

데이터

1. DBMS

- 1.1. 데이터베이스 개요
 - 1.2. RDBMS
 - 1.3. SQL
 - 1.4. SQLite
-

2. ORM, RE

- 2.1. ORM 소개
 - 2.2. SQLAlchemy
 - 2.3. Regular Expression
-

3. HTML, CSS

- 3.1. HTML5 소개
 - 3.2. HTML
 - 3.3. CSS
-

4. JSON, XML

- 4.1. JSON
- 4.2. XML

1. DBMS

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

1.1. 데이터베이스 개요

80%
Unstructured



Vs

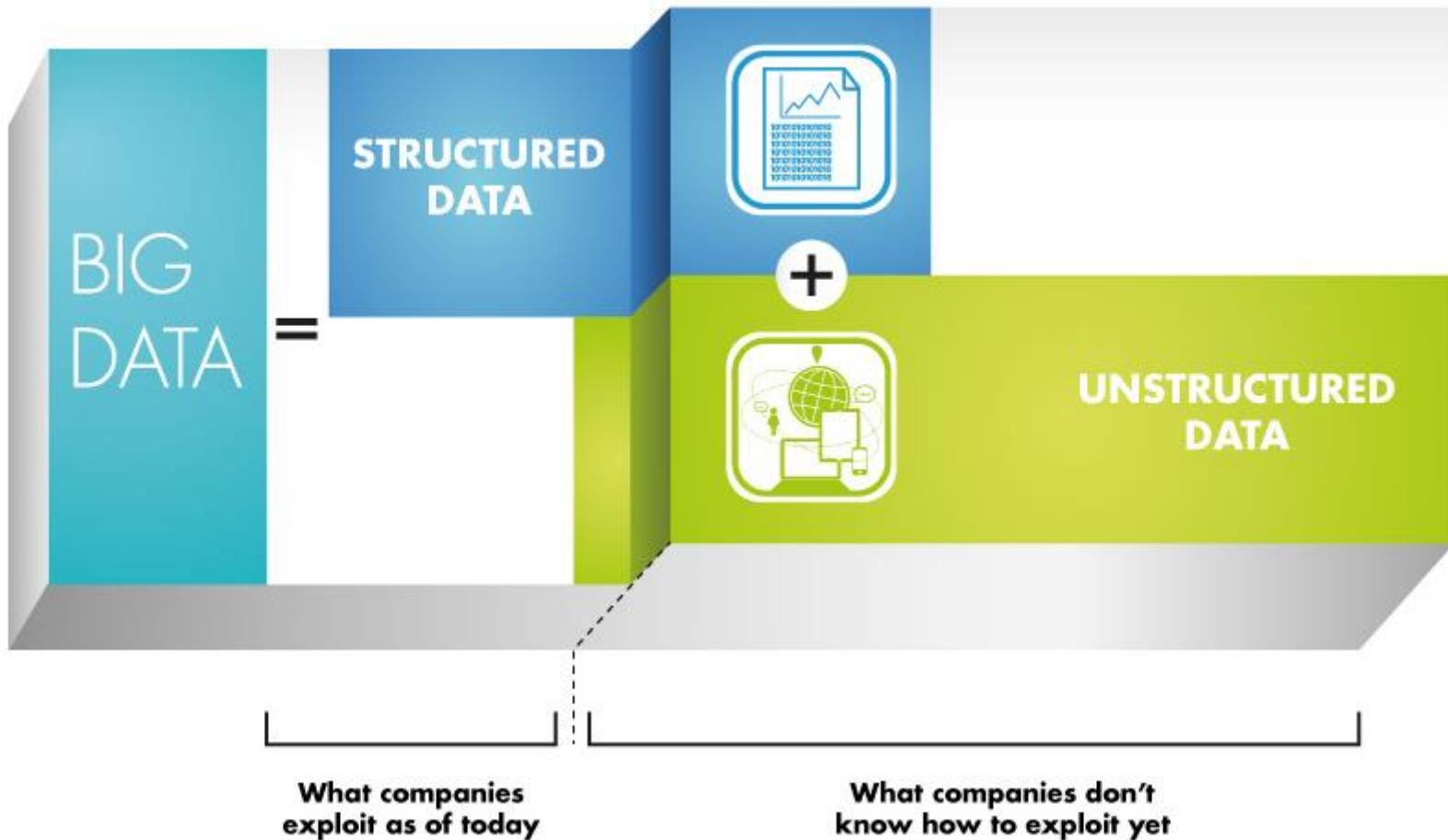
20%
Structured

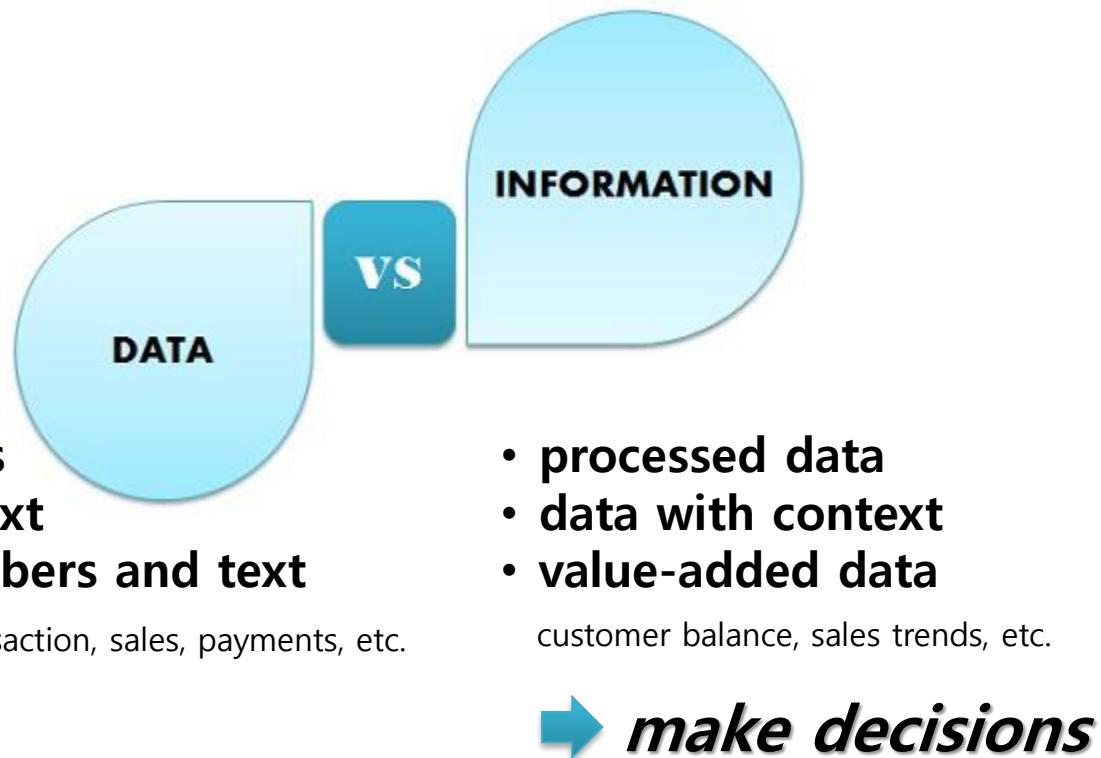
Database



Tables







Database

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

structured

Data

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

Information

- Processed data
 - Documents
 - Audio
 - Images
 - Video

■ 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 지향
- 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

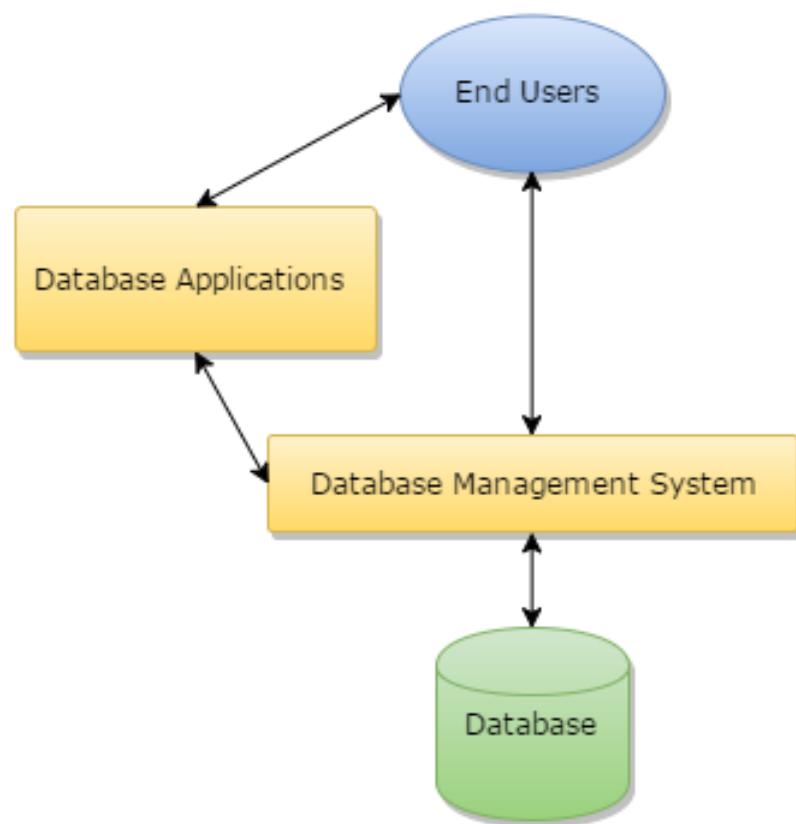
■ DBMS의 역할

- create databases

- insert, update and delete data

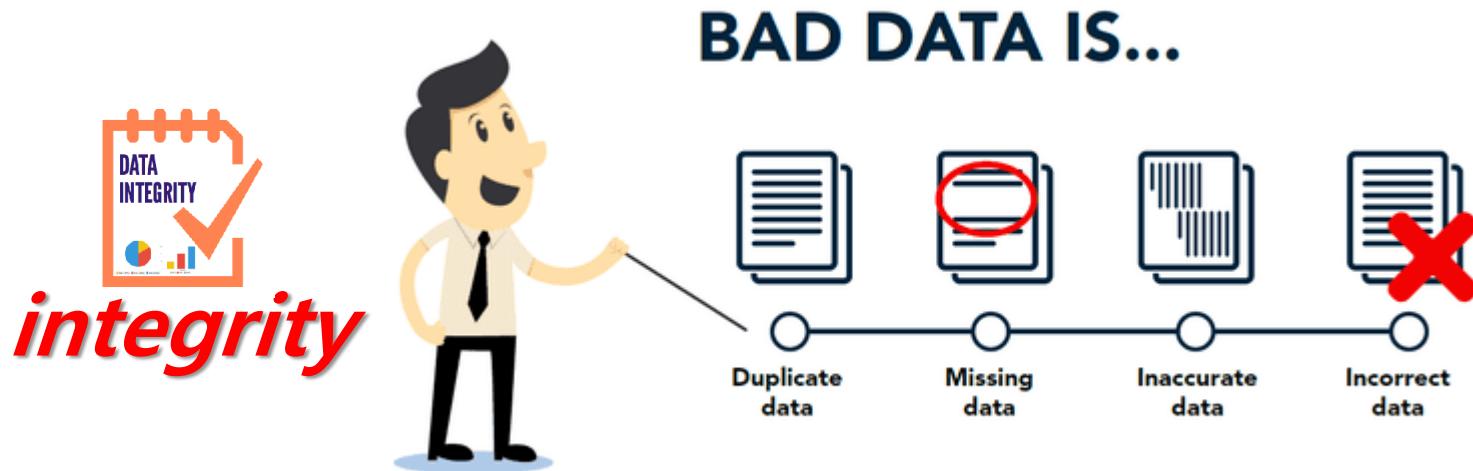
- sort and query data

- create form and report



■ DBMS 역할과 장점

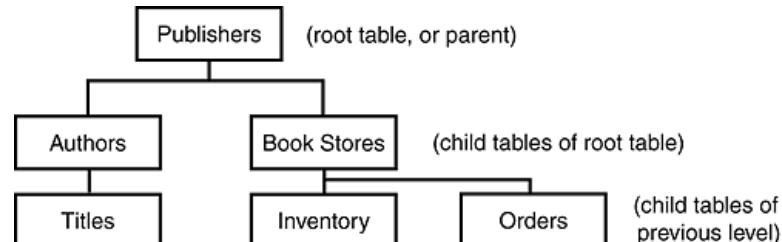
- Improved data sharing
- Improved data security
- Minimized data inconsistency
- Improved data access
- Improved decision making
- Increased end-user productivity
- Reduce application development time



■ DBMS 종류

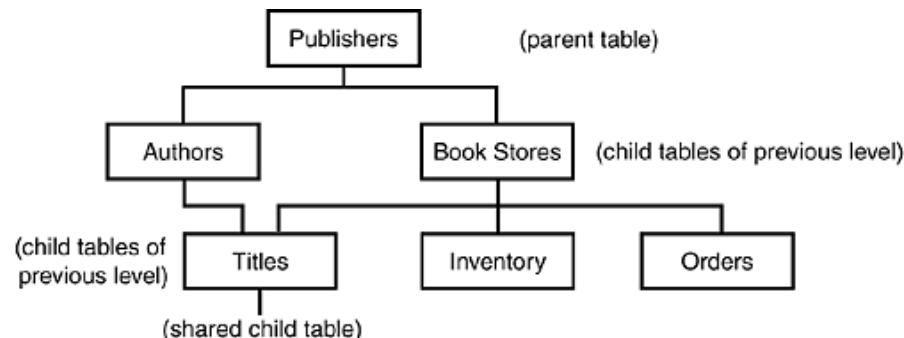
○ 계층형 모델(Hierarchical Database Model)

- 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

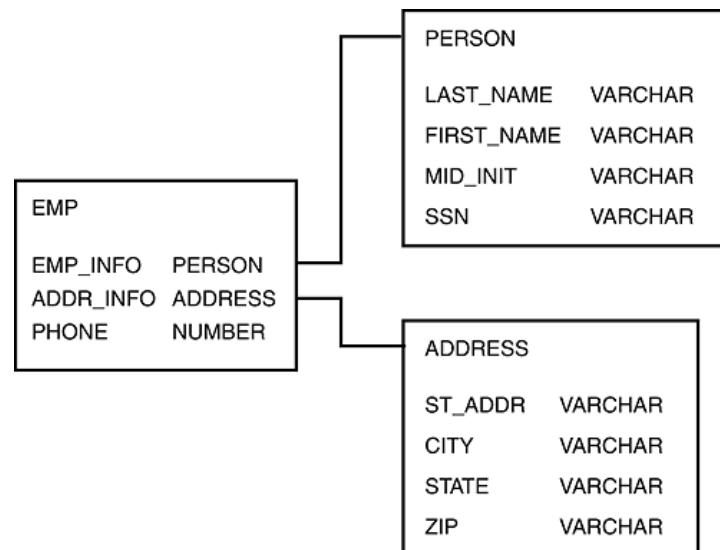


○ 관계 데이터 모델(Relational Database Model)

- 개체를 테이블, 개체 집합들 사이의 관계를 공통 속성으로 연결

○ 객체 관계 모델(Object Relational Database Model)

- 속성-관계
- 개체-관계



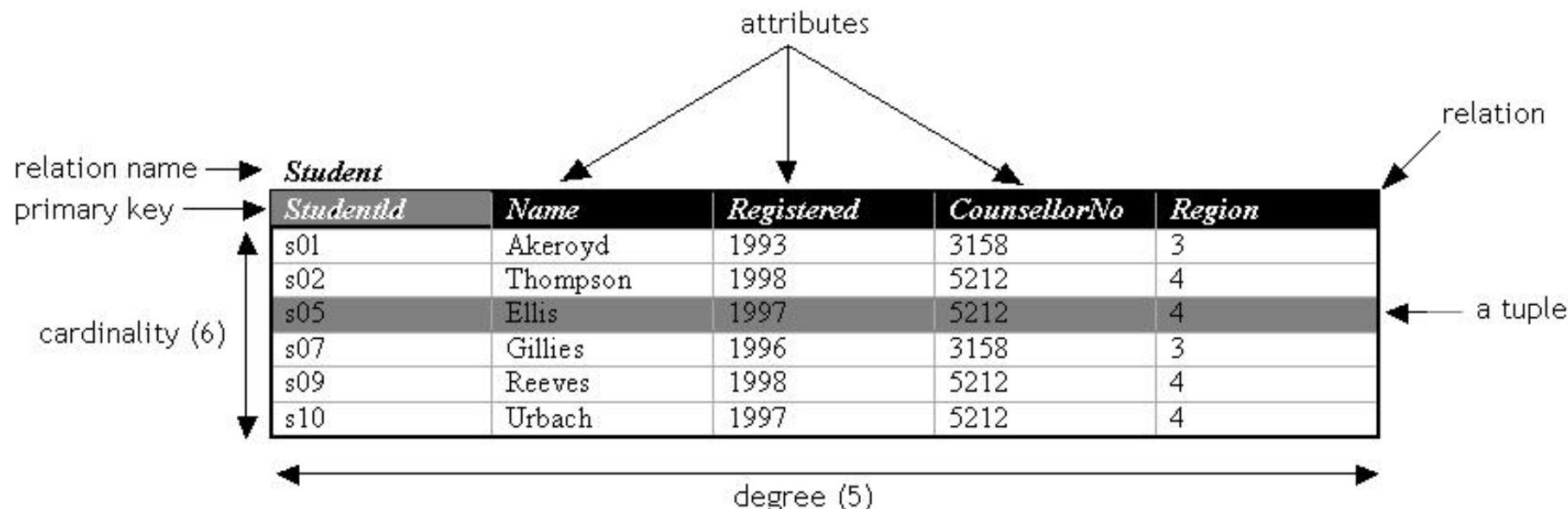
1.2. RDBMS

■ 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



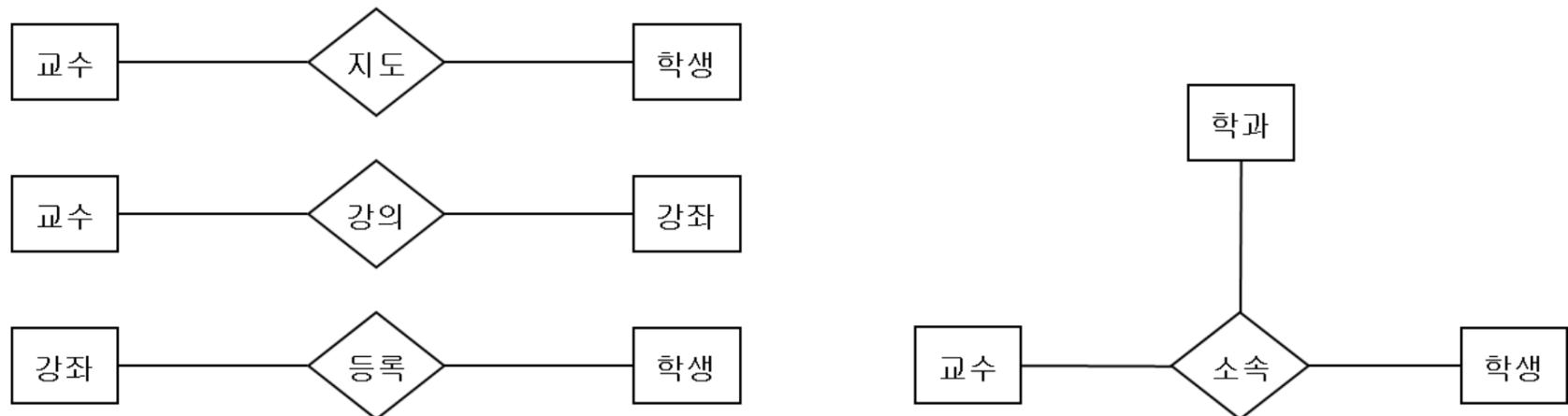
■ 예제: 교수-학생 ER모델

○ 이항 관계

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

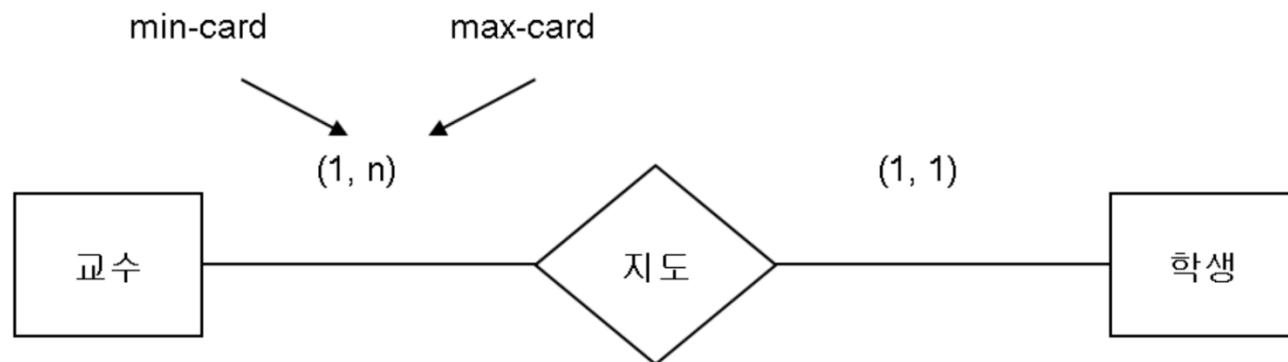
○ 삼항 관계

- 교수와 학생은 특정 학과에 소속된다.



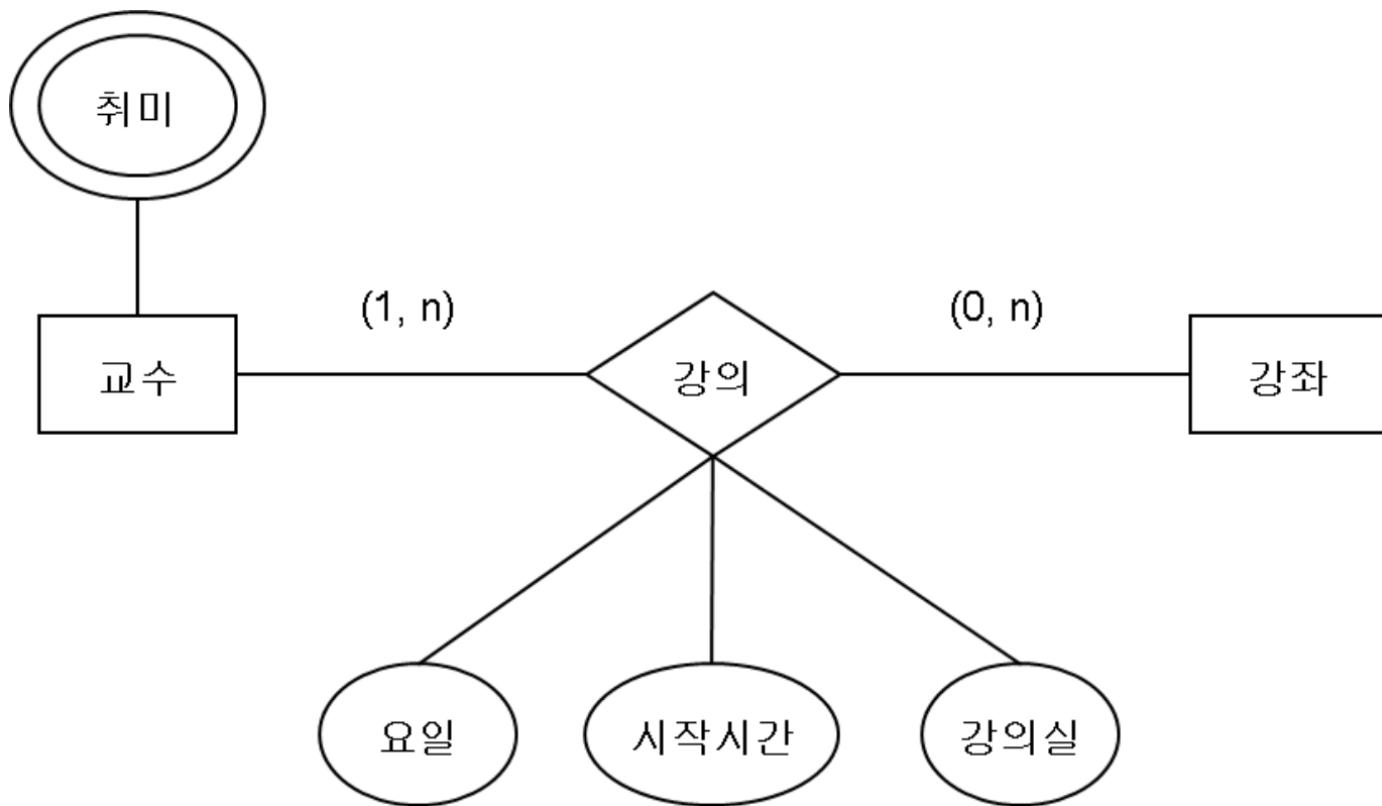
○ 관계 – Mapping Cardinality

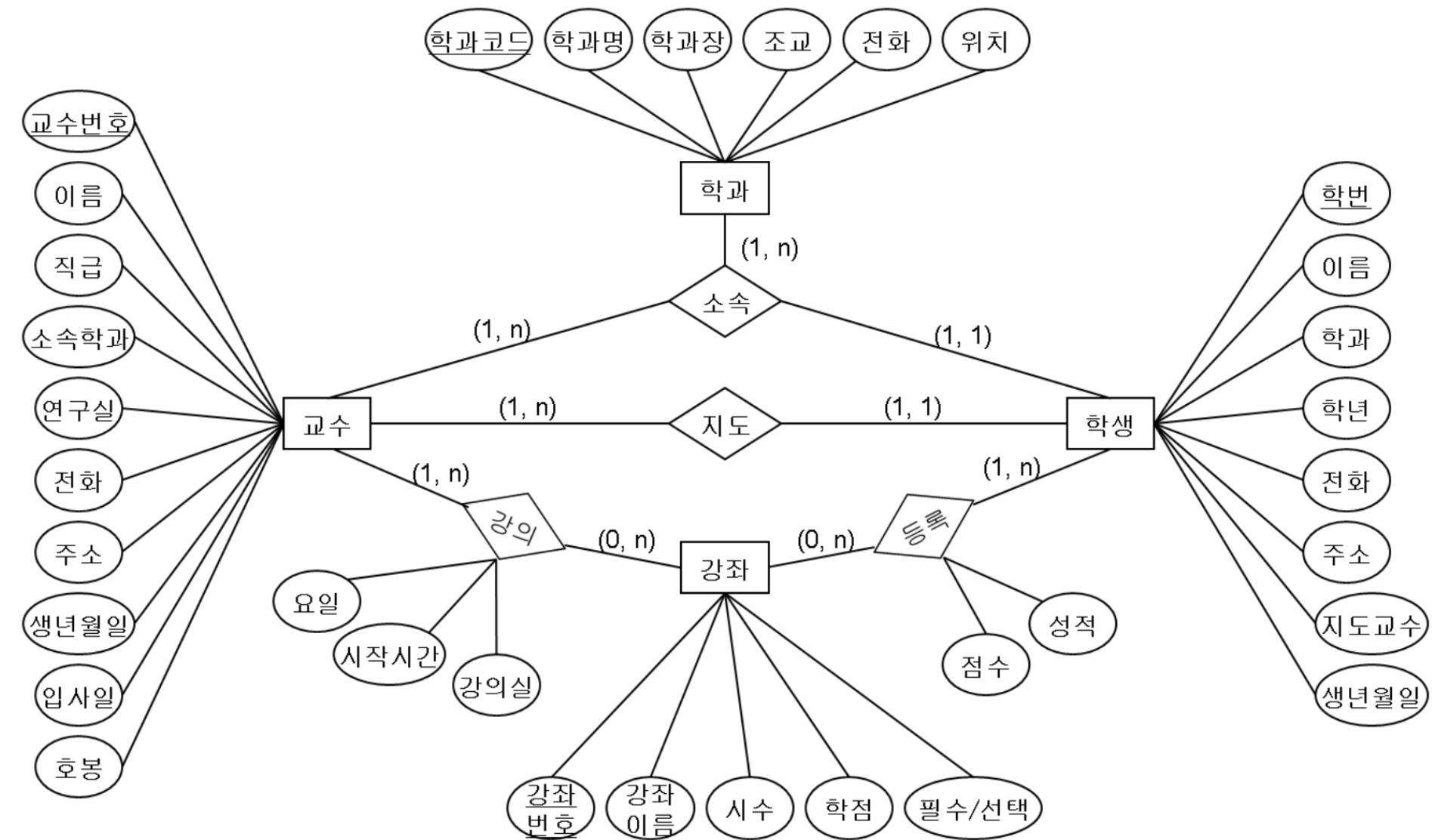
- 조건 1 : 교수는 꼭 학생에 대한 지도를 해야 한다.
 - $\text{min-card}(\text{교수}, \text{지도}) = 1$
- 조건 2 : 교수는 여러 명의 학생을 지도할 수 있다.
 - $\text{max-card}(\text{교수}, \text{지도}) = n$
- 조건 3 : 학생은 꼭 교수에게 지도를 받아야 한다.
 - $\text{min-card}(\text{교수}, \text{지도}) = 1$
- 조건 4 : 학생은 여러 명의 교수에게 지도를 받을 수 없다.
 - $\text{max-card}(\text{교수},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



○ 속성

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



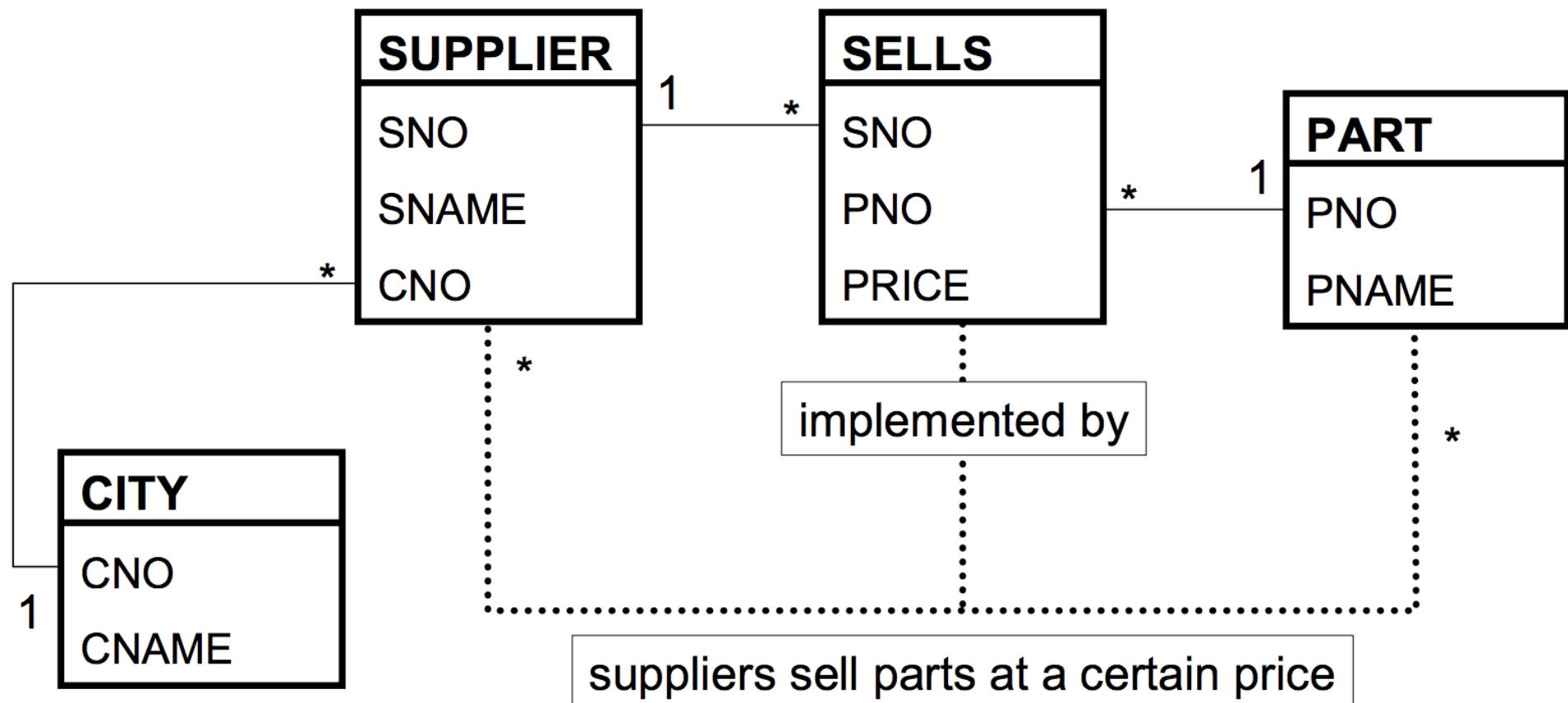


■ 예제: 공급자와 부품

- 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



■ 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)

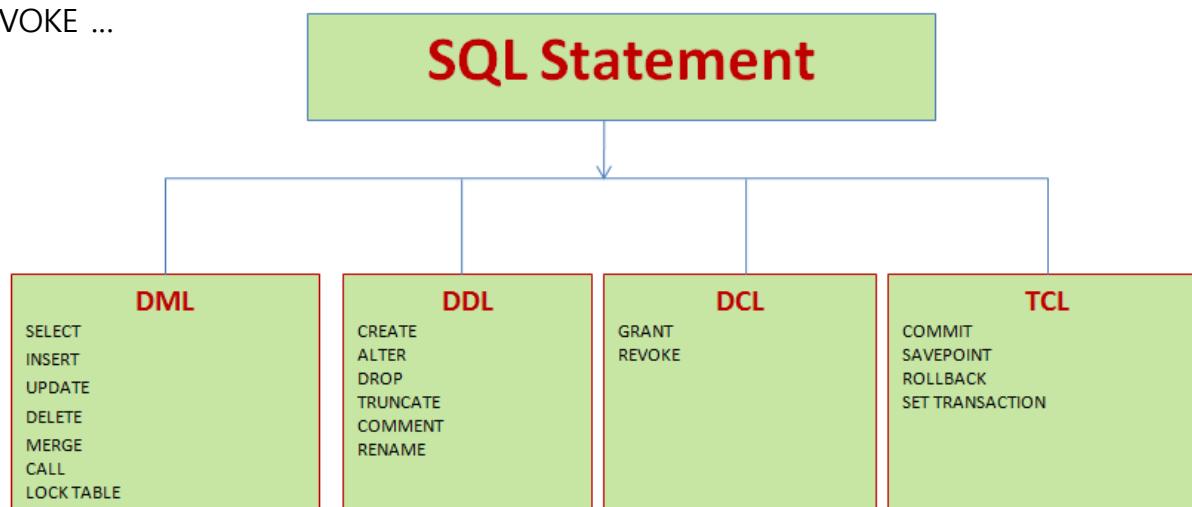
- CREATE, DROP, ALTER, TRUNCATE ...

○ 데이터 조작 언어 DML(Data Manipulation Language)

- INSERT, UPDATE, DELETE, SELECT ...

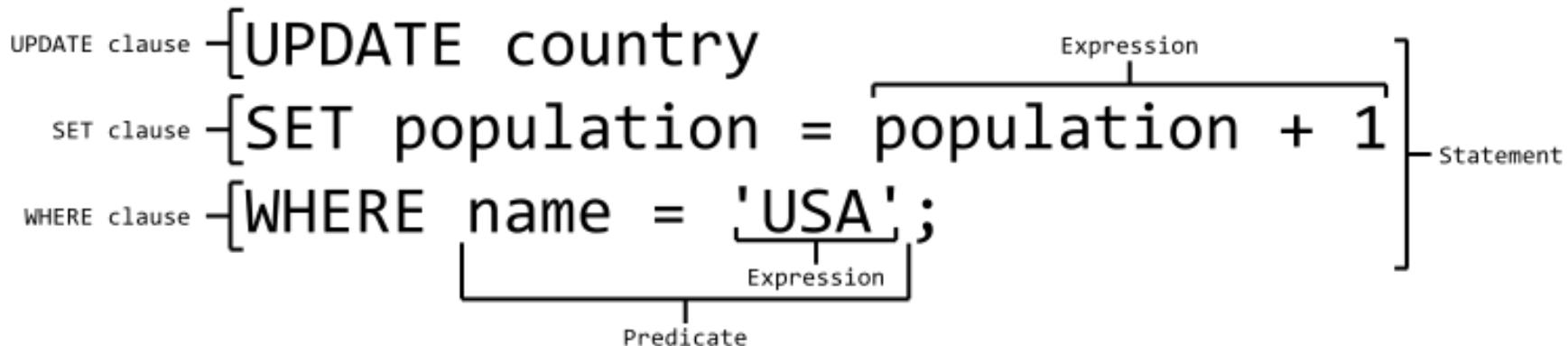
○ 데이터 제어 언어 DCL(Data Control Language)

- GRANT, REVOKE ...



■ SQL 구문

- 문(Statement), 절(Clause), 식(Expression), 솔어(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 :

bookSynopsis **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 :

<i>stat_time</i>	TIME
<i>event_time</i>	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
<i>boolean</i>	Yes/No	Bit	Byte	N/A	Boolean
<i>integer</i>	Number (integer)	Int	Number	Int Integer	Int Integer
<i>float</i>	Number (single)	Float Real	Number	Float	Numeric
<i>currency</i>	Currency	Money	N/A	N/A	Money
<i>string (fixed)</i>	N/A	Char	Char	Char	Char
<i>string (variable)</i>	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
<i>binary object</i>	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												
<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></tbody></table>	C1	C2	1	a	2	b	<code>SELECT * FROM T;</code>	<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></tbody></table>	C1	C2	1	a	2	b
C1	C2													
1	a													
2	b													
C1	C2													
1	a													
2	b													
<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></tbody></table>	C1	C2	1	a	2	b	<code>SELECT C1 FROM T;</code>	<table border="1"><thead><tr><th>C1</th></tr></thead><tbody><tr><td>1</td></tr><tr><td>2</td></tr></tbody></table>	C1	1	2			
C1	C2													
1	a													
2	b													
C1														
1														
2														
<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></tbody></table>	C1	C2	1	a	2	b	<code>SELECT * FROM T WHERE C1 = 1;</code>	<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>1</td><td>a</td></tr></tbody></table>	C1	C2	1	a		
C1	C2													
1	a													
2	b													
C1	C2													
1	a													
<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>1</td><td>a</td></tr><tr><td>2</td><td>b</td></tr></tbody></table>	C1	C2	1	a	2	b	<code>SELECT * FROM T ORDER BY C1 DESC;</code>	<table border="1"><thead><tr><th>C1</th><th>C2</th></tr></thead><tbody><tr><td>2</td><td>b</td></tr><tr><td>1</td><td>a</td></tr></tbody></table>	C1	C2	2	b	1	a
C1	C2													
1	a													
2	b													
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';
```

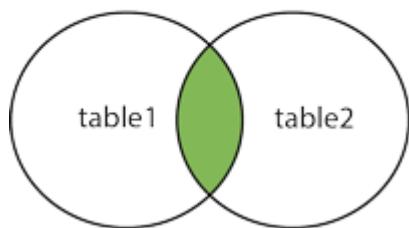
➤ Delete all records

```
DELETE FROM table_name;
```

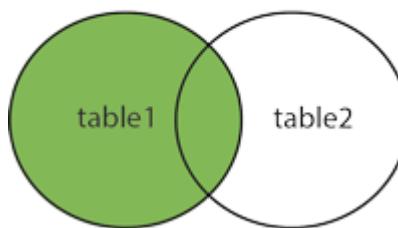
○ Joins

- 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**

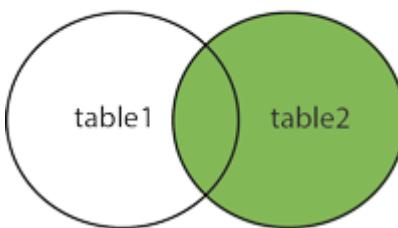
INNER JOIN



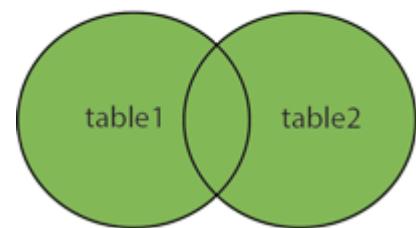
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



- (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		Department table	
LastName	DepartmentID	DepartmentID	DepartmentName
Rafferty	31	31	Sales
Jones	33	33	Engineering
Heisenberg	33	34	Clerical
Robinson	34	35	Marketing
Smith	34		
Williams	NULL		

➤ CROSS JOIN

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

Employee table		Department table	
LastName	DepartmentID	DepartmentID	DepartmentName
Rafferty	31	31	Sales
Jones	33	33	Engineering
Heisenberg	33	34	Clerical
Robinson	34	35	Marketing
Smith	34		
Williams	NULL		

➤ INNER JOIN

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

Employee table		Department table	
LastName	DepartmentID	DepartmentID	DepartmentName
Rafferty	31	31	Sales
Jones	33	33	Engineering
Heisenberg	33	34	Clerical
Robinson	34	35	Marketing
Smith	34	NULL	
Williams	NULL	NULL	

➤ LEFT OUTER JOIN

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

Employee table		Department table	
LastName	DepartmentID	DepartmentID	DepartmentName
Rafferty	31	31	Sales
Jones	33	33	Engineering
Heisenberg	33	34	Clerical
Robinson	34	35	Marketing
Smith	34	NULL	
Williams	NULL	NULL	

➤ RIGHT OUTER JOIN

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

Employee table		Department table	
LastName	DepartmentID	DepartmentID	DepartmentName
Rafferty	31	31	Sales
Jones	33	33	Engineering
Heisenberg	33	34	Clerical
Robinson	34	35	Marketing
Smith	34	NULL	
Williams	NULL	NULL	

➤ 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 SELL WHERE PRICE > 11;
- SELECT SNO, PRICE FROM SELL WHERE PRICE > 11;
- SELECT PNO, PRICE FROM SELL 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 SELL, SUPPLIER, PART (*CROSS JOIN – 8*4*4*)
WHERE SELL.SNO = SUPPLIER.SNO AND SELL.PNO = PART.PNO ORDER BY SNAME, PNAME;

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

1.4. SQLite



■ What is SQLite?

- Open Source Database(Free)
- Serverless(Direct I/O)
- Self-contained(Embedded)
- Single Disk File(Cross-platform)
- Zero-configuration(No setup, No server)
- Supports RDBMS Features(ACID, SQL Syntax, Transactions, etc)

■ SQLite 확인

```
import sqlite3
```

sqlite3.version

The version number of this module, as a string. This is not the version of the SQLite library.

sqlite3.version_info

The version number of this module, as a tuple of integers. This is not the version of the SQLite library.

sqlite3.sqlite_version

The version number of the run-time SQLite library, as a string.

sqlite3.sqlite_version_info

The version number of the run-time SQLite library, as a tuple of integers.

■ Database 연결(생성)

```
conn = sqlite3.connect('경로/이름' or ':memory:')
```

```
sqlite3.connect(database[, timeout, detect_types, isolation_level, check_same_thread, factory, cached_statements, uri])
```

Opens a connection to the SQLite database file *database*. You can use "`:memory:`" to open a database connection to a database that resides in RAM instead of on disk.

When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The *timeout* parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the *timeout* parameter is 5.0 (five seconds).

■ Cursor 생성

```
cur = conn.cursor()
```

cursor(factory=Cursor)

The cursor method accepts a single optional parameter *factory*. If supplied, this must be a callable returning an instance of [Cursor](#) or its subclasses.

