

database-3

☰ 태그	
🕒 날짜	@2023년 6월 8일

SQL (Structured Query Language)

- 현업에서 쓰이는 relational DBMS의 표준언어
- 종합적인 database 언어 : DDL + DML + VDL

SQL 주요용어

relational data model	SQL
relation	table
attribute	column
tuple	row
domain	domain

SQL에서 relation 이란?

- `multiset`(= bag) of tuples @ SQL
- 중복된 tuple을 허용한다.

SQL & RDBMS

SQL은 RDBMS의 표준 언어이지만 실제 구현에 강제가 없기 때문에 RDBMS마다 제공하는 SQL의 스펙이 조금씩 다르다

예제를 통해 SQL을 DB정의하기

IT회사 관련 RDB만들기

- 부서, 사원, 프로젝트 관련 정보들을 저장할 수 있는 관계형 데이터베이스를 만들자
- 사용할 RDBMS는 MySQL(InnoDB)

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.00 sec)
```

```
CREATE DATABASE name(설정할 이름);  
- database를 새로 생성할 때 사용
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| company |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)
```

```
mysql> SELECT database();
- null 일 경우 아직 어떤 데이터 베이스를 사용하겠다고 설정하지 않은 상태
- 어떤 database가 선택되어 있는지 확인하는 명령어
```

```
mysql> SELECT database();
+-----+
| database() |
+-----+
| NULL |
+-----+
1 row in set (0.00 sec)
```

```
mysql> USE company; // USE를 통해 사용할 database를 선택
Database changed // 사용db가 변경되었을때 출력
아래 사진처럼 null이 아닌 company가 활성화
```

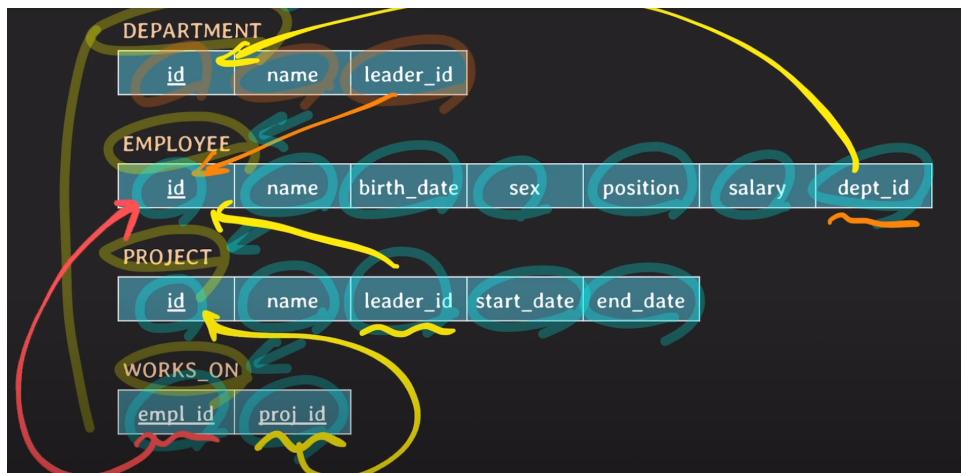
```
mysql> SELECT database();  
+-----+  
| database() |  
+-----+  
| company |  
+-----+  
1 row in set (0.00 sec)
```

```
mysql> DROP DATABASE company;  
mysql> DROP DATABASE db이름;  
database를 지울때 사용
```

DATABASE vs SCHEMA

- MySQL에서는 DATABASE와 SCHEMA가 같은 뜻을 의미
- CREATE DATABASE company = CREATE SCHEMA company
- 다른 RDBMS에서는 의미가 다르게 쓰임
- PostgreSQL에서는 SCHEMA가 DATABASE의 namespace를 의미

table 정의하기



DEPARTMENT에 id는 primary key().

leader_id는 employee의 id를 참조한다.

dept_id는 DEPARTMENT에 id를 참조한다.

project의 leader_id는 employee의 id를 참조

터미널에서 표시되는 형태

```
mysql>create table DEPARTMENT(
    -> id INT PRIMARY KEY,
    -> name VARCHAR(20) NOT NULL UNIQUE,
    -> leader_id INT
    ->);

