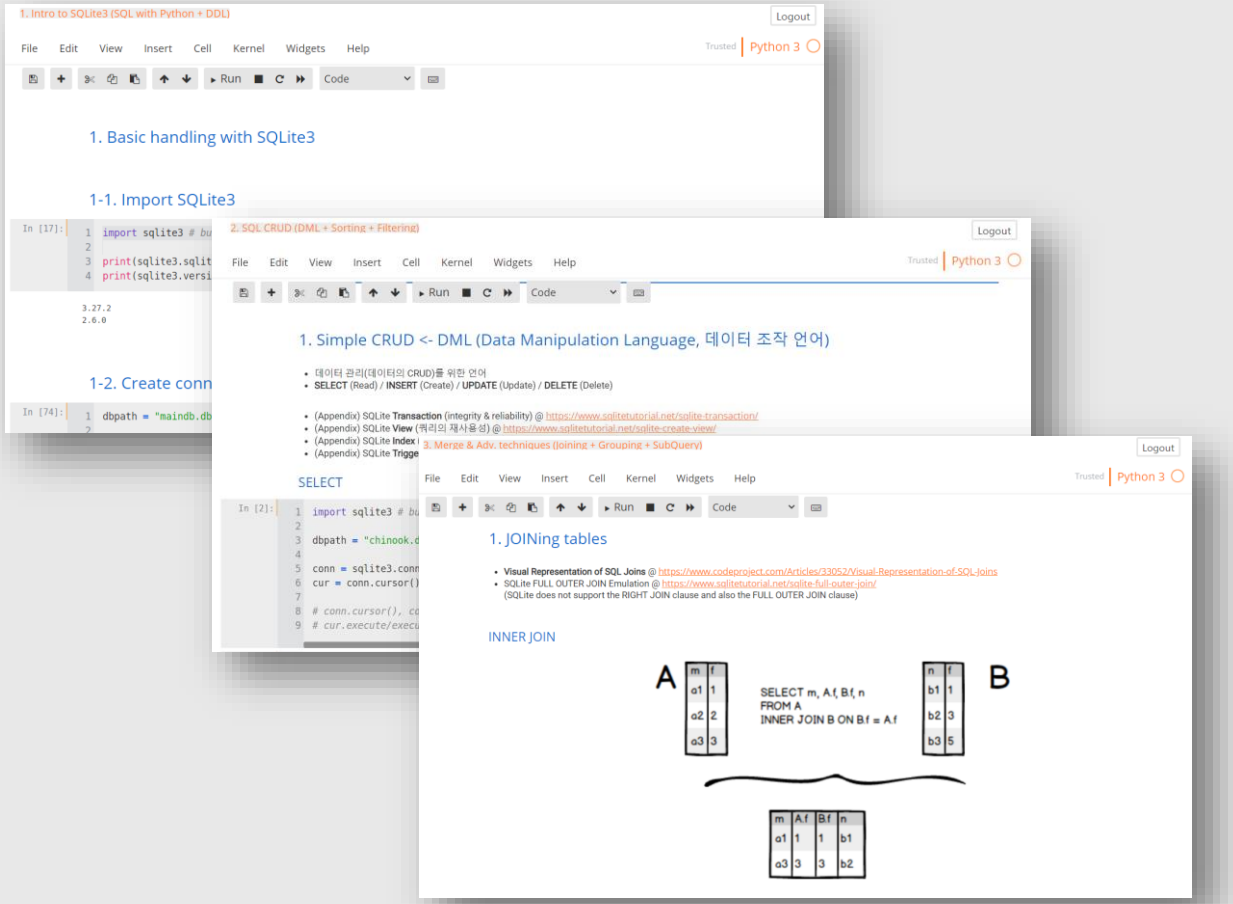
Part 4-2.

