بنام خدا

پایگاه داده ۲

Implementation of Data Warehouse

بصیری
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

ایجاد جداول در انبارداده

- بعد از اینکه طراحی انبارداده ایجاد شد، تمامی بعدها و فکت ها آماده شده اند.
- برای هر فکت یا بعد فیلدها مشخص شده است.
- نوع هر کدام از فیلدها باید مشخص شده باشد.
- لازم است بر اساس طراحی انجام شده، متادیتای جداول انبارداده ایجاد شود.
- بهتر است طول فیلدها را کمی بزرگتر از سورس در نظر بگیریم. چرا؟**

:**
چرا؟ مثلا توی سورس گردش بدهکار داریم این گردش بدهکار رو اگه بخوایم توی انبار داده بزنیم یک سایز بیشتر در نظر بگیر توی انبار داده
طول اسم کاربر توی جدول های سورس 50 بوده اینو توی انبار داده مثلا 60 بگیر
چرا؟
یکی بحث aggregation است و نوع مژری که توی فکت می زنیم ینی تجمعی می زنیم یا تراکنشی می تونه تعیین کننده باشه
بخاطر تغییر یک فیلد در جدول سورس ما مجبور نباشیم توی انبار داده کار سنگینی بکنیم به دلایل مختلفی ممکنه این اتفاق بیوفته مثلا طول اسم رو گرفتیم 60 کاراکتر ما میگیم توی انبار داده 100 کاراکتر ولی اگه به هر دلیلی خواستیم توی سورس سایز رو عوض بکنیم توی انبار داده دیگه دچار مشکل نشیم مثلا اگر خواستیم توی سورس فیلدی 80 داشتیم ولی اینور 70 خب دچار مشکل میشیم و خطا می خوره و متوقف میشه --> پس اینجا دیگه ETL متوقف میشه پس میخوایم با این کار میخوایم جلوی توقف ETL رو بگیریم
چرا توقف ETL و به خطا خوردنش اینقدر برامون مهمه؟ ETL ما افلاین است و معمولا شبا اینو انجام میدیم مثلا 2 شب در این ساعت ممکنه کسی که داره ETL رو مانیتور میکنه اصلا نبوده باشه و خطا می خوره در این حالت اگه به روز نشه چه اتفاقی می افته فرداش؟ تبعاتش:
1- پروسه های ETL معمولا پروسه های سنگینی هستن --> بدیش اینه که روی سورس لود میذاره و قطعا توی انبار داده هم داره لود میذاره --> لود روی سورس: مثلا یه حالتش اینه که میگن اجازه نمیدیم دیتا رو ساعت 8 صب بخونی که در این حالت دیتا دیگه به روز نمی تونه بشه یا ممکنه باز باشه و دیتا اونجا گم بشه - یه چیز دیگه هم هست اینه که این ETL که شب طول کشیده سه ساعت توی روز شده 4 ساعت که طبیعی است چون دیتابیس شلوغ است و این اتفاق ممکنه بیوفته

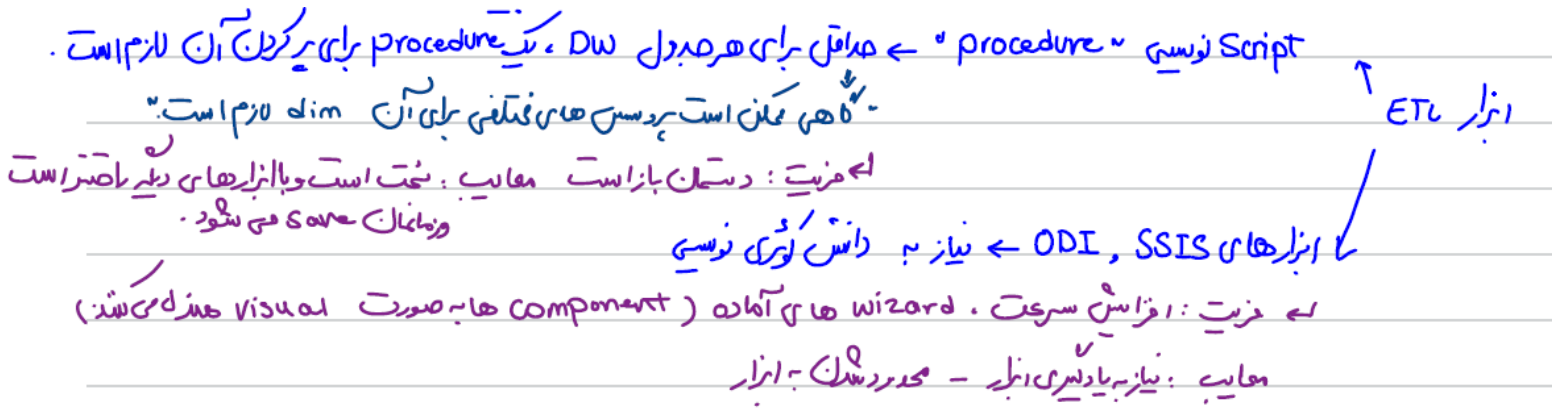
ایجاد مستند ETL

- لازم است نحوه بدست آوردن فیلدهای هر جدول انبارداده بر اساس داده های موجود در منابع داده ای، در این مستند به صورت دقیق مشخص شده باشد.
  - مثلاً فیلد نام مشتری از جدول customer بدست می آید
  - فیلد نوع مشتری، بوسیله ارتباط بین جدول customer و custype روی فیلد typ، در فیلد typedesc بدست می آید.
- این مستند مپ بین جداول سورس و جداول انبارداده را مشخص می کند.
- این مستند توسط تیم پیاده ساز ETL مورد استفاده قرار میگیرد.

