عنوان: **آموزش مفاهیم Data Warehouse**
نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۱:۰
آدرس: www.dotnettips.info
گروه‌ها: OLAP, MDX, SSAS, data warehouse

مفاهیم مقدماتی Data Warehouse :

OLTP (Online Transaction Processing) : سیستم‌هایی می‌باشند که برای اهداف اصلی سازمان استفاده می‌شوند و این سیستم‌ها کار پردازش و ذخیره کردن داده‌ها را در OLTP Database انجام می‌دهند. مانند تمامی سیستم‌های ERP,MIS,...

OLTP Database : پایگاه داده‌ی سیستم‌های OLTP می‌باشد. به طور معمول هر تراکنش کاربر در کمترین زمان ممکن برروی این سیستم‌ها ذخیره می‌گردد و در طول روز بارها دستورات (Insert/Update/Delete) برروی آنها انجام می‌شود. این پایگاه‌های داده، همان Main Data ها یا Source System ها می‌باشند.

ETL (extract, transform, and load) : مراحل انتقال داده از OLTP Database به پایگاه داده‌ی Stage می‌باشد. ETL سیستمی می‌باشد که توانایی اتصال به OLTP را دارد و اطلاعات را از OLTP واکشی می‌کند و به پایگاه داده‌ی Stage انتقال می‌دهد. سپس ETL داده‌ها را مجتمع (integrates) کرده و از Stage به DDS (Dimensional Data Source) انتقال می‌دهد .

Retrieves Data : عملیات واکشی داده‌ها طبق یک سری قوانین و قواعد می‌باشد .

برای انجام عملیات ETL دو روش وجود دارد

1. Data مجتمع (Integrate) و تمیز (Data cleansing) شود و در نهایت وارد Data Warehouse گردد.

2. Data وارد Data Warehouse گردد سپس مراحل مجتمع سازی و پاک سازی داده‌ها بر روی داده‌ها در خود Data Warehouse انجام گردد.

Consolidates Data : برخی شرکت‌ها داده‌های اصلی خودشان را در چندین پایگاه داده دارند. در این حالت برای انجام عملیات ETL باید داده‌ها تحکیم و مجتمع شوند و سپس در Data Warehouse ذخیره شوند.

به طور کلی موارد زیر در فرایند ETL در نظر گرفته می‌شود:

1. Data availability : برخی داده‌ها در یک سیستم وجود دارند ولی در سیستم دیگری وجود ندارند و یا تفاوت در نگهداری داده‌ها در سیستم‌های مختلف داریم. مثلا در یک سیستم آدرس در سه فیلد نگه داری می‌شود (کشور-شهر-آدرس) اما در سیستمی دیگر در دو فیلد(کشور-آدرس) نگه داری می‌شود. در این حالت باید ما در ETL راه کار هایی برای مجتمع کردن این موارد در نظر بگیریم.

2. Time ranges : در سیستم‌های مختلف امکان دارد بعدهای زمانی مختلف باشد . مثلا در یک سیستم بررسی‌ها در بازه‌ی ساعتی و در سیستم دیگر بررسی‌ها در بازه‌ی روزانه یا ماهانه باشد . بنابر این در تجمیع داده‌ها باید این مورد مد نظر گرفته شود.

3. Definitions : تعاریف در سیستم‌های مختلف می‌تواند متفاوت باشد. مثلا در یک سیستم، مبلغ کل فاکتور شامل مالیات می‌باشد ولی در سیستمی دیگر این مبلغ فاقد مالیات می‌باشد.

4. Conversion : در فرآیند ETL باید باز از قواعد موجود در سیستم‌های مختلف آگاهی داشته باشیم. مثلا در یک سیستم ممکن است دما را به صورت سانتیگراد و در دیگری فارنهایت نگه داری کنند.

5. Matching : باید بررسی لازم را انجام دهیم که کدام داده مرتبط با کدام سیستم می‌باشد. به عبارت دیگر کدام سیستم مالک داده می‌باشد و دقیقا داده‌ها در کدام سیستم معتبرتر می‌باشند. مثلا پرسنل، هم در سیستم حسابداری می‌باشند هم در سیستم پرسنلی؛ ولی معمولا داده‌های اصلی از سیستم پرسنلی می‌آیند.

Periodically : عملیات واکشی داده‌ها (Retrieves Data) و مجتمع سازی داده‌ها (Consolidates Data) در فرآیند ETL فقط یکبار اتفاق نمی‌افتد و این مراحل در بازه‌های زمانی خاص تکرار می‌گردند. این واکشی و انتقال داده‌ها می‌تواند در روز چند بار تکرار شود یا می‌تواند چند روز یک بار اجرا گردد و این بستگی دارد به سیاست موجود در Data Warehouse .

DDS (Dimensional Data Source) (Data Warehouse) : یک پایگاه داده از نوع نرمال شده (Normalized) یا بعدی (Dimensional) می‌باشد. که داده‌های مجتمع شده و تمیز شده سیستم‌های OLTP را در خود جای داده است. این پایگاه داده برای واکشی‌های سیستم‌های آنالیز داده مورد استفاده قرار می‌گیرد. ورود اطلاعات در Data Warehouse به صورت Batch می‌باشد و به هیچ عنوان مانند پایگاه داده‌های OLTP ویرایش داده‌ها به صورت Online و هر زمان که داده‌ها تغییر می‌کنند، صورت نمی‌گیرد. اطلاعات در Data Warehouse معمولا به صورت تجمیع شده روزانه، ماهانه، فصلی یا سالانه می‌باشد. DDS ها مجموعه ای از Dimensional Data Mart ها هستند. و عمدتا به صورت denormalized می‌باشند.

Dimensional Data Mart : مجموعه ای از جداول Fact , Dimension می‌باشند که در یک بیزینس خاص باهم در ارتباط و مشترک می‌باشند.

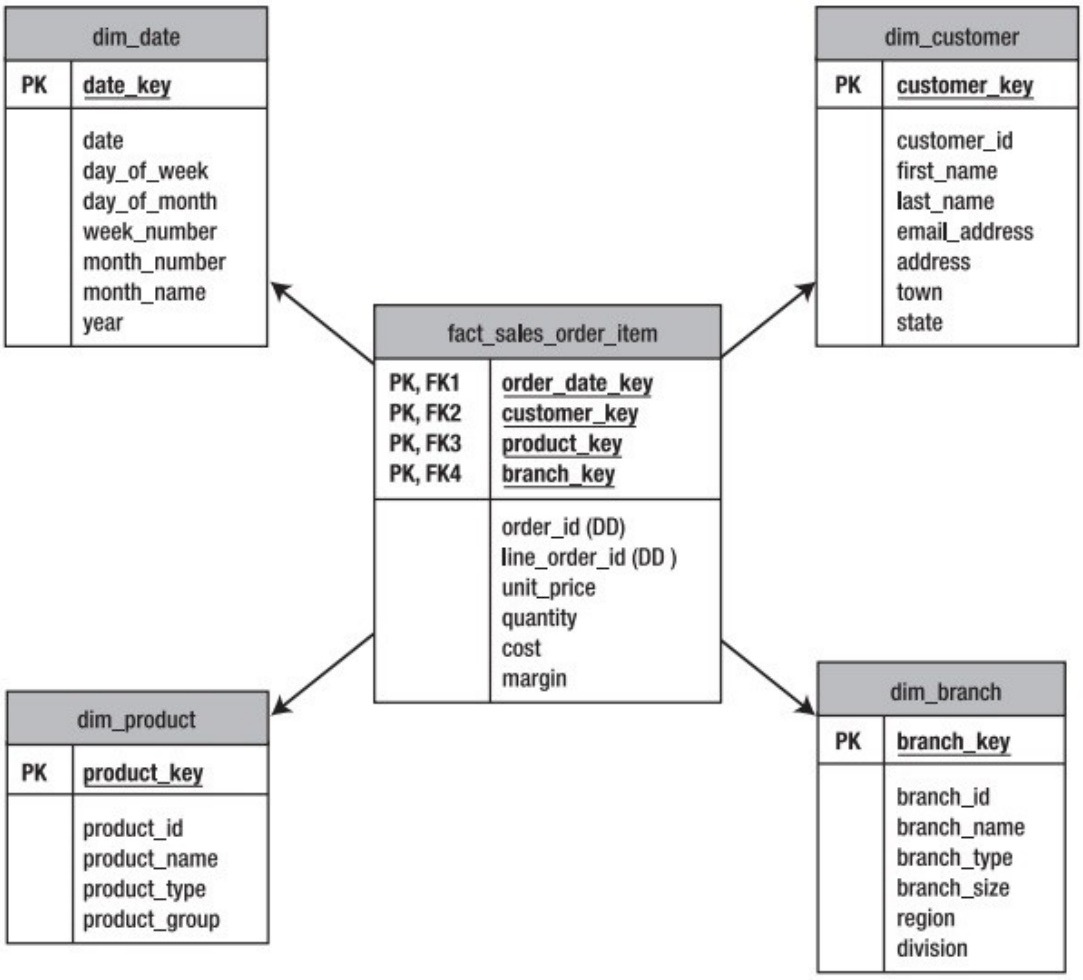
dimensional data store schemas : طراحی‌های مختلفی از جداول Fact , Dimension در DDS وجود دارد که عبارتند از

1. Star schema : ساده‌ترین روش پیاده سازی Data Warehouse

2. Snowflake : در این روش جداول Dimension کمی نرمال سازی بیشتری دارند. سیستم‌های آنالیز داده با این روش بهتر کار می‌کنند.

3. Galaxy schemas : طراحی در این روش بسیار سخت و پیچیده می‌باشد. با این وجود فرایند ETL در این طراحی ساده‌تر انجام می‌شود.

نمونه‌ی طراحی Star به صورت زیر می‌باشد :



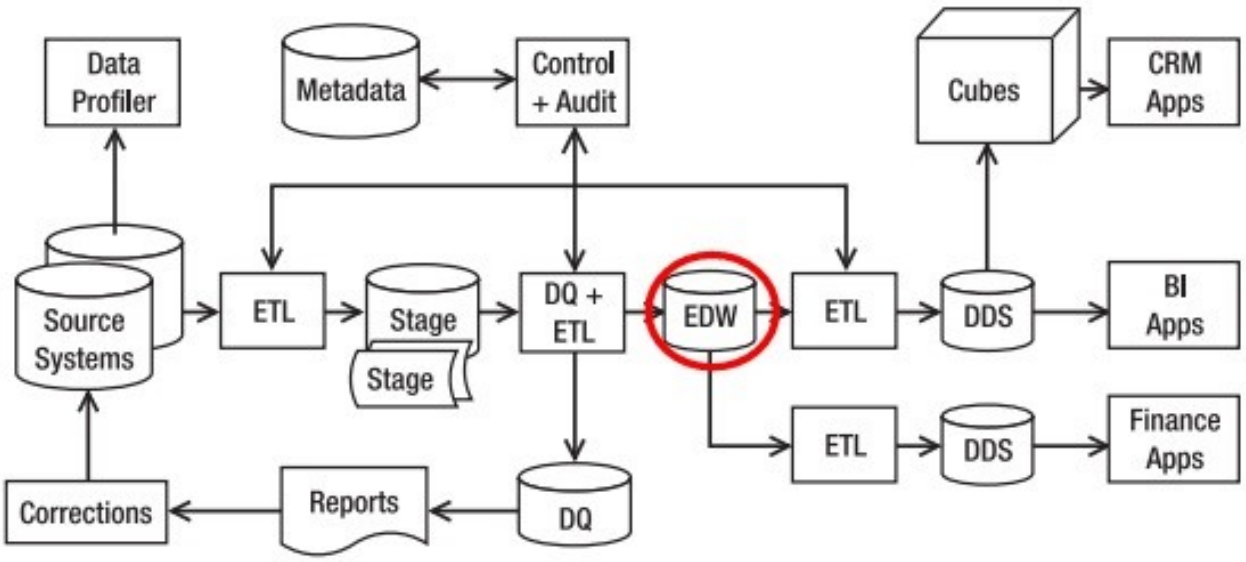
تفاوت‌های DDS و NDS :

1. در DDS ها هیچ گونه نرمال سازی خاصی انجام نمی‌دهیم و عملا تمامی جداول را دینرمال کرده ایم، در حالی که در NDS تمامی جداول تا سطح سوم و گاهی تا سطح پنجم نرمال شده اند.

