



# 데이터베이스에 데이터 삽입

## Database Fundamentals

발표자 이름

날짜

© 2019, Amazon Web Services, Inc. 또는 계열사. All rights reserved.

데이터베이스에 데이터 삽입을 시작하겠습니다.

## 교육 내용

### 이 강의의 핵심

배울 내용은 다음과 같습니다.

- 기존 테이블에 행 삽입하기
- 데이터베이스로 가져오기 전에 데이터를 정리해야 하는 잠재적인 문제 식별하기
- 테이블에 CSV 파일 가져오기

주요 용어:

- INSERT INTO 스테이트먼트
- DESCRIBE 스테이트먼트
- NULL 값



이 모듈에서 학습할 내용은 다음과 같습니다.

- 기존 데이터베이스에서 새 테이블 만들기
- 테이블 생성 시 데이터 유형 구현하기



활동

## 활동: 토론 - 테이블에 행이 필요한 이유



### 활동

- 테이블에 행을 삽입해야 하는 이유를 논의합니다.

INSERT 스테이트먼트

# INSERT

## INSERT INTO 스테이트먼트

- 데이터베이스 테이블을 데이터로 채우기 위한 기초 스테이트먼트입니다.
- SQL INSERT 스테이트먼트는 하나의 레코드 또는 여러 레코드를 테이블에 삽입하는 데 사용됩니다.
- SQL INSERT 스테이트먼트를 **데이터 조작 언어(DML)** 명령이라고 합니다.
- 참고: **열의 순서**는 선택할 때 중요합니다.



## INSERT INTO 스테이트먼트

```
INSERT INTO tableName (col_1,col_2,col_3...col_n)  
VALUES ("val_1","val_2","val_3","val_n");
```

행을 삽입할 때 데이터가 들어갈 열을 지정해야 합니다. 열을 지정하지 않으면 테이블의 모든 열에 값이 추가되어 단일 행이 추가됩니다. 값을 삽입할 때 문자 값 또는 날짜 값에서는 값을 작은따옴표(')로 묶어야 합니다. 하나 이상의 특정 열에 대한 값을 삽입하는 경우 정형 쿼리 언어(SQL)로 열 이름을 지정해야 합니다.

## SQL 파라미터

이름	설명
tableName	데이터를 삽입하는 테이블
col_1,col_2,col_3,...	데이터가 삽입되는 테이블의 각 열
val_1,val_2,val_3,...	각 열에 대한 값

주의:

- **열**은 필드 제목을 나타냅니다.
- **값**은 필드에 입력할 데이터를 나타냅니다.



## INSERT 스테이트먼트의 구문

### I. SQL의 단일 레코드에 데이터를 삽입하기 위한 구문

```
INSERT INTO tableName  
VALUES ("val_1","val_2","val_3","val_n"),  
       ("val_11","val_22","val_32","val_nn");
```

### II. SQL의 다중 레코드에 데이터를 삽입하기 위한 구문

```
INSERT INTO tableName (col_1,col_2,col_3...col_n)  
VALUES ("val_1","val_2","val_3","val_n"),  
       ("val_11","val_22","val_32","val_nn");
```

먼저 INSERT INTO 절 다음에 오는 괄호 안에 **테이블 이름**과 **쉼표로 구분된 열 목록**을 지정합니다. 그런 다음 VALUES 키워드 다음에 오는 괄호 안에 해당 열의 **값을 쉼표로 구분된 목록으로** 입력합니다.

INSERT 스테이트먼트는 2가지 방식으로 사용할 수 있습니다. 첫 번째 예에서는 테이블의 모든 열에 대한 값을 추가할 때 열 이름을 지정할 필요가 없습니다. 두 번째 예에서는 열 이름과 값이 모두 기록됩니다. 열의 개수와 값의 개수는 같아야 합니다. 또한 열의 위치는 해당 값의 위치와 일치해야 합니다.

## INSERT 스테이트먼트(계속)

### employees

employee id	family name	given name
EN1-10	Wang	Xiulan
EN1-12	Ramirez	Diego
EN1-19	Major	Mary
EN1-22	Sarkar	Saanvi
EN1-27	Mansa	Akua
EN1-35	Jackson	Mateo