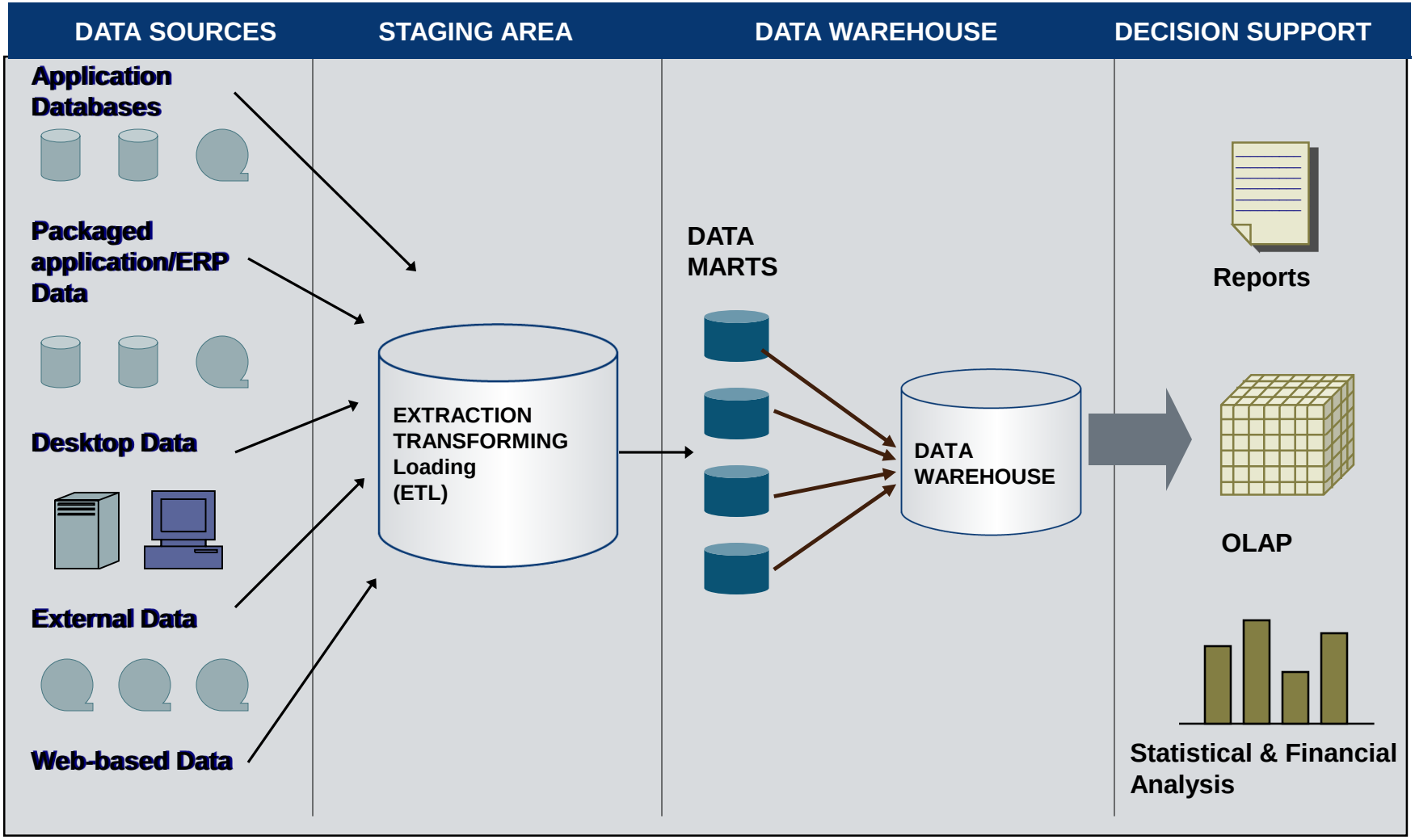
انتخاب ابزار نوشتن ETL

- نوشتن اسکریپت
  - استفاده از پراسیجر
  - مزیت: عدم محدودیت به ابزارها
  - عیب: حجم زیاد پیاده سازی
- استفاده از ابزارهای ETL ← این ابزارها خلاصه اش اینه که ما باید اول قبل از اینکه از اون استفاده بکنیم اصلا بدونیم این ابزار چیه و مفهومشو بلد باشیم
  - نظیر ORACLE ODI, SSIS
  - مزیت: پیاده سازی سریعتر
  - عیب: محدود شدن به ابزار
  - عیب: نیاز با یادگیری ابزار
- استفاده از هر دو روش فوق

نکته: حداقل برای هر جدول داخل انبار داده یک رویه ی مستقل باید داشته باشیم ینی نمی خوایم در داخل یک پروسیجر چندتا جدول انبار داده رو به روز بکنیم، اینا باید جدا باشن در واقع مثلا اگر 10 تا دایمنشن داریم باید حداقل 10 تا پروسیجر داشته باشیم که این 10 تا جدول رو پر بکنه
چرا میگیم حداقل؟ گاهی اوقات ممکنه پروسه های متفاوتی داشته باشیم و اینا همشون انجام میشن تا این جدول ها به روز بشه مثلا هم اینسرت بشه و هم اپدیت بشه تا اون دایمنشنه اون چیزی که میخوایم در بیاد