2. سرعت واکشی و پردازش کوئری‌ها روی DDS خیلی بیشتر از NDS ها می‌باشد.

3. در صورتی که نیاز باشد Data Warehouse های خیلی بزرگ طراحی کنیم با حجم بسیار زیاد توصیه می‌شود از NDS ها استفاده شود در حالی که برای Data Warehouse های کوچک و متوسط بهتر است از DDS ها استفاده شود.

تصویر طراحی یک (EDS) Enterprise Data Source = NDS در زیر آمده است :



History : جداول Data Warehouse میتوانند در طول زمان بسیار بزرگ شوند و دارای تعداد رکورد زیادی گردند. اینکه حداکثر داده‌های چند سال را در Data Warehouse نگه داری کنیم بستگی به سیاست‌های سازمانی دارد که سیستم OLAP برای آن تهیه می‌گردد. استفاده کردن از table partitioning می‌تواند در جبران افزایش تعداد رکورد کمک زیادی به ما بکند.

(slowly changing dimension (SCD : سه روش برای نگه داری سابقه‌ی تغییرات در جداول Dimension وجود دارد.

1. SCD type 1 : هیچ گونه سابقه‌ی تغییراتی را نگه داری نمی‌کنیم

2. SCD type 2 : سابقه‌ی تغییرات در ردیف‌ها نگه داری می‌شود. در این روش هر ردیف، شماره ردیف قبلی را دارد و تعداد نا محدودی از تغییرات را نگه داری می‌کنیم.

3. SCD type 3 : سابقه‌ی تغییرات در ستون‌ها نگه داری می‌شوند و فقط ردیف جاری و آخرین تغییرات را نگه داری می‌کنیم.

Query : فقط ETL حق تغییرات در Data Warehouse را دارد و کاربر نمی‌تواند Data Warehouse را تغییر دهد. البته کاربران حق Query کردن از Data Warehouse را دارند.

دقت داشته باشید که کوئری‌های پیچیده در NDS ها بسیار کندتر از همان کوئری در DDS می‌باشد.

Business Intelligence : مجموعه ای از فعالیت‌ها که در یک سازمان برای شناخت بهتر وضعیت Business آن سازمان انجام می‌شود. نتایج BI کمک بسیاری برای تصمیم گیری‌های تکنیکی و استراتژیکی درون سازمان می‌کند. همچنین کمک به بهبود فرایندهای Business جاری می‌کند.

فعالیت‌های Business Intelligence در سه دسته بندی قرار می‌گیرند :

1. Reporting : گزارشاتی که از Data Warehouse گرفته می‌شود و به کاربر نمایش داده می‌شود و عمدتا این گزارشات به صورت tabular form می‌باشند.

2. OLAP : فعالیت‌های انجام شده روی MDB برای گرفتن گزارشات Drill-Down و ... می‌باشد.

3. Data mining : فرآیند واکشی و داده کاوی داده‌های درون سیستم می‌باشد، که منجر به کشف الگوها و رفتارها و ارتباطات

داده‌ها در سیستم می‌شود. توسط داده کاوی ما متوجه می‌شویم چرا برخی داده‌ها در سیستم تولید شده اند.

a. descriptive analytics : زمانی که از داده کاوی برای شرح وقایع گذشته و حال استفاده می‌شود.

b. predictive analytics : زمانی که از داده کاوی برای پیش بینی وقایع گذشته استفاده می‌شود.

Real time data warehouse : به DW هایی گفته می‌شود که در کمترین زمان، تغییرات OLTP را در خود خواهند داشت. امروزه این نوع DW ها تغییرات 5 دقیقه تا حداکثر 1 ساعت قبل را در خود دارند. برای دسترسی به چنین DW هایی دو راه زیر وجود دارد :

1. بر روی هر جدول، Trigger هایی باشد تا تغییرات را به DW انتقال دهد. (البته برای این منظور باید Business مربوط به ETL را در این تریگرها نوشت)

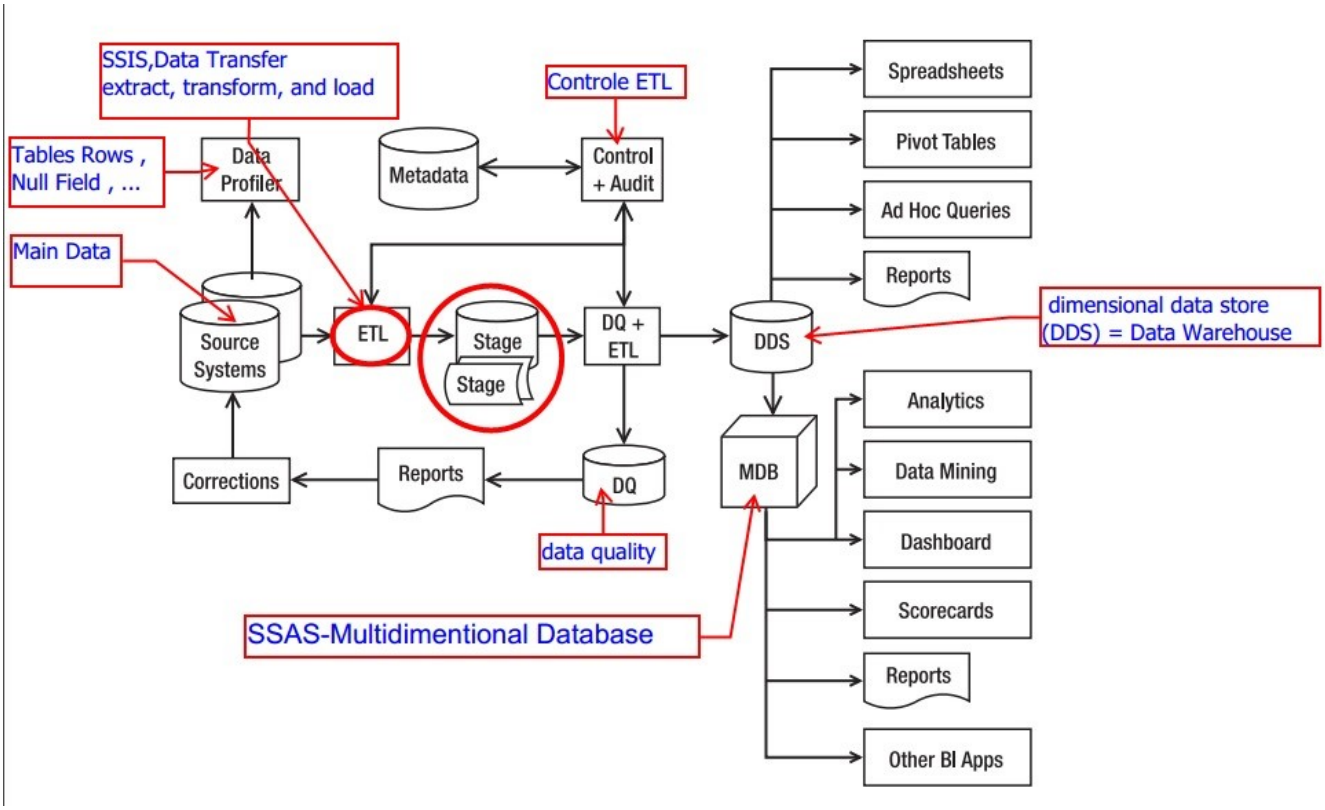
2. سورس برنامه‌های اصلی کاربر (OLTP) تغییر کند تا علاوه بر OLTP Database ها Data Warehouse را هم تغییر دهند.

روش‌های فوق بسیار روی سرعت و کارایی برنامه‌های اصلی تاثیر خواهند گذاشت.

NDS (Normalize Data Source) : در صورتی که طراحی Data Warehouse به صورت Dimensional نباشد و به صورت Normalize باشد، نوع Data Warehouse از نوع NDS می‌باشد.

روش ساخت MDB :

OLTP Database -> ETL -> Stage Database -> DDS (Dimensional Data Source = Data Warehouse) -> SSAS -> MDB



روش ساده‌تر ساخت Data Warehouse :

منظور از Source System همان OLTP Database ها می‌باشد.

به خاطر داشته باشید که Source System ها جزئی از Data Warehouse نمی‌باشند.

از کاربردهای Data Warehouse می‌توان به موارد زیر اشاره کرد

1. Data Mining

2. استفاده در گزارشات

3. تجمیع داده ها

Data Mining کمک به درک بهتر Business جاری در سازمان می‌کند. همچنین منجر به کشف دانش از درون داده‌ها می‌شود.

برای Data Mining می‌توانید از انواع پایگاه داده‌های موجود مانند رابطه ای ، سلسله مراتبی و چند بعدی استفاده کرد . حتا می‌توان از فایل‌های Excel , XML نیز استفاده کرد.

(Customer Relationship Management (CRM :

منظور از مشتری، مصرف کننده‌ی سرویسی است که سازمان شما ارایه می‌کند. یک سیستم CRM شامل تمامی برنامه ایی می‌باشد که تمام فعالیت‌های مشتری را پشتیبانی می‌کند.

(Operational Data Store (ODS :

این پایگاه داده به صورت رابطه ای و نرمال شده می‌باشد و شامل تمامی اطلاعات پایگاه داده ای OLTP می‌باشد که در این پایگاه داده مجتمع شده اند. تفاوت ODS با Data Warehouse در این می‌باشد که داده‌ها در ODS با هر Transaction به روز می‌شوند (سرعت بروز رسانی اطلاعات در ODS بالاتر از DW می‌باشد).

(Master Data Management (MDM :

در یک نگاه می‌توان داده‌ها را به دو دسته تقسیم کرد

1. transaction data

2. master data

transaction data : شامل داده ای transactional در سیستم‌های OLTP می‌باشد.

master data : توضیح دهنده‌ی Business جاری در سازمان می‌باشد.

برای تشخیص این دو نیاز است Business سازمان را به خوبی شناسایی نمایید. به عبارت دیگر رویدادهای Business ی همان transaction data می‌باشند و master data شامل پاسخ‌های این سوال‌ها می‌باشد. چه کسی، چه چیزی و کجا در مورد Business transaction .

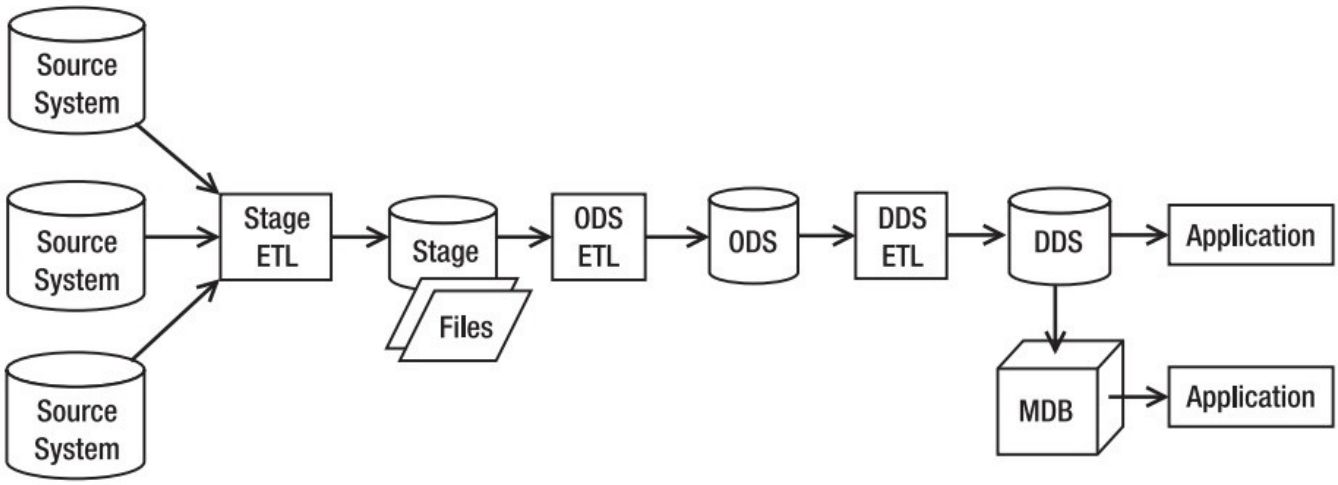
CDI (Customer data integration) : عبارت است از MDM در رابطه با مشتری داده ها. کار این قسمت عبارت است از واکشی، پاک سازی ، ذخیره سازی ، نگه داری و به اشتراک گذاشتن داده ای مشتری می‌باشد.

Unstructured Data : داده ای ذخیره شده در پایگاه داده ، structured Data می‌باشند و داده هایی مانند عکس و فیلم و صوت و ...

SOA (Service-Oriented Architecture) : یک متد ساخت برنامه می‌باشد که در این روش تمامی اجزا برنامه به صورت ماژول هایی دیده می‌شود که در آنها ارتباطات با دیگر سیستم‌ها به صورت سرویس می‌باشد و این زیر سیستم‌ها را می‌توان در پروژه‌های مختلف به کار برد.

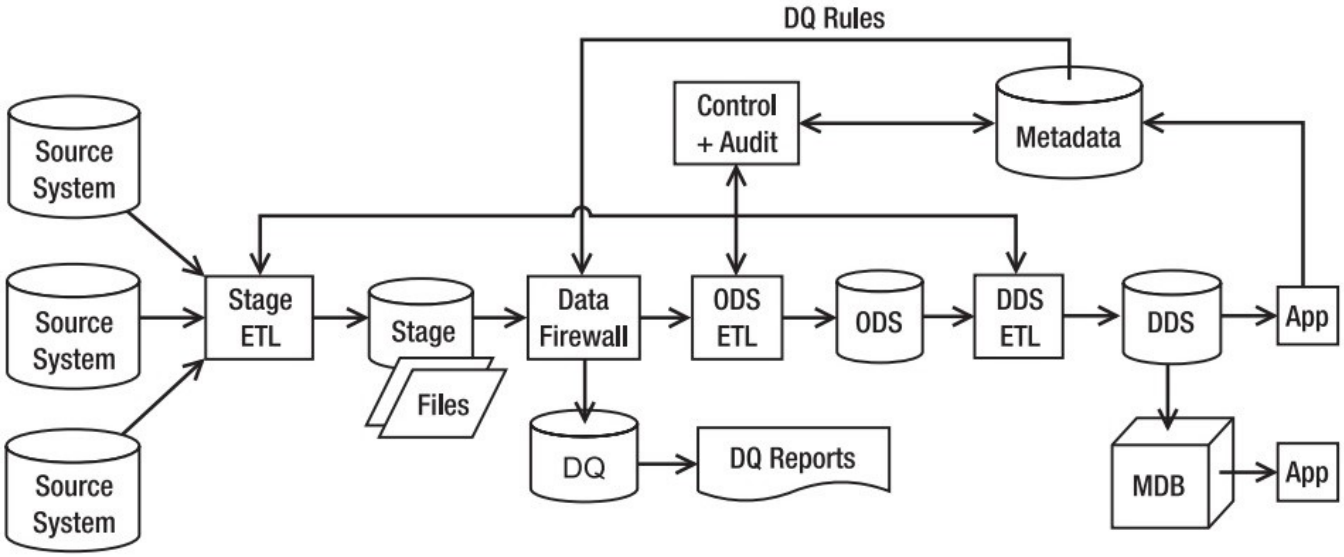
DW : Real-Time Data Warehouse هایی که توسط ETL به روز می‌شوند در هنگامی که یک Transaction روی OLTP اتفاق می‌افتد.

مراحل انتقال داده از OLTP Database به MDB به صورت زیر می‌باشد.



Data quality : مکانیسم اطمینان بخشی از این که در DW داده‌های مناسب و درست وارد می‌شوند. به عبارت دیگر DQ همان firewall برای DW در مقابل داده‌های نامناسب می‌باشد.

برای بهتر مشخص شدن مکان DQ شکل زیر را در نظر بگیرید



نحوه‌ی حرکت داده ای از OLTP به MDB اولین چیزی می‌باشد که شما باید به آن فکر کنید و برای آن روشی را انتخاب نمایید قبل از ساخت Data Warehouse .

چهار روش برای معماری انتقال اطلاعات از OLTP به DW وجود دارد (البته به عنوان نمونه و شما می‌توانید از روش‌های دیگر و طراحی‌های مختلف و ترکیبی نیز بهره ببرید)

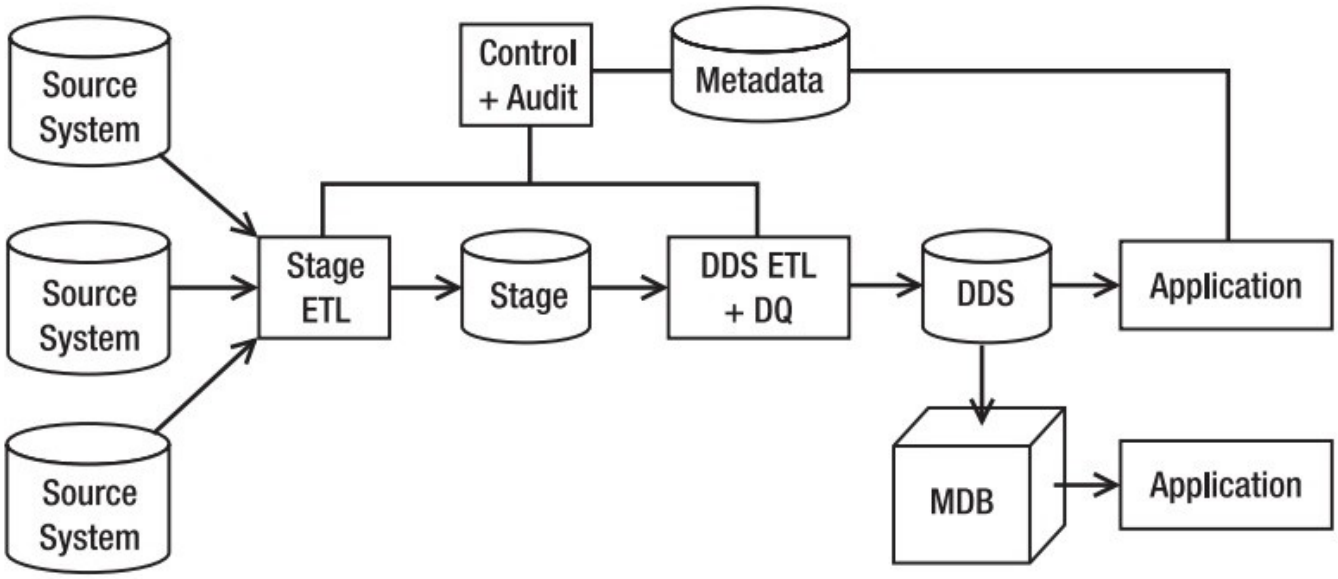
1. single DDS : در این روش فقط DDS , Stage وجود دارد.

2. NDS + DDS : در این روش علاوه بر Stage,DDS از NDS نیز استفاده می‌شود.

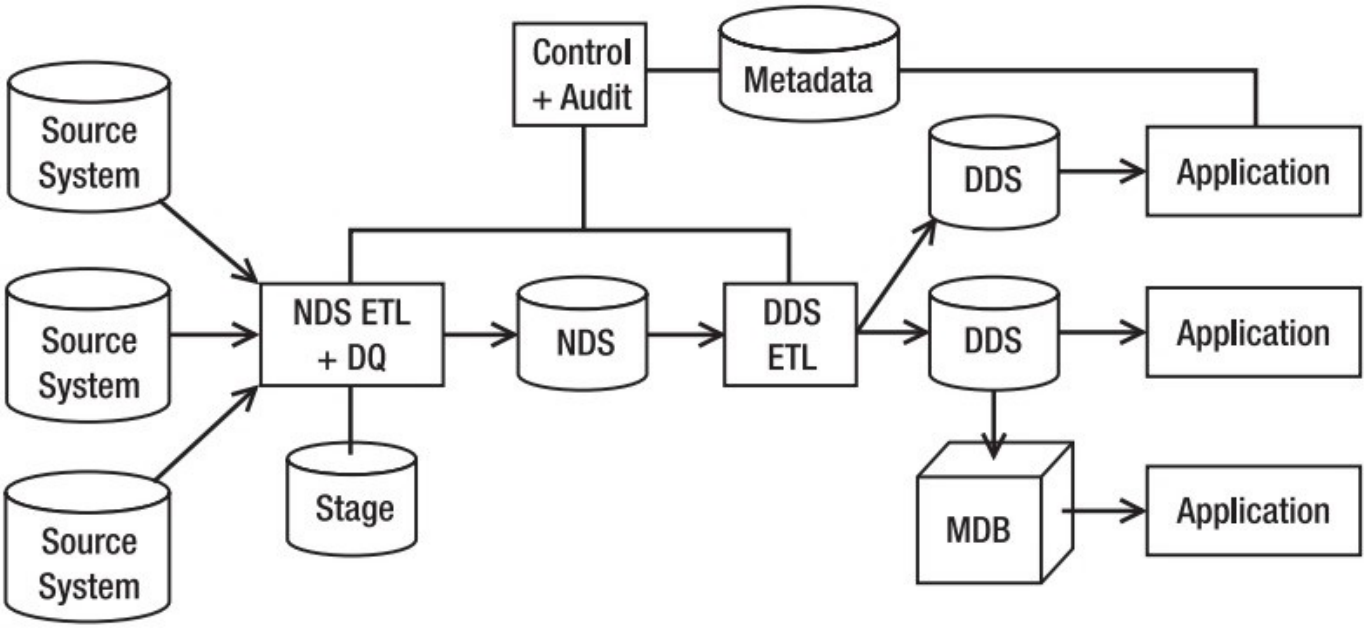
3. ODS + DDS : در این روش از Stage,ODS,DDS استفاده می‌گردد.

4. FDW (federated data warehouse) : استفاده از چندین DW که با هم تجمیع شده اند.

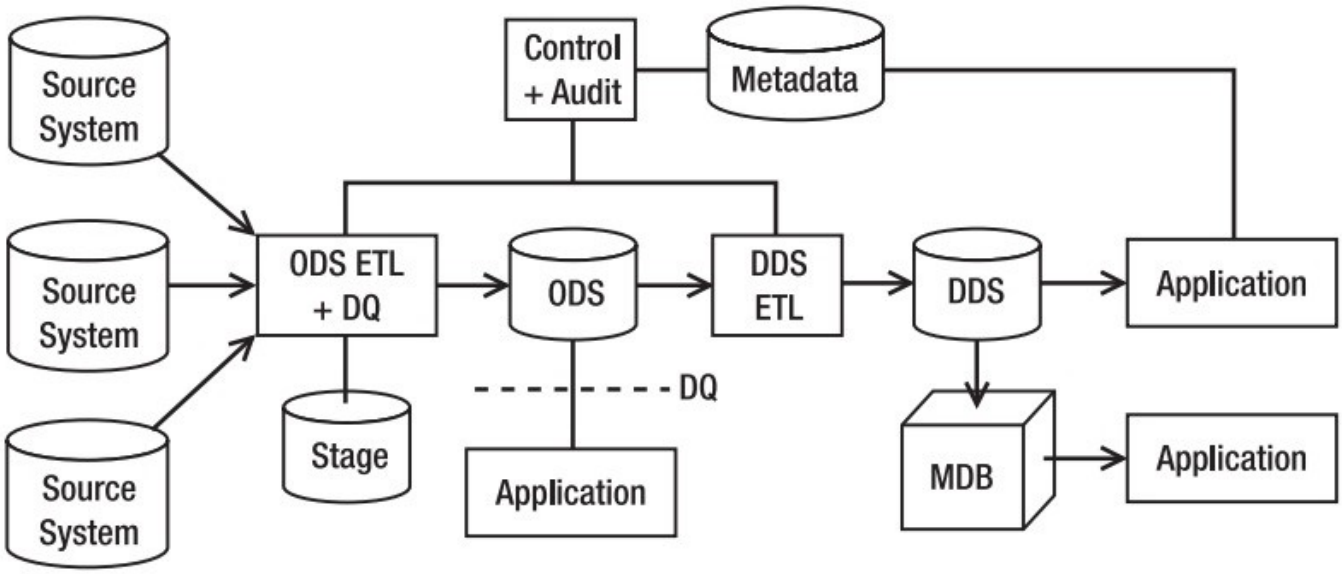
تصویر Single DDS :



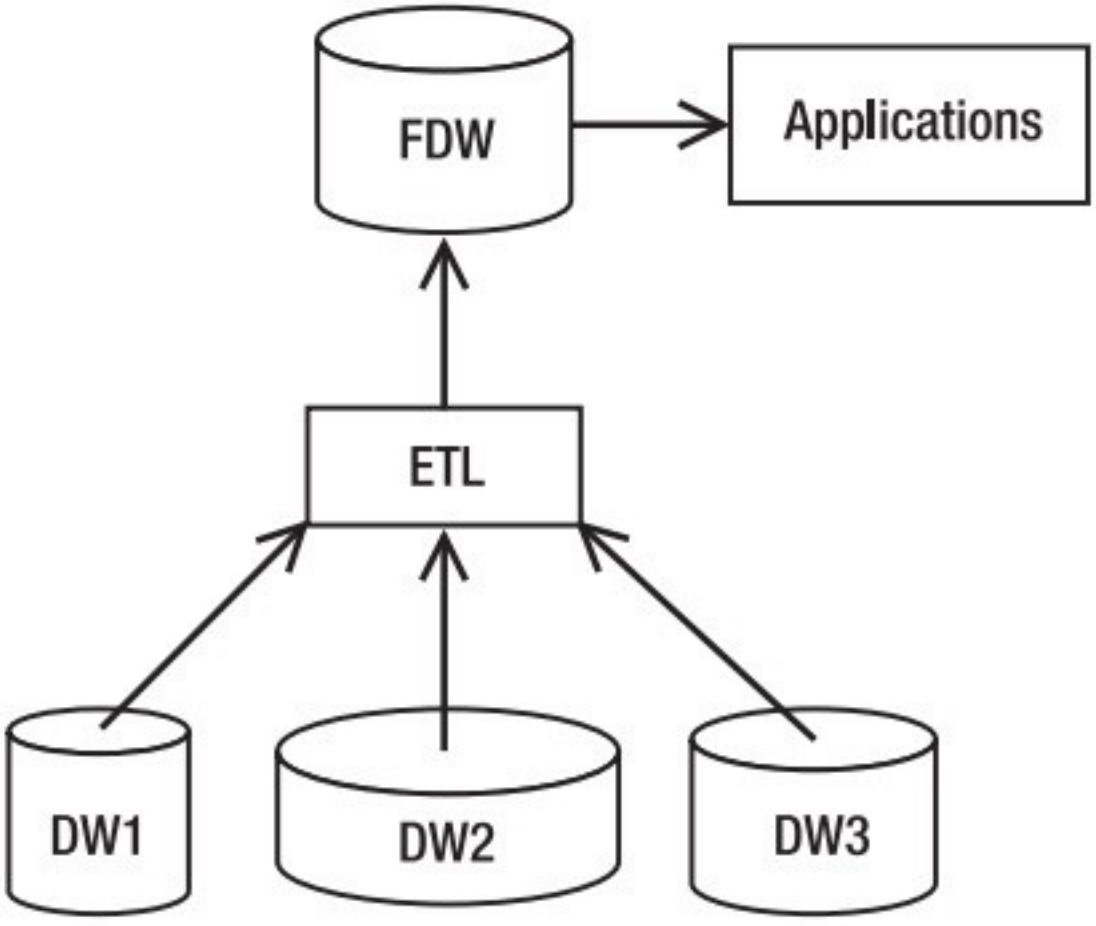
تصویر NDS + DDS :



تصویر ODS + DDS :



تصویر FDW (federated data warehouse) :



منبع : Building a Data Warehouse With Examples in SQL Server انتشارات Apress

نظرات خوانندگان

نویسنده: لیبرتاد
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۳:۳

بسیار عالی
آیا OLTP و DW می‌توانند بر روی یک سرور باشند مثلا" بر روی یک SQL Server باشند

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۵:۴۵

قطعا امکان پذیر می‌باشد. اما توصیه می‌شود اینچنین نباشد. درضمن در نظر داشته باشید که در خیلی از موارد، DBMSهای سیستم‌های OLTP برروی SQL Server نمی‌باشند(مثلا سیستم حسابداری از پایگاه داده‌ی پارادوکس استفاده می‌کند در حالی که سیستم منابع انسانی دارای پایگاه داده‌ی اراکل می‌باشد و ...)

عنوان: **تعریف انبار داده Data Warehouse**
نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۲/۱۵ ۱۰:۳۰
آدرس: www.dotnettips.info
گروه‌ها: data mining, data warehouse

در این مقاله در ادامه‌ی مطلبی که تحت عنوان «آموزش مفاهیم Data Warehouse» توسط آقای شاه قلی منتشر شده بود، به بررسی بیشتر مفهوم انبار داده (Data Warehouse) پرداخته می‌شود.

مقدمه

در سازمان ها، داده‌ها و اطلاعات معمولاً به دو شکل در سیستم‌ها پیاده سازی می‌گردد:

- **سیستم‌های عملیاتی OLTP:**

این سیستم‌ها باعث می‌گردند تا چرخ کسب و کار بگردد. وجود این سیستم‌ها سبب می‌شود تا داده‌های مربوط به کسب و کار، به بانک اطلاعاتی وارد شوند. این سیستم‌ها عموماً:

o به دلیل کوتاهی عملیات دارای سرعت قابل توجهی می‌باشند.
o محیطی جهت ورود داده‌ها می‌باشند.
o معمولاً اپراتورها، استفاده کننده‌های آن هستند.

- **سیستم‌های اطلاعاتی DSS، DW/BI ، OLAP :**

این سیستم‌ها باعث می‌گردند تا چرخش کسب و کار را بنگرید. فلسفه بکارگیری این سیستم‌ها در سازمان این است که اطلاعات مورد نیاز مدیران، از درون داده‌های سیستم‌های عملیاتی موجود، استخراج گردد. این سیستم‌ها عموماً:

o به دلیل آنالیز حجم انبوهی از داده ها، معمولاً کندتر از سیستم‌های عملیاتی می‌باشند.
o محیطی جهت تولید گزارشات تحلیلی و آماری می‌باشند.
o معمولاً مدیران و تصمیم گیرندگان سازمان ها، استفاده کنندگان آن می‌باشند.

سیستم‌های عملیاتی در جامعه ما سابقه بیشتری داشته و متخصصین فناوری اطلاعات عموماً با طراحی و تولید چنین سیستم هایی آشنایی کافی دارند. متاسفانه جایگاه سیستم‌های اطلاعاتی در جامعه ما کمتر شناخته شده و متخصصین فناوری اطلاعات بندرت با مفاهیم و نحوه پیاده سازی آن آشنایی دارند.
این نکته حائز اهمیت است که سیستم‌های اطلاعاتی یک سیستم یا محصول نیستند که بتوان آنها را خریداری کرد. بلکه یک راهبرد (Solution, Approach) هستند و در حقیقت هر راهبردی مربوط به یک نوع کسب و کار (Business) و یا سازمان می‌باشد و نمی‌توان فرمول واحدی را برای حتی سازمان‌های مشابه، ارائه نمود.

گارتنر در ابتدای سال 2011 گزارشی را منتشر کرده که نشان میدهد بازار BI با 9.7 % رشد، ارزشی بالغ بر 10.8 بیلیون دلار داشته، ولی متاسفانه پروژه‌های آن به طور متوسط با 75% شکست مواجه شده است. در حالیکه 4 سال پیش، این رقم حدود 50% بود. این موسسه BI را پنجمین اولویت مدیران IT ذکر کرده است.

مفاهیم و مباحث مربوط به Data Warehouse به اواسط دهه 1980 برمی گردد، به زمانی که IBM تحقیقاتی را در این زمینه شروع کرد و نتیجه آنرا «Information Warehouse» نامید و هنوز هم در برخی منابع از این واژه بجای Data Warehouse استفاده می‌شود. از این پس برای راحتی از اختصار DW بجای Data Warehouse استفاده می‌شود. انبارهای داده جهت رفع نیاز رو به رشد مدیریت داده‌ها و اطلاعات سازمانی که توسط پایگاه‌های داده سیستم‌های عملیاتی غیر ممکن بود، ساخته شدند.

انبار داده به مجموعه ای از داده‌ها گفته می‌شود که از منابع مختلف اطلاعاتی سازمان جمع آوری، دسته بندی و ذخیره می‌شود. در واقع یک انبار داده مخزن اصلی کلیه داده‌های حال و گذشته یک سازمان می‌باشد که برای همیشه جهت انجام عملیات گزارش گیری و آنالیز در دسترس مدیران می‌باشد. انباره‌های داده حاوی داده هایی هستند که به مرور زمان از سیستم‌های عملیاتی آنلاین سازمان، استخراج می‌شوند. بنابراین سوابق کلیه اطلاعات و یا بخش عظیمی از آنها را می‌توان در انباره داده‌ها مشاهده نمود.
از آنجائیکه انجام عملیات آماری و گزارشات پیچیده دارای بار کاری بسیار سنگینی برای سرورهای پایگاه داده می‌باشند، وجود انبار داده سبب می‌گردد که این گونه عملیات تاثیری بر فعالیت برنامه‌های کاربردی سازمان نداشته باشد.

همانگونه که پایگاه داده سیستم‌های عملیاتی سازمان (برنامه‌های کاربردی) به گونه ای طراحی می‌شوند که انجام تغییر، حذف و اضافه داده به سرعت صورت پذیرد، در مقابل انبار داده‌ها دارای معماری ویژه ای می‌باشند که موجب تسریع انجام عملیات آماری و گزارش گیری می‌شود. در حقیقت می‌توان اینگونه بیان نمود که انباره داده یک مخزن فعال و هوشمند از اطلاعات است که قادر است اطلاعات را از محیط‌های گوناگون جمع آوری و مدیریت کرده و نهایتا پخش نماید و در صورت لزوم نیز سیاست‌های تجاری را روی آنها اجرا نماید.

:Bill Inmon

او را پدر DW می‌نامند، از دیدگاه او DW هسته مرکزی چیزی است که او آنرا CIF اختصار (Corporate Information Factory) می‌نامد، که پایه و اساس BI بر مبنای آن قرار دارد. وی از طرفداران Top-Down Design می‌باشد که معتقد است در زمان طراحی باید با دیدی سازمانی، CIF را مدل سازی، ولی بصورت دپارتمانی پیاده سازی کرد (Think Globally, Implement Locally). در این نوع طراحی از DW به Data Mart خواهیم رسید.

:Ralph Kimball Ph.D

به نظر وی DW چیزی نیست جز یک کپی از داده‌های عملیاتی که به طرز خاصی برای گزارشات و تحلیل‌های آماری، آماده و ساختمند شده است. به بیان دیگر DW سیستمی است جهت استخراج، پالایش، تطبیق و تحویل اطلاعات منابع داده ای به یک بانک اطلاعاتی Dimensional و اجرای Query و گزارشات آماری و تحلیلی برای اهداف تصمیم گیری و استراتژیک سازمان.
وی معرفی کننده یکی از اساسی‌ترین مفاهیم طراحی یعنی Dimensional Modeling است؛ ماحصل چنین ایده ای، اساس شکل گیری مدلی است که امروزه کارشناسان آنرا به نام Cube می‌شناسند. وی از طرفداران Bottom-Up Design است که در این نگرش از Data Mart به DW می‌رسیم. این روش به نظر عملی‌تر از روشی می‌باشد که به یکباره DW جامع و کامل برای اهداف سازمانی طراحی و پیاده سازی گردد.

تعریف انبار داده :
W.H.Inmon پدر DW آنرا چنین تعریف می‌کند:

```
The Data Warehouse is a collection of
```

Integrated

```
,
```

Subject-Oriented

```
databases designed to support the DSS function, where each unit of data is
```

Non-Volatile

```
and
```

relevant

```
to some moment in
```

Time

از تعریف فوق دو مورد دیگر نیز به طور ضمنی استنباط می‌شود:

o انبار داده به طور فیزیکی، کاملاً جدا از سایر سیستم‌های عملیاتی است.
o داده‌های DW مجموعه ای Aggregated و Atomic از داده‌های تراکنش‌های سیستم‌های عملیاتی است که سوای کاربرد آنها در سیستم‌های عملیاتی، برای مقاصد مدیریتی نیز استفاده خواهد شد.

به بیان دیگر DW راهبردی است که دسترسی آسان به اطلاعات درست (Right Information)، در زمانی درست (Right Time) ، به کاربران درست (Right Users)، را فراهم می‌آورد تا «تصمیم گیری سازمانی» قابل انجام باشد. DW صرفاً یک محصول نرم افزاری و

یا سخت افزاری نیست که بتوان آنرا خریداری نمود بلکه فراتر از آن و در حقیقت یک محیط پردازشی می‌باشد که کاربران می‌توانند از درون آن اطلاعات مورد نیاز خود را بیابند.
DW اطلاعات خود را از سایر بانک‌های اطلاعاتی از نوع OLTP و یا سایر DWهای لایه پایین‌تر و به صورت دسته ای (Batch) و یا انبوه (Bulk Loading) جمع آوری می‌کند. یک DW به صورت سنتی باید شامل داده‌های Historic سازمان باشد و می‌توان اینگونه بیان نمود که در DW هرچه داده‌های قدیمی‌تری موجود باشد، اعتبار تحلیل‌های آماری سیستم افزایش خواهد یافت.

داده‌های سیستم عملیاتی را نمی‌توان بلافاصله درون بانک اطلاعاتی DW لود نمود، چنین داده هایی باید آماده سازی، پالایش و همگون گردند تا شرایط لود در DW را داشته باشند. حداقل کاری که انتظار داریم یک DW در مورد داده‌ها برای ما برآورده سازد شامل موارد زیر است:

- o استخراج داده‌ها از منابع مختلف (مبدإ)
- o تبدیل داده‌ها به فرمتی یکسان
- o لود داده‌ها به جداول مربوطه (مقصد)

با هر با اجرای پروسه فوق یکی از سه مورد زیر، بسته به نیاز طراحی و محدودیت‌های تکنولوژی رخ خواهد داد:

- o تمام داده‌ها در DW با داده‌های جدید جایگزین خواهند گردید(Full Load, Initial Load, Full Refresh).
- o داده‌های جدید به داده‌های موجود اضافه خواهند گردید Incremental Load (Inserted data).
- o نسخه جدیدی از داده‌های کنونی به سیستم اضافه خواهند گردید Incremental Load (Updated data).

ویژگی‌های داده‌های درون DW

داده‌های DW از نگاه Inmon دارای 4 ویژگی اصلی زیر هستند:

o فقط خواندنی (Non-Volatile):

هیچ رکوردی و یا داده ای Update نخواهد شد و صرفاً رکوردهایی که محتوای مقادیر جدید داده‌ها هستند، به سیستم اضافه خواهند شد.

o موضوع گرا (Subject-Oriented):

منظور از «موضوع» پایه‌های اساسی یک کسب و کار هستند، به شکلی که با حذف یکی از این پایه ها، شاید ماهیت آن کسب و کار از ریشه دگرگون شود. برای مثال موضوعاتی چون «مشتری» و یا «بیمه نامه» برای شرکت‌های بیمه.

o جامع (Integrated):

باید تمامی کدهایی که در سیستم‌های عملیاتی وجود دارند و معانی یکسانی دارند، برای مثال کد جنسیت، در DW به یک روش ذخیره و نمایش داده شوند.

o زمانگرا (Time Variant):

هر رکورد باید حاوی فیلد و یا کلیدی باشد که نمایانگر این باشد که این رکورد در چه زمانی ایجاد، استخراج و ذخیره شده است. از آنجا که داده‌های درون سیستم‌های عملیاتی آخرین و به روزترین داده هر سیستم میباشد، نیازی به وجود چنین عنصری در سیستم‌های OLTP احساس نمی‌گردد، ولی چون در DW تمام داده‌های نسخ قدیمی داده‌های سیستم‌های عملیاتی موجود می‌باشد، باید حتماً مشخص گردد که هر داده ای در سیستم‌های عملیاتی در چه زمانی، چه مقادیری داشته است. این عنصر زمانی کمک می‌کند تا بتوانیم:

- o گذشته را آنالیز کنیم.
- o اطلاعات مربوط به حال حاضر را بدست آوریم.
- o آینده را پیش بینی کنیم.

منبع: کتاب آقای خشایار جام سحر با عنوان بانک داده تجمیعی

Comparison Kimball vs. Inmon Inmon

```
Continuous & Discrete Dimension Management
```

```
Define data management via dates in your data
```

Continuous time

When is a record active

Start and end dates

Discrete time

A point in time

Snapshot

Kimball

Slowly Changing Dimension Management

Define data management via versioning

Type I

Change record as required

No History

Type II

Manage all changes

History is recorded

Type III

Some history is parallel

Limit to defined history

Inmon	Kimball
Subject-Oriented Integrated Non-Volatile Time-Variant	Business-Process-Oriented Stresses Dimensional Model, Not E-R
Top-Down	Bottom-Up and Evolutionary
Integration Achieved via an Assumed Enterprise Data Model	Integration Achieved via Conformed Dimensions
Characterizes Data marts as Aggregates	Star Schemas Enforce Query Semantics

	Inmon	Kimball
Overall approach	Top-down	Bottom-up
Architectural structure	Enterprise-wide DW feeds departmental DBs	Data marts model a business process;enterprise is achieved with conformed dims
Complexity of method	Quite complex	Fairly simple
Data orientation	Subject or data driven	Process oriented
Tools	Traditional ERDs and DIS	Dimensional modeling; departs from traditional relational modeling
End user accessibility	Low	High
Timeframe	Continuous & Discrete	Slowly Changing
Methods	Timestamps	Dimension keys

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.

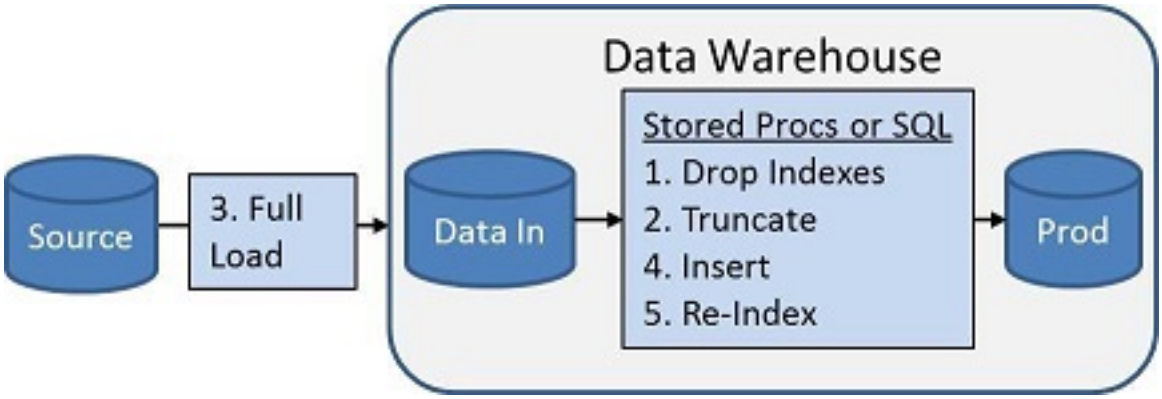
این مقاله بر گرفته از فصل سوم یکی از White Paperهای ماکروسافت با عنوان Microsoft EDW Architecture, Guidance and Deployment Best Practices می‌باشد. که به شرح عملیات Loading در فاز ETL می‌پردازد. از آنجا که به منظور پیاده سازی این نوع پروژه‌ها معمولاً در ایران برون سپاری صورت می‌گیرد و مدیران شرکت‌ها بیشتر 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.

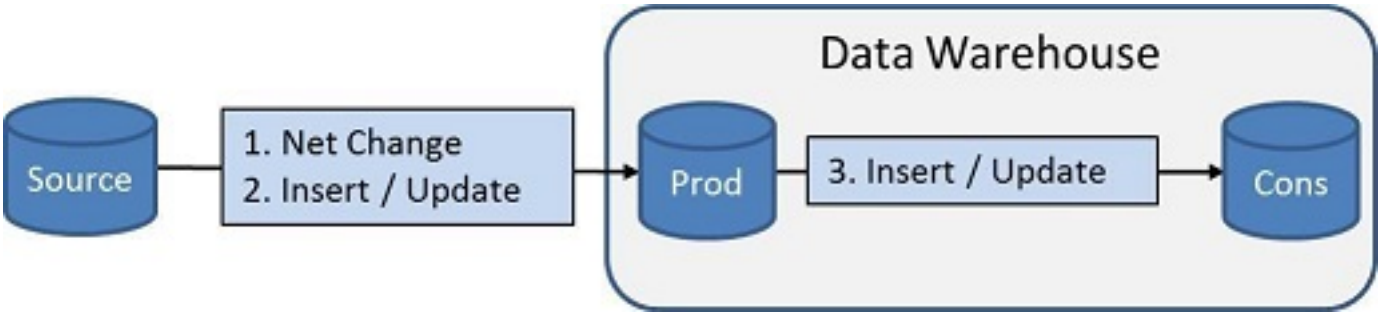
1- Full Load vs Incremental Load نسل‌های اولیه DW (اختصار Data Warehouse) به شکل Full Loads پیاده سازی می‌شدند، به این طریق که هر بار عملیات بارگذاری صورت می‌گرفت، DW از نو دوباره ساخته می‌شد. شکل زیر مراحل مختلف انجام شده در این روش را نمایش می‌دهد:



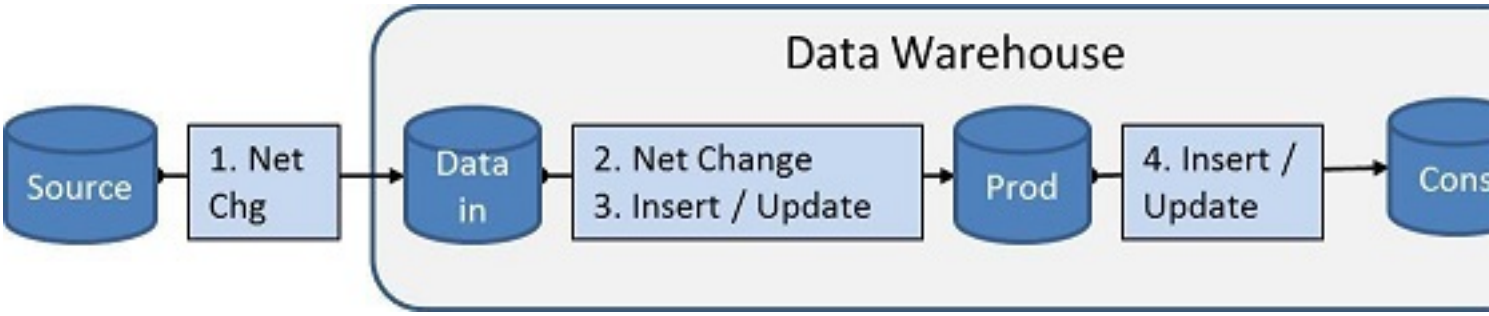
پروسه Full Load شامل مراحل زیر بود:

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.

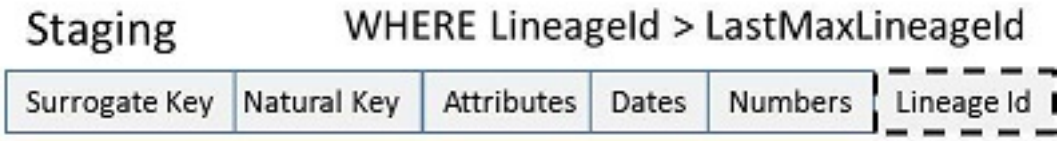
ترکیب این عوامل برای ساخت Incremental Load کارآمد تر، منجر به پیچیده‌تر شدن پیاده سازی و نگهداری آن نیز می‌شود.

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.

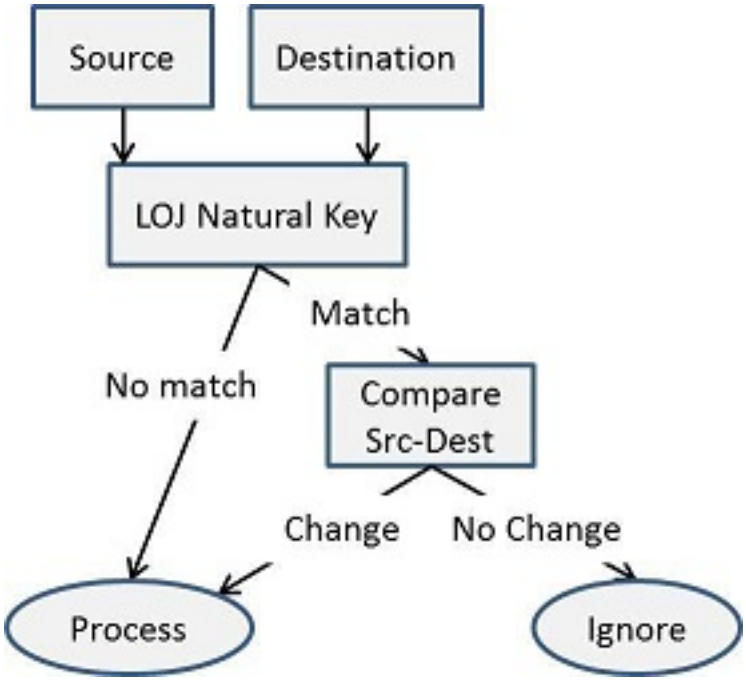
در تکنیک Pull، فرآیند ETL رکوردهای تغییریافته در مبداء را انتخاب می‌کند:
ایده‌آل وجود داشتن یک ستون Last Changed در سیستم مبداء است؛ که از آن می‌توان جهت انتخاب رکوردهای تغییر یافته استفاده نمود.
چنانچه ستون Last Changed وجود نداشته باشد، تمامی رکوردهای مبداء باید با رکوردهای مقصد مقایسه شود.
در تکنیک 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.



