عنوان: **بررسی بارگذاری داده ها در انبار های داده و معرفی الگوهای بکار رفته در آن**
نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۷:۳۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, SSIS, data warehouse, ETL

مقدمه در لینکی که چندی پیش به اشتراک گذاشته بودم؛ به مطلبی تحت این عنوان اشاره شده بود: "آیا از KPI باید به انباره داده و هوش تجاری رسید؟" (بر گرفته از وبلاگ آقای جام سحر) که در آن به موانع پیش روی انجام پروژه‌های BI در ایران پرداخته شده است.

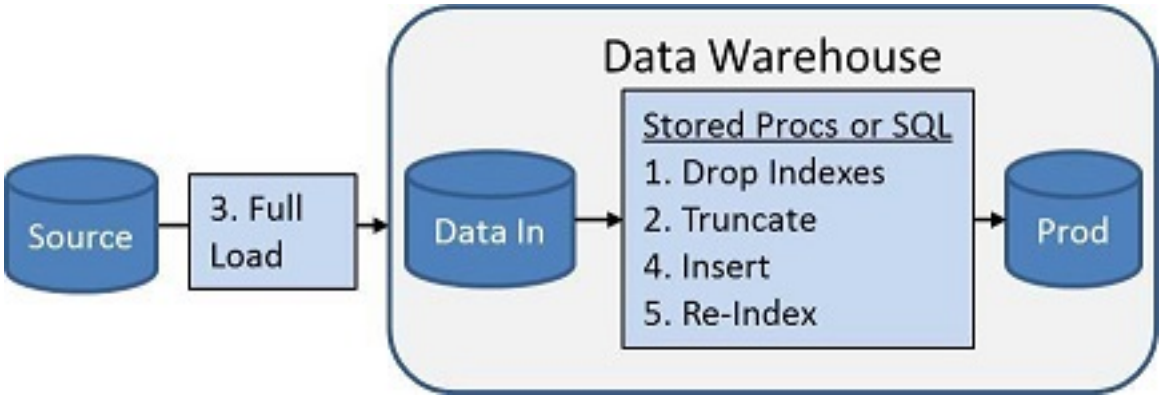
این مقاله بر گرفته از فصل سوم یکی از White Paperهای ماکروسافت با عنوان Microsoft EDW Architecture, Guidance and Deployment Best Practices می‌باشد. که به شرح عملیات Loading در فاز ETL می‌پردازد. از آنجا که به منظور پیاده سازی این نوع پروژه‌ها معمولاً در ایران برون سپاری صورت می‌گیرد و مدیران شرکت‌ها بیشتر درگیر سیستم‌های OLTP هستند و مجری پروژه (شرکت پیمانکار) معمولاً کوتاهترین مسیر را جهت انجام پروژه انتخاب می‌کند(و امروزه نیک میدانیم که "انتخاب مسیرهای کوتاه در زمان کم می‌تواند به پیچیدگی‌های بسیار جدی در دراز مدت منجر شود!") و همچنین از آنجا که متاسفانه به دلیل عدم ثبات مدیریت در ایران معمولاً "مدیریت برای تحویل پروژه تحت فشار است و نه برای مسائل پشتیبانی " و مسائل دیگری از این دست؛ چنانچه در تحویل گیری محصول به درستی تست نرم افزار صورت نگیرد، در نظر گرفتن موارد زیر:

```
Verification: Are we building the product right? ~ Software correctly implements a specific function
```

```
Validation: Are we building the right product? ~   Software is traceable to customer requirements
```

پروژه با شکست مواجه می‌شود و انتظارات مدیران بهره بردار را برآورده نمی‌کند. به هر روی در این مقاله به ترجمه مطالب زیر پرداخته می‌شود، توصیه میکنم در صورتی که با خواندن متن انگلیسی مشکلی ندارید، اصل مقاله مذکور خوانده شود.

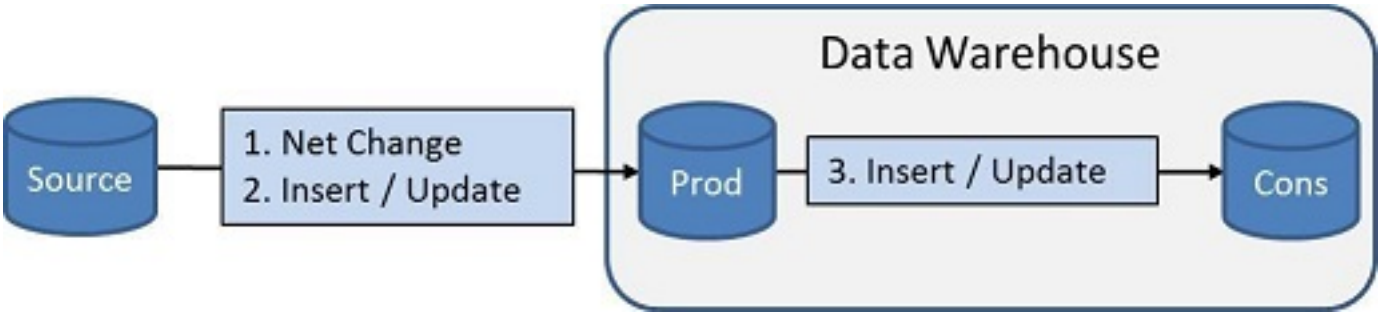
1- Full Load vs Incremental Load نسل‌های اولیه DW (اختصار Data Warehouse) به شکل Full Loads پیاده سازی می‌شدند، به این طریق که هر بار عملیات بارگذاری صورت می‌گرفت، DW از نو دوباره ساخته می‌شد. شکل زیر مراحل مختلف انجام شده در این روش را نمایش می‌دهد:



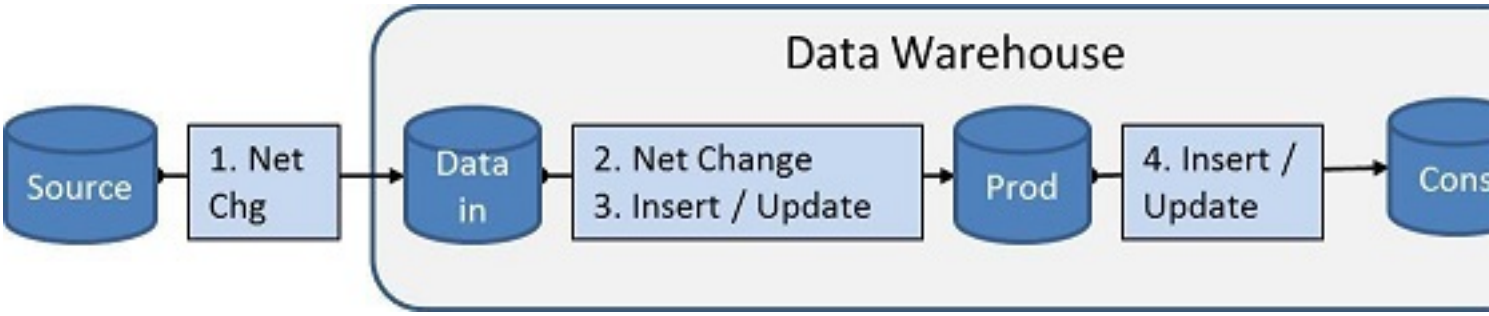
پروسه Full Load شامل مراحل زیر بود:

Drop Indexes: از آنجا که Indexها زمان بارگذاری را افزایش می‌دادند، این عمل صورت می‌پذیرفت.
Truncate Tables: تمامی رکوردهای موجود در جداول حذف می‌شدند.
Bulk Copy
Load Data
Post Process: شامل عملیاتی نظیر شاخص گذاری روی داده هایی است که اخیراً بارگذاری شده اند و....

روی هم رفته Full Load مسئله ای مشکل ساز بود، زیرا نیاز به زمانی برای بارگذاری مجدد داده‌ها داشت و مسئله‌ی مهم‌تر نداشتن امکان دستیابی به گزارشاتی تاریخچه ای با ماهیت زمان برای مشتریان کسب وکار بود. به این دلیل که همواره یک کپی از آخرین داده‌های موجود در سیستم عملیاتی درون DW قرار می‌گرفت؛ که با بکارگیری Full Load اغلب قادر به ارائه‌ی این نوع از گزارشات نبودیم، بدین ترتیب سازمان‌ها به نسل دوم روی آورند که در این دیدگاه از مفهوم Incremental Load استفاده می‌شود. اشکال زیر مراحلی که در این روش انجام می‌شود را نمایان می‌سازد:



Incremental Load with an Extract In area



Incremental Load without an Extract In area

مراحل Incremental Load شامل:

بارگذاری تغییرات نسبت به آخرین فرآیند بارگذاری انجام شده
درج / بروزرسانی تغییرات درون Production area
درج / بروزرسانی Consumption area نسبت به Production area

تفاوت‌های اصلی میان Full Load و Incremental Load در این است که در Incremental Load:

نیازی به پردازش‌های اضافی جهت حذف شاخص ها، پاک کردن تمامی رکوردهای جداول و ساخت مجدد شاخص‌ها نیست.
البته نیاز به رویه ای جهت شناسایی تغییرات می‌باشد.
و همچنین نیاز به بروزرسانی بعلاوه درج رکوردهای جدید نیز می‌باشد.

