❋ فاز دوم و سوم پروژه SQL

❋ درس: پایگاه داده ❋ استاد: دکتر فرضی

❋ موعد بارگذاری فاز دوم: ۱۴۰۰/۱۰/۱۰

❋ موعد بارگذاری فاز سوم: ۱۴۰۰/۱۱/۰۷

✓ **فاز دوم** این پروژه شامل دو بخش است، بخش اول مربوط به نوشتن دستورات SQL و بخش دوم مربوط به اتصال پایگاه داده به برنامه‌ای است که تنها شامل لایه Data باشد.

✓ **فاز سوم** این پروژه شامل پیاده‌سازی لایه Business و Presentation (یا UI) بوده و انجام آن اختیاری است.

توجه فرمایید شما باید روی پایگاه داده‌ای که پیش‌تر در فاز اول پروژه طراحی نموده‌اید تغییرات لازم را اعمال کنید بطوریکه ضمن اصلاح موارد در نظر گرفته نشده، همه جدول‌ها نیز در سطح سومِ نرمال‌سازی قرار گیرند.

دقت کنید که برای تغییر دادن جدول‌ها، باید از دستورات SQL همچون drop، alter و... استفاده کنید.

فاز دوم – بخش اول (SQL):

❖ خروجی این فاز شامل دو نوع فایل است:

- فایل‌هایی با پسوند ***sql*** که دستورات در آنها قرار دارند.
- یک فایل ***pdf*** شامل نمودار گرافیکی فاز قبل و نمودار نهایی (به دست آمده در این فاز) که باید با یکدیگر بطور خلاصه مقایسه کنید. همچنین تمام خروجی‌های حاصل از اجرای دستورات SQL را در این فایل قرار دهید.

❖ دستورات alter و drop و ...:

- کلیه اصلاحاتی که لازم است روی جدول‌های خود اعمال کنید تا بتوانید دستورات این بخش را اجرا نمایید، با استفاده از دستورات SQL پیاده‌سازی کنید.
- دقت کنید که مشکلات جزئی موجود در فاز قبل (برای تمام گروه‌ها) باید برطرف گردد.
- این مشکلات شامل در نظر نگرفتن بسیاری از جداول، استفاده نادرست از ارث‌بری و جدول‌های میانی و نیز نرمال نبودن جدول‌ها است.

❖ دستورات Insert و Update:

۱. تعداد ۱۲ کاربر با مشخصات مختلف به پایگاه داده افزوده شود.

(علی غلامی، زهره رسولی، الهام نیایشی، رضا ملکی، مرتضی مولایی، سعید حجازی، جان ویک، آرمان فدایی، محمد سجادی، علی محمدی)

۲. موجودی حساب ۵ کاربر اول را صفر و مابقی را بین ۵۰ هزار ریال تا ۱۰ میلیون ریال در نظر بگیرید. همچنین، موجودی جان ویک را برابر با ۱۰۰ میلیون ریال قرار دهید. (فرض کنید برای شارژ حساب هر کاربر تنها یک بار اقدام شده است. بنابراین، در جدولی که به تراکنش‌ها اختصاص داده‌اید، کافی است برای این کاربران، تنها یک ردیف داده وجود داشته باشد که نوع تراکنش آن افزایش موجودی باشد.)

۳. دو سفر اتوبوسی VIP برای شرکتی دلخواه (مثلا ایران پیما) در سامانه خود اضافه کنید که اولی از تهران به مشهد و دومی از تهران به بابل می‌رود. تاریخ‌های دو سفر را به ترتیب ۲۳ و ۲۴ آذر ۱۴۰۰ با ظرفیت ۲۲ نفر در نظر بگیرید.

۴. یک نوع سفر هواپیمایی از لس آنجلس به تهران با ظرفیت ۳۰۰ نفر (با شماره صندلی از ۱ تا ۳۰۰) در تاریخ ۲۰ آذر ۱۴۰۰ بسازید. (استفاده از trigger در چنین مواردی ارجح است.)

۵. برای کاربر جان ویک یک بلیط هواپیما از لس آنجلس به تهران رزرو کنید و از موجودی حساب او قیمت بلیط را کم کنید.

۶. برای کاربر سجاد محمدی، بلیط اتوبوس از تهران به مشهد با شماره صندلی ۶ رزرو کنید و قیمت بلیط را از موجودی حساب او کسر کنید.

۷. مشابه شماره ۷، بلیطی برای زهره رسولی با شماره بلیط ۷ (صندلی کنار سجاد محمدی) رزرو کنید.

۸. برای کاربر زهره رسولی بلیط را تعویض کرده و صندلی شماره ۱۰ را رزرو نمایید.

۹. هتل اسپیناس را به پایگاه داده خود اضافه کرده و ۱۰ اتاق ۲ خوابه VIP به همراه صبحانه به آن اضافه کنید.

۱۰. یک اتاق دو خوابه برای جان ویک در روز ۲۰ آذر ۱۴۰۰ رزرو کنید و هزینه آن را نیز از موجودی حساب او کم کنید.

❖ دستور Delete:

کاربرانی را که از تاریخ عضویت آنها، یک هفته گذشته و هیچ‌گونه تراکنشی نداشته‌اند، حذف کنید.

❖ دستور Insert:

جدول جدیدی ساخته و مشخصات تمام کاربرانی که موجودی غیر صفر دارند را در آن درج نمایید.

❖ دستور View:

لیست کاربرانی که موجودی حسابشان صفر نیست نمایش دهید.

استفاده از Transaction: (اختیاری/ امتیازی)

هنگام رزرو بلیط‌ها، از transaction استفاده کنید، تا در صورتی که دو درخواست (request) همزمان برای رزرو یک بلیط مشخص به سمت پایگاه داده ارسال شد، مشکل Inconsistency در پایگاه داده رخ ندهد.

پردازه‌های ذخیره شده (Stored Procedure):

پرسمان‌های زیر را الزاماً بصورت پردازه ذخیره شده بنویسید.

برای هر پردازه از نام مشخص شده استفاده کنید. پردازه‌ها را بصورت پارامتری بنویسید و سپس با مقادیر مشخص شده اجرا کنید. توجه کنید خروجی هیچ پرسمانی نباید تهی باشد[1].

۱. نام پردازه ذخیره شده: **purchasedTickets**

نمایش تعداد کاربران متمایزی که بلیط‌های پرواز از A به B را بصورت X، خریداری کرده‌اند. (دقت کنید که این پرسمان باید برای هر دو نوع پرواز داخلی و خارجی قابل استفاده باشد.)

رفت و برگشت = X و نجف = B و تهران = A

۲. نام پردازه ذخیره شده: **userWalletDetail**

نمایش دلیل آخرین تغییر موجودی کاربرانی که نام خانوادگی آنها به A ختم می‌شود. (دلایل تغییر موجودی: شارژ حساب، خرید بلیط هواپیمای داخلی، خارجی، اتوبوس، هتل)

ی = A

۳. نام پردازه ذخیره شده: **allDomesticFlights**

نمایش نام شرکت‌های هواپیمایی برای تمام پروازهای داخلی که از مبدأ A به مقصد B و در تاریخ C انجام شده‌اند.

C = 1400/12/11 و کیش = B و تهران = A

[1] در این صورت، داده‌های موجود در جدول‌ها را افزایش دهید.

۴. نام پردازه ذخیره شده: **internationalTickets**

نمایش میانگین تعداد حروف نام خانوادگی کاربرانی که بیش از C بار، بلیط‌های پرواز خارجی از X به Y را خریداری کرده‌اند.

X = تهران و Y = نیویورک و C = 2

۵. نام پردازه ذخیره شده: **terminalBuses**

نمایش لیست تمام اتوبوس‌های پایانه A متعلق به شرکت اتوبوسرانی D که در روز E از مبدأ B به مقصد C می‌روند.

A = غرب و B = تهران و C = اصفهان و D = رویال سفر و E = 1400/10/17

۶. نام پردازه ذخیره شده: **hotelOrBusUsers**

نمایش میانگین موجودی حساب کاربرانی که حداقل A بار، هتلی را در شهر B رزرو کرده و از B با اتوبوس خارج شده‌اند.

A = 2 و B = تهران

۷. نام پردازه ذخیره شده: **internationalFlight**

نمایش جزئیات X پرواز خارجی دارای ارزان‌ترین قیمت بلیط از مبدأ Y به مقصد Z در N ماه اخیر (از تاریخ روز جاری که DBMS بازمی‌گرداند).

X = 5 و Y = تهران و Z = تورنتو و N = 3

۸. نام پردازه ذخیره شده: **allInternationalTickets**

نمایش اطلاعات تمام بلیط‌های پرواز خارجی بصورت X که از مبدأ Y به مقصد Z رفته و در فاصله زمانی A تا B بوده است.