```
type(cur)
dir(cur)
```

```
'__new__',
'__setattr__',
'__sizeof__',
 '__str__',
 '__subclasshook__',
'arraysize',
'close',
'connection',
'description',
'execute',
'executemany',
'executescript',
'fetchall',
'fetchmany',
'fetchone',
'lastrowid',
'row_factory',
'rowcount',
'setinputsizes',
'setoutputsizes']
```

■ Data Type

SQLite natively supports the following types: `NULL`, `INTEGER`, `REAL`, `TEXT`, `BLOB`.

Sr.No.	Storage Class & Description
1	NULL The value is a NULL value.
2	INTEGER The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
3	REAL The value is a floating point value, stored as an 8-byte IEEE floating point number.
4	TEXT The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE)
5	BLOB The value is a blob of data, stored exactly as it was input.

■ execute

```
cur.execute('SQL')
```

execute(sql[, parameters])

Executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The `sqlite3` module supports two kinds of placeholders: question marks (qmark style) and named placeholders (named style).

```
cur.execute("create table people (name_last, age)")

who = "Yeltsin"
age = 72

# This is the qmark style:
cur.execute("insert into people values (?, ?)", (who, age))

# And this is the named style:
cur.execute("select * from people where name_last=:who and age=:age", {"who": who, "age": age})

print(cur.fetchone())
```

■ executemany

```
cur.executemany('SQL', params)
```

executemany(sql, seq_of_parameters)

Executes an SQL command against all parameter sequences or mappings found in the sequence *seq_of_parameters*. The `sqlite3` module also allows using an `iterator` yielding parameters instead of a sequence.

```
sql = "insert into people values (?, ?)"  
curData = [('A', 1), ('B', 2), ('C', 3)]  
  
cur.executemany(sql, curData)
```

■ **executescript**

```
cur.executescript(''SQL1; SQL2; ...'')
```

executescript(sql_script)

This is a nonstandard convenience method for executing multiple SQL statements at once. It issues a COMMIT statement first, then executes the SQL script it gets as a parameter.

```
cur.executescript("""
    create table person (
        first_name text primary key,
        last_name  text not null
    );
    insert into person values ('name', 'kim');
""")
```

```
cur.execute('select * from person')
print(cur.fetchall())
```

■ **executescript**

```
cur.executescript(''SQL1; SQL2; ...'')
```

executescript(sql_script)

This is a nonstandard convenience method for executing multiple SQL statements at once. It issues a COMMIT statement first, then executes the SQL script it gets as a parameter.

```
cur.executescript("""
    create table person (
        first_name text primary key,
        last_name  text not null
    );
    insert into person values ('name', 'kim');
""")
```

```
cur.execute('select * from person')
print(cur.fetchall())
```

■ **fetchone**

```
cur.fetchone()
```

fetchone()

Fetches the next row of a query result set, returning a single sequence, or `None` when no more data is available.

■ **fetchmany**

```
cur.fetchmany(size)
```

fetchmany(*size=cursor.arraysize*)

Fetches the next set of rows of a query result, returning a list. An empty list is returned when no more rows are available.

The number of rows to fetch per call is specified by the `size` parameter. If it is not given, the cursor's `arraysize` determines the number of rows to be fetched. The method should try to fetch as many rows as indicated by the `size` parameter. If this is not possible due to the specified number of rows not being available, fewer rows may be returned.

Note there are performance considerations involved with the `size` parameter. For optimal performance, it is usually best to use the `arraysize` attribute. If the `size` parameter is used, then it is best for it to retain the same value from one `fetchmany()` call to the next.

■ fetchall

```
cur.fetchall()
```

fetchall()

Fetches all (remaining) rows of a query result, returning a list. Note that the cursor's arraysize attribute can affect the performance of this operation. An empty list is returned when no rows are available.

■ DDL

○ CREATE

```
conn = sqlite3.connect('create.db')
print("Opened database successfully")

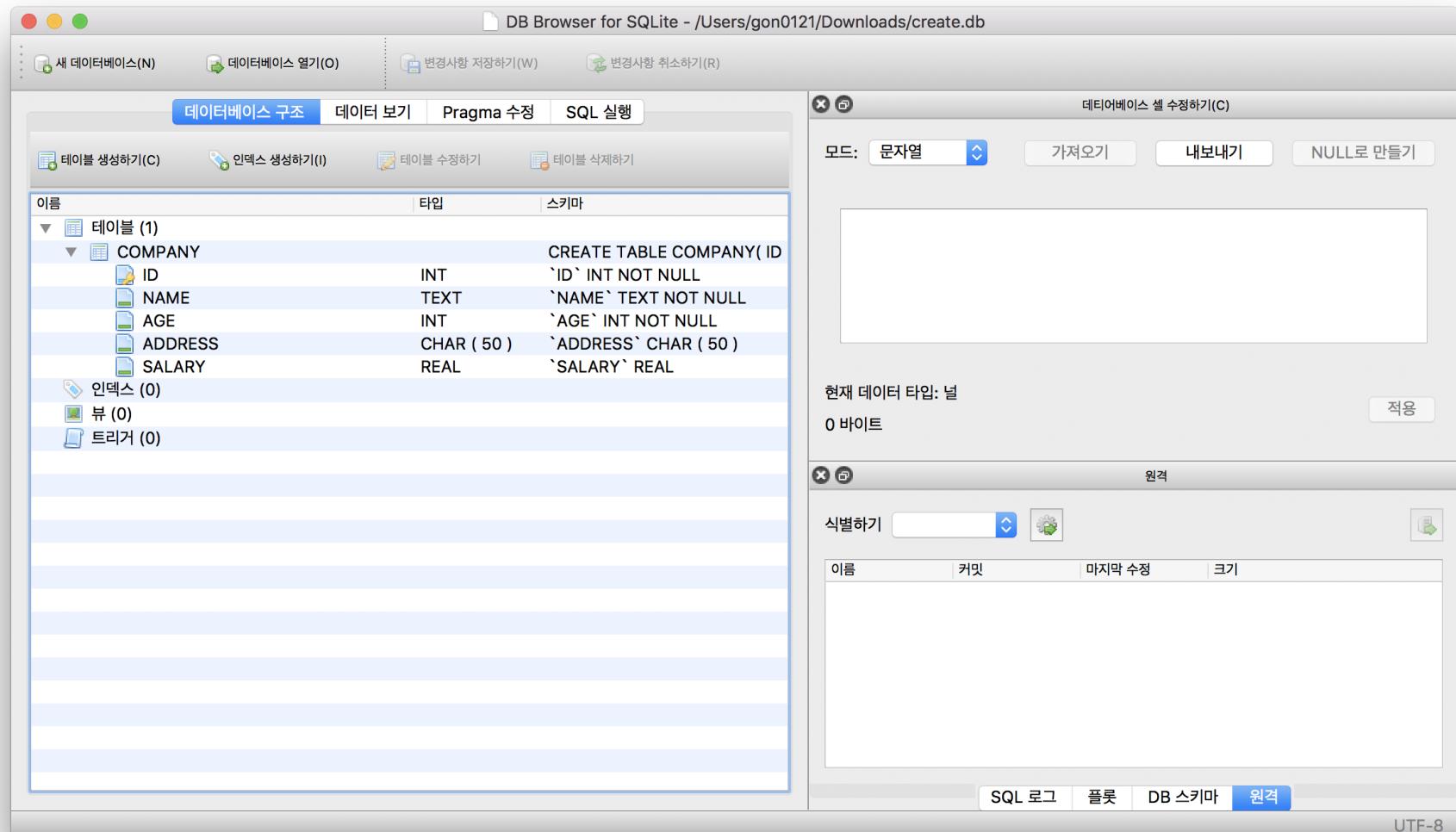
conn.execute('''
    CREATE TABLE COMPANY(
        ID INT PRIMARY KEY     NOT NULL,
        NAME           TEXT    NOT NULL,
        AGE            INT     NOT NULL,
        ADDRESS        CHAR(50),
        SALARY         REAL);
    ''')

print("Table created successfully")
```

○ DROP

```
conn.execute('drop table company')
```

○ CREAT 확인



The screenshot shows the DB Browser for SQLite interface with the database file `/Users/gon0121/Downloads/create.db` open. The left pane displays the schema for the `COMPANY` table:

이름	타입	스키마
COMPANY	CREATE TABLE COMPANY(ID INT `ID` INT NOT NULL, NAME TEXT `NAME` TEXT NOT NULL, AGE INT `AGE` INT NOT NULL, ADDRESS CHAR (50) `ADDRESS` CHAR (50), SALARY REAL `SALARY` REAL)	
ID	INT	`ID` INT NOT NULL
NAME	TEXT	`NAME` TEXT NOT NULL
AGE	INT	`AGE` INT NOT NULL
ADDRESS	CHAR (50)	`ADDRESS` CHAR (50)
SALARY	REAL	`SALARY` REAL

The right pane shows the `데이터베이스 셀 설정하기(C)` dialog, which is currently set to character mode. The bottom right corner of the dialog has a blue "적용" (Apply) button.

At the bottom of the interface, there are tabs for `SQL 로그`, `플롯`, `DB 스키마`, and `원격` (Remote), with `원격` being the active tab. The status bar at the bottom right shows `UTF-8`.

■ DML

○ INSERT

```
conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
              VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
              VALUES (:id, :name, :age, :address, :salary)",
            {'id':2, 'name':'Allen', 'age':25, 'address':'Texas', 'salary':15000.00});

data = [(3, 'Teddy', 23, 'Norway ', 200000.00 ),
        (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )]

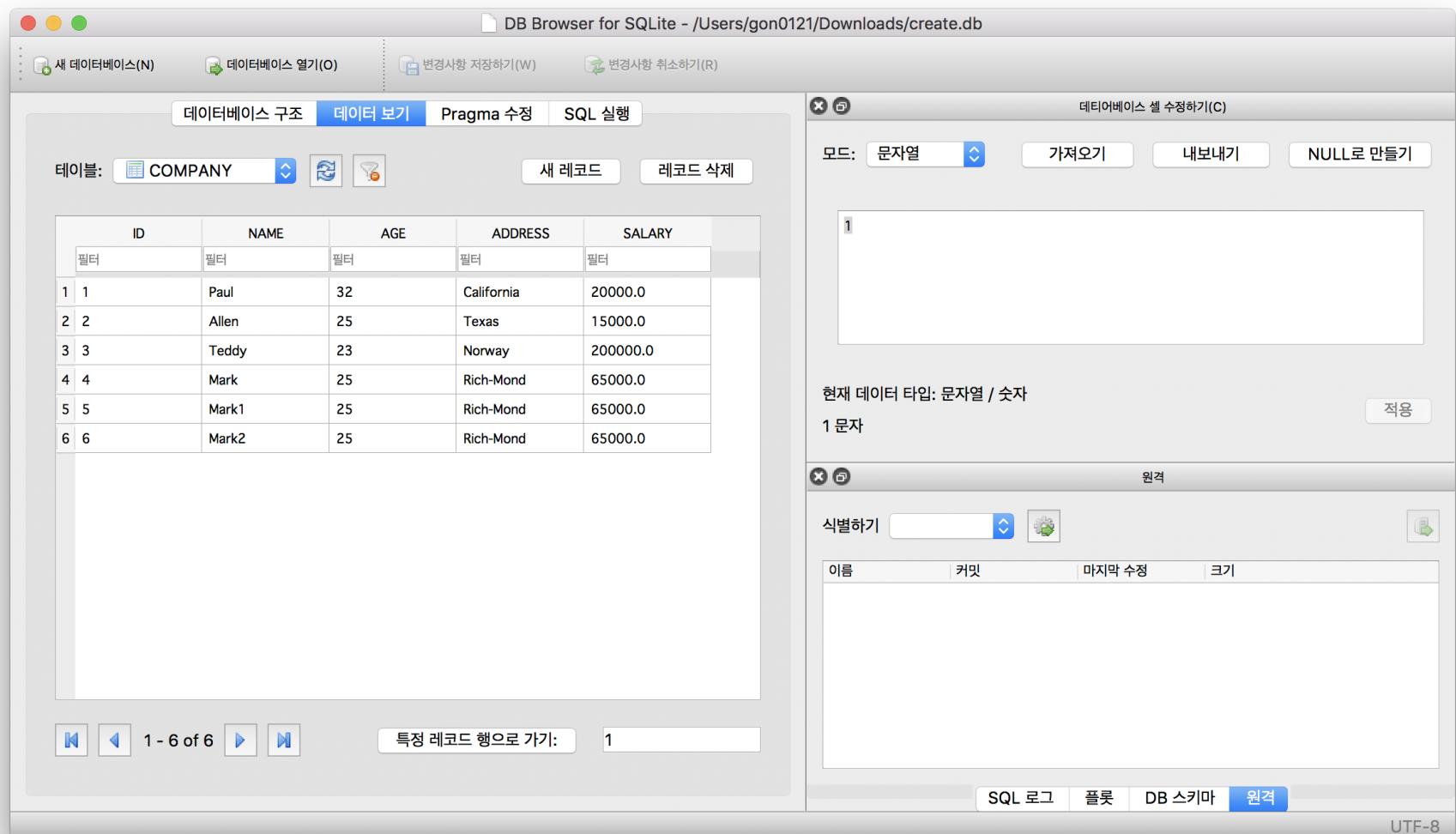
conn.executemany("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
                  VALUES (?, ?, ?, ?, ?)", data);

conn.executescript("""
    INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (5, 'Mark1', 25, 'Rich-Mond ', 65000.00 );

    INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (6, 'Mark2', 25, 'Rich-Mond ', 65000.00 );
""");

conn.commit()
print("Records created successfully")
```

○ INSERT 확인



The screenshot shows the DB Browser for SQLite interface with the following details:

- Toolbar:** Includes "새 데이터베이스(N)", "데이터베이스 열기(O)", "변경사항 저장하기(W)", and "변경사항 취소하기(R)".
- Main Window:** Shows the "COMPANY" table with the following data:

ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000.0
2	Allen	25	Texas	15000.0
3	Teddy	23	Norway	200000.0
4	Mark	25	Rich-Mond	65000.0
5	Mark1	25	Rich-Mond	65000.0
6	Mark2	25	Rich-Mond	65000.0

- Insert Panel:** On the right, a panel titled "데이터베이스 셀 설정하기(C)" is open, showing a single cell with value "1". It includes buttons for "문자열", "가져오기", "내보내기", and "NULL로 만들기".
- Bottom Navigation:** Includes icons for back, forward, and search, a "특정 레코드 행으로 가기:" input field (value: 1), and a "SQL 로그" tab.
- Encoding:** The bottom right corner shows "UTF-8".

SELECT

```
cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("ADDRESS = ", row[2])
    print("SALARY = ", row[3], end='\n\n')

print("Operation done successfully")
```

```
ID = 1
NAME = Paul
ADDRESS = California
SALARY = 20000.0
```

```
ID = 2
NAME = Allen
ADDRESS = Texas
SALARY = 15000.0
```

```
ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY = 200000.0
```

```
ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY = 65000.0
```

```
ID = 5
NAME = Mark1
ADDRESS = Rich-Mond
SALARY = 65000.0
```

```
ID = 6
NAME = Mark2
ADDRESS = Rich-Mond
SALARY = 65000.0
```

```
Operation done successfully
```

○ UPDATE

```
cid = 1

conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = :id", {'id':cid})
conn.commit()

print("Total number of rows updated :", conn.total_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("ADDRESS = ", row[2])
    print("SALARY = ", row[3], "\n")
```

Total number of rows updated : 1

ID = 1

NAME = Paul

ADDRESS = California

SALARY = 25000.0

DELETE

```
conn.execute("DELETE from COMPANY where ID = 2;")

print("Total number of rows deleted :", conn.total_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")
for row in cursor:
    print("ID = ", row[0])
    print("NAME = ", row[1])
    print("ADDRESS = ", row[2])
    print("SALARY = ", row[3], "\n")
```

Total number of rows deleted : 1

ID = 1

NAME = Paul

ADDRESS = California

SALARY = 25000.0

ID = 3

NAME = Teddy

ADDRESS = Norway

SALARY = 200000.0

■ 예제

- Database 생성(열기), Cursor 생성

```
import sqlite3 as sql

con = None

try:
    con = sql.connect('test.db')

    cur = con.cursor()

    cur.execute('SELECT SQLITE_VERSION()')

    data = cur.fetchone()

    print("SQLite Version: ", data)

except Exception as e:
    print("Error: ", e)

finally:
    if con:
        con.close()
```

○ DDL – CREATE, DML – INSERT(execute)

```
con = sql.connect('test.db')

with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")
    cur.execute("INSERT INTO Cars VALUES(1,'Audi',52642)")
    cur.execute("INSERT INTO Cars VALUES(2,'Mercedes',57127)")
    cur.execute("INSERT INTO Cars VALUES(3,'Skoda',9000)")
    cur.execute("INSERT INTO Cars VALUES(4,'Volvo',29000)")
    cur.execute("INSERT INTO Cars VALUES(5,'Bentley',350000)")
    cur.execute("INSERT INTO Cars VALUES(6,'Citroen',21000)")
    cur.execute("INSERT INTO Cars VALUES(7,'Hummer',41400)")
    cur.execute("INSERT INTO Cars VALUES(8,'Volkswagen',21600)")
```

○ DDL – CREATE, DML – INSERT(excutemany)

```
cars = [
    (1, 'Audi', 52642),
    (2, 'Mercedes', 57127),
    (3, 'Skoda', 9000),
    (4, 'Volvo', 29000),
    (5, 'Bentley', 350000),
    (6, 'Hummer', 41400),
    (7, 'Volkswagen', 21600)
]

con = sql.connect('test.db')

with con:
    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS Cars")
    cur.execute("CREATE TABLE Cars(Id INT PRIMARY KEY, Name TEXT, Price INT)")
    cur.executemany("INSERT INTO Cars VALUES(?, ?, ?, ?)", cars)
```

- DDL – DROP, CREATE, DML – INSERT(executeScript)

```
try:  
    con = sql.connect('test.db')  
  
    cur = con.cursor()  
  
    cur.executeScript("""  
        DROP TABLE IF EXISTS Cars;  
        CREATE TABLE Cars(Id INT, Name TEXT, Price INT);  
        INSERT INTO Cars VALUES(1,'Audi',52642);  
        INSERT INTO Cars VALUES(2,'Mercedes',57127);  
        INSERT INTO Cars VALUES(3,'Skoda',9000);  
        INSERT INTO Cars VALUES(4,'Volvo',29000);  
        INSERT INTO Cars VALUES(5,'Bentley',350000);  
        INSERT INTO Cars VALUES(6,'Citroen',21000);  
        INSERT INTO Cars VALUES(7,'Hummer',41400);  
        INSERT INTO Cars VALUES(8,'Volkswagen',21600);  
    """)  
  
    con.commit()  
  
except Exception as e:  
    if con:  
        con.rollback()  
  
        print("Error: ", e)  
  
finally:  
    if con:  
        con.close()
```

- DDL – DROP, CREATE, DML – INSERT(lastrowid)

```
con = sql.connect(':memory:')

with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE Friends(Id INTEGER PRIMARY KEY, Name TEXT);")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Tom');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Rebecca');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Jim');")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Robert');")

    lid = cur.lastrowid
    print("The last Id of the inserted row is", lid)
```

○ DML – SELECT(fetchone)

```
con = sql.connect('test.db')

with con:
    cur = con.cursor()
    cur.execute("SELECT * FROM Cars")

    while True:
        row = cur.fetchone()

        if row == None:
            break

        print(row[0], row[1], row[2])
```

○ DML – SELECT(dictionary, fetchall)

```
con = sql.connect('test.db')

with con:
    con.row_factory = sql.Row

    cur = con.cursor()
    cur.execute("SELECT * FROM Cars")

    rows = cur.fetchall()

    for row in rows:
        print(row["Id"], row["Name"], row["Price"])
```

○ DML – UPDATE(qmark)

```
uId = 1
uPrice = 62300

con = sql.connect('test.db')

with con:
    cur = con.cursor()

    cur.execute("UPDATE Cars SET Price=? WHERE Id=?", (uPrice, uId))

    print("Number of rows updated:", cur.rowcount)
```

○ DML – UPDATE(named)

```
uId = 4

con = sql.connect('test.db')

with con:

    cur = con.cursor()

    cur.execute("SELECT Name, Price FROM Cars WHERE Id=:Id", {"Id": uId})

    row = cur.fetchone()
    print(row[0], row[1])
```

Dump(backup)**iterdump()**

Returns an iterator to dump the database in an SQL text format. Useful when saving an in-memory database for later restoration. This function provides the same capabilities as the `.dump` command in the `sqlite3` shell.

```
# Convert file existing_db.db to SQL dump file dump.sql
import sqlite3

con = sqlite3.connect('existing_db.db')
with open('dump.sql', 'w') as f:
    for line in con.iterdump():
        f.write('%s\n' % line)
```

○ Export / Import

```
cars = (
    (1, 'Audi', 52643),
    (2, 'Mercedes', 57642),
    (3, 'Skoda', 9000),
    (4, 'Volvo', 29000),
    (5, 'Bentley', 350000),
    (6, 'Hummer', 41400),
    (7, 'Volkswagen', 21600)
)

def writeData(data):
    f = open('cars.sql', 'w')

    with f:
        f.write(data)

con = sql.connect(':memory:')

with con:
    cur = con.cursor()

    cur.execute("DROP TABLE IF EXISTS Cars")
    cur.execute("CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")
    cur.executemany("INSERT INTO Cars VALUES(?, ?, ?)", cars)
    cur.execute("DELETE FROM Cars WHERE Price < 30000")

    data = '\n'.join(con.iterdump())

    writeData(data)
```

Export / Import

```
def readData():
    f = open('cars.sql', 'r')

    with f:
        data = f.read()
    return data

con = sql.connect(':memory:')

with con:
    cur = con.cursor()

    rowData = readData()
    cur.executescript(rowData)

    cur.execute("SELECT * FROM Cars")

    rows = cur.fetchall()

    for row in rows:
        print(row)
```

○ Transaction(Commit)

```
try:
    con = sql.connect('test.db')
    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS Friends")
    cur.execute("CREATE TABLE Friends(Id INTEGER PRIMARY KEY, Name TEXT)")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Tom')")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Rebecca')")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Jim')")
    cur.execute("INSERT INTO Friends(Name) VALUES ('Robert')")

    #con.commit()

except Exception as e:

    if con:
        con.rollback()

    print("Error:", e)

finally:
    if con:
        con.close()
```

■ 예제 (ER Model – SQL)

Track	Length	Artist	Album	Genre	Rating	Count
<input checked="" type="checkbox"/> Hell's Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
<input checked="" type="checkbox"/> For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders ...	New Age		31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Sister Golden Hair	3:22	America	Greatest Hits	Easy Listen...	★★★★★	24
<input checked="" type="checkbox"/> Track 01	4:22	Billy Price	Danger Zone	Blues/R&B	★★★★★	26
<input checked="" type="checkbox"/> Track 02	2:45	Billy Price	Danger Zone	Blues/R&B	★★★★★	18
<input checked="" type="checkbox"/> Track 03	3:26	Billy Price	Danger Zone	Blues/R&B	★★★★★	22
<input checked="" type="checkbox"/> Track 04	4:17	Billy Price	Danger Zone	Blues/R&B	★★★★★	18
<input checked="" type="checkbox"/> Track 05	3:50	Billy Price	Danger Zone	Blues/R&B	★★★★★	21
<input checked="" type="checkbox"/> War Pigs/Luke's Wall	7:58	Black Sabbath	Paranoid	Metal	★★★★★	25
<input checked="" type="checkbox"/> Paranoid	2:53	Black Sabbath	Paranoid	Metal	★★★★★	22
<input checked="" type="checkbox"/> Planet Caravan	4:35	Black Sabbath	Paranoid	Metal	★★★★★	25
<input checked="" type="checkbox"/> Iron Man	5:59	Black Sabbath	Paranoid	Metal	★★★★★	26
<input checked="" type="checkbox"/> Electric Funeral	4:53	Black Sabbath	Paranoid	Metal	★★★★★	22
<input checked="" type="checkbox"/> Hand of Doom	7:10	Black Sabbath	Paranoid	Metal	★★★★★	23
<input checked="" type="checkbox"/> Rat Salad	2:30	Black Sabbath	Paranoid	Metal	★★★★★	31
<input checked="" type="checkbox"/> Jack the Stripper/Fairies Wear ...	6:14	Black Sabbath	Paranoid	Metal	★★★★★	24
<input checked="" type="checkbox"/> Bomb Squad (TECH)	3:28	Brent	Brent's Album			1
<input checked="" type="checkbox"/> clay techno	4:36	Brent	Brent's Album			2
<input checked="" type="checkbox"/> Heavy	3:08	Brent	Brent's Album			1
<input checked="" type="checkbox"/> Hi metal man	4:20	Brent	Brent's Album			1
<input checked="" type="checkbox"/> Mistro	2:58	Brent	Brent's Album			1

- 개체(Entity) – 관계(Relationship)

Track

Length

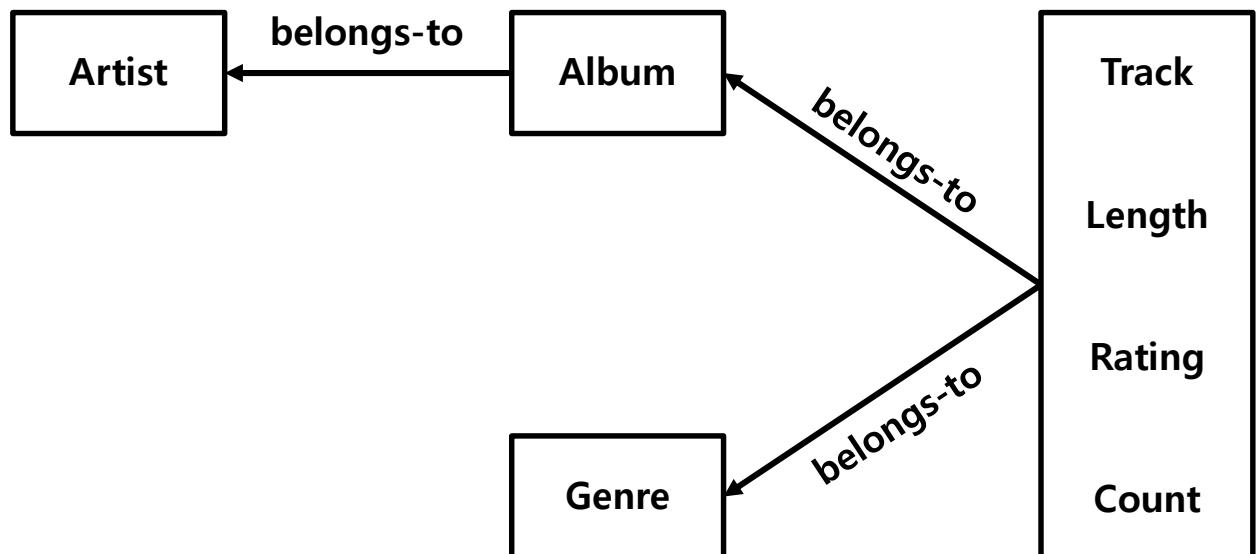
Artist

Album

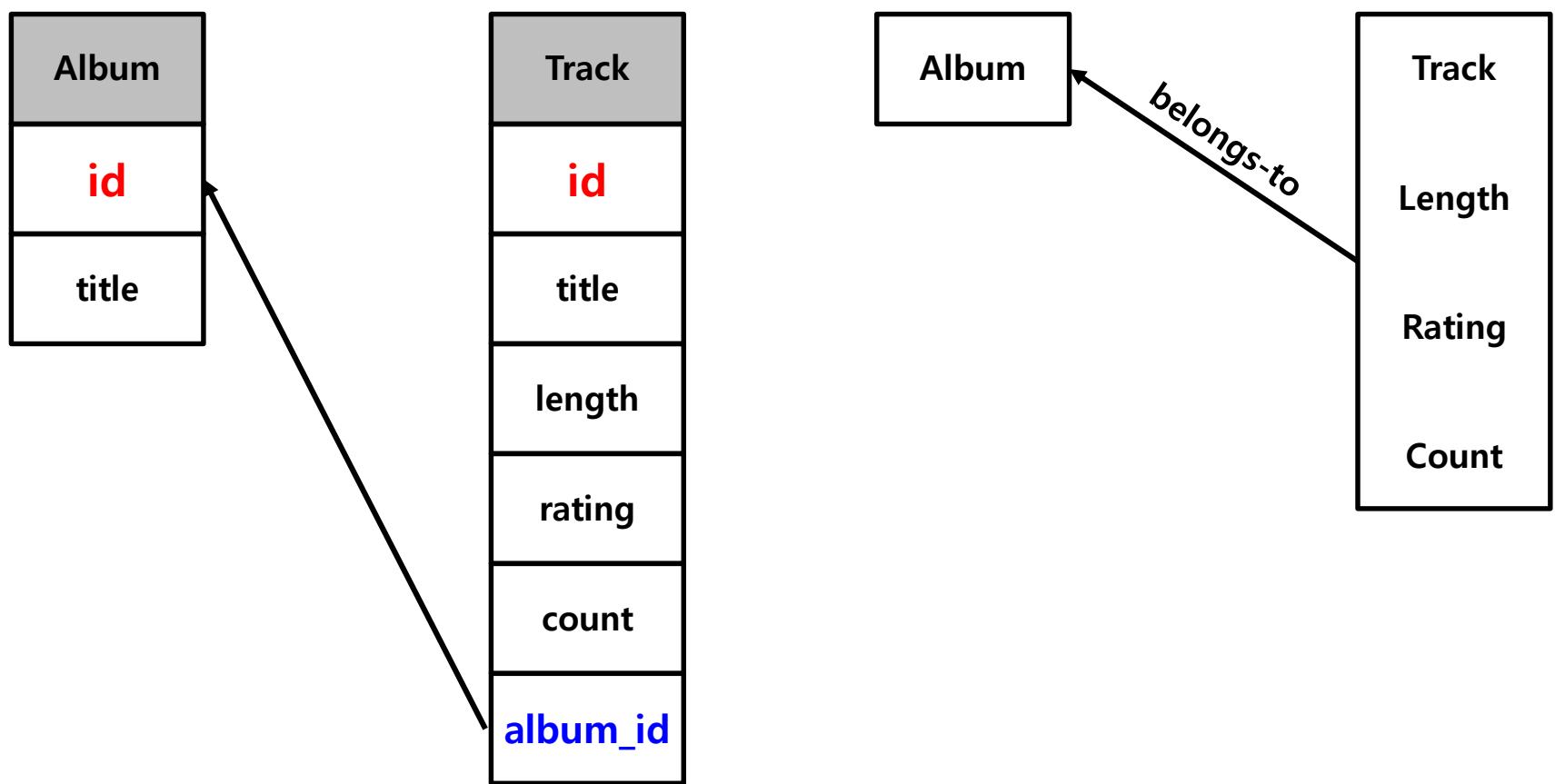
Genre

Rating

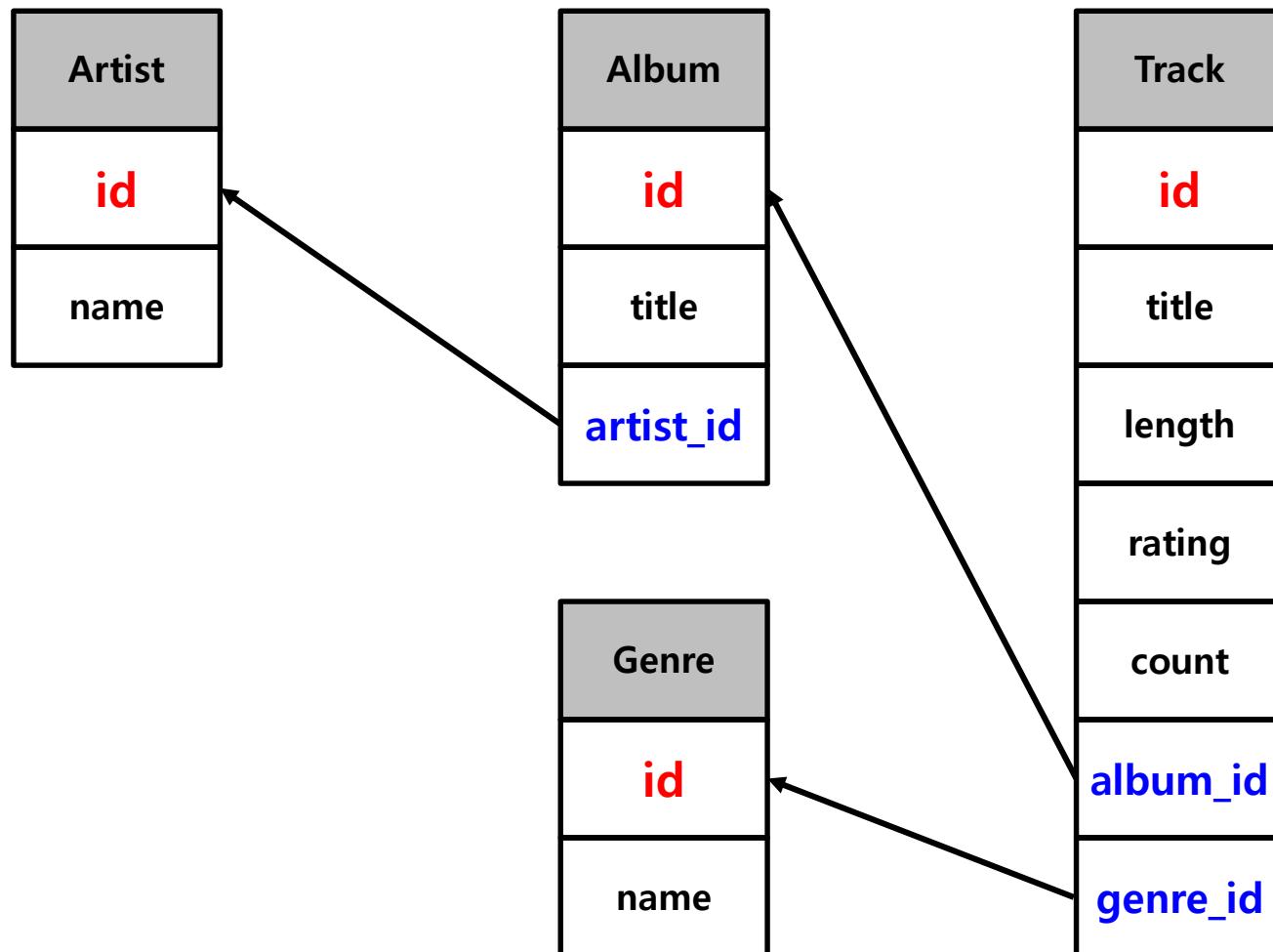
Count



- 기본키(Primary Key) / 참조키(Foreign Key)



○ 기본키(Primary Key) / 참조키(Foreign Key)



○ TABLE 생성

➤ Artist

```
- CREATE TABLE Artist (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name TEXT
)
```

➤ Genre

```
- CREATE TABLE Genre (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name TEXT
)
```

➤ Album

```
- CREATE TABLE Album (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title TEXT,
    artist_id INTEGER
)
```

➤ Track

- CREATE TABLE Track (
 - id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
 - title TEXT,
 - length INTEGER,
 - rating INTEGER,
 - count INTEGER,
 - album_id INTEGER,
 - genre_id INTEGER)

○ INSERT

➤ Artist

- insert into Artist (name) values ('Led Zeppelin')
- insert into Artist (name) values ('AC/DC')

➤ Genre

- insert into Genre (name) values ('Rock')
- insert into Genre (name) values ('Metal')

➤ Album

- insert into Album (title, artist_id) values ('Who Made Who', 2)
- insert into Album (title, artist_id) values ('IV', 1)

➤ Track

- insert into Track (title, rating, length, count, album_id, genre_id) values ('Black Dog', 5, 297, 0, 2, 1)
- insert into Track (title, rating, length, count, album_id, genre_id) values ('Stairway', 5, 482, 0, 2, 1)
- insert into Track (title, rating, length, count, album_id, genre_id) values ('About to Rock', 5, 313, 0, 1, 2)
- insert into Track (title, rating, length, count, album_id, genre_id) values ('Who Made Who', 5, 207, 0, 1, 2)

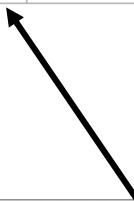
id		name
필터	필터	
1	1	Led Zeppelin
2	2	AC/DC

id		name
필터	필터	
1	1	Rock
2	2	Metal

id			title	artist_id
필터	필터	필터		
1	1		Who Made Who	2
2	2		IV	1

id								title	length	rating	count	album_id	genre_id
필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터
1	1				Black Dog	297	5	0		2		1	
2	2				Stairway	482	5	0		2		1	
3	3				About to Rock	313	5	0		1		2	
4	4				Who Made Who	207	5	0		1		2	

	id	name
	필터	필터
1	1	Led Zepplin
2	2	AC/DC



	id	title	artist_id
	필터	필터	필터
1	1	Who Made Who	2
2	2	IV	1

○ Join

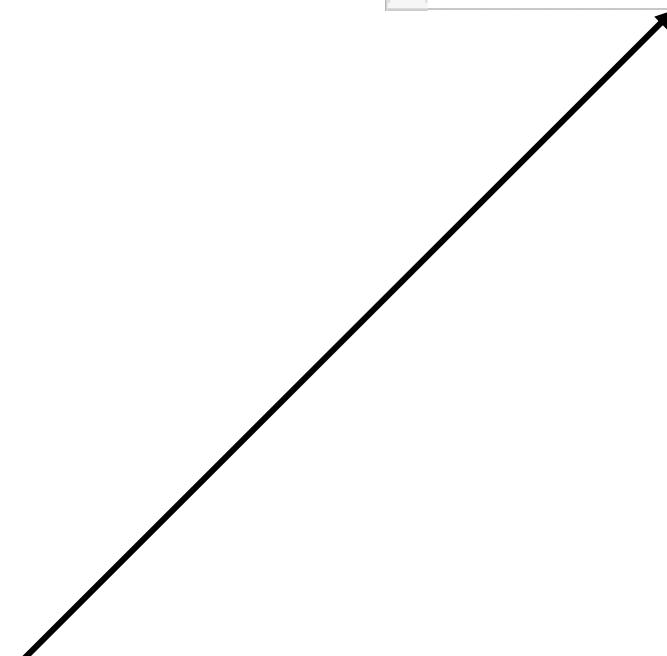
➤ select Album.title, Artist.name
 from Album
 join Artist on
 Album.artist_id = Artist.id

	title	name
1	Who Made Who	AC/DC
2	IV	Led Zepplin

