectname.
```

The *DROP* statement destroys an existing database, table, index, or view.

➢ 예제

```
DROP TABLE employees;
```

The *DROP* statement is distinct from the *DELETE* and *TRUNCATE* statements, in that *DELETE* and *TRUNCATE* do not remove the table itself.

○ TRUNCATE

```
TRUNCATE TABLE table_name;
```

The *TRUNCATE* statement is used to delete all data from a table. It's much faster than *DELETE*.

○ ALTER

```
ALTER objecttype objectname parameters.
```

The *ALTER* statement modifies an existing database object.

➢ parameters

```
ALTER TABLE table_name
ADD column_name datatype;
```

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

➢ 예제

– 이미 만들어진 *sink* 테이블에 *bubbles* 열(필드) 추가

```
ALTER TABLE sink ADD bubbles INTEGER;
ALTER TABLE sink DROP COLUMN bubbles;
```

➢ 예제

ID	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

```
ALTER TABLE Persons
ADD DateOfBirth date;
```

```
ALTER TABLE Persons
MODIFY COLUMN DateOfBirth year;
```

ID	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	
2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

```
ALTER TABLE Persons
DROP COLUMN DateOfBirth;
```

■ DML

○ INSERT

➢ 기본 방법

```
INSERT INTO table (column1 [, column2, column3 ... ]) VALUES (value1 [, value2, value3 ... ])
```

The number of columns and values must be the same.

➢ 빠른 방법

```
INSERT INTO table VALUES (value1, [value2, ... ])
```

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query.

➢ 예제

```
INSERT INTO phone_book (name, number) VALUES ('John Doe', '555-1212');
```

```
INSERT INTO phone_book VALUES ('John Doe', '555-1212');
```

➢ 예제

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

➢ 예제

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

○ SELECT

➢ 빠른 방법

```
SELECT column1, column2, ...
FROM table_name;
```

A **SELECT** statement retrieves zero or more rows from one or more database tables or database views.

In most applications, `SELECT` is the most commonly used data query language (DQL) command.

➢ 예제

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT CustomerName, City FROM Customers;
```

```
SELECT * FROM Customers;
```

➢ 기본 방법

```
SELECT [ALL | DISTINCT] 컬럼명 [,컬럼명...]
FROM 테이블명 [,테이블명...]
[WHERE 조건식]
[GROUP BY 컬럼명 [HAVING 조건식]]
[ORDER BY 컬럼명]
GROUP BY 컬럼명[,컬럼명...]
ORDER BY 컬럼명[,컬럼명...]
```

➢ parameters

`WHERE` specifies which rows to retrieve.

`GROUP BY` groups rows sharing a property so that an aggregate function can be applied to each group.

`HAVING` selects among the groups defined by the GROUP BY clause.

`ORDER BY` specifies an order in which to return the rows.

`AS` provides an alias which can be used to temporarily rename tables or columns.

➢ 예제

Table "T"	Query	Result
C1 C2 1 a 2 b	`SELECT * FROM T;`	C1 C2 1 a 2 b
C1 C2 1 a 2 b	`SELECT C1 FROM T;`	C1 1 2
C1 C2 1 a 2 b	`SELECT * FROM T WHERE C1 = 1;`	C1 C2 1 a
C1 C2 1 a 2 b	`SELECT * FROM T ORDER BY C1 DESC;`	C1 C2 2 b 1 a

○ WHERE

- ➢ used to filter records.
- ➢ used to extract only those records that fulfill a specified condition

- ➢ 연산자

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

➢ 예제

– AND, OR

```
DELETE
FROM   mytable
WHERE  mycol > 100 AND item = 'Hammer'
```

– IN (find any values existing in a set of candidates)

```
SELECT ename WHERE ename IN ('value1', 'value2', ...)
```

```
SELECT ename WHERE ename='value1' OR ename='value2'
```

– BETWEEN (find any values within a range)

```
SELECT ename WHERE ename BETWEEN 'value1' AND 'value2'
```

```
SELECT salary from emp WHERE salary BETWEEN 5000 AND 10000
```

- LIKE (find a string fitting a certain description / % wildcard)
  - » 'a%' / '%a' / '%or%' / '_r%' / 'a_%_%' / 'a%o'

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

- CustomerName
  - » 'a%' / '%a' / '%or%' / '_r%' / 'a_%_%'

- ContactName
  - » 'a%o'

- CustomerName NOT LIKE
  - » 'a%'

○ UPDATE

➢ 기본방법

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

➢ changes the data of one or more records in a table

➢ Either all the rows can be updated, or a subset may be chosen using a condition.

