Introduction to SQL

Prof. Hyuk-Yoon Kwon
https://sites.google.com/view/seoultech-bigdata

Today's Lecture

1. SQL introduction & schema definitions
2. Basic single-table queries
3. Multi-table queries

What you will learn about in this section

1. What is SQL?
2. Basic schema definitions
3. Keys & constraints intro
4. ACTIVITY: CREATE TABLE statements

1. SQL Introduction & Definitions - SQL Motivation

- **Dark times 5 years ago.**
  - Are databases dead?

- **Now, as before: everyone sells SQL**
  - Pig, Hive, Impala
- **“Not-Yet-SQL?”**

Big Data (generated from various Source) : significatn factor brough changes field

NoSQL (NotOnly SQL) DB :Sclability Data & Unstructed Data Processing
-> DB type X Rely on Relational DB Model (Traditional)
-> Use various Data Model to store & manage data
-> oftenly for Large-scale App & App require Scalability / Flexibility
-> designed to be Horizontally Scalable
-> Add New Server to Cluster = Easily handle Large Data & Traffic
EX) MongoDB, Cassandra, Couchbase, Redis ...

DB system determine: how the required data should be processed
SQL role : communicate
main law : communicate with others
X handle Big data -> NOSQL : needed another system
Query : 1 of powerful features of DB

SQL ?
- expressive power, easily learn
- can specify what we want to find/store/how processed by DB
- X have to consider details

Big Data (Various Source에서 생성) : DB 분야를 변화시킨 중요한 요인
-> 기존 DB System으로 처리하기 힘든 대량의 Data를 다루는 기술
=> 새로운 Data 처리 기술 & 방법론 필요성 대두

NoSQL (NotOnly SQL) DB :비정형 Data 처리 & 확장성 문제 해결 위해 개발
-> 관계형 DB Model (전통) 의존 X 인 DBMS 유형
-> 다양한 Data Model로 Data 저장 & 관리
-> 대규모 Web Ap & 확장성 / 유연성 필요한 App 에 자주 쓰임
-> 수평적으로 확장 가능한 설계
-> 새로운 Server를 Cluster에 추가 = 대량의 Data & Traffic 쉬운 처리 가능
EX) MongoDB, Cassandra, Couchbase, Redis ...

Basic SQL - SQL Introduction

SQL stands for
Structured **Q**uery **L**anguage

SQL is a standard language for querying and manipulating data

Acts as an Interface between human & DB systems.

SQL is a very high-level programming language

- This works because it is optimized well!

people X have enough information can run

Many standards out there:

- ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), ….
- Vendors support various subsets

provides Effeciency of calling processing of Query

Syntaxes are different since provides advanced feature to compared to others
=> We need a Standard

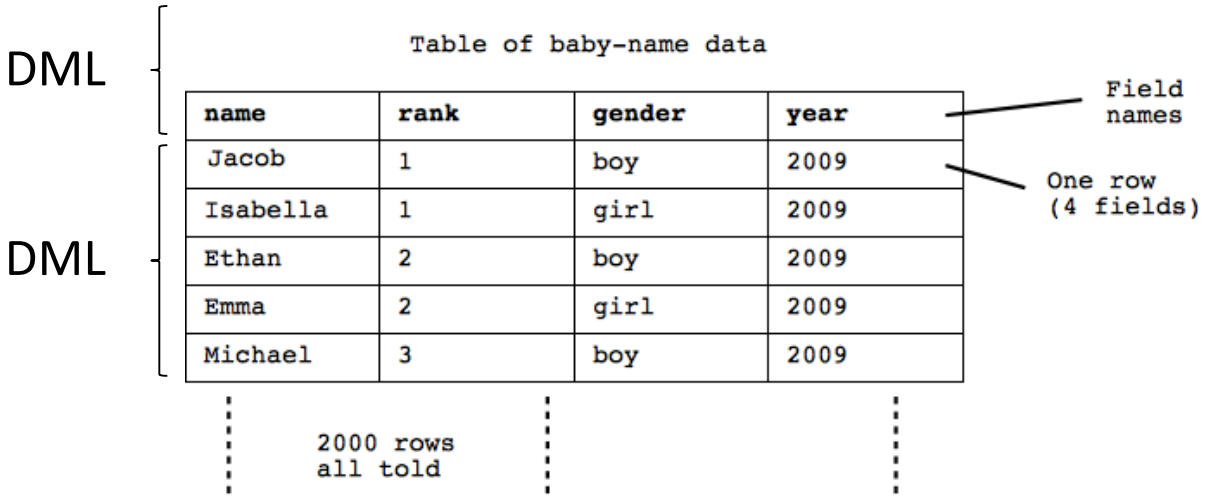
SQL is a...

Data Definition Language (DDL)

- Define relational *schemata = modify*
- Create/alter/delete tables and their attributes

Data Manipulation Language (DML)

- Insert/delete/modify tuples in tables = control
- Query one or more tables



DDL : DB Schema 정의 / 변경 명령어
-> DB 구조 변경
-> New Data Type/Table/View/Index ... 생성 시 사용

DDL : Define / Modify DB Schema
-> use for Alter Structure of DB
-> Create New Data Type/Table/View/Index ...
EX) CREATE, ALTER, DROP ...

DML : Data 조작 명령어
-> DB Record 검색 / 변경
-> New Data 삽입 / 삭제 시 사용

DML : Manipulate Data
-> use for Search / Modify DB Record
-> Insert / Delete New Data
EX) SELECT, INSERT, UPDATE, DELETE ...

Data Types in SQL

Study more: https://dev.mysql.com/doc/refman/5.7/en/char.html

- **Atomic types:**
  - Characters: CHAR(20), VARCHAR(50)
  - Numbers: INT, BIGINT, SMALLINT, FLOAT
  - Others: MONEY, DATETIME, …

- **Every attribute must have an atomic type**
  - Hence tables are flat

Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Relational DB Design Law

-> 관계형 DB에서 2차원적 Table/ 중첩 / 계층적 Data 구조 포함 X

-> Table의 각 행이 단일 개체 / Record 나타냄

-> 각 열이 해당 개체의 특정 속성 / 특성 나타냄

-> Table : Flat & Simple

=> Data 일관성 유지 , Query & 분석 용이하게 함, DB 성능 향상

An **attribute** (or **column**)

: typed data entry present in each tuple in the relation

*Attributes must have an **atomic** type in standard SQL,*

i.e. not a list, set, etc.

Attribute (Column) = 속성 (열) : DB 내 특정 Entity 특성 / 속성

-> Text, Number, Date, Boolyean .. 같은 특정 유형의 Data 저장 시 사용

ex) 직장 ? 이름, 주소, 번호, 이메일, 입사일, 직급 …

-> DB 내 Data를 쉽게 정렬/ 검색/ 조작할 수 있게 함

-> 서로 다른 Entity 간 관계 설정 시 사용 가능

(Data 일관성 & 정확성 보장에 도움됨)

Attribute (Column)

: Characterstic / Property of a Particular Entity

-> used to store specific types of Text, Number, Date, Boolyean .. values

-> allow easily Sort/ Search/ Manipulate data

-> can be used to establish Relationship between different Entities

(helps to ensure data Consistency & Accurancy)

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

The number of tuples is the **cardinality** of the relation

The number of attributes is the **arity** of the relation

*Also referred to sometimes as a **record***

A **relation** or **table**

: a multiset of tuples having the attributes specified by the schema

A **multiset**

1) an unordered list
2) a set with multiple duplicate instances allowed

A **tuple** or **row**

: a single entry in the table having the attributes specified by the schema

Cardinality =Table 간 수학적 관계

: 하나의 Table 내 Record 발생 수

& 다른 Table 내 Record 발생 수 간의 관계 지정

1) 일대일 (1 : 1)
2) 일대다 (1 : N)
3) 다대다 (N : M)

List [1, 1, 2, 3] : Ordered

Set {1, 2, 3} : X Ordered

Multiset {1, 1, 2, 3}

1) X Ordered
2) Allow Duplicated instance

Tuple (or Row)

-> 각 Row는 고유한 식별자 가짐

-> Table 각 열에 해당하는 여러 Attribute Value 포함

=> Table에서 필요한 정보 쉽게 검색 & 필요한 계산 수행 가능

ex) 특정 고객에 대한 정보 포함

Record = Table에 저장된 하나의 행(row)

-> Table은 여러 개의 Record로 구성

-> 각 Record는 Table Column에 대한 값 포함

ex) 각 학생에 대한 정보 = 한 개의 Record

Record는 이름, 학번, 학과 등의 정보를 각각 Column에 저장

-> Table에서 중요한 Data 단위

-> Data 검색, 수정, 삭제 등의 작업에 사용

-> Primary Key에 의해 보통 식별되며 Table 간 관계 표현 시에도 사용

*Also referred to sometimes as a **record***

Table Schemas

- **The schema of a table is the table name, its attributes, and their types:**

  Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

- **A key is an attribute whose values are unique; we underline a key**

  Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

- **To say "don't know the value" we use NULL**
  - NULL has (sometimes painful) semantics, more detail later

Students(sid:string, name:string, gpa: float)

Say, Jim just enrolled in his first class.

In SQL, we may constrain a column to be NOT NULL, e.g., "name" in this table

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

General Constraints

- **We can actually specify arbitrary assertions**
  - E.g. "*There cannot be 25 people in the DB class*"
- **In practice, we don't specify many such constraints. Why?**
  - Performance!
  - Whenever we do something ugly (or avoid doing something convenient) it's for the sake of performance

Summary of Schema Information

- **Schema and Constraints are how databases understand the semantics (meaning) of data**
- **They are also useful for optimization**
- **SQL supports general constraints:**
  - Keys and foreign keys are most important