1120 软件测试-mysql规划:

测试用例分类:

功能测试：主要测试软件的功能是否符合要求，如登录、注册、查看、修改、删除等功能。

```
测试语句:
1.登陆语句:
  mysql -u username -p password
2.创建数据库和表
  create database testDB;
3.创建表格:
  CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
 );
4.插入数据:
  INSERT INTO users (id, username, email, created_at) VALUES (1, 'casinan',
'caisn7049@gmail.com', '2021-01-2');
  INSERT INTO users (id, username, email, created_at) VALUES ('shac',
'test@testem.com','2023-01-01');
  INSERT INTO users (id, username, email, created_at) VALUES ('shac',
'test@testem.com','2023-01-01');
5.查询数据:
  SELECT * FROM users;
  SELECT username, email FROM users WHERE id = 1;
6.修改数据:
  UPDATE users SET email = 'scai1364@gmail.com' WHERE id = 1;
7.删除数据:
  DELETE FROM users WHERE id = 1;
8.添加索引:
  CREATE INDEX idx_email ON users (email);
9.删除索引:
  DROP INDEX idx_email ON users;
10.使用聚合函数:
  SELECT COUNT(*) as user_count FROM users;
  SELECT AVG(length(username)) AS avg_username_length FROM users;
11.分组查询:
  SELECT created_at, COUNT(*) AS count FROM users GROUP BY created_at;
12.连接查询:
  SELECT u1.id, u1.username, u2.email FROM users u1 INNER JOIN users u2 ON u1.id =
u2.id;
13.事务:
  START TRANSACTION;
  UPDATE users SET email = 'a18218061816@gmail.com' WHERE id = 1;
  ROLLBACK;  -- 如果出现问题, 回滚
  SELECT * FROM users where id=1
  COMMIT;    -- 如果一切正常, 则提交事务
  SELECT * FROM users where id=1;
14.使用存储过程和函数
  CREATE PROCEDURE GetUserCount()
BEGIN
    SELECT COUNT(*) AS total_users FROM users;
END;
CALL GetUserCount();
15.使用视图:
  CREATE VIEW user_view AS SELECT id, username, email FROM users;
16.使用触发器:
  CREATE TRIGGER user_insert_trigger AFTER INSERT ON users
FOR EACH ROW
BEGIN
    INSERT INTO user_view (id, username, email) VALUES (NEW.id, NEW.username,
NEW.email);
END;

   INSERT INRO users (id, username, email) VALUES (31, 'casinan',
'caisn7049@gmail.com');
   SELECT * FROM user_view;
17.使用mysql的内置函数:
数学函数:
   SELECT ABS(-10) as absolute_value;
   SELECT CEIL(1.5) as ceiling_value;
   SELECT FLOOR(1.5) as floor_value;
   SELECT ROUND(1.5) as rounding_value;
   SELECT TRUNCATE(1.5) as truncating_value;
   日期函数:
   SELECT CURDATE() as current_date;
   SELECT CURTIME() as current_time;
   SELECT NOW() as current_datetime;
   SELECT DATE_FORMAT('2021-01-01', '%Y-%m-%d') as formatted_date;
   字符串函数:
   SELECT CONCAT('hello', 'world') as concatenated_string;
   SELECT SUBSTRING('hello world', 6) as substring_value;
   SELECT REPLACE('hello world', 'l', 'z') as replaced_string;
   SELECT MD5('hello world') as md5_value;
   SELECT SHA1('hello world') as sha1_value;
   SELECT LEFT('hello world', 5) as left_string;
   SELECT RIGHT('hello world', 5) as right_string;
   SELECT LENGTH('hello world') as length_value;
   SELECT UPPER('hello world') as uppercase_string;
   SELECT LOWER('HELLO WORLD') as lowercase_string;
   条件函数:
   SELECT IF(1>2, 'true', 'false') as condition_value;
   SELECT IFNULL(NULL, 'default_value') as null_value;
