مدرسان: رامتین خسروی، محمدامین صادقی
طراحان: مهسا قزوینی‌نژاد، محمدرضا کیانی

مهلت تحویل: یک‌شنبه ۱۶ مهر ۱۳۹۶، ساعت ۲۳:۵۵

پایگاه‌داده‌ی[1] خانگی

در این تمرین قرار است یک پایگاه داده‌ی ساده را طراحی و پیاده سازی کنید. برای این کار شما با ساختارهای vector و string در زبان C++ آشنا خواهید شد. همچنین مهارت‌هایی مربوط به شکستن یک پروژه‌ی بزرگ به قسمت‌های کوچکتر و همچنین تکنیک برنامه‌سازی بالا به پایین در این پروژه به شما نشان داده خواهند شد.

در ابتدای ایجاد رایانه‌ها هدف از طراحی آن‌ها تنها انجام محاسبات سنگین (Computation) بوده است، اما رفته رفته با گسترش استفاده از رایانه‌ها آن‌ها در ابعاد مختلفی از صنایع مختلف وارد شدند. یکی از مهمترین دغدغه‌های فعلی جمع‌آوری، ثبت، نگهداری و استفاده از اطلاعات است.

در طول سال‌ها روش‌های متفاوتی برای این کار طراحی شده است که هرکدام ویژگی‌های خاص خود را دارند. یکی از ساده‌ترین روش‌ها نگه‌داری اطلاعات، نگه‌داری آن‌ها در فایل‌ها و بر روی دیسک است. در این روش برنامه‌نویس باید ساختار نگه‌داری اطلاعات خود را طراحی و پیاده‌سازی کند. اما نیازمندی‌های مهم دیگری در طول سال‌ها باعث به وجود آمدن روش‌های پیچیده‌تری برای مدیریت اطلاعات شده است. حجم بالای اطلاعات و نیاز به مدیریت آن، آسانی دسترسی، ایجاد، تغییر و پاک‌کردن اطلاعات، مسائل امنیتی و کنترل دسترسی، اطمینان از صحت، ناقص و یا متناقض نبودن اطلاعات، امکان دسترسی همزمان کاربران و ACID[2] از جمله‌ی این نیازمندی‌ها هستند.

از این‌رو پایگاه‌داده‌ها به وجود آمدند. پایگاه‌داده به مجموعه‌ای از داده‌ها با ساختار مشخص گفته می‌شود. پایگاه‌های داده انواع مختلفی از جمله Relational و NoSQL دارند. در این تمرین هدف پیاده‌سازی یک پایگاه‌های داده‌ی رابطه‌ای بسیار ساده است. اطلاعات در این پایگاه‌های داده به صورت ساختارمند در قالب جدول‌هایی نگه‌داری می‌شوند.

سیستم مدیریت پایگاه داده (DBMS[3]) مسئولیت ایجاد دسترسی به این اطلاعات و مدیریت آن‌ها را به عهده دارد. از معروف‌ترین DBMS ها می‌توان به PostgreSQL و MySQL اشاره کرد.

هر DBMS از طریق زبانی استاندارد با پایگاه‌داده‌ها ارتباط برقرار میکند. یکی از مطرح‌ترین این زبان‌ها، زبان SQL[4] است که از چند دستور ساده اما اصولی تشکیل‌شده و با استفاده از آن‌ها می‌توان کلیه عملیات های مورد نیاز را انجام داد.

در این تمرین قصد داریم یک DBMS ساده مبتنی بر زبان SQL را شبیه سازی کنیم.

ساختار پایگاه‌داده

هر پایگاه‌داده محل نگهداری مجموعه‌ای از جدول‌هاست. هر جدول علاوه بر نام خاص خود، شامل ستون‌هایی است که ساختار آن را تشکیل می‌دهند. همچنین اطلاعات در این جدول در سطرها (یا رکوردها) ذخیره می‌شود.

ستون‌ها در واقع محتویات موجود در پایگاه‌داده ما را مشخص می‌کنند و توضیحاتی در مورد آن‌ها ارائه می‌کنند. هر ستون شامل یک نام و نوع‌داده‌ای است که می‌تواند در آن ستون جای بگیرد. SQL انواع مختلفی از داده را پشتیبانی می‌کند مانند int، text، timestamps و ... (شما در این پروژه، می‌توانید فرض‌کنید که تمامی داده‌ها از نوع text هستند.)

