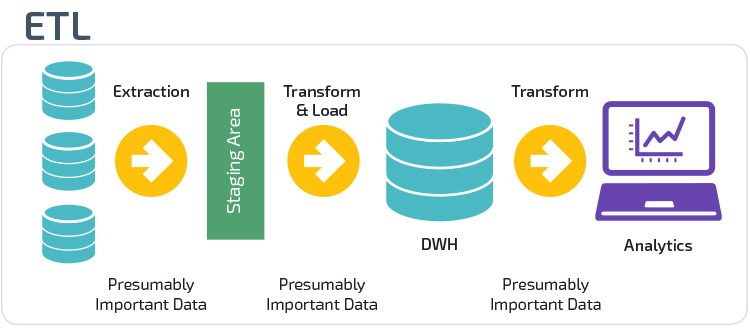
Discussing Databases, Data Warehouses and ETL Processes (Extract, Transform and Load)



Please refer to the glossary for clarification of terms.  
  
Many companies have more than one information system that captures operational data. While larger firms may have only 2 or 3 major information systems to run operations (CRM, SCM, ERP) and it therefore may be easier to merge data for reporting, smaller firms may still run with many smaller systems that each digitize and the transactions of a department or process. While the systems may keep the operations humming along, when it comes time to build reports, perform exploratory analysis, roll-up data to different levels of summarization, integrate performance data with different divisions, regions, and departments, chaos ensues. The problem is that there is lots of data but the smaller systems often do not have reporting or the needed reports. Very often operational systems created do not allow the purchaser of the system to create reports (hey consulting is an important revenue stream for the software company).  
  
Not wanting to pay exorbitant consulting fees to build reporting systems, many firms perform reporting by extracting data into Excel, and making employees suffer in copy/paste hell.

This problem of incorrect reports, and over use of Excel for reporting still exists due to the majority of corporate employees not knowing what a database or SQL is, and worse not valuing IT departments. While PowerBI can solve this problem for smaller firms, when there is a lot of data, it needs to be managed. Data Warehouses (DW) are the answer. Data warehouses hold the corporate data (but not accounting/finance spreadsheets or contracts, etc.) but the data needs to be cleaned and organized before it is checked into the garage (DW). The purpose of this document is to discuss how the data gets into the data warehouse.  
  
*Some reasons the data needs to be pulled together and moved into a data warehouse:*  
a) to move older data out of operational systems to keep operational systems and databases running fast   
b) to merge data from different sources to enable company-wide perspective (ie roll-up from different divisions to corporate)   
c) to merge data from different sources (internal/external data) to enable richer reporting and analysis sessions (e.g. merging in DMV or weather data),   
d) to provide a comprehensive history of business operations,   
e) to ensure reporting and analytics processes do not slow down operations,   
f) to aggregate data at different levels of granularity to allow future analytics  
g) to save aggregated data at different levels of granularity to allow easier reporting and data integration.

Building ETL with batch processing, following ETL best practices, involves:

1. **Reference data** - create a library of data types, data rules and datasets that define the set of permissible values your data may contain. This is important when merging data from different datasets, people and regions. For example, in a country data field, you can define the list of country codes allowed (e.g. USA vs. US). You can also use a currency conversion rate as in converting euros or yen to USD$. This step is very important when data comes from personal systems or many small departmental or homegrown workgroup systems.
2. **Have a data model in your data warehouse - data will be moved into your data warehouse, so you need to have a *plan of the tables and relationships that will house the data (called a data model*). You can research Kimball’s star approach for more detail). You also need mapping rules, for example to map a column from an operational database to a column in the DW. If you have many operational systems feeding the DW then you need many rules. The goal is to reduce the number of tables in the DW (good luck). For example The Adventureworks DB is 80 tables and the DW version is about 15 tables. When you reduce the # of tables the queries run faster.  
     
   *Make a plan to copy data from operational DBs to the DW. Also make a mapping document that maps source data and their datatypes to the columns in the DW*  
     
   You also need a plan to collapse data *(or refresh denormalized tables)* from different tables into a flattened, denormalized table (as demonstrated with AWProductsFlattened (that has 3 tables of mostly dimensional data combined), and ResellerswithGeography (combining geographical data e.g. city, state, region, country in with the reseller dimension data). If you reduce the number of tables and inner joins needed the report and dashboard design is faster and they ‘come out the printer’ faster**
3. **Extract from data sources** - the basis for the success of subsequent ETL steps is *to extract the right data columns correctly*. Most ETL systems combine data from multiple source systems, each with its own data organization and format - including relational databases, non-relational databases, XML, JSON, CSV files, excel files, access databases, etc. Successful extraction converts data into a single format for each column even if the data comes from different types of systems.   
     
   The data is typically extracted from operational databases, where the data represents rows of transaction data. The data is often in many, many tables due to the adherence to database theory (3NF) where you do not have repeating values in any tables.  
     
   You also need to decide *how often the data is extracted*, daily, hourly? Relatedly you need to decide *how often the reports and dashboards are updated*, daily, weekly, hourly? Be careful the more often you want updates, the more expensive the technology will cost (in RAM processing power).  
     
   We used INSERT INTO SELECT \* FROM to copy data from one database to another. This can be automated with stored procedures. You can also use SELECT \* INTO FROM query if you want to create a new table using the data from the source table.
4. **Stage** - you will not typically load retrieved data directly into the target data warehouse*. Data should first enter a staging database (a set of tables and/or arrays)*, making it easier to roll back if errors occur.   
     