BI Architecture



نکته: STAGING AREA یک دیتابیس است و ETL نیست

بک آپ کی بدردمون می خوره؟ وقت هایی که به هر دلیلی انبار داده به مشکل می خوره و میخوایم دوباره فکت ها رو پر کنیم

STAGING AREA

- در این قسمت می توان پردازشهای مقدماتی را در صورت لزوم انجام داد.
  - مثل حذف رکوردهای غیرضروری
  - حذف داده های تکراری
  - نرمال کردن یا هر گونه تغییر مقدار
- در محیطهای با حجم داده بزرگ، معمولاً کپی داده سورس در این محل ذخیره می شود.
  - این کار جهت عدم ایجاد بار روی پایگاه داده های سورس و یا تبدیل داده به فرمت پایگاه داده مورد نظر انجام می گیرد. +
  - می توان جداول با حجم پایین و متوسط را حذف و مجدداً از داده های سورس پر کرد.
    - مانند جدول اطلاعات مشتری، جدول شعب و...
  - جداول سنگین را می توان به صورت افزایشی از منابع سورس به این محل انتقال داد. **
    - مانند تراکنش های سپرده

مثل حذف رکوردهای غیرضروری:
مثلا توی اتوماسیون اداره دانشگاه می خوایم اطلاعات کاربران رو بخونیم بعد رفتیم نگاه کردیم داخلش پیمانکاران هم هست که اینا کاربران دانشگاه نیستن و تکرار هم نیست و قرار هم نیست توی انبار داده بیاریم
مثلا اطلاعاتی هستن که محرمانگی دارن و این رکوردهایی که محرمانگی دارن رو حذف میکنیم

نرمال کردن یا هر گونه تغییر مقدار: مثلا زن و مرد میخوایم 1 و 0 باشه یا ...

+: واکشی اطلاعات از سورس به STAGING AREA معمولا بیشتر طول میکشه در مقایسه با پر کردن انبار داده از STAGING AREA --> یک مولفه خوبی اینجا تعیین کننده است که اون هم این است که STAGING AREA و انبار داده می تونن روی یک سرور باشن ولی سورس و STAGING AREA روی یک سرور نمی تونن باشن چون اصلا سورس ربطی به ما نداره

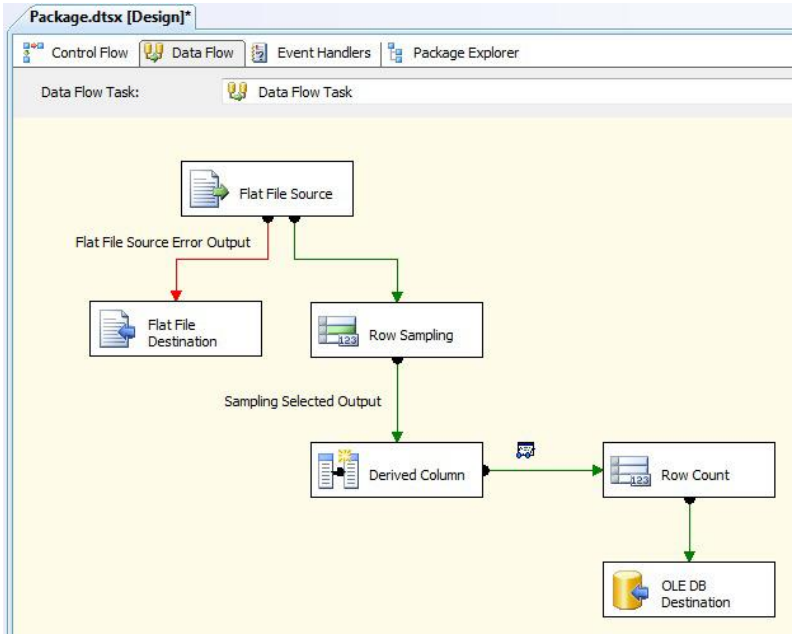
نکته: ETL ما مثلا می خواد ساعت 2 شب ران بشه باید چه اتفاقی بیوفته ؟ اول از سورس یک ETL باید اجرا بشه تا STAGING AREA را پر بکنه و بعدش از STAGING AREA به انبار داده

**: ما اینجا به صورت روتین یک max SA بگیریم و یک max سورس هم باید داشته باشیم و فاصله بین max SA و max سورس رو باید بیایم پر کنیم --> مثلا جدول SA امروز صب که 10ام است و میخواست ETL انجام بشه باید تا 9 ام به روز میشد (چون max سورس که امروز شده 10ام ولی امروز رو ازش کم می کنیم و میشه 9 ام) و الان max SA رو نگاه کردیم و دیدیم 7 ام است ینی تا 7ام به روز شده و الان باید روزای 8 ام و 9 ام پر بشه چون دو روز عقب است