Python 기반 SQL 프로그래밍

goorm

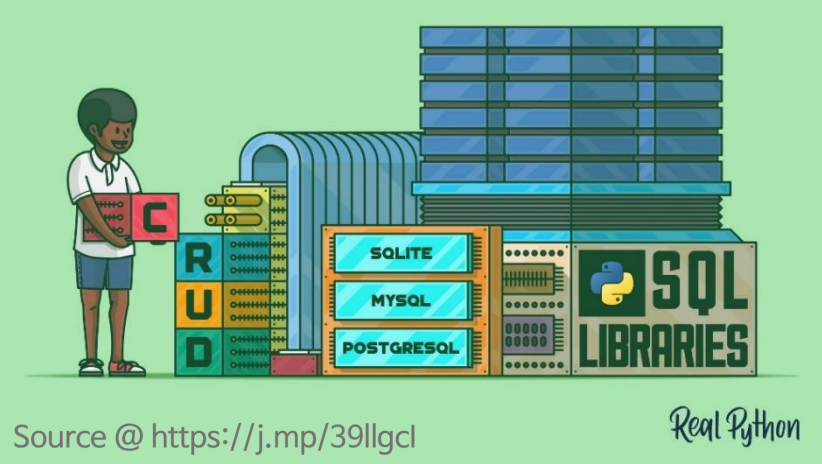


- **DBMS / 3-Steps of Data Modeling / SQL**
- **Intro to SQLite3 with Python** (SQL with Python + DDL)
- **SQL CRUD** (DML + Sorting + Filtering)
- **Merge & Adv. techniques** (Joining + Grouping + SubQuery)
- (Appendix) Crawling & saving data with Selenium+SQLite

* **Data Warehouse** vs **Data Lake** @ http://j.mp/2s6sVdP + **ELT** vs **ETL** @ https://j.mp/3huaSDH
* RDBMS(Relational Database Management System) 이해 @ https://j.mp/2No4YIO
* 7 Database Paradigms @ https://j.mp/3oaRmzc
* DB-Engines Ranking @ https://j.mp/3jmqGsz

DBMS (DataBase Management System)

- 하드웨어에 저장된 데이터베이스를 관리해주는 소프트웨어
- H(ierachical)DBMS인 IMS가 1960년대 최초로 출시된 후, 계층형/네트워크형/관계형/객체형 등으로 발전해 왔음
- 현재는 **관계형 데이터베이스**를 다루는 **R(elational)DBMS**가 널리 사용되고 있음
  ex) Oracle, MySQL(MariaDB), SQLite, Microsoft SQL Server(MS SQL), PostgreSQL 등

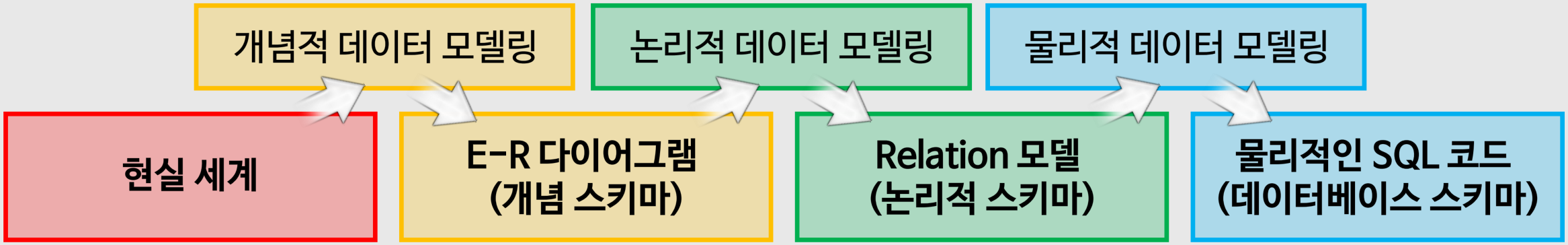


Source @ https://j.mp/39llgcl

Rank			DBMS	Database Model
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Document, Multi-model
6.	6.	6.	IBM Db2	Relational, Multi-model
7.	7.	7.	Elasticsearch	Search engine, Multi-model
8.	8.	8.	Redis	Key-value, Multi-model
9.	9.	↑ 11.	SQLite	Relational
10.	10.	10.	Cassandra	Wide column