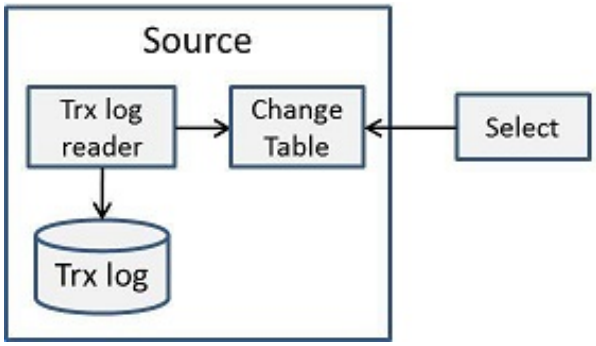
2-2- Pulling Net Changes – No Last Change Column شکل زیر گردش فرآیند را هنگامی که ستون Last Change وجود ندارد؛ نمایش می‌دهد.

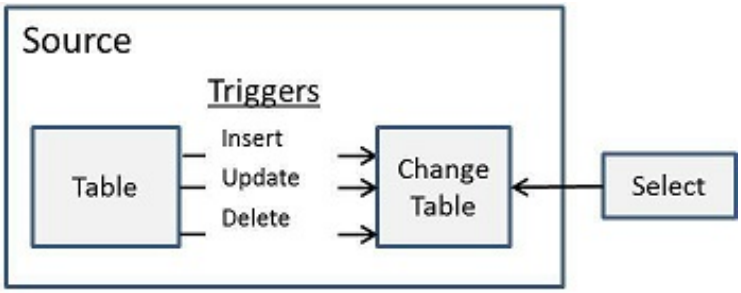


این گردش فرآیند شامل:

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

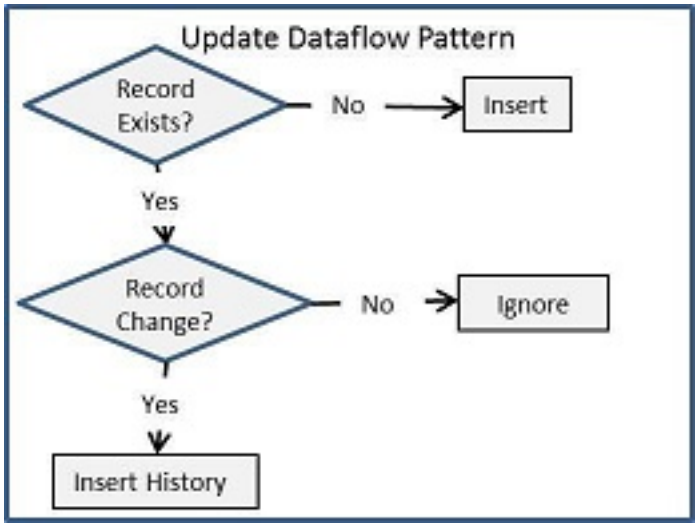
پس از شناسائی رکوردهایی که در مبداء تغییر یافته اند، نیاز داریم تا این تغییرات در مقصد اعمال شود. در این قسمت به معرفی الگوهایی که برای اعمال این تغییرات وجود دارد می‌پردازیم.

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 #

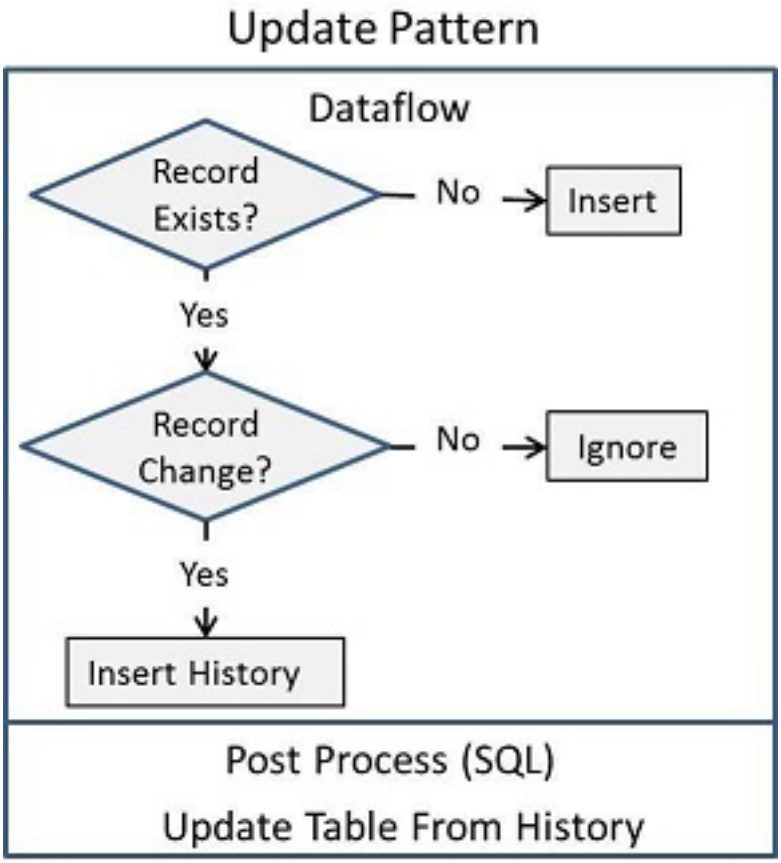
این ستون‌ها به شرح زیر هستند:

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.

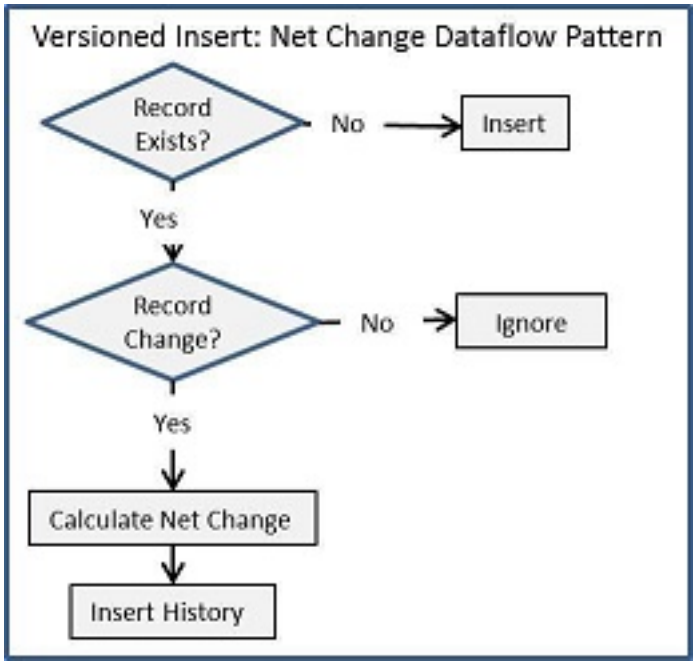
یک روش دیگر برای در نظر گرفتن موارد فوق؛ اضافه کردن یک جدول برای درج ورژن‌ها به الگوی Update است که در شکل زیر نشان داده شده است.



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.

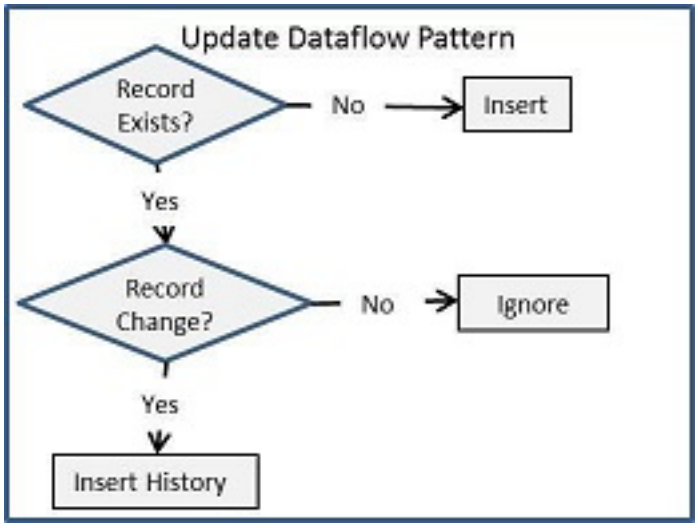
4-1- Basic Data Flow Patterns

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

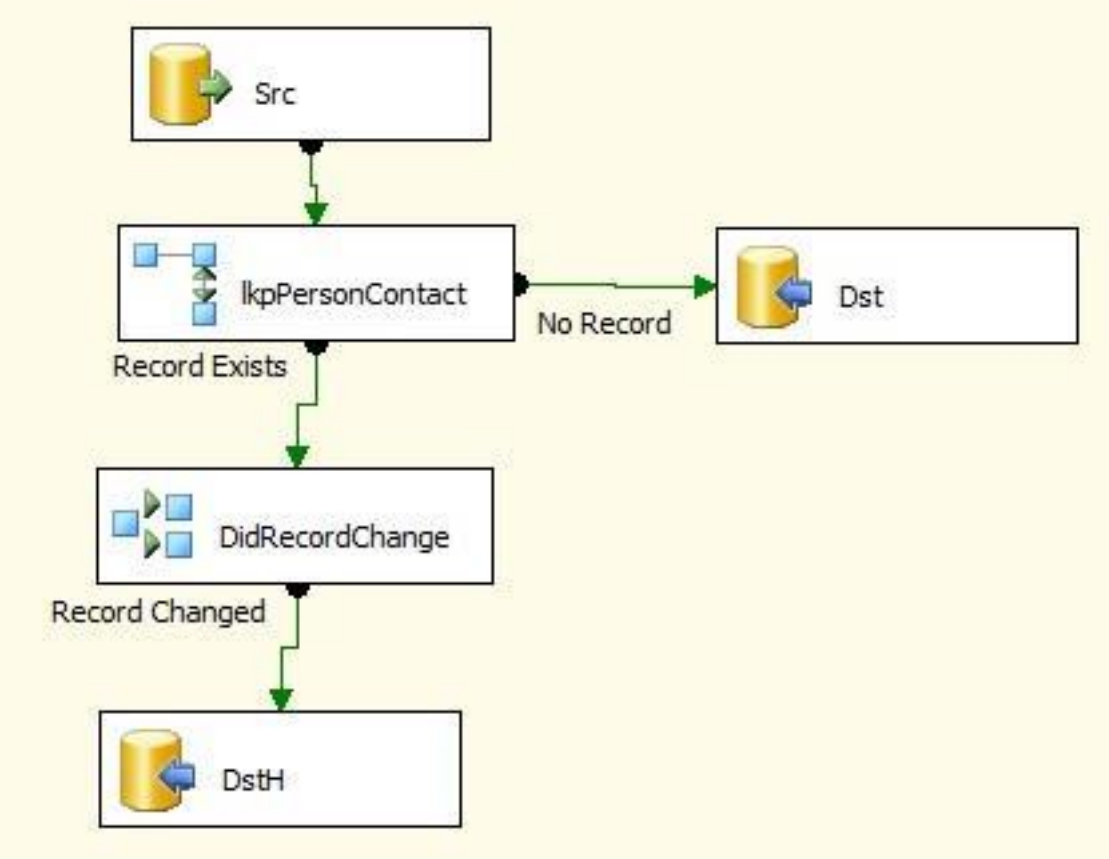
مطابق تصویر زیر جدولی که تنها حاوی ورژن فعلی رکورد هاست؛ از Update Dataflow Pattern استفاده می‌کند.



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.

