**MySQL 중복 키 관리 방법 (INSERT 시 중복 키 관리 방법 (INSERT IGNORE, REPLACE INTO, ON DUPLICATE UPDATE)**

05 Mar 2014

안내

본 문서는 블로그의 운영자인 본인이 Stackoverflow에 올린 답변을 정리한 글입니다. Stackoverflow에 올린 답변 중 한국에 있는 다른 개발자들에게도 도움이 될만한 Q&A를 보기 쉽게 정리했습니다. 가능한 경우는 SQLFiddle에 샘플 데이터도 같이 올려서 실습도 해 볼 수 있도록 하였습니다. 또한 전체 Q&A를 묶어서 PDF 파일로도 배포하고 있습니다. 방문해 주시는 많은 분들에게 도움이 되었으면 좋겠습니다.

Stackoverflow URL

<http://stackoverflow.com/questions/20342518/on-duplicate-key-update-value-inserting-same-values-twice/20342598>

질문

다음과 같이 INSERT 구문을 사용 중이다.

INSERT INTO person VALUES(NULL, 15, 'James', 'Barkely')

ON DUPLICATE KEY UPDATE academy\_id = VALUES(academy\_id);

중복된 값을 여러 번 INSERT한 뒤에 SELECT를 해 보면 중복된 값이 저장되어 있다.

mysql> SELECT \* FROM person;

+----+------------+------------+-----------+

| id | academy\_id | first\_name | last\_name |

+----+------------+------------+-----------+

| 1 | 15 | James | Barkely |

| 2 | 15 | Cynthia | Smith |

| 3 | 15 | James | Barkely |

| 4 | 15 | Cynthia | Smith |

| 5 | 15 | James | Barkely |

+----+------------+------------+-----------+

5 rows in set (0.00 sec)

무엇이 잘못된 것인가?

답변

중복 키 관리를 위해서는 중복 방지를 할 컬럼이 PRIMARY KEY이거나 UNIQUE INDEX이어야 한다. 질문자의 경우 first\_name과 last\_name의 조합을 이용하여 중복 관리를 하려는 것 같다. 따라서다음과 같이 last\_name, last\_name을 PRIMARY KEY로 설정하거나,

ALTER TABLE person ADD PRIMARY KEY (first\_name, last\_name) UNIQUE INDEX를 추가해야 한다.

ALTER TABLE person ADD UNIQUE INDEX (first\_name, last\_name)

INSERT 시 중복 키 관리를 위한 방법에는 다음과 같이 3가지가 있다.

1. INSERT IGNORE
2. REPLACE INTO …
3. INSERT INTO … ON DUPLICATE UPDATE

중복 키 관리는 본 책의 앞 부분에서 잠시 언급되었는데 각각의 특징을 좀 더 자세히 알아보도록 하자.

앞의 person 테이블을 다음과 같이 생성한 뒤에 테스트를 진행하였다.

CREATE TABLE person

(

id INT NOT NULL AUTO\_INCREMENT,

academy\_id INT,

first\_name VARCHAR(20),

last\_name VARCHAR(20),

PRIMARY KEY (id),

UNIQUE INDEX (first\_name, last\_name)

);

INSERT IGNORE

`INSERT IGNORE`는 중복 키 에러가 발생했을 때 신규로 입력되는 레코드를 무시하는 단순한 방법이다. 다음의 예를 보면 중복 키 에러가 발생했을 때 INSERT 구문 자체는 오류가 발생하지 않고, 대신'0 row affected'가 출력된 것을 볼 수 있다.

mysql> INSERT IGNORE INTO person VALUES (NULL, 15, 'James', 'Barkely');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT IGNORE INTO person VALUES (NULL, 15, 'Cynthia', 'Smith');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT IGNORE INTO person VALUES (NULL, 15, 'James', 'Barkely');

Query OK, 0 rows affected (0.00 sec)

mysql> INSERT IGNORE INTO person VALUES (NULL, 15, 'Cynthia', 'Smith');