   SELECT COALESCE(NULL, 'default_value') as coalesce_value;
   聚合函数:
   SELECT COUNT(*) as count_value;
   SELECT SUM(1) as sum_value;
   SELECT AVG(1) as avg_value;
   SELECT MAX(1) as max_value;
   SELECT MIN(1) as min_value;
   位函数:
   SELECT BIT_AND(1, 2) as bit_and_value;
   SELECT BIT_OR(1, 2) as bit_or_value;
   SELECT BIT_XOR(1, 2) as bit_xor_value;
   SELECT BIT_COUNT(1) as bit_count_value;
   随机函数:
   SELECT RAND() as random_value;
   SELECT RAND(10) as random_value;
   SELECT RANDINT(1, 10) as random_int_value;
   系统函数:
   SELECT USER() as user_name;
   SELECT DATABASE() as database_name;
   SELECT VERSION() as version_value;
   SELECT SCHEMA() as schema_name;
   SELECT CONNECTION_ID() as connection_id;
   SELECT SYSTEM_USER() as system_user_name;
   SELECT CURRENT_USER() as current_user_name;
   SELECT SESSION_USER() as session_user_name;
   SELECT LAST_INSERT_ID() as last_insert_id_value;
   SELECT GET_LOCK('test_lock', 10) as lock_status;
   SELECT RELEASE_LOCK('test_lock') as lock_status;
   SELECT MASTER_POS_WAIT(10) as master_position;
   SELECT INET_ATON('192.168.1.1') as ip_address_value;
   SELECT INET_NTOA(3232235777) as ip_address_value;
   SELECT CONVERT('2021-01-01', DATE) as date_value;
   SELECT CONVERT('2021-01-01 12:00:00', DATETIME) as datetime_value;
   SELECT CONVERT('12:00:00', TIME) as time_value;
   SELECT CONVERT('hello', CHAR) as char_value;
   SELECT CONVERT('hello', VARCHAR) as varchar_value;
   SELECT CONVERT('hello', TEXT) as text_value;
   SELECT CONVERT(123, INT) as int_value;
   SELECT CONVERT(123, DECIMAL) as decimal_value;
   SELECT CONVERT(123, FLOAT) as float_value;
   SELECT CONVERT(123, DOUBLE) as double_value;
   SELECT CONVERT(123, BOOLEAN) as boolean_value;
   SELECT CONVERT(123, BINARY) as binary_value;
   SELECT CONVERT(123, VARBINARY) as varbinary_value;
   SELECT CONVERT(123, TINYINT) as tinyint_value;
   SELECT CONVERT(123, SMALLINT) as smallint_value;
   SELECT CONVERT(123, MEDIUMINT) as mediumint_value;
   SELECT CONVERT(123, INT) as int_value;
   SELECT CONVERT(123, BIGINT) as bigint_value;
   SELECT CONVERT(123, DECIMAL) as decimal_value;
   SELECT CONVERT(123, DECIMAL(10,2)) as decimal_value;
   SELECT CONVERT(123, NUMERIC) as numeric_value;
   SELECT CONVERT(123, DATE) as date_value;
   SELECT CONVERT(123, DATETIME) as datetime_value;
   SELECT CONVERT(123, TIME) as time_value;
   SELECT CONVERT(123, YEAR) as year_value;
   SELECT CONVERT(123, CHAR) as char_value;
   JSON函数:
   SET @json = '{"name": "John", "age": 30}';
   SELECT JSON_EXTRACT(@json, '$.name') AS name;  -- 返回 'John'
   SELECTSELECT JSON_UNQUOTE(JSON_EXTRACT(@json, '$.age')) AS age;  -- 返回 '30'
```

压力测试：主要测试软件在高并发、高负载下的表现，如并发量、响应时间、数据库连接数等。 **兼容性测试**：主要测试软件在不同操作系统、不同浏览器、不同版本的浏览器下是否能正常运行。 **安全测试**：主要测试软件在不同攻击方式下是否能正常运行，如SQL注入、XSS攻击、CSRF攻击等。 **单元测试**: 主要测试软件的各个模块是否能正常运行，如数据库连接、业务逻辑、界面显示等。 使用测试框架如Junit、Mocha、PHPUnit等进行单元测试。

```
首先确保自己已经安装了python库:
pip install mysql-connector-python