نوشتن ETL این ETL مربوط به سمت انبار داده است

بهتر است: ینی بالای 90 درصد مواقع بهتره اینطوری باهاش برخورد بکنیم

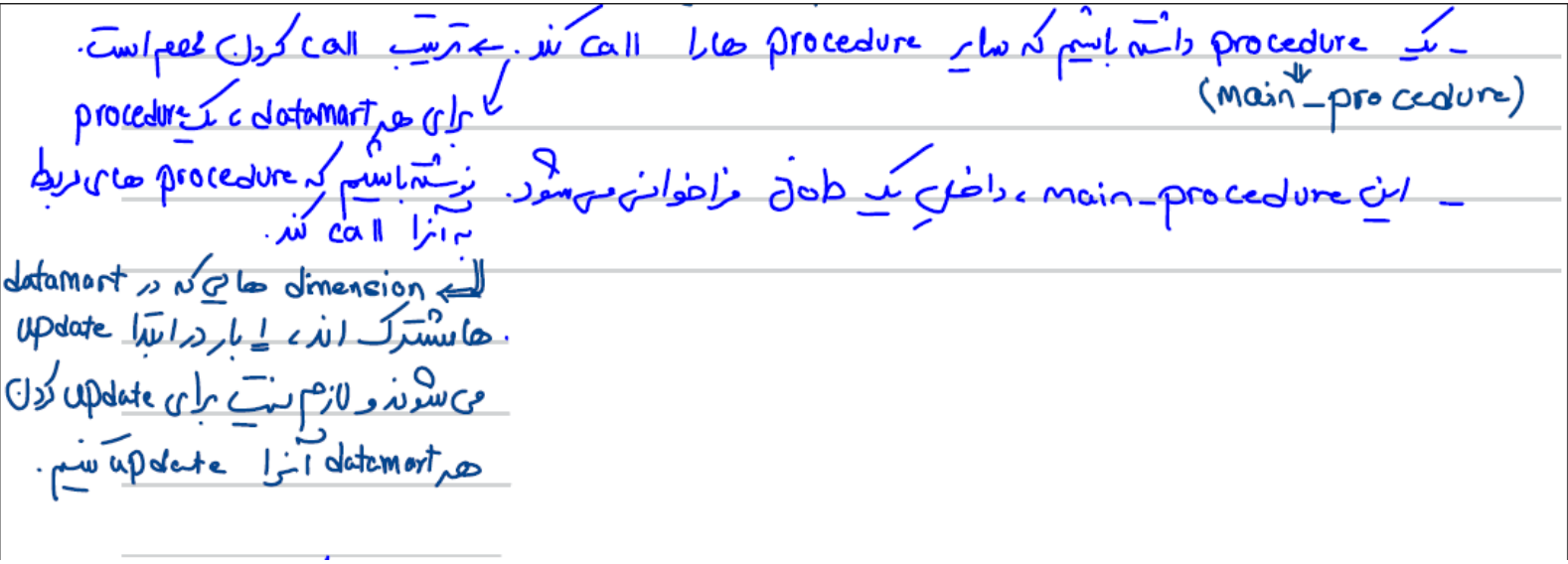
- بهتر است برای هر جدول در انبارداده یک پراسیجر داشته باشیم.
- لازم است ابتدا بعدهای مورد استفاده در یک فکت بروزرسانی شوند و سپس فکت مربوطه بروزرسانی گردد. **
- بهتر است یک پراسیجر اصلی داشته باشیم که سایر پراسیجرها در آن فراخوانی شوند و ترتیب آنها نیز رعایت شود. +
  - فراخوانی این پراسیجر در یک JOB



SA مرتب شده سورس است ولی ساختارش عوض نشده
SA هم حتما باید نرمال باشه

**: چرا؟ بخاطر اون کلیدهاست --> ما رکوردی که میخوایم توی فکت بزنیم کلیدش رو می خوایم از دایمنشن

+:
برای هر دیتامارتی باید از اینا داشته باشیم مثلا اگر دو تا دیتامارت داریم باید دوتا از این پروسیجرها داشته باشیم --> ینی دوتا پروسیجر اصلی



نوشتن ETL (ادامه)

- بهتر است رکورهای حذف شده در سورس، در جداول بعد انبارداده باقی بمانند. 1
- لازم است فکت های از نوع Transaction Fact Table و Periodic Snapshot Fact Table به صورت افزایشی پر شوند. 2
- لازم است پراسیجر به شکلی نوشته شود که در صورتی که در هر مرحله‌ای از اجرای آن پراسیجر، توقف ایجاد شود، با اجرای مجدد پراسیجر، همه چیز به درستی انجام شود. 3
  - بهتر است Insert در بعد یا فکت در یک مرحله انجام شود پیچیدگی کمتر است در این حالت : 4
  - و...
- لازم است برای اجرای اولیه ETL، پراسیجرهای متفاوتی داشته باشیم. 5 (FirstLoad)

