پروژه‌ی نهایی : طراحی دیتابیس، ETL و Data Warehouse

ETL مخفف Extract, Transform and Load است که به معنای استخراج، پالایش و بارگذاری اطلاعات است. از ETL در زمان ساخت انبار داده (Data Warehouse) استفاده می‌شود. فرایندی که به موجب آن اطلاعات از یک یا چند منبع مختلف جمع آوری، پالایش و در نهایت در انبار داده بارگذاری می‌شود. این عملیات می‌تواند توسط یکی از ابزارهای ETL یا Stored Procedure ها یا Function ها و کوئری‌ها انجام گیرد. در این پروژه قصد داریم تا با توجه به نیازمندی‌های بیان شده یک پایگاه داده را طراحی کرده و بسازیم. سپس یک خطلوله (Pipeline) برای ETL بسازیم تا داده‌ها را به انبارداده‌ای که طراحی کرده و ساخته‌ایم انتقال دهد.

- **طراحی پایگاه داده**

پایگاه داده‌ای با نیازمندی‌های زیر طراحی کنید. کوئری‌های مربوط به ساخت آن را با postgresql بنویسید و مدل داده‌ای آن را رسم کنید و ضمیمه‌ی آن قرار دهید. دقت کنید پایگاه داده طراحی شده باید در فرم نرمال ۵ باشد.

- یک کتاب‌خانه دارای تعدادی کتاب است. ممکن است از بعضی از کتاب‌ها بیش از یک جلد در کتاب‌خانه موجود باشد. هر کتاب دارای شابک (شماره‌ی استاندارد بین‌المللی کتاب یا همان ISBN)، عنوان، توضیح، شماره‌ی نسخه، زمان انتشار و نام انتشارات است. یک یا چند نویسنده دارد. به یک یا چند ژانر تعلق دارد. به یک یا چند زبان نوشته شده است. می‌تواند ترجمه شده و یک یا چند مترجم داشته باشد.
  هر یک از اعضای کتاب‌خانه دارای مشخصات فردی مانند نام، تاریخ تولد، تاریخ عضویت، شماره‌ی عضویت، اطلاعات تماس و آدرس است. بدیهی است هر یک از اعضا می‌تواند یک یا چند نسخه از یک یا چند کتاب را تا موعد مشخصی به امانت بگیرد و در این زمان این نسخه‌ها قابلیت به امانت گذاشته شدن مجدد را ندارند.

- **طراحی ETL**

خط لوله‌ی انتقال داده می‌بایست با برقراری اتصال به دو پایگاه داده که هر دو از نوع postgresql هستند داده‌ها را از مبدا به مقصد منتقل کند. به ازای هر جدول در پایگاه داده‌ی مبدا می‌بایست جدولی هم‌نام با آن در پایگاه داده‌ی مقصد وجود داشته باشد که بعد از انجام عمل انتقال دقیقا برابر با پایگاه‌داده‌ی مبدا باشد.

برای انتقال داده‌ها از هر جدول پایگاه داده‌ی مبدا به متناظر آن در مقصد باید به موارد زیر توجه کرد:

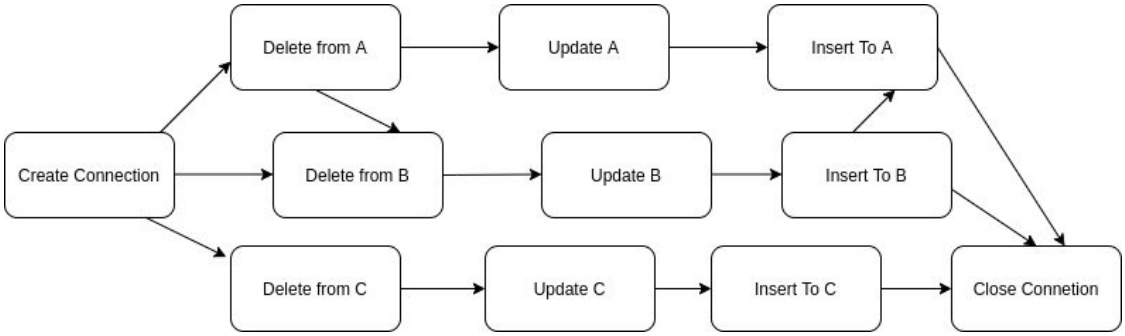
- ممکن است پس از آخرین اجرای ETL رکوردهایی از پایگاه داده‌ی مبدا حذف شده باشند. پس می‌بایست در پایگاه داده‌ی مقصد نیز از جدول هم نام متناظر حذف شوند. (Delete)

- ممکن است پس از آخرین اجرای ETL رکوردهایی در پایگاه داده‌ی مبدا دچار تغییر شده باشند. در این رکوردها primary key که می‌تواند شامل یک ستون یا ترکیبی از ستون‌ها باشد بدون تغییر باقی می‌ماند و یک یا چند عدد از دیگر ستون ها دچار تغییر شده‌اند. می‌بایست این رکوردها در پایگاه داده‌ی مقصد نیز در جدول هم نام متناظر به‌روزرسانی شوند. (Update)

- ممکن است پس از آخرین اجرای ETL رکوردهایی به پایگاه داده‌ی مبدا اضافه شده باشند. پس می‌بایست در پایگاه داده‌ی مقصد نیز به جدول هم نام متناظر اضافه شوند. (Insert)

این رکوردها در هر یک از این شرایط باید شناسایی شوند. این که در هر بار اجرای ETL تمام رکوردهای پایگاه داده‌ی مقصد حذف شده و تمام رکوردهای پایگاه داده‌ی مبدا به مقصد انتقال یابند در حجم کلان داده ممکن نیست و در این پروژه نیز پذیرفته نمی‌شود.

از طرفی باید به این نکته توجه کنید که ممکن است دو جدول به هم وابسته باشند. یعنی در یکی از آن‌ها (الف) foreign key ای به جدول دیگر (ب) وجود داشته باشد. بدیهی است اگر نیاز است رکوردی به جدول (الف) اضافه شود ابتدا می‌بایست رکورد متناظر با آن به جدول (ب) اضافه شده باشد و اگر نیاز است از جدول (ب) رکوردی حذف شود ابتدا می‌بایست رکورد متناظر با آن از جدول (الف) حذف شده باشد.
پس در نتیجه می‌بایست گرافی جهت‌دار و بدون دور (DAG) از گره‌های ریزدانه‌ی وظایف برای انجام به ترتیب آن‌ها ایجاد شود. برای مثال در پایگاه داده‌ی زیر جدول A به جدول B وابسته است و جدول C از هر دو آن‌ها مستقل است. اجرای درست ETL با این DAG امکان پذیر است :
(برای راهنمایی: می‌توانید از topological sort برای پیمایش این DAG استفاده کنید.)



توجه کنید که در مراحل مختلف پیاده‌سازی ETL برای به دست آوردن لیست جداول، برای به دست آوردن انواع Key ها و وابستگی بین آن‌ها و . . . به هیچ عنوان نباید از Hard code استفاده کنید و می‌بایست با استفاده از کوئری مناسب به دیتابیس آن‌ها را به دست آورید.

پیشنهاد می‌شود برای پیاده‌سازی ETL از زبان python استفاده کنید. این یک پیشنهاد است، می‌توانید پیاده‌سازی‌های دیگری، در یکی از سه زبان C، C++ و Java نیز داشته باشید!

شما می‌بایست در انتها source code خود به همراه داکیومنتی که برای توضیح آن تهیه کرده‌اید تحویل دهید.

- **طراحی انبار داده**

نیازمندی اصلی طراحی انبار داده سفر به گذشته است؛ ما دوست داریم بدانیم دیتابیس مبدا در تاریخ معینی در گذشته در چه حالتی قرار داشته است. پس می‌بایست :

- به ازای هر رکوردی که Insert می‌شود بدانیم دقیقا در چه زمانی به انبار داده اضافه شده است. می‌توان در هر جدول یک ستون برای این کار اضافه کرد. (رکوردی که این ستون را دارد با رکورد متناظری که در دیتابیس مبدا این ستون را ندارد برابر در نظر گرفته می‌شود). این یک پیشنهاد است، می‌توانید پیاده‌سازی‌های دیگری نیز داشته باشید!

- به ازای هر رکورد که در دیتابیس مبدا Delete یا Update می‌شود باید این رکورد در انبار داده حفظ شود ولی از جدول هم‌نام متناظر به جدول دیگری که برای تاریخچه در نظر گرفته شده است انتقال یابد. باید مشخص باشد چه زمانی از جدول اصلی خارج شده است و علت آن چه بوده است. (حذف از جدول اصلی یا به روزرسانی آن) این یک پیشنهاد است، می‌توانید پیاده‌سازی‌های دیگری نیز داشته باشید!

این دو شرط برای سفر به گذشته کافی هستند. توضیح که چگونه با استفاده از این دو شرط می‌توان در انبار داده به گذشته سفر کرد! سپس کوئری‌های مربوط به ساخت آن را با postgresql بنویسید و مدل داده‌ای آن را رسم کنید و ضمیمه‌ی آن قرار دهید.