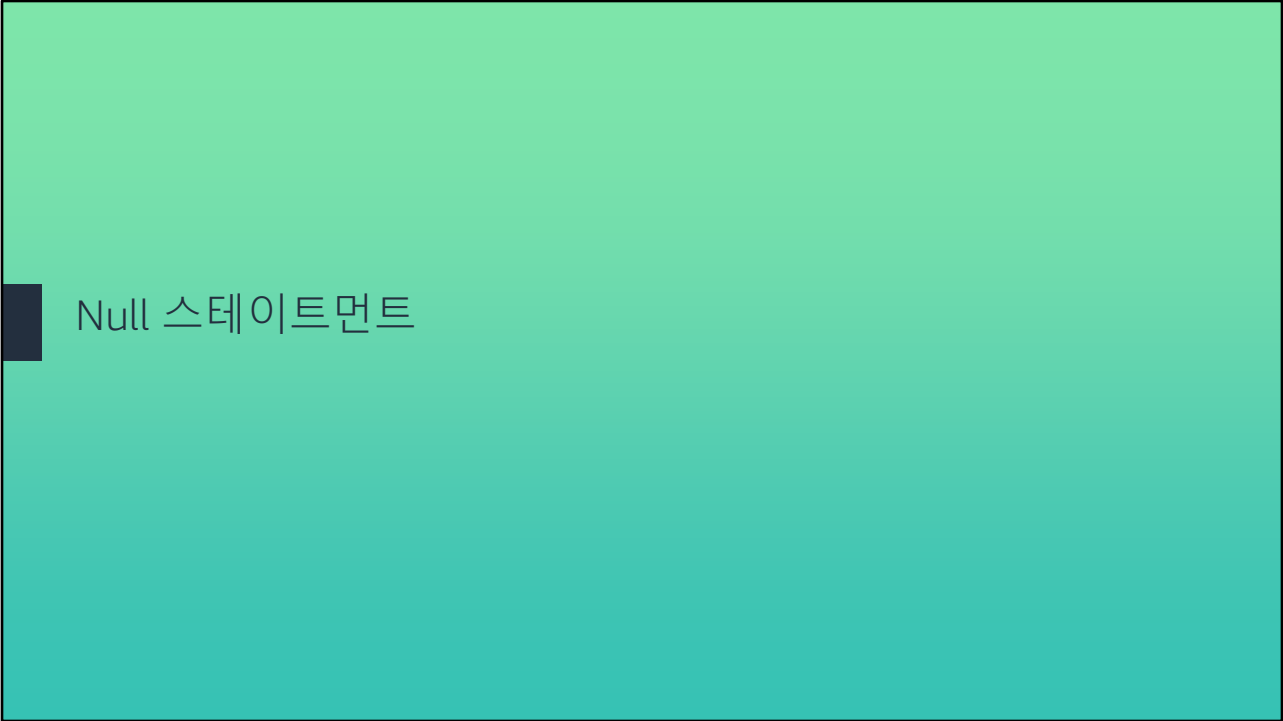
```
INSERT INTO employees
VALUES ("EN1-35", "Jackson",
"Mateo");
```

이 예에서는 INSERT 스테이트먼트 구문을 보여줍니다. 이 정보는 새로 생성된 행에 배치됩니다.

## DESCRIBE 스테이트먼트

일반적으로 테이블에는 열이 2개 이상 있습니다. DESCRIBE 스테이트먼트는 지정된 테이블 또는 보기를 설명합니다.

```
DESCRIBE table_name;
```



Null 스테이트먼트

## Null 스테이트먼트



```
INSERT INTO tableName(col_1) values(NULL);
```

INSERT 스테이트먼트는 열에 NULL 값을 삽입할 수 있습니다. 열에 NOT NULL 제약 조건이 없어야 한다는 조건으로 NULL 값을 **int** 열에 삽입할 수 있습니다.

## Null 스테이트먼트 관련 몇 가지 중요 사항

- 관계형 데이터베이스 관리 시스템(RDBMS)에는 열을 선택할 수 있다는 개념이 기본 전제로 깔려 있습니다.
- NULL은 해당되지 않음 또는 알 수 없음을 의미하며 0이나 공백과 다릅니다.
- 데이터베이스에 NULL이 있으면 데이터베이스에 해당 필드에 대한 값이 없음을 의미합니다.
- Null에는 어떤 값이 아니라 없음을 뜻합니다.
- $5 + \text{null} = \text{null}$
- $\text{null} = \text{null}$ 은 FALSE입니다.
- NOT NULL 절을 사용하여 열을 만든 경우 SELECT를 수행할 때 값을 입력해야 합니다. 값이 없으면 오류 메시지가 표시됩니다.



활동

## 활동: 테이블 만들기



### 활동

- AnyCompany Publishing House 로열티 프로그램을 기존 데이터베이스에 통합하기로 했습니다. 하지만 참여하기로 한 고객의 잔액을 추적할 방법이 필요합니다. **Loyalty**라는 데이터베이스 테이블을 생성해야 하며 다음 필드가 있어야 합니다.
  - Customer ID(고객 ID)
  - Customer family name(고객 성)
  - Customer given name(고객 이름)
  - Loyalty number(로열티 번호)
  - Contact number(연락처 번호)
  - Email address(이메일 주소)
  - Points balance(포인트 잔액)
- 다른 학습자와 함께 사용해야 할 필드와 데이터 유형을 논의합니다.