1: ینی تاریخچه رو نگه داریم --> مثلا دایمنشن شعبه داشتیم و این الان حذف شده ولی خب اطلاعاتش رو می خوایم
دوتا جنبه داره: 1- فکت ها ممکنه که؟؟ 42 ویس 18 2- توی هیستوری می مونه مثلا شعبه یه زمانی بوده ولی الان داخل اینجا نیست --> اینارو داخل جدول ها نگه می داریم به دو دلیل مهم: هم اینکه 42؟؟ و هم اینکه لاگ و هیستوری گذشته داخل فکت ها و انبار داده باشه
منظورش این است که توی اون جدول سورسی که مشتری های ما idهاشون هست ممکنه یک id رو پاک بکنن (ولی این پاک کردن از سورس کار ما نیست ینی ما پاک نکردیم و خود سورس پاک کرده) ولی توی دایمنشن مشتری هنوز وجود داره پس میایم توی فکت مثلا روزانه بازم هر روز براش رکورد ثبت میکنیم
2: ینی داره میگه فکت تراکنشی رو حذف نکن و هر روز پرش کن چرا اینو میگه؟ چون تراکنشی از دو سال پیش تا الان پر شده پس اصلا درست نیست بخوایم پاکش کنیم و دوباره بزنیمش و فقط روز جدید که میاد اون رو میایم اینسرت میکنیم --> مثلا برای تراکنشی که می گه افزایشی ینی هر روز که یک تراکنش جدید میاد بیا بهش اضافه کن
و برای فکت اسنپ شات --> مثلا امروز 17 ام است و دیتابیس ما باید تا 16 ام به روز بشه و ساعت 2 صبح ETL اجرا شد و فکت روزانه و تراکنشی تا 15 ام به روز شدن و ETL باید یه جوری نوشته شده باشه که فقط بره دیتای 16 ام توش بریزه و فکت روزانه هم همینه ینی ما الان 15 ام داریم فقط کافیه بیایم 16 ام هم بهش اضافه کنیم
3: به چه دلیلی ممکنه متوقف بشه؟ 1- دادمون جوریه که به خطا خوردیم و این نوع خطا هندل نشده تا حالا مثلا می خواد اینسرت بکنه توی جدولی که سایزش 10 رقمی است ولی این سایزش 12 تاست در این حالت خطا میده 2- سرور هاردش پر شده 3- سیستم کرش بکنه 4- سیستم عامل کرش بکنه یا سیستم عامل دچار مشکل بشه 5- ممکنه وسطش برق قطع بشه 6- ممکنه خود سرویس دیتابیس کرش بکنه و...

4: مثال:

truncate tmpdimdep

insert dimdep into tmpdimdep

truncate dimdep

insert into dimdep ⇒ * برای اینکه اتمیک بودن شروع شود اگر در خط ۴ crash کند

۴ inner join ۲ و بتوان بدون مشکل دوباره اجرا کرد باید اول کار حذف کنیم که

full join outer tmpdim temp اگر جدول dimdep خالی بود و temp پر بود از اجرای

دوباره پروسیجر جلوگیری کنیم

ادامه اونور --> چرا truncate می کنیم؟ چون یک دیتا ریختیم و اشتباه بوده و باید پاک کنیم

نکته: اینکه بیایم tmp پر بکنیم ینی مثلا اینجا الان 10 روز دیتا داخلشه و 10 روز رو یکجا بریزیم توی فکت، مزیتی نسبت به اینکه بخواد یه روز یه روز زده بشه نداره ولی اینکه یه روز یه روز زده بشه مزیت داره نسبت به اون قبلی --> مزیتی که داره اینه که: مثلا اگر بخوایم 30 روز رو اپدیت کنیم و اگر سر روز 15ام کرش کرد به هر دلیلی کرش کرد 15 روز رو داریم - بحث مموری است مثلا اگر 100 روز عقب باشیم و یک فکت 100 روزه خودش حجم داره و این tmp هم میخواد این 100 روز رو توی فکت بریزه و شاید اینقدر حجم نداشته باشه ولی یه روزه که می ریزیم توی فکت این tmp همیشه فقط یه روز می خواد توی خودش نگه داره و سربار 100 روزه رو ایجاد نکردیم توی اینجا در این حالت
اینجا بحث اینسرت داخل فکت شدن است --> وقتی بلاک های ما کوچیک تر باشه و عملیات کمتر داشته باشیم --> فضای tmp های دیتابیسی و جدول های خودمون که داریم جنریت میکنیم خیلی قابل کنترل تر است - این فضای موقتی بهتره خیلی سایز بزرگی براش نگیریم
نکته: چرا فقط یکبار اینسرت بکنیم داخل فکت؟ این کار پیچیدگی رو کمتر میکنه
نکته: کسی که مسئول و مدیر دیتابیس انبار داده است چی رو باید هر روز چک کنه؟
ETL شب قبل درست اجرا شده یا نه - حجم دیتابیسی که برای فردا میخواد اجرا بشه منطقی است و جای خالی داره یا نه - بک آپ داده ها درست گرفته شده باشه - رم انبار داده که الان داره اشغال میشه الان در چه وضعیتی است - چک کردن یکسری لاگ ها که حجم دیتایی که داره ذخیره میشه و حجم دیتایی که از سورس اومده درست انجام شدن اینا یا نه - دیتای کامل اومده باشه --> دغدغه اش این نیست که رکوردی که داخل دایمنشن مثلا مشتری خورده درسته یا غلطه
5: می خوایم بیایم اولین بار فکت و دایمنشن رو پر بکنیم و بهتره که روال ETL رو اینجا جدا بکنیم وقتی ETL که برای اولین بار صرفا داره اجرا میشه رو بخوایم بنویسیم منطقا باید فکتها رو truncate بکنیم --> معمولا پروسه های ایجاد جداست
چرا truncate بکنیم ؟ ما رفتیم ETL نوشتیم و 4 روز دیتا ریخته شده و رفتن چک کردن و دیدن اشتباهه --> اینا معمولا پروسه هایی است که ما توی job نمی ذاریم چون یه بار قراره اجرا بشن برای این که این روال اول کار انجام بشه --> توی یک ETL دیگه ای میایم این کارو انجام میدیم و میذاریمش بره جلو تا زمانی که لازمه دیتا رو پر بکنه و این ETL رو می ذاریم کنار و دستی هم اینو انجام دادیم و بعدش می ریم سراغ ETL روزانه ای که داریم