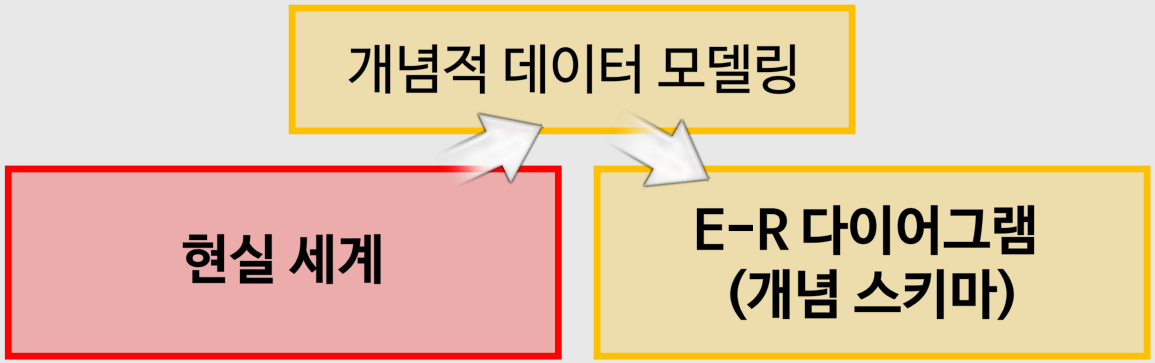
* E-R Model - 개념적 설계 @ https://j.mp/30sMEl4
* [DB이론] DB 설계 개요 @ https://j.mp/2CtZWWt

데이터 모델링 3단계



1. 개념적 데이터 모델링

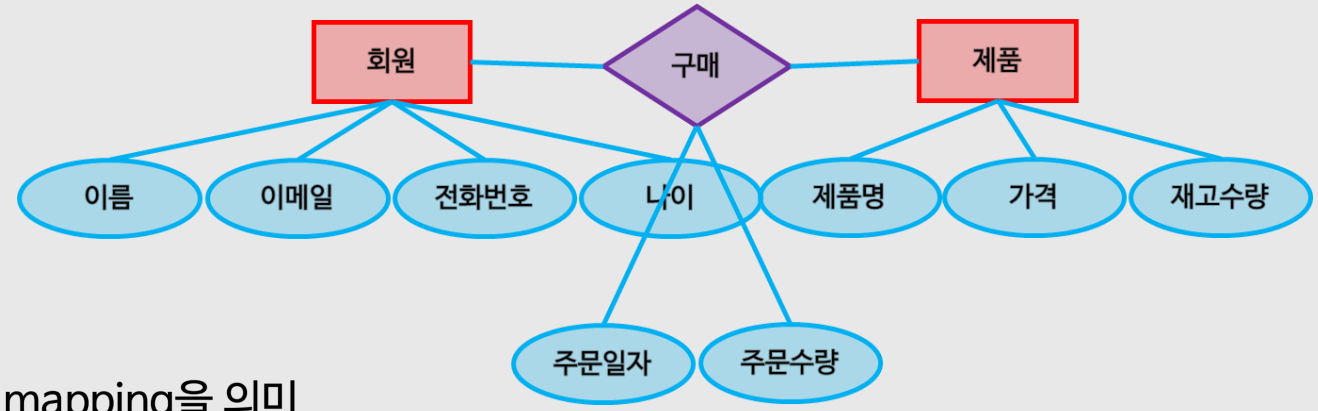
현실 세계로부터 개체(Entity)를 추출하고,
개체들 간의 관계를 정의하여,
E(ntity)-R(elationship) 다이어그램을 만드는 과정



개체 (Entity, 사각형) : 회원, 제품
저장할 가치가 있는 중요 데이터를 가진 사람이나 사물 개념 등

속성 (Attribute, 타원) : 이름, 이메일, 전화번호, 나이, …
의미 있는 데이터의 가장 작은 논리적 단위

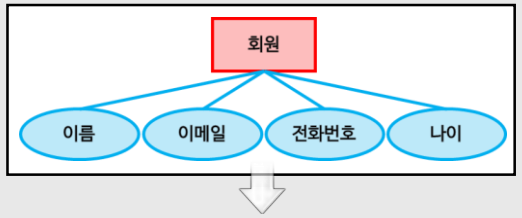
관계 (Relationship, 마름모) : 구매
개체와 개체 사이의 연관성 및 개체 집합 사이의 대응 관계 즉 매핑mapping을 의미



* Tidy data vs Messy data @ https://j.mp/39nDpqp & https://j.mp/3ePYNFP
* 관계 모델 (Relation Model): 논리적 설계 @ https://j.mp/2CuLkWO