## 활동: 테이블 만들기 - 솔루션

### Loyalty

Customer ID	Family Name	Given Name	Loyalty Number	Contact Number	Email Address	Points Balance



데이터 가져오기

## CSV 가져오기

- **확인**
  - 쉼표로 구분된 값(CSV) 파일에는 테이블의 열 수 및 각 열의 데이터 유형에 맞는 데이터가 있습니다.
- 가져오려는 CSV 파일에 해당하는 테이블 이름으로 MySQL에 테이블을 **생성합니다**.
- 다음 방법 중 하나로 명령을 통해 **가져옵니다**.
  - CSV 파일에는 식별자로 간주되지 않는 열 머리가 포함되어 있으므로 **\n**을 입력하여 CSV 파일의 첫 번째 행이 무시되어야 합니다.
  - 헤더 열 제목 없이 CSV 파일의 새로운 버전을 생성한 다음 빈 문자열을 NULL을 나타내는 **\n**으로 바꿉니다. 그런 다음 CSV 파일을 MySQL이 열 수 있는 디렉터리로 이동합니다.

## CSV 가져오기(계속)

다음 스테이트먼트는 임시 파일에서 데이터를 **Loyalty** 테이블로 가져옵니다.

```
LOAD DATA INFILE 'c:/tmp/loyalty.csv' INTO TABLE loyalty  
FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES  
TERMINATED BY '\n' IGNORE 1 ROWS;
```



데이터 정리하기

## 데이터 정리하기

- 데이터-입력 오류
- 잘못된 데이터 유형
- 잘못된 코드
- 비논리적 데이터
- 데이터 누락
- 중복 데이터
- 텍스트에서 사용할 수 없는 문자
- 검열되거나 잘린 데이터
- 일관성 없는 데이터 표현
- 데이터는 최소 단위가 아님
- 깨진 참조



데이터베이스는 시간이 지남에 따라 정리되지 않은 상태가 되어 문제가 발생할 수 있습니다. 일부 SQL 문자열 함수는 데이터를 정리하는 데 사용할 수 있습니다. LEFT, RIGHT 및 TRIM 을 사용하여 데이터를 자르면 문자열의 특정 요소만 선택하고 특정 문자를 제거할 수 있습니다. CONCAT을 사용하면 여러 열의 문자열을 결합하고 함께 넣을 수 있습니다. LOWER 와 같은 다른 스테이트먼트를 사용하여 문자열의 모든 문자를 소문자로 만들 수 있습니다. 마찬가지로 UPPER를 사용하여 모든 문자를 대문자로 표시할 수 있습니다.

활동

## 활동: 오류 식별하기



### 활동

- 데이터베이스에서 다음 유형의 오류가 발생할 수 있습니다. 기능적 데이터베이스에서 이러한 오류가 어떻게 표현되는지 논의합니다. 또한 이러한 오류를 해결할 수 있는 기술에 대해서도 논의합니다.
  - 데이터-입력 오류
  - 잘못된 데이터 유형
  - 잘못된 코드
  - 비논리적 데이터
  - 데이터 누락
  - 중복 데이터
  - 텍스트에서 사용할 수 없는 문자
  - 검열되거나 잘린 데이터
  - 일관성 없는 데이터 표현
  - 데이터는 최소 단위가 아님
  - 깨진 참조



## 학습 내용 확인 질문

INSERT INTO 스테이트먼트를 사용하는 두 가지 방법은 무엇입니까?

NULL의 목적은 무엇입니까?

정답:

1. **INSERT INTO** 스테이트먼트를 사용하여 **데이터베이스**에 새 **데이터**를 추가할 수 있습니다.  
**INSERT INTO** 스테이트먼트는 **테이블**에 새 레코드를 추가할 수도 있습니다.
2. **NULL**이라는 용어는 누락된 값을 나타내는 데 사용됩니다.

# 핵심 사항



© 2020, Amazon Web Services, Inc. 또는 계열사. All rights reserved.

26

- 행을 삽입할 때 데이터가 들어갈 열을 지정해야 합니다.
- NULL 값에 대해 테스트할 때는 IS NULL 및 IS NOT NULL 명령을 사용합니다.
- 데이터베이스는 시간이 지남에 따라 정리되지 않은 상태가 되어 문제가 발생할 수 있습니다. 일부 SQL 문자열 함수는 데이터를 정리하는 데 사용할 수 있습니다.

aws re/start

이 강의에서 다룬 핵심 사항은 다음과 같습니다.

- 행을 삽입할 때 데이터가 들어갈 열을 지정해야 합니다.
- NULL 값에 대해 테스트할 때는 IS NULL 및 IS NOT NULL 명령을 사용합니다.
- 데이터베이스는 시간이 지남에 따라 정리되지 않은 상태가 되어 문제가 발생할 수 있습니다. 일부 SQL 문자열 함수는 데이터를 정리하는 데 사용할 수 있습니다.