عنوان: **مدیریت اطلاعات وابسته به زمان در بانک‌های اطلاعاتی رابطه‌ای**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۴:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, NoSQL, database design

در یک برنامه فروشگاه، جداول مشتری و خریدهای او را درنظر بگیرید. خرید 3 سال قبل مشتری خاصی به آدرس قبلی او ارسال شده‌است. خرید امروز او به آدرس جدید او ارسال خواهد شد. سؤال: آیا با وارد کردن و به روز رسانی آدرس جدید مشتری، باید سابقه اطلاعاتی قبلی او حذف شود؟ اجناس ارسالی پیشین او، واقعا به آدرس دیگری ارسال شده‌اند و نه به آدرس جدید او. چگونه باید اینگونه اطلاعاتی را که در طول زمان تغییر می‌کنند، در بانک‌های اطلاعاتی رابطه‌ای نرمال شده مدیریت کرد؟ از این نمونه‌ها در دنیای کاری واقعی بسیارند. برای مثال قیمت اجناس نیز چنین وضعی را دارند. یک بستنی مگنوم، سال قبل 300 تومان بود؛ امسال شده است 1500 تومان. یک سطل ماست 2500 تومان بود؛ امروز همان سطل ماست 6500 تومان است. چطور باید سابقه فروش این اجناس را نگهداری کرد؟

منابع مطالعاتی مرتبط

این موضوع اینقدر مهم است که تابحال چندین کتاب در مورد آن تالیف شده است:

Temporal Data & the Relational Model
Trees and Hierarchies in SQL
Developing Time-Oriented Database Applications in SQL
Temporal Data: Time and Relational Databases
Temporal Database Entries for the Springer Encyclopedia of Database Systems
Temporal Database Management

نکته مهمی که در این مآخذ وجود دارند، واژه کلیدی « Temporal data » است که می‌تواند در جستجوهای اینترنتی بسیار مفید واقع شود.

بررسی ابعاد زمان

فرض کنید کارمندی را استخدام کرده‌اید که ساعتی 2000 تومان از ابتدای فروردین ماه حقوق دریافت می‌کند. حقوق این شخص از ابتدای مهرماه قرار است به ساعتی 2400 تومان افزایش یابد. اگر مامور مالیات در بهمن ماه در مورد حقوق این شخص سؤال پرسید، ما چه پاسخی را باید ارائه دهیم؟ قطعا در بهمن ماه عنوان می‌کنیم که حقوقش ساعتی 2400 تومان است؛ اما واقعیت این است که این عدد از ابتدای استخدام او ثابت نبوده است و باید تاریخچه تغییرات آن، در نحوه محاسبه مالیات سال جاری لحاظ شود.
بنابراین در مدل سازی این سیستم به دو زمان نیاز داریم:
الف) actual time یا زمان رخ دادن واقعه‌ای. برای مثال حقوق شخصی در تاریخ ابتدای مهر ماه تغییر کرده است. به این تاریخ در منابع مختلف Valid time نیز گفته می‌شود.
ب) record time یا زمان ثبت یک واقعه؛ مثلا زمان پرداخت حقوق. به آن Transaction time هم گفته شده است.
یک مثال:

```
record date  actual date  حقوق دریافتی
1392/01/01  1392/01/01  2000/روز
1392/02/01  1392/01/01  2000/روز
...
1392/07/01  1392/07/01  2400/روز
...
1392/17/01  1392/07/01  2400/روز
```

در این لیست، ریز حقوق پرداختی به یک شخص را ملاحظه می‌کنید. actual dateها، زمان‌هایی هستند که حقوق پایه شخص در

آن‌ها تغییر کرده و record dateها زمان‌هایی هستند که به شخص حقوق داده شده‌است.
به ترکیب Valid Time و Transaction Time، اصطلاحا Bitemporal data می‌گویند.

مشکلات طراحی‌های متداول اطلاعات وابسته به زمان

در طراحی‌های متداول، عموما یک جدول کارمندان وجود دارد و یک جدول لیست حقوق‌های پرداختی. رکوردهای لیست حقوق‌های پرداختی نیز توسط یک کلید خارجی به اطلاعات هر کارمند متصل است؛ از این جهت که نمی‌خواهیم اطلاعاتی تکراری را در جدول لیست حقوقی ثبت کنیم و طراحی نرمال سازی شده‌ای مدنظر می‌باشد.
خوب؛ اول مهرماه حقوق شخصی تغییر کرده است. بنابراین کارمند بخش مالی اطلاعات شخص را به روز می‌کند. با این کار، کل سابقه حقوق‌های پرداختی شخص نیز از بین خواهد رفت. چون وجود این کلید خارجی به معنای استفاده از آخرین اطلاعات به روز شده یک کارمند در جدول لیست حقوقی است. الان اگر از جدول لیست حقوقی گزارش بگیریم، کارمندان همواره از آخرین حقوق به روز شده خودشان استفاده خواهند کرد.

راه حل‌های متفاوت مدل سازی اطلاعات وابسته به زمان
برای رفع این مشکل مهم، راه حل‌های متفاوتی وجود دارند که در ادامه آن‌ها را بررسی خواهیم کرد.

الف) نگهداری اطلاعات وابسته به زمان در جداول نهایی مرتبط
اگر حقوق پایه شخص در زمان‌های مختلف تغییر می‌کند، بهتر است عدد نهایی این حقوق پرداختی نیز در یک فیلد مشخص، در همان جدول لیست حقوقی ثبت شود. این مورد به معنای داشتن «داده‌ای تکراری» نیست. از این جهت که داده‌ای تکراری است که اطلاعات آن در تمام زمان‌ها، دارای یک مقدار و مفهوم باشد و اطلاعات حقوق یک شخص اینچنین نیست.

ب) نگهداری اطلاعات تغییرات حقوقی در یک جداول جداگانه
یک جدول ثانویه حقوق جاری کارمندان مرتبط با جدول اصلی کارمندان باید ایجاد شود. در این جدول هر رکورد آن باید دارای بازه زمانی (valid_start_time و valid_end_time) مشخصی باشد. مثلا از تاریخ X تا تاریخ Y، حقوق کارمند شماره 11 ، 2000 تومان در ساعت بوده است. از تاریخ H تا تاریخ Z اطلاعات دیگری ثبت خواهند شد. به این ترتیب با گزارشگیری از جدول لیست حقوق‌های پرداخت شده، سابقه گذشته اشخاص محو نشده و هر رکورد بر اساس قرارگیری در یک بازه زمانی ثبت شده در جدول ثانویه حقوق جاری کارمندان تفسیر می‌شود.
در این حالت باید دقت داشت که بازه‌های زمانی تعریف شده، با هم تداخل نکنند و برنامه ثبت کننده اطلاعات باید این مساله را به ازای هر کارمند کنترل کند و یا با ثبت record_date، اجازه ثبت بازه‌های تکراری را نیز بدهد (توضیحات در قسمت بعد).
به این جدول، یک Temporal table نیز گفته می‌شود. نمونه دیگر آن، نگهداری قیمت یک کالا است از یک تاریخ تا تاریخی مشخص. به این ترتیب می‌توان کوئری گرفت که بستنی مگنوم فروخته شده در ماه آبان سال قبل، بر مبنای قیمت آن زمان، دقیقا چقدر فروش کرده است و نه اینکه صرفا بر اساس آخرین قیمت روز این کالا گزارشگیری کنیم که در این حالت اطلاعات نهایی استخراج شده صحیح نیستند.
حال اگر به این طراحی در جدولی دیگر Transaction time یا زمان ثبت یک رکورد یا زمان ثبت یک فروش را هم اضافه کنیم، به جداول حاصل Bitemporal Tables می‌گویند.

مدیریت به روز رسانی‌ها در جداول Temporal
در جداول Temporal، حذف فیزیکی اطلاعات مطلقا ممنوع است؛ چون سابقه سیستم را تخریب می‌کند. اگر اطلاعاتی در این جداول دیگر معتبر نیست باید تنها تاریخ پایان دوره آن به روز شوند یا یک رکورد جدید بر اساس بازه‌ای جدید ثبت گردد.
همچنین به روز رسانی‌ها در این جداول نیز معادل هستند با یک Insert جدید به همراه فیلد record_date و نه به روز رسانی واقعی یک رکورد قبلی (شبیه به سیستم‌های حسابداری باید عمل کرد).
یک مثال:
فرض کنید حقوق کارمندی که مثال زده شد، در مهرماه به ساعتی 2400 تومان افزایش یافته است و حقوق نهایی نیز پرداخته شده است. بعد از یک ماه مشخص می‌شود که مدیر عامل سیستم گفته بوده است که ساعتی 2500 تومان و نه ساعتی 2400 تومان! (از این نوع مسایل در دنیای واقعی زیاد رخ می‌دهند!) خوب؛ اکنون چه باید کرد؟ آیا باید رفت و رکورد ساعتی 2400 تومان را به روز کرد؟ خیر. چون سابقه پرداخت واقعی صورت گرفته را تخریب می‌کند. به روز رسانی شما ابدا به این معنا نخواهد بود که دریافتی

واقعی شخص در آن تاریخ خاص، ساعتی 2500 بوده است.
بنابراین در جداول Temporal، تنها «تغییرات افزودنی» مجاز هستند و این تغییرات همواره به عنوان آخرین رکورد جدول ثبت می‌شوند. به این ترتیب می‌توان اصطلاحا «مابه التفاوت» حقوق پرداخت نشده را به شخص خاصی، محاسبه و پرداخت کرد (می‌دانیم در یک بازه زمانی خاص به او چقد حقوق داده‌ایم. همچنین می‌دانیم که این بازه در یک record_date دیگر لغو و با عددی دیگر، جایگزین شده‌است).

برای مطالعه بیشتر

Bitemporal Database Table Design - The Basics

Temporal Data Techniques in SQL

Database Design: A Point in Time Architecture

Temporal database

Temporal Patterns

راه حلی دیگر؛ استفاده از بانک‌های اطلاعاتی NoSQL

بانک‌های اطلاعاتی NoSQL برخلاف بانک‌های اطلاعاتی رابطه‌ای برای اعمال Read بهینه سازی می‌شوند و نه برای Write. در چند دهه قبل که بانک‌های اطلاعاتی رابطه‌ای پدیدار شدند، یک سخت دیسک 10 مگابایتی حدود 4000 دلار قیمت داشته است. به همین جهت مباحث نرمال سازی اطلاعات و ذخیره نکردن اطلاعات تکراری تا این حد در این نوع بانک‌های اطلاعاتی مهم بوده است. اما در بانک‌های اطلاعاتی NoSQL امروزی، اگر قرار است فیش حقوقی شخصی ثبت شود، می‌توان کل اطلاعات جاری او را یکجا داخل یک سند ثبت کرد (از اطلاعات شخص در آن تاریخ تا اطلاعات تمام اجزای فیش حقوقی در قالب یک شیء تو در توی JSON). به همین جهت بسیار سریع هستند برای اعمال Read و گزارشگیری. همچنین این نوع سیستم‌ها برای نگهداری نگارش‌های مختلف یک سند بهینه سازی شده‌اند و جزو ساختار توکار آن‌ها است. بنابراین در این نوع سیستم‌ها اگر نیاز است از یک سند خاصی گزارش بگیریم، دقیقا اطلاعات همان تاریخ خاص را دارا است و اگر اطلاعات پایه سیستم را به روز کنیم، از امروز به بعد در سندهای جدید ثبت خواهد شد. این نوع سیستم‌ها رابطه‌ای نیستند و بسیاری از مباحث نرمال سازی اطلاعات در آن‌ها ضرورتی ندارد. قرار است یک فیش حقوقی شخص را نمایش دهیم؟ خوب، چرا تمام اطلاعات مورد نیاز او را در قالب یک شیء JSON تو در توی حاضر و آماده نداشته باشیم؟

نظرات خوانندگان

نویسنده: ناصر
تاریخ: ۱۳۹۲/۰۷/۱۷ ۰:۵۴

من هم قبلا از این روش برای قیمت‌های جدید و قدیم یک کالا در یک سیستم فروشگاهی استفاده کردم.
با نوشتن یک تریگر که به محض تغییر روی قیمت کالا ، بلافاصله داخل یک جدول دیگه این تغییرات درج میشد.
و موقع نمایش ، قیمت جدید و قدیم هر دو با هم به مشتری نمایش داده میشد.

نویسنده: سیروس
تاریخ: ۱۳۹۲/۰۷/۱۸ ۱۱:۱۱

به نظر من در نگهداری به این روش نیازی به تاریخ پایان نیست، مثلا هنگام تغییر قیمت کالا، رکوردی با تاریخ روز در جدول temporal ثبت می‌کنیم و در تغییر دوباره رکورد جدید دیگری ثبت می‌شود.
کارکردن به این روش آسانتر به نظر می‌رسد و یک فیلد کمتر داریم و نیازی هم به چک کردن درست بودن بازه‌ی تاریخی نیست.

Date	Price	ProdcutId
1392/01/01	1000	1
1392/03/05	1500	1
1392/06/27	1780	1

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۳/۰۴/۰۲ ۱۱:۴۵

البته روش شما برای حالتی مناسب است که بازه‌های تاریخی به هم متصل باشند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۰۸ ۱۰:۲۶

یک نکته‌ی تکمیلی

SQL Server 2016 مفهومی به نام Temporal Tables را پشتیبانی می‌کند.