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

➢ 예제

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

➢ Update table

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

➢ Multiple Records

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

➢ Update Warning

```
UPDATE Customers
SET ContactName='Juan';
```

○ DELETE

➢ 기본방법

```
DELETE FROM table_name
WHERE condition;
```

➢ removes one or more records from a table

➢ A subset may be defined for deletion using a condition, otherwise all records are removed

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

➢ 예제

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

➢ Delete

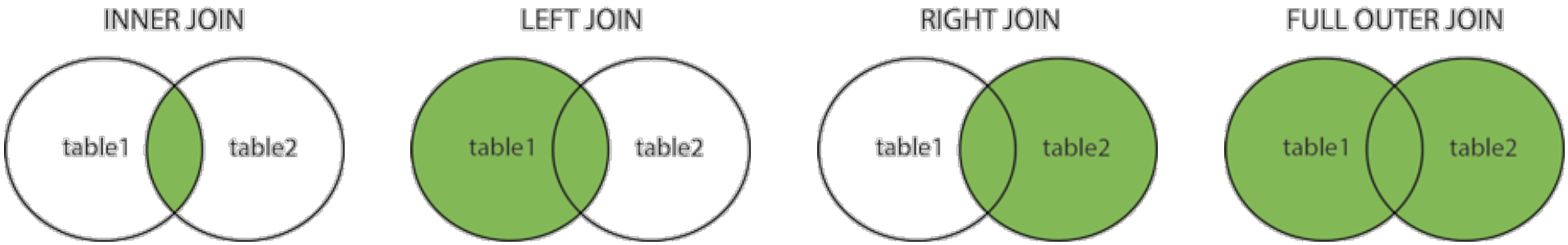
```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

➢ Delete all records

```
DELETE FROM table_name;
```

○ Joins

- combines columns from one or more tables in a relational database, based on a related column between them
- JOIN: ***INNER***, ***LEFT OUTER***, ***RIGHT OUTER***, ***FULL OUTER***, ***CROSS***



- (INNER) **JOIN**
  - returns records that have matching values in both tables

- **LEFT** (OUTER) JOIN
  - return all records from the left table, and the matched records from the right table

- **RIGHT** (OUTER) JOIN
  - return all records from the right table, and the matched records from the left table

- **FULL** (OUTER) JOIN
  - return all records when there is a match in either left or right table

➢ 예제

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996

○ SQL 예제

➢ CREATE, INSERT

```
CREATE TABLE department
(
 DepartmentID INT Primary key,
 DepartmentName VARCHAR(20)
);

CREATE TABLE employee
(
 LastName VARCHAR(20),
 DepartmentID INT references department(DepartmentID)
);

INSERT INTO department VALUES(31, 'Sales');
INSERT INTO department VALUES(33, 'Engineering');
INSERT INTO department VALUES(34, 'Clerical');
INSERT INTO department VALUES(35, 'Marketing');

INSERT INTO employee VALUES('Rafferty', 31);
INSERT INTO employee VALUES('Jones', 33);
INSERT INTO employee VALUES('Heisenberg', 33);
INSERT INTO employee VALUES('Robinson', 34);
INSERT INTO employee VALUES('Smith', 34);
INSERT INTO employee VALUES('Williams', NULL);
```

➢ Table

Employee table

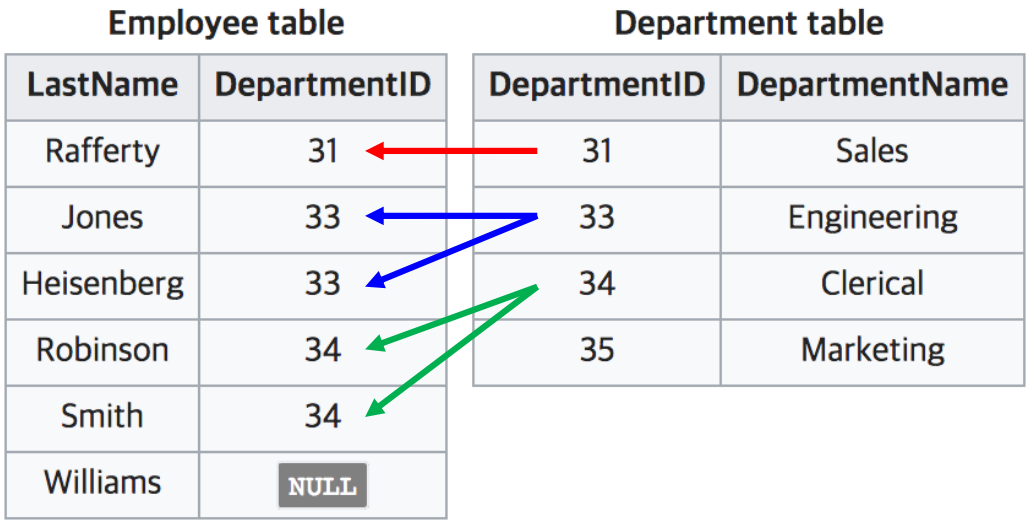
LastName	DepartmentID
Rafferty	31
Jones	33
Heisenberg	33
Robinson	34
Smith	34
Williams	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

➢ CROSS JOIN

