以下是你提供的内容的中文翻译：

---

1 项目概述

在计算机科学中，“数据库”是一个有组织的数据集合，便于访问、管理和更新。它本质上是一个存储系统，可以存放大量信息，例如学生记录、产品库存或客户信息。最流行的数据库类型是关系型数据库，其中数据以表格形式组织。

你的任务

在这个项目中，你将实现一个迷你数据库管理系统（即 miniDB），支持创建表、更新记录、查询表等功能。你的系统应能够处理我们简化版本的 SQL（即 miniSQL）命令，并输出查询结果。

什么是 SQL?

SQL（结构化查询语言）是管理和操作关系型数据库的标准语言，允许用户创建、读取、更新和删除数据，以及定义和修改数据库结构。在这里，你将使用专门为 miniDB 设计的简化版 SQL。

我们的 miniDB 中的表格是什么样的?

表 1 显示了我们迷你数据库中使用的一张表。每一行是一个“记录”，每一列是一个“字段”。你的数据库需要管理多个表。

表 1：显示学生信息的表

ID	Name	GPA	Major
1000	Jay Chou	3.0	Microelectronics
1001	Taylor Swift	3.2	Data Science
1002	Bob Dylan	3.5	Financial Technology

miniDB 的使用

将你的程序编译成一个名为 minidb 的可执行文件，它接受两个命令行参数：第一个是输入的

SQL 文件，第二个是输出文件。程序的基本使用格式如下：

```
./minidb input.sql output.txt
```

2 miniSQL 语法

2.1 创建数据库和使用数据库

一个数据库包含多个表。在对特定表进行操作之前，用户必须指定他们想要使用的数据库。你的迷你数据库系统将支持基本的数据库管理功能，使用“CREATE DATABASE”和“USE DATABASE”命令。

创建数据库
用户可以使用以下语法创建数据库：

```
CREATE DATABASE database_name;
```

你的 minidb 应能够创建数据库。在退出 minidb 之前，程序应将当前所有表的内容存储到磁盘上的文件中。

在进行任何与表相关的操作（查询、更新、删除等）之前，用户必须指定要操作的数据库。这一步通过“USE DATABASE”语句完成。

使用数据库
“USE DATABASE”命令用于切换到特定数据库，因此随后的命令（如创建表、插入数据或查询）将适用于选定的数据库。

```
USE DATABASE database_name;
```

2.2 创建表

在本项目中，minidb 将支持一个简化的 CREATE TABLE 命令，具有三种数据类型：FLOAT、TEXT 和 INTEGER。此命令允许用户通过指定表名和列定义表。每一列必须有一个唯一的名称和三种支持的数据类型之一。

语法

```
CREATE TABLE table_name (
    column1_name column1_type,
    column2_name column2_type,
    column3_name column3_type,
    ...
);
```

例如，若用户希望创建一张与表 1 相同的表，用户需要输入以下创建表命令：

```
CREATE TABLE student (
    ID INTEGER,
    Name TEXT,
    GPA FLOAT,
    Major TEXT
);
```

2.3 删除表

在本项目中，minidb 还将支持一个简化的 DROP TABLE 命令，允许用户从数据库中删除表及其所有相关数据。一旦表被删除，该表中的所有数据将被永久移除，任何引用已删除表的查询将失败。此命令用于释放数据库空间或删除过时或不需要的表。

语法

```
DROP TABLE table_name;
```

例如，若用户希望删除名为 student 的表，用户将输入以下 DROP TABLE 命令：

```
DROP TABLE student;
```

此命令永久删除 student 表及其所有数据，确保该表不再可访问或查询。

2.4 数据插入

在使用 CREATE TABLE 命令创建表后，可以使用 INSERT INTO 命令向其中插入记录。此命令指定表名，后跟与在表架构中定义的列相同顺序的列值列表。

语法

```
INSERT INTO table_name VALUES (value1, value2, ...);
```

例如，要插入上述学生表中显示的数据，可以使用：

```
INSERT INTO student VALUES (1000, 'Jay Chou', 3.0, 'Microelectronics');
INSERT INTO student VALUES (1001, 'Taylor Swift', 3.2, 'Data Science');
INSERT INTO student VALUES (1002, 'Bob Dylan', 3.5, 'Financial Technology');
```

每个 INSERT INTO 命令将一行数据添加到表中，其值对应于 CREATE TABLE 中定义的列。

2.5 数据查询：基础

假设你有一个学生表，包含以下列和数据：

**ID	Name	GPA	Major **
1000	Jay Chou	3.0	Microelectronics
1001	Taylor Swift	3.2	Data Science
1002	Bob Dylan	3.5	Financial Technology

语法

```
SELECT column1, column2, ... FROM table_name;
```

示例
假设你有如表 1 的学生表。

1. **选择特定列**：如果你想检索所有学生的 Name 和 GPA 列，可以使用：

```
SELECT Name, GPA FROM student;
```