2. 논리적 데이터 모델링

E-R 다이어그램을 바탕으로,
데이터베이스에 저장할 수 있는
논리적인 구조를 Relation 모델로 표현하는 과정



member ← **릴레이션 (Relation)** : 개체에 대한 데이터를 2차원 테이블 구조로 표현한 것 (Relationship도 포함)

회원 ID	이름	이메일	전화번호	나이
1	김철수	kim...@...	010-…	25
2	이수연	lee...@...	010-…	32
3	최지인	choi…@...	010-…	22

속성 (Attribute) : 열 (Column) == 필드 (Field)

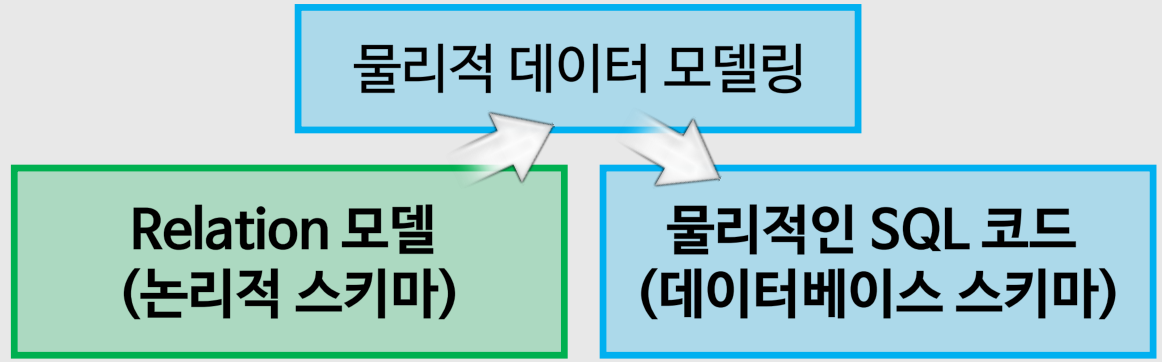
(1, 김철수, kim...@..., 010-…, 25)
튜플 (Tuple) : 행 (Row) == 레코드 (Record)
== 인스턴스 (Instance)

+ **차수 (Degree)** : 릴레이션 내 속성(column)의 총 개수 (위 예시에서 5)
+ **카디널리티 (Cardinality)** : 릴레이션 내 튜플(row)의 총 개수 (위 예시에서 3)

* 웹 기반 ERD 툴 & **SQL 자동 생성** 프로그램, **AQueryTool** @ https://goo.gl/V815Qx
* 웹 기반 ERD &DB 모델링 도구, **ERDCloud** @ https://j.mp/3FpgxGu

3. 물리적 데이터 모델링

Relation 모델을 DBMS의 종류에 따라
실제 물리 저장 장치에 저장할 수 있는
물리적 구조(ex. SQL)로 구현하는 과정



회원 : 회원번호, 이름, 이메일, 전화번호, 성별

```
CREATE TABLE member (
  memid INT PRIMARY KEY,
  name VARCHAR(40),
  email VARCHAR(40),
  phone VARCHAR(40),
  age INT
);
```

제품 : 제품번호, 제품명, 가격, 재고수량

```
CREATE TABLE product(
  proid INT PRIMARY KEY,
  proname VARCHAR(40),
  price INT,
  stock INT
);
```

구매 : 구매번호, 회원번호(FK), 제품번호(FK), 주문일자, 주문수량

```
CREATE TABLE order(
  orderid INT PRIMARY KEY,
  memid INT,
  proid INT,
  orderdate TIMESTAMP,
  quantity INT,
  FOREIGN KEY(memid) REFERENCES member(memid),
  FOREIGN KEY(proid) REFERENCES product(proid)
);
```

회원 : 회원번호, 이름, 이메일, 전화번호, 나이
제품 : 제품번호, 제품명, 가격, 재고수량
구매 : 구매번호, 회원번호(FK), 제품번호(FK), 주문일자, 주문수량

SQL (Structured Query Language, 구조적 질의 언어)

- 관계형 데이터베이스 시스템 (RDBMS)에서 데이터를 관리 & 처리하기 위해 만들어진 언어
- SQL의 표준으로 ANSI SQL이 정립
- 여러 DBMS 프로그램에서 ANSI SQL을 기반으로 개발된 자체적인 SQL을 사용하며 서로 다소간의 차이를 갖고 있음

DDL (Data Definition Language, 데이터 정의 언어)

- 각 릴레이션(데이터베이스 테이블)을 정의하기 위해 사용하는 언어
- CREATE (테이블 생성) / ALTER (테이블 변경) / DROP (테이블 삭제) 등

DML (Data Manipulation Language, 데이터 조작 언어)

- 데이터 관리(데이터의 CRUD)를 위한 언어
- SELECT (Read) / INSERT (Create) / UPDATE (Update) / DELETE (Delete)

DCL (Data Control Language, 데이터 제어 언어)

- 사용자 관리 & 사용자별 권한(릴레이션 및 데이터에 대한 관리/접근)을 다루기 위한 언어
- GRANT (권한 부여) / REVOKE (권한 해제)

Tips for SQL programming

* SQL Joins Visualizer @ http://j.mp/2HYqmi1 + 생활코딩 SQL JOIN @ http://j.mp/2I4rvo2
* 50 SQL Query Questions and Answers for Practice @ https://j.mp/3hWaPzu
* **공공 데이터를 이용한 상권 분석** (with **SQLite3**) @ https://j.mp/3Auu793
* 데이터분석, 먹고 들어가기 위한 SQL 공부법 @ https://j.mp/3ahJQKn
* SQL **SELECT 쿼리 실행 순서 & 처리 과정** @ https://j.mp/3amc0Wi
* How to Become a SQL Expert @ https://j.mp/373B1Gk

1. SQL은 기본적으로 **대소문자를 가리지 않음***

2. SQL 명령은 **반드시 세미콜론으로 끝**나야 함

3. **고유의 값은 따옴표로** 감싸줌

   ex) SELECT * FROM member WHERE name = 'Jone';

4. 한 줄 주석 : **--** ~~~, 여러 줄 주석 : **/*** ~~~ ***/**

   ex) -- SELECT * FROM member;

   ex) /*
   SELECT * FROM member
   WHERE memid=10;
   */

5. 반복을 통해 아래 키워드들의 순서를 익히기

```
SELECT ~~~
FROM ~~~
INNER JOIN ~~ ON ~.~ = ~.~
WHERE ~~~
GROUP BY ~~~
HAVING ~~~
ORDER BY ~~~
LIMIT ~ OFFSET ~
```

* 서버 환경 혹은 DBMS 종류에 따라 데이터베이스 이름 또는 필드 이름에 대해 대소문자를 구분하기도 함

Details of SQL

1. WHERE 절에 조건으로 사용할 수 있는 연산자

분류	연산자	예
비교	=, <>, <, <=, >, >=	WHERE milliseconds < 250
범위	BETWEEN, NOT BETWEEN	WHERE total BETWEEN 14.91 AND 18.86
집합	IN, NOT IN	WHERE mediatypeid IN (1, 4, 5)
패턴	LIKE, NOT LIKE	WHERE name LIKE '%Wild%'
NULL	IS NULL, IS NOT NULL	WHERE composer IS NULL
복합조건	AND, OR, NOT	WHERE (milliseconds < 250) AND (name LIKE '%Wild%')

Details of SQL

* SQL 검색 조건 조합 및 패턴 매칭 (그 외 SQL 관련 포스팅 다수) @ https://j.mp/39yGlAR
* Various wildcard(%, _) examples for SQLite3 LIKE @ https://j.mp/3eVeOdy
* SQLite String Functions (LENGTH, UPPER, ||, etc.) @ https://j.mp/39lXPA8

2. 문자열 와일드카드의 종류

와일드카드	매칭되는 문자열 패턴	문자열 패턴 예시
%	0개 이상의 문자열과 일치	'%wild%' : wild를 포함하고 **앞 뒤에 문자열이 0개 이상** 있는 문자열
[ ]	1개의 문자와 일치	'[0-8] %' : 0~8 중 숫자 **하나로 시작하는** 문자열
[^]	1개의 문자와 불일치	'[^0-9] %' : 0~9 사이 숫자 **하나로 시작하지 않는** 문자열
_	특정 위치의 1개의 문자와 일치	'%Br_wn%' : Br 과 wn 사이에 **1개의 문자열이 존재**하는 문자열
+	문자열을 연결	'test' + 'string' : 'test string'