```
> select Track.title, Genre.name
   from Track
   join Genre on
     Track.genre_id = Genre.id
```

	title	name
1	Black Dog	Rock
2	Stairway	Rock
3	About to Rock	Metal
4	Who Made Who	Metal

	id	name
	필터	필터
1	1	Rock
2	2	Metal

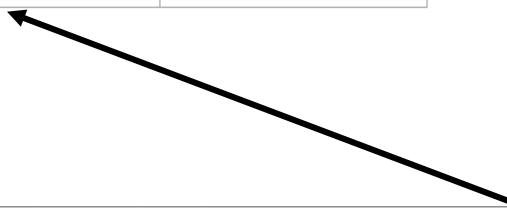


	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

➤ select Album.title, Track.title
 from Track
 join Album on
 Track.album_id = Album.id

	title	title
1	IV	Black Dog
2	IV	Stairway
3	Who Made Who	About to Rock
4	Who Made Who	Who Made Who

	id	title	artist_id
	필터	필터	필터
1	1	Who Made Who	2
2	2	IV	1



	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

➤ select Track.title, Artist.name, Album.title, Genre.name

from Track

join Artist join Album join Genre on

Track.album_id = Album.id

and

Track.genre_id = Genre.id

and

Album.artist_id = Artist.id

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

	id	title	length	rating	count	album_id	genre_id
	필터	필터	필터	필터	필터	필터	필터
1	1	Black Dog	297	5	0	2	1
2	2	Stairway	482	5	0	2	1
3	3	About to Rock	313	5	0	1	2
4	4	Who Made Who	207	5	0	1	2

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

<input checked="" type="checkbox"/> Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
<input checked="" type="checkbox"/> For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23

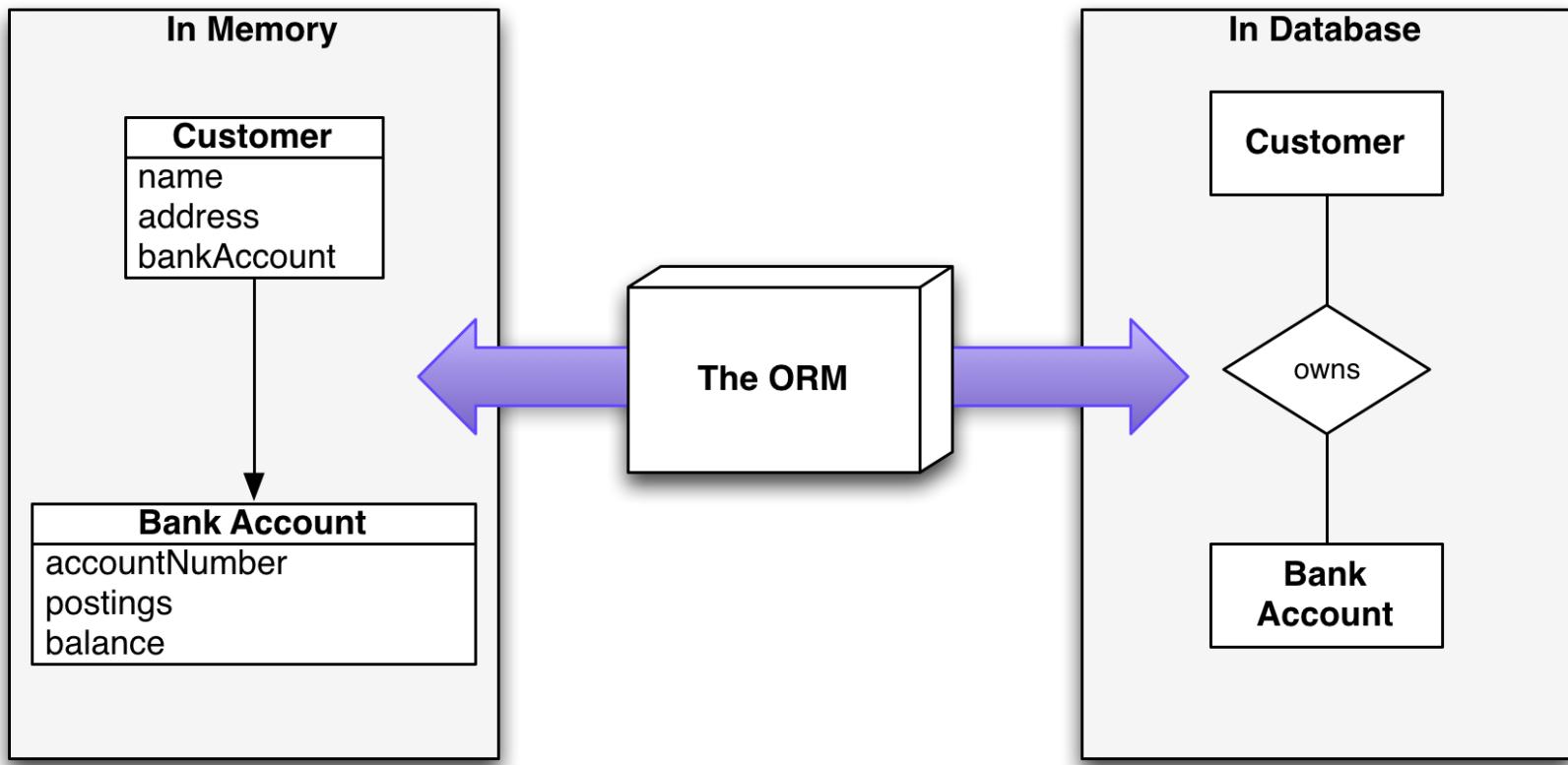
2. ORM, RE

2.1. ORM 소개

2.2. SQLAlchemy

2.3. Regular Expression

2.1. ORM 소개



■ What is ORM?

- ORM stands Object Relational Mapping
- Programming technique for converting data between incompatible type systems using object-oriented programming languages

■ Why use ORM?

- Mismatch between the object model and the relational database
 - RDBSs represent data in tabular format
 - Object-Oriented languages represent data as an interconnected graph of objects
- ORM frees the programmer from dealing with simple repetitive database queries
- Automatically mapping the database to business objects
- Programmers focus more on business problems and less with data storage
- The mapping process can aid in data verification and security before reaching the database
- ORM can provide a caching layer above the database

■ Disadvantages

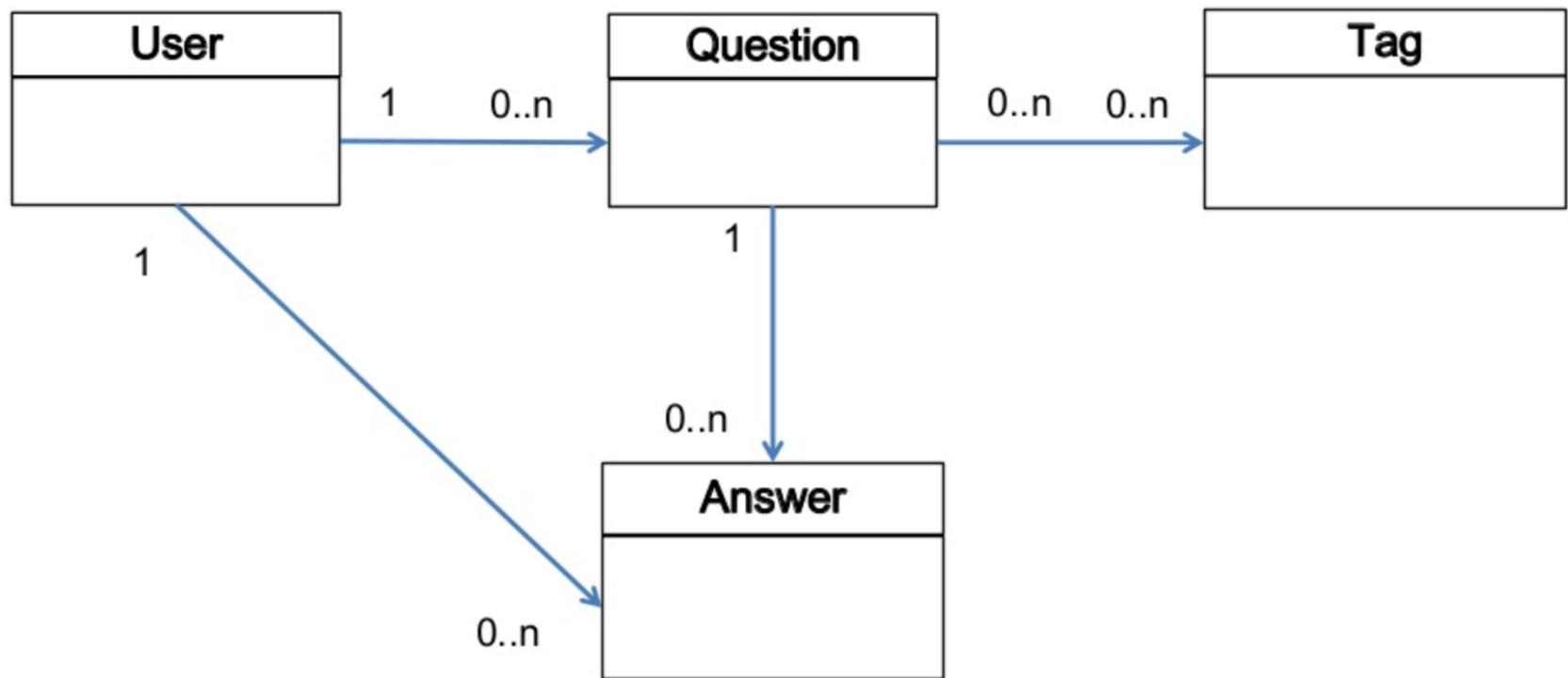
- Potentially increasing processing overhead

■ 예제

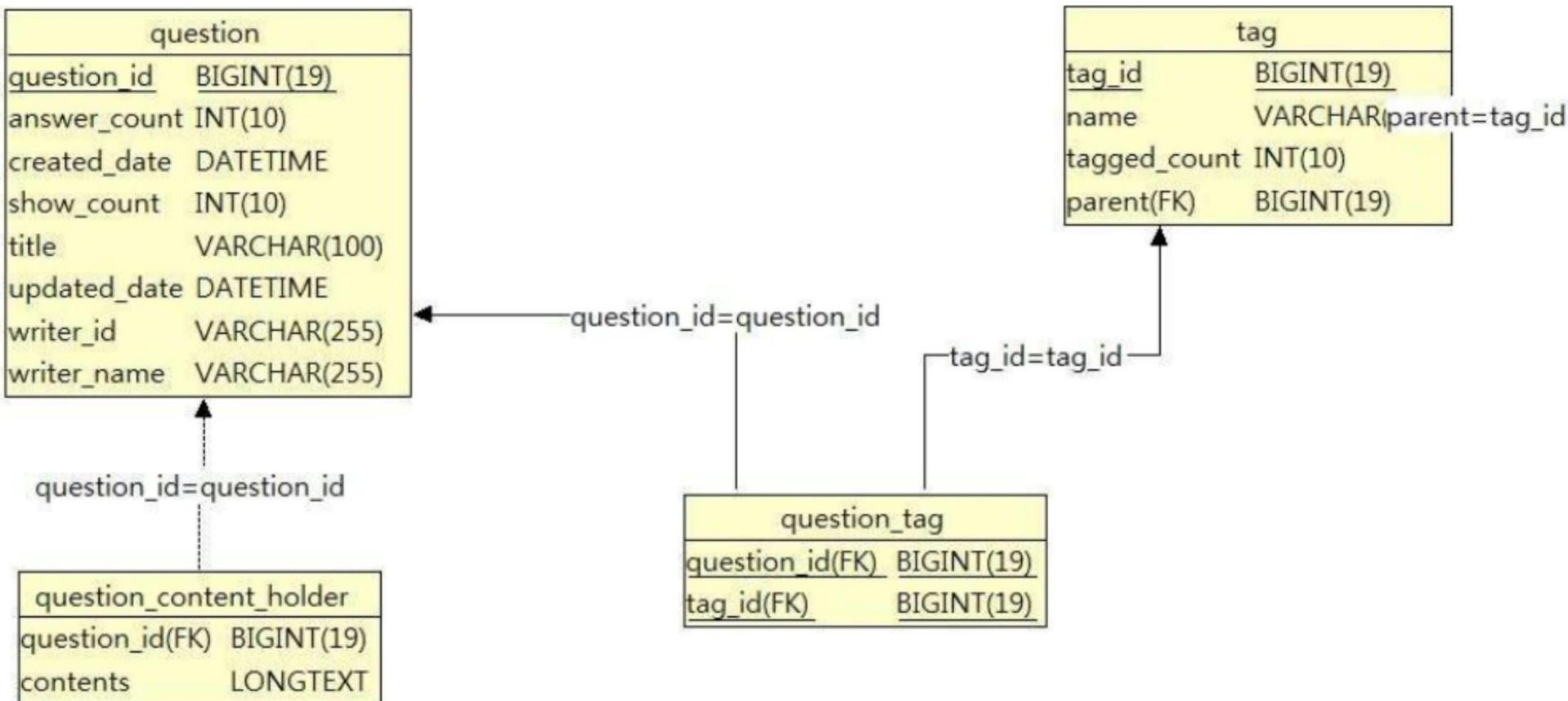
○ 요구사항

- 사용자는 질문 할 수 있어야 한다
- 질문에 대한 답변을 할 수 있어야 한다
- 질문할 때 태그를 추가할 수 있어야 한다
- 태그는 태그 풀에 존재하는 태그만 추가할 수 있다
- 태그가 추가될 경우 해당 태그 수는 +1 증가, 삭제될 경우 해당 태그 수는 -1 감소되어야 한다

○ 개체-관계 모델(ER Model)



○ 데이터베이스 설계



○ 질문 추가 시

- 질문 등록
- 태그 등록
 - 태그 풀에서 해당 태그 ID 가져오기
 - 태그 풀에서 해당 태그 수 증감하기

```
INSERT INTO question VALUES(?, ?, ?, ?, ?, ?); → question_id = 1
```

```
SELECT tag_id, name FROM tag WHERE name="python"; → tag_id = 1
```

```
SELECT tag_id, name FROM tag WHERE name="alchemy"; → tag_id = 2
```

```
INSERT INTO question_tag VALUES(1, 1);
```

```
INSERT INTO question_tag VALUES(1, 2);
```

```
UPDATE tag SET tagged_count = tagged_count + 1 where name="python";
```

```
UPDATE tag SET tagged_count = tagged_count + 1 where name="alchemy";
```

○ 질문 수정 시

- 질문 수정
- 태그 수정
 - 태그 풀에서 해당 태그 ID 가져오기
 - 태그 풀에서 해당 태그 수 증감하기

```
UPDATE question SET title=?, contents=? WHERE question_id = 1;
```

```
SELECT tag_id, name FROM tag WHERE name="python"; → tag_id = 1
```

```
SELECT tag_id, name FROM tag WHERE name="alchemy"; → tag_id = 2
```

```
SELECT tag_id, name FROM tag WHERE name="orm"; → tag_id = 3
```

```
INSERT INTO question_tag VALUES(1, 3);
```

```
DELETE FROM question_tag where question_id=1 and tag_id=2;
```

```
UPDATE tag SET tagged_count = tagged_count - 1 where name="alchemy"
```

```
UPDATE tag SET tagged_count = tagged_count + 1 where name="orm"
```

테이블 간의 관계보다 **데이터베이스 대한 처리**에 집중

○ 객체-관계로 접근

```
public class Question {  
    private long qid;  
    private string title;  
    [ ... ]  
  
    public processTags(string tag) {  
    }  
}
```

```
public class Tag {  
    private long tid;  
    private string tag;  
    [ ... ]  
  
    public add(string tag) {  
    }  
}
```

INSERT INTO question VALUES(?, ?, ?, ?, ?, ?); → question(1)
SELECT tag_id, name FROM tag WHERE name="python"; → tag.getByName("python")

■ ORM in Python

- ORM allows a developer to write Python code instead of SQL to create, read, update and delete
 - Developers can use the programming language they are comfortable with to work with a database instead of writing SQL statements or stored procedures
- ORMs make it theoretically possible to switch an application between various relational databases
 - In practice however, it's best to use the same database for local development as is used in production

Relational database (such as PostgreSQL or MySQL)

ID	FIRST_NAME	LAST_NAME	PHONE
1	John	Connor	+16105551234
2	Matt	Makai	+12025555689
3	Sarah	Smith	+19735554512
...

Python objects

```
class Person:  
    first_name = "John"  
    last_name = "Connor"  
    phone_number = "+16105551234"
```

```
class Person:  
    first_name = "Matt"  
    last_name = "Makai"  
    phone_number = "+12025555689"
```

```
class Person:  
    first_name = "Sarah"  
    last_name = "Smith"  
    phone_number = "+19735554512"
```

ORMs provide a bridge between
**relational database tables, relationships
and fields and Python objects**

■ ORMs

- Python ORM libraries are not required for accessing relational databases
 - In fact, the low-level access is typically provided by another library called a *database connector*

web framework	None	Flask	Flask	Django
ORM	SQLAlchemy	SQLAlchemy	SQLAlchemy	Django ORM
database connector	(built into Python stdlib)	MySQL-python	psycopg	psycopg
relational database	 SQLite	 MySQL	 PostgreSQL	 PostgreSQL

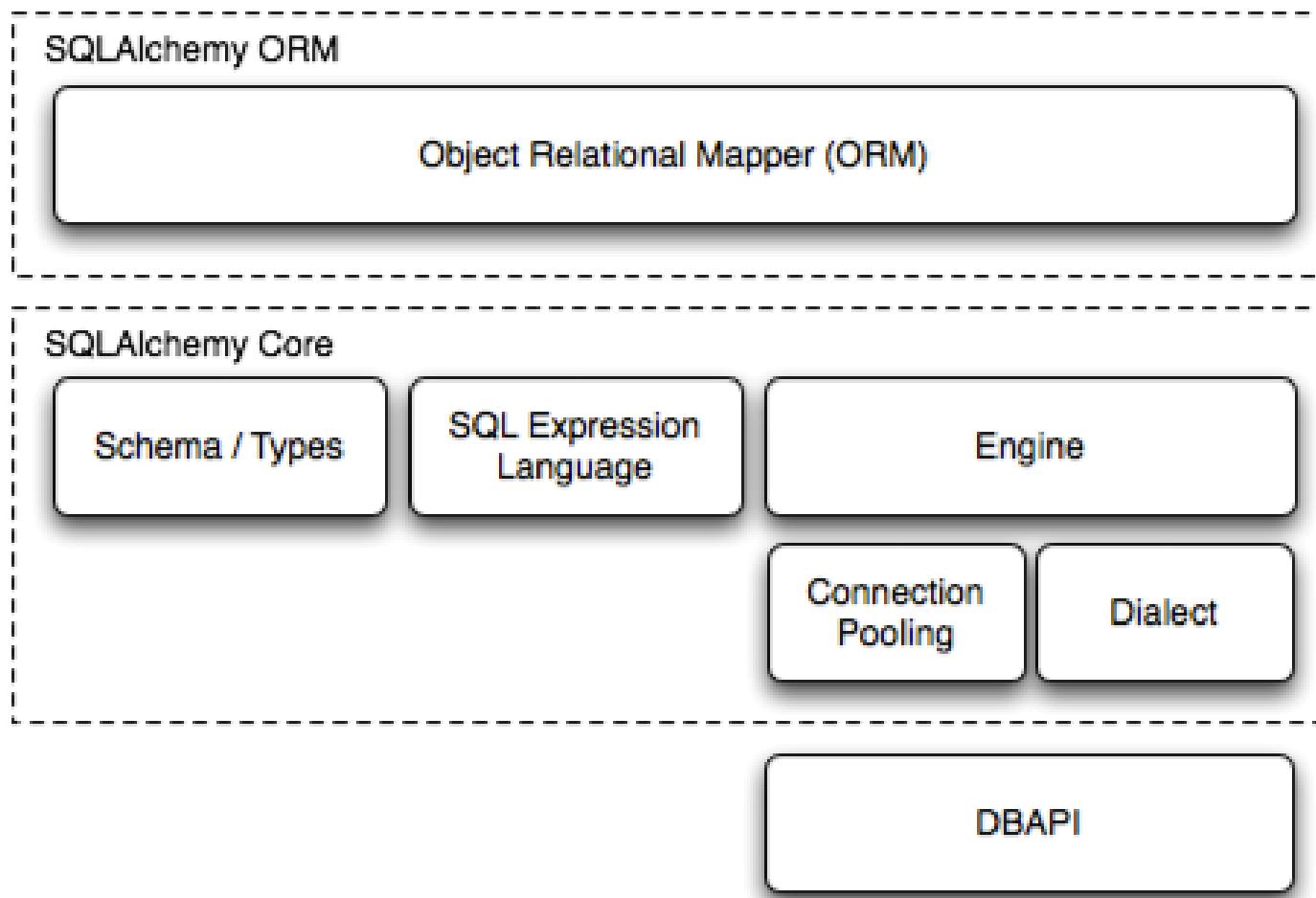
2.2. SQLAlchemy



■ What is SQLAlchemy?

- SQLAlchemy is a well-regarded database toolkit and ORM implementation written in Python
- SQLAlchemy provides a generalized interface for creating and executing database-agnostic code without needing to write SQL statements.

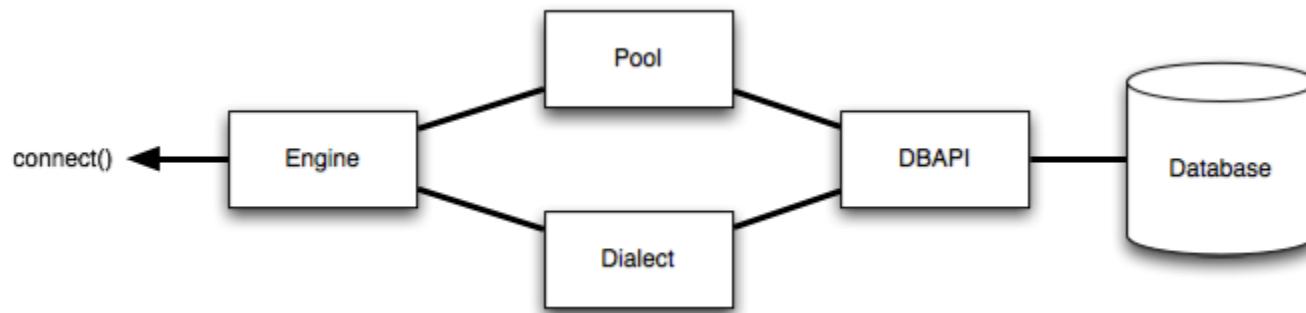
■ SQLAlchemy architecture



■ SQLAlchemy CORE

○ Engine

- starting point for any SQLAlchemy application
- a registry which provides connectivity to a particular database server



○ Dialect

- communicate with various types of DBAPI implementations and databases
- interprets generic SQL and database commands in terms of specific DBAPI and database backend
 - Firebird, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SQLite, Sybase

○ Connection Pool

- holds a collection of database connections in memory for fast re-use

■ Connecting

- create_engine

```
sqlalchemy.create_engine(*args, **kwargs)
```

```
dialect+driver://username:password@host:port/database
```

```
# sqlite://<hostname>/<path>
# where <path> is relative:
engine = create_engine('sqlite:///foo.db')
```

- 예제

```
import sqlalchemy
sqlalchemy.__version__
```

```
from sqlalchemy import create_engine

engine = create_engine("sqlite://", echo=True)
#engine = create_engine("sqlite:///memory:", echo=True)
#engine = create_engine("sqlite:///test.db", echo=True)

print(engine)
```

- Lazy connecting
- The echo flag is a shortcut to setting up SQLAlchemy logging

■ Create

○ Table

```
class sqlalchemy.schema.Table(*args, **kw)
```

- Table object constructs a unique instance of itself based on its name and optional schema name within the given MetaData object
- name, metadata, columns, constraint, ...

○ Column

```
class sqlalchemy.schema.Column(*args, **kwargs)
```

- Represents a column in a database table
- name, type, constraint, autoincrement, default, nullable, ...

○ MetaData

```
class sqlalchemy.schema.MetaData
```

- A collection of Table objects and their associated schema constructs
- Holds a collection of Table objects as well as an optional binding to an Engine or Connection. If bound, the Table objects in the collection and their columns may participate in implicit SQL execution
- Table objects themselves are stored in the MetaData.tables dictionary

○ 예제

```
from sqlalchemy import Table, Column, Integer, String, MetaData, ForeignKey

metadata = MetaData()
users = Table('users', metadata,
    Column('id', Integer, primary_key=True),
    Column('name', String),
    Column('fullname', String),
)

addresses = Table('addresses', metadata,
    Column('id', Integer, primary_key=True),
    Column('user_id', None, ForeignKey('users.id'))),
    Column('email_address', String, nullable=False)
)

metadata.create_all(engine)
```

■ Insert

○ insert

```
insert(values=None, inline=False, **kwargs)
```

- Represents an INSERT construct
- generate an insert() construct against this TableClause

○ compile

```
compile(bind=None, dialect=None, **kw)
```

- Compile this SQL expression
- Return value is a Compiled object
- Compiled object also can return a dictionary of bind parameter names and values using the params accessor

○ 예제

```
insert = users.insert()  
print(insert)  
  
insert = users.insert().values(name='kim', fullname='Anonymous, Kim')  
print(insert)  
  
insert.compile().params
```

■ Executing

○ Connection

```
class sqlalchemy.engine.Connection
```

- Provides high-level functionality for a wrapped DB-API connection
- Provides execution support for string-based SQL statements as well as ClauseElement, Compiled and DefaultGenerator objects

○ execute

```
execute(object, *multiparams, **params)
```

- Executes a SQL statement construct and returns a ResultProxy

○ ResultProxy

```
class sqlalchemy.engine.ResultProxy(context)
```

- Wraps a DB-API cursor object to provide easier access to row columns

○ 예제

```
conn = engine.connect()
conn

insert.bind = engine
str(insert)

result = conn.execute(insert)

result.inserted_primary_key
```

➤ execute의 params 사용

```
insert = users.insert()

result = conn.execute(insert, name="lee", fullname="Unknown, Lee")

result.inserted_primary_key
```

➤ DBAPI의 executemany() 사용

```
conn.execute(addresses.insert(), [
    {"user_id":1, "email_address":"anonymous.kim@test.com"},
    {"user_id":2, "email_address":"unknown.lee@test.com"}
])
```

■ Select

○ select

```
sqlalchemy.sql.expression.select
```

- Construct a new Select
- columns, whereclause, from_obj, group_by, order_by, ...

○ 예제

```
from sqlalchemy.sql import select

query = select([users])
result = conn.execute(query)

for row in result:
    print(row)
```

```
result = conn.execute(select([users.c.name, users.c.fullname]))

for row in result:
    print(row)
```

■ ResultProxy

○ fetchone

fetchone()

- Fetch one row, just like DB-API cursor.fetchone()

○ fetchall

fetchall()

- Fetch all rows, just like DB-API cursor.fetchall()

○ 예제

```
result = conn.execute(query)

row = result.fetchone()
print("id -", row["id"], ", name -", row["name"], ", fullname -", row["fullname"])

row = result.fetchone()
print("id -", row[0], ", name -", row[1], ", fullname -", row[2])

result = conn.execute(query)
rows = result.fetchall()

for row in rows:
    print("id -", row[0], ", name -", row[1], ", fullname -", row[2])

result.close()
```

■ Conjunctions

○ 예제

```
from sqlalchemy import and_, or_, not_

print(users.c.id == addresses.c.user_id)

print(users.c.id == 1)

print((users.c.id == 1).compile().params)

print(or_(users.c.id == addresses.c.user_id, users.c.id == 1))

print(and_(users.c.id == addresses.c.user_id, users.c.id == 1))

print(and_(
    or_(
        users.c.id == addresses.c.user_id,
        users.c.id == 1
    ),
    addresses.c.email_address.like("a%")
)
)

print((
    (users.c.id == addresses.c.user_id) |
    (users.c.id == 1)
) & (addresses.c.email_address.like("a%")))
```

■ Selecting

○ 예제

```
result = conn.execute(select([users]).where(users.c.id==1))
for row in result:
    print(row)

result = conn.execute(select([users, addresses]).where(users.c.id==addresses.c.user_id))
for row in result:
    print(row)

result = conn.execute(select([users.c.id, users.c.fullname, addresses.c.email_address])
                      .where(users.c.id==addresses.c.user_id))
for row in result:
    print(row)

result = conn.execute(select([users.c.id, users.c.fullname, addresses.c.email_address])
                      .where(users.c.id==addresses.c.user_id)
                      .where(addresses.c.email_address.like("un%")))
for row in result:
    print(row)
```

■ Join

○ join

```
join(right, onclause=None, isouter=False, full=False)
```

- Return a Join from this FromClause to another FromClause

○ 예제

```
from sqlalchemy import join

print(users.join(addresses))

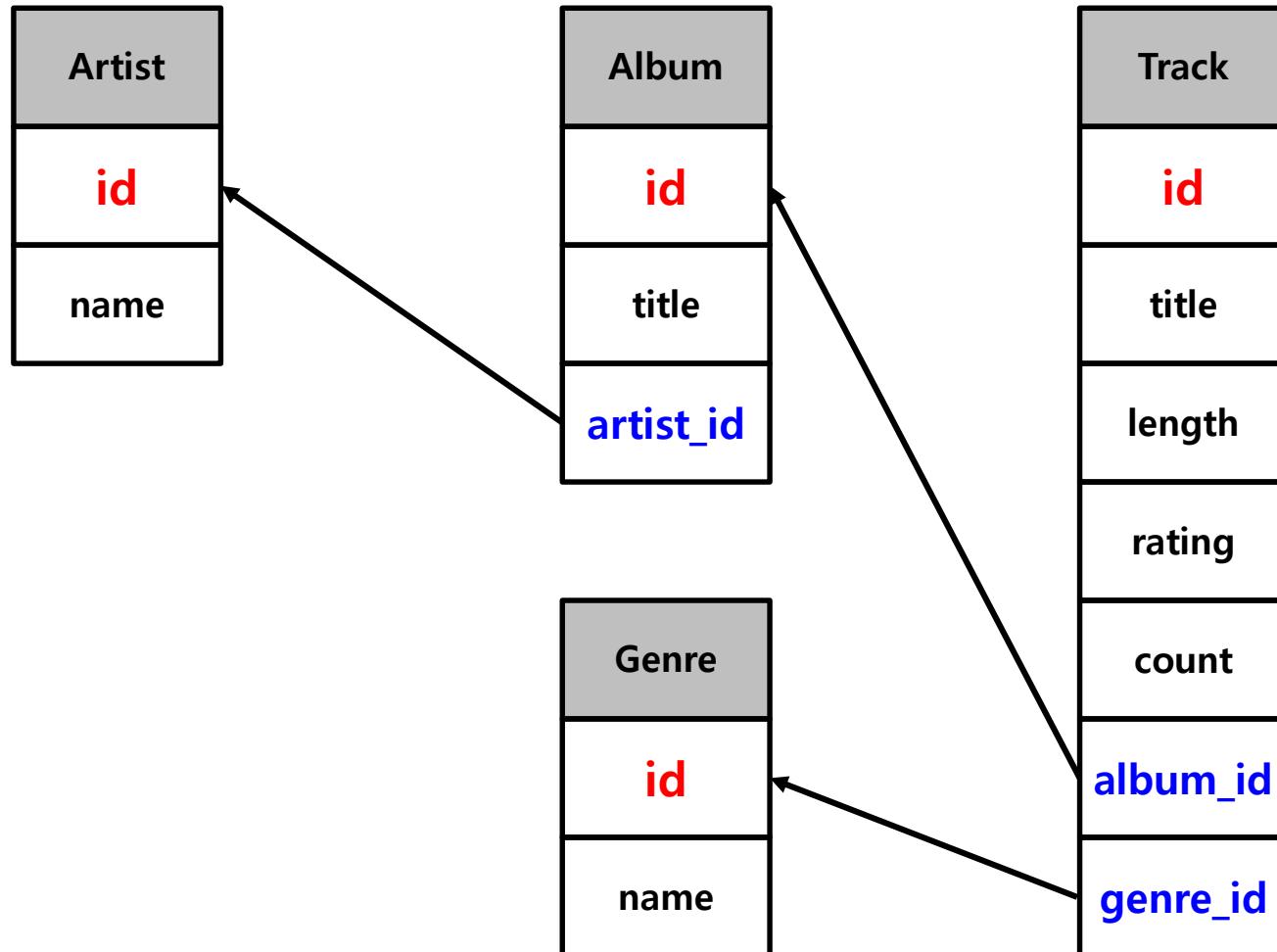
print(users.join(addresses, users.c.id == addresses.c.user_id))
```

- ON condition of the join, as it's called, was automatically generated based on the ForeignKey object

```
query = select([users.c.id, users.c.fullname, addresses.c.email_address]) \
    .select_from(users.join(addresses))

result = conn.execute(query).fetchall()
for row in result:
    print(row)
```

■ 예제



○ Create

```
artist = Table("Artist", metadata,
               Column("id", Integer, primary_key=True),
               Column("name", String, nullable=False),
               extend_existing=True)

album = Table("Album", metadata,
              Column("id", Integer, primary_key=True),
              Column("title", String, nullable=False),
              Column("artist_id", Integer, ForeignKey("Artist.id")),
              extend_existing=True)

genre = Table("Genre", metadata,
              Column("id", Integer, primary_key=True),
              Column("name", String, nullable=False),
              extend_existing=True)

track = Table("Track", metadata,
              Column("id", Integer, primary_key=True),
              Column("title", String, nullable=False),
              Column("length", Integer, nullable=False),
              Column("rating", Integer, nullable=False),
              Column("count", Integer, nullable=False),
              Column("album_id", Integer, ForeignKey("Album.id")),
              Column("genre_id", Integer, ForeignKey("Genre.id")),
              extend_existing=True)

metadata.create_all(engine)
```

○ SHOW TABLES

```
tables = metadata.tables
for table in tables:
    print(table)

for table in engine.table_names():
    print(table)
```

○ Insert

```
conn.execute(artist.insert(), [
    {"name": "Led Zeppelin"}, 
    {"name": "AC/DC"}])
])

conn.execute(album.insert(), [
    {"title": "IV", "artist_id": 1}, 
    {"title": "Who Made Who", "artist_id": 2}])
)

conn.execute(genre.insert(), [
    {"name": "Rock"}, 
    {"name": "Metal"}])
)

conn.execute(track.insert(), [
    {"title": "Black Dog", "rating": 5, "length": 297, "count": 0, "album_id": 1, "genre_id": 1}, 
    {"title": "Stairway", "rating": 5, "length": 482, "count": 0, "album_id": 1, "genre_id": 1}, 
    {"title": "About to rock", "rating": 5, "length": 313, "count": 0, "album_id": 2, "genre_id": 2}, 
    {"title": "Who Made Who", "rating": 5, "length": 297, "count": 0, "album_id": 2, "genre_id": 2}])
)
```

○ Select

```
artistResult = conn.execute(artist.select())
for row in artistResult:
    print(row)

albumResult = conn.execute(album.select())
for row in albumResult:
    print(row)

genreResult = conn.execute(genre.select())
for row in genreResult:
    print(row)

trackResult = conn.execute(track.select())
for row in trackResult:
    print(row)
```

○ Where

```
trackResult = conn.execute(select([track])
                           .where(
                               and_(track.c.album_id == 1, track.c.genre_id == 1)
                           )
                           )
for row in trackResult:
    print(row)
```

○ Update

```
from sqlalchemy import update

conn.execute(track.update().values(genre_id=2).where(track.c.id==2))
conn.execute(track.update().values(genre_id=1).where(track.c.id==3))
```

○ Where

```
trackResult = conn.execute(select([track])
                           .where(
                               and_(track.c.album_id == 1,
                                   or_(track.c.genre_id == 1,
                                       track.c.genre_id == 2,)))
                           )
for row in trackResult:
    print(row)
```

○ Join

```
print(track.join(album))

result = conn.execute(track
                      .select()
                      .select_from(track.join(album)))

for row in result.fetchall():
    print(row)

result = conn.execute(track
                      .select()
                      .select_from(track.join(album))
                      .where(album.c.id==1))

for row in result.fetchall():
    print(row)
```

○ Multiple Join

```
print(track.join(album))
print(track.join(album).join(genre))
print(track.join(album).join(artist))
print(track.join(album).join(genre).join(artist))

result = conn.execute(select([track.c.title, album.c.title, genre.c.name, artist.c.name])
                      .select_from(track.join(album).join(genre).join(artist)))

for row in result.fetchall():
    print(row)

result = conn.execute(track
                      .select()
                      .select_from(track.join(album).join(genre).join(artist))
                      .where(
                          and_(
                              genre.c.id==1,
                              artist.c.id==1,
                          )
                      )
                  )

for row in result.fetchall():
    print(row)
```

○ Open/Close

```
from sqlalchemy import create_engine, MetaData

engine = create_engine("sqlite:///alchemy_core.db", echo=True)
conn = engine.connect()

metadata = MetaData(bind=engine, reflect=True)
metadata.reflect(bind=engine)

for row in metadata.tables:
    print(row)
```

```
tables = metadata.tables
for table in tables:
    print(table)

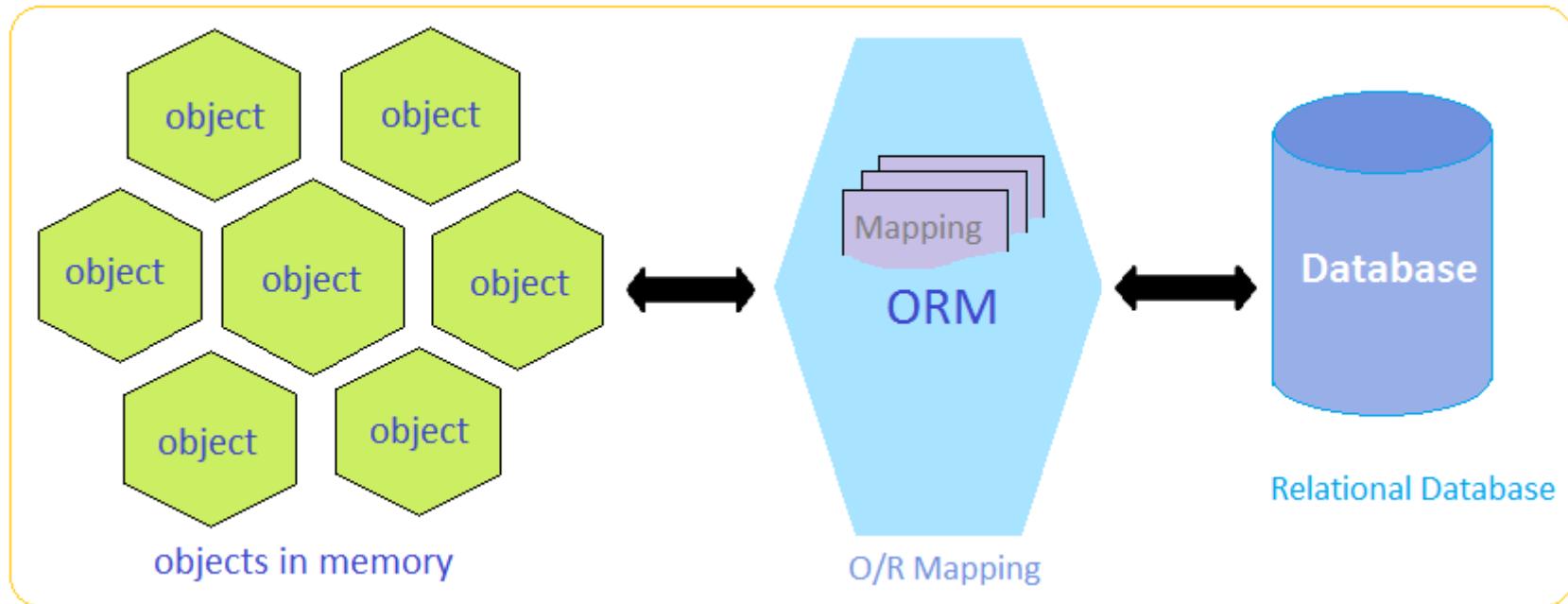
#album

track = metadata.tables["Track"]
track

for row in conn.execute(track.select()).fetchall():
    print(row)

conn.close()
metadata.clear()
```

■ SQLAlchemy ORM



Data Mapping to Classes

■ Declare

○ declarative_base

```
sqlalchemy.ext.declarative.declarative_base
```

- Construct a base class for declarative class definitions

○ 예제

```
from sqlalchemy import create_engine

engine = create_engine("sqlite:///memory:", echo=True)
```

```
from sqlalchemy.ext.declarative import declarative_base

base = declarative_base()
```

- declarative_base() callable returns a new base class from which all mapped classes should inherit

■ Create

- declarative_base 상속 class 선언

```
class User(base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True)
    name = Column(String)
    fullname = Column(String)
    password = Column("passwd", String)

    def __repr__(self):
        return "<T'User(name='%s', fullname='%s', password='%s')>" \
            % (self.name, self.fullname, self.password)
```

- a new Table and mapper() will have been generated
- table and mapper are accessible via __table__ and __mapper__ attributes

- Schema

```
User.__table__
```

- Create Table

```
base.metadata.create_all(engine)
```

- Create Instance

```
kim = User(name="kim", fullname="anonymous, Kim", password="kimbap heaven")

print(kim)
print(kim.id)
```

■ Session

○ Session

```
class sqlalchemy.orm.session.Session
```

- Session establishes all conversations with the database and represents all the objects
- Manages persistence operations for ORM-mapped objects

○ sessionmaker

```
class sqlalchemy.orm.session.sessionmaker
```

- sessionmaker factory generates new Session objects when called
- sessionmaker class is normally used to create a top level Session configuration

○ 예제

```
from sqlalchemy.orm import sessionmaker

Session = sessionmaker(bind=engine)
session = Session()
```

■ Insert

○ add

```
add(instance, _warn=True)
```

- Place an object in the Session, persisted to the database on the next flush operation.
- Repeated calls to add() will be ignored

○ addall

```
add_all(instaces)
```

- Add the given collection of instances to this Session

○ 예제

```
session.add(kim)

session.add_all([
    User(name="lee", fullname="unknown, Lee", password="123456789a"),
    User(name="park", fullname="nobody, Park", password="Parking in Park")
])
```

- Pending. no SQL has yet been issued and the object is not yet represented by a row in the database

■ Update

- dirty

dirty

- The set of all persistent instances considered dirty
- Instances are considered dirty when they were modified but not deleted

- is_modified

is_modified(instance, include_collections=True, passive=True)

- Return True if the given instance has locally modified attributes

- 예제

```
kim.password = "password"

session.dirty

session.is_modified(kim)
```

■ Commit

- commit

```
commit()
```

- Flush pending changes and commit the current transaction
- If no transaction is in progress, this method raises an InvalidRequestError

■ Select

- query

```
class sqlalchemy.orm.query.Query(entities, session=None)
```

- ORM-level SQL construction object
- Query is the source of all SELECT statements generated by the ORM

- 예제

