توضیحات تکمیلی انبار داده

آرمان رضایی – ۹۷۲۳۰۳۴

این فایل جهت توضیح بخش‌هایی از مدل انبار داده که شاید بلافاصله بدیهی به نظر نیایند تهیه شده است.

برخلاف پایگاه داده‌های سنتی که از مدل Relational استفاده می‌کنند، این روش خیلی برای انبارهای داده بهینه نیست. علت این امر آن است که در انبارهای داده، هدف، ذخیره تمام اطلاعاتِ (کنونی و تاریخی) پایگاه داده اصلی، بدون از بین رفت هیچ بخشی، می‌باشد. در نتیجه بیشتر از آنکه به سه عملیات INSERT، UPDATE و DELETE علاقه‌مند باشیم، به کوئری کردن داده‌های موجود در انبار علاقه‌مندیم. لذا مدل‌های Normalize شده در این بخش خیلی بهینه نیستند، چراکه نیازمند JOINهای متعددی بین جداول مختلف می‌باشند که سرعت کوئری‌ها را کاهش می‌دهند. به همین خاطر است که طراحی این انبار داده از شمای ستاره‌ای (Star Schema) استفاده خواهد کرد. در این مدل، حقایق (Facts) شامل اطلاعات تراکنش‌های کتابخانه (قرض دادن و پس گرفتن کتاب‌ها) بوده و دو جدول books و users حکم ابعاد (Dimensions) را خواهند داشت. همچنین، از آنجایی که تمام CONSTRAINTها حین ورود داده‌ها به دیتابیس اصلی بررسی و رعایت شده‌اند، نیازی به ایجاد چنین شروطی بر روی انبار داده نیست (که این امر، سرعت INSERT داده‌های جدید در انبار داده را بسیار افزایش خواهد داد).

یکی از چالش های اصلی در طراحی انبارهای داده، احتمال تغییر اطلاعات در Dimensionها می‌باشد. این تغییرات به ندرت و بدون هیچ نظم خاصی رخ می‌دهند (مثلا، تغییر شماره تلفن خانه‌ی یک کاربر). به این گونه Dimensionها اصطلاحا Slowly Changing Dimension یا به اختصار SCD می‌گویند. چند نمونه مشکلاتی که ممکن است در SCDها پیش بیاید:

- در صورتی که یک کاربر از اعضای کتابخانه حذف شود (و در نتیجه user_id آن دیگر در جدول users موجود نباشد)، دیگر قادر نخواهیم بود اطلاعات تاریخی این کاربر را در انبار داده جست‌وجو کنیم و همچنین Referential Integrity جدول حقایق نقض خواهد شد.
- در صورتی که اطلاعات یک کتاب به روزرسانی شوند، اطلاعات تاریخی این کتاب از بین خواهند رفت.

برای غلبه بر مشکلات حاصل از SCDها متدولوژی‌های متفاوتی تحت عناوین «SCD Management Type 1» تا «SCD Management Type 6» ارائه شده‌اند. بنده برای رعایت تعادل بین پیچیدگی در پیاده‌سازی و بهینه بودن مدل، از Type 4 در طراحی انبار داده استفاده کرده‌ام که همان در نظر گیری جداولی جداگانه برای history تحت عناوین books_history و users_history می‌باشد. توجه کنید که در این حالت حفظ Referential Integrity غیرممکن می‌باشد (فلش‌های موجود در شمای مدل صرفا جهت شفاف‌سازی رسم شده‌اند) ولی این به این معنا نیست که داده‌های موجود در جداول غلط خواهند بود؛ کاملا برعکس: با استفاده از این جداول و سه عمل UNION، INTERSECT و

EXCEPT روی مجموعه‌ها می‌توان نوع و زمان دقیق هر رویدادی در کتابخانه را به دست آورد. در ادامه به توضیحات مفصل هر جدول خواهیم پرداخت.

جدول books

شامل تمام اطلاعات کتاب‌ها می‌باشد. این جدول در واقع حاصل یک JOIN بزرگ بین جداول books، publishers، genres و roles در پایگاه داده اصلی می‌باشد. همچنین یک ستون timestamp نیز اضافه شده است که تاریخ و زمان دقیق اضافه شدن داده‌ها به انبار داده را ثبت خواهد کرد. توجه کنید که در این جدول ستون‌هایی مانند genres، authors و translators چندمقداره (Multi-Valued) بوده و داده‌ها در آن‌ها به صورت آرایه ذخیره خواهند شد.

جدول books_history

جهت ذخیره‌ی تغییرات رخ داده در جدول books استفاده می‌شود. ساختاری بسیار مشابه با جدول books داشته و تنها تفاوت ساختاری آن‌ها در ستون action می‌باشد که دو مقدار updated و deleted را اختیار کرده و بیانگر نوع تغییر رخ داده در جدول books می‌باشد؛ به عبارت دیگر، به طور کامل از نابود شدن هرگونه اطلاعاتی جلوگیری خواهد نمود.

جدول users

شامل تمام اطلاعات users در دیتابیس اصلی می‌باشد. ستون timestamp جهت ثبت زمان دقیق ذخیره‌سازی در انبار داده پیاده‌سازی شده است.

جدول users_history

جهت ذخیره‌ی تغییرات رخ داده در جدول users استفاده می‌شود. ساختاری بسیار مشابه با جدول users داشته و تنها تفاوت ساختاری آن‌ها در ستون action می‌باشد که دو مقدار updated و deleted را اختیار کرده و بیانگر نوع تغییر رخ داده در جدول users می‌باشد؛ به عبارت دیگر، به طور کامل از نابود شدن هرگونه اطلاعاتی جلوگیری خواهد نمود.

جدول borrows

شامل تاریخچه‌ی تمام تراکنش‌های کتابخانه می‌باشد. در نتیجه هیچ داده‌ای از میان آن حذف نخواهد شد. ستون action نوع تراکنش (borrowed یا returned) و ستون timestamp زمان تراکنش را ذخیره خواهند نمود. از طریق این جدول (و JOIN بین دو جدول books و users) می‌توان دقیقا پی برد که در چه تاریخ و زمانی، چه عملی توسط کدام شخص و روی کدام کتاب اجرا شده است.

TRIGGERها

جهت اطمینان از از دست نرفتن هیچ داده‌ای، دو trigger بر روی جداول books و users پیاده‌سازی شده است؛ به این صورت که پس از هرگونه عملیات UPDATE یا DELETE روی هرکدام از این جداول، شکل قدیمی داده‌های تغییر یافته (OLD.*) در جداول history متناظر INSERT خواهند شد. همچنین، یک تریگر دیگر بر روی جدول borrows نوشته شده تا در صورت بازگردانده شدن یک کتاب به کتابخانه، این تغییر را در انبار داده منعکس نمایند.

دیگر فایل‌ها

فایل test.sql جهت درستی آزمایی CONSTRAINT و TRIGGERهای موجود در پایگاه داده نوشته شده است.