->는 라인이 변경됨을 표시
```

각각의 라인은 attribute를 생성,
attributename datatype primarykey라면 설정

attribute data type : 숫자

종류	설명	사이즈	MySQL
정수	정수를 저장할 때 사용	1 byte	TINYINT
		2 byte	SMALLINT
		3 byte	MEDIUMINT
		4 byte	INT or INTEGER
		8 byte	BIGINT
부동 소수점 방식 (floating-point)	- 실수(real number)를 저장할 때 사용	4 byte	FLOAT
	- 고정 소수점 방식에 비해 정확하지 않다	8 byte	DOUBLE or DOUBLE PRECISION
고정 소수점 방식 (fixed-point)	- 실수를 정확하게 저장할 때 사용	variable	DECIMAL or NUMERIC
	- DECIMAL(5, 2) => [-999.99 ~ 999.99]	variable	

정수 : size 별로 1~8byte 타입마다 각각의 이름이 있다.

고정 소수점 : 돈처럼 정확하게 저장되어야 하는 경우 고정 소수점 방식이 좋음.

mysql에서는 decimal 과 numeric이 동일하다

decimal(precision(전체숫자에서의 몇자리인지, 위의 경우 총5자리 숫자),

scale(소수점 몇자리까지 표시할거냐, 위의 경우 소수점2번째 자리까지))

attribute data type : 문자열

종류	설명	MySQL
고정 크기 문자열	<ul style="list-style-type: none"> - 최대 몇 개의 '문자'를 가지는 문자열을 저장할지를 지정 - 저장될 문자열의 길이가 최대 길이보다 작으면 나머지를 space로 채워서 저장 - name char(4) 일 때 다음과 같이 저장 : 'a █', '한국 █', '고고고고', 'wow █' 	CHAR(n) (0 <= n <= 255)
가변 크기 문자열	<ul style="list-style-type: none"> - 최대 몇 개의 '문자'를 가지는 문자열을 저장할지를 지정 - 저장될 문자열의 길이 만큼만 저장 - name varchar(4) 일 때 다음과 같이 저장 : 'a', '한국', '고고고고', 'wow' 	VARCHAR(n) (0 <= n <= 65,535)
사이즈가 큰 문자열	- 사이즈가 큰 문자열을 저장할 때 사용	TINYTEXT TEXT MEDIUMTEXT LONGTEXT

char 보다 varchar가 좋은거 아니냐?

-> mysql의 경우 varchar쓸때의 storage상으로는 이점은 있겠지만 시간적인 성능에서 char에 비해 안좋다는 얘기가 있다.

-> mysql에서는 phonenumbers같이 문자열의 크기가 고정되어있는 경우는 char를 권장

-> 문자열의 크기가 가변적인경우 varchar를 권장.

mysql에서 varchar보다 더 많은 문자열의 길이를 저장할 수 있는 것은 mediumtext와 longtext이다.

attribute data type : 날짜와 시간

종류	설명	MySQL
날짜	- 년, 월, 일을 저장 - YYYY-MM-DD	DATE ('1000-01-01' ~ '9999-12-31')
시간	- 시, 분, 초를 저장 - hh:mm:ss or hhh:mm:ss	TIME ('-838:59:59' ~ '838:59:59')
날짜와 시간	- 날짜와 시간을 같이 표현 - YYYY-MM-DD hh:mm:ss - TIMESTAMP는 time-zone이 반영됨	DATETIME ('1000-01-01 00:00:00' to '9999-12-31 23:59:59') TIMESTAMP ('1970-01-01 00:00:01' UTC ~ '2038-01-19 03:14:07' UTC)

시간에서 hh처럼 3자리까지 표시하는 이유는 경과된 시간을 표시하고 싶을때 3자리로 표시할 수 있다.

timestamp로 저장을 할때는 utc기준

attribute data type : 그 외

종류	설명	MySQL
byte-string	(문자열이 아니라) byte string을 저장	BINARY VARBINARY BLOB type
boolean	- true, false를 저장 - MySQL에는 따로 없음	TINYINT로 대체해서 사용
위치 관련	위치 관련 정보를 저장	GEOMETRY etc
JSON	- json 형태의 데이터를 저장 - e.g. {"name": "messi", "age": 38}	JSON

byte-string : 예로 보안과 관련되어 암호화하고 싶을 때 암호화 키를 저장

boolean : tinyint를 0과 1로 저장해서 처리

key constraints : PRIMARY KEY

- primary key : table의 tuple을 식별하기 위해 사용, 하나 이상의 attribute(s)로 구성
- primary key는 중복된 값을 가질 수 없으며, null도 값으로 가질 수 없다.

PLAYER				
<u>team_id</u>	<u>back_number</u>	<u>birth_date</u>	<u>name</u>	<u>phone_number</u>
team_132	7
team_132	7
team_371	NULL

- team_id, back_number가 pk _ 언더바로 표시
- 중복된 값, null을 포함하기 때문에 제약위반
- 선언 방법

- primary key를 선언하는 방법은 아래와 같다

attribute 하나로 구성될 때	attribute 하나 이상으로 구성될 때
<pre>create table PLAYER (id INT PRIMARY KEY ...);</pre>	<pre>create table PLAYER (team_id VARCHAR(12), back_number INT, ... PRIMARY KEY (team_id, back_number));</pre>

unique key

- unique로 지정된 attribute(s)는 중복된 값을 가질 수 없다.
- 단, null은 중복을 허용할 수도 있다. (RDBMS마다 다름)

PLAYER

team_id	back_number	birth_date	name	phone_number
...	010-1111-2312
...	010-0876-1921
...	010-1111-2312

↑ UNIQUE

- 중복된 데이터를 가질 수 없다.
- 선언 방법

attribute 하나로 구성될 때	attribute 하나 이상으로 구성될 때
<pre>create table PLAYER (id INT UNIQUE ...);</pre>	<pre>create table PLAYER (team_id VARCHAR(12), back_number INT, ... UNIQUE (team_id, back_number));</pre>

NOT NULL constraint

- attribute가 not null로 지정되면 해당 attribute는 null을 값으로 가질 수 없다.

PLAYER				
team_id	back_number	birth_date	name	phone_number
...	010-1111-2312
...	NULL
...	010-1111-2312

- 선언방법

```
create table Student (
    ...
    phone_number  INT  NOT NULL  UNIQUE,
    ...
);
```

- 보통 not null과 unique를 함께 사용한다.

EMPLOYEE

```
mysql> create table EMPLOYEE (
->   id          INT      PRIMARY KEY,
->   name        VARCHAR(30) NOT NULL,
->   birth_date  DATE,
->   sex         CHAR(1)   CHECK (sex in ('M', 'F')),
->   position    VARCHAR(10),
->   salary      INT      DEFAULT 50000000,
->   dept_id    INT,
->   FOREIGN KEY (dept_id) references DEPARTMENT(id)
->       on delete SET NULL on update CASCADE,
->   CHECK (salary >= 50000000)
-> );
```

attribute DEFAULT

- attribute의 default값을 정의할 때 사용
- 새로운 tuple을 저장할 때 해당 attribute에 대한 값이 없다면 default 값으로 저장

EMPLOYEE					DEFAULT 50000000
<u>id</u>	name	birth_date	...	salary	
...
...
...

- 선언방법

```
create table Orders (
    ...
    menu  varchar(15)  DEFAULT '짜장면',
    ...
);
```

CHECK constraint

- attribute의 값을 제한하고 싶을 때 사용
 - 아래의 경우 5천 미만의 값을 입력 x

CHECK (salary >= 50000000)
↓

EMPLOYEE				
<u>id</u>	name	birth_date	...	salary
...	10000000
...
...

- 선언방법

- CHECK를 선언하는 방법은 아래와 같다

attribute 하나로 구성될 때	attribute 하나 이상으로 구성될 때
<pre>create table EMPLOYEE (... age INT CHECK (age >= 20));</pre>	<pre>create table PROJECT (start_date DATE, end_date DATE, ... CHECK (start_date < end_date));</pre>

FOREIGN KEY : Referential integrity constraint

- attribute(s)가 다른 table의 primary key나 unique key를 참조할 때 사용

EMPLOYEE			DEPARTMENT		
<u>id</u>	...	department_id	<u>id</u>
...	...	256	128
...	...	1024	256
...	...	256	1024

- foreignkey로 지정된 값은 참조하는 키에 반드시 존재하는 값이여야 한다.
- employee의 department_id에 있는 값이 department에 id에는 반드시 존재하여야 한다.
- 선언방법

reference_option	설명
CASCADE	참조값의 삭제/변경을 그대로 반영
SET NULL	참조값이 삭제/변경 시 NULL로 변경
RESTRICT	참조값이 삭제/변경되는 것을 금지
NO ACTION	RESTRICT와 유사
SET DEFAULT	참조값이 삭제/변경 시 default 값으로 변경

- 여러개의 FK라면 dept_id 자리에 나열하면 된다.
- references 참조할 테이블명(attribute name)
- 참조값이 삭제되거나 업데이트 될 때 어떤 옵션을 줄 것인가
 - cascade : DEPARTMENT가 삭제/변경 시 EMPLOYEE도 동일하게 삭제/변경.
 - SET NULL : DEPARTMENT가 삭제/변경 시 EMPLOYEE는 NULL로 변경
 - RESTRICT : DEPARTMENT가 삭제/변경되는 것을 금지, 참조가 되고 있다면 금지