   This is similar to putting items purchased at yard sales on your driveway to inspect them closer and hose them down (washing them) before putting the item into the correct place in your garage or closet.  
     
   For example all data can be imported into SQL Server database tables in a staging database. Reports are not made from this staging area*. Rather the data is processed and transformed (the next steps) and verified before combining it into the DW.*
5. **Data validation** - *an automated process confirms whether data pulled from sources has the expected values* - for example, in a database of financial transactions from the past year, a date field should contain valid dates within the past 12 months. The validation engine rejects data if it fails the validation rules. You analyze rejected records, on an ongoing basis, to identify what went wrong, correct the source data, or modify extraction to resolve the problem in the next batches.  
     
   Compare the automated processes that can be envisioned, designed in SQL queries (or similar special purpose ETL technologies that may require no SQL) with …  
     
   manual data validation in Excel. Which process do you want to work with in your future career?
6. **Transform data** - *removing extraneous (pruning the columns) or erroneous data (cleaning), applying business rules to categorize records, checking data integrity (ensuring that the data was not corrupted in source, or corrupted by ETL, and that no data was dropped in previous stages), and creating aggregates as necessary.*   
     
   You also may *denormalize data* to simplify the data model into a star schema (ie collapse hierarchies into one denormalized table. For example rather than have separate tables for cities, states, counties, rather you place all of this data into a dimension table for example named dimGeography. The flattened table has lots of duplicate values but overall the system will run much faster, and development time is reduced.  
     
   *Data is aggregated and stored at different levels in different tables*, if you must analyze revenue, you can summarize the dollar amount of invoices into a daily or monthly total. You will need to program and test a series of rules or functions that can achieve the required transformations, and run them on the extracted data.

*Many columns of derived data (stored in both dimension and fact tables) and measures are created in the transformation process. These columns and measures are metrics (analytics*). It may take 30 new columns and measures to assess business profits and performance for one business process. Very often the metrics are so complicated that you need to calculate intermediate values which for example become the numerator and denominator in another calculation.   
  
In SQL we used DATEPART(), DATENAME(), NTILE(), CASE(), GROUP BY() functions and simple column calculations (i.e., multiplying columns together and then subtracting out values from another column) to produce columns of metrics. These calculated columns become the values that are shown in the graphs and maps.  
  
a) DATEPART() was used to extract out different date and time formats such as day of week or month.  
  
b) DATENAME() was used to pull out textual days of the week and month (Monday and January, etc.)  
  
c) NTILE() was used to categorize the data rows into equally sized groups (for example quartiles). The number of records in each group are not equal, but the cut-off values used to create the groups are equally sized (ie 0-25%, 26-50%, 51-75%, 76-100%).  
  
d) CASE() was used to add columns of textual information that signify categories (such as top customer, frequent customer, new customer). CASE() was also used to discretize (aka bucket or bin) data, putting a value into a column that can be used to put records into groups to reduce the granularity of the data such as putting customers into groups by 10-year age bins (20’s, 30’s, 40’s). While NTILE() creates equal sized groups (ie terciles or quartiles), you can use CASE() to put records into groups with different sizes (ie 20-35, 35-43, 44-50, etc.)  
  
You can also use CASE to perform different calculations based on your specifications. For example if a table of data had merged data from three different subsidiaries, then the data records from each subsidiary can be calculated in a different way.   
  
We used case statements to build new columns of analytics used in slicers, and to categorize data. We also used case statements inside UPDATE SET statements to perform different functionality for different groups. There are many ways to use CASE statements, to perform different functionality.

e) GROUP BY() was used to add summary aggregated values (averages, summations and counts) from fact tables into a dimension table. For example averaging all the sales for each customer and displaying that summary data in the master data record for each customer.

f) UPDATE SET can be used   
  
Remember that data can be saved into the DW at any level of detail (granularity) you will probably save the detail operational data, but you can also s*ave summarized data as well to speed up data retrieval and merging.* For example you can compile transaction data by store level to investigate store performance.  
  
*It is a good idea to put as many of the data calculations into the automated ETL process rather than inside a pivot table or report. The ETL process is administered by IT professionals with different data quality inspection*s. There are rarely any quality controls on reports that BI analysts make (which is troubling). Try to have the calculations of ratios, aggregations, averages, and other metrics done without human intervention).

1. **Publish to data warehouse** - loading the data to the target tables. Some data warehouses overwrite existing information every time the ETL pipeline loads a new batch - this might happen daily, weekly or monthly. In other cases, *ETL can add new data without overwriting, with a timestamp indicating it is new.* You must do this carefully to prevent the data warehouse from “exploding” due to disk space and performance limitations.
2. **Back up data offsite or into the cloud. – it would be terrible to have a natural disaster destroy all data records. Hurricanes, etc. are a real phenomenon. Some firms may not allow cloud based storage (ie hospitals, banks) so the DBA would need a plan to *back up data to another site*. Also *archive old data* into a different partition on the database server to keep the reporting processing speedy.**  
     
     
   Final Note:  
     
   Analytics and reports are best run by connecting to the data warehouse. This ensures that the data is cleaned, not changing and available widely to different analysts and managers. No one is chasing versions of excel data files.

Image and some content from   
*https://panoply.io/data-warehouse-guide/3-ways-to-build-an-etl-process/*