نکته: جملاتی که اولش "بهتر داره" چیزایی که حتما باید توی پروژه داشته باشیم

نوشتن ETL (ادامه)

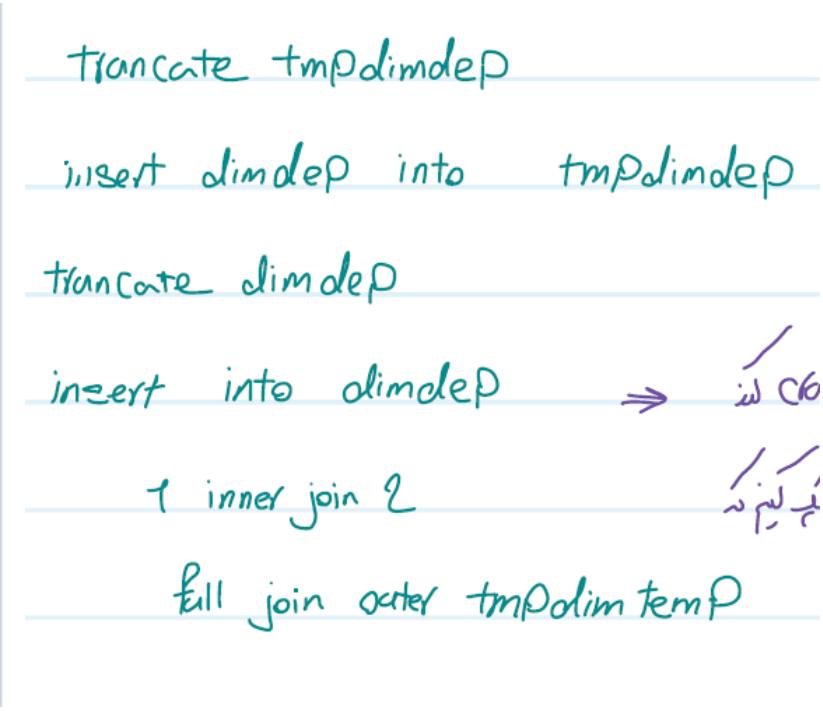
- بهتر است برای بخش‌های مختلف هر پراسیجر لاگ ثبت شود
  - شامل لحظه انجام عملیات
  - توضیحی در مورد کاری که انجام شده است
  - نام جدولی که تغییر کرده است
  - و.....

خوبه که یک جدول لاگ داشته باشیم

- بهتر است از * Select در ETL استفاده نشود 1
- بهتر است از دستور Update استفاده نشود (کار اضافه)

1: چرا؟ به خطا می خوریم و ETL گیر میکنه --> هم توی مرحله خوندن از سورس به SA اگر همچین چیزی داشته باشیم مثلا جدول ما 4 تا فیلد داره و توی سورس بوده 5 تا در این حالت به خطا می خوره
توی ETL دوم هم ممکنه این اتفاق بیوفته
کلا استفاده از select * توی هیچ جایی از ETL کار جالبی نیست که بخوایم استفاده بکنیم
مثلا توی جدول سورسی که ما داریم select * می زنیم که بریزیم توی جدول جدید توی SA و اگر توی سورس یک فیلد اضافه کنیم و اصلا هم در جریان این کار نباشیم در این حالت خطا می ده

مثال لاگ:
مثلا توی مثال زیر 4 تا استپ داریم پس توی جدول لاگ 4 تا رکورد میخوره و توی رکورد اول می گه truncate شده ینی توضیحش اینه که میگه tmpdimdep رو اومدیم truncate کردیم در این لحظه ینی در چه لحظه ای شروع شده و در چه لحظه ای تموم شده --> چجوری به لحظه های شروع و پایان برسیم؟ دوتا متغییر می گیریم و تایم سیستم رو قبلش و بعدش در اوردیم و توی این جدول می زنیم و یه اختلاف هم می گیریم از این دوتا که بفهمیم چقدر طول کشیده
رکورد دوم نشان دهنده اینسرت شدن توی جدول است
رکورد سوم: truncate رو نشون میده
رکورد چهارم: اینسرت رو نشون میده



نوشتن ETL (ادامه)

- لازم است با توجه به حجم داده و نحوه استفاده، فکتهای تاریخچه ای پارتیشن بندی داشته باشند

ایندکس ینی انگار یک درخت واره برای داده ها ایجاد میکنیم

- لازم است با توجه به نحوه بکارگیری و واکشی داده ها، ایندکسهای مناسب روی فیلدهای انبارداده وجود داشته باشد (کار اضافه)
  - Bitmap
  - Normal
  - Unique
  - و...