```
for row in session.query(User):  
    print(type(row))  
    print(row.id, row.name, row.fullname, row.password)
```

○ filter

filter(*criterion)

- Apply the given filtering criterion to a copy of this Query, using SQL expressions
- Allow you to use regular Python operators with the class-level attributes on your mapped class

○ filter_by

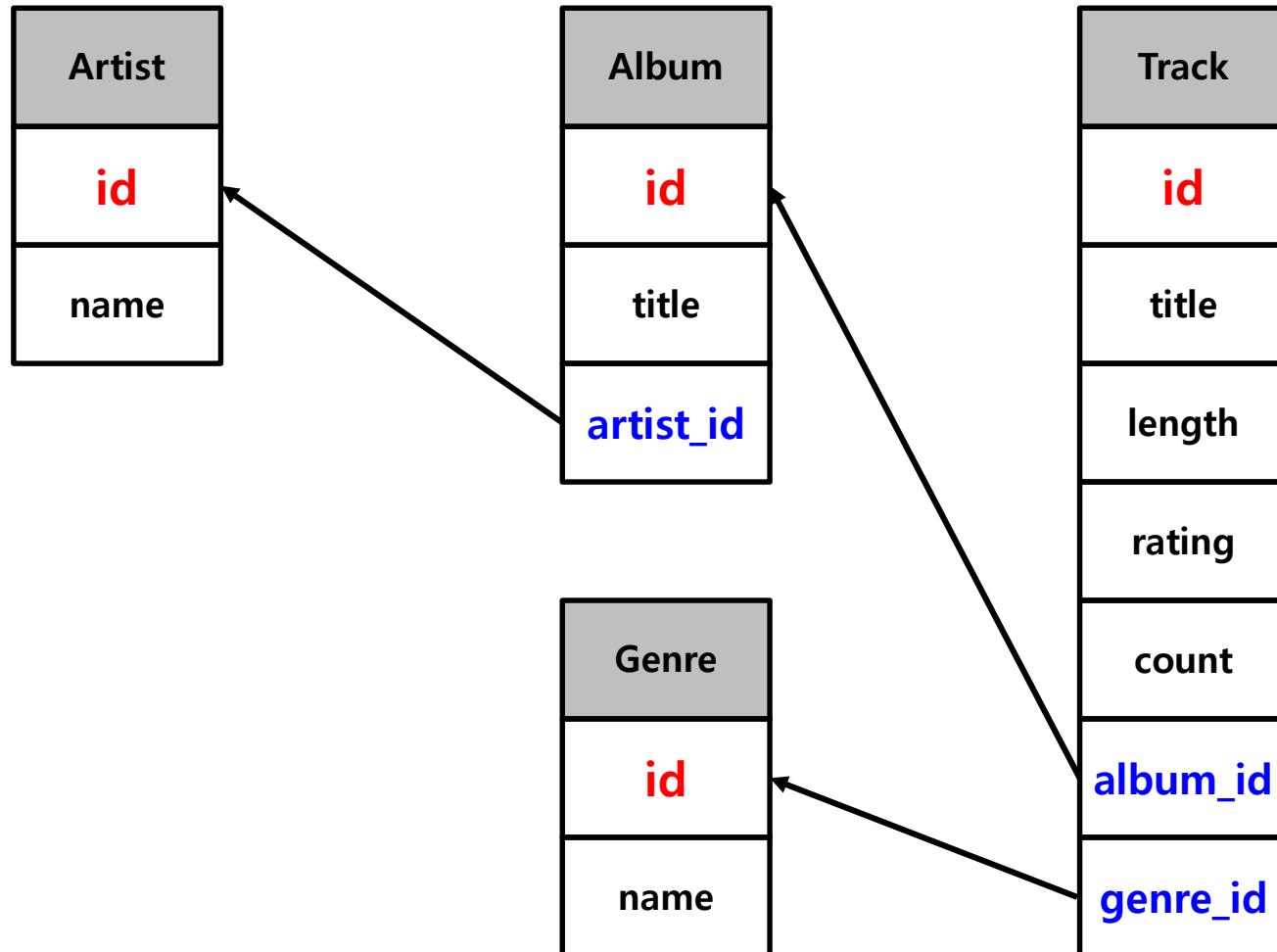
filter_by(kwargs)**

- apply the given filtering criterion to a copy of this Query, using keyword expressions

○ 예제

```
for row in session.query(User.id, User.fullname).filter(User.name == "lee"):  
    print(type(row))  
    print(row.id, row.fullname)  
  
for row in session.query(User.id, User.fullname).filter_by(name = "lee"):  
    print(type(row))  
    print(row.id, row.fullname)
```

■ 예제



○ Create

```
class Artist(Base):
    __tablename__ = "Artist"

    id = Column(Integer, primary_key=True)
    name = Column(String)

class Album(Base):
    __tablename__ = "Album"

    id = Column(Integer, primary_key=True)
    title = Column(String)
    artist_id = Column(Integer, ForeignKey("Artist.id"))

class Genre(Base):
    __tablename__ = "Genre"

    id = Column(Integer, primary_key=True)
    name = Column(String)

class Track(Base):
    __tablename__ = "Track"

    id = Column(Integer, primary_key=True)
    title = Column(String)
    length = Column(Integer)
    rating = Column(Integer)
    count = Column(Integer)
    album_id = Column(Integer, ForeignKey("Album.id"))
    genre_id = Column(Integer, ForeignKey("Genre.id"))
```

○ Insert

```
artist1 = Artist(name="Led Zeppelin")
artist2 = Artist(name="AC/DC")

session.add_all([artist1, artist2])
session.commit()

album = [Album(title="IV", artist_id=artist1.id),
         Album(title="Who Made Who", artist_id=artist2.id)]

session.add_all(album)
session.commit()

session.add_all([Genre(name="Rock"), Genre(name="Metal")])
session.commit()

album1 = session.query(Album).filter(Album.artist_id==artist1.id).one()
album2 = session.query(Album).filter(Album.artist_id==artist2.id).one()

genre1 = session.query(Genre).filter(Genre.name=="Rock").filter(Genre.id==1).one()
genre2 = session.query(Genre).filter(Genre.name=="Metal").filter(Genre.id==2).one()

track = [Track(title="Black Dog", rating=5, length=297, count=0, album_id=album1.id, genre_id=genre1.id),
         Track(title="Stairway", rating=5, length=482, count=0, album_id=album1.id, genre_id=genre2.id),
         Track(title="About to rock", rating=5, length=313, count=0, album_id=album2.id, genre_id=genre1.id),
         Track(title="Who Made Who", rating=5, length=297, count=0, album_id=album2.id, genre_id=genre2.id)]
session.add_all(track)
session.commit()
```

○ Join

```
result = session.query(Track.title, Album.title, Genre.name, Artist.name) \
    .select_from(Track) \
    .join(Album)\ 
    .join(Genre)\ 
    .join(Artist).all()

for row in result:
    print(row)

songList = [dict(Track=row[0], Album=row[1], Genre=row[2], Artist=row[3]) for row in result]

songList
```

■ Relationship

○ relationship

sqlalchemy.orm.relationship

- Provide a relationship between two mapped classes
- This corresponds to a parent-child or associative table relationship

○ back_populates / backref

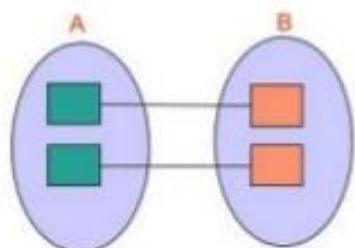
- indicates the string name of a property to be placed on the related mapper's class that will handle this relationship in the other direction

○ uselist

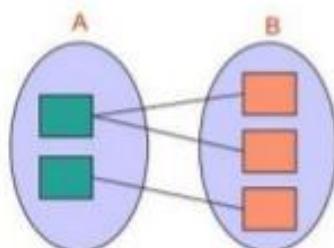
- a boolean that indicates if this property should be loaded as a list or a scalar

■ Relationship Model

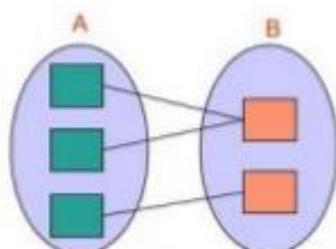
Types of Relationships



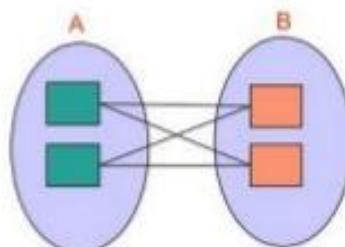
1:1 Relationship



1:N Relationship



N:1 Relationship



N:M Relationship

■ 예제

```
class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True)
    name = Column(String)
    fullname = Column(String)
    password = Column("passwd", String)

    #addresses = relationship("Address", back_populates="user")
    #addresses = relationship("Address", back_populates="user", uselist=False)

    def __repr__(self):
        return "<User(name='%s', fullname='%s', password='%s')>" % (self.name, self.fullname, self.password)
```

```
class Address(Base):
    __tablename__ = "addresses"

    id = Column(Integer, primary_key=True)
    email_address = Column(String, nullable=False)
    user_id = Column(Integer, ForeignKey("users.id"))

    #user = relationship("User", back_populates="addresses", uselist=False)
    #user = relationship("User", back_populates="addresses")
    #user = relationship("User")

    def __repr__(self):
        return "<Address(email_address='%s')>" % self.email_address
```

■ 예제

```
class Artist(Base):
    __tablename__ = "Artist"

    id = Column(Integer, primary_key=True)
    name = Column(String)

    albumList = relationship("Album", back_populates="artist")

class Album(Base):
    __tablename__ = "Album"

    id = Column(Integer, primary_key=True)
    title = Column(String)
    artist_id = Column(Integer, ForeignKey("Artist.id"))

    artist = relationship("Artist", back_populates="albumList", uselist=False)
    trackList = relationship("Track", back_populates="album")

class Genre(Base):
    __tablename__ = "Genre"

    id = Column(Integer, primary_key=True)
    name = Column(String)

    trackList = relationship("Track", back_populates="genre")

class Track(Base):
    __tablename__ = "Track"

    id = Column(Integer, primary_key=True)
    title = Column(String)
    length = Column(Integer)
    rating = Column(Integer)
    count = Column(Integer)
    album_id = Column(Integer, ForeignKey("Album.id"))
    genre_id = Column(Integer, ForeignKey("Genre.id"))

    album = relationship("Album", back_populates="trackList", uselist=False)
    genre = relationship("Genre", back_populates="trackList", uselist=False)
```

■ Insert

```
track1 = Track(title="Black Dog", rating=5, length=297, count=0)
track2 = Track(title="Stairway", rating=5, length=482, count=0)
track3 = Track(title="About to rock", rating=5, length=313, count=0)
track4 = Track(title="Who Made Who", rating=5, length=297, count=0)
```

```
track1.album = track2.album = Album(title="IV")
track3.album = track4.album = Album(title="Who Made Who")
```

```
track1.genre = track3.genre = Genre(name="Rock")
track2.genre = track4.genre = Genre(name="Metal")
```

```
track1.album.artist = track2.album.artist = Artist(name="Led Zepplin")
track3.album.artist = track4.album.artist = Artist(name="AC/DC")
```

■ Select

```
print("Title: %s, Album: %s, Genre: %s, Artist: %s" %
      (track1.title, track1.album.title, track1.genre.name, track1.album.artist.name))
print("Title: %s, Album: %s, Genre: %s, Artist: %s" %
      (track3.title, track3.album.title, track3.genre.name, track3.album.artist.name))
```

```
print("TrackID: %d, AlbumID: %d, GenreID: %d, Artist: %d" %
      (track1.id, track1.album.id, track1.genre.id, track1.album.artist.id))
```

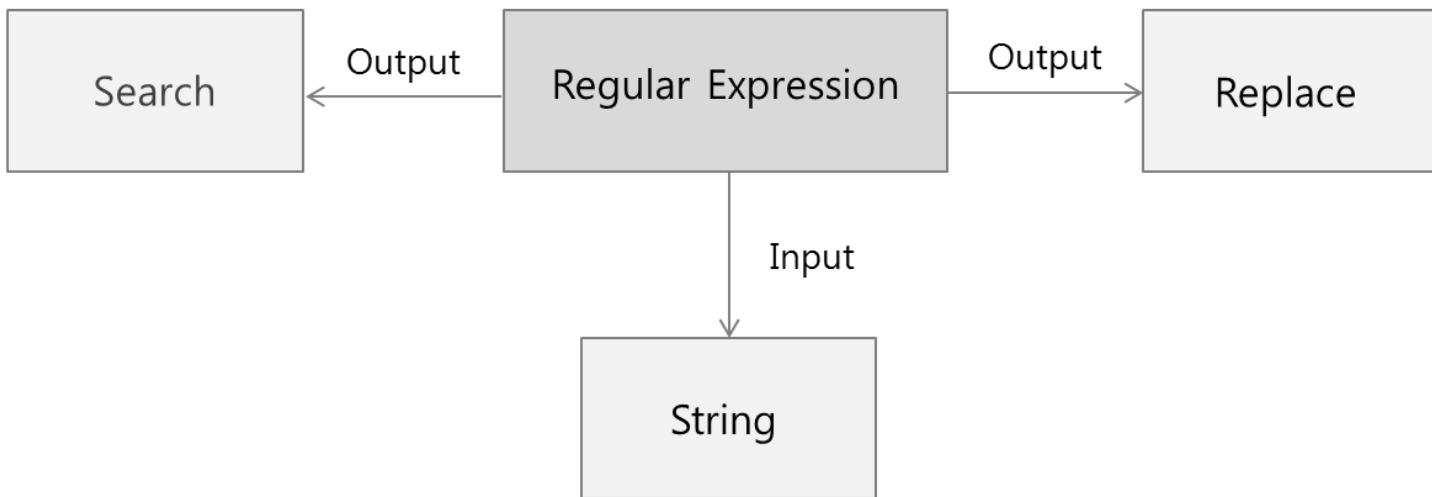
```
print("TrackID: %d, AlbumID: %d, GenreID: %d, Artist: %d" %
      (track3.id, track3.album.id, track3.genre.id, track3.album.artist.id))
```

2.3. Regular Expression

regular
expression

■ What is RE?

- a special text string for describing a search pattern
- searching, replacing, and parsing text with complex patterns of characters



■ Why use RE?

- Processes large amounts of text over and over again / Extremely fast
- Usually this pattern is then used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation

■ But, There is a learning curve

■ General concepts

REGEX | General Concepts

- ❑ Alternative: |
- ❑ Grouping: ()
- ❑ Quantification: ? + * {m,n}
- ❑ Anchors: ^ \$
- ❑ Meta-characters: . [] [-] [^]
- ❑ Character Classes: \w \d \s \W ...

■ Meta characters

○ . ^ \$ * + ? { } [] \wedge | ()

➤ .(dot)

- a.b
- a+모든문자+b
- aab, a0b, ~~a~~b~~c~~

➤ *(asterisk)

- ab*c
- a+b(0번 이상 반복)+c
- ac, abc, abbbbbbbc

➤ +(plus)

- ab+c
- a+b(1번 이상 반복)+c
- ~~a~~c, abc, abbbbbbbc

➤ {n | min, | min, max}(curly braces)

- {0,}=*, {1,}=+
- ab{1}c, ab{2,6}c
- a+b(1번 반복, 2번-6번)+c
- abc, abbc, abbbbbbbc

➤ ?(question)

- ab?c, b{0,1}
- a+b(1개 있거나, 없거나)+c
- ac, abc, ~~a~~bb~~b~~c

정규표현식	표현	설명
x		문자열이 x로 시작합니다.
$x\$$		문자열이 x로 끝납니다.
$.x$		임의의 한 문자를 표현합니다. (x가 마지막으로 끝납니다.)
x^+		x가 1번이상 반복합니다.
$x?$		x가 존재하거나 존재하지 않습니다.
x^*		x가 0번이상 반복합니다.
$x y$		x 또는 y를 찾습니다. (or연산자를 의미합니다.)
(x)		()안의 내용을 캡쳐하며, 그룹화 합니다.
$(x)(y)$		그룹화 할 때, 자동으로 앞에서부터 1번부터 그룹 번호를 부여해서 캡쳐합니다. 결과값에 그룹화한 Data가 배열 형식으로 그룹번호 순서대로 들어갑니다.
$(x)(?:y)$		캡쳐하지 않는 그룹을 생성할 경우 ?:를 사용합니다. 결과값 배열에 캡쳐하지 않는 그룹은 들어가지 않습니다.
x^n		x를 n번 반복한 문자를 찾습니다.
$x^{n,}$		x를 n번이상 반복한 문자를 찾습니다.
$x^{n,m}$		x를 n번이상 m번이하 반복한 문자를 찾습니다.

■ Character classes

- [], [-], [^], \d, \D, \s, \S, \w, \W

➤ [](square braket)

- [abc]
- a, b or c

➤ -

- [a-z]
- a부터 z까지

➤ ^

- [^a-z]
- a부터 z까지를 제외

■ Anchors

- ^, \$, \b, \B

➤ ^ - Starting position

➤ \$ - Ending position

■ Grouping

➤ ()(parenthesis)

- (abc)

정규표현식	표현	설명
[xy]		x,y중 하나를 찾습니다.
[^xy]		x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)
[x-z]		x~z 사이의 문자중 하나를 찾습니다.
\w^		^(특수문자)를 식에 문자 자체로 포함합니다. (escape)
\w\B		문자와 공백사이의 문자를 찾습니다.
\wB		문자와 공백사이가 아닌 값을 찾습니다.
\wd		숫자를 찾습니다.
\wD		숫자가 아닌 값을 찾습니다.
\ws		공백문자를 찾습니다.
\wS		공백이 아닌 문자를 찾습니다.
\wt		Tab 문자를 찾습니다.
\wv		Vertical Tab 문자를 찾습니다.
\ww		알파벳 + 숫자 + _ 를 찾습니다.
\wW		알파벳 + 숫자 + _ 을 제외한 모든 문자를 찾습니다.

■ Match method

○ Greedy vs Lazy(non greedy)

- Greedy – tries to find the last possible match
- Lazy – tries to find the first possible match

Greedy quantifier	Lazy quantifier	Description
*	*?	Match zero or more times.
+	+?	Match one or more times.
?	??	Match zero or one time.
{n}	{n}?	Match exactly n times.
{n,}	{n,}?	Match at least n times.
{n,m}	{n,m}?	Match from n to m times.

Add a ? to a quantifier to make it ungreedy i.e lazy.

Example:

test string : stackoverflow

greedy reg expression : s.*o output: **stackoverflow**

lazy reg expression : s.*?o output: **stackoverflow**

■ 예제

○ <https://www.regexr.com/>

○ <https://www.regexper.com/>

○ Alternative

- cat|mat → 'car' or 'mat'
- regular|expression → 'regular' or 'expression'

○ Grouping

- gr(e|a)y → 'grey' or 'gray'
- py(pi|tho(n|nic) → 'pypi' or 'python' or 'pythonic'

○ Quantification

- colou?r → 'color' or 'colour'
- 81*5 → '85' or '815' or '8111115'
- 81+5 → '815' or '8111115'
- go{2,3}gle → 'google' or 'gooogle'
- go{2,} → 'goo' or 'ooooooooooooooo'

○ Anchors – match the starting or ending position

- ^obje → 'object' or 'object-oriented'
- ^2018 → '2018' or '2018-07-10'
- gram\$ → 'program' or 'kilogram'

○ Meta characters – match a single character

- bat. → 'bat' or 'bats' or 'bata'
- [xyz] → 'x' or 'y' or 'z'
- [aeiou] → any vowel
- [0123456789] → any digit
- [a-c] → 'a' or 'b' or 'c'
- [a-zA-Z] → all letters(uppercase, lowercase)
- [0-9] → all digits
- [^aeiou] → any non-vowel
- [^0-9] → any non digit
- [^xyz] → any character, but not 'x' or 'y' or 'z'

○ Character classes

- \d – digits / \D – non digits
- \s – a single white space character / \S – non white space
- \w – alphanumeric character [a-zA-Z0-9_] / \W – non alphanumeric
- \b – word boundaries / \B – non word boundaries

○ 'Hello World'

- (H . .) . (o . .)
 - 'Hell', 'o W'
- l+
 - He'll'o, Wor'l'd
- H.?e
 - 'He'
- l.+?o
 - 'llo'
- el*o
 - 'ello'
- l{1,2}
 - He'l'lo, He'll'o, Wor'l'd
- [aeiou]+
 - 'e', 'o', 'o'
- (Hello|Hi|Pogo)
 - 'Hello'
- llo\b
 - 'llo'
- \w
 - 'H', 'e', 'l', 'l', 'o', 'W', 'o', 'r', 'l', 'd'
- \W
 - ''

○ 'In Hello World'

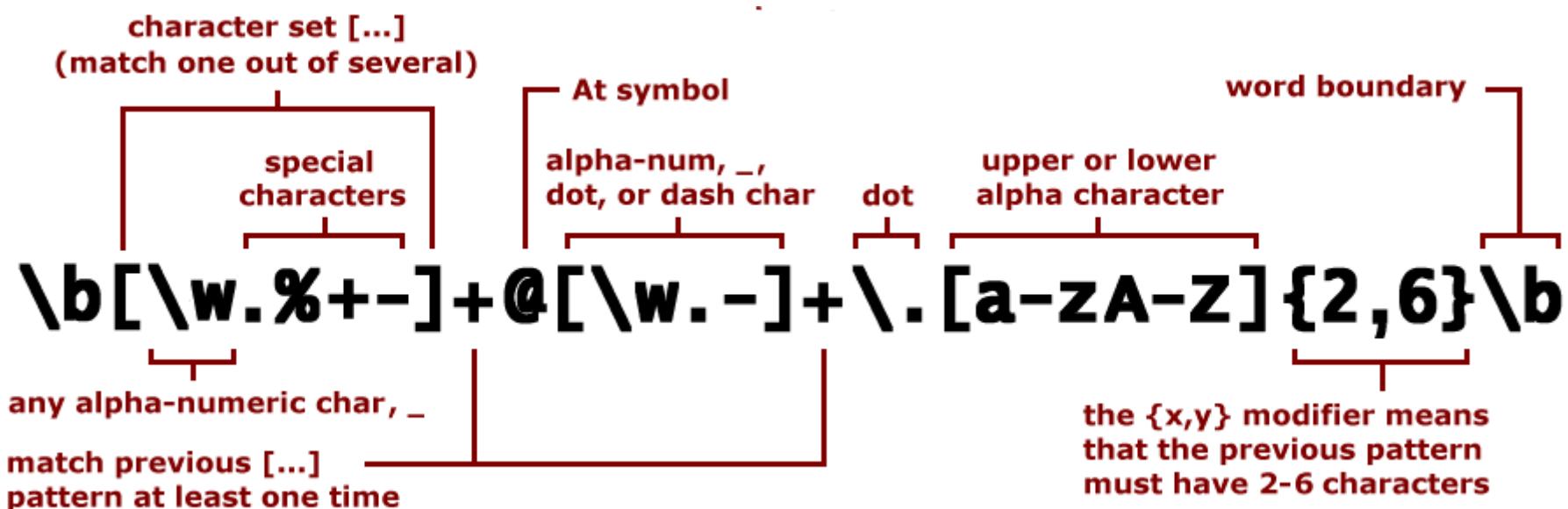
- \s.*\s
 - ' Hello '

○ 'Hello World'

- \S.*\S
 - 'Hello World'
- ^He
 - 'He'
- rld\$
- el*o
 - 'ello'
- [^Hlo]
 - 'e', ' ', 'W', r', 'd'

○ 예제

➤ username@domain.com



○ 예제

```
<h1 style="color:red">Heading</h1>
```

```
<([A-Z][A-Z0-9]*)[^>]*>(.*)<\/\1>
```

+00-00-0000-0000

+00 00 0000 0000

000-0000-0000

000 0000 0000

```
([\+]?[0-9]{2}[\s-]?)?0?1[0-9]{1}[\s-]?[0-9]{4}[\s-]?[0-9]{4}
```

○ re

```
import re
```

re.compile(pattern, flags=0)

Compile a regular expression pattern into a [regular expression object](#), which can be used for matching using its [match\(\)](#), [search\(\)](#) and other methods, described below.

re.search(pattern, string, flags=0)

Scan through *string* looking for the first location where the regular expression *pattern* produces a match, and return a corresponding [match object](#). Return `None` if no position in the string matches the pattern; note that this is different from finding a zero-length match at some point in the string.

re.match(pattern, string, flags=0)

If zero or more characters at the beginning of *string* match the regular expression *pattern*, return a corresponding [match object](#). Return `None` if the string does not match the pattern; note that this is different from a zero-length match.

Note that even in [MULTILINE](#) mode, `re.match()` will only match at the beginning of the string and not at the beginning of each line.

re.split(pattern, string, maxsplit=0, flags=0) ↴

Split *string* by the occurrences of *pattern*. If capturing parentheses are used in *pattern*, then the text of all groups in the pattern are also returned as part of the resulting list. If *maxsplit* is nonzero, at most *maxsplit* splits occur, and the remainder of the string is returned as the final element of the list.

○ search vs match

```
content = "Hello World"

print(re.search("W", content))
print(re.match("W", content))
```

- search
 - find something **anywhere** in the string and return a match object
- match
 - find something at the **beginning** of the string and return a match object

Method	목적
match()	문자열의 처음부터 정규식과 매치되는지 조사한다.
search()	문자열 전체를 검색하여 정규식과 매치되는지 조사한다.
findall()	정규식과 매치되는 모든 문자열(substring)을 리스트로 리턴한다
finditer()	정규식과 매치되는 모든 문자열(substring)을 iterator 객체로 리턴한다

○ sub

```
re.sub(pattern, repl, string, count=0, flags=0)
```

Return the string obtained by replacing the leftmost non-overlapping occurrences of *pattern* in *string* by the replacement *repl*. If the pattern isn't found, *string* is returned unchanged.

```
data = """
park 800905-1049118
kim  700905-1059119
"""

result = []
for line in data.split("\n"):
    word_result = []
    for word in line.split(" "):
        if len(word) == 14 and word[:6].isdigit() and word[7:].isdigit():
            word = word[:6] + "-" + "*****"
        word_result.append(word)
    result.append(" ".join(word_result))
print("\n".join(result))
```

```
data = """
park 800905-1049118
kim  700905-1059119
"""

pat = re.compile("(\\d{6})[-]\\d{7}")
print(pat.sub("\g<1>-*****", data))
```

○ Meta characters

➤ |

```
p = re.compile('Crow|Servo')
m = p.match('CrowHello')
print(m)
```

➤ ^

```
print(re.search('^Life', 'Life is too short'))
print(re.search('^Life', 'My Life'))
```

➤ \$

```
print(re.search('short$', 'Life is too short'))
print(re.search('short$', 'Life is too short, you need python'))
```

➤ +

```
p = re.compile('(ABC)+')
m = p.search('ABCABCABC OK?')
print(m.group())
```

○ Meta characters

➤ \b

```
p = re.compile(r'\bclass\b')
print(p.search('no class at all'))
print(p.search('one subclass is'))
print(p.search('the declassified algorithm'))
```

➤ \B

```
p = re.compile(r'\Bclass\B')
print(p.search('no class at all'))
print(p.search('one subclass is'))
print(p.search('the declassified algorithm'))
```

➤ group

```
p = re.compile(r"\w+\s+\d+[-]\d+[-]\d+")
m = p.search("park 010-1234-1234")
print(m)

p = re.compile(r"(\w+)\s+\d+[-]\d+[-]\d+")
m = p.search("park 010-1234-1234")
print(m.group(1))
```

○ group

Match.group([group1, ...])

Returns one or more subgroups of the match. If there is a single argument, the result is a single string; if there are multiple arguments, the result is a tuple with one item per argument. Without arguments, *group1* defaults to zero (the whole match is returned). If a *groupN* argument is zero, the corresponding return value is the entire matching string; if it is in the inclusive range [1..99], it is the string matching the corresponding parenthesized group. If a group number is negative or larger than the number of groups defined in the pattern, an `IndexError` exception is raised. If a group is contained in a part of the pattern that did not match, the corresponding result is `None`. If a group is contained in a part of the pattern that matched multiple times, the last match is returned.

method	목적
group()	매치된 문자열을 리턴한다.
start()	매치된 문자열의 시작 위치를 리턴한다.
end()	매치된 문자열의 끝 위치를 리턴한다.
span()	매치된 문자열의 (시작, 끝)에 해당되는 튜플을 리턴한다.

➤ 예제

```
m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")
print(m.group(0))
print(m.group(1))
print(m.group(2))
print(m.group(1, 2))
```

```
p = re.compile(r"(\w+) (\w+)")
m = p.search("Isaac Newton, physicist")
print(m.group())

p.sub("\g<2> \g<1>", "Isaac Newton, physicist")
```

■ 문제

다음 중 정규식 `a[.]{3,}b` 과 매치되는 문자열은 무엇일까?

1. acccb
2. a....b
3. aaab
4. a.cccb

```
>>> import re
>>> p = re.compile("[a-z]+")
>>> m = p.search("5 python")
>>> m.start() + m.end()
```

다음과 같은 문자열에서 핸드폰 번호 뒷자리인 숫자 4개를 #####로 바꾸는 프로그램을 정규식을 이용하여 작성해 보자.

```
"""
park 010-9999-9988
kim 010-9909-7789
lee 010-8789-7768
"""
```

다음은 이메일 주소를 나타내는 정규식이다. 이 정규식은 park@naver.com, kim@daum.net, lee@myhome.co.kr 등과 매치된다. 긍정형 전방 탐색 기법을 이용하여 .com, .net이 아닌 이메일 주소는 제외시키는 정규식을 작성해 보자.

```
.*[@].*[.].*$
```

4. HTML, CSS

1.1. HTML5 소개

1.2. HTML

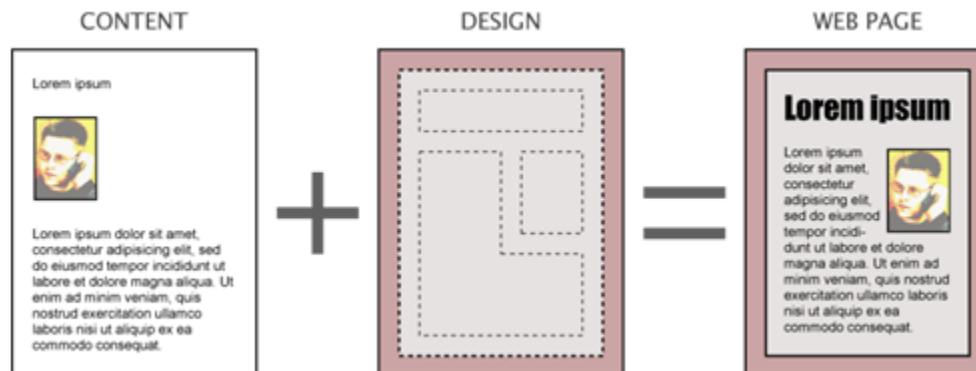
1.3. CSS

4.1. HTML 소개



■ Separation of Content and Presentation(Style)

- design principle as applied to the authoring and presentation of content

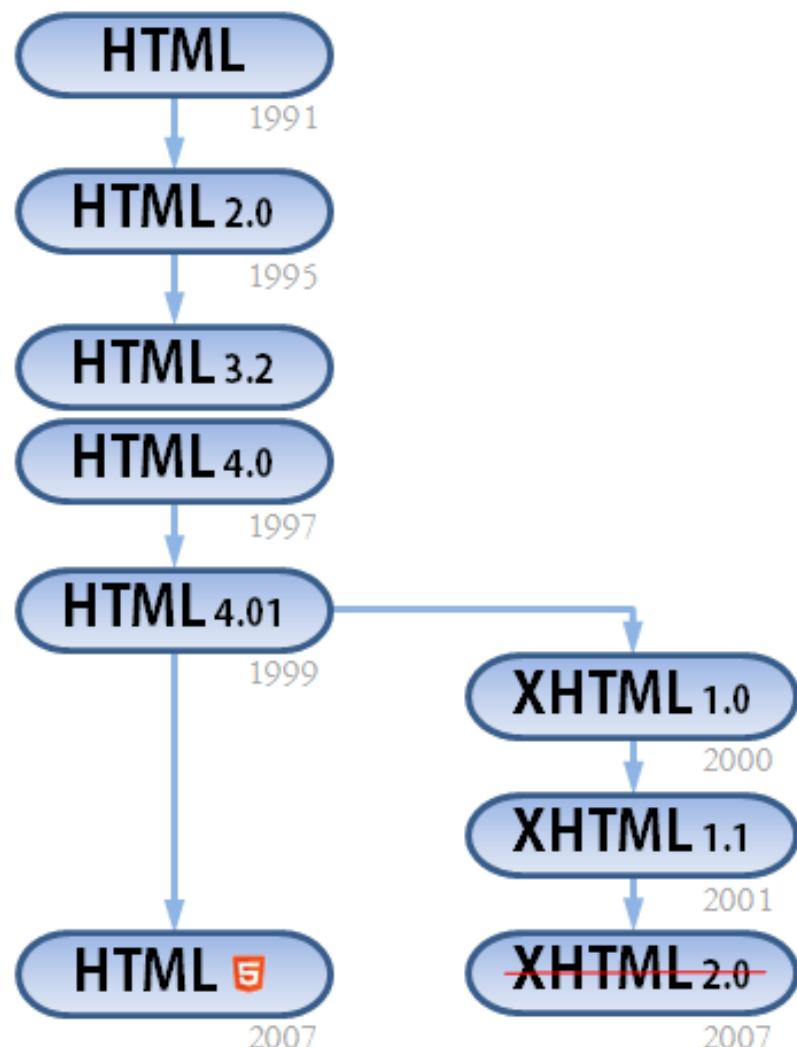




■ What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

■ HTML History



■ Human / Machine Readable Format – Semantic Web

Semantic Web



Group of methods and technologies to allow
machines to understand the meaning - or
"semantics" - of information on the World Wide Web.
(wikipedia)

RDF - Resource Description Framework

■ What are Semantics?

- 예를 들어,

< i > Some Text < / i >

Italic : 이탤릭체

< em > Some Text < / em >

Emphasis : 강조

- 스타일은 CSS가 담당하도록 함

■ From the Web of Document to the Web of Data

'Web of Documents'

- * Collections of 'simple' human readable documents
- * In HTML4, we had <link>

'Web of Data'

- * Collections of machine readable / parsable data
- * Building support
 - * Document hierarchies
 - * ex. Microdata

○ Microdata

```
1 <div itemscope itemtype="http://schema.org/Movie">
2   <h1 itemprop="name">Avatar</h1>
3   <span>Director: <span itemprop="director">James Cameron</span> (born August 16, 1954)</span>
4   <span itemprop="genre">Science fiction</span>
5   <a href="https://youtu.be/0AY1XIkJ7bY" itemprop="trailer">Trailer</a>
6 </div>
```

itemscope	Itemtype	Movie	
	itemprop	(itemprop name)	(itemprop value)
	itemprop	director	James Cameron
	itemprop	genre	Science Fiction
	itemprop	name	Avatar
	itemprop	https://youtu.be/0AY1XIkJ7bY	Trailer

adding **meanings of data** into HTML document

■ Semantic Markup

- HTML, CSS, JS의 분리 (= 웹표준화)

- HTML : '구조'와 '내용'을 담고 있음
- CSS : '표현'을 담고 있음
- JS : '동작'을 담고 있음

- '표현'을 위한 속성을 사용하지 않음

```
<table cellpadding="0" width="100%">
```

- Inline Style을 사용하지 않음

```
<li style="display: none; ">
```

- Inline Javascript를 사용하지 않음

```
<div onclick="fnClose();">그만 보기</div>
```

■ Semantic Markup

- 접근성이 좋아짐
- SEO(Search Engine Optimization)
- 수정이 용이해짐
- 코드 가독성이 좋아짐
- 코드와 데이터의 재사용성이 높아짐

■ Web as an Application Platform

At the era of HTML4, we had already 'DHTML', aka.
Dynamic HTML.

First, we used them as 'interacting web'.
And we have found the 'AJAX'

In HTML5, many new APIs are included.

- * Graphics – SVG, Canvas
- * Acceleration – Image/Video hardware acceleration
- * AV – Audio / Video elements had added

■ HTML5 technologies

- * Divided HTML5 -- Making small and related formats
- * CSS as 'General Formatter',
including Web as well as Paper Publishing
- * HTML5 as 'Web of Data'
Microdata and RDFa
- * HTML5 as 'Collection of WebAPI'
web storage, web sockets etc.
- * HTML5 as 'Client Graphics'
Canvas, SVG etc.

■ Client Graphics

Canvas is included in HTML5, and also SVG is avail.

- * Canvas
 - * Script based Graphics
 - * Image will be handled as 'pixel data array'
- * SVG – Scalable Vector Graphics
 - * XML based Graphics
 - * Image will be handled as 'DOM' object

■ HTML5 summary

- * New elements
 - ** Semantics -- section, article, aside, hgroup, etc.
 - ** Multimedia -- video, audio, embed, canvas
 - ** Widgets -- progress, meter, command, details, etc.
- * Deleted elements
 - ** CSS / font related.
 - ** frame, frameset related.
- * Simple doctype

■ HTML5 – New Elements

- 추가된 태그들

article, aside, audio, bdi, canvas, command,
datalist, details, embed, figcaption, figure,
footer, header, hgroup, keygen, mark, meter,
nav, output, progress, rp, rt, ruby, section,
source, summary, time, track, video, wbr

■ HTML5 – Semantics

Many semantics related elements have added.
And also, you can use Microdata/RDFa for semantics.

- * section – general sectioning/grouping for contents
 - For h* with its continuing contents
 - * article – section for an independent contents
 - * aside – 'aside' contents, such as references
 - * hgroup – grouping for h*
-
- * header – section header (not 'head')
 - * footer – section footer
 - * nav – site navigation

■ Creating better Markup – Heading

- **Heading을 사용한다**

<h1>, <h2>, <h3>, <h4>, <h5>, <h6> 사용

1. HTML5 Doctor
 1. *Featured Article*
 1. Pushing and Popping with the History API
 2. *Untitled Section*
 1. Recent Comments
 2. HTML5 Element Index
 1. Head
 2. Sections
 3. Grouping
 4. Tables
 5. Forms
 6. Forms 2
 7. Interactive
 8. Edits
 9. Embedded
 10. Text-level
 11. Text-level 2
 3. Sponsors
 4. Ask the HTML5 Doctor
 5. What are you up to?
 1. *Untitled Section*
 6. HTML5 Gallery

사이트 구축 기술문서

사이트 기획

왜 Drupal 인가?

개발 이슈

인증 통합

개발문서/메뉴얼 형식

메뉴 네비게이션

OG 그룹 시스템

인터페이스 개선

이용자 보상 제도

개발 내용

관리자 매뉴얼

■ Creating better Markup – List

- List 태그를 사용한다

내용에 따라 , , <dl>을 사용

ol : Ordered List - 순서가 있는 목록

ul : Unordered List - 순서가 없는 목록

dl : Definition List - 정의 목록

■ Creating better Markup – Label

- Form에 Label을 사용한다