```
SELECT *
FROM employee CROSS JOIN department;
```



Employee table

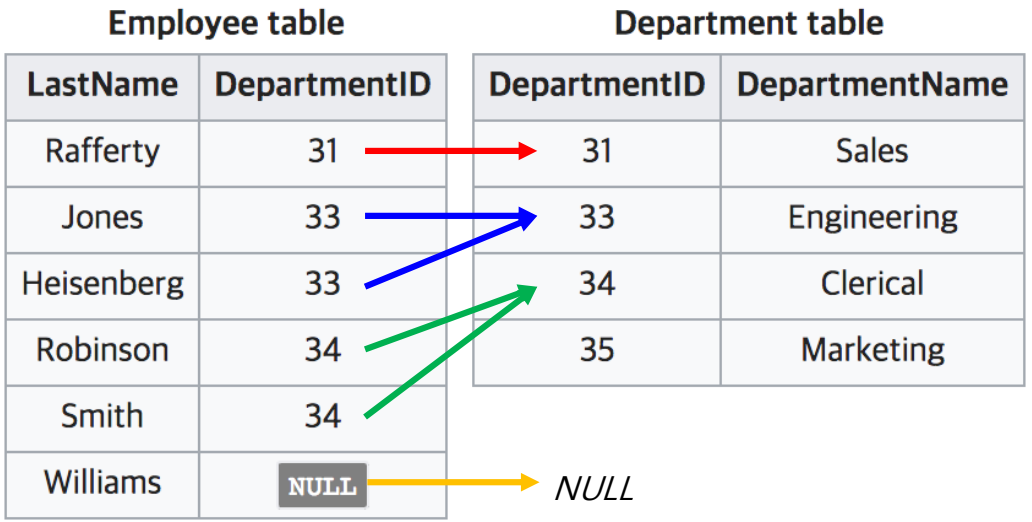
LastName	DepartmentID
Rafferty	31
Jones	33
Heisenberg	33
Robinson	34
Smith	34
Williams	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

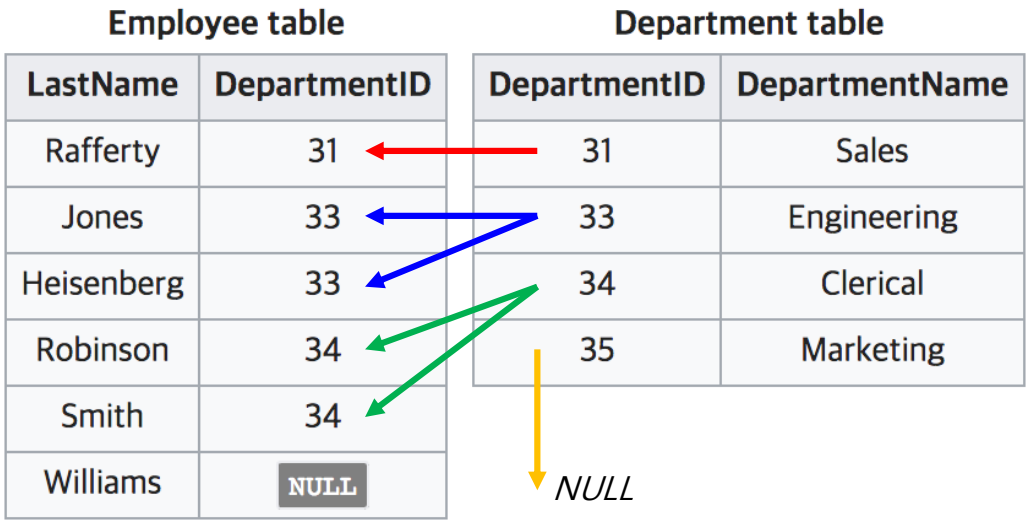
➢ INNER JOIN