Details of SQL

* SQLite **Date** Functions (DATE, TIME, DATETIME, STRFTIME, etc) @ https://j.mp/3ePTINL
* SQLite **Aggregate** Functions (AVG, COUNT, SUM, etc.) @ https://j.mp/2ZRxuqA
* SQLite Window Functions (**Ranking** & Value) @ https://j.mp/3fW8CUf

3. Aggregation function(집계 함수)의 종류

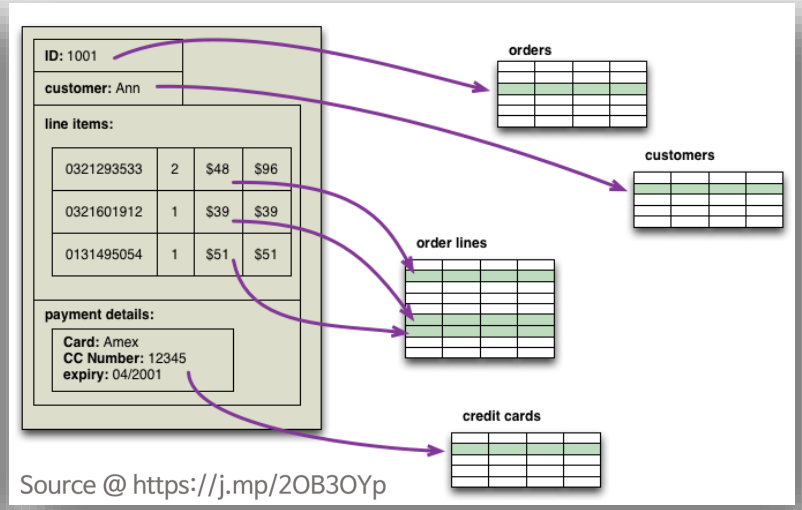
Agg. Func.	계산 종류	예시
SUM	합계	SELECT SUM(milliseconds) AS length,
AVG	평균	SELECT AVG(milliseconds) FROM tracks
COUNT	개수	SELECT COUNT(InvoiceId) InvoiceCount FROM invoices
MAX/MIN	최대값/최소값	SELECT MAX(milliseconds) FROM tracks
ROUND	반올림	SELECT ROUND(AVG(milliseconds), 2) FROM tracks

* MongoDB Data Modeling in Production @ https://goo.gl/m8vnto + MongoDB 강좌(문서) @ https://goo.gl/ee2Lau
* NoSQL Databases tutorial (Tabular/Document/Key-value/Graph) @ https://j.mp/3lyFHKD
* NoSQL 데이터 모델과 모델링 절차 @ https://j.mp/2Wyrykj
* NoSQL 간단한 소개 @ https://j.mp/308eD9w

NoSQL ("Not only SQL")

- 대부분 오픈소스 / **관계형 모델을 사용하지 않음 / 명시적인 스키마가 없음** (Column, Datatype 등 X -> Higher flexibility)
- 대용량 데이터의 클러스터 내 멀티 노드 분산 저장에 특화 (higher scalability and fault tolerance)
- 크게 4가지 종류가 존재 (**Key-Value / Document / Wide Column / Graph**)
- **NoSQL에서 처리하기 어려운 5가지 작업 : Sorting / Join / Grouping / Range Query / Index**

Rank			DBMS	Database Model
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Document, Multi-model
6.	6.	6.	IBM Db2	Relational, Multi-model
7.	7.	7.	Elasticsearch	Search engine, Multi-model
8.	8.	8.	Redis	Key-value, Multi-model
9.	9.	↑ 11.	SQLite	Relational
10.	10.	10.	Cassandra	Wide column



Source @ https://j.mp/2OB3OYp

- RDB : Oracle, MySQL(MariaDB), SQLite, Microsoft SQL Server(MS SQL), PostgreSQL, etc.
- No-SQL : **MongoDB, ElasticSearch, Redis, Cassandra, HBase, Amazon DynamoDB**, etc.
+ **NoSQL과 RDB는 대립되는 관계가 아님** (상황과 목적에 따라 2가지 종류의 DB를 모두 쓸 수 있음)

End of Document