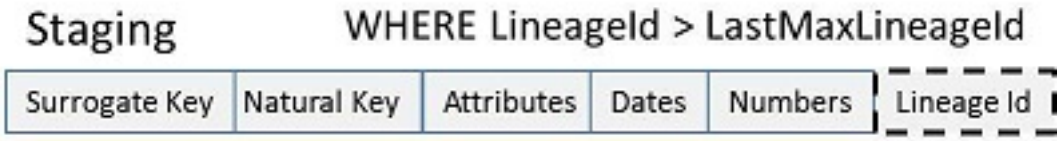
ترکیب این عوامل برای ساخت Incremental Load کارآمد تر، منجر به پیچیده‌تر شدن پیاده سازی و نگهداری آن نیز می‌شود.

2- Detecting Net Changes
فرآیند لود افزایشی ETL، بایست قادر به شناسائی رکوردهای تغییریافته در مبداء باشد، که این عمل با استفاده از هر یک از تکنیک‌های Push یا Pull انجام می‌شود.

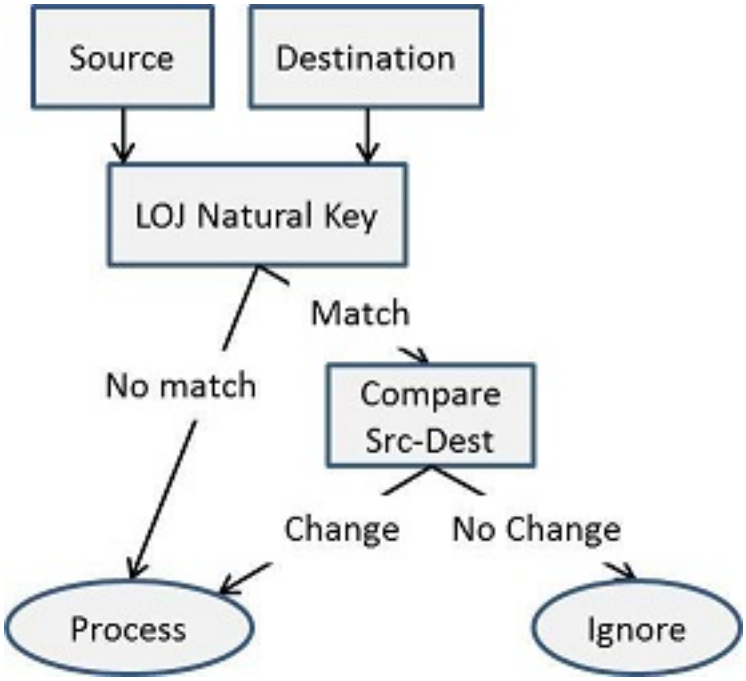
در تکنیک Pull، فرآیند ETL رکوردهای تغییریافته در مبداء را انتخاب می‌کند:
ایده‌آل وجود داشتن یک ستون Last Changed در سیستم مبداء است؛ که از آن می‌توان جهت انتخاب رکوردهای تغییر یافته استفاده نمود.
چنانچه ستون Last Changed وجود نداشته باشد، تمامی رکوردهای مبداء باید با رکوردهای مقصد مقایسه شود.
در تکنیک Push، مبداء تغییرات را شناسائی می‌کند و آنها را به سمت مقصد Push می‌کند؛ این درخواست می‌تواند توسط فرآیند ETL انجام شود.
از آنجایی که پردازش ETL معمولاً در زمان هایی که Peak کاری وجود ندارد، اجرا می‌شود، استفاده از مکانیسم Pull برای شناسایی تغییرات نسبت به مکانسیم Push ارجحیت دارد.

2-1- Pulling Net Changes – Last Change Column بیشتر جداول در سیستم‌های مبداء حاوی ستون هایی هستند که زمان ایجاد و یا اصلاح رکوردها را ثبت می‌کنند. در نوع دیگری از سیستم‌های مبداء ستونی با مقدار عددی وجود دارد، که هر زمان رکوردی تغییر یافت به آن ستون مقداری اضافه می‌شود. هر دوی این تکنیک‌ها به فرآیند ETL اجازه می‌دهند، بطور کارآمدی رکوردهای تغییریافته را انتخاب کند. (با مقایسه، بیشترین مقدار قرار گرفته در آن ستون؛ که در طول آخرین اجرای فرآیند ETL بدست آمده است).
نمونه ای از جداول سیستم مبداء که دارای تغییرات زمانی است در شکل زیر نمایش داده می‌شود.

همچنین شکل زیر نشان می‌دهد، چگونه یک مقدار عددی می‌تواند به منظور انتخاب رکوردهای تغییریافته استفاده شود.



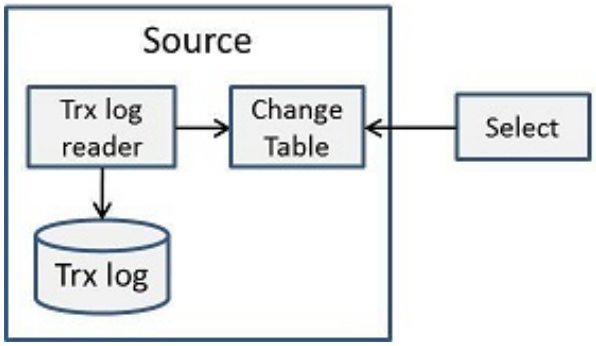
2-2- Pulling Net Changes – No Last Change Column شکل زیر گردش فرآیند را هنگامی که ستون Last Change وجود ندارد؛ نمایش می‌دهد.

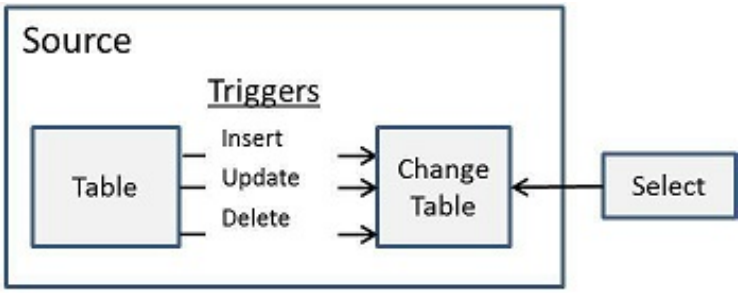


این گردش فرآیند شامل:

Join میان مبداء و مقصد با استفاده از یک دستور Left Outer Join است.
تمامی رکوردهای مبداء که در مقصد وجود ندارند، پردازش می‌شوند.
زمانی که رکوردی در مقصد وجود داشته باشد مقادیر داده‌های مبداء و مقصد مقایسه می‌شوند.
تمامی رکوردهای مبداء که تغییر یافته اند پردازش می‌شوند.
از آنجایی که تمامی رکوردها پردازش می‌شوند، این روش بویژه برای جداول حجیم؛ روش کارآمدی نیست.

2-3- Pushing Net Changes دو متد متداول Push وجود دارد که در تصویر زیر نمایش داده شده است.





تفاوت این دو روش به شرح زیر است:

در سناریو اول (شکل سمت چپ)؛ بانک اطلاعاتی رابطه ای سیستم مبداء Transaction Log را مرتب مانیتور می‌کند تا تغییرات را شناسائی کرده و در ادامه تمامی این تغییرات را در جدولی در مقصد درج می‌کند.
در سناریو دوم؛ توسعه دهندگان Trigger هایی ایجاد می‌کنند تا هر زمان که رکوردی تغییر یافت، تغییرات در جدولی که در مقصد وجود دارد درج گردد.

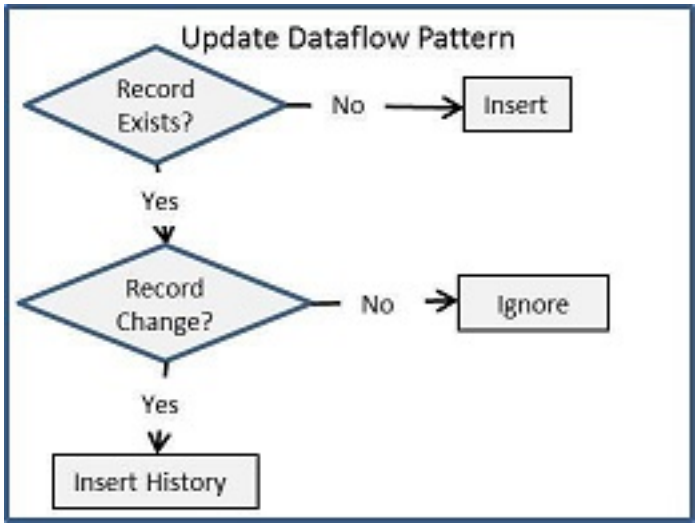
مسئله ای که در هر دو مورد وجود دارد Load اضافه ای است؛ که روی سیستم مبداء وجود دارد و می‌تواند Performance سیستم‌های OLTP را تحت تاثیر قرار دهد. به هر روی سناریو نخست معمولاً کاراتر از سناریویی است که از Trigger استفاده می‌کند.

3- ETL Patterns
پس از شناسائی رکوردهایی که در مبداء تغییر یافته اند، نیاز داریم تا این تغییرات در مقصد اعمال شود. در این قسمت به معرفی الگوهایی که برای اعمال این تغییرات وجود دارد می‌پردازیم.

