مستندات فنی و جامع فرآیند به‌روزرسانی انبار داده (Incremental ETL)

۱. معماری کلی به‌روزرسانی

این فرآیند بر اساس یک معماری چندلایه، مقاوم و با عملکرد بالا طراحی شده است که اصول کلیدی آن به شرح زیر است:

- ۱.۱. پردازش مجموعه‌گرا (Set-Based Processing):

تمام پروسیجرها به صورت مجموعه‌گرا عمل می‌کنند. این بدان معناست که تمام داده‌های جدید در یک مرحله و به صورت یکجا پردازش می‌شوند که منجر به افزایش شدید سرعت و کاهش چشمگیر بار روی سرور پایگاه داده می‌شود.

- ۱.۲. مقاومت در برابر قطعی (Resilience) با جدول کنترلی:

یک جدول کنترلی مرکزی به نام Audit.ETL_Control ایجاد شده است. این جدول، تاریخ و زمان دقیق آخرین اجرای موفق ETL را برای هر فرآیند اصلی نگهداری می‌کند. اگر فرآیند ETL به هر دلیلی برای چند روز متوقف شود، در اولین اجرای موفق بعدی، تمام داده‌های مربوط به روزهای از دست رفته به صورت خودکار شناسایی و پردازش خواهند شد.

- ۱.۳. قابلیت اجرای مجدد (Idempotency):

تمام پروسیجرهای به‌روزرسانی به گونه‌ای طراحی شده‌اند که اگر به هر دلیلی چند بار در یک روز اجرا شوند، داده تکراری ایجاد نکنند. این ویژگی با استفاده از دستورات MERGE یا بررسی NOT EXISTS قبل از درج داده‌ها پیاده‌سازی شده است.

- ۱.۴. استفاده بهینه از جداول موقت (Temp Tables):

برای افزایش عملکرد، در ابتدای هر پروسیجر، فقط رکوردهای جدید از جداول استیجینگ خوانده شده و در یک جدول موقت کوچک (#NewFacts) ریخته می‌شوند. تمام JOINهای سنگین و محاسبات، روی این جدول موقت انجام می‌شود.

۲. مراحل کدی پروسیجرهای به‌روزرسانی ابعاد (Dimension Updates)

این پروسیجرها ابعاد انبار داده را با آخرین تغییرات Staging Area همگام‌سازی می‌کنند.

Dim.UpdateAllDimensions (**پروسیجر اصلی**)

۱. **آغاز فرآیند:** شروع عملیات و درج یک رکورد لاگ برای ردیابی.

۲. **اجرای پروسیجرهای ابعاد ساده:** فراخوانی پروسیجرهایی که فقط داده‌های جدید را درج می‌کنند (UpdateDimLeaveType و UpdateDimTerminationReason).

۳. **اجرای پروسیجر** Dim.UpdateDimDepartment: مدیریت تغییرات نام و مدیر دپارتمان‌ها با بازنویسی اطلاعات.

٤. **اجرای پروسیجر** Dim.UpdateDimJobTitle: مدیریت تغییرات دسته‌بندی مشاغل با افزودن ستون تاریخچه.

۵. **اجرای پروسیجر** Dim.UpdateDimEmployee: مدیریت کامل تاریخچه کارکنان با ایجاد رکوردهای جدید برای هر تغییر.

٦. **ثبت موفقیت:** در صورت اتمام موفقیت‌آمیز تمام مراحل، لاگ مربوطه را به‌روزرسانی می‌کند.

۷. **مدیریت خطا:** در صورت بروز هرگونه خطا در هر یک از مراحل، عملیات را متوقف کرده و خطا را در لاگ ثبت می‌کند.

Dim.UpdateDimDepartment **(برای بعد دپارتمان)**

۱. **استخراج داده‌ها:** تمام رکوردهای جدول StagingDB.HumanResources.Department را به همراه نام مدیر (که از StagingDB.HumanResources.Employee استخراج شده) در یک جدول موقت به نام #SourceDepartments درج می‌کند.

۲. **مقایسه و ادغام:** با استفاده از دستور MERGE، جدول #SourceDepartments را با جدول Dim.DimDepartment بر اساس DepartmentID مقایسه می‌کند:

- **اگر رکورد موجود بود و تغییری داشت** (WHEN MATCHED AND ...): اگر DepartmentName, ManagerName یا IsActive تغییر کرده باشد، رکورد موجود در Dim.DimDepartment را با مقادیر جدید بازنویسی می‌کند.
- **اگر رکورد جدید بود** (WHEN NOT MATCHED BY TARGET): رکورد جدید را به عنوان یک سطر جدید در Dim.DimDepartment درج می‌کند.

۳. **ثبت عملیات:** نتیجه (تعداد کل رکوردهای تغییر یافته یا درج شده) را در جدول لاگ ثبت می‌کند.

٤. **پاک‌سازی:** جدول موقت را حذف می‌کند.

Dim.UpdateDimJobTitle **(برای بعد عناوین شغلی)**

۱. **درج رکوردهای جدید:** ابتدا با یک کوئری INSERT ... WHERE NOT EXISTS، تمام عناوین شغلی جدید را از StagingDB.HumanResources.JobTitle که هنوز در Dim.DimJobTitle وجود ندارند، اضافه می‌کند.

۲. **به‌روزرسانی رکوردهای تغییریافته:** سپس با یک دستور UPDATE، رکوردهای موجود را بررسی می‌کند. اگر فیلد JobCategory تغییر کرده بود، مقدار قدیمی CurrentJobCategory را در ستون PreviousJobCategory کپی کرده و مقدار جدید را در CurrentJobCategory قرار می‌دهد.

۳. **ثبت عملیات:** تعداد رکوردهای درج شده و به‌روز شده را در لاگ ثبت می‌کند.

Dim.UpdateDimEmployee **(برای بعد کارمندان)**

۱. **شناسایی تغییرات:** با JOIN کردن StagingDB.HumanResources.Employee و Dim.DimEmployee (جایی که IsCurrent=۱)، رکوردهایی که در فیلدهای کلیدی (مانند آدرس، تلفن، شغل، وضعیت تأهل یا وضعیت استخدامی) تغییر داشته‌اند را شناسایی و در جدول موقت #EmployeeChanges ذخیره می‌کند.

۲. **منقضی کردن رکوردهای قدیمی:** تمام رکوردهای Dim.DimEmployee را که کلیدشان در #EmployeeChanges وجود دارد، با تنظیم EndDate به تاریخ دیروز و IsCurrent به ۰، غیرفعال می‌کند.

۳. **درج نسخه‌های جدید:** به ازای هر رکورد در #EmployeeChanges، یک سطر جدید در Dim.DimEmployee با اطلاعات به‌روز، StartDate برابر با امروز و IsCurrent برابر با ۱ درج می‌کند.

٤. **درج کارمندان کاملاً جدید:** با یک کوئری INSERT ... WHERE NOT EXISTS، کارمندانی را که در استیجینگ وجود دارند اما هنوز هیچ رکوردی در Dim.DimEmployee ندارند، درج می‌کند.

٥. **ثبت عملیات:** تعداد کل رکوردهای درج شده و به‌روز شده را در لاگ ثبت می‌کند.

٦. **پاک‌سازی:** جدول موقت را حذف می‌کند.

۳. مراحل کدی پروسیجرهای به‌روزرسانی جداول فکت (Fact Table Updates)

این پروسیجرها داده‌های جدید را از Staging Area می‌خوانند و آن‌ها را در جداول فکت درج می‌کنند.

Fact.UpdateAllFacts **(پروسیجر اصلی)**

۱. **آغاز فرآیند:** شروع عملیات و درج یک رکورد لاگ.

۲. **اجرای پروسیجرهای آپدیت فکت:** تمام پروسیجرهای آپدیت فکت‌ها را به ترتیب منطقی فراخوانی می‌کند.

۳. **به‌روزرسانی تاریخ کنترلی: مهم‌ترین گام:** پس از اجرای موفقیت‌آمیز تمام پروسیجرها، تاریخ و زمان فعلی را به عنوان LastLoadDate جدید در جدول Audit.ETL_Control به‌روزرسانی می‌کند. این کار تضمین می‌کند که در اجرای بعدی، پردازش از ادامه همین نقطه انجام شود.

٤. **ثبت موفقیت و مدیریت خطا:** مانند پروسیجر اصلی ابعاد، نتیجه نهایی را در لاگ ثبت می‌کند.

پروسیجرهای به‌روزرسانی برای فکت‌های رویداد-محور

این پروسیجرها رویدادهای جدیدی که از آخرین اجرای موفق ETL رخ داده‌اند را در انبار داده ثبت می‌کنند.

UpdateFactTermination **و** UpdateFactSalaryPayment **(فکت‌های تراکنشی)**

- **مراحل کدی:**

۱. **خواندن تاریخ آخرین اجرا:** تاریخ LastLoadDate را از جدول Audit.ETL_Control می‌خواند.

۲. **ایجاد جدول موقت:** فقط رکوردهای جدید از جدول استیجینگ مربوطه (که LoadDate آن‌ها بزرگتر از LastLoadDate است) را در یک جدول موقت (#NewFacts) می‌ریزد.

۳. **یافتن کلیدهای صحیح:** با استفاده از INNER JOIN به ابعاد اصلی و OUTER APPLY برای یافتن آخرین سابقه معتبر، کلیدهای جایگزین را برای هر رکورد در جدول موقت پیدا می‌کند.

٤. **درج رکوردهای جدید:** رکوردهای حل‌شده از مرحله قبل را در جدول فکت نهایی درج می‌کند، به شرطی که از قبل وجود نداشته باشند (NOT EXISTS).

UpdateFactEmployeeAttendance (**فکت بدون فکت**)

- **توضیح نوع:** این جدول با اینکه معیار عددی ندارد (Factless)، اما هر سطر آن نشان‌دهنده یک رویداد یا تراکنش (حضور یا غیبت) است. بنابراین، الگوی به‌روزرسانی آن مشابه جداول تراکنشی است و فقط رویدادهای جدید را اضافه می‌کند.
- **مراحل کدی:**

۱. **خواندن تاریخ آخرین اجرا:** تاریخ LastLoadDate از جدول Audit.ETL_Control خوانده می‌شود.

۲. **ایجاد جدول موقت:** فقط رکوردهای جدید Attendance در یک جدول موقت (#NewAttendance) بارگذاری می‌شوند.

۳. **یافتن کلید صحیح EmployeeKey:** با استفاده از OUTER APPLY، آخرین نسخه معتبر کارمند تا قبل از تاریخ حضور و غیاب پیدا می‌شود.

٤. **درج رکوردهای جدید:** رکوردهای جدید را در جدول فکت درج می‌کند و با NOT EXISTS از درج داده تکراری جلوگیری می‌نماید.

Fact.UpdateFactMonthlyEmployeePerformance (**فکت تصویر دوره‌ای**)

۱. **حذف ماه جاری:** تمام رکوردهای مربوط به ماه تقویمی جاری را از جدول فکت حذف می‌کند.

۲. **تجمیع مجدد داده‌ها:** با استفاده از یک CTE، تمام داده‌های حضور و غیاب را از ابتدای ماه جاری تا به امروز از جدول SA.Attendance جمع‌آوری کرده و تمام معیارها (مجموع ساعات کاری، میانگین، تعداد تاخیر و...) را دوباره محاسبه می‌کند.

۳. **درج مجدد:** نتایج محاسبه‌شده را به عنوان سطرهای جدید برای ماه جاری در جدول فکت درج می‌کند.

Fact.UpdateFactEmployeeLifecycle (**فکت تجمعی**)

۱. **درج کارمندان جدید:** ابتدا با یک کوئری INSERT ... WHERE NOT EXISTS، رکوردهای اولیه را برای کارمندان کاملاً جدیدی که هنوز در این جدول فکت وجود ندارند، ایجاد می‌کند.

۲. **به‌روزرسانی با رویداد ترک خدمت:** با JOIN کردن جدول فکت با رکوردهای جدید ترک خدمت در استیجینگ، سطرهای مربوط به کارمندانی که به تازگی ترک خدمت کرده‌اند را پیدا کرده و فیلدهای TerminationDateKey, DaysToTermination, FinalSalary و... را برای آن‌ها UPDATE می‌کند.

۳. **به‌روزرسانی با رویداد آموزش:** با یک CTE، آخرین وضعیت آموزشی هر کارمند (تعداد کل و تاریخ آخرین آموزش) را محاسبه کرده و رکوردهایی را که وضعیتشان در جدول فکت با این مقادیر جدید مغایرت دارد، UPDATE می‌کند.