结果：

Name	GPA
Jay Chou	3.0
Taylor Swift	3.2
Bob Dylan	3.5

2. **选择所有列**：要检索表中的所有列，可以使用 * 符号：

```
SELECT * FROM student;
```

结果：

ID	Name	GPA	Major
1000	Jay Chou	3.0	Microelectronics
1001	Taylor Swift	3.2	Data Science
1002	Bob Dylan	3.5	Financial Technology

这个简单的选择根据指定的列检索学生表中的行，只显示你需要的数据。行的顺序可以是任意的。

2.6 数据查询：“WHERE”子句

minidb 中的 WHERE 子句允许你根据特定条件过滤记录，使用基本比较运算符和逻辑连接符。在这个简化版本中，WHERE 子句仅支持以下条件：

- 比较：column > value，column < value，column = value
- 逻辑连接符：AND 和 OR 用于组合多个条件

WHERE 语法

```
SELECT column1, column2 FROM table_name WHERE condition1 AND/OR condition2;
```

示例
假设你有学生表：

1. **单一条件**：要检索 GPA 大于 3.0 的学生姓名，可以使用：

```
SELECT Name FROM student WHERE GPA > 3.0;
```

结果：

Name
Taylor Swift
Bob Dylan

2. **使用 AND 的多个条件**：要查找 GPA 大于 3.0 且主修数据科学的学生，可以使用：

```
SELECT Name FROM student WHERE GPA > 3.0 AND Major = 'Data Science';
```

结果：

Name
Taylor Swift

3. **使用 OR 的多个条件**：要检索 GPA 小于 3.1 或主修金融技术的学生，可以使用：

```
SELECT Name FROM student WHERE GPA < 3.1 OR Major = 'Financial Technology';
```

结果：

Name
Jay Chou
Bob Dylan

通过使用 WHERE 与支持的比较运算符和逻辑连接符，你可以根据特定条件检索数据，使查询既灵活又高效。

2.7 数据查询：“INNER JOIN”子句

INNER JOIN 连接两个表中的记录，基于指定条件。当我们将表 A 的 X 列与表 B 的 Y 列进行 INNER JOIN 时，如果表 A 的 X 列中的值等于表 B 的 Y 列中的值，则表 A 中的每条记录与表 B 中的一条记录相匹配。

只有满足此条件的记录对才会包含在结果中，形成来自两个表的相关数据的组合视图。

换句话说，INNER JOIN 条件有效地链接表 A 和表 B 中的记录，哪里表 A 的 X 与表 B 的 Y 之间有匹配。

语法

```
SELECT table1.column1, table2.column1
FROM table1
INNER JOIN table2
ON table1.X = table2.Y;
```

INNER JOIN 子句根据列之间的匹配条件连接两个表的记录。当我们对表 A

的 X 列和表 B 的 Y 列使用 INNER JOIN 时，表 A 中的每条记录与表 B 中的记录配对，其中 X 和 Y 的值相等。仅满足此条件的记录会包含在结果中，提供来自两个表的相关数据的组合视图。

在使用 INNER JOIN 组合多个表的数据时，必须通过使用表名后跟点和列名来指定每列所属的表。这被称为完全限定的列名，有助于避免模糊，特别是在两个表都有相同名称的列时。

例如，如果学生和课程注册表都有名为 StudentID 的列，则使用 student.StudentID 和 course_enrollment.StudentID 可以清楚地知道每列来自哪个表。

示例
假设你有两个表，学生和课程注册，如下所示：

StudentID	Name
1	Jay Chou
2	Taylor Swift
3	Bob Dylan

StudentID	Name
4	Omnipotent Youth Society

要检索每个学生的姓名及其注册的课程，我们可以在 StudentID 列上将学生和课程注册表进行 INNER JOIN：

```
SELECT student.Name, course_enrollment.Course
FROM student
INNER JOIN course_enrollment ON student.StudentID = course_enrollment.StudentID;
```

结果：

StudentID	Course
1	Microelectronics
2	Data Science
2	Machine Learning
3	Financial Technology
4	Mathematics

姓名和课程的组合：

Name	Course
Jay Chou	Microelectronics
Taylor Swift	Data Science
Taylor Swift	Machine Learning
Bob Dylan	Financial Technology
Omnipotent Youth Society	Mathematics

在这个示例中，结果包括每对在两个表中 StudentID 值相等的记录，显示每个学生的姓名及其注册的课程。

通过 INNER JOIN，学生和课程注册之间的每个匹配基于 StudentID 在最终输出中产生一行，显示 INNER JOIN 如何跨表组合相关数据。

2.8 数据更新

SQL 中的 UPDATE 命令用于修改表中现有记录。在 minidb 中，UPDATE 命令允许你根据与 WHERE 子句指定的条件更改特定列中的值。这有助于确保仅更新某些行，而不是修改表中的每一行。

UPDATE 语法

```
UPDATE table_name
SET column1 = new_value1, column2 = new_value2, ...
WHERE condition;
```

- **table_name**：要更新记录的表的名称。
- **SET**：指定要更新的列及其新值。
- **WHERE**：（可选）指定条件以确定哪些行被更新。如果没有 WHERE，将更新表中的所有行。

示例
假设你有一个学生表：

1. **更新特定行**：如果你想将 Jay Chou 的 GPA 更新为 3.6，你可以指定一个 WHERE 条件来仅针对他的记录：

```
UPDATE student
SET GPA = 3.6
WHERE Name = 'Jay Chou';
```

结果：

ID	Name	GPA
1000	Jay Chou	3.6
1001	Taylor Swift	3.2

ID	Name	GPA
1002	Bob Dylan	3.5

2. **更新多行**：要将所有 GPA 小于 3.5 的学生的 GPA 增加 0.1，可以使用以下命令：

```
UPDATE student
SET GPA = GPA + 0.1
WHERE GPA < 3.5;
```

结果：

ID	Name	GPA
1000	Jay Chou	3.6
1001	Taylor Swift	3.3
1002	Bob Dylan	3.5

在这个示例中，只有满足 WHERE 条件的记录被更新，这使得 UPDATE 命令在目标更改时非常强大。如果没有 WHERE 子句，则表中的所有行都将使用指定的新值进行更新。

2.9 数据删除

SQL 中的 DELETE 命令用于从表中删除记录。与 UPDATE 类似，DELETE 命令可以使用 WHERE 子句来指定条件，从而允许你仅删除某些行，而不是表中的所有行。

DELETE 语法

```
DELETE FROM table_name
WHERE condition;
```

- **table_name**：要从中删除记录的表的名称。
- **WHERE**：（可选）指定条件以确定哪些行被删除。如果没有 WHERE，则会删除表中的所有行。

使用 WHERE 的 DELETE 示例

假设你想删除学生表中 GPA 小于 3.0 的记录：

```
DELETE FROM student
WHERE GPA < 3.0;
```

只有满足 WHERE 条件的行会被删除。如果没有 WHERE 子句，则学生表中的所有行都将被移除。

3 输出格式

每个查询（SELECT 语句）的结果将以 CSV 格式打印，遵循以下规则：

1. **输出格式**：每个结果以 CSV 格式打印，字段之间用逗号分隔。
2. **标题行**：输出的第一行将包含列名，也用逗号分隔。
3. **文本字段**：文本字段用双引号（""）括起来。数据中不会包含含有双引号的文本字段，因此无需进一步格式化。
4. **数字字段**：
    - 整数字段原样打印。
    - 浮点字段打印为两位小数，必要时四舍五入。

minidb 执行以下 SELECT 查询：

```
SELECT ID, Name, GPA FROM student;
```

并返回以下数据：

ID	Name	GPA
1000	Jay Chou	3.00
1001	Taylor Swift	3.20
1002	Bob Dylan	3.50

CSV 格式的输出将是：

```
ID,Name,GPA
1000,"Jay Chou",3.00
1001,"Taylor Swift",3.20
1002,"Bob Dylan",3.50
```

3.1 示例 1

这是一个示例 input.sql 和相应的 output.csv，基于简化的 miniSQL 格式。

input.sql

```
CREATE DATABASE db_university;
USE DATABASE db_university;
CREATE TABLE student (
    ID INTEGER,
    Name TEXT,
    GPA FLOAT
);
INSERT INTO student VALUES (1000, 'Jay Chou', 3.0);
INSERT INTO student VALUES (1001, 'Taylor Swift', 3.2);
INSERT INTO student VALUES (1002, 'Bob Dylan', 3.5);
SELECT ID, Name, GPA FROM student;
SELECT ID, Name, GPA FROM student WHERE GPA > 3.1;
```

output.csv

```
ID,Name,GPA
1000,"Jay Chou",3.00
1001,"Taylor Swift",3.20
1002,"Bob Dylan",3.50
---
ID,Name,GPA
1001,"Taylor Swift",3.20
1002,"Bob Dylan",3.50
```

在这个 output.csv 中：

- 每个查询结果之间用一行（---）分隔，以清楚区分连续查询的输出。

- 文本字段用双引号括起来，而整数和浮点数直接打印，浮点数格式为两位小数。

3.2 示例 2

在执行示例 1 后，minidb 应将 db_university 的信息序列化到磁盘。现在，当 minidb 读取 SQL 语句“use database”时，需要加载先前的数据库内容。

这是一个示例 input2.sql 和相应的 output2.csv，基于简化的 miniSQL 格式。

input2.sql

```
USE DATABASE db_university;
SELECT ID, Name, GPA FROM student WHERE GPA > 3.1;
```

output2.csv

```
ID,Name,GPA
1001,"Taylor Swift",3.20
1002,"Bob Dylan",3.50
```

4 错误报告

minidb 应能够检查 miniSQL 的语法，并报告发生错误的行号。

如果有任何其他具体需求或调整，请告诉我！