MySQL 기초 실습 가이드

안녕하세요! 오늘은 우분투에서 MySQL을 사용하는 방법을 함께 배워보겠습니다. 여러분이 이미 Hadoop과 MySQL Server 5.7을 설치했으니, 이제 본격적으로 MySQL을 다뤄보는 시간이에요.

1. MySQL 접속하기

먼저 MySQL에 접속해보겠습니다. 터미널을 열고 다음 명령어를 입력하세요:

```
sudo mysql -u root -p
```

패스워드를 입력하면 MySQL 프롬프트가 나타날 거예요. `mysql>` 이라고 보이면 성공입니다!

2. 기본 명령어들 알아보기

현재 시간 확인하기

MySQL이 잘 작동하는지 간단히 확인해보겠습니다:

```
sql
SELECT NOW();
```

현재 시간이 나오면 MySQL이 정상적으로 작동하고 있는 거예요.

사용자 정보 확인하기

현재 접속한 사용자 정보를 확인해보세요:

```
sql
SELECT USER();
```

MySQL基础实习指南

大家好！今天我们一起学习如何在Ubuntu中使用MySQL。既然大家已经安装了Hadoop和MySQL Server 5.7，现在就让我们正式开始学习MySQL的使用方法吧。

1. 连接MySQL

首先让我们连接到MySQL。打开终端，输入以下命令：

```
sudo mysql -u root -p
```

输入密码后会出现MySQL提示符。看到`mysql>`就表示连接成功了！

2. 了解基本命令

查看当前时间

让我们简单检查一下MySQL是否正常工作：

```
sql
SELECT NOW();
```

如果显示当前时间，说明MySQL正常运行。

查看用户信息

确认当前连接的用户信息：

```
sql
SELECT USER();
```

3. 데이터베이스 다루기

3.1 현재 존재하는 데이터베이스 보기

시스템에 어떤 데이터베이스들이 있는지 먼저 확인해보겠습니다:

```sql
SHOW DATABASES;
```

기본적으로 `information_schema`, `mysql`, `performance_schema`, `sys` 등이 보일 거예요. 이들은 MySQL 시스템이 사용하는 기본 데이터베이스들입니다.

3.2 새로운 데이터베이스 생성하기

이제 우리만의 데이터베이스를 만들어보겠습니다:

```sql
CREATE DATABASE flinkdb;
```

`Query OK`라는 메시지가 나오면 성공적으로 생성된 거예요!

3.3 데이터베이스 선택하기

방금 만든 `flinkdb`를 사용하도록 설정해보겠습니다:

```sql
USE flinkdb;
```

`Database changed`라는 메시지가 나오면 이제 `flinkdb`가 현재 작업 중인 데이터베이스가 된 거예요.

3.4 현재 데이터베이스의 테이블 확인하기

현재 데이터베이스에 어떤 테이블들이 있는지 확인해보세요:

```sql
SHOW TABLES;
```

아직 테이블을 만들지 않았으니 `Empty set`이라고 나올 거예요.

3. 操作数据库

3.1 查看现有数据库

首先让我们看看系统中存在哪些数据库：

```sql
SHOW DATABASES;
```

通常会看到`information_schema`、`mysql`、`performance_schema`、`sys`等。这些都是MySQL系统使用的基本数据库。

3.2 创建新数据库

现在让我们创建属于自己的数据库：

```sql
CREATE DATABASE flinkdb;
```

看到`Query OK`消息就表示创建成功了！

3.3 选择数据库

将刚才创建的`flinkdb`设置为当前数据库：

```sql
USE flinkdb;
```

看到`Database changed`消息表示`flinkdb`已成为当前工作数据库。

3.4 查看当前数据库中的表

查看当前数据库里面存在哪些表：

```sql
SHOW TABLES;
```

由于还没有创建表，会显示`Empty set`。

4. 테이블 만들고 다루기

4.1 테이블 생성하기

이제 `person`이라는 테이블을 만들어보겠습니다. 이 테이블에는 다음과 같은 컬럼들이 있을 거예요:

- `id`: 자동으로 증가하는 번호
- `xm`: 이름
- `xb`: 성별
- `csny`: 출생년월일

```sql
CREATE TABLE person (
    id INT AUTO_INCREMENT PRIMARY KEY,
    xm VARCHAR(50) NOT NULL,
    xb VARCHAR(10) NOT NULL,
    csny DATE NOT NULL
);
```

```
mysql> create table person(
    -> id int auto_increment primary key,
    -> xm varchar(50) not null,
    -> xb varchar(10) not null,
    -> cany date not null
    -> );
Query OK, 0 rows affected (0.03 sec)
```

4.2 테이블 구조 확인하기

방금 만든 테이블의 구조를 확인해보겠습니다:

```sql
DESCRIBE person;
```

또는 짧게

```sql
DESC person;
```

컬럼 이름, 데이터 타입, NULL 허용 여부 등이 표시될 거예요.