نکته: زمانی که از ایندکس استفاده می کنیم سرچ سریعتر میشه و اینسرت کندتر انجام میشه و اگر نداشته باشیم ایندکس اینسرت تندتره و واکشی اطلاعات سخت تره

نکته: وقتی ایندکس بندی استفاده میکنیم به غیر از داده جدول، یک ساختار داده دیگه هم در کنارش داره نگه داشته میشه که بعضا می تونه هم لول اون دادمون حجم بگیره مثلا دیتامون 500 گیگابایت است و ایندکس های این جدول هم 500 گیگابایت حجم گرفته خودش (گاها اینطوریه) --> پس ایندکس حجم داره و ایندکس روی چیزی گذاشته میشه که توی سرچ خیلی استفاده میشه و اگر توی سرچ خیلی استفاده میکنیم ایندکس می ذاریم براش مثلا نوع مشتری خیلی داره استفاده میشه پس روی این ایندکس می ذاریم

پارتیشن روی جدول معنا داره، روی پایگاه داده معنایی نداره --> دیتای یک جدول روی فایل ذخیره میشه و داریم با استفاده از پارتیشن میگیم داده هایی که توی این جدول ریخته میشه رو همه رو توی یک فایل نریز
نکته: وقتی همش توی یک فایل باشه با زمانی که همش توی یک فایل نباشه فرقش در این است که پرفرمنس تحت تاثیر قرار میگیره ینی داده های یک جدول به جای اینکه روی یک فایل بریزم روی 4 تا فایل می ریزیم --> حالا این که روی 4 تا فایل باشه فرقش با یک فایل چی ؟ اگر هدفمند اومدیم داده ها رو پارتیشن بندی کردیم در این حالت بهتر است مثلا 4 سال داده داشتیم و 4 تا پارتیشن بندی هم انجام دادیم ینی سال 99 یه جا و 1400 یه جا و 1401 یه جا و 1402 یه جا حالا در چه صورت پرفرمنس بهتره؟ مثلا میخوایم با داده های 1402 کار بکنیم و کاری به داده های 99 اینا نداریم دیگه
نکته: اگر کوئری هایی که می زنیم اردرشون ماهانه باشه --> پارتیشن هم ماهانه می ذاریم و اگر کوئری روزانه باشه پارتیشن رو روزانه می ذاریم --> پس واسه چه تیپ کوئری در نظر گرفته میشه مهمه
نکته: پارتیشن ها توی جدول های سورس اردرش کمتر است معمولا
نکته: پارتیشن بندی توی بک آپ گرفتن هم بهمون کمک میکنه

نکته: ایا برای ستون میشه پارتیشن گذاشت؟ معنایی نداره
نکته: ایندکس روی ستون گذاشته میشه و روی جدول معنایی نداره که بگیم برای این جدول داریم ایندکس می ذاریم

نوشتن ETL (ادامه)

- در شرایطی که زیرساخت مناسبی در اختیار باشد می توان از Parallel Hint در اوراکل استفاده کرد
  - به منظور افزایش سرعت اجرا
  - همیشه این Hint مناسب نیست (کار اضافه)

(parallel(8+ = ینی این کد رو برو توی 8 تا کور cpu اجراش بکن و این با شناخت از داده و سرور باید انتخاب بشه --> این توی کد اسلاید سومی بود
گاها hint هایی که نوشته میشه ممکنه روی پرفرمنس کوئری اثر بدتر بذاره مثلا برای مد گرفتن اصلا کمکی نمی کنه چون نمی تونه موازی پیش بره برای این

نکات ETL

- **حذف اطلاعات از جداول حجیم با دستور delete زمانبر و مشکل ساز خواهد بود.** 1
- **برای truncate نمودن جداول حجیم ابتدا ایندکس های آنرا حذف و سپس جدول را truncate کنید.** 2
- **در صورت وجود ایندکس در جدول، truncateجدول در زمان بیشتری انجام خواهد شد.**
- **به خصوص در شرایطی که حجم اطلاعات جدول زیاد باشد انجام این کار با wait ها و مشکلاتی همراه خواهد بود.**

1: چرا؟ چون ریکاوری توش فعاله و داره این امکان رو برای ما فراهم میکنه که اگر اشتباه شد بتونیم دوباره ریکاوری کنیم داده ها و انگار داره اینو حفظ میکنه برای خودش پس کندتره
کی دیلیت استفاده بکنیم و کی truncate؟ جایی که اطمینان خاطر داریم که نمی خوایم این دیتا رو دیگه در این حالت از دستور truncate استفاده میکنیم
2: چرا؟ وقتی روی جدول ایندکس وجود داره و داریم truncate میکنیم در این حالت کار کند انجام میشه و دیتابیس گیر میکنه --> پس زمان بر است پس اول ایندکس رو دراپ می کنیم و بعد می ریم سراغ truncate کردن

نکات ETL

- Drop کردن جداول می‌تواند بسیار پرریکسی باشد و در صورت وقوع اشتباه در این کار، نداشتن نسخه پشتیبان بار کاری زیادی را به شما تحمیل کند.
- حتی در صورت وجود پشتیبان نیز بازگرداندن آن ممکن است زمانگیر باشد.
- توصیه می‌شود جداولی را که می‌خواهید drop کنید ابتدا rename و در فرصت مناسب drop کنید.

نکات ETL

*

- joinهای بزرگ در زمان ETL بهتر است به چند join کوچکتر شکسته شوند تا مشکلاتی نظیر محدودیت سایز temp را نداشته باشیم.
- در صورتی که از حلقه استفاده می‌کنید از اجرای کوئری های تکراری با نتیجه یکسان در داخل حلقه خودداری و اینگونه اسکریپت ها را به بیرون از حلقه منتقل کنید.
  - مثلا اگر تعدادی جدول را با هم join می‌کنید که نتیجه همواره ثابت است (نتیجه اجرا در بیرون از حلقه مانند درون حلقه است)، این کار را بیرون از حلقه انجام دهید. 2

:*
چرا این کار بهتره؟ تفاوت در حجم رکوردها --> وقتی که داریم همزمان سه تا جوین رو انجام میدیم حجم رکوردهایی که داره جنریت میشه و بعد میاد فیلتر میشه به مراتب بیشتر از حالتی است که ما یک و دو رو جوین کردیم و حالا یک و دو رو ساخت و بعد این حاصل رو با یک چیز دیگه میایم جوین می کنیم و اینجا حالت هاش کمتره در مقایسه با اون سه تایی

:2
مثلا دایمنشن کاستومر رو با چند دایمنشن دیگه جوین کردیم و یک جدول x شده و این جدول x رو هم داریم با یک تراکنشی جوین میکنیم و بعد می ریزیم توی فکتی --> میگه این x که می سازی توی لوپ نساز چون این x ربطی به لوپ نداره این x رو بیا بیرون از لوپ بساز حالا این x که اماده شده بیا داخل لوپ از این x استفاده بکن
چون x ربطی به لوپ نداره اگه بیرون از لوپ نیاریمش باعث میشه یه هزینه اضافی بخوایم پرداخت کنیم

نکات ETL

- **تاریخ شروع و پایان را در ای تی ال هارد کد نکنید.**
- **در ابتدای ای تی ال first load بهتر است sequence را به incremental نیز می‌توانیم در ابتدای ای تی ال مقدار اولیه و آنرا به max(مقدار استفاده شده) + interval ریست کنیم.**

نکات ETL

این مثال ها روی اوراکل است:

- **در صورتی که تابعی بر روی فیلدی اعمال می‌شود، ایندکس ها(هر نوع ایندکسی؟ نمره اضافه) و پارتیشن ها در خروجی آن استفاده نمی‌شوند و کارآیی کوئری تا حد بسیار زیادی کاهش می‌یابد. مثلا اگر جدولی دارای partition روی فیلد effdate یا ایندکس روی این فیلد باشد فراخوانی‌های زیر از ایندکس و پارتیشن استفاده نمی‌کند:**

* - Where trunc(effdate) = to_date(20140101,'yyyymmdd') اینجا پارتیشن بندی نداریم

- Where to_char(effdate,'yyyymmdd','nls_calendar=persian') = '13930203'

تابع to_char : تبدیل میلادی به شمسی

- **بهتر است شرط به این شکل باشد:**

- Where effdate>= to_date(20140101,'yyyymmdd') and effdate< to_date(20140102,'yyyymmdd')

اینجا روی effdate پارتیشن بندی داریم و خود این پارتیشن ها هم مشخصه که توی چه بازه ای هستن و روی این effdate اومدیم where زدیم پس بعدش می ره سراغ اون پارتیشن هایی که بهش الان مربوطه

- Where effdate>= to_date(13930203','yyyymmdd' ,'nls_calendar=persian') and effdate< to_date(13930204,'yyyymmdd' ,'nls_calendar=persian')

مثلا یک فیلد داریم به اسم کاستومر id و بعد این کاستومر id رو میایم یک فانکشن می ذاریم پشتش مثلا فانکشن substring اول رو بهم بده --> حالا این substring کاستومر id با کاستومر id فرق میکنه --> اگر روی کاستومر id ایندکس بوده باشه روی substring کاستومر id دیگه ایندکس نیست
روی بحث پارتیشن هم به همین صورت است مثلا یک فیلد داشتیم دیت تایم که این دیت تایم

: *
مثلا توی این مثال روی فیلد effdate پارتیشن بندی کردیم و وقتی می زنیم trunc(effdate) مساوی یه چیزی اون وقت این پارتیشن رو از دست میدیم

نکات ETL

- **حتما از اسکریپت‌های خود بک آپ تهیه کنید.**
- **در زمان پیاده سازی، از insert، update، delete و اعمال اینگونه دستورات در سایر اسکیماها پرهیز کنید. مثلا از اسکیمای A اطلاعات اسکیمای B را تغییر ندهید. بهتر است اسکیمای B رویه‌ای داشته باشد و اسکیمای A آنرا فراخوانی کند. (اسکیما را در SQL SERVER همان دیتابیس در نظر بگیرید)**
- **می توان برای اطلاعات پایه ای از جداولی جدا از فکتها و ابعاد استفاده کرد. مثل اینکه ۰ یعنی مرد و ۱ یعنی زن**

مراحل انجام پروژه

- شناخت داده سورس و نیازمندی‌های داده‌ای حوزه مورد نظر
- طراحی
  - بر اساس مدل STAR
  - انواع فکت
  - SCD در بعدها
- مستند ای تی ال
  - برای هر بعد و یا فکت
- ایجاد انبارداده
  - ایجاد جداول در یک دیتابیس
- پیاده‌سازی ای تی ال
  - نوشتن پراسیجرها
- تست
- زمان‌بندی اجرا