3-1- Destination load Patterns
تشخیص چگونگی اضافه نمودن تغییرات در مقصد تابع دو عامل زیر است:

آیا رکورد هم اینک در مقصد وجود دارد؟
الگوی استفاده شده برای جدول مقصد به کدام شکل است؟ (Update یا Versioned Insert)

فلوچارت زیر نشان می‌دهد، به چه شکل جداول مقصد متاثر از چگونگی پردازش رکوردهای مبداء قرار دارند. توجه داشته باشید که عمل بررسی بطور جداگانه و در یک لحظه صورت می‌گیرد.



Versioned Insert Pattern 3-2-

Kimball Type II Slowly Changing Dimension نمونه ای از الگوی Versioned Insert است؛ که در آن نمونه ای از یک موجودیت دارای ورژن‌های متعددی است. مطابق تصویر زیر؛ این الگو به ستون‌های اضافه ای نیاز دارند که وضعیت نمونه ای از یک رکورد را نمایش دهد.

Surrogate Key	Natural Key	Data...	Start Date	End Date	Record Status	Version #

این ستون‌ها به شرح زیر هستند:

Start Date: زمانی که وضعیت آن نمونه از رکورد فعال می‌شود.
End Date: زمانی که وضعیت آن نمونه از رکورد غیر فعال می‌شود.
Record Status: وضعیت‌های یک رکورد را نشان می‌دهد، که حداقل به شکل Active یا Inactive است.
Version #: این ستون که اختیاری می‌باشد، ورژن آن نمونه از رکورد را ثبت می‌کند.

برای مثال شکل زیر؛ بیانگر وضعیت اولیه رکوردی در این الگو است:

Surrogate Key	Natural Key	Data...	Start Date	End Date	Status	Ver #
100	AB549		2001-02-05	NULL	A	1

فرض کنید که این رکورد در تاریخ March 2 , 2010 در سیستم مبداء تغییر می‌کند. فرآیند ETL این تغییر را شناسائی می‌کند و همانند تصویر زیر؛ به شکل نمونه ای ثانویه از این رکورد، اقدام به درج آن می‌کند.

Surrogate Key	Natural Key	Data...	Start Date	End Date	Status	Ver #
100	AB549		2001-02-05	2010-03-02	I	1
537	AB549		2010-03-02	NULL	A	2

توجه داشته باشید زمانی که رکورد دوم در جدول درج می‌شود، به منظور بازتاب این تغییر؛ رکورد اول به شکل زیر بروزرسانی می‌گردد:

End Date: تا این زمان وضعیت این رکورد فعال بوده است.
Record Status:که Active به Inactive تغییر پیدا می‌کند.

در برخی از پیاده سازی‌های DW عمدتاً از الگوی Versioned Insert استفاده می‌شود و هرگز از الگوی Update استفاده نمی‌شود. مزیت این استراتژی در این است که تمامی تاریخچه تغییرات ردیابی و ثبت می‌شود. به هر روی غالباً هزینه ثبت کردن این تغییرات منجر به ایجاد نسخه‌های زیادی از تغییرات می‌شود. تیم DW برای مواردی که تغییرات متاثر از گزارشات تاریخچه ای نیستند، می‌توانند الگوی Update را در نظر گیرند.

Update Pattern 3-3-

الگوی Update روی رکورد موجود، تغییرات سیستم مبداء را بروزرسانی می‌کند. مزیت این روش در این است که همواره یک رکورد وجود دارد و در نتیجه باعث ایجاد Queryهای کارآمدتر می‌شود. تصویر زیر بیانگر ستون هایی است که برای پشتیبانی از الگوی Update بایست ایجاد کرد.

Surrogate Key	Natural Key	Data...	Record Status	Version #

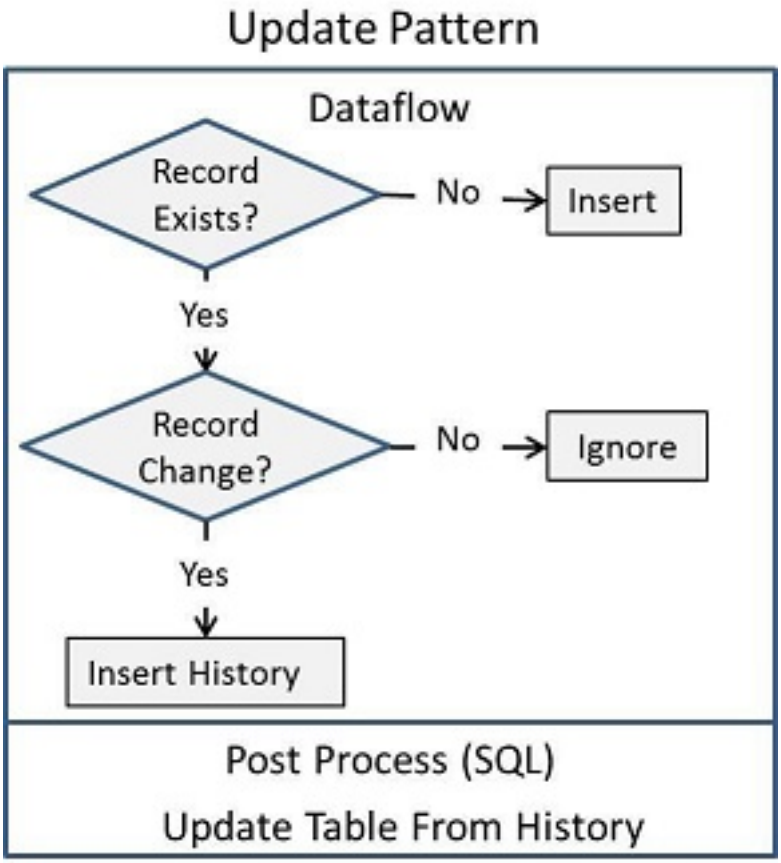
این ستون‌ها به شرح زیر هستند:

Record Status: وضعیت‌های یک رکورد را نشان می‌دهد که حداقل به شکل Active یا Inactive است.
Version #: این ستون که اختیاری می‌باشد، ورژن آن نمونه از رکورد را ثبت می‌کند.

موارد اصلی الگوی Update عبارتند از:

تاریخ ثبت نمی‌شود. ابزاری ارزشمند برای نظارت بر داده ها، تغییرات تاریخی است و زمانی که ممیزی داده رخ می‌دهد؛ می‌تواند مفید واقع شود.
بروزرسانی‌ها یک الگوی مبتنی بر مجموعه هستند. استفاده از بروزرسانی هر بار یک رکورد در ابزار ETL خیلی کارآمد (موجه) نیست.

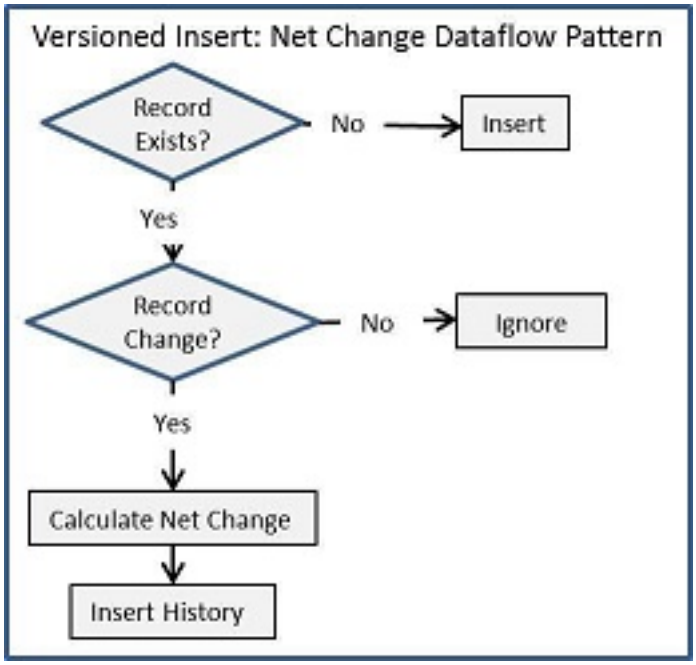
یک روش دیگر برای در نظر گرفتن موارد فوق؛ اضافه کردن یک جدول برای درج ورژن‌ها به الگوی Update است که در شکل زیر نشان داده شده است.



اضافه نمودن یک جدول تاریخچه، که تمامی تغییرات سیستم مبداء را ثبت می‌کند؛ نظارت و ممیزی داده‌ها را نیز فراهم می‌کند و همچنین بروزرسانی‌های کارآمد مبتنی بر مجموعه را برای جداول DW به ارمغان می‌آورد.

3-4- Versioned Insert: Net Changes

این الگو غالباً در جداول حجیم Fact که بروزرسانی آنها پر هزینه است استفاده می‌شود. شکل زیر منطق استفاده شده در این الگو را نشان می‌دهد.



توجه داشته باشید در این الگو:
مقادیر مالی و عددی محاسبه شده؛ به عنوان یک Net Change از نمونه قبلی رکورد در جدول Fact ذخیره می‌شود.
هیچ گونه فعالیت Post Processing صورت نمی‌گیرد (از قبیل بروزرسانی جداول Fact پس از کامل شدن Data Flow). هدف استفاده از این الگو اجتناب از بروزرسانی روی جداول بسیار حجیم می‌باشد.
عدم بروزرسانی و همچنین اندازه جدول Fact زمینه ای را فراهم می‌کند که منطق شناسائی رکوردهای تغییریافته پیچیده تر می‌شود. این پیچیدگی از آنجا ناشی می‌شود که نیاز به مقایسه رکوردهای جدول Fact آتی با جدول Fact موجود می‌باشد.