Query OK, 0 rows affected (0.00 sec)

mysql> INSERT IGNORE INTO person VALUES (NULL, 15, 'James', 'Barkely');

Query OK, 0 rows affected (0.00 sec)

당연히 SELECT의 결과는 2건만 존재한다.

mysql> SELECT \* FROM person;

+----+------------+------------+-----------+

| id | academy\_id | first\_name | last\_name |

+----+------------+------------+-----------+

| 1 | 15 | James | Barkely |

| 2 | 15 | Cynthia | Smith |

+----+------------+------------+-----------+

2 rows in set (0.00 sec)

AUTO\_INCREMENT 컬럼의 값이 1, 2인 것에 주목하라

REPLACE INTO

"REPLACE INTO"는 "INSERT INTO" 구문에서 INSERT를 REPLACE로 바꾼 구문이다. 사용 방법은 "INSERT INTO"와 완벽히 동일하다.

mysql> REPLACE INTO person VALUES (NULL, 15, 'James', 'Barkely');

Query OK, 1 row affected (0.00 sec)

mysql> REPLACE INTO person VALUES (NULL, 15, 'Cynthia', 'Smith');

Query OK, 1 row affected (0.00 sec)

mysql> REPLACE INTO person VALUES (NULL, 15, 'James', 'Barkely');

Query OK, 2 rows affected (0.00 sec)

mysql> REPLACE INTO person VALUES (NULL, 15, 'Cynthia', 'Smith');

Query OK, 2 rows affected (0.00 sec)

mysql> REPLACE INTO person VALUES (NULL, 15, 'James', 'Barkely');

Query OK, 2 rows affected (0.00 sec)

REPLACE INTO의 결과는 INSERT IGNORE와 다르게 중복 키 오류 발생 시 ‘2 rows affected’가 출력되었다. SELECT 결과는 다음과 같다.

mysql> SELECT \* FROM person;

+----+------------+------------+-----------+

| id | academy\_id | first\_name | last\_name |

+----+------------+------------+-----------+

| 4 | 15 | Cynthia | Smith |

| 5 | 15 | James | Barkely |

+----+------------+------------+-----------+

2 rows in set (0.00 sec)

id가 4, 5로 변하였다. 이를 ‘2 rows affected’와 함께 종합적으로 판단한다면 “REPLACE INTO”는 중복 키 오류 발생 시 기존 레코드를 삭제하고 새로운 레코드를 입력한 것이다. 그래서 ‘2 rows affected’가 출력되었다. 1건은 DELETE, 1건은 INSERT로 보면 되고, 새로운 레코드가 입력되면서 AUTO\_INCREMENT 컬럼의 값이 매번 새롭게 발급되었다.

AUTO\_INCREMENT는 흔히 레코드를 식별할 수 있는 id로 사용되는데 이 값이 변경될 수 있으므로 “REPLACE INTO”는 그다지 좋은 방법이 아니다.

ON DUPLICATE UPDATE

ON DUPLICATE UPDATE는 중복 키 오류 발생 시 사용자가 원하는 값을 직접 설정할 수 있다는 장점이 있다. 우선 기본적인 사용 방법을 보자.

mysql> INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely')

-> ON DUPLICATE KEY UPDATE academy\_id = VALUES(academy\_id);

Query OK, 1 row affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'Cynthia', 'Smith')

-> ON DUPLICATE KEY UPDATE academy\_id = VALUES(academy\_id);

Query OK, 1 row affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely')

-> ON DUPLICATE KEY UPDATE academy\_id = VALUES(academy\_id);

Query OK, 0 rows affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'Cynthia', 'Smith')

-> ON DUPLICATE KEY UPDATE academy\_id = VALUES(academy\_id);

Query OK, 0 rows affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely')

-> ON DUPLICATE KEY UPDATE academy\_id = VALUES(academy\_id);

Query OK, 0 rows affected (0.00 sec)

INSERT 결과만 보면 “INSERT IGNORE”와 동일하다.

mysql> SELECT \* FROM person;