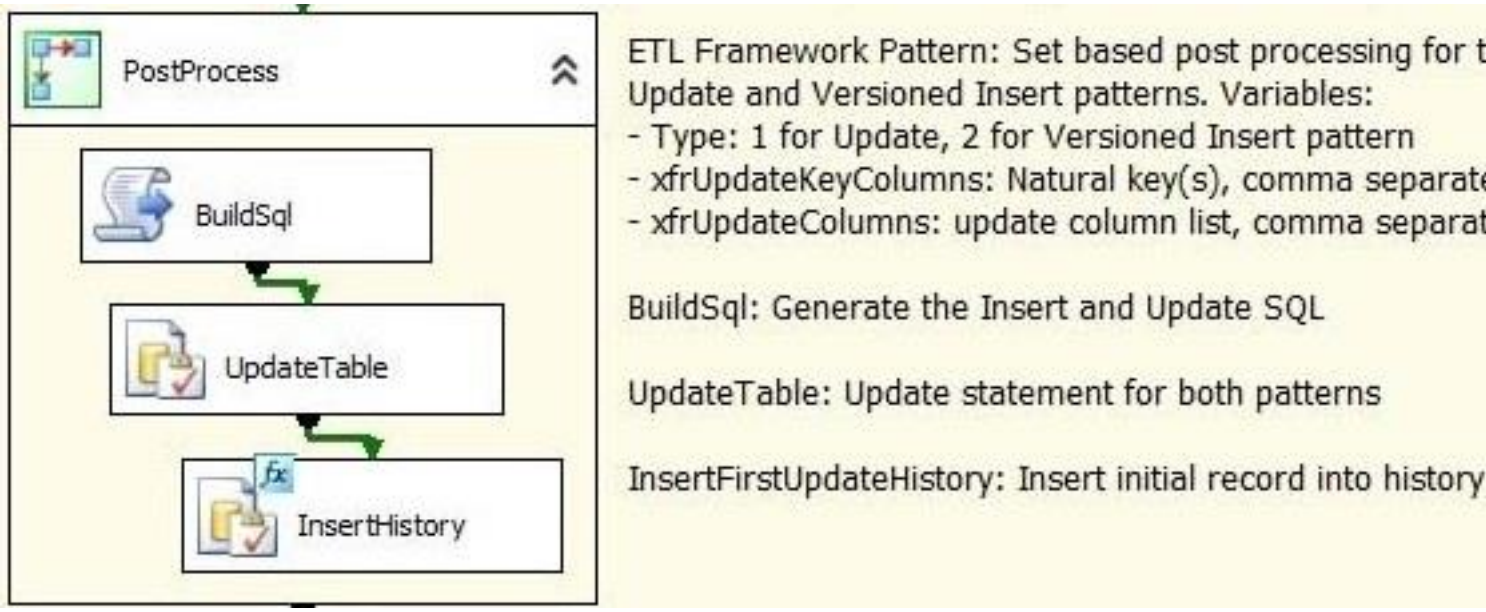
Detection Inserts:

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:

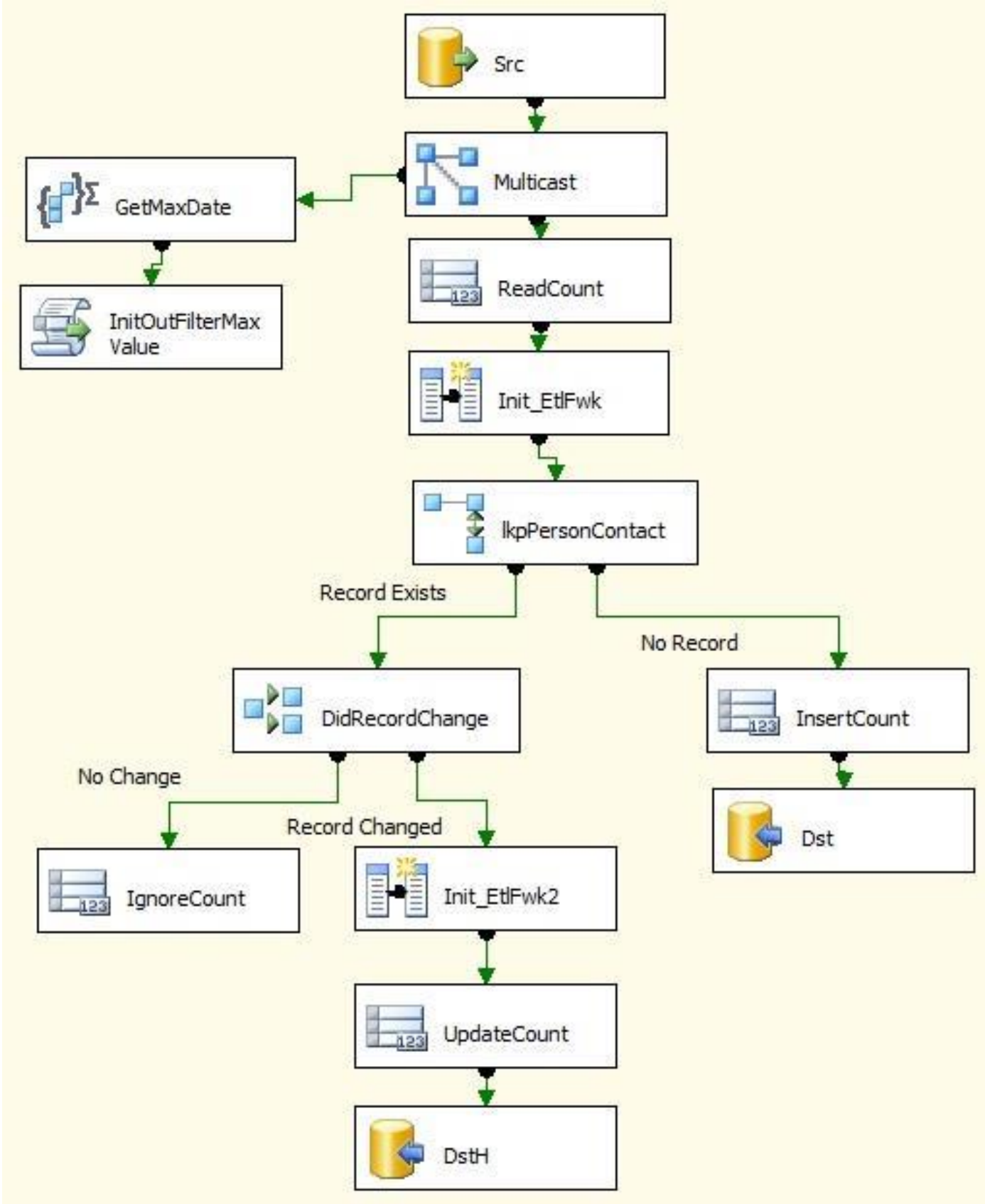
بروزرسانی رکوردهای جداول با استفاده از رکوردهای درج شده در جدول تاریخچه.
درج تمامی رکوردهای جدید (نسخه اولیه و در درون جدول تاریخچه). کلید اصلی جداولی که ستون آنها IDENTITY است مقدار نامشخصی دارد؛ تا زمانی که درج صورت گیرد، این به معنای آن است که پیش از انتقال آنها به جدول تاریخچه نیاز است منتظر درج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

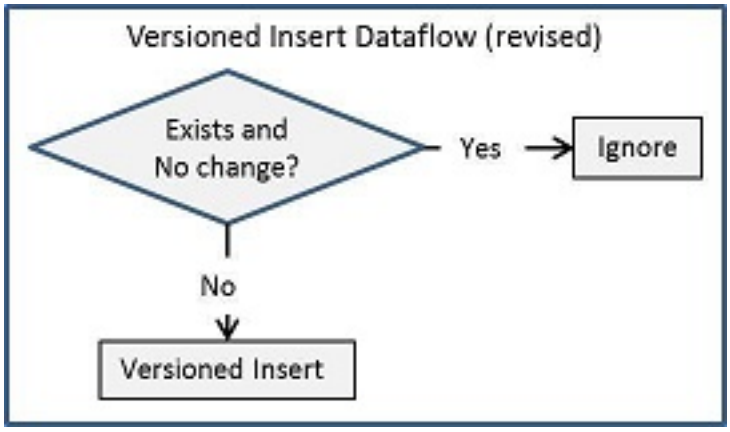
پشتیبانی از ستون هایی که کارهای ممیزی و نظارت بر داده‌ها را آسانتر می‌کنند.
تعداد سطرها شاخص مناسبی است که می‌تواند بهبود آن Data Flow خاص را فراهم کند. ناظر اطلاعات با استفاده از تعداد رکوردها می‌تواند ناهنجاری‌ها را شناسائی کند.

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-

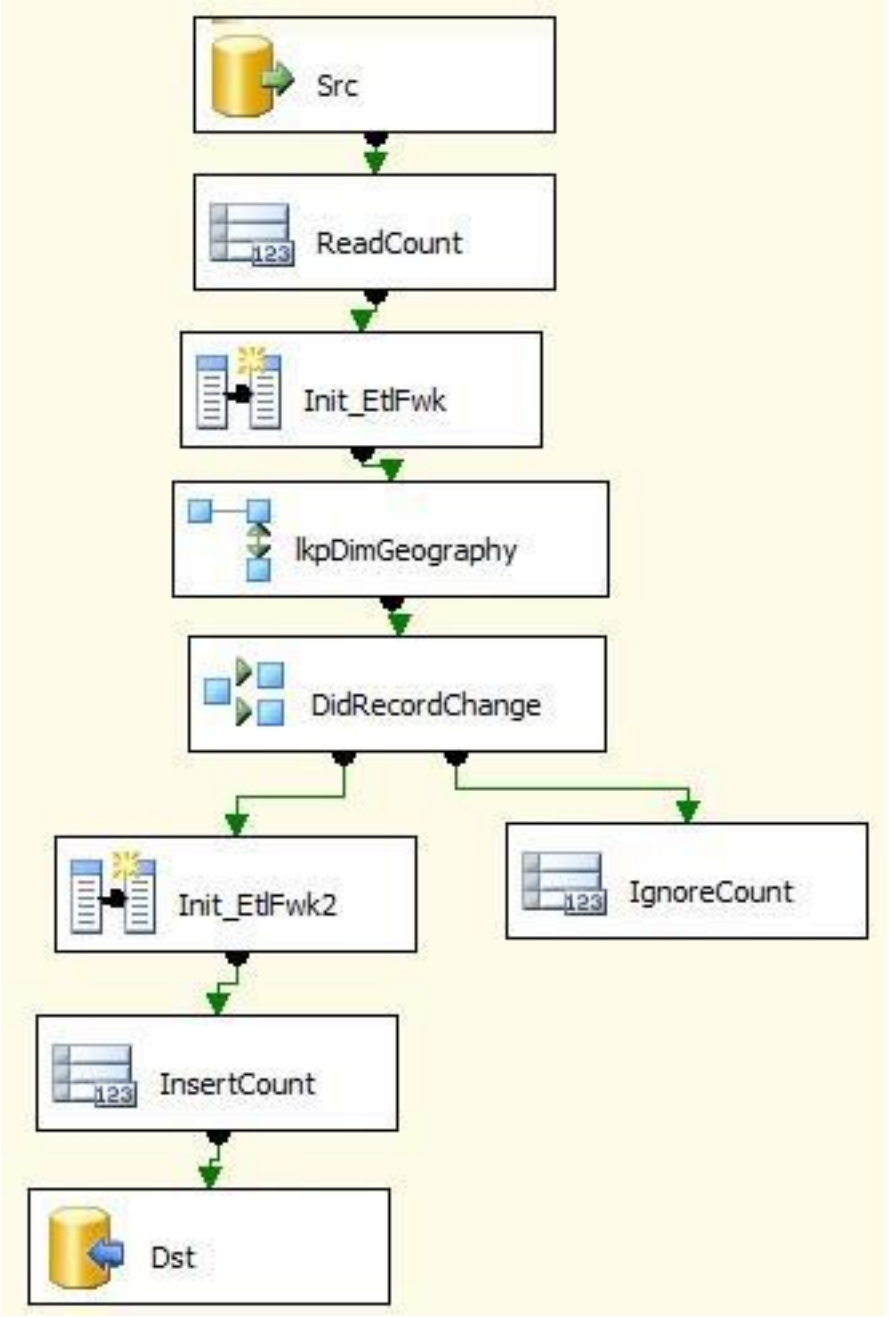
جدولی که به صورت Versioned Insert پر شده است می‌تواند از Versioned Insert Dataflow Pattern استفاده کند. همانند شکل زیر که گردش کار در آن برای کارآئی بیشتر بازنگری شده است.



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.

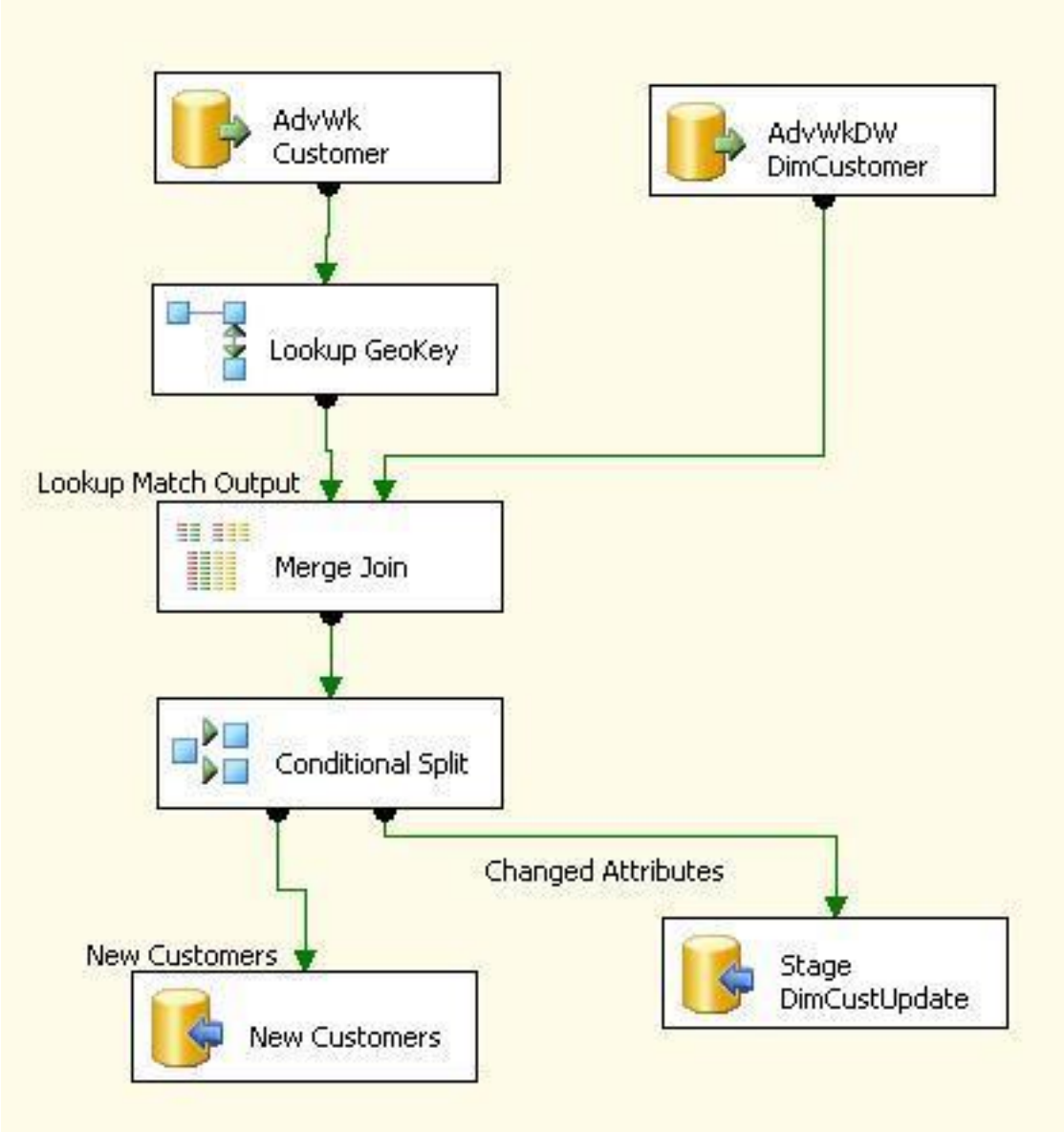
4-2- Dimension Patterns

بروزرسانی Dimension موارد زیر را شامل می‌شود:

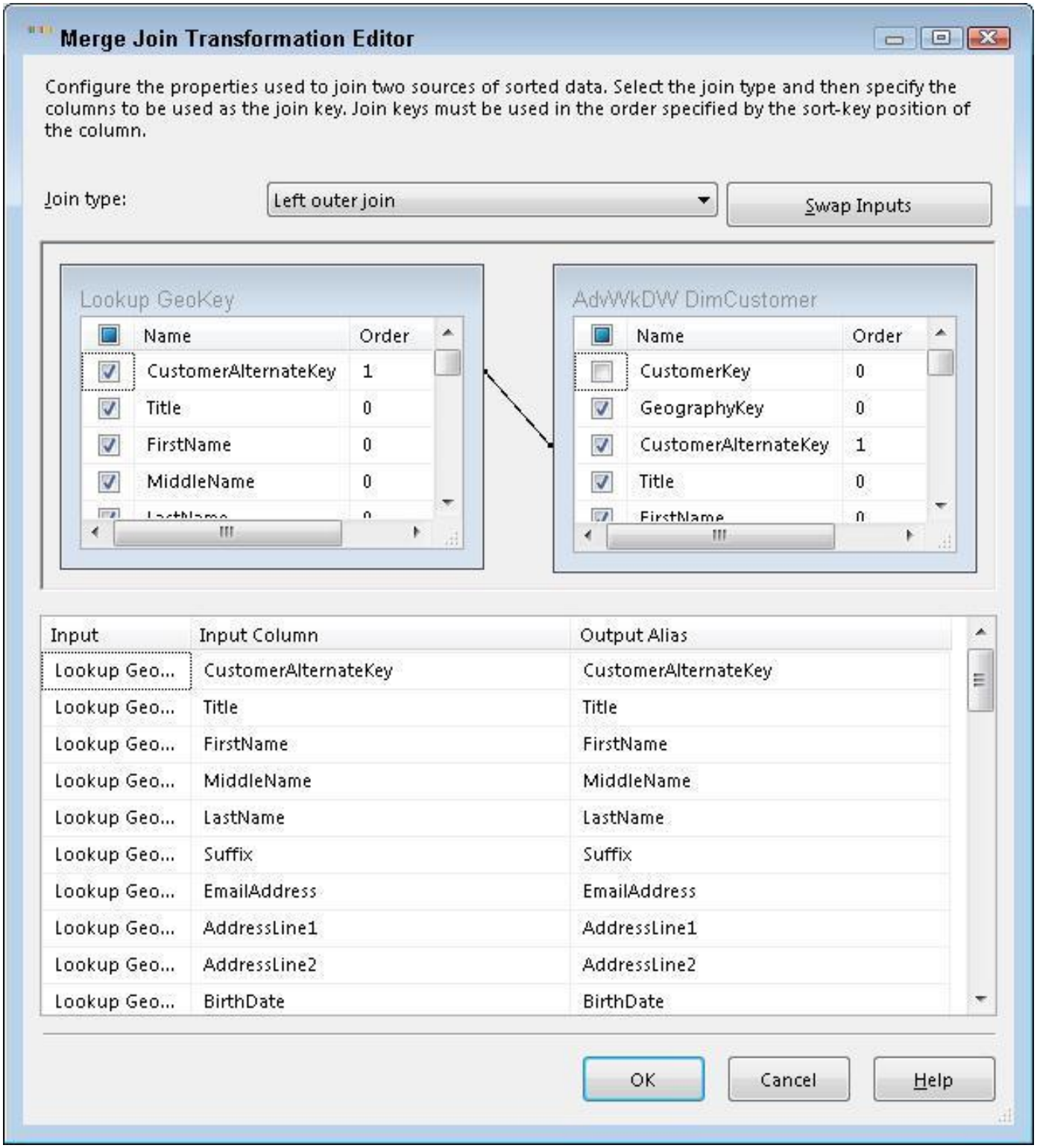
پیگیری تاریخچه
انجام بروزرسانی
تشخیص رکوردهای جدید
مدیریت surrogate keys

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:

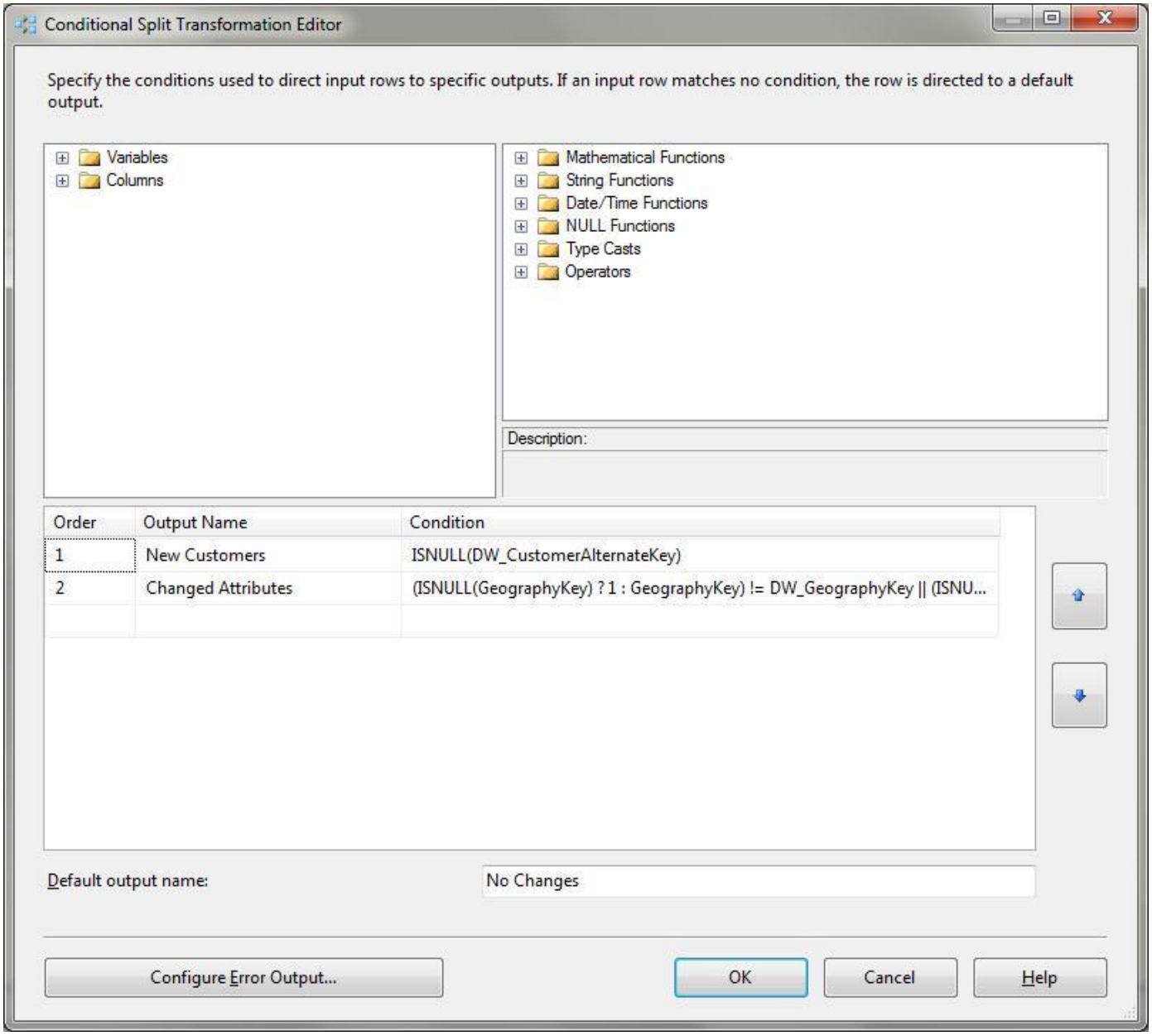
منبع فوقانی سمت چپ، رکوردها را در یک SSIS از یک سیستم مبداء (یا یک سیستم میانی) به شکل Pull دریافت می‌کند. منبع فوقانی سمت راست، داده‌ها را از خود جدول Dimension به شکل Pull دریافت می‌کند.
با استفاده از Merge Join رکوردها از طریق Source Key شان مقایسه می‌شوند. (در شکل بعدی جزئیات این مقایسه نمایش داده 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
```

4-3- Fact Table Patterns

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.

4-3-1- Inserts

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.

4-3-2- Updates

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: