DBMS 系统需求分析文档

一、引言

1.1 编写目的

本文档旨在详细描述简易 DBMS 系统的需求，包括功能性需求和非功能性需求，分析系统涉及的角色及其功能，并通过用例图及用例说明来清晰展示系统的行为和交互。该文档将作为系统设计、开发和测试的重要依据。

1.2 项目背景

随着数据量的不断增长和数据管理需求的日益复杂，开发一个简易的数据库管理系统（DBMS）变得十分必要。该系统将为用户提供一个方便快捷的数据管理平台，满足他们在数据存储、查询、更新等方面的基本需求。

1.3 定义、首字母缩写词和缩略语

DBMS：Database Management System，数据库管理系统

DDL：Data Definition Language，数据定义语言

DML：Data Manipulation Language，数据操作语言

二、功能性需求

2.1 数据定义

创建数据库：用户能够创建一个新的数据库，指定数据库名称和相关属性。

创建表：在已有的数据库中创建新的表，定义表的结构，包括字段名、数据类型、主键、外键等约束。

修改表结构：可以对已存在的表进行结构修改，如添加字段、修改字段数据类型、删除字段等。

删除表：能够删除不再需要的表及其相关数据。

删除数据库：删除整个数据库及其包含的所有表和数据。

2.2 数据操作

插入数据：向表中插入新的数据记录，支持单条和批量插入。

查询数据：执行各种查询操作，包括简单查询（基于字段值的过滤查询）、多表关联查询等，能够指定查询条件、选择需要返回的字段。

更新数据：根据指定的条件更新表中的数据记录。

删除数据：按照条件删除表中的数据记录。

2.3 数据管理

数据库备份：将数据库的数据和结构备份到指定的文件中，以便在需要时进行恢复。

数据库恢复：从备份文件中恢复数据库到备份时的状态。

三、非功能性需求

3.1 性能需求

响应时间：在处理小规模数据时，系统能够保持高效响应，查询操作平均响应时间尽可能短，数据更新、插入操作能够快速完成。

吞吐量：系统应能够支持一定数量的并发操作，确保在多用户同时访问时的性能稳定。

3.2 易用性需求

用户界面：设计简洁明了的用户界面，方便用户进行各种操作，降低用户学习成本。

操作流程：系统的操作流程应简单直观，符合用户的使用习惯。

3.3 可靠性需求

数据完整性：确保数据在存储和操作过程中的完整性，防止数据丢失或损坏。

系统稳定性：系统应具备较高的稳定性，能够长时间运行而不出现故障。

3.4 安全性需求

用户认证：对用户进行身份认证，确保只有授权用户能够访问系统。

数据加密：对敏感数据进行加密存储，防止数据泄露。

四、系统角色

4.1 管理员

负责系统的整体管理和维护，包括用户管理、权限分配等。

可以创建、修改和删除数据库、表等对象。

进行数据库的备份和恢复操作。

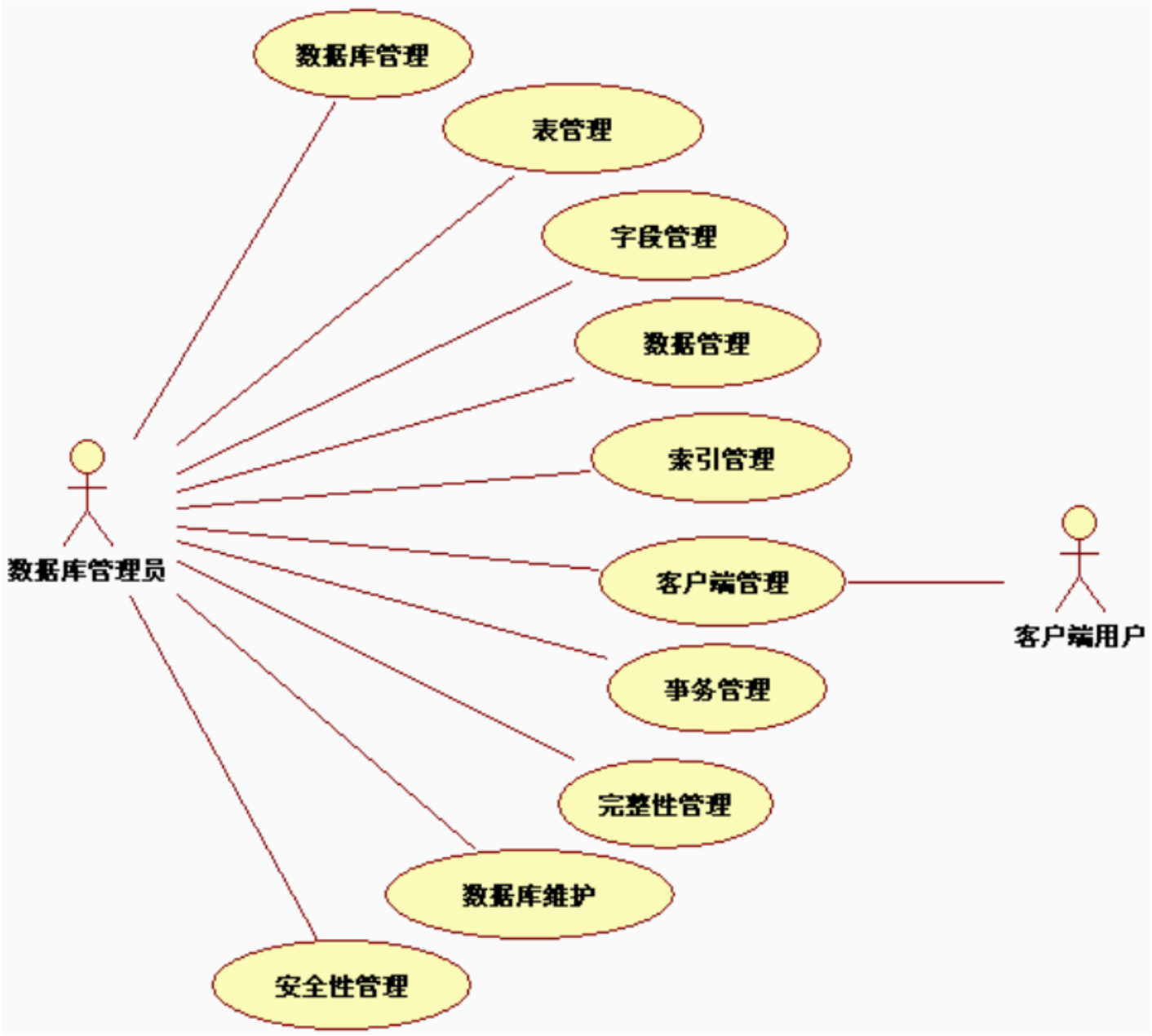
4.2 普通用户

可以在授权的数据库中进行数据的插入、查询、更新和删除操作。

只能访问和操作自己有权限的数据库和表。

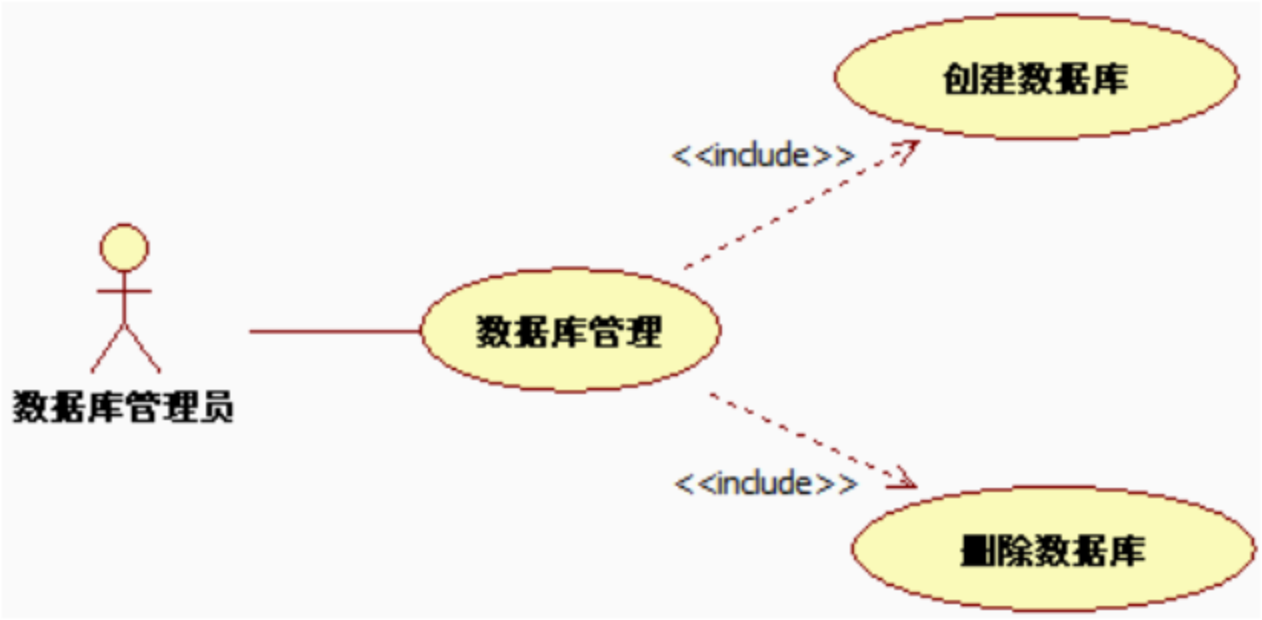
五、用例图

数据库管理系统高层系统用例图：



六、用例说明

6.1 创建数据库



用例名称：创建数据库

参与者：管理员

前置条件：管理员已登录系统

后置条件：成功创建数据库

基本流程：

管理员选择创建数据库功能，输入数据库名称（如 `testDB`）。

系统验证名称合法性（不允许以数字开头、不含特殊字符等）。

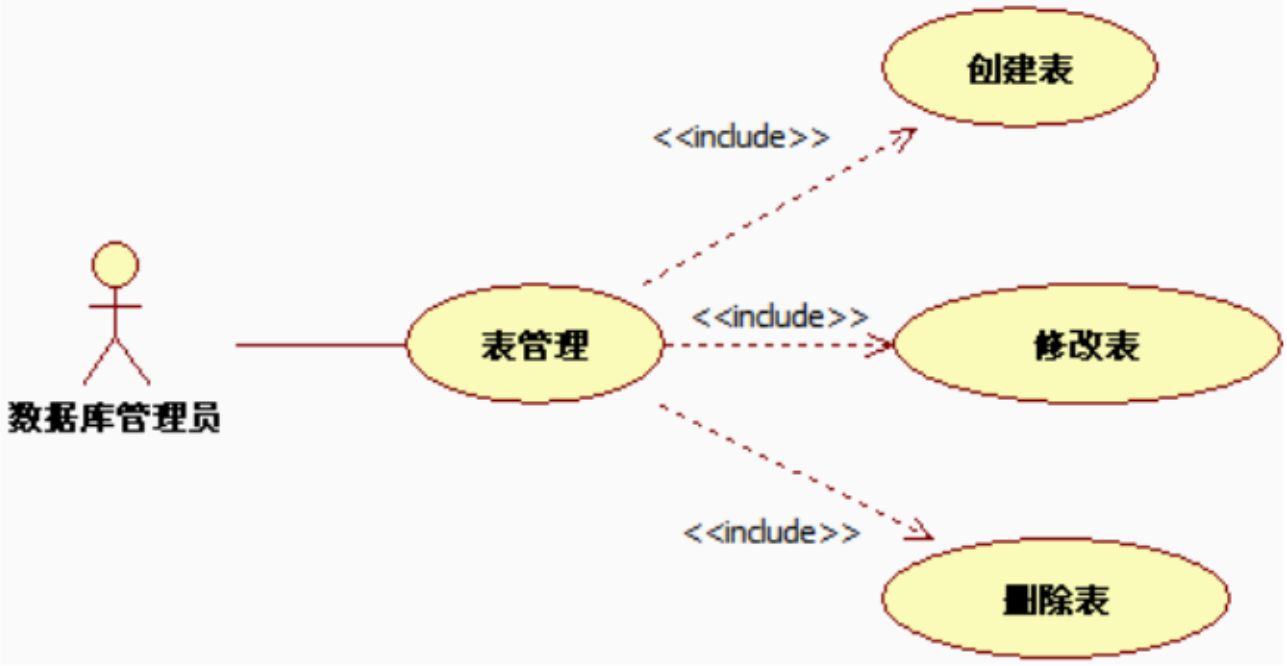
系统创建数据库并记录元数据。

测试语句示例：

```
\-- 正常创建

CREATE DATABASE testDB;&#x20;

SHOW DATABASES; -- 预期结果：包含 testDB

\-- 创建已存在数据库（报错）

CREATE DATABASE testDB;&#x20;

SHOW DATABASES; -- 预期结果：ERROR: 数据库 'testDB' 已存在

\-- 创建无效名称数据库（以数字开头，报错）

CREATE DATABASE 123DB;&#x20;

\-- 预期结果：ERROR: 数据库名称不合法
```

```
查询结果
> CREATE DATABASE testDB;
切换到数据库: testDB
操作成功: 数据库 'testDB' 创建成功

----------------
> CREATE DATABASE test;
切换到数据库: test
操作成功: 数据库 'test' 创建成功

----------------
> CREATE DATABASE testdb1;
切换到数据库: testdb1
操作成功: 数据库 'testdb1' 创建成功
```

6.2 创建表



用例名称：创建表

参与者：管理员

前置条件：已选择数据库（如 testDB ）

后置条件：成功创建表并存储结构信息

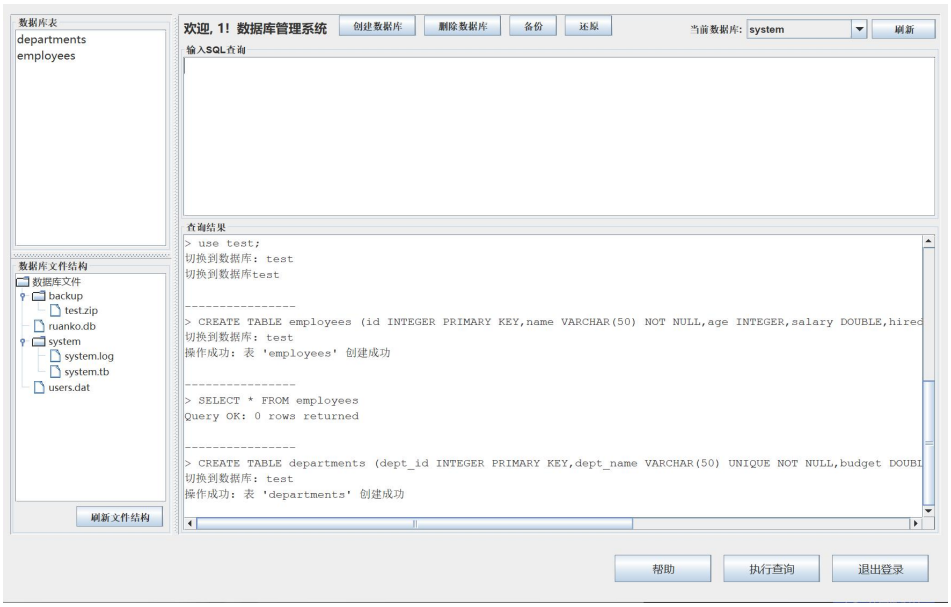
基本流程：

输入表名（如 employees ）和字段定义（含数据类型、约束）。

系统验证表名唯一性和字段合法性（如主键不可重复）。

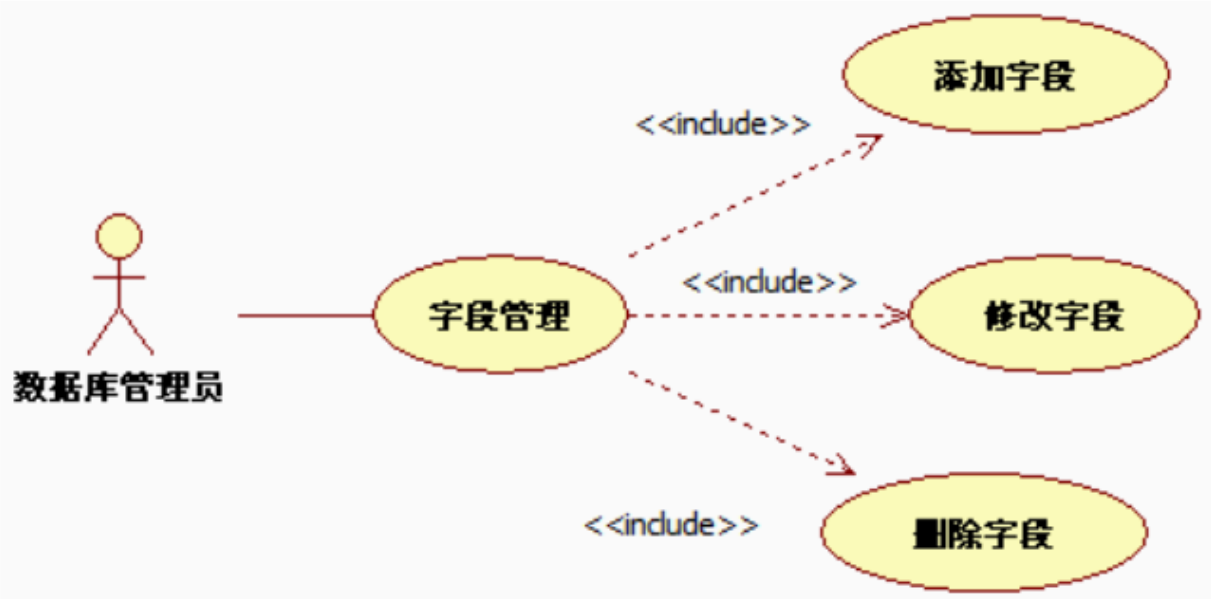
创建表文件（ .tdf 、 .trd 等）并写入结构。

测试语句示例：




```
\-- 正常创建表（含主键、非空约束）
CREATE TABLE employees (
&#x20;   id INTEGER PRIMARY KEY,
&#x20;   name VARCHAR(50) NOT NULL,
&#x20;   age INTEGER,
&#x20;   salary DOUBLE
);
SHOW TABLES; -- 预期结果：包含 employees
\-- 创建同名表（报错）
CREATE TABLE employees (id INTEGER);&#x20;
\-- 预期结果：ERROR: 表 'employees' 已存在
\-- 创建含约束的表
CREATE TABLE departments (
&#x20;   dept\_id INTEGER PRIMARY KEY,
&#x20;   dept\_name VARCHAR(50) UNIQUE NOT NULL
);
```

6.3 插入数据



用例名称：插入数据

参与者：管理员、普通用户

前置条件：已选择表（如 `employees`）

后置条件：数据写入表记录文件（`.trd`）

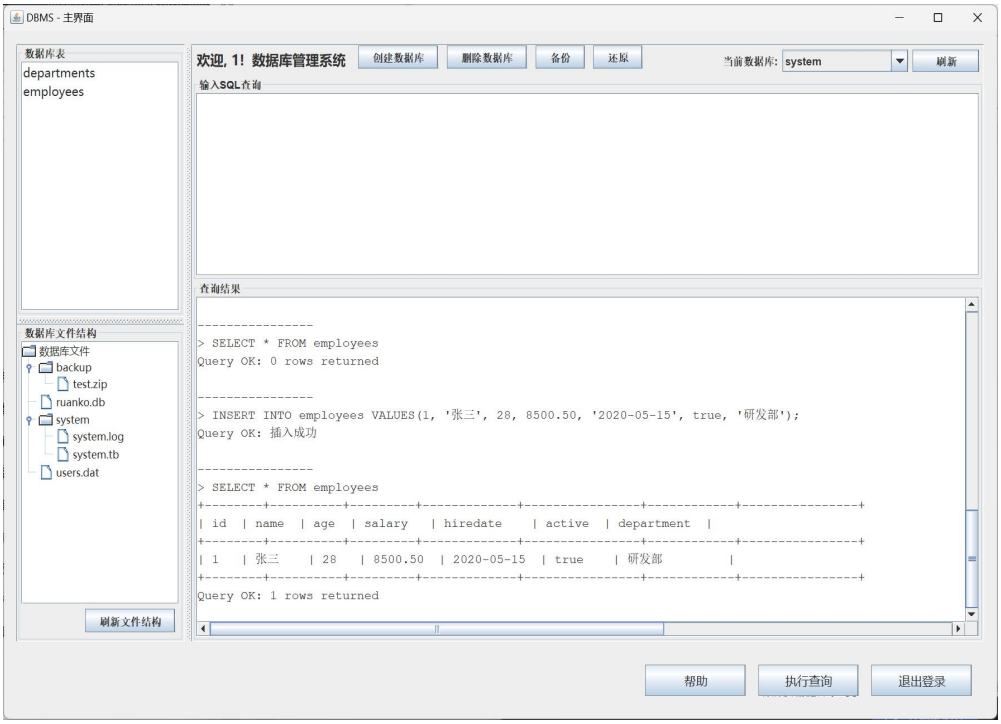
基本流程：

输入数据值，匹配表结构字段顺序或指定字段列表。

系统验证数据类型（如 `INTEGER` 字段不可输入字符串）和约束（如主键唯一）。

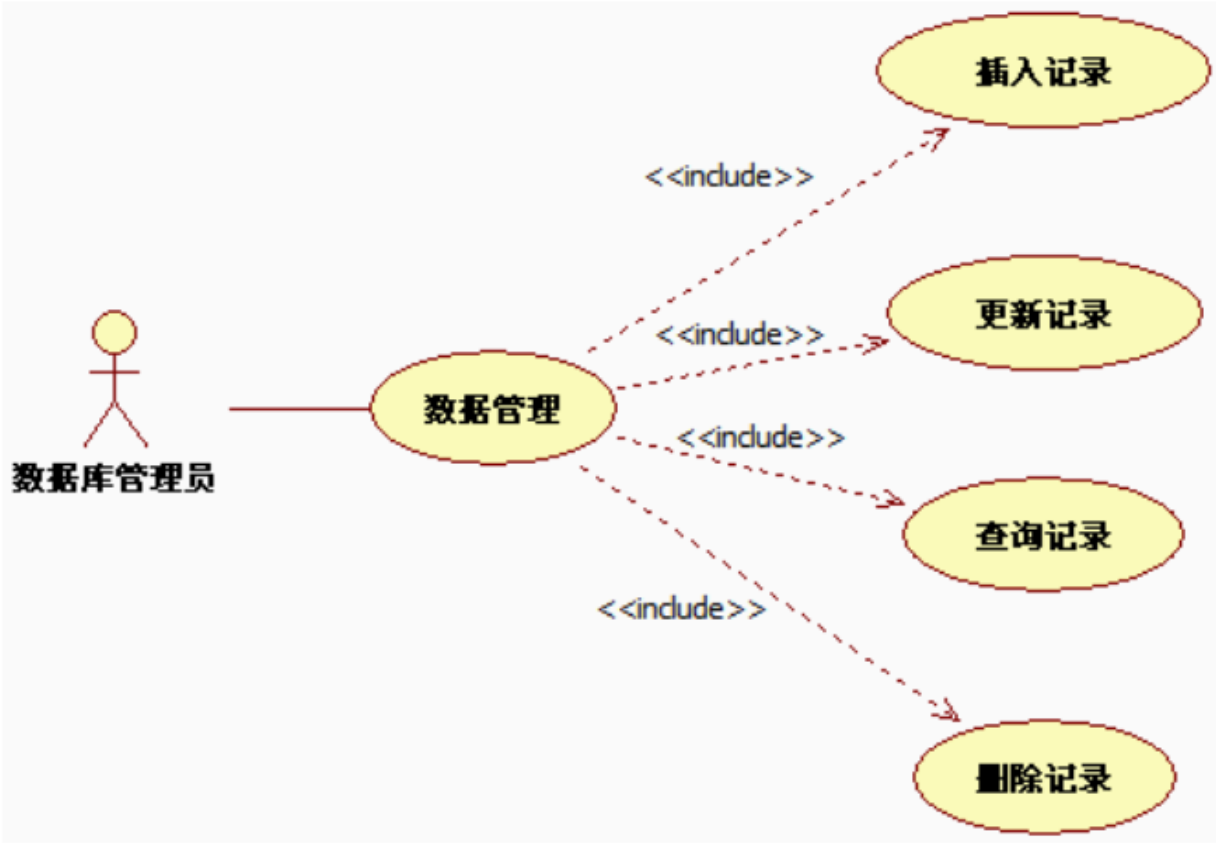
将数据写入文件末尾。

测试语句示例：



```
\-- 插入完整记录（含所有字段）

INSERT INTO employees VALUES (1, '张三', 28, 8500.50);

\-- 插入部分字段（仅指定 id、name）

INSERT INTO employees (id, name) VALUES (2, '李四');

\-- 插入重复主键（报错）

INSERT INTO employees VALUES (1, '王五', 30, 9000.00);&#x20;

\-- 预期结果：ERROR: 主键冲突，值 '1' 已存在
```

6.4 查询数据



用例名称：查询数据

参与者：管理员、普通用户

前置条件：已选择表（如 `employees`）

后置条件：返回符合条件的数据行

基本流程：

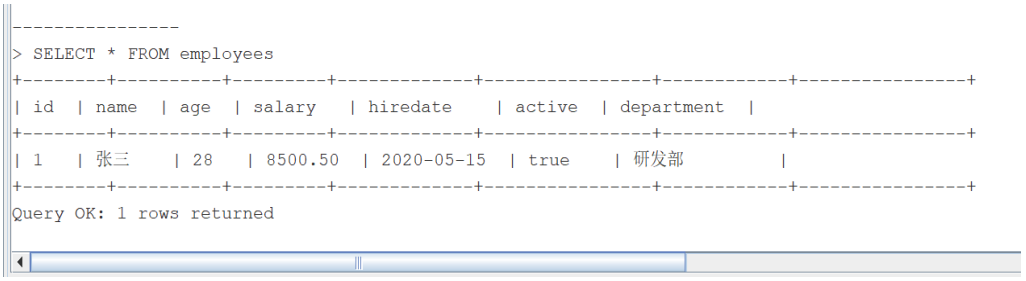
解析查询语句，提取表名、字段、过滤条件。

读取表结构文件（ `.tdf` ）和记录文件（ `.trd` ），按条件过滤数据。

格式化结果并返回（如控制台输出或界面展示）。

测试语句示例：

```
\-- 查询所有数据

SELECT \* FROM employees;

\-- 预期结果：显示所有插入的记录

\-- 带条件查询（年龄大于 30）

SELECT \* FROM employees WHERE age > 30;

\-- 查询不存在的表（报错）

SELECT \* FROM nonexistent\_table;

\-- 预期结果：ERROR: 表 'nonexistent\_table' 不存在
```



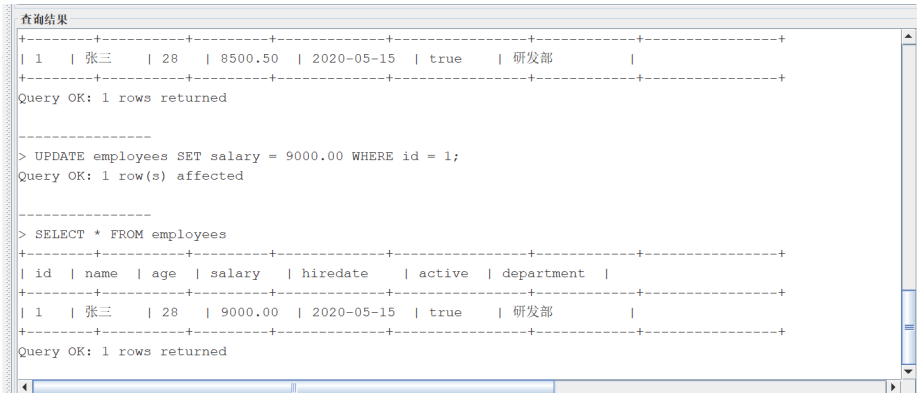
```
----------------
> SELECT * FROM employees
+--------+----------+---------+-------------+----------------+------------+----------------+
| id   | name   | age   | salary    | hiredate     | active   | department   |
+--------+----------+---------+-------------+----------------+------------+----------------+
| 1    | 张三     | 28    | 8500.50   | 2020-05-15   | true     | 研发部          |
+--------+----------+---------+-------------+----------------+------------+----------------+
Query OK: 1 rows returned
```

6.5 更新数据

用例名称：更新数据

参与者：管理员、普通用户

前置条件：已选择表（如 `employees` ）



```
查询结果
+--------+----------+---------+-------------+----------------+------------+----------------+
| 1    | 张三     | 28    | 8500.50   | 2020-05-15   | true     | 研发部          |
+--------+----------+---------+-------------+----------------+------------+----------------+
Query OK: 1 rows returned

----------------
> UPDATE employees SET salary = 9000.00 WHERE id = 1;
Query OK: 1 row(s) affected

----------------
> SELECT * FROM employees
+--------+----------+---------+-------------+----------------+------------+----------------+
| id   | name   | age   | salary    | hiredate     | active   | department   |
+--------+----------+---------+-------------+----------------+------------+----------------+
| 1    | 张三     | 28    | 9000.00   | 2020-05-15   | true     | 研发部          |
+--------+----------+---------+-------------+----------------+------------+----------------+
Query OK: 1 rows returned
```

后置条件：修改表记录文件（ `.trd` ）中的指定行

基本流程：

解析更新语句，提取更新值和过滤条件（如 `WHERE id = 1` ）。

验证条件有效性（如非空字段不可更新为 `NULL` ）。

遍历文件数据，匹配条件行并修改值。

测试语句示例：

```
\-- 正常更新（修改薪资）
UPDATE employees SET salary = 9000.00 WHERE id = 1;
\-- 更新不存在的记录（无影响）
UPDATE employees SET salary = 10000 WHERE id = 999;
\-- 预期结果：操作成功: 0 条记录已更新
\-- 更新违反非空约束（报错）
UPDATE employees SET name = NULL WHERE id = 1;
\-- 预期结果：ERROR: 字段 'name' 不允许为空
```

6.6 删除数据

用例名称：删除数据

参与者：管理员、普通用户

前置条件：已选择表（如 `employees`）

后置条件：从表记录文件（`.trd`）中移除指定行

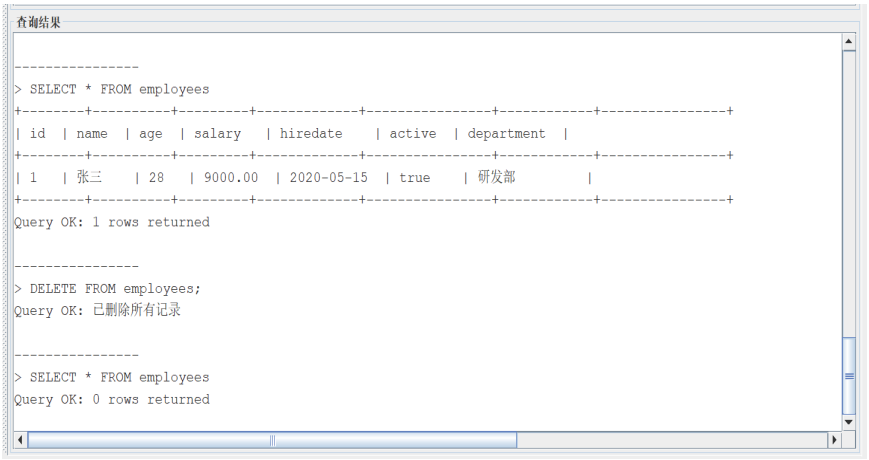
基本流程：

解析删除语句，提取过滤条件（如 `WHERE id = 2`）。

读取数据文件，过滤出需保留的行并重新写入文件。

测试语句示例：

```
\-- 删除指定记录（按主键）

DELETE FROM employees WHERE id = 2;

\-- 删除多条记录（按部门）

DELETE FROM employees WHERE department = '研发部';

\-- 删除不存在的记录（无影响）

DELETE FROM employees WHERE id = 999;

\-- 预期结果：操作成功：0 条记录已删除

\-- 删除所有记录（清空表）

DELETE FROM employees;
```



6.7 数据库备份

用例名称：数据库备份

参与者：管理员

前置条件：已选择数据库（如 `testDB`）

后置条件：生成备份文件（如 `testDB.zip`）

测试语句示例：

```
\-- 执行备份（假设通过命令行指定路径）

BACKUP DATABASE testDB TO 'D:\desktop\save\testDB.zip';

\-- 验证备份（删除数据库后恢复）

DROP DATABASE testDB;

RESTORE DATABASE testDB FROM 'D:\desktop\save\testDB.zip';

SHOW DATABASES; -- 预期结果：testDB 恢复
```

6.8 数据库恢复

用例名称：数据库恢复

参与者：管理员

前置条件：存在有效备份文件

后置条件：数据库恢复至备份时的状态

测试语句示例：

```
\-- 从备份恢复数据库

RESTORE DATABASE testDB FROM 'D:\desktop\save\testDB.zip';

USE testDB;

SHOW TABLES; -- 预期结果：恢复表结构

SELECT \* FROM employees; -- 预期结果：恢复数据记录
```