 - NO ACTION : RESTRICT 과 유사, 한 트랜잭션 내에 여러 개 SQL이 실행되는 동안에는 참조값이 삭제/변경되는 것을 허용하지만, 그 트랜잭션이 끝났을 때 Referential integrity constraint를 위반하고 있다면 이것은 금지한다.라는 의미
 - SET DEFAULT : DEPARTMENT가 삭제/변경 시 EMPLOYEE는 DEFAULT 값으로 변경
 - MySQL의 경우 SET DEFAULT를 제대로 지원하지 않는다.

constraint 이름 명시하기

- 이름을 붙이면 어떤 constraint를 위반했는지 쉽게 파악할 수 있다.
- constraint를 삭제하고 싶을 때 해당 이름으로 삭제 가능

create table TEST (age INT CONSTRAINT age_over_20 CHECK (age > 20));	CONSTRAINT age_over_20 이름을 붙였을 때	CONSTRAINT age_over_20 이름을 생략했을 때
	Check constraint 'age_over_20' is violated.	Check constraint 'test_chk_1' is violated.

- 이름을 통해 쉽게 파악 가능
- 이름을 붙이면 직관적으로 확인 가능
- 이름이 없을 경우 show create table 테이블명; 을 입력하면 결과 화면에서 test_chk_1이 무엇을 의미하는지 확인 가능

```
mysql> create table PROJECT (
->   id          INT      PRIMARY KEY,
->   name        VARCHAR(20) NOT NULL    UNIQUE,
->   leader_id   INT,
->   start_date  DATE,
->   end_date    DATE,
->   FOREIGN KEY (leader_id) references EMPLOYEE(id)
->       on delete SET NULL on update CASCADE,
->   CHECK (start_date < end_date)
-> );
```

```
mysql> create table WORKS_ON (
->   empl_id     INT,
->   proj_id     INT,
->   PRIMARY KEY (empl_id, proj_id),
->   FOREIGN KEY (empl_id) references EMPLOYEE(id)
->       on delete CASCADE on update CASCADE,
->   FOREIGN KEY (proj_id) references PROJECT(id)
->       on delete CASCADE on update CASCADE
-> );
```

테이블이 만들어진 뒤에 스키마를 변경할 경우

ALTER TABLE

- table의 schema를 변경하고 싶을 때 사용

유형	MySQL 예제
attribute 추가	ALTER TABLE employee ADD blood VARCHAR(2);
attribute 이름 변경	ALTER TABLE employee RENAME COLUMN phone TO phone_num;
attribute 타입 변경	ALTER TABLE employee MODIFY COLUMN blood CHAR(2);
table 이름 변경	ALTER TABLE logs RENAME TO backend_logs;
primary key 추가	ALTER TABLE log ADD PRIMARY KEY (id);
...	...

- ALTER TABLE 변경하려는 TABLE이름 작업

```
mysql> ALTER TABLE department ADD FOREIGN KEY (leader_id)
      -> REFERENCES employee(id)
      -> on update CASCADE
      -> on delete SET NULL;
```

- 이미 서비스 중인 table의 schema를 변경하는 것이라면 변경 작업 때문에 서비스의 백엔드에 영향이 없을지 검토한 후에 변경하는 것이 중요

DROP TABLE

- table을 삭제할 때 사용
- DROP TABLE table_name;

database구조를 정의할 때 중요한 점

- 만들려는 서비스의 스펙과 데이터 일관성, 편의성, 확장성 등등을 종합적으로 고려하여 DB 스키마를 적절하게 정의하는 것이 중요하다.