سطرها یا رکوردها نیز در واقع داده‌های ما، در یک پایگاه‌داده هستند.

برای مثال در پایگاه‌داده مورد نیاز برای نگه‌داری اطلاعات یک فروم، جدول‌هایی مانند users , posts نگه‌داری می‌شوند. که هر یک از این جدول‌ها تمامی رکوردهای مربوط به ماهیت خود را نگه‌داری می‌کنند. برای مثال جدول users تمامی کاربران سایت و اطلاعاتشان را در خود جای می‌دهد. همان طور که در شکل مشاهده می‌کنید این جدول شامل سطرها و ستون‌هایی می‌باشد.

---

[1] Database

[2] https://en.wikipedia.org/wiki/ACID

[3] Database Management System

[4] Structured Query Language

id	name	family
۱	Hasan	Kachal
۲	Mohammad Reza	Kiani
۳	Mahsa	Ghazvini Nejad

این جدول شامل ۳ ستون با نام‌های id و name و family است و همان‌طور که مشاهده می‌کنید دارای ۳ رکورد(سطر) است که داده‌های هر رکورد منطبق بر ترتیب ستون‌های جدول اند. برای مثال چیزی که در رکورد اول این جدول می‌بینیم، فردی با id برابر با ۱ است که name آن Hasan و family اش Kachal می‌باشد.
بنابراین با کمک این جدول می‌توانیم اطلاعات مورد نظر را از کاربرهای یک سایت ذخیره کنیم و همین‌طور با جداول مشابه برای موضوع‌های مختلف می‌توانیم کل اطلاعات و داده‌های مورد نیازمان را به صورت ساختار یافته نگه‌داری کنیم.

دستورات SQL

دستورهایی که شما باید در این پروژه پیاده‌سازی کنید از دستورهای ساده و کاربردی SQL محسوب می‌شوند.
هر یک از command ها از حداکثر ۴ قسمت نوع دستور، نام جدول موردنظر ، conditions (در صورت وجود) و values (در صورت وجود) تشکیل‌شده‌اند.

برای مثال :

```
SELECT * FROM users WHERE name = family;
```

دستور بالا از نوع دستور SELECT می‌باشد که به جدول users اشاره می‌کند و conditions آن name = family می‌باشد.

```
CREATE TABLE users (id text, name text, family text);
```

و دستور بالا نیز از نوع دستور CREATE است که به جدول users اشاره دارد.
همچنین values آن برابر با id text , name text, family text است.
حال به توضیح دقیق ساختار این دستورها می‌پردازیم:

دستور CREATE TABLE

این دستور برای ساخت یک جدول استفاده می‌شود و در آن ساختار جدول نظیر نام ستون‌ها را مشخص میکنیم(نوع داده برای ما اهمیتی ندارد و ما در این پروژه همه را text فرض میکنیم)
ساختار این دستور به صورت زیر است:

```
CREATE TABLE <table_name> (
	<column_name_1> <data_type>,
	<column_name_2> <data_type>,
	....
);
```

که در آن <table_name> نام جدول، <column_name_1> نام ستون اول و <data_type> نوع داده است که در این پروژه، همیشه مقدار آن برابر text خواهد بود.

- دقت کنید که تمام دستور بالا **در یک خط ورودی استاندارد** وارد می شوند و نمایش بالا تنها برای خوانایی بیشتر است.
- برای درک بیشتر به دستور زیر دقت کنید که هدف آن ساخت جدولی به نام users و با ستون های id, name, family می‌باشد:

```
CREATE TABLE users (id text, name text, family text);
```

- خروجی این دستور عبارت زیر می‌باشد:

```
Query OK
```

دستور INSERT

این دستور برای افزودن یک رکورد جدید(سطر جدید) به یک جدول خاص استفاده می‌شود.
شکل کلی آن به صورت زیر است:

```
INSERT INTO <table_name> VALUES (<value1>, <value2>, <value3> ,... );
```