+----+------------+------------+-----------+

| id | academy\_id | first\_name | last\_name |

+----+------------+------------+-----------+

| 1 | 15 | James | Barkely |

| 2 | 15 | Cynthia | Smith |

+----+------------+------------+-----------+

2 rows in set (0.00 sec)

SELECT 결과를 보니, 중복 키 오류 발생 시 기존 레코드는 그대로 남아 있는 것 같다. 즉, id 값이 변경되지 않았다. 그렇다면 “ON DUPLICATE UPDATE”는 “INSERT IGNORE” 대비 장점은 없을까?아니다. 복잡하고 어려운 대신에 중복 키 오류 발생 시 사용자가 원하는 행동을 지정할 수 있다는 장점이 있다. 예를 위해 person 테이블 구조를 다음과 같이 변경했다.

CREATE TABLE person

(

id INT NOT NULL AUTO\_INCREMENT,

academy\_id INT,

first\_name VARCHAR(20),

last\_name VARCHAR(20),

insert\_cnt INT,

PRIMARY KEY (id),

UNIQUE INDEX (first\_name, last\_name)

);

그런 뒤 다음과 같은 INSERT 구문을 실행하였다.

mysql> INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely', 1)

-> ON DUPLICATE KEY UPDATE insert\_cnt = insert\_cnt + 1;

Query OK, 1 row affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'Cynthia', 'Smith', 1)

-> ON DUPLICATE KEY UPDATE insert\_cnt = insert\_cnt + 1;

Query OK, 1 row affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely', 1)

-> ON DUPLICATE KEY UPDATE insert\_cnt = insert\_cnt + 1;

Query OK, 2 rows affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'Cynthia', 'Smith', 1)

-> ON DUPLICATE KEY UPDATE insert\_cnt = insert\_cnt + 1;

Query OK, 2 rows affected (0.00 sec)

mysql>

mysql> INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely', 1)

-> ON DUPLICATE KEY UPDATE insert\_cnt = insert\_cnt + 1;

Query OK, 2 rows affected (0.00 sec) SELECT를 해 보면 insert\_cnt에는 해당 중복 값이 몇 번 INSERT 시도가 되었는지 기록되어 있을 것이다.

mysql> SELECT \* FROM person;

+----+------------+------------+-----------+------------+

| id | academy\_id | first\_name | last\_name | insert\_cnt |

+----+------------+------------+-----------+------------+

| 1 | 15 | James | Barkely | 3 |

| 2 | 15 | Cynthia | Smith | 2 |

+----+------------+------------+-----------+------------+

2 rows in set (0.00 sec)

이 외에도 다양한 용도로 활용될 수 있다.

주의해야 할 점은 INSERT 구문에 주어진 값으로 UPDATE하고자 할 때는 항상 “VALUES(column)”과 같이 VALUES()로 감싸야 한다는 점이다. 만약 다음과 같이 VALUES() 없이 사용한다면 기존에존재하는 레코드의 컬럼 값을 의미하게 된다.

INSERT INTO person VALUES (NULL, 15, 'James', 'Barkely')

ON DUPLICATE KEY UPDATE academy\_id = academy\_id;

앞과 같이 사용했을 때, 기존 person 테이블에 존재하는 ‘James Barkely’의 academy\_id가 13이었다면, INSERT 후에도 academy\_id는 여전히 13이다.

요약

* INSERT IGNORE
* REPLACE INTO …
* INSERT INTO … ON DUPLICATE UPDATE

| **분류** | **특징** |
| --- | --- |
| INSERT IGNORE | 기존 레코드가 남아 있음 기존 레코드의 AUTO\_INCREMENT 값은 변하지 않음 |
| REPLACE INTO | 기존 레코드가 삭제되고, 신규 레코드가 INSERT됨 따라서 AUTO\_INCREMENT의 값이 변경됨 |
| ON DUPLICATE UPDATE | INSERT IGNORE의 장점 포함함 중복 키 오류 발생 시, 사용자가 UPDATE될 값을 지정할 수 있음 |

http://jason-heo.github.io/mysql/2014/03/05/manage-dup-key2.html