pip install pymysql
创建测试:
```

```
import unittest
import mysql.connector
from mysql.connector import Error

class TestMySQLDatabase(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        # 连接到 MySQL 数据库
        cls.connection = mysql.connector.connect(
            host="localhost",
            user="caisn",
            password="csn",
            database="test_db"
        )
        cls.cursor = cls.connection.cursor()

        # 创建测试表
        cls.cursor.execute("CREATE TABLE IF NOT EXISTS users (id INT
AUTO_INCREMENT PRIMARY KEY, username VARCHAR(255), email VARCHAR(255))")

    def setUp(self):
        # 在每个测试之前清理数据
        self.cursor.execute("DELETE FROM users")

    def test_insert_user(self):
        # 插入用户
        self.cursor.execute("INSERT INTO users (username, email) VALUES (%s, %s)",
('user_caisn', 'caisn7049@gmail.com'))
        self.connection.commit()

        # 验证插入
        self.cursor.execute("SELECT * FROM users WHERE username = %s",
('user_caisn',))
        result = self.cursor.fetchone()

        self.assertIsNotNone(result)  # 确保结果不为 None
        self.assertEqual(result[1], 'user_caisn')  # 验证用户名
        self.assertEqual(result[2], 'caisn7049@gmail.com')  # 验证邮箱

    def test_multiple_insertions(self):
        # 批量插入用户
        users = [
            ('user1', 'user1@example.com'),
            ('user2', 'user2@example.com'),
            ('user3', 'user3@example.com')
        ]

        self.cursor.executemany("INSERT INTO users (username, email) VALUES (%s, %s)", users)
        self.connection.commit()

        # 验证插入
        self.cursor.execute("SELECT COUNT(*) FROM users")
        result = self.cursor.fetchone()

        self.assertEqual(result[0], 3)  # 确保插入了3个用户

    @classmethod
    def tearDownClass(cls):
        # 清理测试表
        cls.cursor.execute("DROP TABLE IF EXISTS users")
        cls.connection.close()

if __name__ == '__main__':
    unittest.main()
```

回归测试: 主要测试软件的各个模块是否能正常运行，如数据库连接、业务逻辑、界面显示等。 **集成测试**: 主要测试软件的各个模块是否能正常运行，如数据库连接、业务逻辑、界面显示等。 **负载测试**: 主要测试软件在高负载下的表现，如并发量、响应时间、数据库连接数等。 **自动化测试**: 主要测试软件的各个模块是否能正常运行，如数据库连接、业务逻辑、界面显示等。 **手动测试**: 主要测试软件的各个模块是否能正常运行，如数据库连接、业务逻辑、界面显示等。 **数据完整性测试**: 主要测试软件的数据是否符合要求，如数据完整性、数据一致性、数据准确性等。 **性能测试**: 主要测试软件在不同负载下表现，如并发量、响应时间、数据库连接数等。

```
数据库性能测试是一种评估数据库在特定条件下的响应时间, 数据处理能力和资源使用情况的过程.通过性能测试, 可以识别潜在的瓶颈, 优化数据库性能, 确保应用程序能够承载预期的负载, 以下是进行数据库性能测试的一些步骤和方法:
1. 准备测试: 响应时间, 吞吐量, 并发用户数, 系统资源使用情况
2. 准备测试环境:

3.选择性能测试工具:
  这里预备选取: 1)SysBench-是一个多线程的性能测试工具, 专门针对MYSQL数据库的基准测试.特点: 可以测试CPU, 内存, 文件I/O和数据库性能, 包括OLPTP性能基准测试
   2) MYSQL Benchmarks: 是MYSQL自带的基准测试工具, 可以提快速测试SQL查询的行呢个数, 并发用户数, 响应时间等. 特点: 简单易用, 适合小型数据库测试.
   3) MYSQL Enterprise Monitor:
    是MYSQL 提供的商业工具, 用于监控和优化MYSQL数据库的性能, 特点: 提供实时监控, 查询分析和告警系统, 适合企业级应用
```

```
4.设计测试用例:
根据目标设计具体的测试用例,包括:
单个查询性能测试(测试单独SQL查询性能)
并发测试,
压力测试,
负载测试.
基准测试,
5.执行测试: 运行测试用例, 收集性能数据, 分析结果, 评估性能瓶颈.
6.分析结果: 分析测试结果, 确定瓶颈, 优化方案, 再次执行测试.


**冒烟测试**: 主要测试软件的各个模块是否能正常运行，如数据库连接、业务逻辑、界面显示等。
```