- تعداد value ها به اندازه تعداد ستون های جدول مورد نظر می‌باشد و باید به ترتیب در جدول وارد شوند.
- به عنوان مثال به دستور زیر دقت کنید که هدف از آن افزودن یک رکورد به جدول users است که دارای مقدار 1 برای ستون id، مقدار Mohammad Reza برای ستون name و مقدار Kiani برای ستون family می‌باشد:

```
INSERT INTO users VALUES ( “1” , “Mohammad Reza” , “Kiani” );
```

- چون در این پروژه تمامی داده‌ها از نوع text در نظر گرفته‌شده‌اند، در این قسمت value ها باید درون «" "» (دابل کوتیشن) قرار گیرند.
- خروجی این دستور مانند دستور CREATE TABLE می‌باشد.

دستور SELECT

این دستور برای انتخاب گروهی از سطرهای یک جدول استفاده می‌شود و خروجی آن نیز یک جدول است که به آن جدول نتیجه می‌گوییم و باید در خروجی استاندارد چاپ شود.
شکل کلی این دستور به یکی از دو شکل زیر است:

```
SELECT * FROM <table_name>;
SELECT * FROM <table_name> WHERE <conditions>;
```

- در این دستور می‌توانیم مشخص کنیم که چه ستون‌هایی از جدول مبدأ در جدول پاسخ نمایش‌داده شوند. «*» به معنی انتخاب تمامی ستون‌های یک جدول برای نمایش در جدول نتایج است.همچنین می‌توان به جای «*» نام ستون‌های مورد نظر را مشخص کرد.. اما برای سادگی از این ویژگی صرف نظر کرده و همواره تنها از حالت انتخاب تمامی ستون‌ها («*») پشتیبانی می‌کنیم.
- قسمت conditions خود به صورت زیر است:

```
WHERE <value1> = <value2> , <value3> = <value4> , … ;
```

- قسمت‌های conditions با «,» از هم جدا شده‌اند[5] و تنها سطرهایی از داده‌های جدول انتخاب شده و در جدول نتیجه جای خواهد گرفت که تمامی شرط‌های این قسمت بر روی آن سطر، برقرار باشد. (در واقع می‌توانید تصور کنید که شرط های آمده در قسمت conditions با یک‌دیگر AND شده‌اند.)
- در این قسمت، هر یک از value ها می‌توانند نام یکی از ستون‌ها و یا یک مقدار باشند(در صورتی که مقدار ثابت مد نظر باشد، value درون «" "» (دابل کوتیشن) قرار دارد)
- برای درک بیشتر به مثال های زیر دقت کنید:

```
SELECT * FROM users;
```

انتخاب تمام رکورد های جدول users

```
SELECT * FROM users WHERE name=”Hasan”;
```

انتخاب تمام رکورد های جدول users که مقدار ستون name آن ها برابر Hasan می‌باشد.

```
SELECT * FROM users WHERE “Hasan” =name;
```

دقیقا معادل همان دستور بالا می‌باشد.

```
SELECT * FROM users WHERE name = family;
```

انتخاب تمام رکورد های جدول users که مقدار ستون name آن‌ها با مقدار ستون family آن‌ها برابر است.

```
SELECT * FROM users WHERE name = “Hasan” , family = “Kachal”;
```

انتخاب تمام رکورد های جدول users که مقدار ستون name آن‌ها برابر Hasan و مقدار ستون family آن‌ها برابر Kachal است.

- خروجی این دستور نمایش جدولی می‌باشد که در ادامه فرمت آن توضیح داده خواهد شد.

نکات

- دقت کنید که زبان SQL نسبت به کوچک و یا بزرگ بودن دستورات حساس نیست ولی برای سادگی فقط دستورات با حروف بزرگ را پشتیبانی کنید.
- در پایان تمام دستورات SQL، کاراکتر «;» (سمی کالن) می آید.
- دقت کنید که برنامه شما نباید به فاصله بین کلمات مختلف در دستورات (مثلا بین SELECT و *) حساس باشد و قسمت‌های مختلف دستورات ممکن است با هر تعداد white space از هم جدا شوند.
- تضمین می‌شود که تمامی دستورات مطابق فرمت داده شده باشند.