4. 创建和操作表

4.1 创建表

现在让我们创建一个名为`person`的表。这个表包含以下字段：

- `id`：自动增长的序号
- `xm`：姓名
- `xb`：性别
- `csny`：出生年月

```sql
CREATE TABLE person (
    id INT AUTO_INCREMENT PRIMARY KEY,
    xm VARCHAR(50) NOT NULL,
    xb VARCHAR(10) NOT NULL,
    csny DATE NOT NULL
);
```

```
mysql> create table person(
    -> id int auto_increment primary key,
    -> xm varchar(50) not null,
    -> xb varchar(10) not null,
    -> cany date not null
    -> );
Query OK, 0 rows affected (0.03 sec)
```

4.2 查看表结构

查看刚才创建的表的结构：

```sql
DESCRIBE person;
```

或者简写为：

```sql
DESC person;
```

会显示字段名、数据类型、是否允许NULL等信息。

4.3 데이터 삽입하기

이제 테이블에 데이터를 넣어보겠습니다:

```sql
INSERT INTO person (xm, xb, csny) VALUES ('张三', '男', '1997-01-02');
INSERT INTO person (xm, xb, csny) VALUES ('李四', '女', '1996-12-02');
```

`id`는 `AUTO_INCREMENT`로 설정했기 때문에 자동으로 1, 2가 들어갈 거예요.

4.4 데이터 조회하기

입력한 데이터를 확인해보겠습니다:

```sql
SELECT * FROM person;
```

두 명의 정보가 잘 들어갔는지 확인해보세요!

4.5 데이터 수정하기

张三의 출생년월일을 수정해보겠습니다:

```sql
UPDATE person SET csny = '1971-01-10' WHERE xm = '张三';
```

수정이 잘 됐는지 다시 확인해보세요:

```sql
SELECT * FROM person;
```

4.6 데이터 삭제하기

张三의 레코드를 삭제해보겠습니다:

```sql
DELETE FROM person WHERE xm = '张三';
```

다시 확인해보면 李四만 남아있을 거예요:

```sql
SELECT * FROM person;
```

4.3 插入数据

现在向表中插入数据：

```sql
INSERT INTO person (xm, xb, csny) VALUES ('张三', '男', '1997-01-02');
INSERT INTO person (xm, xb, csny) VALUES ('李四', '女', '1996-12-02');
```

由于`id`设置了`AUTO_INCREMENT`，会自动插入1、2。

4.4 查询数据

查看插入的数据：

```sql
SELECT * FROM person;
```

确认两个人的信息是否正确插入！

4.5 修改数据

修改张三的出生年月：

```sql
UPDATE person SET csny = '1971-01-10' WHERE xm = '张三';
```

再次查询确认修改是否成功：

```sql
SELECT * FROM person;
```

4.6 删除数据

删除张三的记录：

```sql
DELETE FROM person WHERE xm = '张三';
```

再次查询，应该只剩下李四的记录：

```sql
SELECT * FROM person;
```

5. 정리하기

실습이 끝났으니 만들었던 것들을 정리해보겠습니다.

5.1 테이블 삭제하기

```sql
DROP TABLE person;
```

5.2 데이터베이스 삭제하기

```sql
DROP DATABASE flinkdb;
```

5.3 MySQL 종료하기

```sql
EXIT;
```

또는

```sql
QUIT;
```

5. 清理工作

实习结束后，让我们清理创建的内容。

5.1 删除表

```sql
DROP TABLE person;
```

5.2 删除数据库

```sql
DROP DATABASE flinkdb;
```

5.3 退出MySQL

```sql
EXIT;
```

或者：

```sql
QUIT;
```

6. 유용한 추가 명령어들

현재 선택된 데이터베이스 확인하기

```sql
SELECT DATABASE();
```

MySQL 버전 확인하기

```sql
SELECT VERSION();
```

도움말 보기

```sql
HELP;
```

6. 实用的附加命令

查看当前选中的数据库

```sql
SELECT DATABASE();
```

查看MySQL版本

```sql
SELECT VERSION();
```

查看帮助

```sql
HELP;
```

7. 실습 요약

오늘 배운 주요 명령어들을 정리해보겠습니다:

데이터베이스 관련:
- `SHOW DATABASES;` - 데이터베이스 목록 보기
- `CREATE DATABASE 이름;` - 데이터베이스 생성
- `USE 이름;` - 데이터베이스 선택
- `DROP DATABASE 이름;` - 데이터베이스 삭제

테이블 관련:
- `SHOW TABLES;` - 테이블 목록 보기
- `CREATE TABLE 이름 (...);` - 테이블 생성
- `DESC 테이블명;` - 테이블 구조 보기
- `DROP TABLE 이름;` - 테이블 삭제

데이터 관련:
- `INSERT INTO 테이블명 VALUES (...);` - 데이터 삽입
- `SELECT * FROM 테이블명;` - 데이터 조회
- `UPDATE 테이블명 SET ... WHERE ...;` - 데이터 수정
- `DELETE FROM 테이블명 WHERE ...;` - 데이터 삭제

참고: 모든 SQL 명령어는 세미콜론(`;`)으로 끝나야 합니다. 깜빡먹지 마세요! 😊

7. 实习总结

让我们总结一下今天学习的主要命令：

数据库相关：
- `SHOW DATABASES;` - 查看数据库列表
- `CREATE DATABASE 名称;` - 创建数据库
- `USE 名称;` - 选择数据库
- `DROP DATABASE 名称;` - 删除数据库

表相关：
- `SHOW TABLES;` - 查看表列表
- `CREATE TABLE 名称 (...);` - 创建表
- `DESC 表名;` - 查看表结构
- `DROP TABLE 名称;` - 删除表

数据相关：
- `INSERT INTO 表名 VALUES (...);` - 插入数据
- `SELECT * FROM 表名;` - 查询数据
- `UPDATE 表名 SET ... WHERE ...;` - 修改数据
- `DELETE FROM 表名 WHERE ...;` - 删除数据

注意： 所有SQL命令都必须以分号(`;`)结尾。千万别忘记！😊