4- Data Integration Best Practices

هم اکنون پس از آشنایی با مفاهیم و الگوهای توزیع داده‌ها به ارائه تعدادی نمونه می‌پردازیم؛ که بتوان این ایده‌ها و الگوها را در عمل پوشش داد.

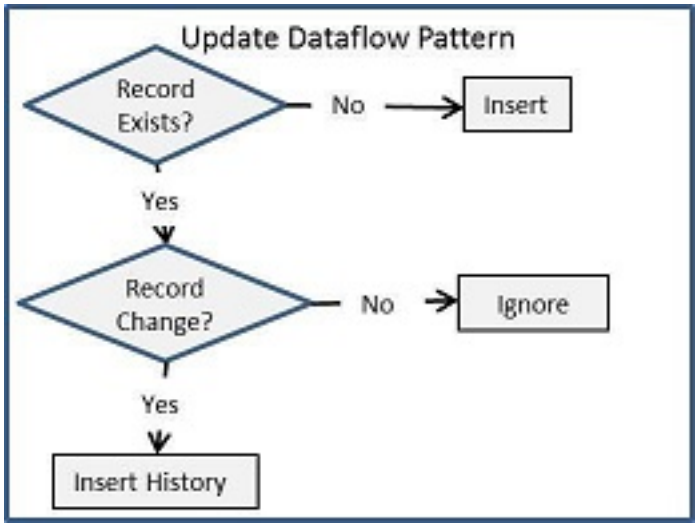
4-1- Basic Data Flow Patterns

هر یک از الگوهای Update Pattern و Versioned Insert Pattern می‌توانند برای انواعی از جداول بکار روند که معروفترین آن‌ها توسط Kimball ساخته شده اند.

Slowly Changing Dimension Type I (SCD I): از Update Pattern استفاده می‌کند.
Slowly Changing Dimension Type II (SCD II): از Versioned Insert Pattern استفاده می‌کند.
Fact Table: نوع الگویی که استفاده می‌کند به نوع جدول Fact ای که Load خواهد شد بستگی دارد.

4-1-1- Update Pattern

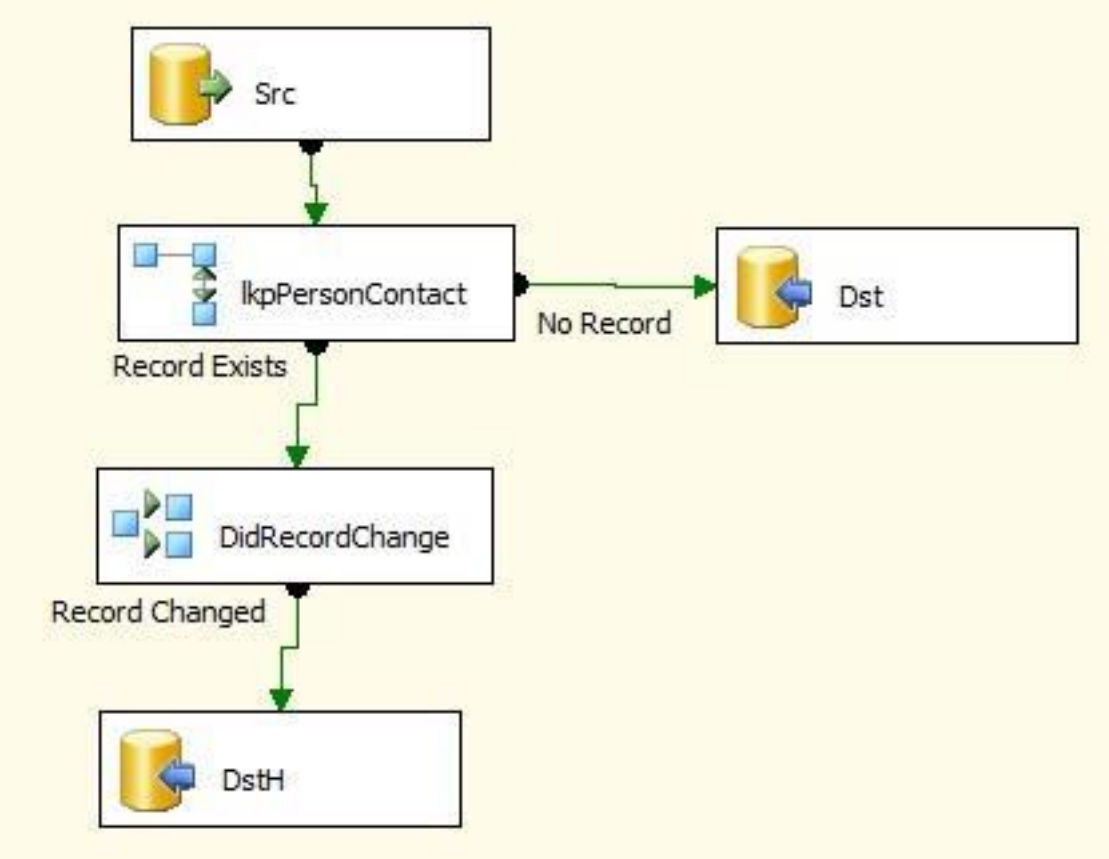
مطابق تصویر زیر جدولی که تنها حاوی ورژن فعلی رکورد هاست؛ از Update Dataflow Pattern استفاده می‌کند.



مواردی که در مورد این گردش کاری باید در نظر داشت به شرح زیر است:

این Data Flow فقط سطرهایی را به یک مقصد اضافه خواهد کرد. SSIS دارای گزینه "Table or view fast load" می‌باشد که بارگذاری‌های انبوه و سریع را پشتیبانی می‌کند.
درون یک Data Flow بروزرسانی رکوردها را می‌توان با استفاده از تبدیل OLE DB Command انجام داد. توجه داشته باشید خروجی‌های این تبدیل در یک دستور Update به ازای هر رکورد بکار می‌رود؛ مفهوم بروزرسانی انبوه در این Data Flow وجود ندارد. بدین ترتیب الگوی فعلی ارائه شده؛ تنها رکوردها را درج می‌کند و هرگز در این Data Flow رکوردها Update نمی‌شوند.
هر جدول دارای یک جدول تاریخچه است که برای ذخیره همه فعالیت‌های مرتبط با آن بکار می‌رود. یک رکورد در جدول تاریخچه زمانی درج خواهد شد؛ که رکورد مبداء در مقصد وجود داشته باشد ولی دارای مقداری متفاوت باشد.
راه دیگر فرستادن تغییرات رکوردها به یک جدول کاری است که پس از پایان یافتن فرآیند Update ، خالی (Truncate) می‌شود.
مزیت نگهداری تمامی رکوردها در یک جدول تاریخچه؛ ایجاد یک دنباله ممیزی است که می‌تواند برای نظارت بر داده‌ها به منظور نمایان ساختن موارد مطرح شده توسط مصرف کننده‌های کسب و کار استفاده شود.
گزینه‌های متفاوتی برای تشخیص تغییرات رکوردها وجود دارد که در ادامه به شرح آنها می‌پردازیم.

شکل زیر نمایش دهنده چگونگی پیاده سازی Update Dataflow Pattern در یک SSIS می‌باشد:



این SSIS شامل عناصر زیر است:

Destination table lookup:

به منظور تشخیص اینکه رکورد در جدول مقصد وجود دارد از "lkpPersonContact" استفاده می‌کنیم.

Change detection logic:

با استفاده از "DidRecordChange" مبداء و مقصد مقایسه می‌شوند. اگر تفاوتی بین مبداء و مقصد وجود نداشت؛ رکورد نادیده گرفته می‌شود. چنانچه بین مبداء و مقصد تفاوت وجود داشت؛ رکورد در جدول تاریخچه درج خواهد شد.