[5] در SQL از عبارت‌های AND و OR برای اتصال شرایط WHERE به یکدیگر استفاده می‌شود و در این جا هدف از جایگزینی آن‌ها با کاما صرفا ساده شدن پروژه است.

مراحل پروژه

با توجه به اینکه این تمرین، نخستین تمرین کامپیوتری شماست، فرآیند طراحی و پیاده‌سازی را به صورت مرحله به مرحله توضیح خواهیم داد. پیاده‌سازی هر مرحله نمره جداگانه‌ای دارد و قویاً توصیه می‌کنیم که شما نیز مطابق با همین روند، برنامه خود را توسعه دهید.

مرحله اول

در این مرحله از شما میخواهیم که ساختار کلی پایگاه‌داده و جدول‌های خود را با استفاده از struct های مورد نیاز طراحی کنید و برای عملیات‌های ایجاد جدول دلخواه و اضافه کردن داده به یک جدول توابعی را پیاده‌سازی نمایید. در این مرحله، نیازی به خواندن دستور ها با ساختار گفته‌شده از ورودی استاندارد[6] نیست. شما تنها کافی‌است بتوانید از داخل تابع main قابلیت کار با جدول‌ها را به کمک این توابع داشته‌باشید.

برای مثال، شما باید بتوانید با فراخوانی تابعی که نام جدول و مشخصات ستون‌های آن را با ساختار متناسب دریافت می‌کند، جدول جدیدی بسازید. همچنین، باید بتوانید با فراخوانی تابع دیگری که نام جدول و اطلاعات یک رکورد جدید را به عنوان ورودی دریافت می‌کند، رکورد جدیدی به جدول اضافه کنید.

برای تست این قسمت شما می‌توانید جدولی با نام users را که دارای ستون‌های id , name , family است با فراخوانی تابع مورد نظر در داخل main ایجاد کرده و سپس به این جدول سه رکورد از مشخصات خود یا اطرافیانتان اضافه کنید.

توجه) ساختار برنامه‌ی شما باید به گونه‌ای باشد که که به راحتی و تنها با تغییری جزئی در تابع main بتوان جدول جدید با ساختاری کاملا متفاوت ایجاد و رکورد های جدیدی را در این جدول ذخیره‌سازی کرد.

برای مثال شبه کد زیر شامل توابعی برای ایجاد جدول و افزودن رکورد به آن است. دقت کنید که دلیلی ندارد اسم توابع شما و نحوه ورودی گرفتنشان مانند شبه کد زیر باشد.

```
main(){
	...
	create_table("users",...);
	insert("users",...);
	insert("users",...);
	...
}
```

مرحله دوم

در این مرحله باید تابعی کمکی را پیاده‌سازی نمایید که با گرفتن یک جدول به عنوان ورودی، اطلاعات کلی جدول را در خروجی استاندارد چاپ کند. این اطلاعات شامل نام جدول و تعداد سطر های جدول می‌باشد (دقت کنید که ستون های یک جدول، یک سطر به حساب نمی آیند). فرمت خروجی شما باید به این شکل باشد:

```
Table <table_name> has columns [<col1>, <col2>,...] and #<number_of_rows> rows of data
```

برای مثال:

```
Table users has columns [id, name, family] and #3 rows of data
```

به این ترتیب می‌توانید از این تابع برای تست و اطمینان از درستی برنامه‌ی خود استفاده نمایید.

[6] STDIN

مرحله سوم

در این مرحله تابعی برای عملیات انتخاب اطلاعات از یک جدول (دستور SELECT) را پیاده‌سازی کنید. این تابع باید با گرفتن جدول و condition های مورد نظر (که توضیح داده شدند) جدولی را به عنوان جدول خروجی ایجاد کند که شامل تمام سطرهایی از جدول اصلی است که شرایط موجود در قسمت conditions برروی آن سطر صدق می‌کند. برای تست این تابع از تابع کمکی مرحله‌ی قبل استفاده نمایید.

مرحله چهارم

شما تا این قسمت برای اجرای دستورات SQL، توابع اجرایی آن‌ها را از داخل تابع main فراخوانی می‌کردید. در این مرحله باید توابع لازم برای خواندن و پردازش این سه دستور SQL(مطابق قالب ذکر شده) از ورودی استاندارد را پیاده‌سازی کرده و سپس توابع متناسب با این دستورهارا اجرا نمایید.

مرحله پنجم

در این مرحله باید تابعی را پیاده سازی کنید که یک جدول را به عنوان ورودی دریافت کرده و کلیه اطلاعات جدول نظیر ستون‌ها و سطرهای موجود در آن را به شکلی جذاب[7] که در ادامه توضیح داده می‌شود در خروجی استاندارد چاپ کند.

خروجی شما باید مانند شکل زیر باشد که خروجی جدول users می‌باشد:

```
+----+---------+-------------+
| id | name    | family      |
+----+---------+-------------+
| 1  | Hasan   | Kachal      |
| 2  | Felfeli | Gol Baghali |
| 3  | Koofte  | Ghelgheli   |
+----+---------+-------------+
3 rows in set
```

نکات

- همان طور که ملاحظه می‌کنید هر جدول شامل ۳ بخش می‌باشد:
  ۱. نام ستون‌های جدول است که هر یک با کاراکتر pipe line (|) و فاصله ای مناسب از هم جدا شده‌اند.
  ۲. سطر های جدول می‌باشند که داده‌های هر ستون منطبق بر ستون‌های جدول اند.
  ۳. تعداد سطر های جدول را نشان میدهد که به یکی از ۳ شکل زیر است:

```
<number_of_rows> rows in set // if number_of_rows > 1
<number_of_rows> row in set // if number_of_row = 1
Empty set // if number_of_row = 0
```

- دقت کنید که در حالت آخر دیگر جدولی نباید کشیده‌شود و تنها چاپ عبارت فوق کافی است.

- عرض ستون‌ها باید به اندازه‌ای باشد که جدول به شکلی منظم دیده‌شود. یعنی ابتدا شما باید در میان داده‌های مربوط به هر ستون و البته نام هر ستون، طولانی‌ترین رشته را بیابید و عرض ستون را با توجه به آن لحاظ کنید. دقت کنید که هر داده باید با هر یک از کاراکتر های pipe line دور خود، حداقل یک اسپیس فاصله داشته باشد.

[7] Pretty Print

- همان طور که مشاهده میکنید ابتدا و انتهای جدول و در بین نام ستون‌ها و اولین سطر داده، یک خط افقی قرار دارد که از کاراکتر های dash (ـ) و کاراکتر plus (+) تشکیل شده است. کاراکتر + تنها در مرز ستون‌ها باید استفاده شوند.

- دقت کنید که در صورت پیاده سازی این مرحله دیگر به تابع مرحله دوم نیازی **ندارید** و این تابع آن را در بر می‌گیرد.

ورودی و خروجی نمونه

ورودی	خروجی
`CREATE TABLE users (id text, name text , family text);` `INSERT INTO users VALUES (“1”, “Hasan”, “Kachal”);` `INSERT INTO users VALUES (“2”,“Mohammad Reza”, “Kiani”);` `INSERT INTO users VALUES (“3”,“Mahsa”,“Ghazvini Nejad”);` `SELECT * FROM users;` `SELECT * FROM users WHERE id = “1”;` `SELECT * FROM users WHERE id = “1” , name = family;`	`Query OK` `Query OK` `Query OK` `Query OK` `+----+---------------+----------------+` `| id	name	family |` `+----+---------------+----------------+` `| 1	Hasan	Kachal |` `| 2	Mohammad Reza	Kiani |` `| 3	Mahsa	Ghazvini Nejad |` `+----+---------------+----------------+` `3 rows in set` `+----+-------+--------+` `| id	name	family |` `+----+-------+--------+` `| 1	Hasan	Kachal |` `+----+-------+--------+` `1 row in set` `Empty set`

نحوه‌ی تحویل

فایل برنامه‌ی خود را با نام A1-SID.cpp در صفحه‌ی CECM درس بارگذاری کنید که در آن SID شماره‌ی دانشجویی شماست.

- برنامه‌ی شما باید در سیستم‌عامل لینوکس و با مترجم g++ با استاندارد c++98 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- از صحت فرمت ورودی‌ها و خروجی‌های برنامه‌ی خود مطمئن شوید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.