X = فرست کلاس و Y = تهران و Z = فرانکفورت و A = 1400/09/25 و B = 1400/09/30

۹. نام پردازه ذخیره شده: **joinedUsers**

نمایش مشخصات کاربرهایی که یا تاریخ عضویت آنها در بازه A و B باشد، یا در ماه گذشته، در مجموع کمتر از C سفر هوایی یا اتوبوسی داشته‌اند.

A = 1400/11/25 و B = 1400/10/13 و C = 2

۱۰. نام پردازه ذخیره شده: **myTrips**

نمایش لیست "سفرهای من" برای کاربری با نام X و نام خانوادگی Y.

X = سجاد و Y = محمدی

۱۱. نام پردازه ذخیره شده: **chargedUsers**

نمایش تعداد کاربرانی که حداقل X بار حسابشان را شارژ کرده و حداکثر Y خرید داشته‌اند.

X = 7 و Y = 15

۱۲. نام پردازه ذخیره شده: **allBusTickets_1**

نمایش نام شرکت‌های اتوبوس‌رانی که تاکنون سفری از مبدأ Y به مقصد Z نداشته‌اند.

Y = کرج و Z = قم

۱۳. نام پردازه ذخیره شده: **allBusTickets_2**

نمایش بلیط اتوبوس دارای زودترین تاریخ حرکت یا بالاترین ظرفیت متعلق به شرکت اتوبوسرانی Z که دارای مبدأ X و مقصد Y بوده و قیمتی کمتر از C ریال دارد.

X = شیراز و Y = اصفهان و Z = سفر و سیر و C = 900000

۱۴. نام پردازه ذخیره شده: **twoTrips**

نمایش نام و نام خانوادگی کاربرانی که سفر داخلی (اتوبوس یا هواپیما) از مبدأ X به مقصد Y داشته، سپس از Y سفر خارجی به مقصد Z داشته‌اند (تاریخ سفر دوم باید بعد از تاریخ سفر اول باشد).

X = تهران و Y = بندرعباس و Z = دبی

۱۵. نام پردازه ذخیره شده: **lastChange**

نمایش دلیل و تاریخ آخرین تغییر در حساب شخصی با نام X و نام خانوادگی Y.

X = زهره و Y = رسولی

۱۶. نام پردازه ذخیره شده: **reserveHotel_1**

نمایش مشخصات کاربرانی که N بار هتلی را در شهر X در بازه تاریخی A تا B رزرو کرده‌اند و M بار بلیط اتوبوس از مبدأ Y در بازه تاریخی C تا D گرفته‌اند.

N = 3 و X = کیش و A = 1398/10/15 و B = 1400/10/15

M = 4 و Y = تهران و C = 1396/01/25 و D = 1399/05/11

۱۷. نام پردازه ذخیره شده: **reserveHotel_2**

نمایش مشخصات کاربرانی که در بازه زمانی A تا B در تمام هتل‌های کمتر از سه ستاره شهر C اتاق رزرو کرده‌اند.

C = یزد و B = 1400/09/20 و A = 1400/05/20

۱۸. نام پردازه ذخیره شده: **reserveHotel_3**

نمایش نام خانوادگی کاربرانی که تاکنون یا فقط در هتل‌های شهر A اتاق رزرو کرده‌اند، یا هیچ سفری (هوایی یا اتوبوسی) به مقصد B نداشته‌اند، به ترتیب صعودی حروف الفبا.

B = بندرعباس و A = مشهد

۱۹. نام پردازه ذخیره شده: **reserveHotel_4**

نمایش مشخصات کاربرانی که در یکی از هتل‌های شهر B بیش از C روز اقامت داشته‌اند.

C = 7 و B = کیش

۲۰. نام پردازه ذخیره شده: **reserveHotel_5**

نمایش مشخصات هتل‌های بیش از A ستاره که تاکنون بیشترین تعداد مسافر را اسکان داده‌اند.

A = 2

Triggers (اختیاری / امتیازی):

هنگامی که یک نوع سفر با ظرفیت مشخص ساخته می‌شود (پرواز و اتوبوس) در جدول بلیط‌ها، بلیط‌هایی با شماره ۱ تا ظرفیت مورد نظر به صورت خودکار ساخته شود. (دقت کنید که در trigger می‌توان Stored Procedure نیز صدا زد!)

راهنماها:

Transactions:
https://www.tutorialspoint.com/sql/sql-transactions.htm
https://www.geeksforgeeks.org/sql-transactions/

Sequence:
https://www.geeksforgeeks.org/sql-sequences/?ref=lbp
https://stackoverflow.com/questions/10062328/sequence-vs-identity

Triggers:
https://www.geeksforgeeks.org/sql-trigger-student-database/
https://www.edureka.co/blog/triggers-in-sql/
https://www.sqlshack.com/learn-sql-sql-triggers/

کار با تقویم شمسی:

https://stackoverflow.com/questions/6706692/how-to-convert-datetime-in-persian-in-sql-server
https://raresql.com/2013/05/08/sql-server-how-to-convert-gregorian-dates-to-hijri-date-with-formatting/

فاز دوم – بخش دوم (اتصال به برنامه):

در نهایت یک برنامه کامل را در نظر بگیرید (یعنی نرم افزاری شامل لایه‌های Data، Business/Application و Presentation).

در این فاز تنها لازم است لایه Data را پیاده کنید، به این معنی که در یک زبان برنامه‌نویسی دلخواه با استفاده از استانداردهای اتصال به پایگاه داده (مانند JDBC و ODBC و ADO.NET) برنامه را به پایگاه داده وصل کنید. (به جلسات کلاس حل تمرین عملی مراجعه کنید.)

- جلسه مربوط به استانداردهای اتصال: 1400/09/06
- جلسه مربوط به مثال عملی از لایه Data: 1400/09/27

❖ ابتدا یک interface به صورت زیر در برنامه خود ایجاد کنید:

```
Interface Repository<Entity,ID extends Serializable> {
        Entity findById(ID id);
        List<Entity> findByIDs(Collections<ID> ids);
        List<Entity> findAll();
        Boolean deleteByID(ID id);
        Boolean DeleteByIDs(Collection<ID> ids);
        Entity save(Entity E); // saves the Entity, if already exists updates it, then return the entity with the given id by the database
        …
}
```

برای این کار دو پکیج Model و Repository را در نظر گرفته و به ازای هر entity در پایگاه داده حداقل یک کلاس بسازید. فیلدهای مناسبی در نظر بگیرید و این کلاس‌ها را در پکیج مدل قرار دهید. سپس، برای هر entity،

یک کلاس DAO بسازید که واسط Repository را پیاده کند. سپس، متدهای این کلاس‌های DAO را پیاده‌سازی کنید. مثلاً برای یک User entity که id آن از جنس bigint باشد، کلاس User را مطابق زیر بسازید:

Class UserDao implements Repository<User,Long>

فاز ۳ (اختیاری/ امتیازی):

لایه Business و لایه Presentation را پیاده‌سازی کنید.

این برنامه می‌تواند یک اپ دسکتاپ (با local database) یا وب‌سایت یا موبایل باشد.

منظور از این دو لایه این است که برنامه شما باید یک UI داشته باشد که از طریق آن بتوان با برنامه کار کرد. منطق برنامه نیز باید نوشته شود، UI خالی نباشد و برنامه قابل استفاده باشد.

نکات مهم:

- تحویل فاز دوم و سوم این پروژه به صورت گروهی و در دو مرحله بارگذاری در سایت و تحویل شفاهی (آنلاین) انجام می‌پذیرد. (بارگذاری فایل‌های پروژه توسط **تمام** اعضای گروه الزامی است.)
- دقت کنید که تمام فایل‌های مربوط به هر دو بخش و خروجی‌های به دست آمده را باید بصورت فشرده شده بارگذاری نمایید.

  در صورت بالا بودن حجم فایل‌ها می‌توانید یک repository مانند *gitlab* یا *github* بصورت *public* ایجاد و لینک آن را بارگذاری کنید.
- هر گروه در فاز پیشین مشخص شده است. (گروه خود را تغییر **ندهید**.)
- در صورت عدم حضور برای ارائه شفاهی، نمره‌ای برای فازهای دوم (و یا سوم) این پروژه منظور **نخواهد شد**.
- تحویل شفاهی (آنلاین) برای فاز دوم در بازه **۱۴ تا ۱۶ دی** انجام می‌شود. (زمان دقیق متعاقباً در کانال درس اعلام می‌گردد.)
- تحویل شفاهی (آنلاین) برای فاز سوم در نیمه اول ماه بهمن (پس از امتحانات) انجام می‌شود. (زمان دقیق متعاقباً در کانال درس اعلام می‌گردد.)
- در صورت سؤال یا ابهام می‌توانید از طریق گروه تلگرامی و یا با یکی از شناسه‌های *@Shayan_Daneshvar* و یا *@Ali_E99* مطرح فرمایید.

موفق و سربلند باشید ☺