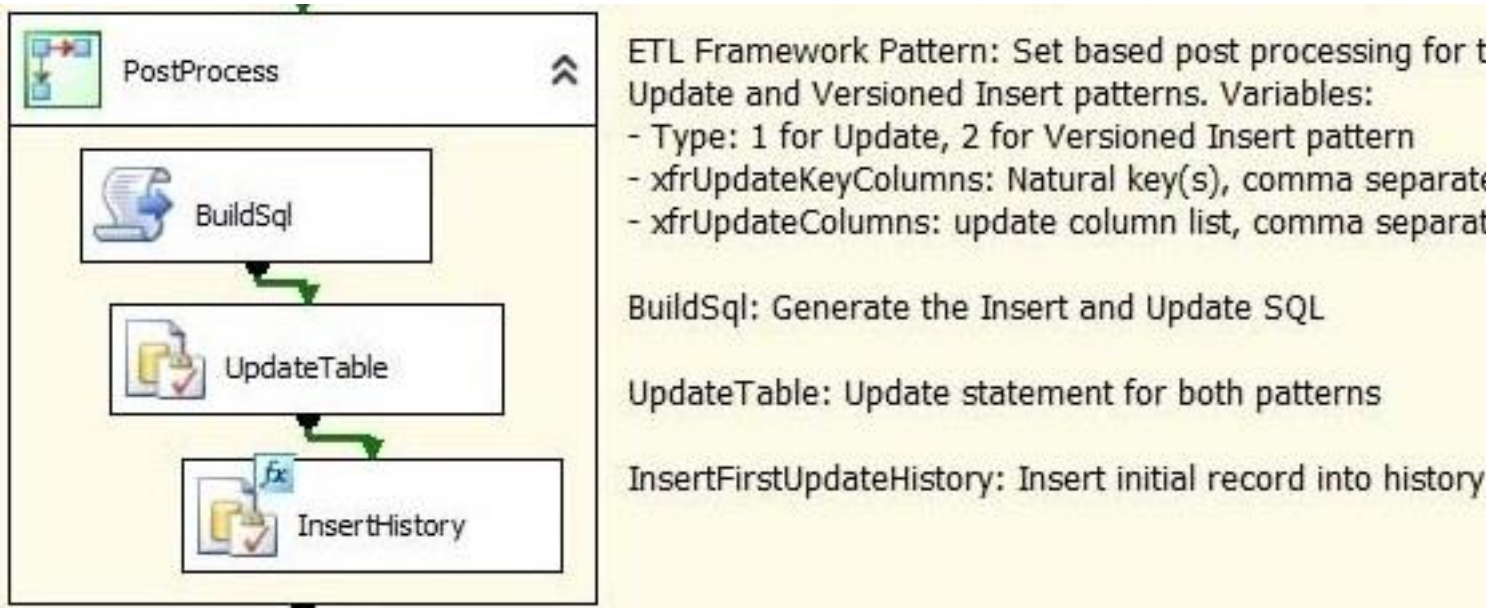
Detection Inserts:

رکوردها در جدول مقصد درج خواهند شد در صورتیکه در آن وجود نداشته باشند.

Destination History Inserts:

رکوردها در جدول تاریخچه مقصد درج خواهند شد، در صورتیکه (در مقصد) وجود داشته باشند.

پس از اتمام Data Flow یک روال Post-processing مسئولیت بروزرسانی رکوردهای جدول اصلی و رکوردهای ذخیره شده در جدول تاریخچه را بر عهده دارد که می‌تواند مطابق تصویر زیر با استفاده از یک Execute Process Task پیاده سازی شود.



PostProcess مسئولیت اجرای تمامی فعالیت‌های زیر را در این الگو برعهده دارد که شامل:

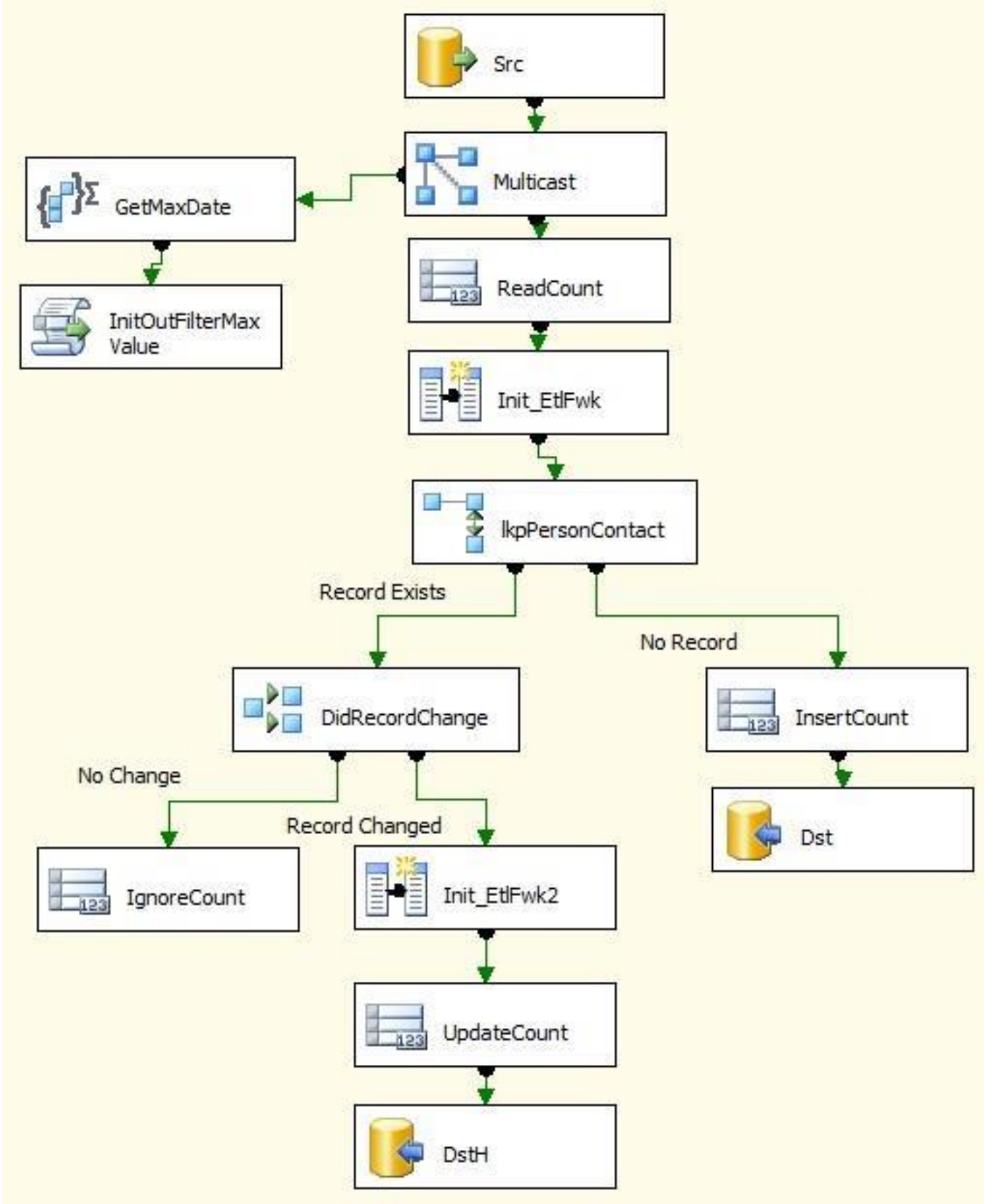
بروزرسانی رکوردهای جداول با استفاده از رکوردهای درج شده در جدول تاریخچه.
درج تمامی رکوردهای جدید (نسخه اولیه و در درون جدول تاریخچه). کلید اصلی جداولی که ستون آنها IDENTITY است مقدار نامشخصی دارد؛ تا زمانی که درج صورت گیرد، این به معنای آن است که پیش از انتقال آنها به جدول تاریخچه نیاز است منتظر درج شدن آنها باشیم.

Update Pattern – ETL Framework 4-1-2-

تصویر زیر بیانگر انجام این عملیات با استفاده از ابزارهای ETL است.
در نگاه نخستین ممکن است Data Flow از نوع اصلی خود پیچیده‌تر به نظر آید؛ که در واقع این گونه نیز هست، زیرا در فاز توسعه بیشتر Frameworkها جهت پیاده سازی به یک زمان اضافه‌تری نیاز دارند. به هر روی این زمان جهت اجتناب از هزینه روزانه تطبیق داده‌ها گرفته خواهد شد.
مزایای حاصل شده از افزودن این منطق اضافی عبارت است از:

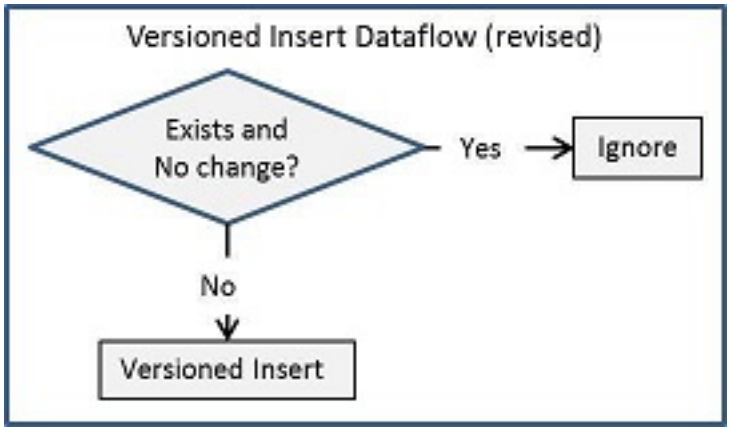
پشتیبانی از ستون هایی که کارهای ممیزی و نظارت بر داده‌ها را آسانتر می‌کنند.
تعداد سطرها شاخص مناسبی است که می‌تواند بهبود آن Data Flow خاص را فراهم کند. ناظر اطلاعات با استفاده از تعداد رکوردها می‌تواند ناهنجاری‌ها را شناسائی کند.