```
SELECT employee.LastName, employee.DepartmentID, department.DepartmentName
FROM employee
INNER JOIN department ON
employee.DepartmentID = department.DepartmentID
```



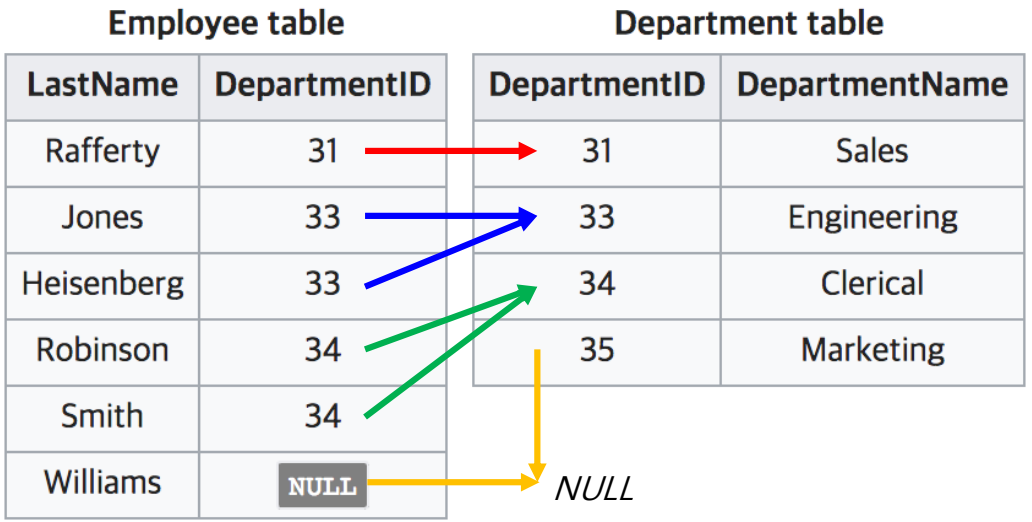
➢ LEFT OUTER JOIN

```
SELECT *
FROM employee
LEFT OUTER JOIN department ON employee.DepartmentID = department.DepartmentID;
```



➢ RIGHT OUTER JOIN

```
SELECT *
FROM employee RIGHT OUTER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```



➢ FULL OUTER JOIN

```
SELECT *
FROM employee FULL OUTER JOIN department
  ON employee.DepartmentID = department.DepartmentID;
```

○ SQL 예제

CITY:

CNO	CNAME
1	London
2	Paris
3	Rome
4	Vienna

SUPPLIER:

SNO	SNAME	CNO
1	Smith	1
2	Jones	2
3	Adams	1
4	Blake	3

PART:

PNO	PNAME
1	Screw
2	Nut
3	Bolt
4	Cam

SELLS:

SNO	PNO	PRICE
1	1	10
1	2	8
2	4	38
3	1	11
3	3	6
4	2	7
4	3	4
4	4	45

➢ QUERY

- SELECT * FROM PART;
- SELECT * FROM SELLS WHERE PRICE > 11;
- SELECT SNO, PRICE FROM SELLS WHERE PRICE > 11;
- SELECT PNO, PRICE FROM SELLS WHERE PNO = 1 AND PRICE <= 10;

- SELECT * FROM SUPPLIER, PART; *(CROSS JOIN – 4*4)*
- SELECT S.SNAME, C.CNAME FROM SUPPLIER AS S, CITY AS C WHERE S.CNO = C.CNO;
- SELECT SNAME AS LondonSuppliers FROM SUPPLIER, CITY *(CROSS JOIN – 4*4)*
  WHERE SUPPLIER.CNO = CITY.CNO AND CNAME = 'London';
- SELECT SNAME, PNAME, PRICE FROM SELLS, SUPPLIER, PART *(CROSS JOIN – 8*4*4)*
  WHERE SELLS.SNO = SUPPLIER.SNO AND SELLS.PNO = PART.PNO ORDER BY SNAME, PNAME;

- INSERT INTO SUPPLIER VALUES (1, 'Smith', 1);
- UPDATE SELLS SET PRICE = 15 WHERE SNO = 1 AND PNO = 1;
- DELETE FROM SUPPLIER WHERE SNAME = 'Smith';