```
<form>
    <label for="name">이름</label>
    <input type="text" name="name" id="name">
    <label for="gender">성별</label>
    <select name="gender" id="gender">
        <option value="male">남성</option>
        <option value="female">여성</option>
    </select>
</form>
```

■ Creating better Markup – 표현 태그 지양

- 표현을 위한 태그를 사용하지 않는다

	굵은 글씨	<tt>	타이프체
<i>	이탤릭체	<u>	밑줄
<big>	큰 글씨	<center>	중앙 정렬
<small>	작은 글씨	<nobr>	줄 바꿈 제한
<blink>	깜빡임		글씨 모양
<strike>	가로줄	<marquee>	흐르는 글씨
<s>	가로줄		

■ Creating better Markup – 표현 태그 지양/의미 태그 지향

- 표현을 위한 태그를 사용하지 않는다

	굵은 글씨
<i>	이탤릭체
<big>	큰 글씨
<small>	작은 글씨
<blink>	깜빡임
<strike>	가로줄
<s>	가로줄

<tt>	타이프체
<u>	밑줄
<center>	중앙 정렬
<nobr>	줄 바꿈 제한
	글씨 모양
<marquee>	흐르는 글씨

<p>	문단
	강조
	강한 강조
<q>	짧은 인용
<cite>	작품 제목
<abbr>	약자 표기
<dfn>	의미 정의
<code>	개발 코드

<samp>	시스템 출력
<kbd>	키보드 입력
<var>	변수
<ins>	추가된 내용
	삭제된 내용
<address>	연락처
<blockquote>	인용문

- 의미를 담은 태그를 가능한 지켜 사용한다

■ HTML5 – Multimedia

Graphical rendering are suggested for browsers.

- * progress – progress bar etc.
- * meter – results of measuring
- * menu / button / command – menu list
- * details / summary – detailed informations by user req.
- * keygen – PKI key generation

■ HTML5 – Delete

Elements only for display (w/o semantics) are deleted.
All of these elements could be replaced by 'CSS'.

such as 'font'

could be replaced with <div style="font-family: *">

Frame feature had deleted.

- * Adding scroll bar for some region – use CSS
- * Including common contents – use SSI or iframe

■ HTML5 – Delete

- 제거된 태그들

<big>	큰 글씨	<acronym>	두문자어
<center>	중앙 정렬	<applet>	애플릿
<dir>	파일 목록	<basefont>	기본 서체
	글씨 모양	<frame>	프레임
<tt>	타이프체	<frameset>	프레임 셋
<u>	밑줄	<noframes>	프레임미지원
<xmp>	형식화된	<strike>	가로줄

■ HTML5 – Simple Doctype

Doctype is simplified – <!DOCTYPE html>

- * Aren't we need to specify version of html?
 - most of current contents will be valid as HTML5
 - versioning of HTML might not be important

- HTML5요소의 브라우저 지원
 - 대부분의 브라우저 지원 가능
- : 모든 브라우저 호환을 고려하여 지정된 DTD

<!DOCTYPE html>

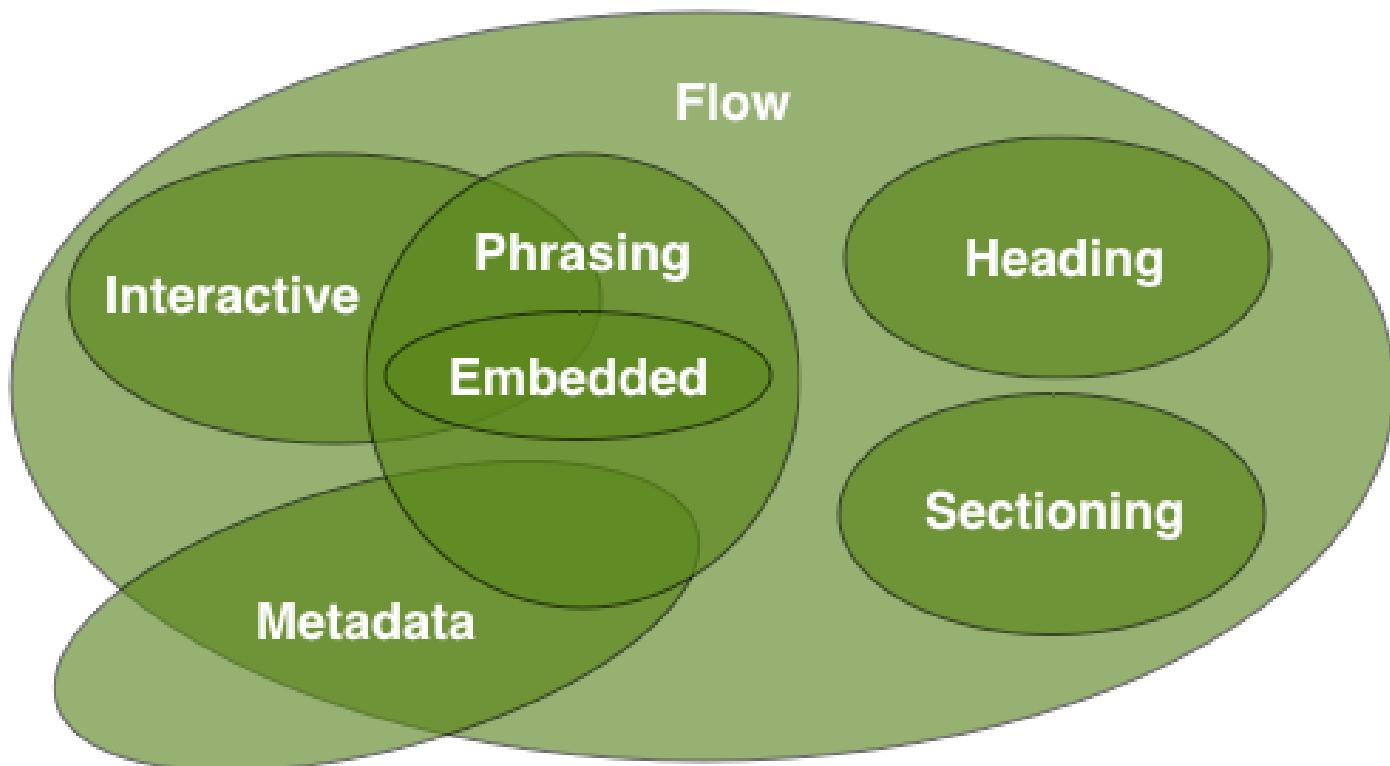
http://en.wikipedia.org/wiki/Quirks_mode *참조

: Charset, script, style의 경우도 마찬가지

<meta charset="utf-8">
<script> ... </script>

■ HTML5 Content Model

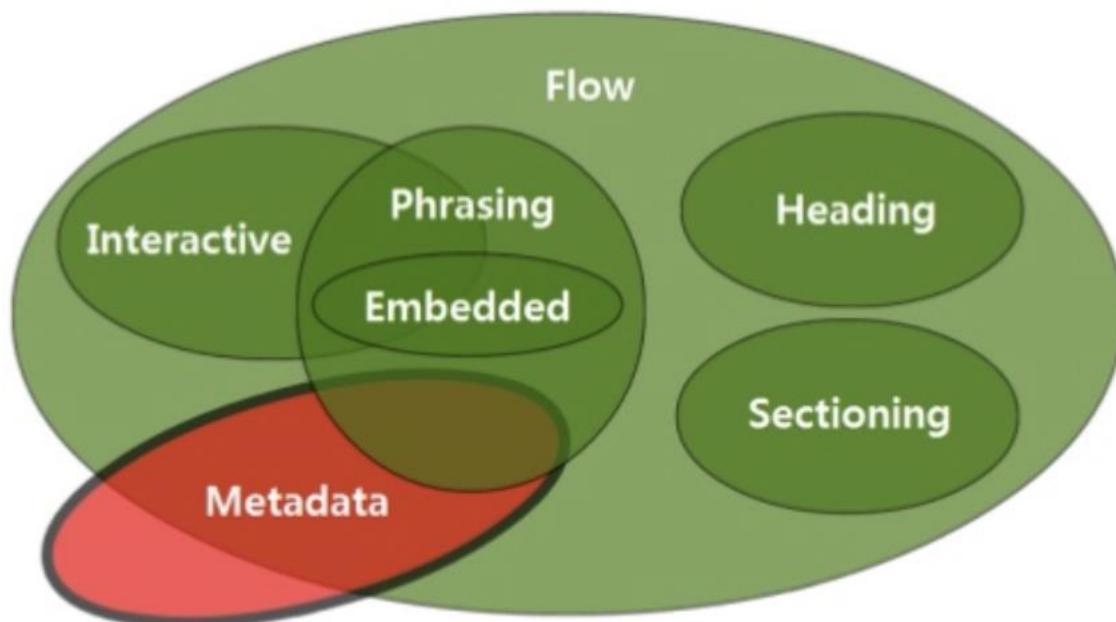
- HTML5에서는 모든 태그들을 용도에 따라 분류하여 컨텐츠 모델을 생성



○ Metadata

- **Metadata : 메타데이터**

CSS설정, script 설정, 문서간의 관계 설정 등의 정보

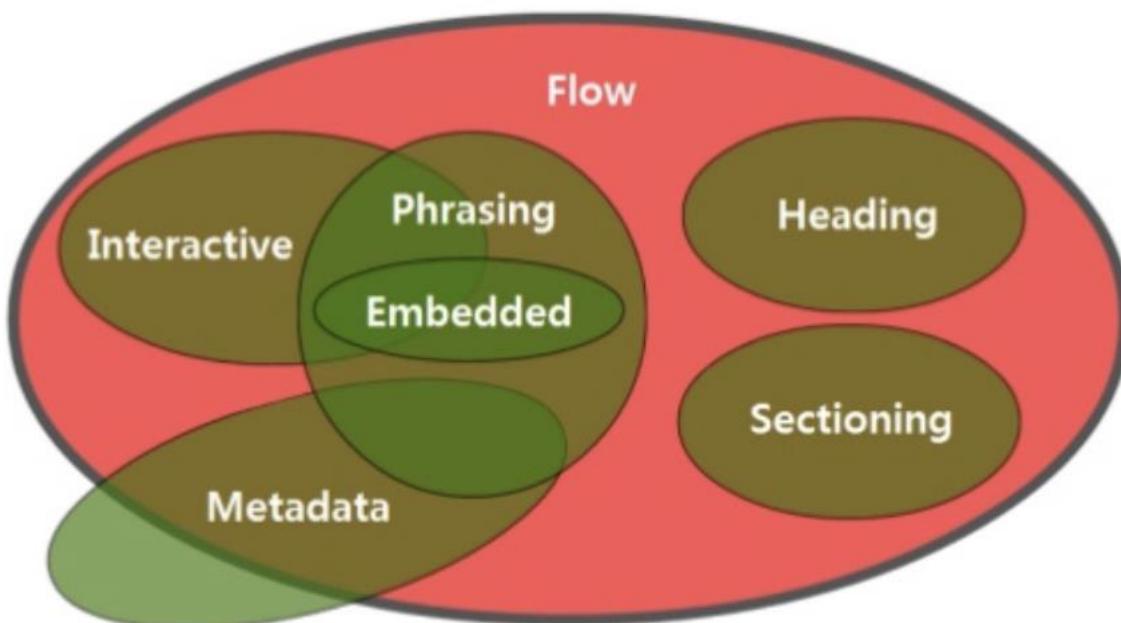


**base,
command,
link, meta,
noscript,
script,
style, title**

○ Flow

- **Flow : 플로우**

body내의 요소 대부분은 플로우 컨텐츠로 분류

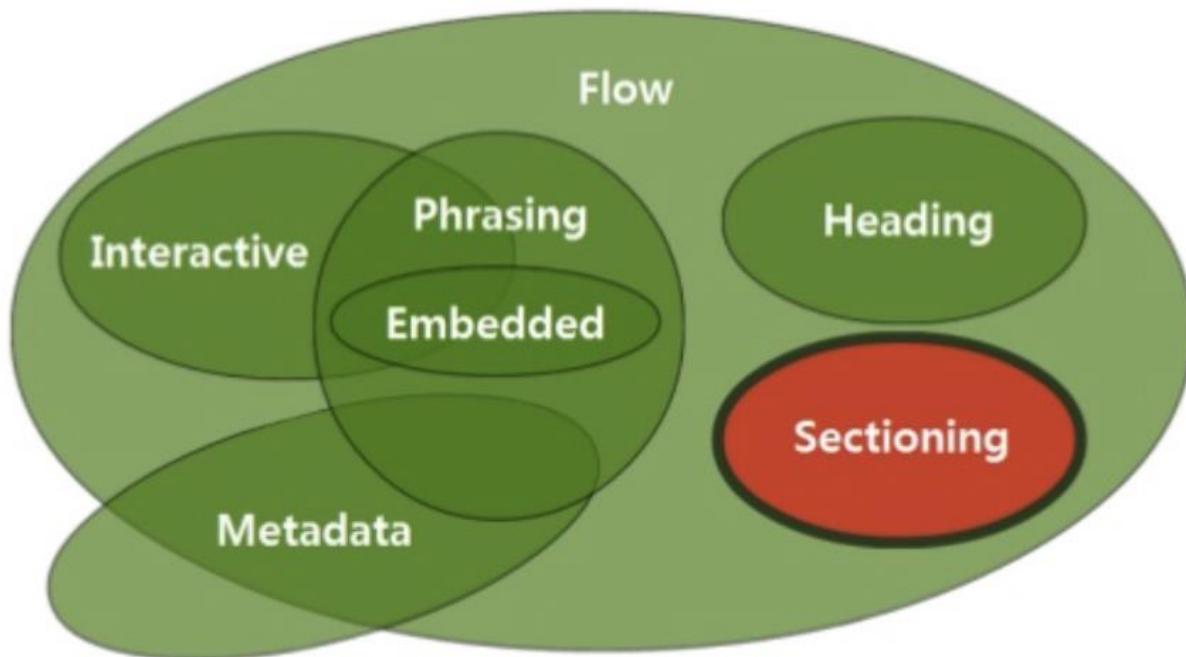


a, abbr, address, area, article, aside, audio, b, bdi, bdo, blockquote, br, button, canvas, cite, code, command, datalist, del, details, dfn, div, dl, em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, keygen, label, link, map, mark, math, menu, meta, meter, nav, noscript, object, ol, output, p, pre, progress, q, ruby, s, samp, script, section, select, small, span, strong, style, sub, sup, svg, table, textarea, time, u, ul, var, video, wbr

○ Section

- **Sectioning : 구역**

문서의 내용을 분류하는 구역을 생성



**article,
aside,
nav,
section**

○ Section

- Outline

Section은 문서의 Outline을 생성한다



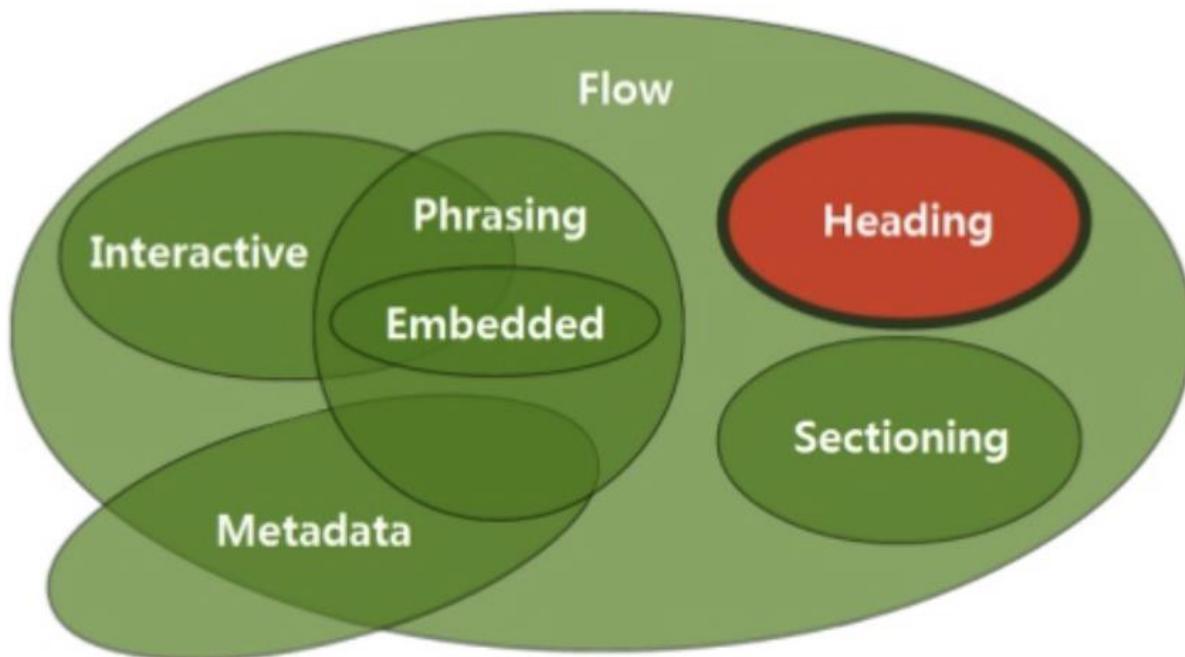
1. HTML5 Doctor
 1. *Featured Article*
 1. Pushing and Popping with the History API
 2. *Untitled Section*
 1. Recent Comments
 2. HTML5 Element Index
 1. Head
 2. Sections
 3. Grouping
 4. Tables
 5. Forms
 6. Forms 2
 7. Interactive
 8. Edits
 9. Embedded
 10. Text-level
 11. Text-level 2
 3. Sponsors
 4. Ask the HTML5 Doctor
 5. What are you up to?
 1. *Untitled Section*
 6. HTML5 Gallery

○ Heading

- **Heading : 머리말**

각 섹션의 머리말을 정의함.

섹션 요소가 없는 경우 Heading이 섹션을 만들게 됨



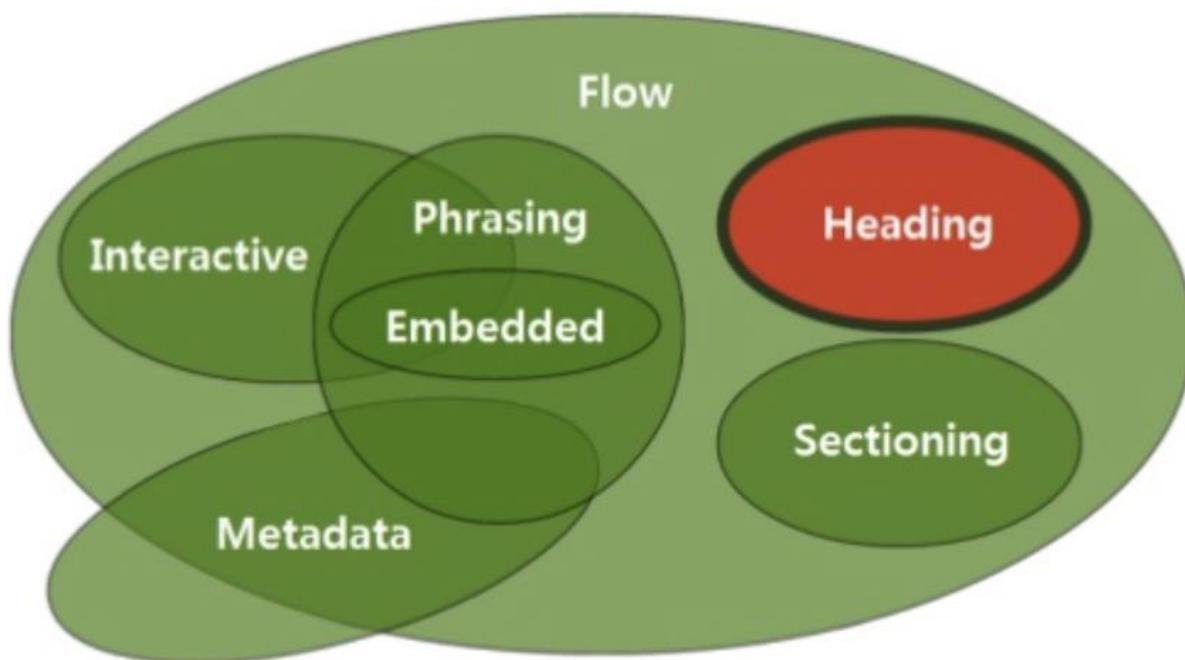
**h1, h2, h3,
h4, h5, h6,
hgroup**

○ Heading

- **Heading : 머리말**

각 섹션의 머리말을 정의함.

섹션 요소가 없는 경우 Heading이 섹션을 만들게 됨



**h1, h2, h3,
h4, h5, h6,
hgroup**

Heading

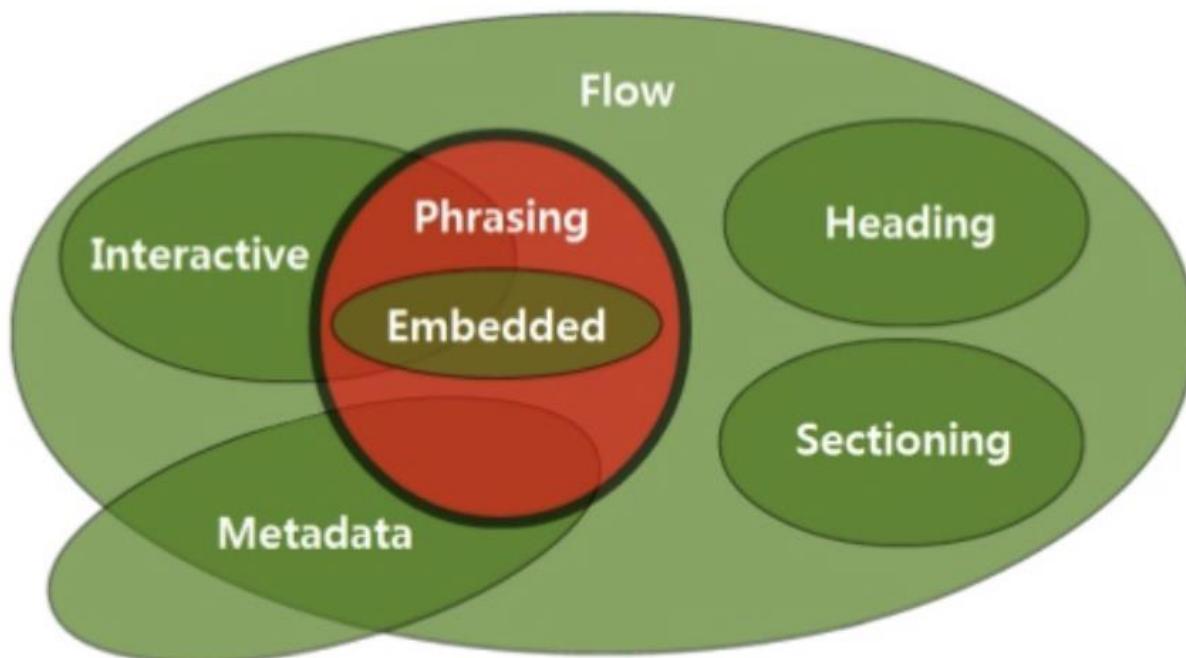
HTML5에서는 각 섹션별로 별도의 h1을 가질 수 있음

```
<h1>멋진 사이트</h1>
<section>
  <h1>프로필</h1>
  <p>저는 멋지게 살고 있습니다.</p>
  <section>
    <h1>직업</h1>
    <p>시골에서 농사짓고 있습니다.</p>
  </section>
</section>
<section>
  <h1>연락처</h1>
  <p>제 이름을 소리치면 제가 달려올 겁니다.</p>
</section>
```

○ Phrasing

• **Phrasing : 구문**

문단의 구성요소 (구문이 모여 문단이 됨)

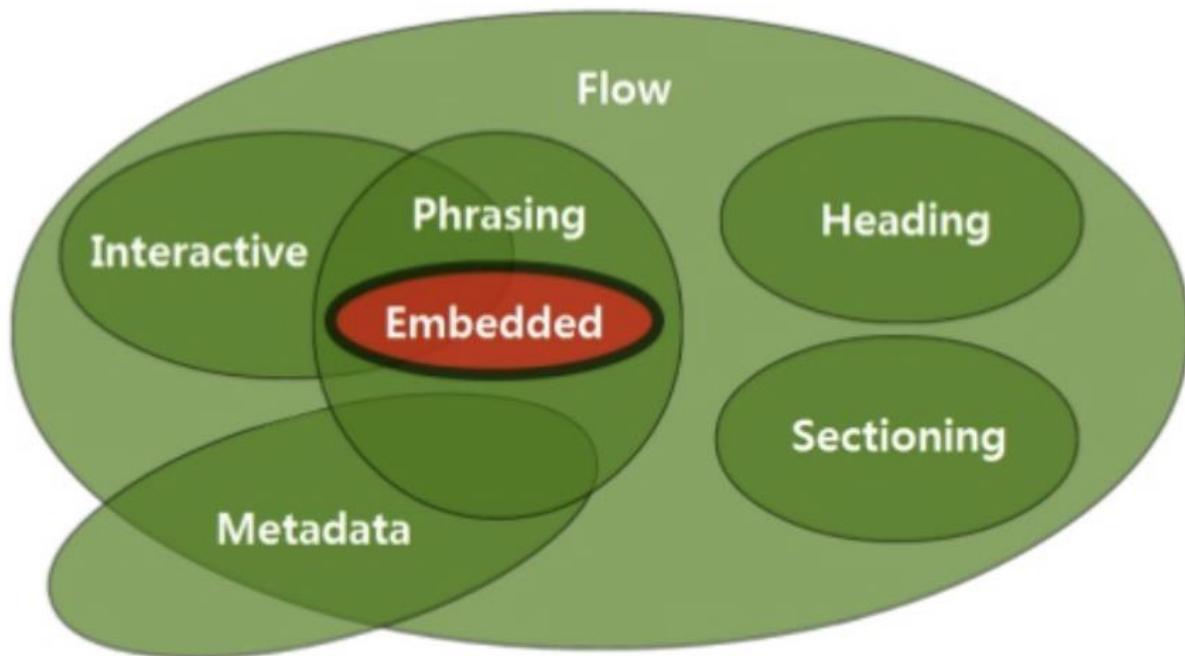


a, abbr, area, audio, b, bdi,
bdo, br, button, canvas,
cite, code, command,
datalist, del, dfn, em,
embed, i, iframe, img,
input, ins, kbd, keygen,
label, link, map, mark,
math, meta, meter,
noscript, object, output,
progress, q, ruby, s, samp,
script, select, small, span,
strong, sub, sup, svg,
textarea, time, u, var, video,
wbr

○ Embedded

- **Embedded : 임베디드**

다른 자원(미디어, 문서, 그래픽 등)을 문서에 삽입



**audio,
canvas,
embed,
iframe,
img, math,
object,
svg, video**

○ 예제

```
<!doctype html>
<head>
    <meta charset="utf-8">
    <title>문서 제목</title>
</head>
<body>

    <div id="container">

        <header>
            ... 헤더 ...
        </header>

        <div id="main">
            ... 내용 ...
        </div>

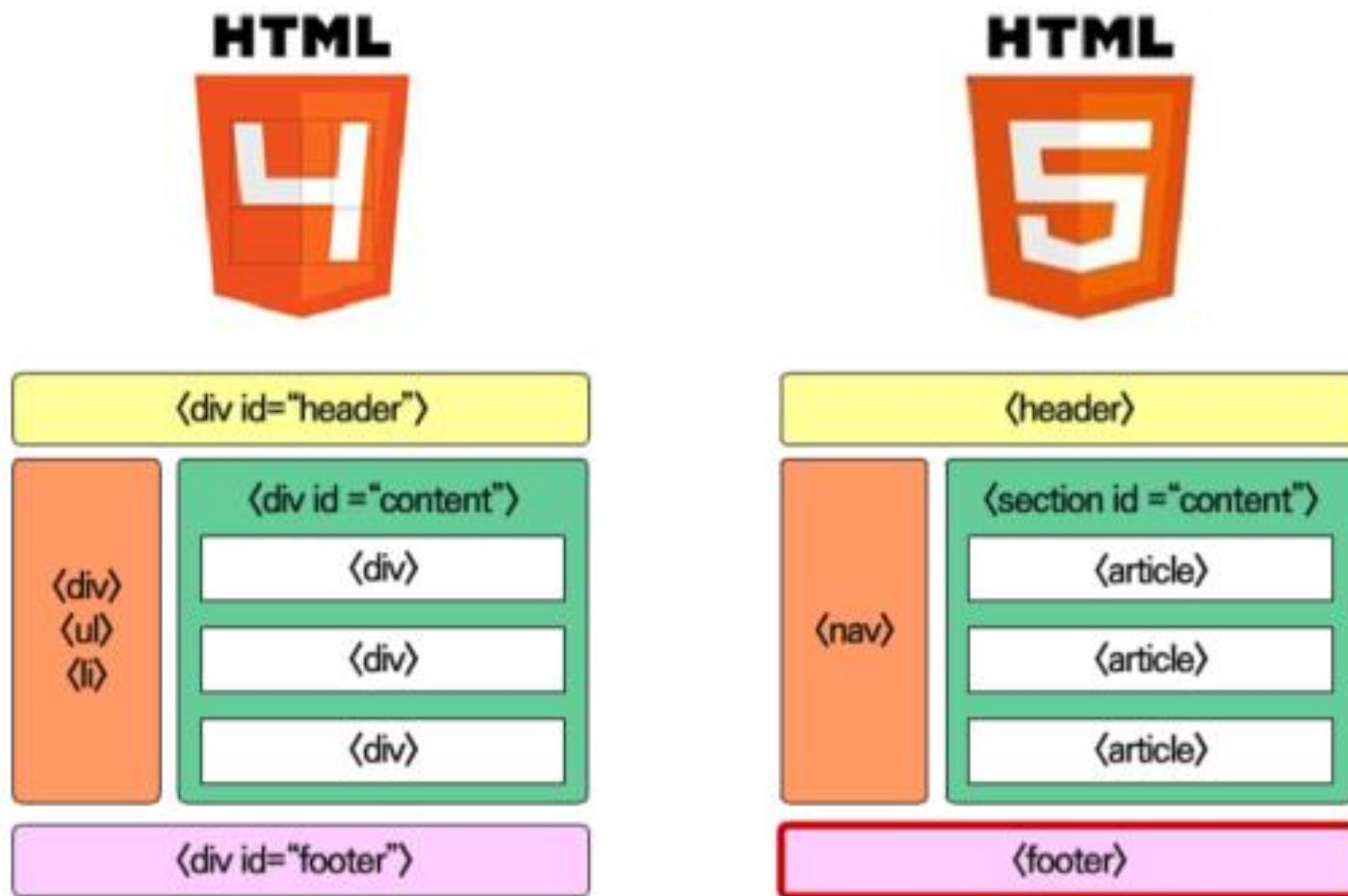
        <footer>
            ... 푸터 ...
        </footer>

    </div>

</body>
</html>
```

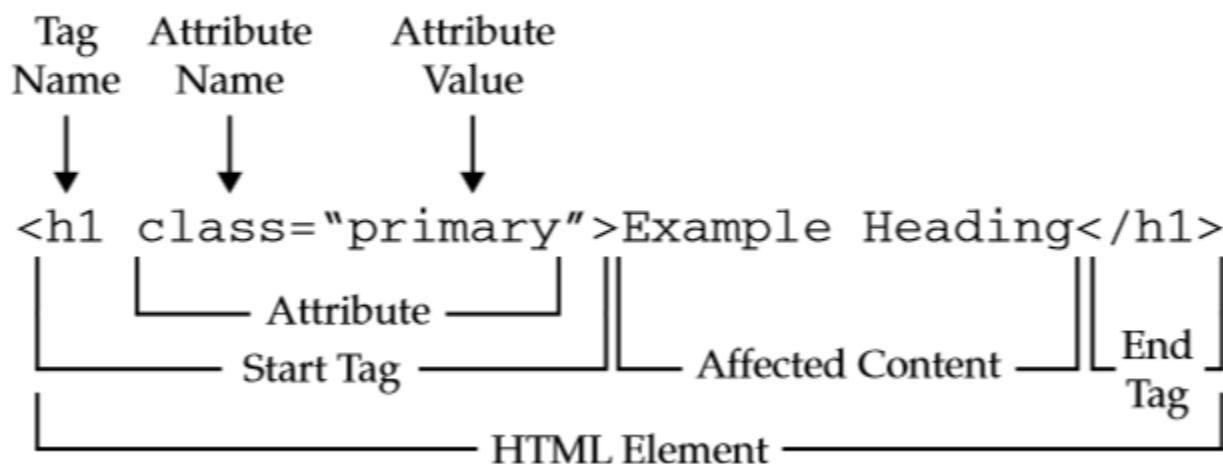
4.2. HTML

■ HTML4 vs HTML5



■ Element(Tag)

A graphical overview of the HTML markup syntax shown so far is presented here:



○ HTML 기본 구조

```
<!doctype html>
<html lang="ko">
<head>
  <meta charset="utf-8">
  <title>Simple HTML document</title>
</head>
<body>
  <h1>Hello World!</h1>
</body>
</html>
```

The `<head>` element is a container for metadata (data about data) and is placed between the `<html>` tag and the `<body>` tag.

HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

The following tags describe metadata: `<title>`, `<style>`, `<meta>`, `<link>`, `<script>`, and `<base>`.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

A `<meta>` viewport element gives the browser instructions on how to control the page's dimensions and scaling.

The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport `<meta>` tag:

○ Headings

```
<!doctype html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <title>Example of HTML headings tag</title>
</head>
<body>
    <h1>Heading level 1</h1>
    <h2>Heading level 2</h2>
    <h3>Heading level 3</h3>
    <h4>Heading level 4</h4>
    <h5>Heading level 5</h5>
    <h6>Heading level 6</h6>
</body>
</html>
```

- Headings are important!

Search engines use the headings to index the structure and content of your web pages.

Users skim your pages by its headings. It is important to use headings to show the document structure.

`<h1>` headings should be used for main headings, followed by `<h2>` headings, then the less important `<h3>`, and so on.

○ Paragraphs

```
<!doctype html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <title>Example of HTML Paragraphs</title>
</head>
<body>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
</body>
</html>
```

Tag	Description
<u><p></u>	Defines a paragraph
<u>
</u>	Inserts a single line break
<u><pre></u>	Defines pre-formatted text

○ Links

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <title>Example of HTML Links</title>
</head>
<body>
    <p><a href="https://www.tutorialrepublic.com/">Tutorial Republic</a></p>
    <p><a href="/examples/images/kites.jpg"></a></p>
    <p><a href="https://www.google.com/" target="_blank">Google Search</a></p>
</body>
</html>
```

The `title` attribute specifies extra information about an element.

The information is most often shown as a tooltip text when the mouse moves over the element.

The `target` attribute can have one of the following values:

- `_blank` - Opens the linked document in a new window or tab
- `_self` - Opens the linked document in the same window/tab as it was clicked (this is default)
- `_parent` - Opens the linked document in the parent frame
- `_top` - Opens the linked document in the full body of the window
- `framename` - Opens the linked document in a named frame

○ Links

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <title>Example of HTML Links</title>
</head>
<body>
    <p><a href="https://www.tutorialrepublic.com/">Tutorial Republic</a></p>
    <p><a href="/examples/images/kites.jpg"></a></p>
    <p><a href="https://www.google.com/" target="_blank">Google Search</a></p>
</body>
</html>
```

The `title` attribute specifies extra information about an element.

The information is most often shown as a tooltip text when the mouse moves over the element.

The `target` attribute can have one of the following values:

- `_blank` - Opens the linked document in a new window or tab
- `_self` - Opens the linked document in the same window/tab as it was clicked (this is default)
- `_parent` - Opens the linked document in the parent frame
- `_top` - Opens the linked document in the full body of the window
- `framename` - Opens the linked document in a named frame

○ Image

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <h2>The picture Element</h2>
    <picture>
        <source media="(min-width: 650px)"
srcset="https://www.w3schools.com/html/img_pink_flowers.jpg">
        <source media="(min-width: 465px)"
srcset="https://www.w3schools.com/html/img_white_flower.jpg">
        
    </picture>
    <p>Resize the browser to see different versions of the picture loading at
different viewport sizes. The browser looks for the first source element where the
media query matches the user's current viewport width, and fetches the image
specified in the srcset attribute.</p>
</body>
</html>



<map name="workmap">
    <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
    <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
    <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```

List

```
<ul>
    <li>Chocolate Cake</li>
    <li>Black Forest Cake</li>
    <li>Pineapple Cake</li>
</ul>
<ol>
    <li>Mix ingredients</li>
    <li>Bake in oven for an hour</li>
    <li>Allow to stand for ten minutes</li>
</ol>
<dl>
    <dt>Bread</dt>
    <dd>A baked food made of flour.</dd>
    <dt>Coffee</dt>
    <dd>A drink made from roasted coffee beans.</dd>
</dl>
```

Tag	Description
<u></u>	Defines an unordered list
<u></u>	Defines an ordered list
<u></u>	Defines a list item
<u><dl></u>	Defines a description list
<u><dt></u>	Defines a term in a description list
<u><dd></u>	Describes the term in a description list

○ Block and Inline

➤ Block

- A block-level element always starts on a new line and takes up the full width available

```
<div>Hello</div>  
<div>World</div>
```

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>	<div>	<dl>	<dt>
<fieldset>	<figcaption>	<figure>	<footer>	<form>	<h1>-<h6>	<header>	<hr>	
<main>	<nav>	<noscript>		<output>	<p>	<pre>	<section>	<table>
<tfoot>		<video>						

➤ Inline

- An inline element does not start on a new line and only takes up as much width as necessary

```
<span>Hello</span>  
<span>World</span>
```

<a>	<abbr>	<acronym>		<bdo>	<big>	 	<button>	<cite>
<code>	<dfn>		<i>		<input>	<kbd>	<label>	<map>
<object>	<q>	<samp>	<script>	<select>	<small>			<sub>
<sup>	<textarea>	<time>	<tt>	<var>				

Layout

1 COLUMN WIDE



2 COLUMN, LEFT



2 COLUMN, RIGHT



3 COLUMN, LEFT



3 COLUMN, RIGHT



3 COLUMN, CENTER

○ 예제

- DTD(Document Type Declaration) 선언

```
<!DOCTYPE html>
```

- 언어 속성 설정

```
<html lang="ko">
```

- 문서 정보(meta) 설정

```
<!-- 스마트 기기 등 다양한 해상도를 위해 설정 -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- HTML 문서에 대한 제작자 설정 -->
<meta name="author" content="빅데이터">

<!-- HTML 문서에 대한 설명 설정 -->
<meta name="description" content="빅데이터 비정형 데이터(HTML5) 예제">

<!-- HTML 문서에 대한 키워드 설정 -->
<meta name="keywords" content="빅데이터, 비정형, HTML5">

<!-- HTML 문서 재로딩 설정 -->
<meta http-equiv="refresh" content="10, url=http://www.naver.com">

<!-- 검색엔진(Crawler) 허가 설정 -->
<meta name="robots" content="all" >
```

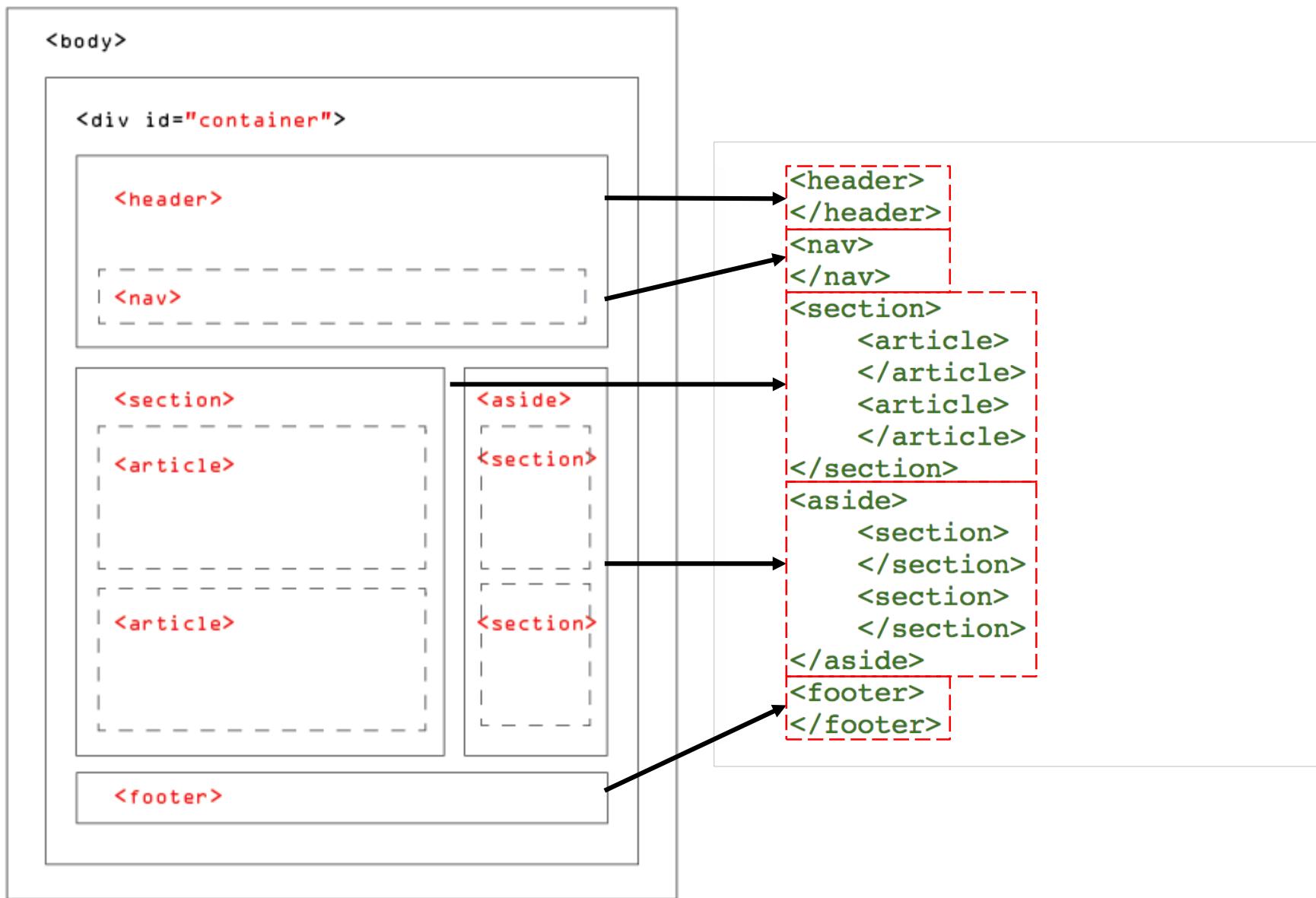
➤ Robot

`robots` which defines the behaviour that cooperative crawlers, or "robots", should use with the page. It is a comma-separated list of the values below:

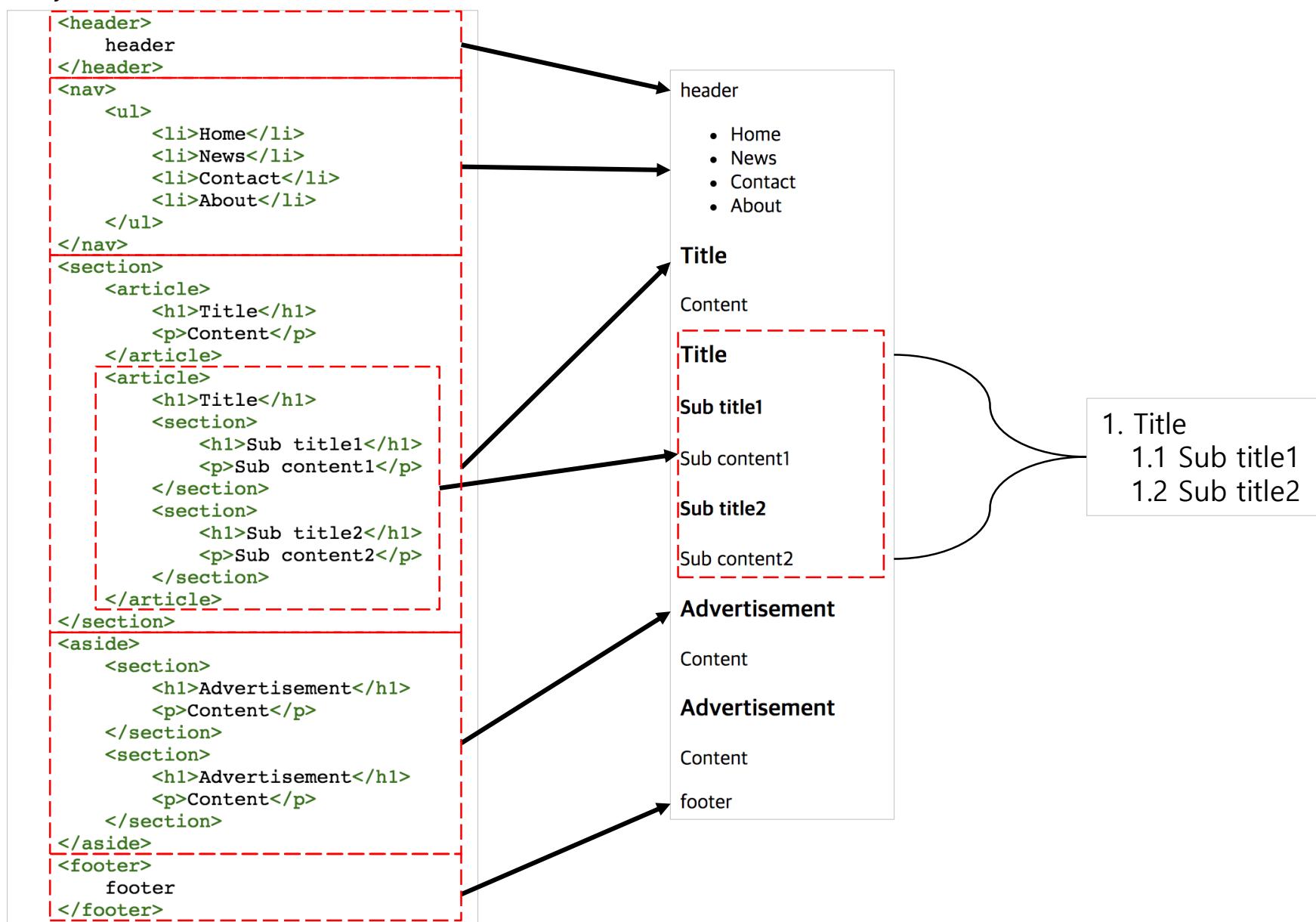
Values for the content of `<meta name="robots">`

Value	Description	Used by
<code>index</code>	Allows the robot to index the page (default).	All
<code>noindex</code>	Requests the robot to not index the page.	All
<code>follow</code>	Allows the robot to follow the links on the page (default).	All
<code>nofollow</code>	Requests the robot to not follow the links on the page.	All
<code>none</code>	Equivalent to <code>noindex, nofollow</code>	 Google
<code>noodp</code>	Prevents using the  Open Directory Project description, if any, as the page description in search engine results.	 Google,  Yahoo,  Bing
<code>noarchive</code>	Requests the search engine not to cache the page content.	 Google,  Yahoo,  Bing
<code>nosnippet</code>	Prevents displaying any description of the page in search engine results.	 Google,  Bing
<code>noimageindex</code>	Requests this page not to appear as the referring page of an indexed image.	 Google
<code>nocache</code>	Synonym of <code>noarchive</code> .	 Bing

➤ Layout (2 Column with Header and Footer)



➤ Layout (2 Column with Header and Footer) Skeleton



➤ Layout (2 Column with Header and Footer) main

```
<main>
  <header>
    header
    <nav>
      <ul>
        <li>Home</li>
        <li>News</li>
        <li>Contact</li>
        <li>About</li>
      </ul>
    </nav>
  </header>
  <section>
    <article>
      <h1>Title</h1>
      <p>Content</p>
    </article>
    <article>
      <h1>Title</h1>
      <section>
        <h1>Sub title1</h1>
        <p>Sub content1</p>
      </section>
      <section>
        <h1>Sub title2</h1>
        <p>Sub content2</p>
      </section>
    </article>
  </section>
  <aside>
    <section>
      <h1>Advertisement</h1>
      <p>Content</p>
    </section>
    <section>
      <h1>Advertisement</h1>
      <p>Content</p>
    </section>
  </aside>
  <footer>
    footer
  </footer>
</main>
```

header

- Home
- News
- Contact
- About

Title

Content

Title

Sub title1

Sub content1

Sub title2

Sub content2

Advertisement

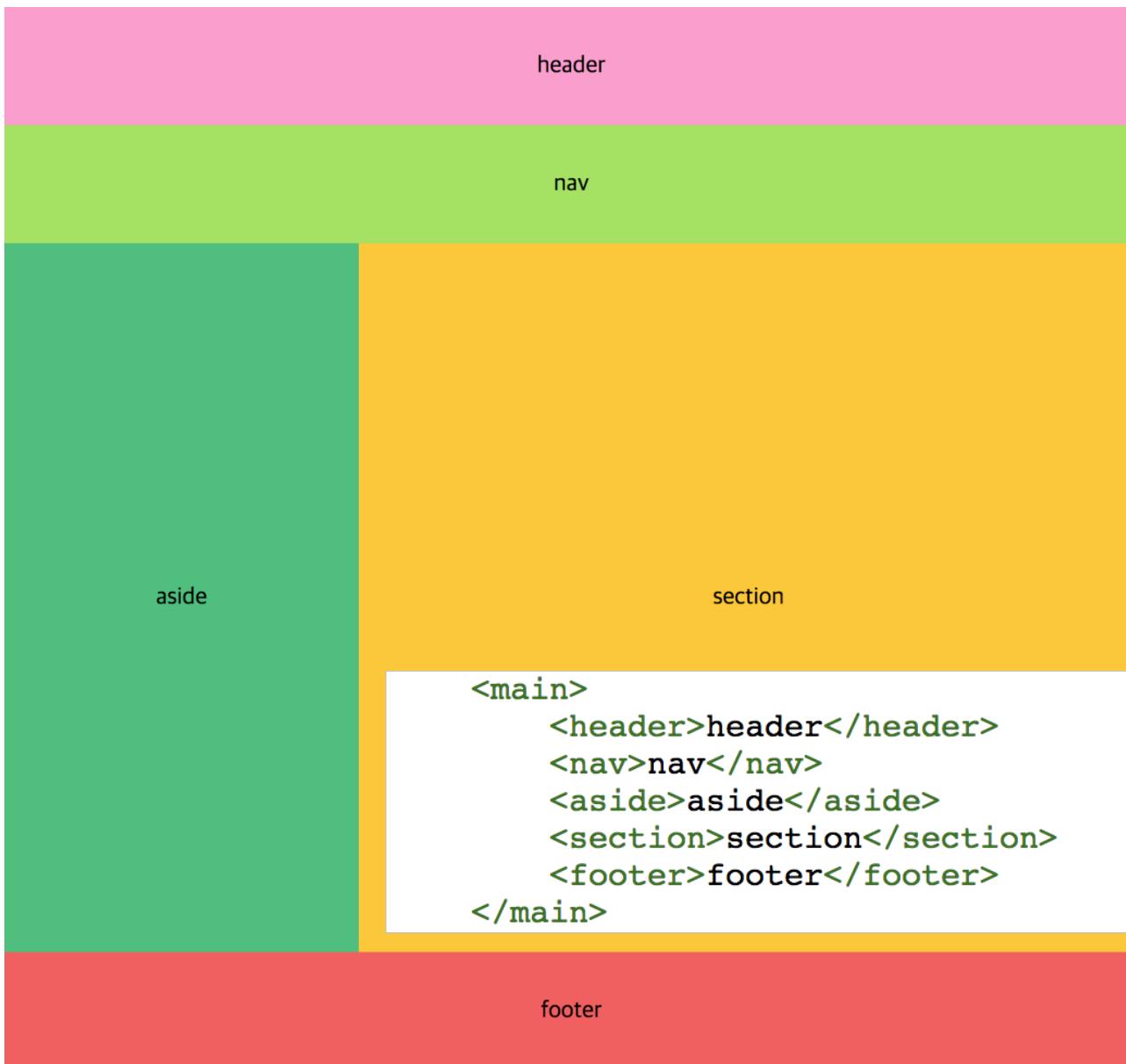
Content

Advertisement

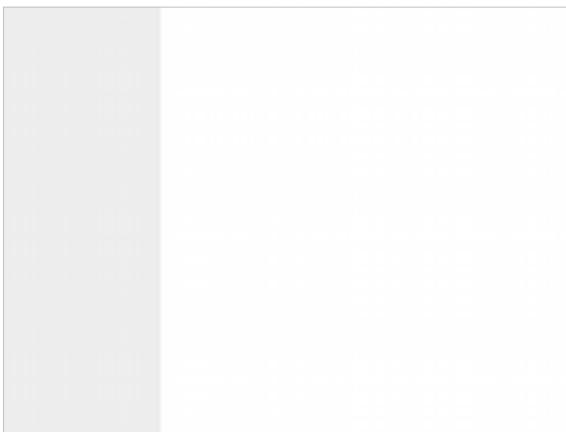
Content

footer

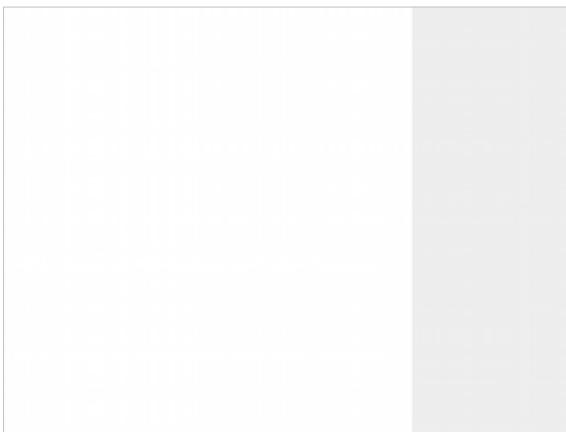
➤ Layout (2 Column – Right)



➤ Layout (2 Column – Left Menu)



➤ Layout (2Column – Right Menu)



```
<main>
  <nav id="nav">
    <h1>Left heading</h1>
    <ul>
      <li><a href="#">Link 1</a></li>
      <li><a href="#">Link 2</a></li>
      <li><a href="#">Link 3</a></li>
      <li><a href="#">Link 4</a></li>
      <li><a href="#">Link 5</a></li>
    </ul>
  </nav>

  <section id="content">
    <h1>Heading</h1>
    <p>Text</p>
  </section>
</main>
```

4.3. CSS



■ What is CSS?

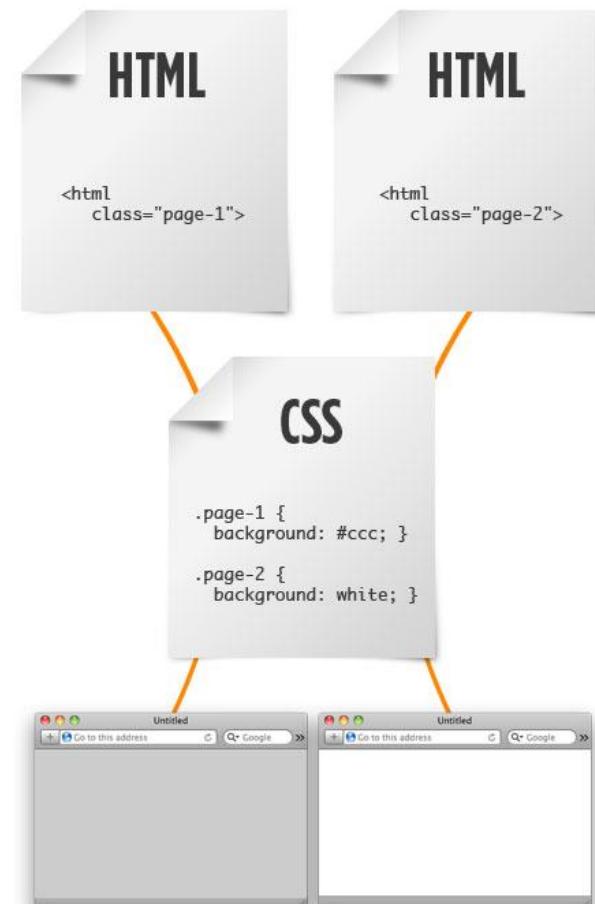
- CSS stands for Cascading Style Sheets
- The standard language for defining styles on web pages
- Describes how HTML elements should be displayed
- Defining styles for any structured document format (Not only HTML but XML, SVG, MathML, etc)
- Styles are set using CSS properties

■ Why use CSS?

- Define styles for web pages
- Better type and layout controls
 - a designer can do things that are not possible with HTML alone
- Less work
 - edit one stylesheet to change the presentation of an entire site
- Potentially smaller documents
 - code and multiple tags are unnecessary
- Improved accessibility
 - for different devices and screen sizes
- Reliable browser support
 - mostly

■ CSS solved a Big Problem

- HTML was created to describe the content of a web page
- Style formatting were added to the HTML 3.2 specification
 - a long and expensive process
- CSS removed the style formatting from the HTML page



■ 예제

○ Before

```
<!DOCTYPE html>
<html>
<head>
<title>Example</title>
</head>
<body>
<main>
<h1>HTML Page</h1>
<p>This is a basic web page.</p>
</main>
</body>
</html>
```

HTML Page

This is a basic web page.

○ After

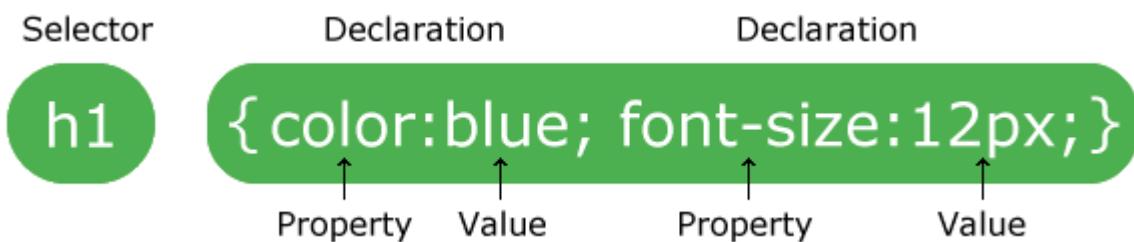
```
<style>
main {
  width: 200px;
  height: 200px;
  padding: 10px;
  background: beige;
}
h1 {
  font-family: fantasy, cursive, serif;
  color: olivedrab;
  border-bottom: 1px dotted darkgreen;
}
p {
  font-family: sans-serif;
  color: orange;
}
</style>
</head>
```

HTML Page

This is a basic web page.

■ Syntax

- Consists of a selector and a declaratory block



- The declaration block contains one or more declarations separated by semicolons
- Each declaration includes a CSS property name and a value, separated by a colon
- Grouping selectors

```
h1, h2, h3, h4, h5, h6 { color: blue }
```

- Applying multiple properties

```
h1 { color:blue; font-family: arial, helvetica, "sans serif" }
```

■ Selector

- to find(select) HTML elements based on their element name, id, class, attribute, and more

Selector	Example	Example description
<u>.class</u>	.intro	Selects all elements with <u>class</u> ="intro"
<u>#id</u>	#firstname	Selects the element with <u>id</u> ="firstname"
*	*	Selects all elements
<u>element</u>	p	Selects all <p> <u>elements</u>
<u>element,element</u>	div, p	Selects all <div> elements and all <p> elements
<u>element element</u>	div p	Selects all <p> elements <u>inside</u> <div> elements
<u>element>element</u>	div > p	Selects all <p> elements where the <u>parent</u> is a <div> element
<u>element+element</u>	div + p	Selects all <p> elements that are placed immediately <u>after</u> <div> elements
<u>[attribute]</u>	[target]	Selects all elements with a <u>target</u> <u>attribute</u>
<u>[attribute=value]</u>	[target=_blank]	Selects all elements with target="_blank"
<u>[attribute~=value]</u>	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
<u>[attribute =value]</u>	[lang =en]	Selects all elements with a lang attribute value starting with "en"
<u>[attribute^=value]</u>	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
<u>[attribute\$=value]</u>	a[href\$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
<u>[attribute*=value]</u>	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"

<u>:active</u>	a:active	Selects the active link
<u>::after</u>	p::after	Insert something after the content of each <p> element
<u>::before</u>	p::before	Insert something before the content of each <p> element
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children (including text nodes)
<u>:enabled</u>	input:enabled	Selects every enabled <input> element
<u>:first-child</u>	p:first-child	Selects every <p> element that is the first child of its parent
<u>::first-letter</u>	p::first-letter	Selects the first letter of every <p> element
<u>::first-line</u>	p::first-line	Selects the first line of every <p> element
<u>:first-of-type</u>	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
<u>:focus</u>	input:focus	Selects the input element which has focus
<u>:hover</u>	a:hover	Selects links on mouse over
<u>:in-range</u>	input:in-range	Selects input elements with a value within a specified range
<u>:invalid</u>	input:invalid	Selects all input elements with an invalid value
<u>:lang(<i>language</i>)</u>	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)
<u>:last-child</u>	p:last-child	Selects every <p> element that is the last child of its parent
<u>:last-of-type</u>	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
<u>:link</u>	a:link	Selects all unvisited links
<u>:not(<i>selector</i>)</u>	:not(p)	Selects every element that is not a <p> element
<u>:nth-child(<i>n</i>)</u>	p:nth-child(2)	Selects every <p> element that is the second child of its parent

■ 예제

```
<head>
<style>
  div {
    padding:10px;
    background-color:rgba(255,0,0,.5);
  }
  #id {
    color:rgb(0,0,255);
  }
  div.class {
    background-color:yellow;
  }
  p.class {
    color:green;
  }
  div + p {
    background-color:hsl(90,50%,50%);
  }
</style>
</head>
<body>
<div id="id">CSS Element</div>
<div class="class">CSS Class</div>
<p class="class">CSS classes can be very useful</p>
</body>
```

CSS Element

CSS Class

CSS classes can be very useful

■ Font

Property	Description
<u>font</u>	Sets all the font properties in one declaration
<u>font-family</u>	Specifies the font family for text
<u>font-size</u>	Specifies the font size of text
<u>font-style</u>	Specifies the font style for text
<u>font-variant</u>	Specifies whether or not a text should be displayed in a small-caps font
<u>font-weight</u>	Specifies the weight of a font

○ Families

generic family - a group of font families with a similar look (like "Serif" or "Monospace")
font family - a specific font family (like "Times New Roman" or "Arial")

○ Size

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

■ 한글폰트

```
<head>
<style>
    .serif {font-family:"바탕체", BatangChe, serif;}
    .sansserif {font-family:"맑은고딕", "Malgun Gothic", sans-serif;}

    .size {font-size:1.0em;}      The em size unit is recommended by the W3C.
    .asize {font-size:16px;}
    .rsize {font-size:100%;}

    .bold {font-weight:bold;}
    .italic {font-style:italic;}
    .normal {font-weight:normal; font-style:normal;}
</style>
</head>
<body>
    <h1 class="size normal">CSS font-family</h1>
    <p class="serif asize">This is a <span class="bold">paragraph</span>, shown in the
    <span class="italic">BatanChe</span> font.</p>
    <p class="sansserif rsize">This is a paragraph, shown in the Malgun Gothic font.</p>
</body>
```

돋움 : Dotum

굴림 : Gulim

바탕 : Batang

궁서 : Gungsuh

새굴림: New Gulim

돋움체 : DotumChe

굴림체 : GulimChe

바탕체 : BatangChe

궁서체 : GungsuhChe

맑은 고딕 : Malgun Gothic

CSS font-family

This is a paragraph, shown in the *BatanChe* font.

This is a paragraph, shown in the Malgun Gothic font.

■ Text

Property	Description
<u>color</u>	Sets the color of text
<u>direction</u>	Specifies the text direction/writing direction
<u>letter-spacing</u>	Increases or decreases the space between characters in a text
<u>line-height</u>	Sets the line height
<u>text-align</u>	Specifies the horizontal alignment of text
<u>text-decoration</u>	Specifies the decoration added to text
<u>text-indent</u>	Specifies the indentation of the first line in a text-block
<u>text-shadow</u>	Specifies the shadow effect added to text
<u>text-transform</u>	Controls the capitalization of text
<u>text-overflow</u>	Specifies how overflowed content that is not displayed should be signaled to the user
<u>unicode-bidi</u>	Used together with the <u>direction</u> property to set or return whether the text should be overridden to support multiple languages in the same document
<u>vertical-align</u>	Sets the vertical alignment of an element
<u>white-space</u>	Specifies how white-space inside an element is handled
<u>word-spacing</u>	Increases or decreases the space between words in a text

■ 예제

```
<head>
<style>
p {padding:5px; border:1px solid #000;}
.rtl {direction:rtl; unicode-bidi:bidi-override;}
.letter {letter-spacing:10px;}
.word {word-spacing:10px;}
.line {line-height:200%;}
.align {text-align:center; /* left|right|justify */}
.decoration {text-decoration:underline; /* overline|line-through */}
.indent {text-indent:1.0em;}
.shadow {text-shadow:1px 1px 1px; /* h-shadow v-shadow blur-radius color */}
.transform {text-transform:capitalize; /* uppercase|lowercase */}
.overflow {width:100px;text-overflow:ellipsis;white-space:nowrap;overflow:hidden;}
</style>
</head>
<body>


Right to Left



Text Text



Align



Decoration



Indent 1.0em(default 16px). Note:



Negative values are allowed. The first line will be indented to the left if the value is negative.



Indent 1.0em(default 16px). Note:

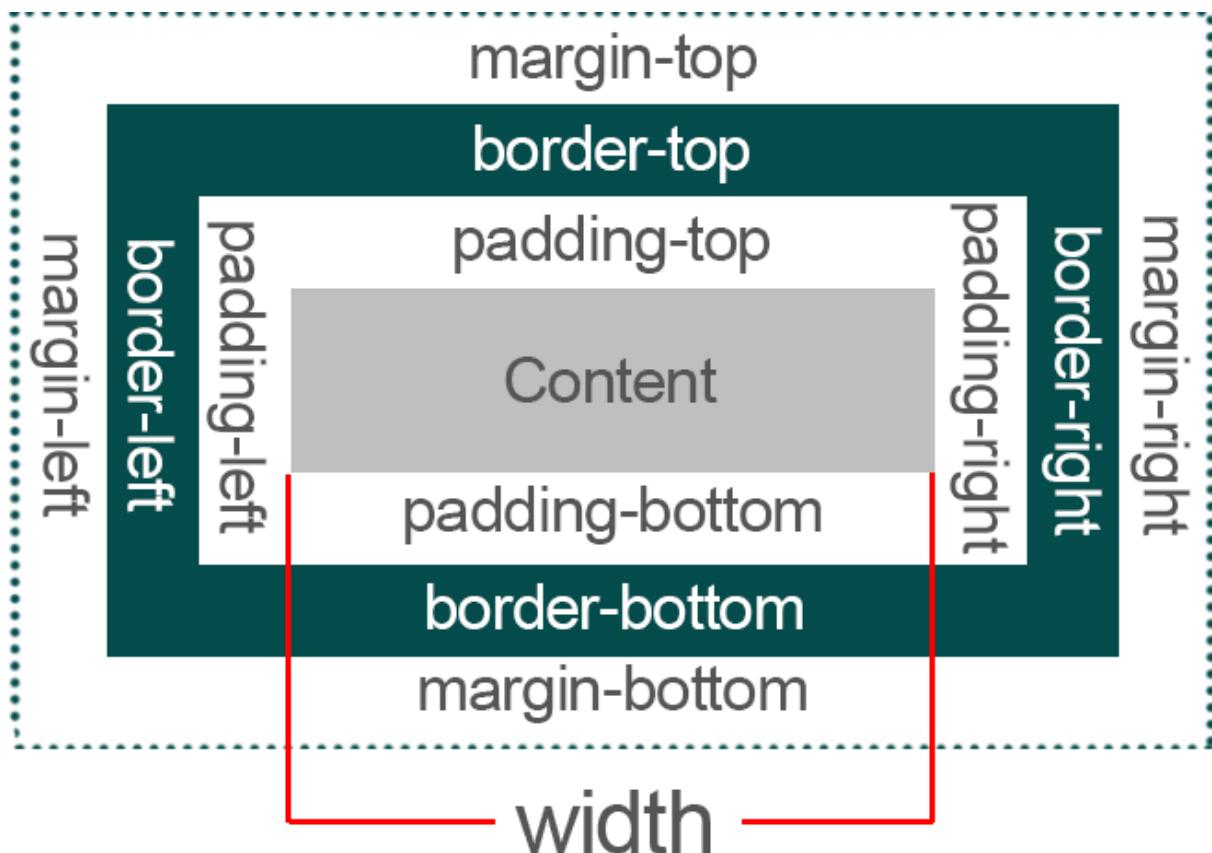


Negative values are allowed. The first line will be indented to the left if the value is negative.


</body>
```

tfeL ot thgIR
Text T e x t
Align
Decoration
Indent 1.0em(default 16px). Note: Negative values are allowed. The First Line Will Be Indented To The Left If The Value Is Negative.
Indent 1...

■ Box Model(margin/padding)



```
box-sizing: content-box|border-box|initial|inherit;
```

■ 예제

```
<head>
<style>
div {
    background-color: lightgrey;
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
</style>
</head>
<body>
    <h2>Demonstrating the Box Model</h2>
```

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.**</p>**

<div>This text is the actual content of the box. We have added a 25px padding, 25px margin and a 25px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.**</div>**

</body>

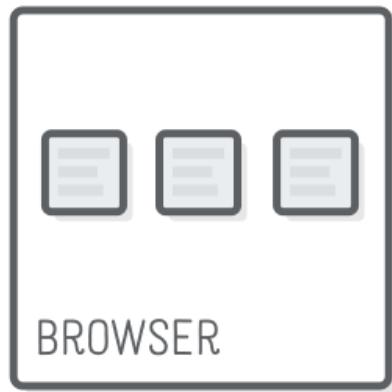
Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

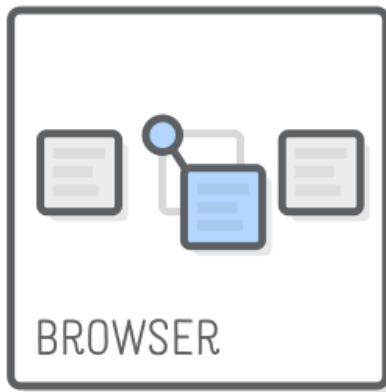
This text is the actual content of the box. We have added a 25px padding, 25px margin and a 25px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

■ Position

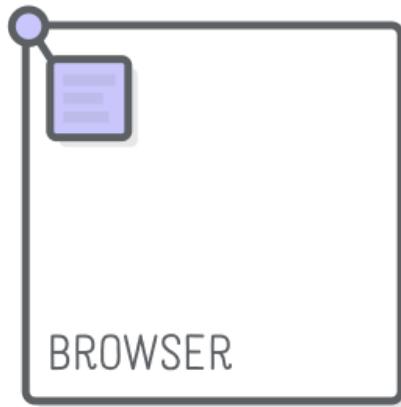
STATIC



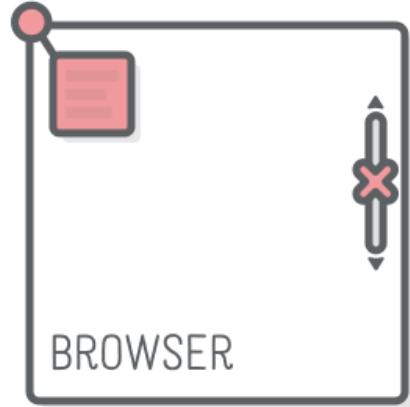
RELATIVE



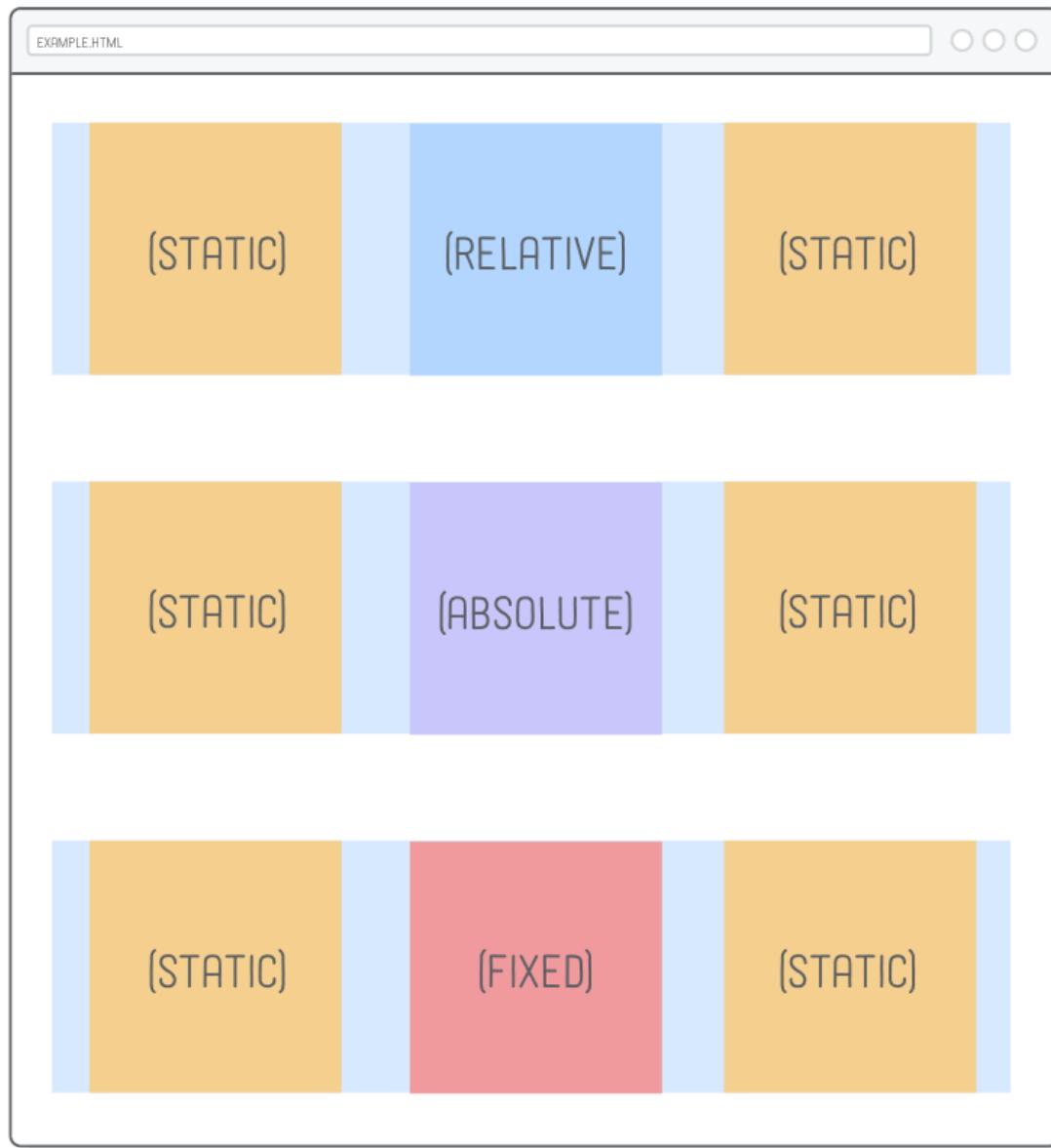
ABSOLUTE



FIXED

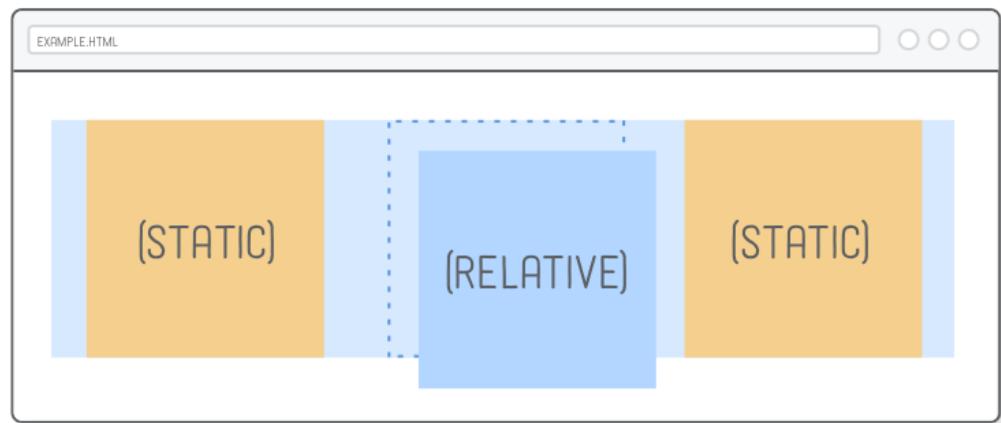


■ Position

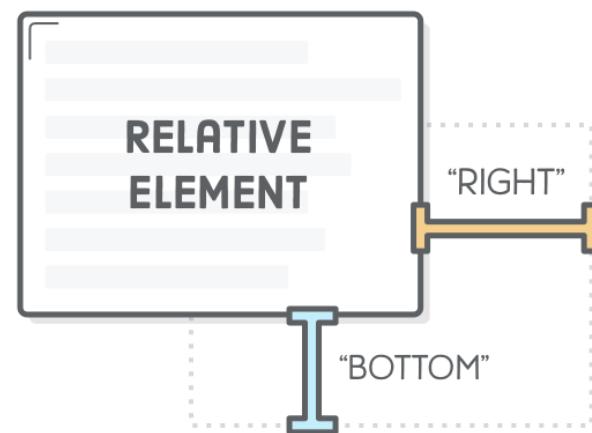
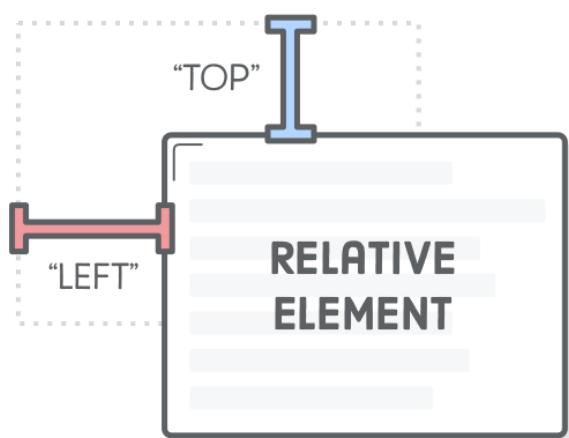


■ Position – Relative

```
.item-relative {  
    position: relative;  
    top: 30px;  
    left: 30px;  
}
```



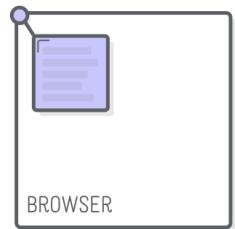
ORIGINAL POSITION



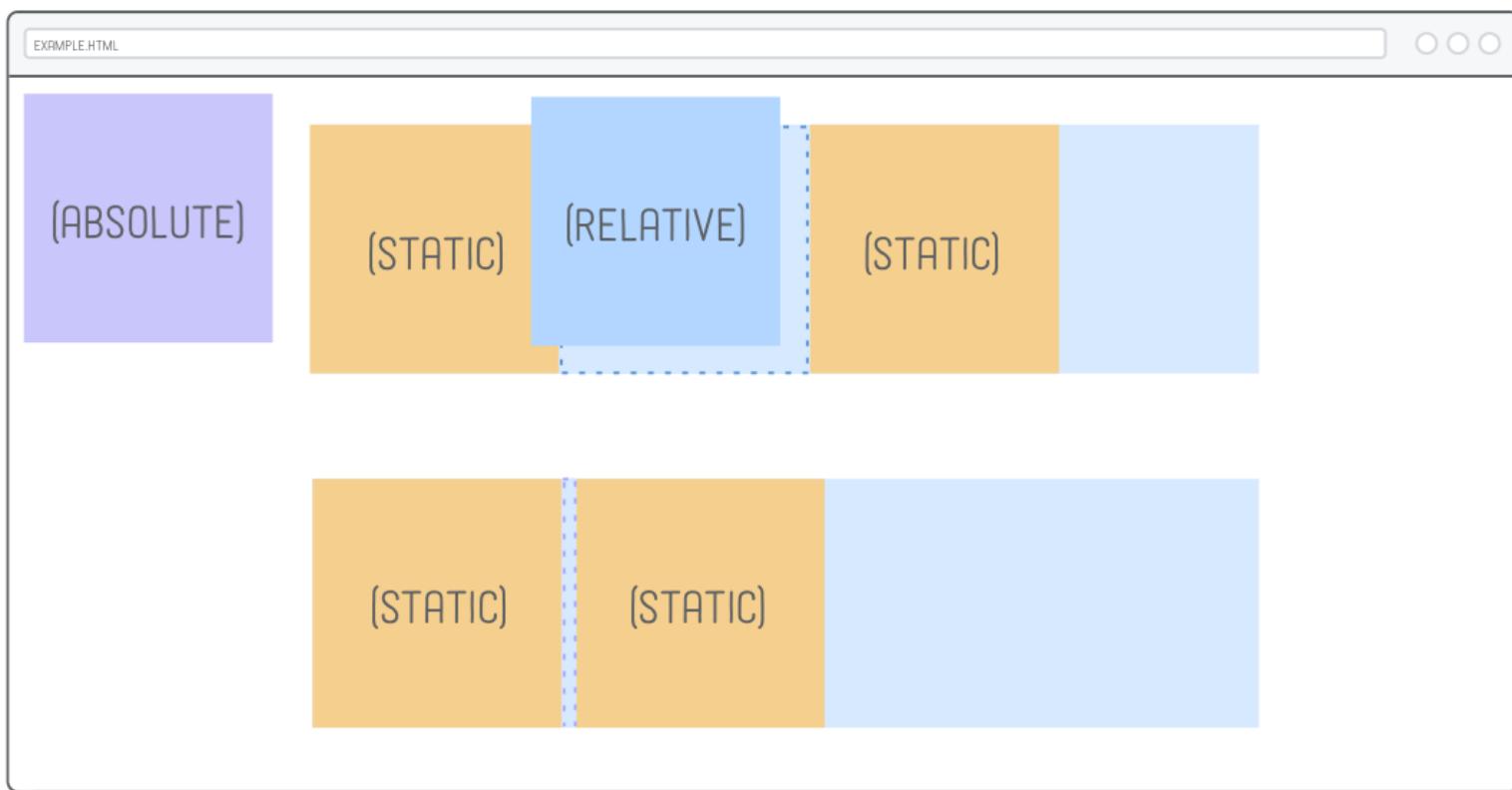
ORIGINAL POSITION

■ Position – Absolute

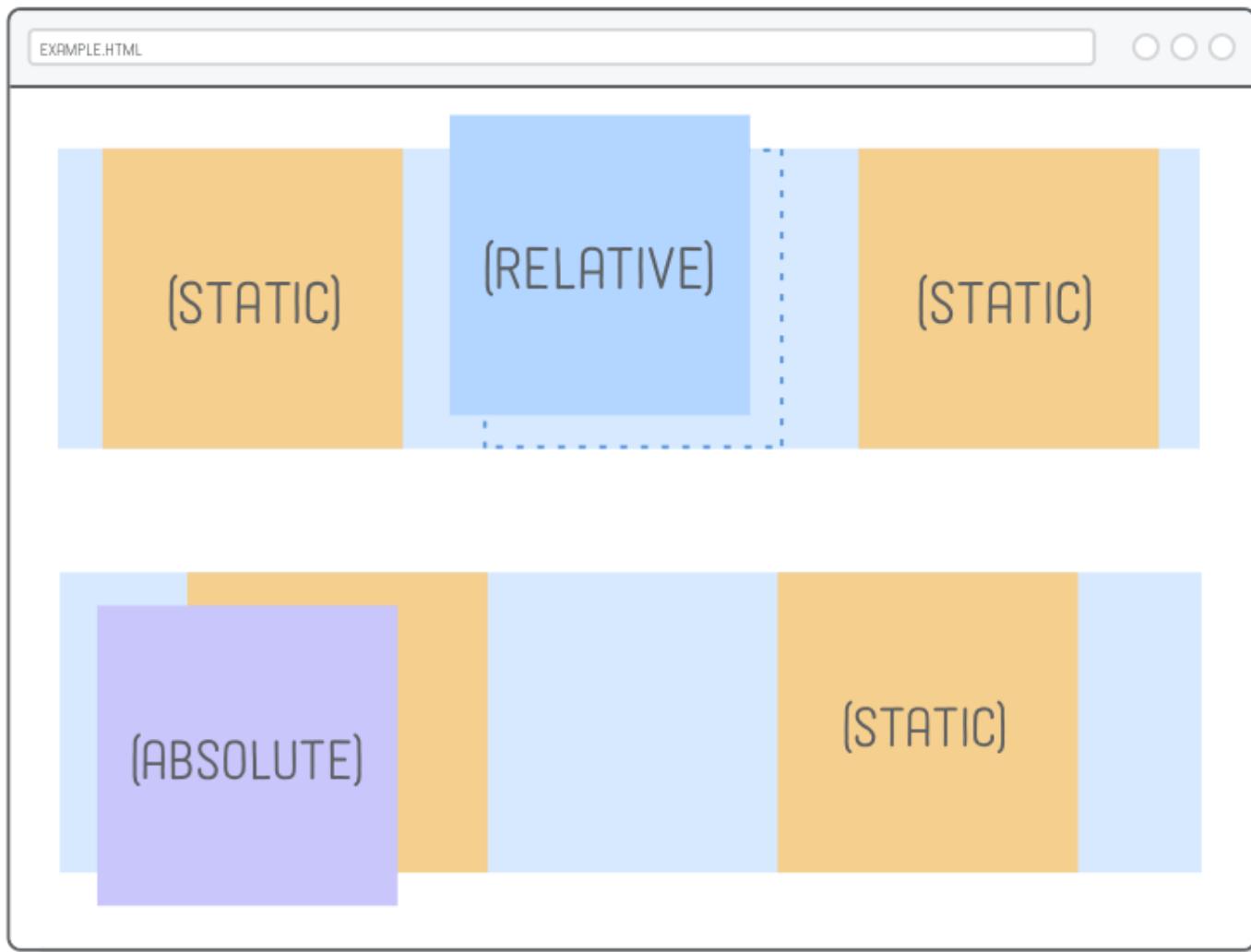
```
.item-absolute {  
    position: absolute;  
    top: 10px;  
    left: 10px;  
}
```



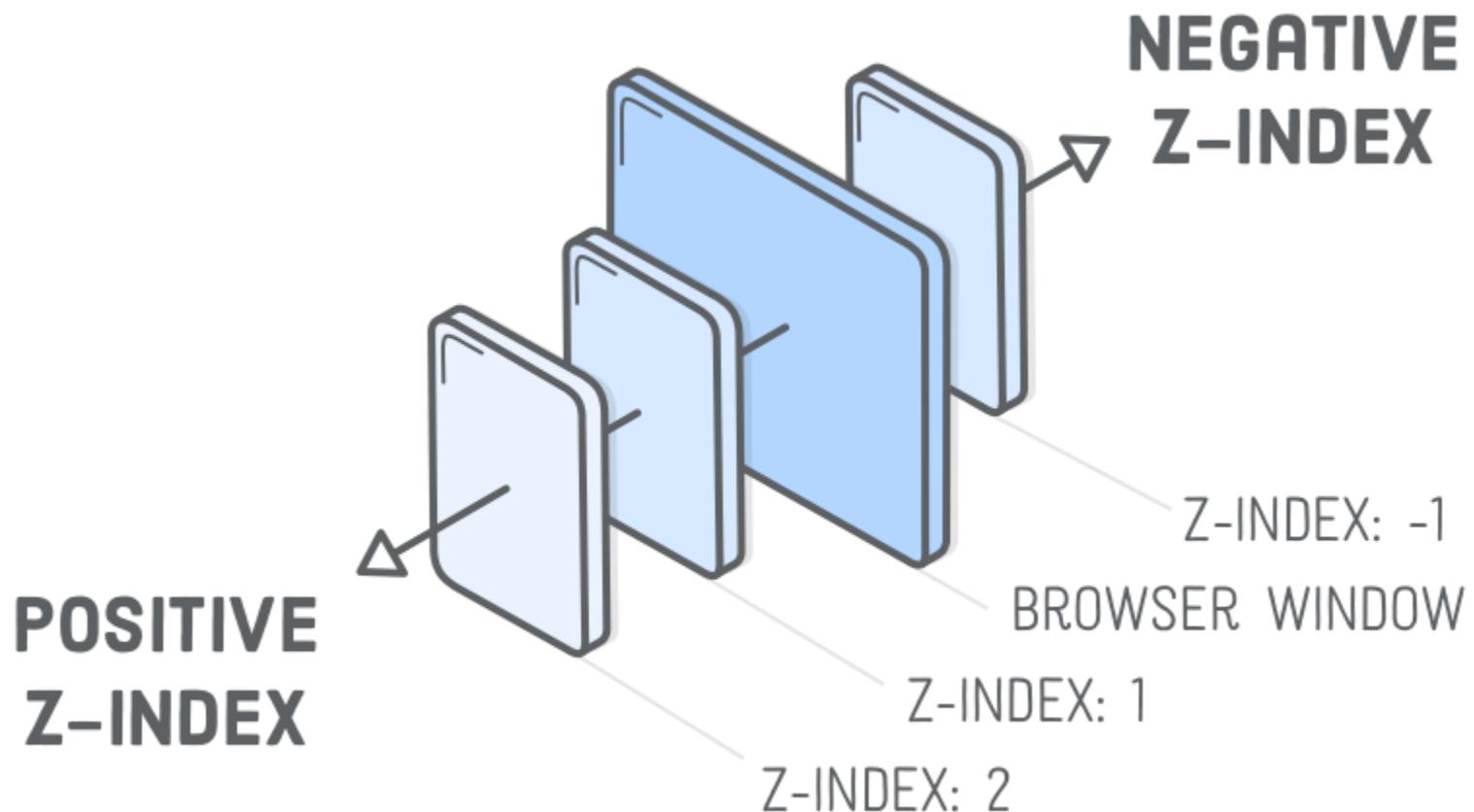
ABSOLUTE POSITIONING



■ Position – Relatively Absolute

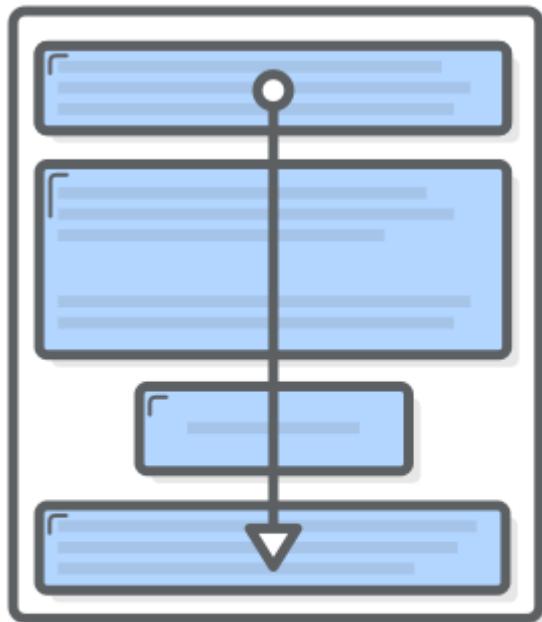


■ Z-index

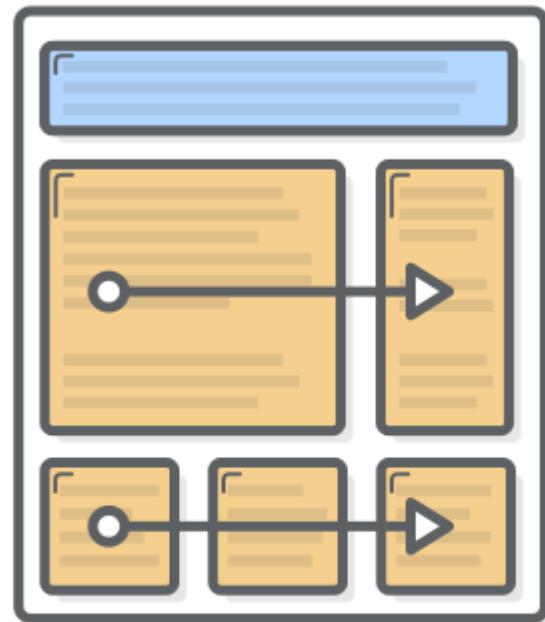


■ Float

- The float property is used for positioning and layout on web pages

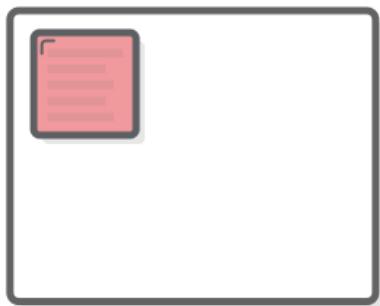


VERTICAL FLOW



HORIZONTAL FLOW

■ Multiple floats



LEFT ALIGN

FLOAT: LEFT;



CENTER ALIGN

MARGIN: 0 AUTO;



RIGHT ALIGN

FLOAT: RIGHT;



SIDE BAR LEFT
CONTENT LEFT



SIDE BAR LEFT
CONTENT RIGHT



SIDE BAR RIGHT
CONTENT LEFT

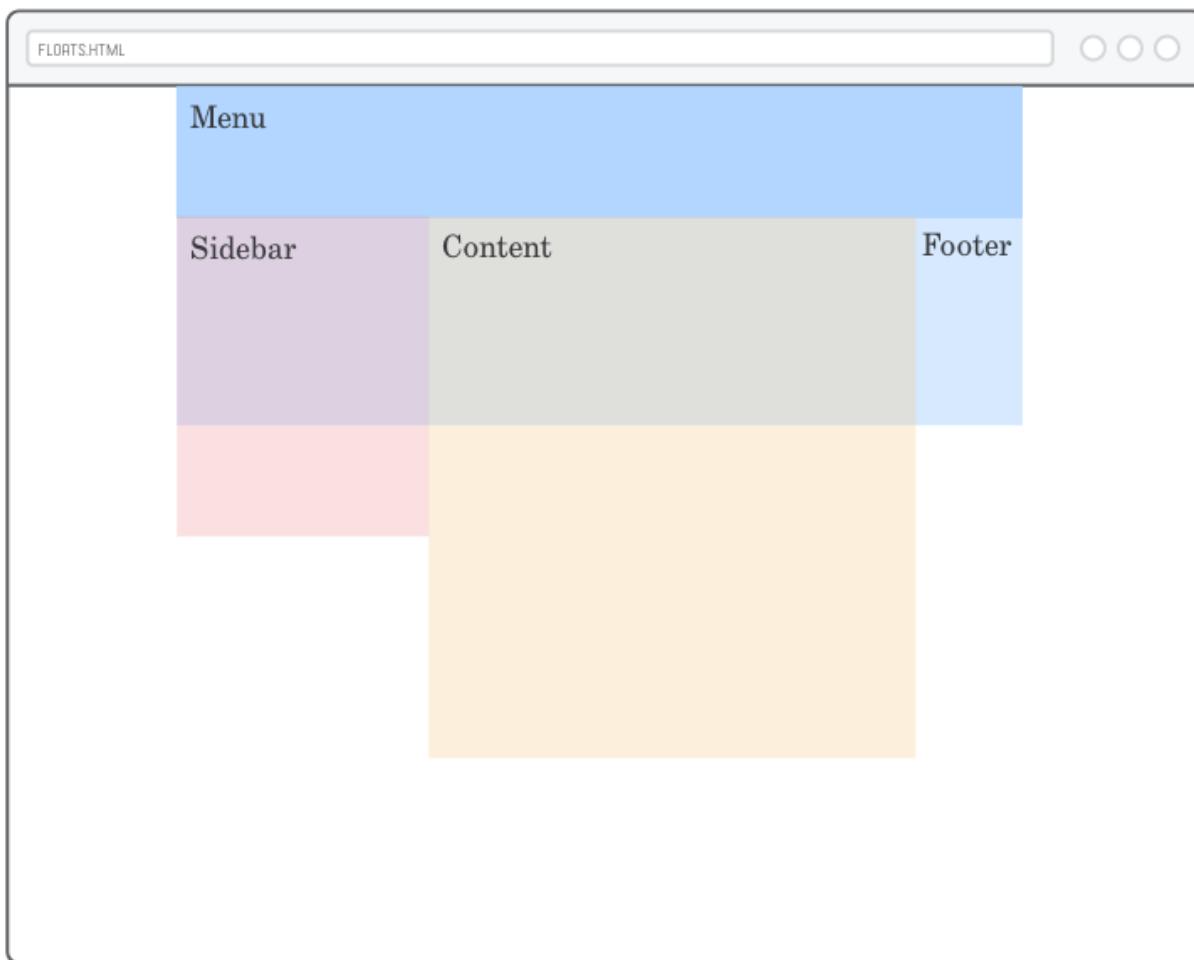


SIDE BAR RIGHT
CONTENT RIGHT

■ After a float

○ Problem

- floated boxes are removed from the normal flow of the page
- height of floated elements don't contribute to the vertical position of the footer
- it simply sticks itself below the last element that wasn't floated.



■ Clearfix Hack

- Elements after a floating element will flow around it. Use the "clearfix" hack to fix the problem

 Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Phasellus imperdiet, nulla et
 dictum interdum...

without clearfix



 Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Phasellus imperdiet, nulla et
 dictum interdum...

with clearfix



```
.clearfix::after {  
  content: "";  
  clear: both;  
  display: table;  
}
```

■ 예제

```
<main>
  <header>
    header
    <nav>
      <ul>
        <li>Home</li>
        <li>News</li>
        <li>Contact</li>
        <li>About</li>
      </ul>
    </nav>
  </header>
  <section>
    <article>
      <h1>Title</h1>
      <p>Content</p>
    </article>
    <article>
      <h1>Title</h1>
      <section>
        <h1>Sub title1</h1>
        <p>Sub content1</p>
      </section>
      <section>
        <h1>Sub title2</h1>
        <p>Sub content2</p>
      </section>
    </article>
  </section>
  <aside>
    <section>
      <h1>Advertisement</h1>
      <p>Content</p>
    </section>
    <section>
      <h1>Advertisement</h1>
      <p>Content</p>
    </section>
  </aside>
  <footer>
    footer
  </footer>
</main>
```

header

- Home
- News
- Contact
- About

Title

Content

Title

Sub title1

Sub content1

Sub title2

Sub content2

Advertisement

Content

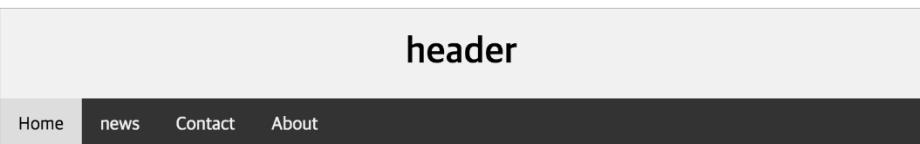
Advertisement

Content

footer

○ main, header

```
* {  
    margin:0;  
    padding:0;  
}  
  
main {  
    margin:0;  
    padding:0;  
}  
  
header {  
    background-color:#F1F1F1;  
    text-align:center;  
}  
  
header > h1 {  
    padding:20px;  
}  
  
nav {  
    overflow:hidden;  
    background-color:#333;  
}  
  
nav li {  
    float:left;  
    display:block;  
    color:#f2f2f2;  
    text-align:center;  
    padding:14px 16px;  
}  
  
nav li:hover {  
    background-color:#ddd;  
    color:black;  
}
```



○ float, position

```
header + section {  
    float:left;  
    width:70%;  
}  
  
section + aside {  
    float:left;  
    width:30%;  
}  
  
article, aside > section {  
    padding:20px;  
    margin:20px;  
    background-color:white;  
}  
  
article section {  
    background-color: #aaa;  
    width:calc(100%-10px);  
    margin-top:20px;  
    padding:10px;  
}  
  
p {  
    padding:10px 0;  
    text-align:center;  
    border:1px solid #000;  
}
```



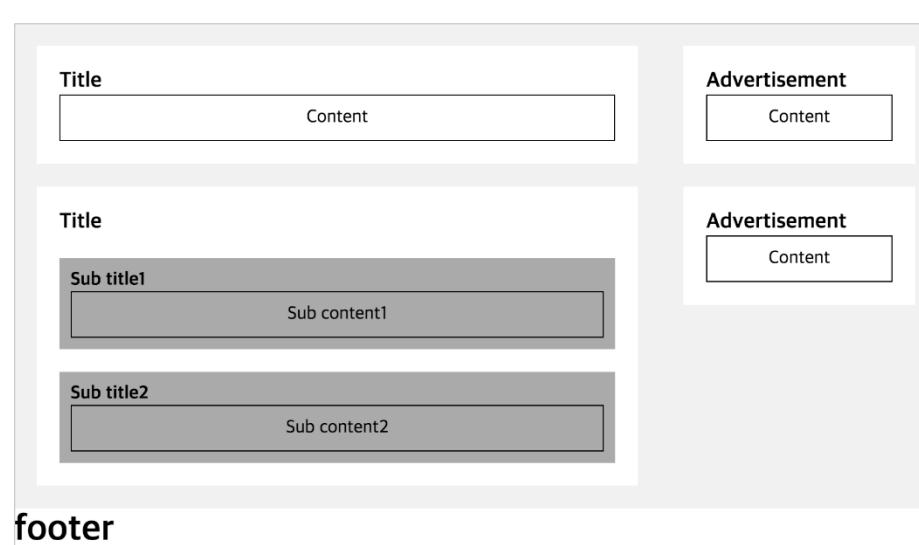
○ clearfix

```
<header>
  <h1>header</h1>
  <nav>
    <ul>
      <li>Home</li>
      <li>news</li>
      <li>Contact</li>
      <li>About</li>
    </ul>
  </nav>
</header>
<main class="clearfix">
  <section>
    <article>
      <h1>Title</h1>
      <p>Content</p>
    </article>
    <article>
      <h1>Title</h1>
      <section>
        <h1>Sub title1</h1>
        <p>Sub content1</p>
      </section>
      <section>
        <h1>Sub title2</h1>
        <p>Sub content2</p>
      </section>
    </article>
  </section>
  <aside>
    <section>
      <h1>Advertisement</h1>
      <p>Content</p>
    </section>
    <section>
      <h1>Advertisement</h1>
      <p>Content</p>
    </section>
  </aside>
</main>
<footer>
  <h1>footer</h1>
</footer>
```

```
main {
  margin:0;
  padding:0;
  background:#f1f1f1;
}

main > section {
  float:left;
  width:70%;
}

.clearfix:after {
  content:"";
  clear:both;
  display:table;
}
```



○ footer

```
footer {  
    padding: 20px;  
    text-align: center;  
    background: #ddd;  
}
```

header

- Home
- News
- Contact
- About

Title

Content

Title

Sub title1

Sub content1

Sub title2

Sub content2

Advertisement

Content

Advertisement

Content

footer



header

Home news Contact About

Title

Content

Title

Sub title1

Sub content1

Sub title2

Sub content2

Advertisement

Content

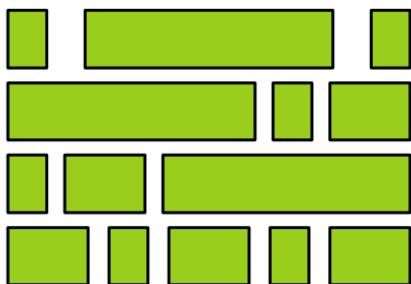
Advertisement

Content

footer

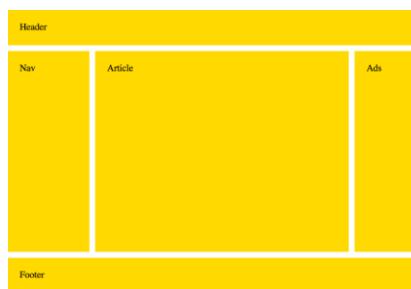
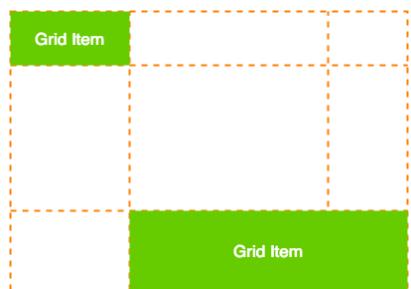
■ Further reading

○ FlexBox



```
#main > article {  
  flex: 1;  
}  
#main > nav,  
#main > aside {  
  flex: 0 0 20vw;  
  background: beige;  
}
```

○ Grid



```
grid-template-areas:  
  "header header header"  
  "nav article ads"  
  "footer footer footer";
```

○ BootStrap

○ Font Awesome

○ Web Font

○ @Media Query

5. JSON, XML

5.1. JSON

5.2. XML

5.1. JSON



■ What is JSON?

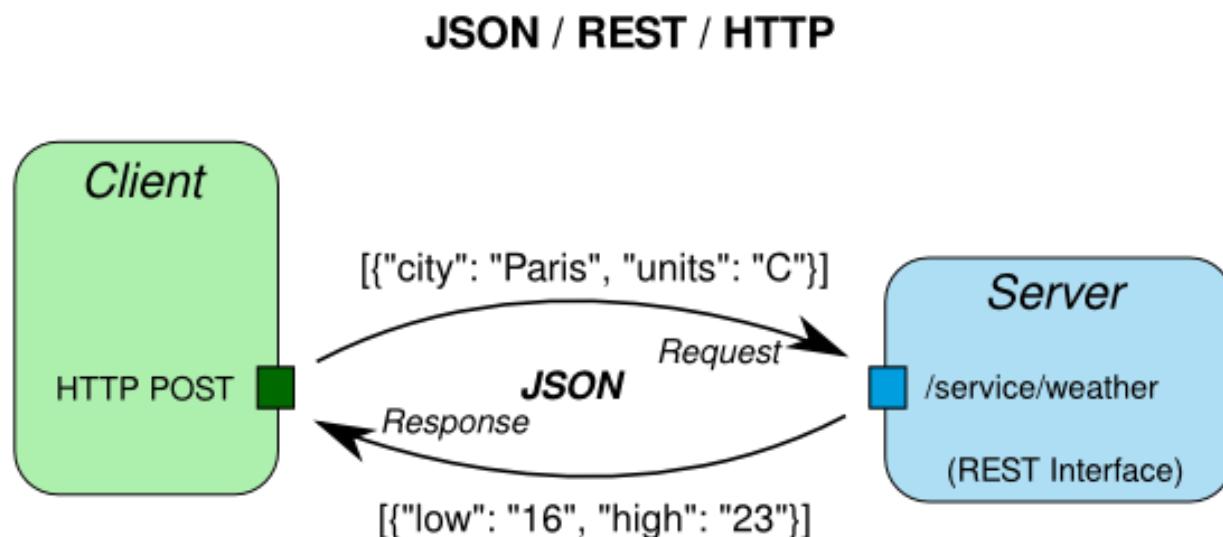
- JSON stands for JavaScript Object Notation
- Designed for human-readable data interchange
- Extended from the JavaScript scripting language
- JSON Internet Media type is **application/json**, extention is **.json**

■ Why use JSON?

- Used for serializing and transmitting structured data over network connection

■ Characteristics

- Easy to read and write
- Programming Language independent
- A lightweight text-based interchange format



■ 예제

```
{  
    "Make" : "Volkswagen",  
    "Year" : 2003,  
    "Model" : {  
        "Base" : "Golf",  
        "Trim" : "GL"  
    },  
    "Colors" : ["White", "Pearl", "Rust"],  
    "PurchaseDate" : "2006-10-05T00:00:00Z"  
}
```



- Easy processing

Javascript:

```
var car = { "Make" : "Volkswagen" };
console.log(car.Make);
// Output: Volkswagen

car.Year = 2003;
console.log(car);
// Output: { "Make" : "Volkswagen", "Year" : 2003 }
```

Storage size**XML: 225 Characters**

```
<Car>
  <Make>Volkswagen</Make>
  <Year>2003</Year>
  <Model>
    <Base>Golf</Base>
    <Trim>GL</Trim>
  </Model>
  <Colors>
    <Color>White</Color>
    <Color>Pearl</Color>
    <Color>Rust</Color>
  </Colors>
  <PurchaseDate>
    2006-10-05 00:00:00.000
  </PurchaseDate>
</Car>
```

JSON: 145 Characters

```
{
  "Make" : "Volkswagen",
  "Year" : 2003,
  "Model" : {
    "Base" : "Golf",
    "Trim" : "GL"
  },
  "Colors" :
    ["White", "Pearl", "Rust"],
  "PurchaseDate" :
    "2006-10-05T00:00:00.000Z"
}
```

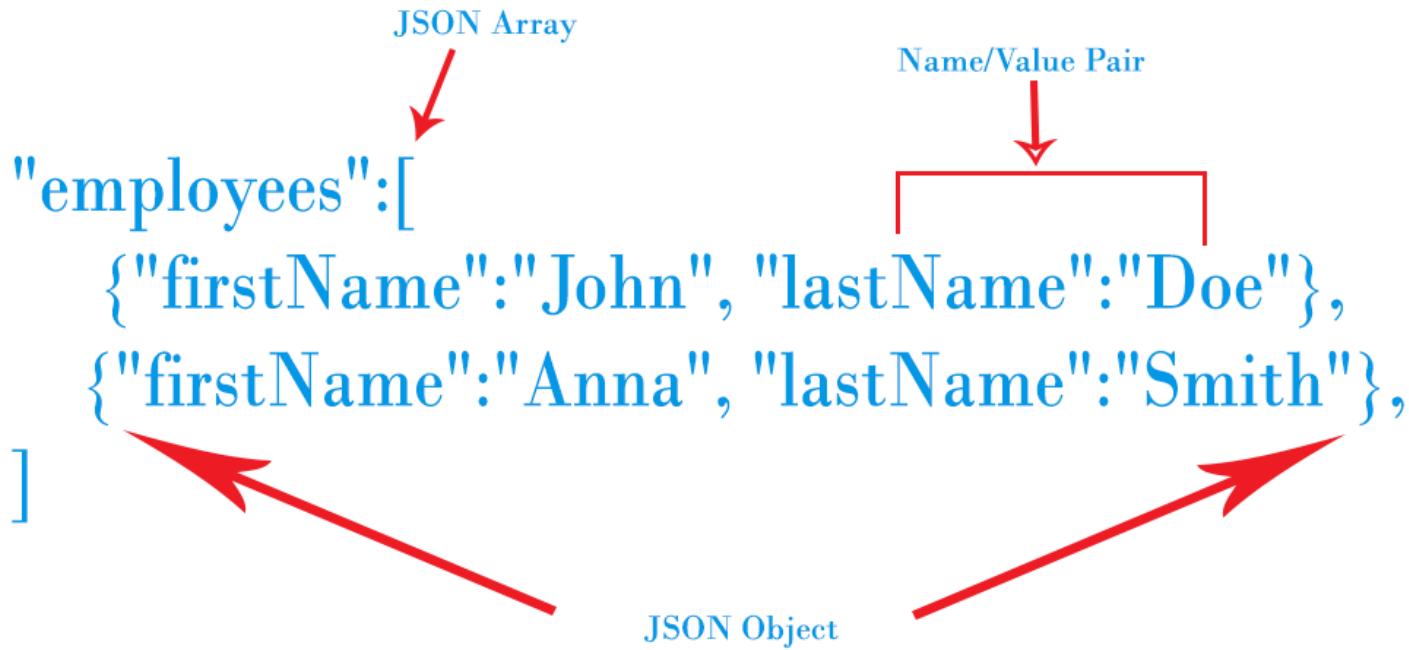
○ JSON API

{ json:api }

기상청에서 제공하는 Open-API 목록을 조회하고 활용 신청할 수 있도록 링크를 제공합니다.

번호	서비스 명	유형	자료포맷	등록일	조회수	상세보기
27	낙뢰정보 낙뢰정보조회서비스	REST	XML	2018-03-20	228	
26	태풍정보 태풍정보조회서비스	REST	JSON	2018-03-20	156	
25	동네예보통보문 동네예보통보문조회서비스	REST	JSON	2018-03-20	684	
24	항공기상전문 항공기상전문서비스	REST	XML	2018-03-20	82	
23	세계 주요공항 항공기상전문 세계 주요공항 항공기상전문 서비스	REST	XML	2018-03-20	23	

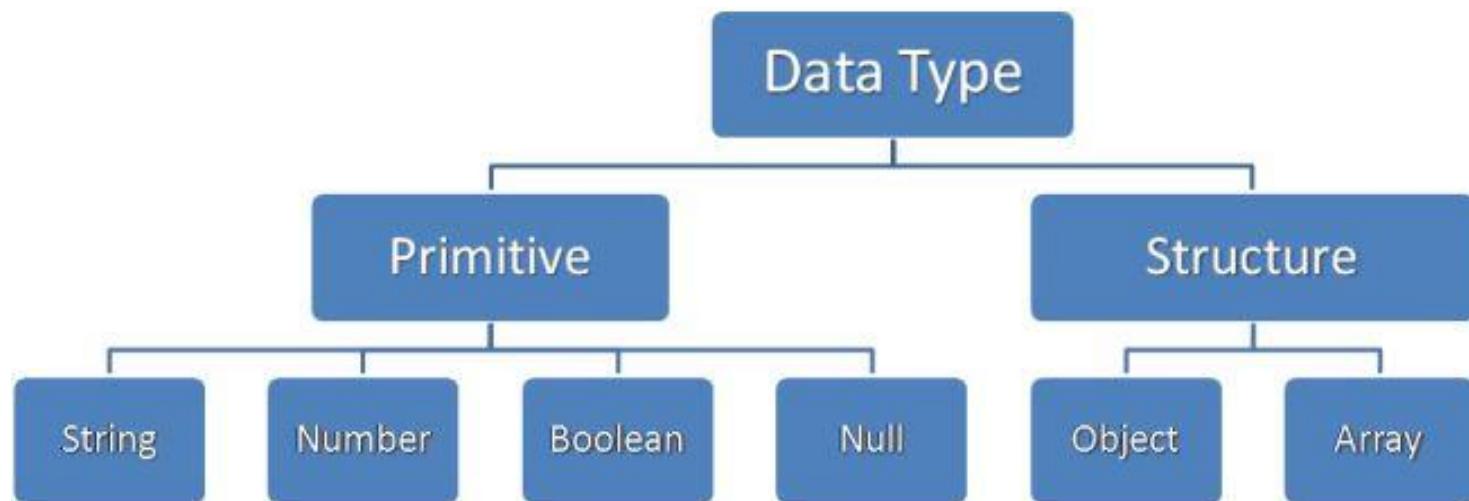
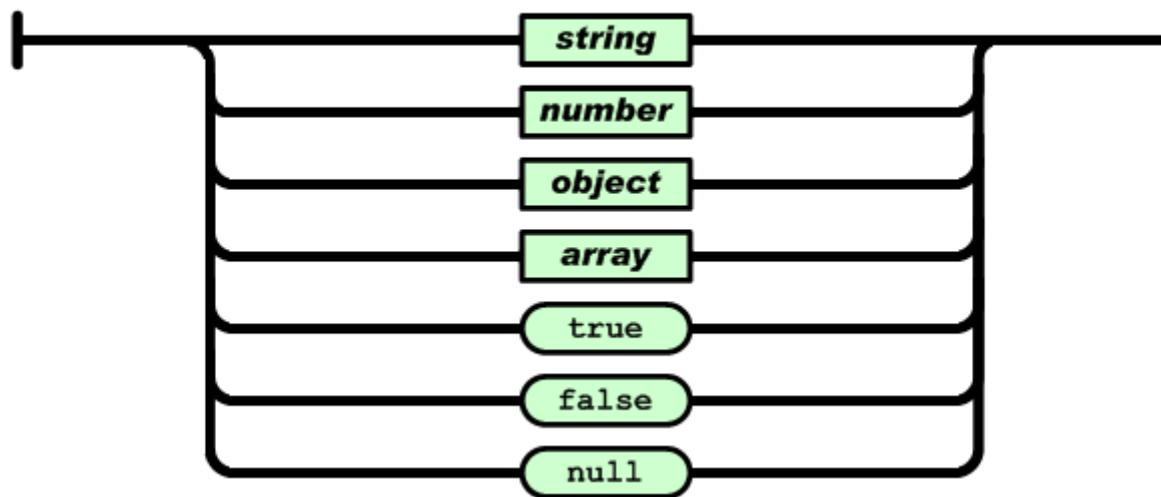
■ JSON Concept



- JSON Object
 - name, value pairs
- JSON Array
 - Ordered collections

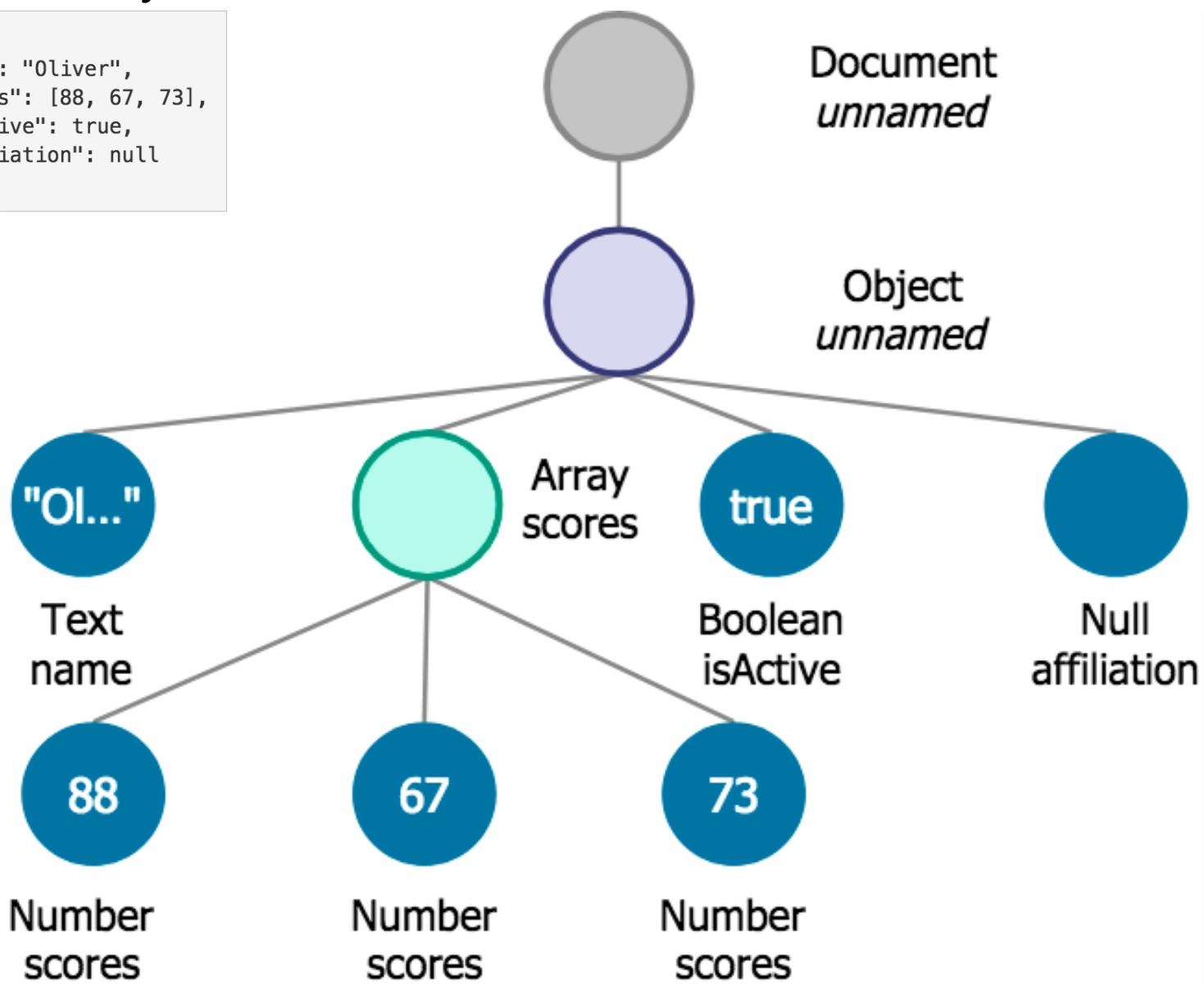
■ JSON Data Type

value



■ JSON Hierarchy

```
{  
  "name": "Oliver",  
  "scores": [88, 67, 73],  
  "isActive": true,  
  "affiliation": null  
}
```



■ 예제

○ JSON data types

- String
- Number
- Boolean
- Null
- Object
- Array

```
<script>
    var who = { "name" : "Kim" };
    var age = { "age" : 30 };
    var car = { "car" : true };
    var middle = { "middle name" : null };

    var personObj = { "name" : "kim", "age" : 30, "car" : true, "middle name" : null };

    var personList = { "list" : [ "Kim", "Lee", "Park" ] };
</script>
```

○ Accessing object

```
<script>
    var personObj = { "name" : "kim", "age" : 30, "car" : true, "middle name" : null };

    var name1 = personObj.name;
    var name2 = personObj["name"];

    console.log(name1, name2);
</script>
```

○ Looping an object

```
<script>
    var personObj = { "name" : "kim", "age" : 30, "car" : true, "middle name" : null };

    for (x in personObj) {
        console.log(x, personObj[x]);
    }
</script>
```

○ Nested objects

```
<script>
  var personObj = { "name" : "kim",
                    "age" : 30,
                    "car" : {
                      "car1" : "Hyundai",
                      "car2" : "Kia",
                      "car3" : "Chevrolet"
                    }
      };

  console.log(personObj.car.car1);
</script>
```

○ Modify values

```
<script>
  var personObj = { "name" : "kim",
                    "age" : 30,
                    "car" : {
                      "car1" : "Hyundai",
                      "car2" : "Kia",
                      "car3" : "Chevrolet"
                    }
      };

  personObj.car.car1 = "Mercedes";

  console.log(personObj.car.car1);
</script>
```

○ Delete object properties

```
<script>
  var personObj = { "name" : "kim",
                    "age" : 30,
                    "car" : {
                      "car1" : "Hyundai",
                      "car2" : "Kia",
                      "car3" : "Chevrolet"
                    }
      };

  delete personObj.car.car3;

  console.log(personObj.car);
</script>
```

○ Accessing array values

```
<script>
  var personObj = { "name" : "kim", "age" : 30,
                    "cars" : [ "Hyundai", "Kia", "Chevrolet" ] };

  console.log(personObj.cars[0]);
</script>
```

○ Looping through an array

```
<script>
  var personObj = { "name" : "kim", "age" : 30,
                    "cars" : [ "Hyundai", "Kia", "Chevrolet" ] };

  for (x in personObj.cars) {
    console.log(personObj.cars[x]);
  }

  for (i=0; i<personObj.cars.length; i++) {
    console.log(personObj.cars[i]);
  }
</script>
```

○ Nested arrays

```
<script>
  var personObj = { "name" : "kim", "age" : 30,
                    "cars" : [
                      { "name" : "Hyundai", "models" : [ "Avante", "Sonata", "grandeur" ] },
                      { "name" : "Kia", "models" : [ "K3", "K5", "K7" ] },
                      { "name" : "Chevrolet", "models" : [ "Cruze", "Malribu", "Impala" ] }
                    ] };

  for (i in personObj.cars) {
    console.log(personObj.cars[i].name);

    for (j in personObj.cars[i].models) {
      console.log(personObj.cars[i].models[j]);
    }
  }
</script>
```

○ Delete array items

- Shift, Pop, Splice

```
delete personObj.cars[2].models[0];
console.log(personObj.cars);

delete personObj.cars[2];

console.log(personObj.cars);
```

○ Stringify

- a common use of JSON is to exchange data to/from a web server
- when sending data to a web server, the data has to be a string

```
JSON.stringify(value[, replacer[, space]])
```

value – The JSON object to convert to a JSON string.

replacer – A function that alters the behavior of the stringification process. If this value is null or not provided, all properties of the object are included in the resulting JSON string.

space – A String or Number object that's used to insert white space into the output JSON string for readability purposes. If this is a Number, it indicates the number of space characters to use as white space.

- Javascript object

```
<script>
  var obj = { name : "Kim", "age" : 30, "car" : true };

  console.log(obj);
  console.log(JSON.stringify(obj));

  objStr = JSON.stringify(obj);

  console.log(obj.name);
  console.log(objStr.name);
</script>
```

➤ Javascript array

```
<script>
  var arr = [ "Kim", 30, true ];

  console.log(arr);
  console.log(JSON.stringify(arr));

  arrStr = JSON.stringify(arr);

  console.log(arr[0]);
  console.log(arrStr[0]);
</script>
```

➤ replacer, space

```
<script>
  var obj = { name : "Kim", "age" : 30, "car" : true };

  console.log(obj);
  console.log(JSON.stringify(obj));

  objStr = JSON.stringify(obj, ["name"]);

  console.log(obj);
  console.log(objStr);

  objStr = JSON.stringify(obj, function(k,v){if(v==="Kim")return;else return v;});

  console.log(obj);
  console.log(objStr);

  objStr = JSON.stringify(obj, null, 10/* ' ', '\t' */);

  console.log(obj);
  console.log(objStr);
</script>
```

○ Parse

- a common use of JSON is to exchange data to/from a web server
- When receiving data from a web server, the data is always a string

```
JSON.parse(text[, reviver])
```

text – The string to parse as JSON object.

reviver – A function that prescribes how the value originally produced by parsing is transformed, before being returned.

- parse

```
<script>
  var str = '{ "name" : "Kim", "age" : 30, "car" : ["Ray", "Spark"] }';

  console.log(str);
  console.log(str.name);

  obj = JSON.parse(str);

  console.log(obj);
  console.log(obj.name);
</script>
```

Reviver

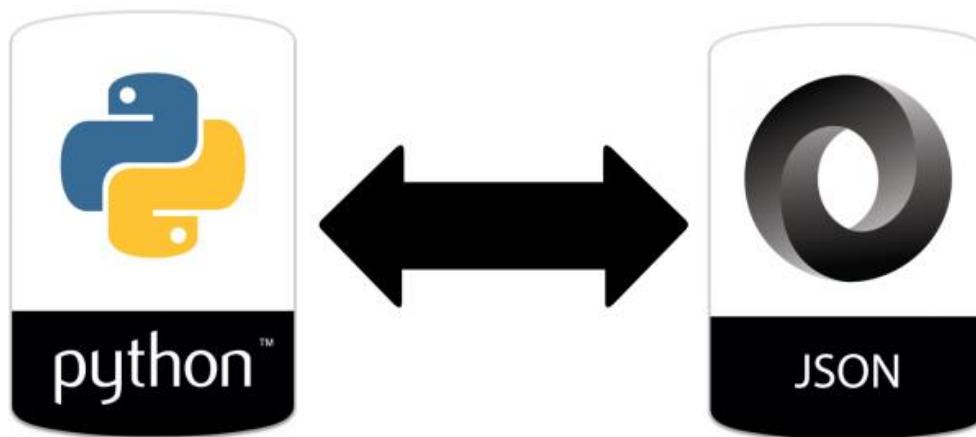
```
<script>
  var str = '{ "name" : "Kim", "age" : 30, "car" : [ "Ray", "Spark" ] }';

  console.log(str);
  console.log(str.name);

  obj = JSON.parse(str, function(k,v) {
    if (typeof(v) == "number")
      return v.toString();
    else
      return v; } );

  console.log(obj);
  console.log(obj.age);
</script>
```

■ JSON with Python



○ JSON library

```
import json
```

○ Encoding

- dump, dumps

○ Decoding

- load, loads

○ Data types

Python	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

○ dump / dumps

```
json.dump(obj, fp, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True,  
cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw)
```

```
json.dumps(obj, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True,  
cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw)
```

Serialize *obj* to a JSON formatted str using this conversion table. The arguments have the same meaning as in `dump()`.

```
name = ("Kim", "Lee", "Park")  
age = [30, 28, 31]  
person = {"name": "Kim", "age": 30, "car": False}
```

```
nameStr = json.dumps(name)  
ageStr = json.dumps(age)  
personStr = json.dumps(person, indent=" ")
```

```
kname = ("김", "이", "박")  
knameStr = json.dumps(kname)  
print(knameStr)  
  
knameStr = json.dumps(kname, ensure_ascii=False)  
print(knameStr)
```

○ load / loads

```
json.load(fp, *, cls=None, object_hook=None, parse_float=None, parse_int=None,  
parse_constant=None, object_pairs_hook=None, **kw)
```

```
json.loads(s, *, encoding=None, cls=None, object_hook=None, parse_float=None,  
parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)
```

Deserialize *s* (a `str`, `bytes` or `bytearray` instance containing a JSON document) to a Python object using this conversion table.

```
personObj = json.loads(personStr)  
person == personObj
```

○ With a file

```
person["car"] = ["레이", "모닝"]

with open("person.json", "w") as f:
    json.dump(person, f)
```

```
with open("person.json", "r") as f:
    personObj = json.load(f)

personObj
```

○ 예제

```
import json
import urllib.request

url = "http://ip.jsontest.com"

obj = {"name": "김이박", "age": 30}
objStr = json.dumps(obj)
objByte = objStr.encode("utf-8")

req = urllib.request.Request(url, data=objByte, headers={'content-type': 'application/json'})
res = urllib.request.urlopen(req)

resByte = res.read()
resStr = resByte.decode("utf-8")
resObj = json.loads(resStr)

print(resByte, type(resByte))
print(resStr, type(resStr))
print(resObj, type(resObj))
```

○ 전국 대기오염도 현황 Open API

➤ 서비스

- <https://www.data.go.kr/subMain.jsp#/L3B1YnlvcG90L215cC9Jcm9zTXIQYWdlL29wZW5EZXZHdWIkZVBhZ2UkQF4wMTIkQF5wdWJsaWNEYXRhRGV0YWlsUGs9dWRkaTo3MDkxMTBINy1kN2IxLTQ0MjEtOTBiYS04NGE2OWY5ODBjYWJfMjAxNjA4MDgxMTE0JEBeWFpbkZsYWc9dHJ1ZQ==>

➤ URL

- <http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInqireSvc/getMsrstnAcctoRltmMesureDnsty>

➤ Parameters

- stationName
- dateTerm = *daily* | *month* | *3month*
- pageNo
- numOfRows
- ver = *1.1* | *1.2* | *1.3* | *1.4*
- _returnType = *json*

○ 전국 대기오염도 현황 Open API 활용

```
url = "http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInqireSvc/getMsrstnAcctoRltmMesureDnsty"

params = {
    "serviceKey": "k6C4a%2FFJDFxsVHTVxI2p0%2F1ZYidISwqO6LPY9LSoqFoAB6AxIA9vn9eluB8j48P5h5xs9h04VEpS%2BpJbRgiXJQ%3D%3D",
    "numOfRows": 10,
    "pageSize": 10,
    "pageNo": 1,
    "startPage": 1,
    "stationName": "성북구",
    "dataTerm": "DAILY",
    "ver": "1.3",
    "_returnType": "JSON"
}

params[ "serviceKey" ] = urllib.parse.unquote(param[ "serviceKey" ])
params = urllib.parse.urlencode(params)
params = params.encode( "utf-8" )

req = urllib.request.Request(url, data=params)
res = urllib.request.urlopen(req)

resStr = res.read()

resStr = resStr.decode( "utf-8" )
resObj = json.loads(resStr)

resJSON = json.dumps(resObj, indent="  ")
print(resJSON)
```

5.2. XML



■ What is XML?

- XML stands for eXtensible Markup Language
- Designed to store and transport data
- Designed to be both human- and machine-readable
- XML is a W3C Recommendation

■ Why use XML?

- Universally accepted standard way of structuring data
- Provides a well-defined structure for communication
- Software- and hardware-independent tool for storing and transporting data

■ Characteristics

- Text-based (Unicode)
 - more readable, easier to document, easier to debug
- No predefined tags
 - the author must define both the tags and the document structure
- Extensible
 - supporting rich structure, like objects or hierarchies or relationships
 - most XML applications will work as expected even if new data is added (or removed).
- Validity
 - supporting validation and well-formed properties

■ Well-formed Documents

- An XML document with correct syntax is called **Well Formed**

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Tove</to>
    <From>Jani</from>
    <heading>Reminder</pheading>
    <body>Don't forget me this weekend!</body>
</note>
```

■ Valid Documents

- A **well formed** XML document is not the same as a **valid** XML document

- DTD - The original Document Type Definition
- XML Schema - An XML-based alternative to DTD

■ DTD

- define the structure of an XML document

```
<!DOCTYPE note SYSTEM "Note.dtd">
```

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

- With a DTD

- You can use a DTD to verify your own data
- You can also verify that the data you receive from the outside world is valid

- XML does not require a DTD/Schema

- when you are working with small XML files, creating DTDs may be a waste of time

■ XML Schema

- XML Schema Definition is an XML-based alternative to DTD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

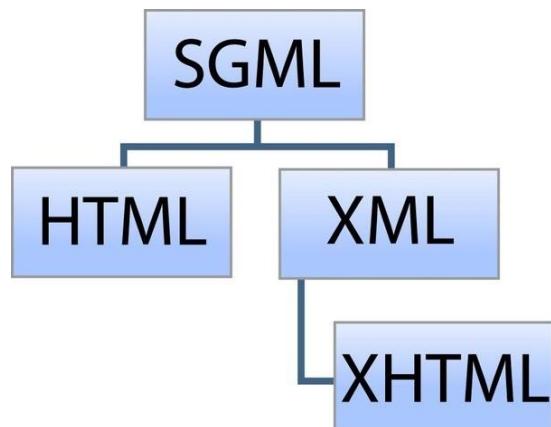
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

- XML Schemas are More Powerful than DTD

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

■ XML vs HTML



XML

```
<firstName>Maria</firstName>
<lastName>Roberts</lastName>
<dateBirth>12-11-1942</dateBirth>
```

HTML

```
<font size="3">Maria Roberts</font>
<b>12-11-1942</b>
```

○ Comparison

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

COMPARISON

XML

- Extensible set of tags
- Content orientated
- Standard Data infrastructure
- Allows multiple output forms
- Content and format can be placed together.

HTML

- Fixed set of tags
- Presentation oriented
- No data validation capabilities
- Single presentation
- Content and format are separate; formatting is contained in a style sheet.

■ XML vs JSON

- JSON is lightweight thus simple to read and write.
- JSON supports array data structure.
- JSON files are more human readable.
- JSON has no display capabilities .
- Provides scalar data types and the ability to express structured data through arrays and objects.
- Native object support.
- XML is less simple than JSON.
- XML doesn't support array data structure.
- XML files are less human readable.
- XML provides the capability to display data because it is a markup language.
- Does not provide any notion of data types. One must rely on XML Schema for adding type information.
- Objects have to be expressed by conventions, often through a mixed use of attributes and elements.

Similarities between JSON and XML

Both are simple and open.

Both supports unicode. So internationalization is supported by JSON and XML both.

Both represents self describing data.

Both are interoperable or language-independent.

■ XML vs JSON

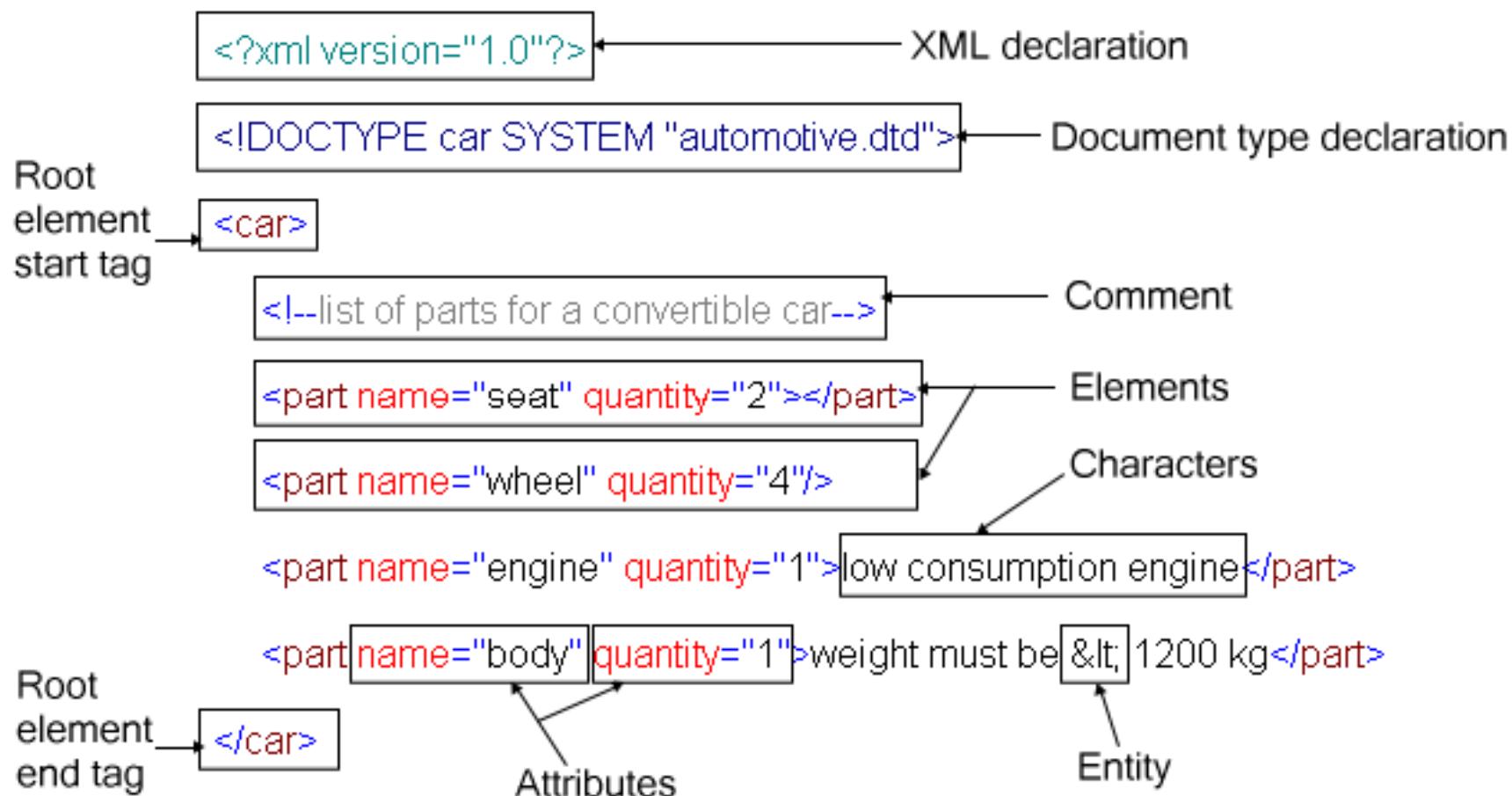
XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

JSON

```
{ "empinfo" :
  {
    "employees": [
      {
        "name": "James Kirk",
        "age": 40,
      },
      {
        "name": "Jean-Luc Picard",
        "age": 45,
      },
      {
        "name": "Wesley Crusher",
        "age": 27,
      }
    ]
  }
}
```

■ Syntax



○ XML Prolog

- The XML prolog is optional. If it exists, it must come first in the document
- UTF-8 is the default character encoding for XML documents

○ Closing Tag

- All elements **must** have a closing tag

○ Case Sensitive

- Opening and closing tags must be written with the same case

○ Properly Nested

- all elements **must** be properly nested within each other

```
<b><i>This text is bold and italic</i></b>
```

○ Quote

- the attribute values must always be quoted

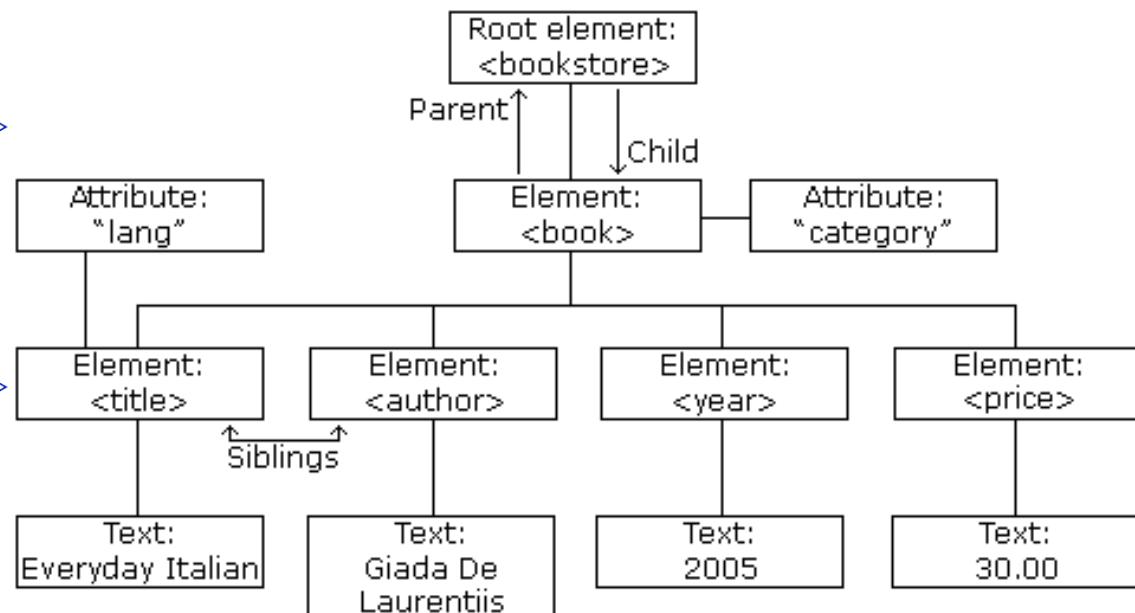
○ Entity References

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

■ XML Tree

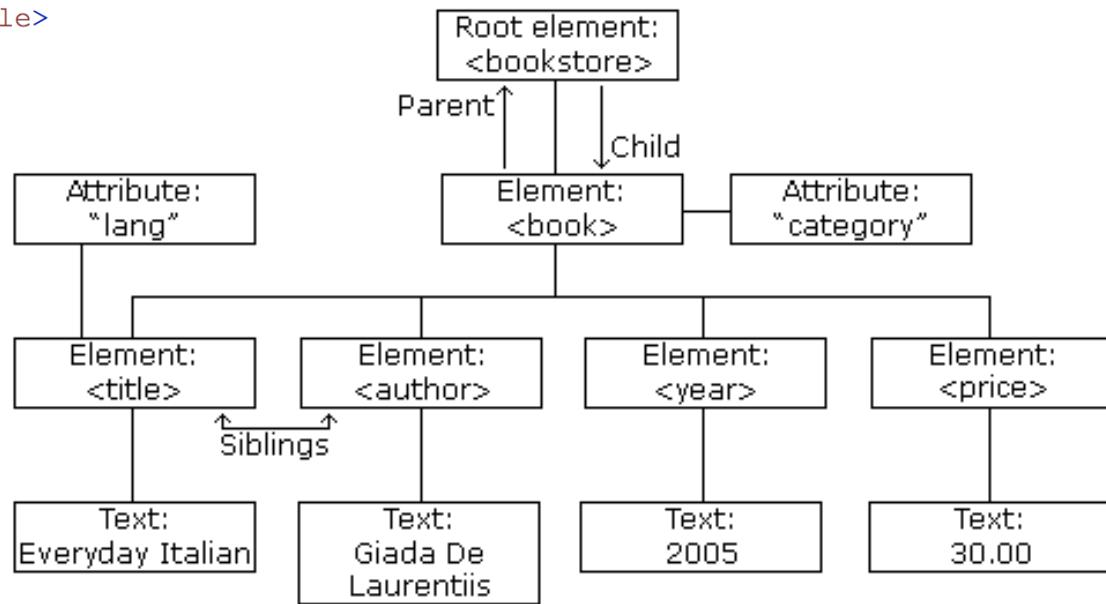
- XML documents are formed as element trees.
- An XML tree starts at a **root** element and branches from the root to **child** elements
- The terms **parent**, **child**, and **sibling** are used to describe the **relationships** between elements

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```



```

<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
  
```



■ XML Elements

- Everything from (including) the element's start tag to (including) the element's end tag
- Naming rules
 - Element names are case-sensitive
 - Element names must start with a letter or underscore
 - Element names cannot start with the letters xml (or XML, or Xml, etc)
 - Element names can contain letters, digits, hyphens, underscores, and periods
 - Element names cannot contain spaces

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

Avoid "-". If you name something "first-name", some software may think you want to subtract "name" from "first".

Avoid ". ". If you name something "first.name", some software may think that "name" is a property of the object "first".

Avoid ":". Colons are reserved for namespaces.

Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software doesn't support them.

■ XML Attributes

- Attributes are designed to contain data related to a specific element
- Attribute values must always be quoted. Either single or double quotes can be used

```
<person gender="female">
```

```
<gangster name='George "Shotgun" Ziegler'>
```

■ XML Elements vs Attributes

- There are no rules about when to use attributes or when to use elements in XML

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

■ Namespace

- XML Namespaces provide a method to avoid element name conflicts

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

HTML table information

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

information about a table (a piece of furniture)

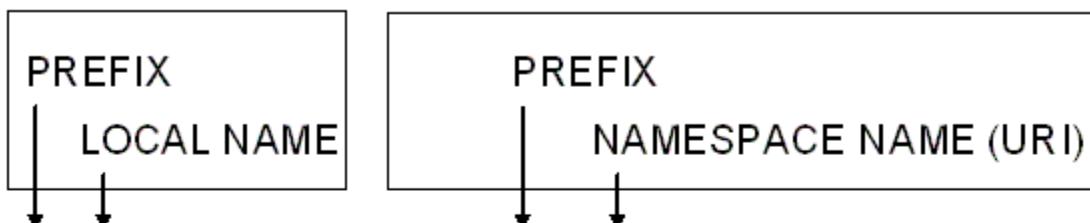
- Solving the name conflict

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

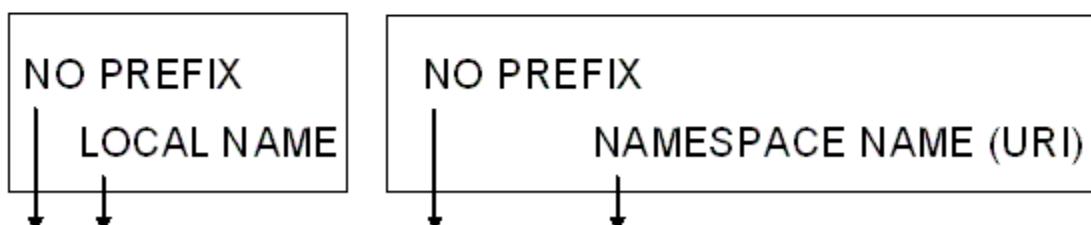
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

xmlns

- a namespace for the prefix must be defined
- the namespace can be defined by an xmlns attribute in the start tag of an element

QUALIFIED NAME “BK” NAMESPACE DECLARATION

<BK:BOOKSTORE XMLNS:BK="http://www.example.org/bookstore"/>

QUALIFIED NAME DEFAULT NAMESPACE DECLARATION

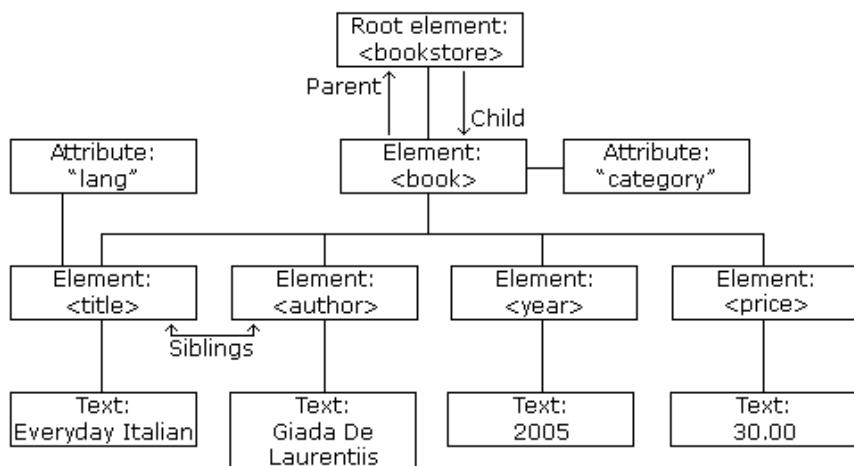
<BOOKSTORE XMLNS="http://www.example.org/bookstore"/>

■ DOM

- The DOM defines a standard for accessing and manipulating documents

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- A standard object model for XML
- A standard programming interface for XML
- Platform- and language-independent
- A W3C standard



```
getElementsByName("title")[0].childNodes[0].nodeValue
```

```

<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J. K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
  
```

○ 예제

```
<body>
<p id="demo"></p>

<script>
    var parser, xmlDoc;
    var text = "\n        <bookstore>\n            <book>\n                <title>Everyday Italian</title>\n                <author>Giada De Laurentiis</author>\n                <year>2005</year>\n            </book>\n        </bookstore>"\n\n    parser = new DOMParser();
    xmlDoc = parser.parseFromString(text,"text/xml");
\n\n    document.getElementById("demo").innerHTML =
    xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>
</body>
```

■ DOM Parser

○ Methods

- `x.getElementsByTagName(name)` - get all elements with a specified tag name
- `x.appendChild(node)` - insert a child node to x
- `x.removeChild(node)` - remove a child node from x

○ Properties

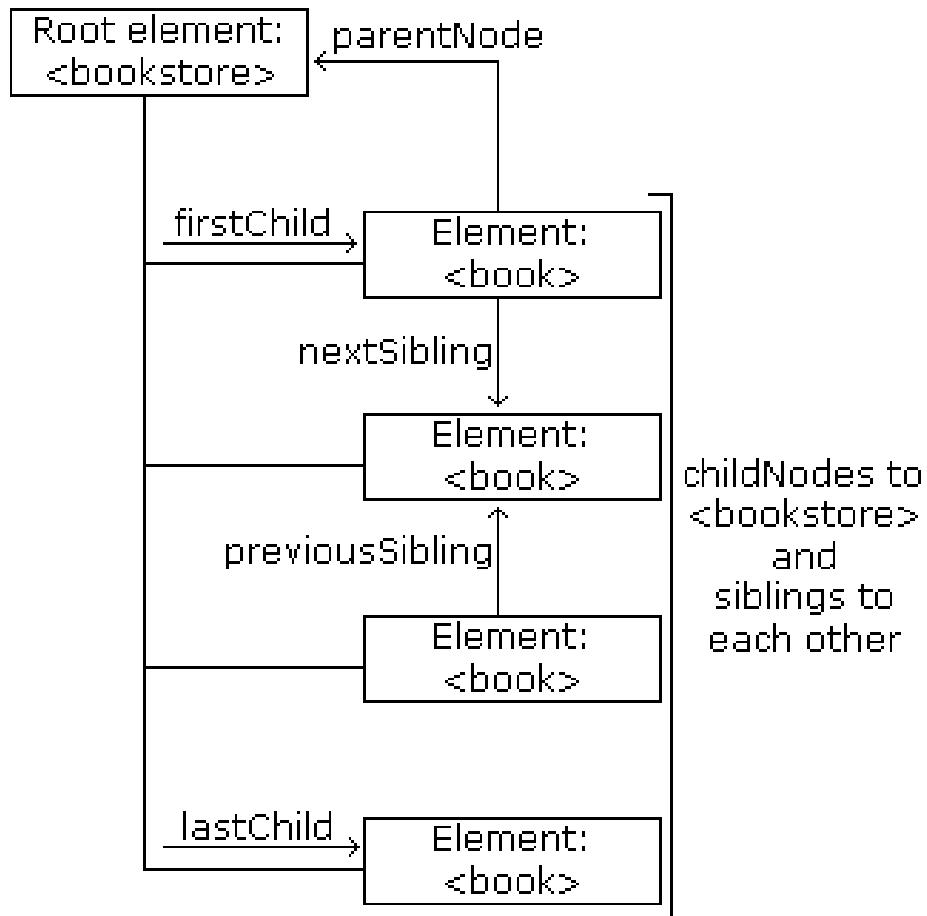
- `x.nodeName` - the name of x
- `x.nodeValue` - the value of x
- `x.parentNode` - the parent node of x
- `x.childNodes` - the child nodes of x
- `x.attributes` - the attributes nodes of x

○ 예제

```

<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>
  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>

```



○ Avoid empty text node

- Firefox, and some other browsers, will treat empty white-spaces or new lines as text nodes
- Internet Explorer will not

○ DOM Navigating

```
<script>
  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var xmlDoc = this.responseXML;

      bookList = xmlDoc.getElementsByTagName("book");

      console.log(bookList[0].parentNode.childNodes);

      console.log(bookList[0].parentNode);

      console.log(bookList[0].parentNode.firstChild.nextSibling);

      console.log(bookList[0].parentNode.childNodes[3]);

      console.log(bookList[0].parentNode.childNodes[3].previousSibling.previousSibling);

      console.log(bookList[0].parentNode.lastChild.previousSibling);
    }
  };

  xhttp.open("GET", "books.xml", true);
  xhttp.send();
</script>
```

○ DOM handling

```
<script>
    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var xmlDoc = this.responseXML;

            bookList = xmlDoc.getElementsByTagName("book");

            console.log(bookList[0].childNodes[1]);
            bookList[0].childNodes[1].firstChild.nodeValue = "Everyday Korean";
            console.log(bookList[0].childNodes[1]);

            bookList[0].setAttribute("category", "history");
            console.log(bookList[0].getAttribute("category"));

            console.log(bookList);
            xmlDoc.documentElement.removeChild(bookList[0]);
            console.log(bookList);

            newNode = xmlDoc.createElement("edition");
            newText = xmlDoc.createTextNode("first");
            newNode.appendChild(newText);
            console.log(newNode);

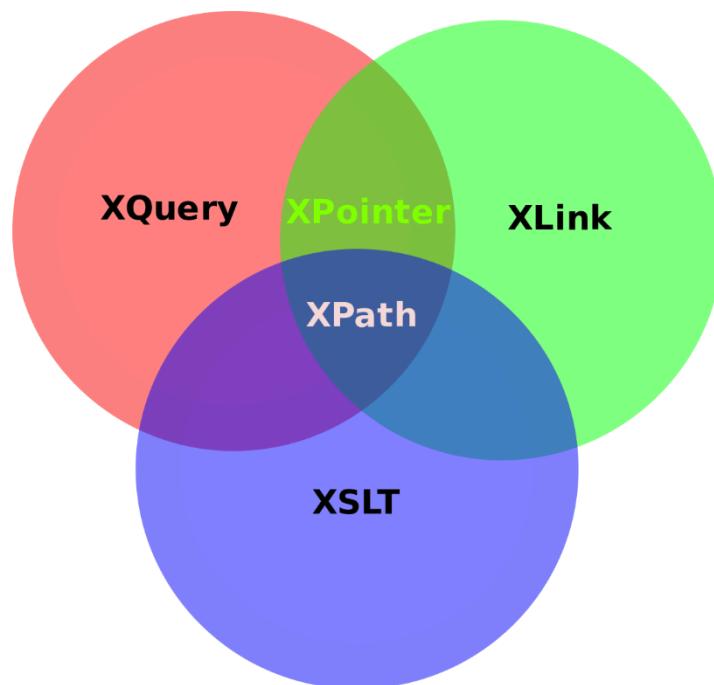
            bookList[0].appendChild(newNode);
            console.log(bookList[0]);

            console.log(bookList[0].lastChild);
        }
    };

    xhttp.open("GET", "books.xml", true);
    xhttp.send();
</script>
```

■ XPath

- XPath is a major element in the XSLT standard.
- XPath can be used to navigate through elements and attributes in an XML document
 - XPath is a syntax for defining parts of an XML document
 - XPath uses path expressions to navigate in XML documents
 - XPath contains a library of standard functions
 - XPath is a major element in XSLT and in XQuery
 - XPath is a W3C recommendation



○ Syntax

Expression	Description
<i>nodename</i>	Selects all nodes with the name " <i>nodename</i> "
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore
	Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

○ 예제

XPath Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

➤ Chrome, Firefox, Edge, Opera, and Safari

```
xmlDoc.evaluate(xpath, xmlDoc, null, XPathResult.ANY_TYPE,null);
```

○ 예제

```
<script>
  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var xmlDoc = this.responseXML;

      pathList = [ "/bookstore/book[1]",
        "/bookstore/book[last()]",
        "/bookstore/book[last()-1]",
        "/bookstore/book[position()<3]",
        "//title[@lang]",
        "//title[@lang='en']",
        "/bookstore/book[price>35.00]",
        "/bookstore/book[price>35.00]/title"];

      for (i=0; i<pathList.length; i++) {
        console.log((i+1), pathList[i]);

        var nodes = xmlDoc.evaluate(pathList[i], xmlDoc, null, XPathResult.ANY_TYPE, null);

        while (result = nodes.iterateNext()) {
          console.log(result);
        }
      }
    };
  };

  xhttp.open("GET", "books.xml", true);
  xhttp.send();
</script>
```

■ XML with Python

```
//input  
[@name='token']  
/@value
```



- XML library
 - <https://wiki.python.org/moin/PythonXml>

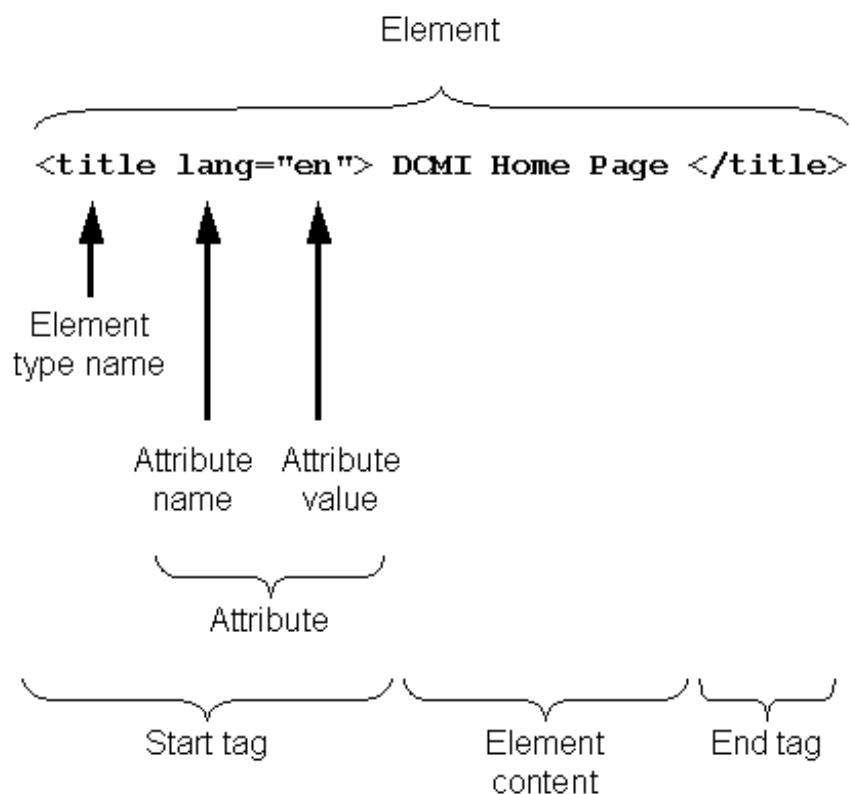
- xml

```
import xml.etree.ElementTree as et
```

- lxml

```
from lxml import etree
```

■ Building XML



○ Element

```
class xml.etree.ElementTree.Element(tag, attrib={}, **extra)
```

Element class. This class defines the Element interface, and provides a reference implementation of this interface.

The element name, attribute names, and attribute values can be either bytestrings or Unicode strings. *tag* is the element name. *attrib* is an optional dictionary, containing element attributes. *extra* contains additional attributes, given as keyword arguments.

➤ append

append(*subelement*)

Adds the element *subelement* to the end of this elements internal list of subelements.

➤ insert

insert(*index, element*)

Inserts a subelement at the given position in this element.

➤ set

set(*key, value*)

Set the attribute *key* on the element to *value*.

➤ get

get(*key, default=None*)

Gets the element attribute named *key*.

○ SubElement

xml.etree.ElementTree. SubElement(*parent, tag, attrib={}, **extra*)

Subelement factory. This function creates an element instance, and appends it to an existing element.

○ 예제

```
bookStore = et.Element("bookstore")
book1 = et.Element("book", category="cooking")
bookStore.insert(0, book1)

title1 = et.Element("title")
title1.attrib["lang"] = "en"
title1.text = "Everyday Italian"
book1.append(title1)

et.SubElement(book1, "author").text = "Giada De Laurentiis"
et.SubElement(book1, "year").text = "2005"
et.SubElement(book1, "price").text = "30.00"

book2 = et.Element("book", {"category": "children"})
bookStore.append(book2)

title2 = et.Element("title")
title2.attrib["lang"] = title1.get("lang")
title2.text = "Harry Potter"
book2.append(title2)

et.SubElement(book2, "author").text = "Giada De Laurentiis"
et.SubElement(book2, "year").text = "2005"
et.SubElement(book2, "price").text = "30.00"

et.dump(bookStore)
```

○ dump

`xml.etree.ElementTree.dump(elem)`

Writes an element tree or element structure to sys.stdout. This function should be used for debugging only.

■ Parsing

○ parse

```
xml.etree.ElementTree.parse(source, parser=None)
```

Parses an XML section into an element tree.

source is a filename or file object containing XML data.

parser is an optional parser instance. If not given, the standard `XMLParser` parser is used.

Returns an `ElementTree` instance.

○ XML, fromstring

```
xml.etree.ElementTree.XML(text, parser=None)
```

Parses an XML section from a string constant.

This function can be used to embed “XML literals” in Python code.

text is a string containing XML data.

parser is an optional parser instance. If not given, the standard `XMLParser` parser is used.

Returns an `Element` instance.

```
xml.etree.ElementTree.fromstring(text)
```

Parses an XML section from a string constant. Same as `XML()`.

text is a string containing XML data. Returns an `Element` instance.

○ Element

➤ getchildren

➤ items

items()

Returns the element attributes as a sequence of (name, value) pairs. The attributes are returned in an arbitrary order.

➤ keys

keys()

Returns the elements attribute names as a list. The names are returned in an arbitrary order.

➤ find

find(*match*)

Finds the first subelement matching *match*. *match* may be a tag name or path. Returns an element instance or `None`.

➤ findall

findall(*match*)

Finds all matching subelements, by tag name or path. Returns a list containing all matching elements in document order.

➤ findtext

findtext(*match*, *default=None*)

Finds text for the first subelement matching *match*. *match* may be a tag name or path.

Returns the text content of the first matching element, or *default* if no element was found.

Note that if the matching element has no text content an empty string is returned.

○ 예제

```
root = et.XML(et.tostring(bookStore))

# self.children
print(len(root))
for childNode in root:
    print(childNode.tag, childNode.attrib)

root.clear()

root = et.fromstring(et.tostring(bookStore))

# self.children, list(elem)
childNodes = root.getchildren()
print(len(childNodes))
for childNode in childNodes:
    print(childNode.tag, childNode.items())

for childNode in childNodes[0]:
    print(childNode.tag, childNode.keys())
    if childNode.keys() != []:
        print([childNode.get(k) for k in childNode.keys()])

book = root.find("book")
print(book.tag, book.get("category"))

bookList = root.findall("book")
for book in bookList:
    print(book.tag, book.get("category"))

title = root.find("./title")
print(type(title), title.text)

titleList = root.findall("./title")
print([title.text for title in titleList])

title = root.findtext("./title")
print(type(title), title)

book = root.find("./book[@category='children']")
print(book, book.tag)
```

■ XML File

○ write

```
write(file, encoding="us-ascii", xml_declaration=None, default_namespace=None, method="xml")
```

Writes the element tree to a file, as XML. *file* is a file name, or a file object opened for writing. *encoding* [1] is the output encoding (default is US-ASCII).

xml_declaration controls if an XML declaration should be added to the file.

Use `False` for never, `True` for always, `None` for only if not US-ASCII or UTF-8 (default is `None`).

default_namespace sets the default XML namespace (for “`xmlns`”).

method is either `"xml"`, `"html"` or `"text"` (default is `"xml"`). Returns an encoded string.

○ parse

```
parse(source, parser=None)
```

Loads an external XML section into this element tree

source is a file name or file object.

parser is an optional parser instance. If not given, the standard `XMLParser` parser is used.

Returns the section root element.

○ ElementTree

```
class xml.etree.ElementTree.ElementTree(element=None, file=None)
```

ElementTree wrapper class.

This class represents an entire element hierarchy,
and adds some extra support for serialization to and from standard XML.

○ 예제

➤ write

```
from xml.etree.ElementTree import ElementTree

tree = ElementTree(root)
tree.write("book_xml.xml", encoding="utf-8", xml_declaration="utf-8")
```

➤ parse

```
from xml.etree.ElementTree import parse

tree = parse("book_xml.xml")
root = tree.getroot()

for node in root.iter():
    print(node.tag, node.text)
```

➤ ElementTree

```
tree = ElementTree(file="book_xml.xml")
root = tree.getroot()

for node in root.iter():
    print(node.tag, node.text)
```

○ iter

iter(tag=None)

Creates a tree iterator with the current element as the root.

The iterator iterates over this element and all elements below it, in document (depth first) order.

If tag is not None or '*', only elements whose tag equals tag are returned from the iterator.

If the tree structure is modified during iteration, the result is undefined.

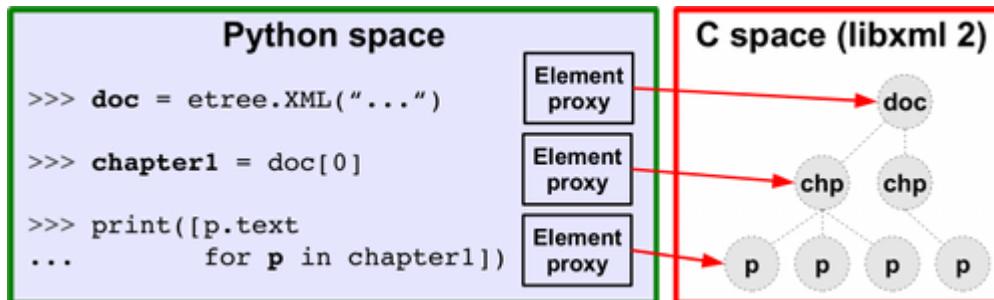
■ lxml



lxml - XML and HTML with Python

lxml is the most feature-rich and easy-to-use library for processing XML and HTML in the Python language.

- Being based on libxml2, lxml.etree holds the entire XML tree in a C structure



■ Element

`Element(tag, attrib=None, nsmap=None, **_extra)`

Element factory. This function returns an object implementing the Element interface.

An Element is the main container object for the ElementTree API. Most of the XML tree functionality is accessed through this class. Elements are easily created through the Element factory:

○ SubElement

`SubElement(parent, tag, attrib=None, nsmap=None, **_extra)`

Subelement factory. This function creates an element instance, and appends it to an existing element.

○ tostring

```
tostring(element_or_tree, encoding=None, method="xml", xml_declaration=None, pretty_print=False, with_tail=True, standalone=None, doctype=None, exclusive=False, with_comments=True, inclusive_ns_prefixes=None)
```

Serialize an element to an encoded string representation of its XML tree.

Defaults to ASCII encoding without XML declaration. This behaviour can be configured with the keyword arguments 'encoding' (string) and 'xml_declaration' (bool). Note that changing the encoding to a non UTF-8 compatible encoding will enable a declaration by default.

You can also serialise to a Unicode string without declaration by passing the `unicode` function as encoding (or `str` in Py3), or the name 'unicode'. This changes the return value from a byte string to an unencoded unicode string.

The keyword argument 'pretty_print' (bool) enables formatted XML.

The keyword argument 'method' selects the output method: 'xml', 'html', plain 'text' (text content without tags) or 'c14n'. Default is 'xml'.

The `exclusive` and `with_comments` arguments are only used with C14N output, where they request exclusive and uncommented C14N serialisation respectively.

Passing a boolean value to the `standalone` option will output an XML declaration with the corresponding `standalone` flag.

The `doctype` option allows passing in a plain string that will be serialised before the XML tree. Note that passing in non well-formed content here will make the XML output non well-formed. Also, an existing doctype in the document tree will not be removed when serialising an `ElementTree` instance.

You can prevent the tail text of the element from being serialised by passing the boolean `with_tail` option. This has no impact on the tail text of children, which will always be serialised.

○ tounicode

```
tounicode(element_or_tree, method="xml", pretty_print=False, with_tail=True, doctype=None)
```

Serialize an element to the Python unicode representation of its XML tree.

○ dump

```
dump(elem, pretty_print=True, with_tail=True)
```

Writes an element tree or element structure to `sys.stdout`. This function should be used for debugging only.

○ 예제

```
bookStore = etree.Element("bookstore")
book1 = etree.SubElement(bookStore, "book")
book2 = etree.SubElement(bookStore, "book", attrib={"category": "children"})
book1.attrib["category"] = "cooking"

title1 = etree.Element("title", lang="en")
title1.text = "Everyday Italian"
book1.append(title1)

etree.SubElement(book1, "author").text = "Giada De Laurentiis"
etree.SubElement(book1, "year").text = "2005"
etree.SubElement(book1, "price").text = "30.00"

title2 = etree.Element("title")
title2.set("lang", title1.get("lang"))
title2.text = "Harry Potter"
book2.append(title2)

etree.SubElement(book2, "author").text = "Giada De Laurentiis"
etree.SubElement(book2, "year").text = "2005"
book2.insert(3, etree.Element("price"))

print(len(book2))
book2[-1].text = "30.00"

xmlBytes = etree.tostring(bookStore, encoding="utf-8", pretty_print=True, xml_declaration=True)
xmlStr = etree.tounicode(bookStore, pretty_print=True)
print(type(xmlBytes), type(xmlStr))
etree.dump(bookStore)
```

■ Parsing

○ parse

`parse(source, parser=None, base_url=None)`

Return an ElementTree object loaded with source elements. If no parser is provided as second argument, the default parser is used.

The `source` can be any of the following:

- a file name/path
- a file object
- a file-like object
- a URL using the HTTP or FTP protocol

○ XML, fromstring

`XML(text, parser=None, base_url=None)`

Parses an XML document or fragment from a string constant. Returns the root node (or the result returned by a parser target). This function can be used to embed "XML literals" in Python code, like in

`fromstring(text, parser=None, base_url=None)`

Parses an XML document or fragment from a string. Returns the root node (or the result returned by a parser target).

To override the default parser with a different parser you can pass it to the `parser` keyword argument.

The `base_url` keyword argument allows to set the original base URL of the document to support relative Paths when looking up external entities (DTD, XInclude, ...).

○ ElementTree

```
ElementTree(element=None, file=None, parser=None)
ElementTree wrapper class.
```

➤ 예제

```
xml = etree.XML(etree.tostring(bookStore))
xmlTree = etree.ElementTree(xml)
xmlRoot = xmlTree.getroot()

print(xmlTree.docinfo.xml_version)
print(xmlTree.docinfo.encoding)
print(xmlTree.docinfo.doctype)
print(xmlTree.docinfo.root_name)

print(len(xmlRoot))
for childNode in xmlRoot:
    print(childNode.tag, childNode.attrib)
```

○ 예제

```
xml = etree.fromstring(etree.tostring(bookStore))
xmlTree = etree.ElementTree(xml)
xmlRoot = xmlTree.getroot()

childNodes = xmlRoot.getchildren()

print(len(childNodes))
for childNode in childNodes:
    print(childNode.tag, childNode.items())

for childNode in childNodes[0]:
    print(childNode.tag, childNode.keys())
    if childNode.keys() != []:
        print([childNode.get(k) for k in childNode.keys()])

book = xmlRoot.find("book")
print(book.tag, book.get("category"))

bookList = xmlRoot.findall("book")
for book in bookList:
    print(book.tag, book.get("category"))

title = xmlRoot.find("./title")
print(type(title), title.text)

titleList = xmlRoot.findall("./title")
print([title.text for title in titleList])

title = xmlRoot.findtext("./title")
print(type(title), title)

book = xmlRoot.find("./book[@category='children']")
print(book, book.tag)

for childNode in xmlRoot.iter():
    print(childNode.tag, childNode.text)

for childNode in xmlRoot.iter("book"):
    print(childNode.tag, childNode.text)
```

○ 예제

➤ write

```
xmlTree.write("book_tree.xml")
etree.ElementTree(xmlRoot).write("book_root.xml")
```

➤ parse

```
xmlTree = etree.parse("book_tree.xml")
xmlRoot = xmlTree.getroot()

etree.dump(xmlRoot)

xmlTree = etree.parse("book_root.xml")
xmlRoot = xmlTree.getroot()

etree.dump(xmlRoot)
```

○ 전국 대기오염도 현황 Open API

➤ 서비스

- <https://www.data.go.kr/subMain.jsp#/L3B1YnlvcG90L215cC9Jcm9zTXIQYWdlL29wZW5EZXZHdWlkZVBhZ2UkQF4wMTIkQF5wdWJsaWNEYXRhRGV0YWlsUGs9dWRkaTo3MDkxMTB1Ny1kN2IxLTQ0MjEtOTBiYS04NGE2OWY5ODBjYWJfMjAxNjA4MDgxMTE0JEBeWFpbkZsYWc9dHJ1ZQ==>

➤ URL

- <http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInqireSvc/getMsrstnAcctoRltmMesureDnsty>

➤ Parameters

- stationName
- dateTerm = *daily* | *month* | *3month*
- pageNo
- numOfRows
- ver = *1.1* | *1.2* | *1.3* | *1.4*
- _returnType = *json*

○ 전국 대기오염도 현황 Open API 활용

```
#resStr = resStr.decode("utf-8")
xmlObj = etree.fromstring(resStr)
xmlRoot = etree.ElementTree(xmlObj).getroot()

etree.dump(xmlRoot)

for node in xmlRoot.iter():
    print(node.tag, node.text)

itemList = xmlRoot.findall("./item")

print(len(itemList))
for item in itemList:
    for i in range(len(item)):
        print(item[i].tag, item[i].text)

pm25List = xmlRoot.findall("./item/pm25Value")
for item in pm25List:
    print(item.tag, item.text)
#    print(item.get(i).tag)
#    etree.dump(item.get(i))
```