بهره برداران ETL و ناظران اطلاعات می‌توانند با استفاده از خلاصه تعداد رکوردها درک بیشتری درباره فعالیت‌های آن کسب کنند. پس از آنکه تعداد رکوردها، مشکوک به نظر آمد؛ تحقیقات بیشتری می‌تواند اتفاق افتد. (با عمیق‌تر شدن در جزئیات گزارشات)



Versioned Insert Pattern 4-1-3-

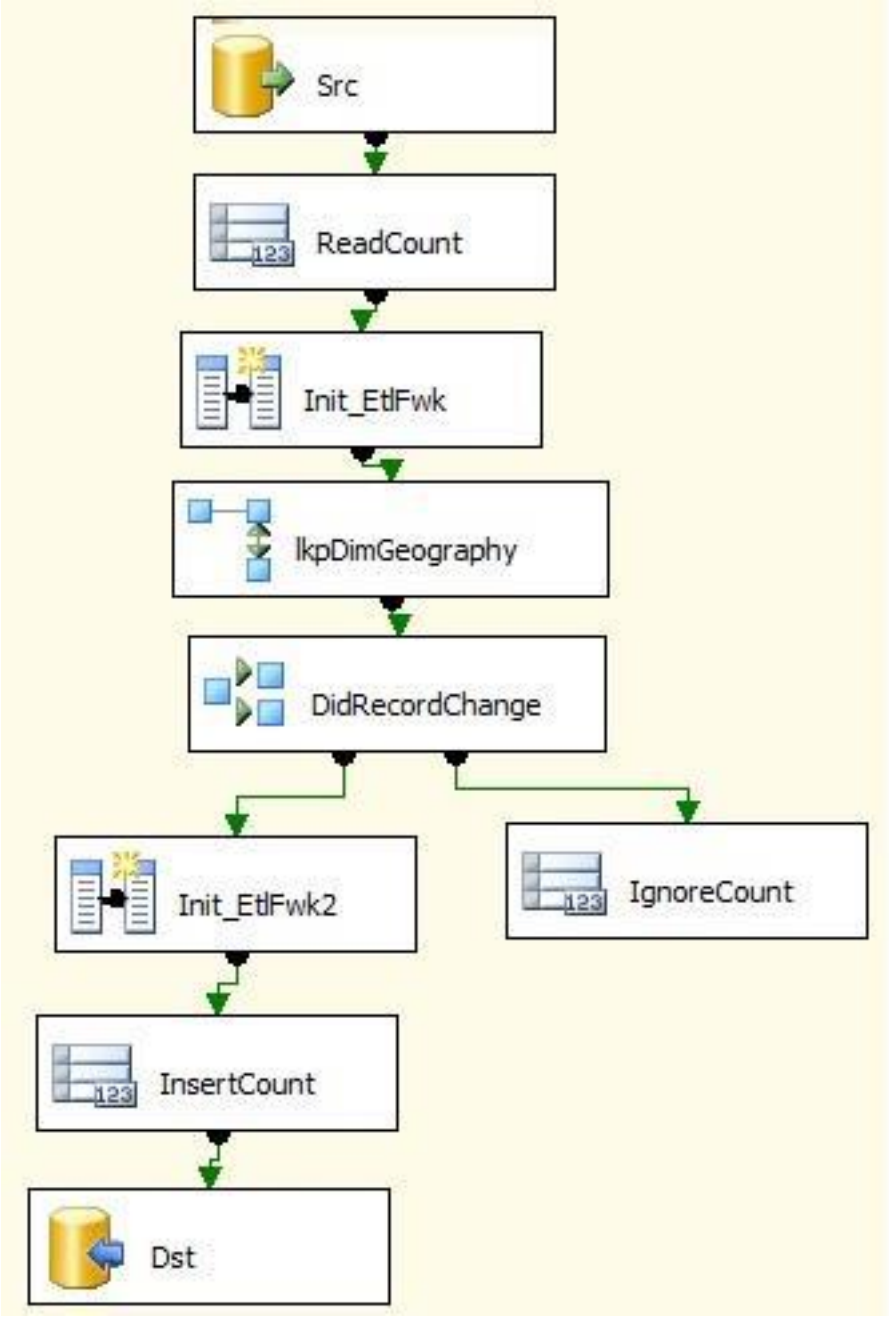
جدولی که به صورت Versioned Insert پر شده است می‌تواند از Versioned Insert Dataflow Pattern استفاده کند. همانند شکل زیر که گردش کار در آن برای کارآئی بیشتر بازنگری شده است.



توجه داشته باشید Data Flow در این روش شامل:

تمامی رکوردهای جدید و تغییر یافته در جدول Versioned Insert قرار می‌گیرند.
این روش دارای Data Flow ساده‌تری نسبت به الگوی Update می‌باشد.

شکل زیر SSIS versioned insert data flow pattern را نشان می‌دهد:



تعدادی نکته در Data Flow فوق وجود دارد که عبارتند از:

در شیء "lkpDimGeography" گزینه "Redirect rows to no match output" با مقدار "Ignore Failures" تنظیم شده است.
شیء "DidRecordChange" بررسی می‌کند چنانچه ستون‌های مبداء و مقصد یکسان باشند، آیا کلید اصلی جدول مقصد Not Null است. اگر این عبارت True ارزیابی شود، رکورد نادیده گرفته می‌شود.
منطق شناسائی تغییرات دربردارنده تغییرات ستون داده ای در مبداء نمی‌باشد.
ستون و تعداد رکوردها مشابه با Data Flow قبلی (ETL Framework) می‌باشد.

4-1-4- Update vs. Versioned Insert

الگوی Versioned Insert نسبت الگوی Update دارای پیاده سازی ساده‌تر و فعالیت‌های I/O کمتری است. از منظر دیگر، جدولی که از الگوی Update استفاده می‌کند، دارای تعداد رکوردهای کمتری است که می‌تواند به معنای Performance بهتر نیز تعبیر شود. ممکن است سوالی مطرح شود، اینکه چرا برای انجام کار به جدول تاریخچه نیاز است؛ این جدول را که نمی‌توان Truncate نمود،

پس چرا به منظور بروزرسانی از جدول اصلی استفاده می‌شود؟ پاسخ این پرسش در این است که جدول تاریخچه، ناظر اطلاعات و ممیزین داده را قادر می‌سازد، تغییرات در طول زمان را پیگیری نمایند.

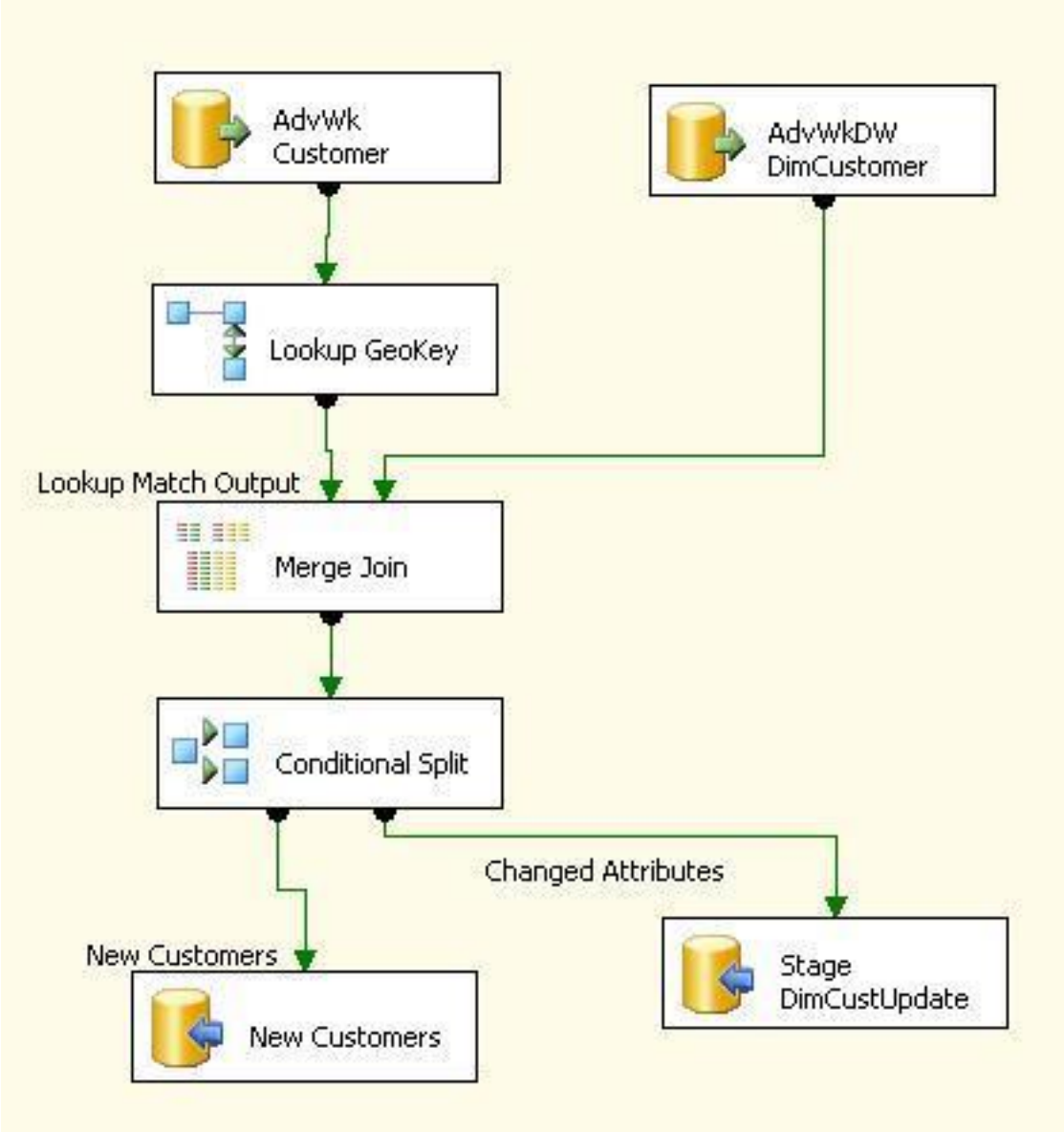
4-2- Dimension Patterns

بروزرسانی Dimension موارد زیر را شامل می‌شود:

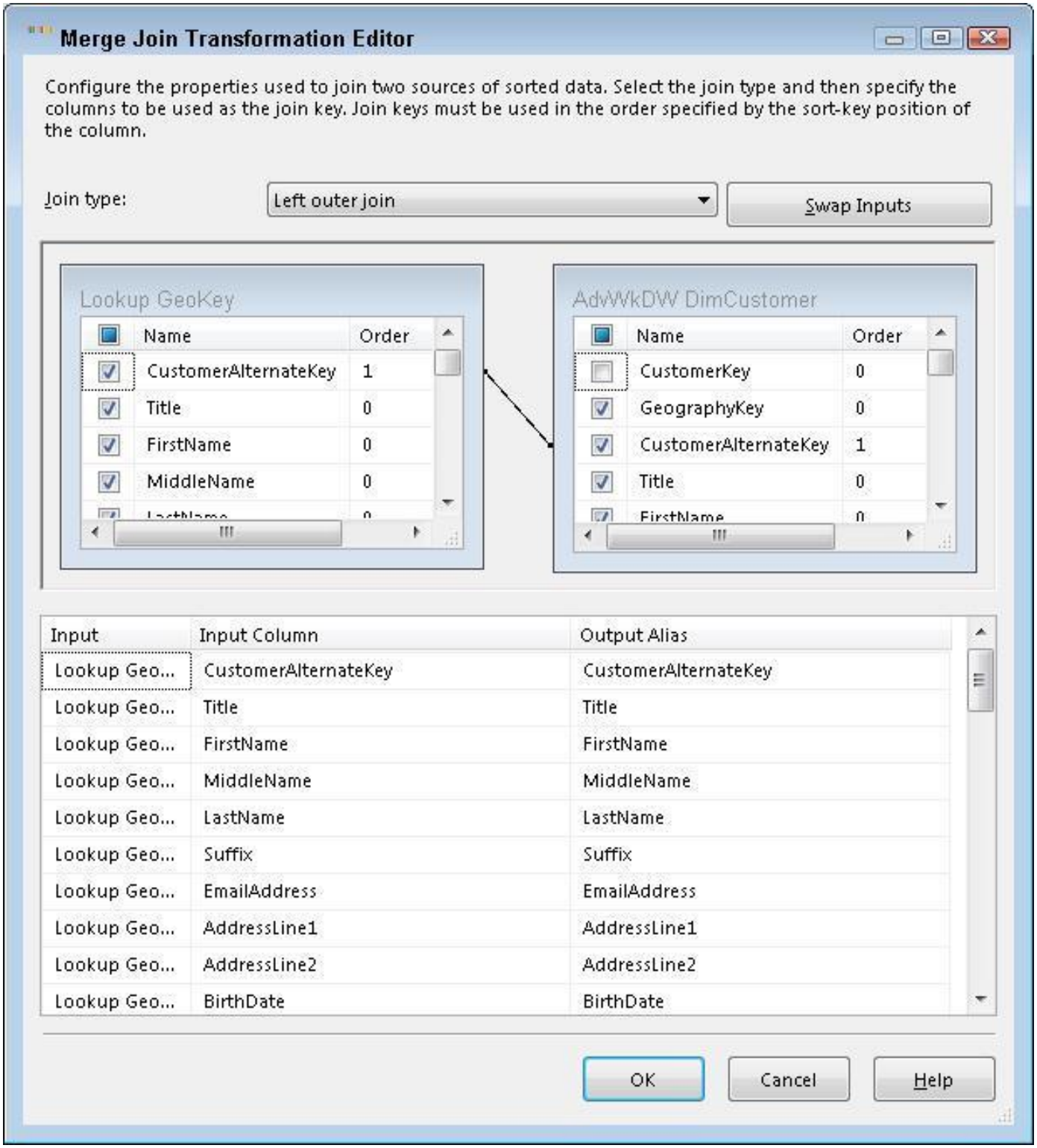
پیگیری تاریخچه
انجام بروزرسانی
تشخیص رکوردهای جدید
مدیریت surrogate keys

چنانچه با یک Dimension کوچک مواجه هستید (با مقدار هزاران رکورد یا کمتر، که با صدها هزار رکورد یا بیشتر ضدیت دارد)، می‌توانید از تبدیل "Slowly Changing Dimension" که بصورت Built-in در SSIS موجود است، استفاده نمائید. به هر روی با آنکه این تبدیل چندین ویژگی محدودکننده Performance دارد، اغلب کارآمدتر از پروسسه هایی که توسط خودتان ایجاد می‌شود. در واقع فرآیند بارگذاری در جداول Dimension با مقایسه داده‌ها بین مبداء و مقصد انجام می‌شود. به طور معمول مقایسه روی یک ورژن جدید و یا مجموعه ای از سطرهای جدید یک جدول با مجموعه داده‌های موجود در جدول متناظرش صورت می‌گیرد. پس از تشخیص چگونگی تغییر در داده ها، یک سری عملیات درج و بروزرسانی انجام می‌شود. شکل زیر نمونه ای از پردازش سریع در Dimension را نمایش می‌دهد؛ که شامل مراحل اساسی زیر است:

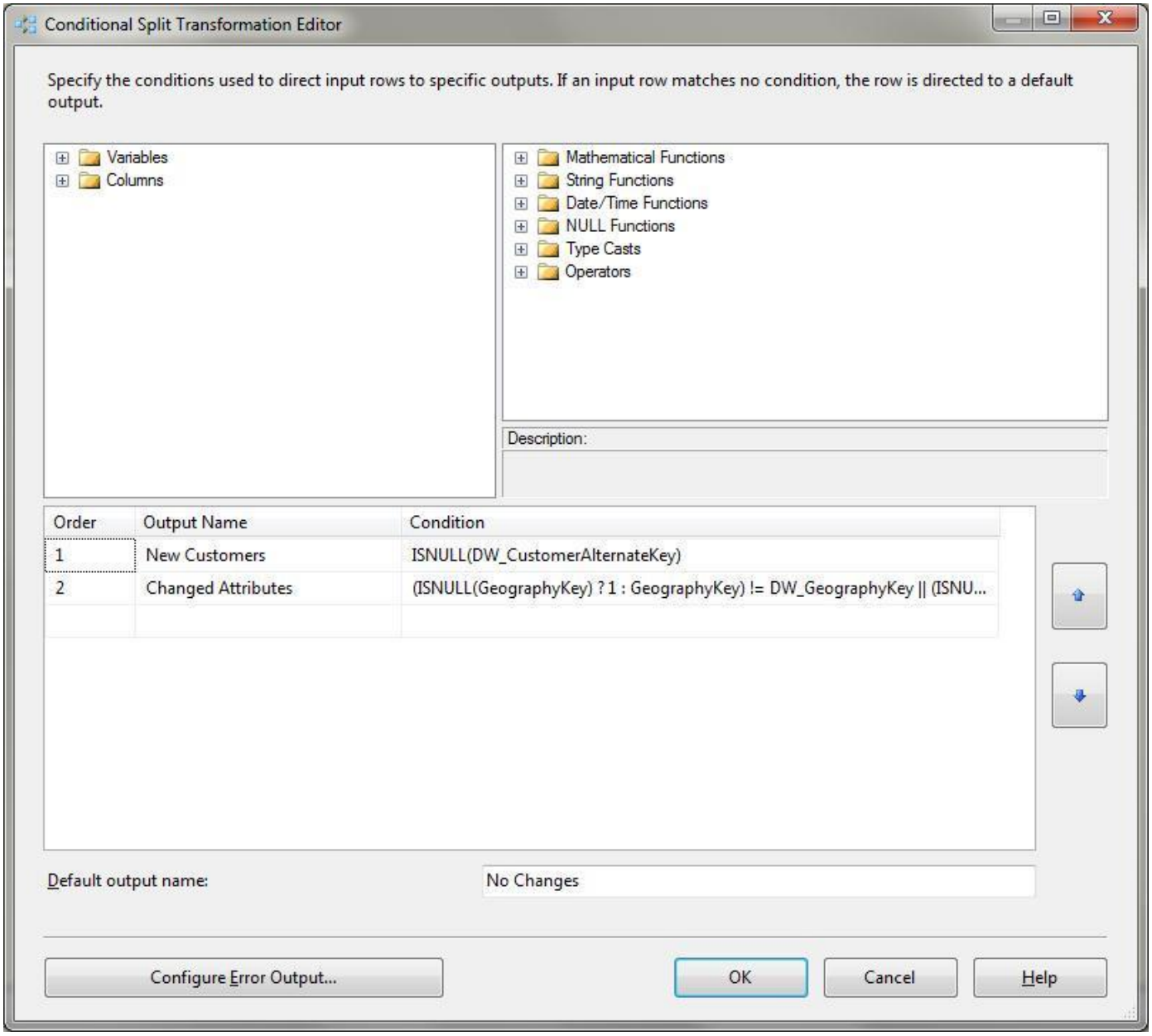
منبع فوقانی سمت چپ، رکوردها را در یک SSIS از یک سیستم مبداء (یا یک سیستم میانی) به شکل Pull دریافت می‌کند. منبع فوقانی سمت راست، داده‌ها را از خود جدول Dimension به شکل Pull دریافت می‌کند.
با استفاده از Merge Join رکوردها از طریق Source Key شان مقایسه می‌شوند. (در شکل بعدی جزئیات این مقایسه نمایش داده شده است.)
با استفاده از یک Conditional Spilt داده‌ها ارزیابی می‌شوند؛ سطرها یا مستقیماً در جدول Dimension درج می‌شوند (منبع تحتانی سمت چپ) و یا در یک جدول عملیاتی (منبع تحتانی سمت راست) جهت انجام بروزرسانی درج می‌شوند.
در گام پایانی (که نمایش داده نشده) مجموعه ای از بروزرسانی بین جدول عملیاتی و جدول Dimension صورت می‌گیرد.



با Merge Join ارتباطی بین رکوردهای مبداء و رکوردهای مقصد برقرار می‌شود. (در این مثال "CustomerAlternateKey"). هنگامی که از این دیدگاه استفاده می‌کنید، خاطر جمع شوید که نوع Join با مقدار "Left outer join" تنظیم شده است؛ بدین ترتیب قادر هستید تا رکوردهای جدید را از مبداء تشخیص دهید؛ از آنجا که هنوز در جدول Dimension قرار نگرفته اند.



گام پایانی به منظور تشخیص اینکه آیا رکورد، جدید یا تغییر یافته است (یا بلاتکلیف است)، مقایسه داده هاست. شکل زیر نمایش می‌دهد چگونه این ارزیابی با استفاده از تبدیل "Conditional Spilt" صورت می‌گیرد.



Conditional Spilt مستقیماً با استفاده از یک Adapter تعریف شده روی مقصد یا یک جدول کاری بروزرسانی که از یک Adapter تعریف شده روی مقصد استفاده می‌کند؛ توسط مجموعه دستور Update زیر، رکوردها را در جدول Dimension قرار می‌دهد. دستور Update زیر مستقیماً با استفاده از روش Join روی جدول Dimension و جدول کاری، مجموعه ای را بصورت انبوه بروزرسانی می‌کند.

```
UPDATE AdventureWorksDW2008R2.dbo.DimCustomer
    SET AddressLine1 = stgDimCustomerUpdates.AddressLine1
    , AddressLine2 = stgDimCustomerUpdates.AddressLine2
    , BirthDate = stgDimCustomerUpdates.BirthDate
    , CommuteDistance = stgDimCustomerUpdates.CommuteDistance
    , DateFirstPurchase = stgDimCustomerUpdates.DateFirstPurchase
    , EmailAddress = stgDimCustomerUpdates.EmailAddress
    , EnglishEducation = stgDimCustomerUpdates.EnglishEducation
    , EnglishOccupation = stgDimCustomerUpdates.EnglishOccupation
    , FirstName = stgDimCustomerUpdates.FirstName
    , Gender = stgDimCustomerUpdates.Gender
    , GeographyKey = stgDimCustomerUpdates.GeographyKey
    , HouseOwnerFlag = stgDimCustomerUpdates.HouseOwnerFlag
    , LastName = stgDimCustomerUpdates.LastName
    , MaritalStatus = stgDimCustomerUpdates.MaritalStatus
    , MiddleName = stgDimCustomerUpdates.MiddleName
```

```
    , NumberCarsOwned = stgDimCustomerUpdates.NumberCarsOwned
    , NumberChildrenAtHome = stgDimCustomerUpdates.NumberChildrenAtHome
    , Phone = stgDimCustomerUpdates.Phone
    , Suffix = stgDimCustomerUpdates.Suffix
    , Title = stgDimCustomerUpdates.Title
    , TotalChildren = stgDimCustomerUpdates.TotalChildren
FROM AdventureWorksDW2008.dbo.DimCustomer DimCustomer
  INNER JOIN dbo.stgDimCustomerUpdates ON
DimCustomer.CustomerAlternateKey = stgDimCustomerUpdates.CustomerAlternateKey
```

Fact Table Patterns 4-3-

جداول Fact به پردازش‌های منحصر به فردی نیازمند هستند، نخست به کلیدهای Surrogate جدول Dimension نیاز دارند تا Measureهای محاسبه شدنی را بدست آورند. این اعمال از طریق تبدیلات Lookup، Merge Join و Derived Column صورت می‌گیرد. با بروزرسانی ها، تفاضل رکوردها و یا Snapshot بیشتر این فرآیندهای دشوار انجام می‌شوند.

Inserts 4-3-1-

روی اغلب جداول Fact عمل درج صورت می‌گیرد؛ که کار متداولی در جدول Fact می‌باشد. شاید ساده‌ترین کار که در فرآیند ساخت ETL صورت می‌گیرد، عملیات درج روی تنها تعدادی از جدول Fact می‌باشد. درج کردن در صورت لزوم بارگذاری انبوه داده ها، مدیریت شاخص‌ها و مدیریت پارتیشن‌ها را شامل می‌شود.

Updates 4-3-2-

بروزرسانی روی جداول Fact معمولاً به یکی از سه طریق زیر انجام می‌گیرد:

از طریق یک تغییر یا بروزرسانی رکورد
از طریق یک دستور Insert خنثی کننده (Via an Insert of a compensating transaction)
با استفاده از یک SQL MERGE

در موردی که تغییرات با فرکانس کمی روی جدول Fact صورت می‌گیرد و یا فرآیند بروزرسانی قابل مدیریت است؛ ساده‌ترین روش انجام یک دستور Update روی جدول Fact می‌باشد. نکته مهمی که هنگام انجام بروزرسانی باید به خاطر داشته باشید، استفاده از روش بروزرسانی مبتنی بر مجموعه است؛ به همان طریق که در قسمت الگوهای Dimension ذکر آن رفت.
در طریقی دیگر (درج compensating) می‌توان اقدام به درج رکورد تغییر یافته نمود، تا ترجیحاً بروزرسانی روی آن صورت گیرد. این استراتژی به سادگی داده‌های جدول Fact میان سیستم مبداء و مقصد را که تغییر یافته اند، به صورت یک رکورد جدید درج خواهد کرد. تصویر زیر مثالی از اجرای موارد فوق را نمایش می‌دهد.

Source ID	Measure Value
12345	80

Source data

Source ID	Measure Value
12345	100

Current fact table data

Source ID	Current Measure Value
12345	100
12345	**- 20**

New fact table data

در آخرین روش از یک دستور SQL MERGE استفاده می‌شود که در آن با استفاده از ادغام و مقایسه، تمامی داده‌های جدید و تغییر یافته جدول Fact، درج و یا بروزرسانی می‌شوند. نمونه ای از استفاده دستور Merge به شرح زیر است:

```
MERGE dbo.FactSalesQuota AS T
USING SSIS_PDS.dbo.stgFactSalesQuota AS S
ON T.EmployeeKey = S.EmployeeKey
AND T.DateKey = S.DateKey
WHEN MATCHED AND BY target
THEN INSERT(EmployeeKey, DateKey, CalendarYear, CalendarQuarter, SalesAmountQuota)
VALUES(S.EmployeeKey, S.DateKey, S.CalendarYear, S.CalendarQuarter, S.SalesAmountQuota)
WHEN MATCHED AND T.SalesAmountQuota != S.SalesAmountQuota
THEN UPDATE SET T.SalesAmountQuota = S.SalesAmountQuota
;
```

اشکال این روش Performance است؛ گرچه این دستور به سادگی عملیات درج و بروزرسانی را انجام می‌دهد ولی به صورت سطر به سطر عملیات انجام می‌شود (در هر زمان یک سطر). در موقعیت هایی که با مقدار زیادی داده مواجه هستید، اغلب بهتر است به صورت انبوه عملیات درج و به صورت مجموعه عملیات بروزرسانی انجام گیرد.

4-3-3- Managing Inferred Members

زمانیکه یک ارجاع در جدول Fact به یک عضو Dimension که هنوز بارگذاری نشده‌است بوجود آید؛ یک Inferred Member تعبیر می‌شود. به سه طریق می‌توان این Inferred Memberها را مدیریت نمود:

رکوردهای جدول Fact پیش از درج اسکن شوند؛ ایجاد هر Inferred Member در Dimension و سپس بارگذاری رکوردها در جدول Fact

در طول عملیات بارگذاری روی Fact؛ هر رکورد مفقوده شده به یک جدول موقتی ارسال شود، رکوردهای مفقوده شده به Dimension اضافه شود، در ادامه مجدداً آن رکوردهای Fact در جدول Fact بارگذاری شوند.

در یک Data Flow زمانی که یک رکورد مفقود شده، بلاتکلیف تعبیر می‌شود؛ آن زمان یک رکورد به Dimension اضافه شود و

Surrogate Key بدست آمده را برگردانیم؛ سپس Dimension بارگذاری شود.

شکل زیر این موارد را نمایش می‌دهد: