High level goals for the Integrations with vendor to support the development of Database Data Quality

Many technical attempts to address data quality will not function unless they are part of the larger company culture driven from the top of the organization down to all operation and business levels.

Roles to consider:

From the business side of the house, we’ll need representatives to ﬁll the following roles:

* **Business sponsor.** The sponsor is the DW/BI system’s ultimate client, as well as its strongest advocate.
* **Business lead.** The business project lead is a well-respected person who is highly involved in the project, communicating with the project manager on a daily basis.
* **Business users.** Optimally, the business users are the enthusiastic fans of the DW/BI environment. You need to involve them early and often, beginning with the project scope and business requirements.

From the business or IT.

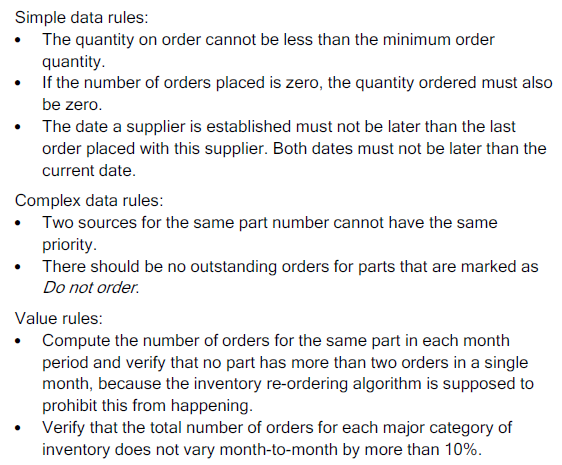
1. Data Steward – responsible for driving the organizational agreement on the definition, business rules and subject areas mastered in the data warehouse. Typically this role belongs to the subject matter expert for the business and being able to address cross-functional issues across the company is key to that success.
2. Quality Assurance Analysts who ensure the data loaded into the warehouse is accurate and complete. Identifying data errors and owning that correction thru to final resolution. Verifing the business integrity of the BI system is critical for this role. Data Quality errors are the indication of broken business processes.
3. **BI application designer/developer.** BI application resources are responsible for designing and developing the starter set of analytic templates, as well as providing ongoing BI application support.
4. **Project manager.** The project manager is a critical position. This person should be comfortable with and respected by business executives, as well as technical resources. The project manager’s communication and project management skills must be stellar.
5. **Technical architect.** The architect is responsible for the overall technical architecture. This person develops the plan that ties together the required technical functionality and helps evaluate products on the basis of the overall architecture.
6. **Data architect/modeler.** This resource likely comes from a transactional data background with heavy emphasis on normalization. This person should embrace dimensional modeling concepts and be empathetic to the require- ments of the business rather than focused strictly on saving space or reducing the ETL workload.
7. **Database administrator.** Like the data modeler, the database administrator must be willing to set aside some traditional database administration truisms, such as having only one index on a relational table.
8. **Metadata coordinator.** This person helps establish the metadata repository strategy and ensures that the appropriate metadata is collected, managed, and disseminated.
9. **ETL architect/designer.** This role is responsible for designing the ETL envi- ronment and processes.
10. **ETL developer.** Based on direction from the ETL architect/designer, the devel- oper builds and automates the processes, likely using an ETL tool.

DW/BI projects are vulnerable to scope creep largely due to a strong need to satisfy business users’ requirements. You have several options when con- fronted with changes:

Increase the scope (by adding time, resources, or budget), play the zero-sum game (by retaining the original scope by giving up something in exchange), or say “no” (without actually saying “no” by handling the change as an enhancement request).

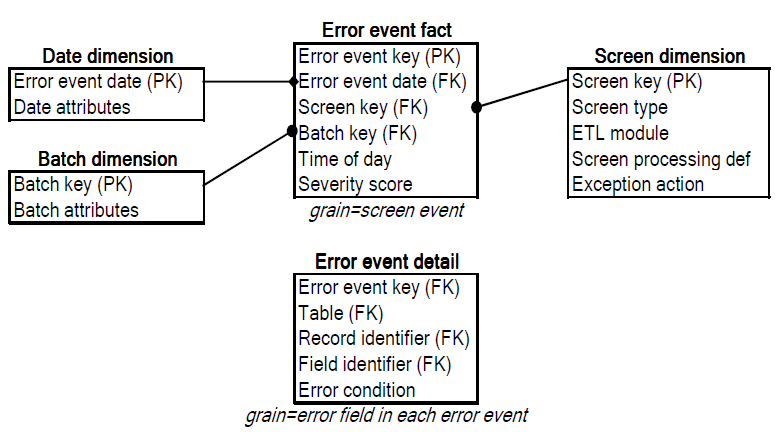
The most important thing about scope decisions is that they shouldn’t be made in an IT vacuum. The right answer depends on the situa- tion. Now is the time to leverage the partnership with the business to arrive at an answer that everyone can live with.

Rules for data that must be agreed asap, what are the rules for the data.



When errrors occur , the must be placed into the appropriate Error Event Schema.

This is a centralized dimensional schema whose purpose is to record every error event, anywhere in the data warehouse pipeline.



The dimensions of the error event fact table include the calendar date of the error, the batch job in which the error occurred, and the screen which produced the error. The calendar date is not a minute and second time stamp of the error, but rather provides a way to constrain and summarize error events by the usual attributes of the calendar, such as weekday or last day of a fiscal period. The time-of-day fact is a full relational date-time stamp that specifies precisely when the error occurred.

The error event schema includes a second *error event detail* fact table at a lower grain. Each record in this table identifies an individual field in a specific data record that participated in an error. Thus a complex structure or business rule error that triggers a single error event record in the higher level error event fact table may generate many records in this error event detail fact table. The two tables are tied together by the error event key, which is a foreign key in this lower grain table. In other words, there is a strict 1-to-many relationship between records in the parent error event fact table and the child error event detail fact table. The error event detail table identifies the table, record, field, and precise error condition, and likewise could optionally inherit the date, screen, and batch dimensions from the higher grain error event fact table. Thus a complete description of complex multi- field, multi-record errors is preserved by these tables The error event detail table could also contain a precise date-time stamp, to provide a full description of aggregate threshold error events where many records generate an error condition over a period of time.

This format is useful for calculating the time interval between error events because you can take the difference between two date-time stamps to get the number of seconds separating events.

Batch schedule date-time stamp

Actual batch starting and ending date-time stamps

Total number of records processed in batch run

Total number of screen tests performed in batch run

Total number of errors encountered in batch run

Database, processor, memory, and disk contention

Maximum error severity score in batch run

We are tracking failure to comply with know data quality rules such as :

* Value must be not null.
* Value must be not null.
* Value must be one character and from a finite fixed list.
* Value must be within a range.
* Value must fit a specific field pattern such as 29999.
* Value must either be null or greater than 5 characters of free text.
* Value must not be in a specific exclusion list.
* Value must not fail spell checker.

We also test data relationship:

* testing foreign key/primary key relationships between fields in two tables.
* A combination of fields must implement a primary key for the surrounding table.
* An inventory history part number must appear in the inventory master.
* All inventory parts must have at least one source.
* All suppliers must supply at least one part.
* A supplier may have no orders.

We can further further subdivides business rule screens into three subcategories, including simple data rules, complex data rules, and value rules.

**Simple data rules:**

* The quantity on order cannot be less than the minimum order quantity.
* If the number of orders placed is zero, the quantity ordered must also be zero.
* The date a supplier is established must not be later than the last order placed with this supplier. Both dates must not be later than the current date.

**Complex data rules:**

\*Two sources for the same part number cannot have the same priority.

\*There should be no outstanding orders for parts that are marked as

**Value rules:**

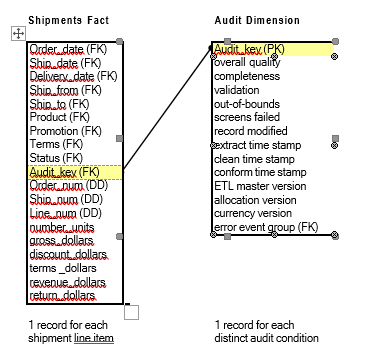
\*Compute the number of orders for the same part in each month period and verify that no part has more than two orders in a single month, because the inventory re-ordering algorithm is supposed to prohibit this from happening.

\*Verify that the total number of orders for each major category of inventory does not vary month-to-month by more than 10%.

When errors occure we can mark Bad fact table data can be tagged with the audit dimension.

# The Audit Dimension

The audit dimension is a normal dimension that is assembled by the ETL process for each fact table.



Focus on adding *business* value across the Miso enterprise

* *Dimensionally* structure the data that’s delivered to the business and adding deep value for future integration needs.
* Iteratively develop the DW/BI environment in manageable *lifecycle* increments rather than attempting a galactic Big Bang approach. This will involve breaking down work and managing vendor resources.

Diagram

Description automatically generated with medium confidence

Program/Project Planning) First Phase

1. What is Miso’s program and culture readiness to begin a DW/BI initiative. Here we obtain scope, budget and resources to perform specific tasks.
2. The second readiness factor is having a strong, compelling business motivation for tackling the DW/BI initiative lead by a project sponsor
3. The technical and resource feasibility must be identified, but data feasibility is the most crucial. Are we collecting real data in real operational source systems to support the business requirements? Data feasibility is a major concern because there is no short-term ﬁx if you’re not already collecting reasonably clean source data at the right granularity.

Business Requirements Definition) Second Phase

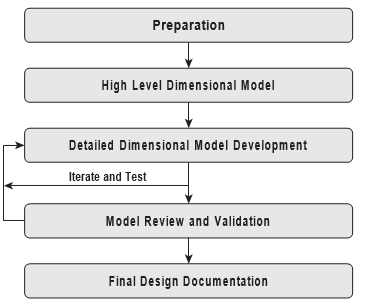
1. There’s a two-way arrow between program/project planning and the business requirements, alignment of requirements is absolutely critical.

Technical architecture design

1. Establishes the overall framework to support the integration of multiple technologies.

Dimensional Modeling

1. emanating from business requirements definition focuses on data, translating the requirements into a dimensional model. After completing the high-level model, the design team dives into the dimension tables with attribute deﬁnitions, domain values, sources, relationships, data quality concerns, and transformations. After the dimensions are identiﬁed, the fact tables are modeled. The last phase of the process involves reviewing and validating the model with interested parties. The primary goals are to create a model that meets the business requirements, verify that data is available to populate the model, and provide the ETL team with a solid starting source-to-target mapping.
2. . The process is complete when the model clearly meets the business’s requirements. A typical design requires three to four weeks for a single business process dimensional model, but the time required can vary depending on the team’s experience, the availability of detailed business requirements, the involvement of business representatives or data stewards authorized to drive to orga- nizational consensus, the complexity of the source data, and the ability to leverage existing conformed dimensions.



Identify Source and Target data

**DimOrderProfile** Dimension OrderProfile

|  |
| --- |
| **Table Name** |
| Table Type |
| Display Name |
| Description |
| Used in schemas |
| Size |

Order Profile is the “junk” dimension for miscellaneous information about order transactions Orders

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Target** | | | | | **Source** | | | | |
| Column Name | Description | Datatype | Size | Example Values | SCD  Type | Source System | Source Table | Source Field Name | Source Datatype | ETL Rules |
| OrderProfileKey | Surrogate primary key | smallint |  | 1, 2, 3... |  | Derived |  |  |  | Surrogate key |
| OrderMethod | Method used to place order (phone, fax, internet) | varchar | 8 | Phone, Fax, Internet | 1 | OEI | OrderHeader | Ord\_Meth | int | 1=Phone, 2=Fax, 3=Internet |
| OrderSource | Source of the order (reseller, direct sales) | varchar | 12 | Reseller, Direct Sales | 1 | OEI | OrderHeader | Ord\_Src | char | R=Reseller, D=Direct Sales |
| CommissionInd | Indicates whether order is commissionable or not | varchar | 14 | Commission, Non- Commission | 1 | OEI | OrderHeader | Comm\_Code | int | 0=Non-Commission, 1=Commission |

**Figure 18-3:** Sample detailed dimensional design worksheet.

##### The 34 Subsystems of ETL

With an understanding of the existing requirements, realities, and constraints, you’re ready to learn about the 34 critical subsystems that form the architec- ture for every ETL system. This chapter describes all 34 subsystems with equal emphasis. The next chapter then describes the practical steps of implementing those subsystems needed for each particular situation. Although we have adopted the industry vernacular, ETL, to describe these steps, the process really has four major components:

* **Extracting**. Gathering raw data from the source systems and usually writing it to disk in the ETL environment before any signiﬁcant restructuring of the data takes place. Subsystems 1 through 3 support the extracting process.
* **Cleaning and conforming**. Sending source data through a series of processing steps in the ETL system to improve the quality of the data received from the

source, and merging data from two or more sources to create and enforce con- formed dimensions and conformed metrics. Subsystems 4 through 8 describe the architecture required to support the cleaning and conforming processes.

* **Delivering**. Physically structuring and loading the data into the presentation

server’s target dimensional models. Subsystems 9 through 21 provide the capabilities for delivering the data to the presentation server.

* **Managing**. Managing the related systems and processes of the ETL

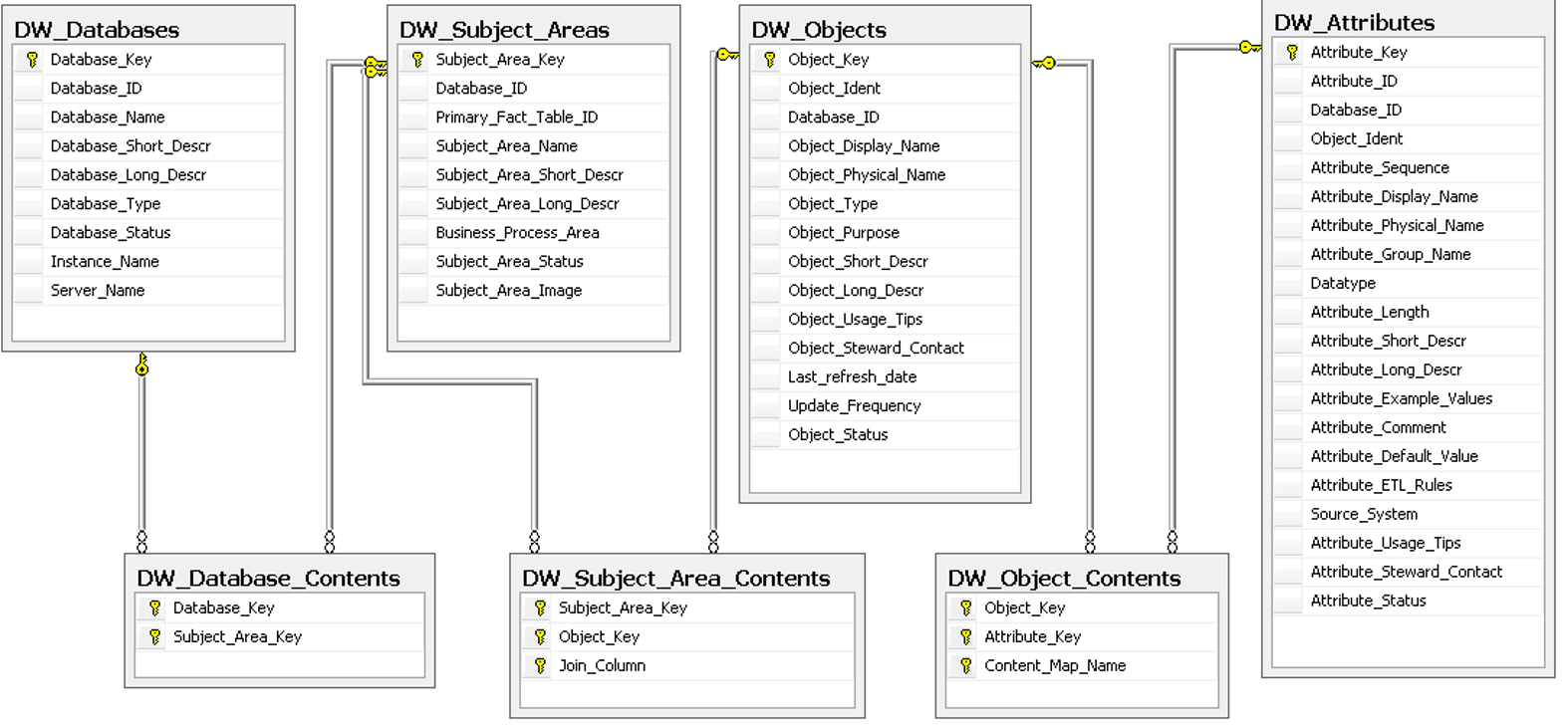
environment in a coherent manner. Subsystems 22 through 34 describe the components needed to support the ongoing management of the ETL system.

Across the entire Misorobotics landscape that includes clouds, applications, hardware, software, middleware, people and systems we are generating thousands of mission critical "TOUCH POINTS" every moment of the day. The key to our companies current and future prosperity comes down to how well we gather, retain and disseminate these Touch Points. Metadata is data about our data and the absolute key area we must concentrate on to gain the operational maturity needed to scale exponentially, create one version of the truth and make data our most critical asset.

These critical datasets and datasources include unformatted, unstructured , undocumented, non useable, duplicated and often wrongly interpreted data found in both spreadsheets, wiki websites, operational datasets and countless databases. Without the proper inventory, management and data consolidation of these critical assets and the data they produce, we will be unable to optimize and efficiently manage the huge investment in resources and systems we implement today.

This initiative seeks to build a master system of record that can manage our company’s assets by cataloging the most critical components to include cloud,applications, dataflows, processes, hardware, software , and our most valued asset the data itself. We seek to leverage this understanding of the companies metadata in order to better manage the below critical but not limited to the below components of our IT Portfolio Management System.

We need an inventory of all systems, xls, confluence, emails, sources systems of record, databases.



The subject areas consist of the below key areas but not limited to :

| * Hardware assets (Both onsite, offsite at both internal and Customer locations) * Hardware configurations of all deployed systems * Hardware locations when all assets reside * Hardware costs (purchase price, leasing fees, maintenance fees, etc.) |
| --- |
| * Cloud/Software license and cost * Cloud/Software installations and configuration management * IT lifecycle cost (E = evolving, S = stable, A = aging, O = obsolete/unsupported) |
| * System purposes, Inputs/outputs * System integration * Projects/Initiatives * Project costs (Estimate vs Actual) (Resources both internal, consultant) * Project status, Completion |

**Need for Capturing Data about DATA (The METADATA).**

We must track the Data Lineage, what happenes to that data, if it was moved from one system to the next, how it was transformed, aggregated and what was loaded into a yet to be discussed Data Warehouses for analytics purposes.

Diagram

Description automatically generated

We propose that this future system of record contain at mimimum the below Datasets/Datamodel necessary to manage Misorobotics assets. Here is the High level Metadata Map to identify most sources of data for the company and how we will store that information in one place to orchestrate our processes. We will then use this metadata mapping to perform a series of transformations to maintain the existing transactional systems as well as load into the data warehouse. First is the ETL transformation image below:

Diagram

Description automatically generated with medium confidence

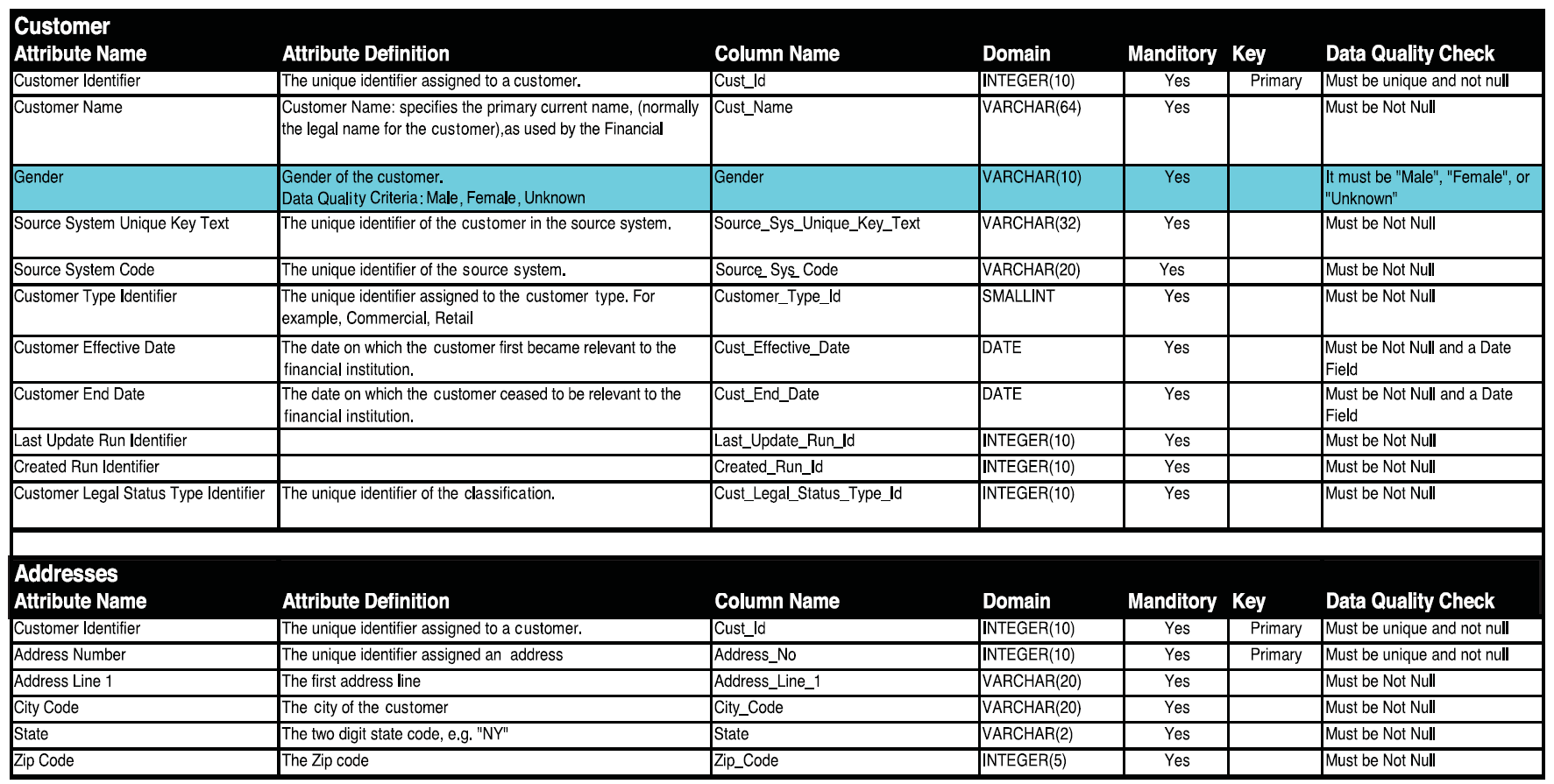
and the dependent datasets that must be populated in order to orchistrate the ETL from existence in operational systems to being aggregated into Analytics via the companies Datawarehouses.

**Here is the METAMODEL**

A picture containing text, metal, screenshot, attached

Description automatically generated

As we encounter new Source systems, we will need to add this level of detail (see xls below) into the model above.

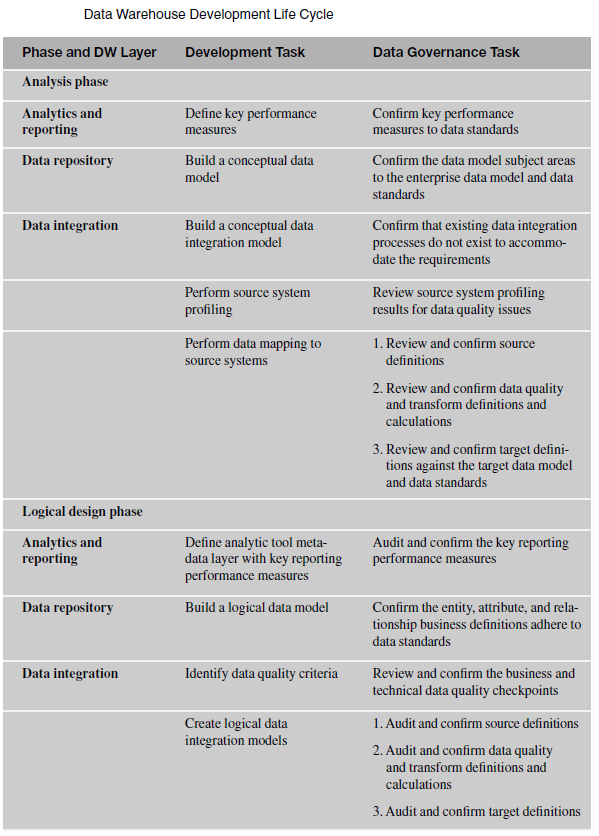


What the below models will store is this exact transformation rules:

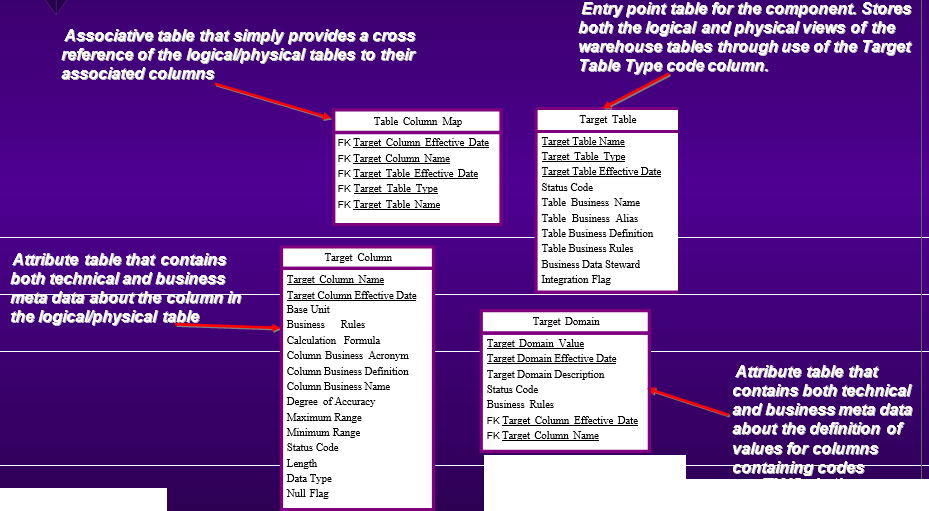
Diagram

Description automatically generated

While we consider the transformation necessary, here is the development lifecycle of a Data Warehouse



**Here is a quick overview for each highlighted area in the above METAMODEL**



These Target Entities will be destination entities in the future data warehouse.

**Target Table Name - contains either the logical or the actual physical, database name, of the warehouse table**

**Target Table Effective Date - provides a mean of distinguishing revisions made to a particular table**

**Status Code - indicates the current approval status of the warehouse table using domain values such as pending, approved or removed.**

**Table Business Name - contains the common name of the table that will be presented to users through the front-end access method of the warehouse**

**Table Business Alias - contains a common alternative name or acronym of the table (e.g., alias for profitability ratio is PR)**

**Table Business Definition - contains a detailed descriptive explanation of the business process the table plays in the enterprise allowing the end user to make an educated determination about its use**

**Table Business Rules - is used to denote any conventions or practices the table must adhere to beyond referential integrity constraints**

**Business Data Steward - lists the individual and/or groups that are responsible in the organization for defining this particular table**

**Integration Flag - is used as a quick query method to denote which tables obtain source**

**Target Column Name - contains either the logical or the actual physical, database name, of the warehouse field**

**Target Column Effective Date - provides a mean of distinguishing revisions made to a particular field**

**Base Unit - contains a description or code of the measurement unit used for entries in the field (e.g., dollars, pounds, meters)**

**Business Rules - used to denote any conventions or practices the column must follow. For example, an amount field must have a corresponding currency value**

**Calculation Formula - contains a description of the method used to determine the value stored in the field including the names of other tables/columns used**

**Business Acronym - contains a common acronym coding of the field (e.g., UOM for unit of measure)**

**Business Definition - contains a detailed descriptive explanation of the business meaning of the column in the context of the enterprise**

**Business Name - contains the common name of the column that will be shown to users through the front-end access method of the warehouse**

**Degree of Accuracy - is used to denote the required number of places after the decimal point that entries in this column are required to contain. For example, currency conversions involving the Euro monetary unit should contain six decimals of precision to accurately convert**

**Maximum Range and Minimum Range - indicate the upper and lower bounds of numeric values for the column**

**Status Code - indicates the current approval status of the column (e.g., pending, approved, removed)**

**Length, Data Type and Null Flag - provide the basic technical information required to describe the fields into a data modeling tool or to a DBMS. For example, a company name column could be described as having a length of hundred, data type of character or varchar and does not allow null values**

**Target Domain Value - code for a column. For example, the column country code has domain values that include USA for United States and CAN for Canada, etc.**

**Target Domain Effective Date - provides a mean of distinguishing revisions made to a particular code**

**Target Domain Description - contains a detailed descriptive explanation of the business meaning of the code**

**Status Code - indicates the current approval status of the code (e.g., pending, approved, removed)**

**Business Rules -used to denote any conventions or practices the code must follow**

Source Systems:

Contains physical information about the various sources of data feeding the data warehouse Information can originate from databases, file extracts, spreadsheets, Internet, and other formats Information can be used to alert data warehouse team of potential changes that will impact the DW model, ETL processes, and reports Information can be utilized by both business and technical analysts for reconciliation and auditing

Text

Description automatically generated with medium confidence

Source ID - used to uniquely identify a particular system of record (e.g., ERP, Order Management 1, Trouble Ticket, Client Extract 5)

Source Effective Date - provides a mean of distinguishing revisions made to an operational system or extract file

Source Format Type - provides a means to identify the category of the source information such as a server/database, directory/file or spreadsheet file

Source DBMS - contains the physical names of these items for cases where operational information is being extracted directly from a source database table

Source Description - contains a detailed narrative describing the particular source of information for the warehouse

Source Update Frequency - denotes the cycle at which this source is updated (e.g., daily, weekly, monthly, annually)

Status Code - indicates the current approval status of the source system or extract file (e.g., pending, approved, cancelled)

Source to Target mappings:

Contains the cross reference mappings nd semantic resolution between the source operational systems and the target physical data warehouse model Key feature of the repository Level of detail for the semantic resolution description will vary based on firm’s goals.

Table

Description automatically generated with medium confidence

The logical grouping of business data will be store in these subject areas.

Text

Description automatically generated

Subject Area Id - uniquely identifies the group of tables within the enterprise from a business perspective

Subject Area Effective Date - identifies when this particular business view was last updated or created

Status Code - indicates the current approval status of the grouping (e.g., pending, approved, cancelled)

Subject Area Description - contains a detailed description of the business grouping

Individual performance metrics will be stored here

Text

Description automatically generated

Batch Cycle ID - a sequential identifier assigned during each load cycle to the data warehouse regardless of the refresh frequency (e.g., daily, weekly, monthly, etc.)

Load Date - provides a date/time stamp of when the specific ETL process ran

Elapsed Time and CPU Time - provide information on the computer resources that the ETL process used during this batch cycle

Process Return Code - captures the completion, warning or error return code value from the ETL process for examination by the data acquisition developer

Process Return Code Message - contains a narrative explanation of warning or error return codes for the particular operating system (if available)

Process Id - uniquely identify the ETL procedure in the warehouse environment

Process Effective Date - identifies when this particular process was included into the load cycle steps

Process Description - contains a detailed description of the ETL process from a technical and business perspective

Process Owner - lists the individual and/or groups that are technically responsible for this process

Performance of the data warehouse itself will be stored here

Text

Description automatically generated

Query ID - is a sequential identifier assigned by the DBMS or the data monitoring tool to uniquely identify a query request

Query Start and End Time - capture the date/time of when the query was initiated and completed

Number Of Rows Returned - indicates just that the count of rows successfully gathered by the query

Size of the Result Set - contains the size in kilobytes of the row returned in the query. This can be useful in diagnosing report problems with query reporting tools

User Id - contains the operating system identifier of the individual or group who requested the query. This column can be useful for identifying business needs from users who have frequent requests or long running queries

Server and Database Name - identify which version of the warehouse database is being queried such as development, QA, production or training Query Elapsed Time – clock time that it took the query to run

**Need to identify what operational systems we will be considering to master.**

We are looking to convert the operational data coming from Sippy and Cook Right (MySQL databases for now) ( more to come ) into Data AS A Service Offerings to both internal and external Customers. The operational data must be converted into time based, fact and dimension models so that the following usecases can be satisfied

1. Separation of Customer data across all of Miso Platforms by brand, product line, location and other hierarchies.
2. Ease of use and accessibility of data and aggregations by both internal and external customers.
3. Speed of analytics results and the ability to extract the operational data when the rolledup analytics require additional details.
4. Creation of clear hierarchies where role based authentication will determine what data can be seen , preconfigured by established user types and roles.
5. NoCode / Lowcode management and maintenance, means we can implement changes quickly from thought to implementation, full stack deployment and architecture thru a single language, industry standard SQL.
6. RESTAPI’s will allow master data to be shared across all of Miso and to Specific user types.
7. The entire application lifecycle and framework is entirely in a browser and cloud based.

We will be building this entire process on our own, with existing staff in parallel to existing efforts in all Miso related areas. There is no Buy vs Build, this effort is entirely cloud based without the benefit of professional services dedicated to this effort alone.

Key assumptions:

1. We see moderate to high level of complex data transformations, a majority of the data must be cleansed with hieararcy data for pivot and data warehouse filtering added in where the operation data is created. Data volumns are moderate with a few gigs of data initially.
2. The data is being replicated into the cloud from the many operational systems in place.
3. The Datawarehouse business rules execute based on a DBMS\_SCHEDULER that maintains job execution history, status and orchestration.

Typical Resources needed for a typical Data Warehousing Initiative that we have not identified yet:

 **Project Manager**: This person will oversee the progress and be responsible for the success of the data warehousing project.

 **Data Architect** : This role is responsible for building the many subject areas , creation of the entire database architecture and infrastructure, backup/recovery plan, as well as [performance tuning](https://www.1keydata.com/datawarehousing/performance.html).

 **Technical Architect**: This role is responsible for developing and implementing the overall technical architecture of the Cloud infrastructure.

 **ETL Developer**: This role is responsible for planning, developing, and deploying the extraction, transformation, and loading routine for the data warehouse from operational data sets.

 **Front End Developer**: This person is responsible for developing the front-end, whether it is client-server or over the web.

 **Trainer**: A significant role is the trainer. After the data warehouse is implemented, a person on the data warehouse team needs to work with the end users to get them familiar with how the front end is set up so that the end users can get the most benefit out of the data warehouse system.

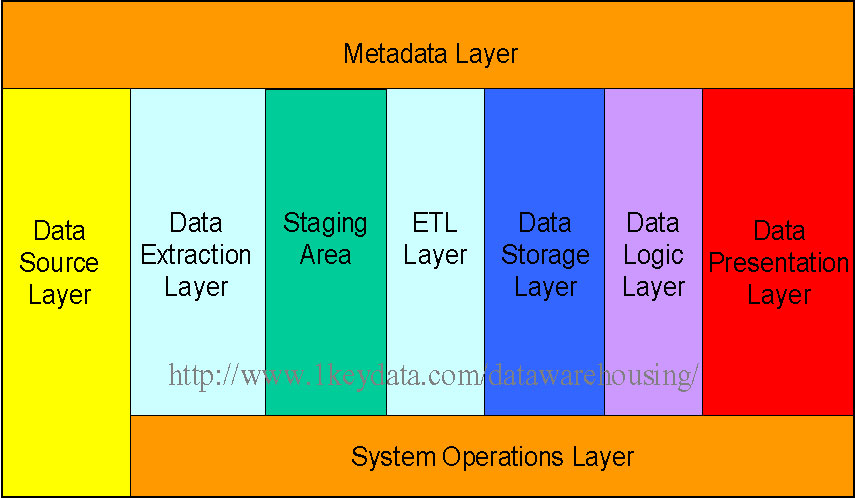
 **Data Modeler**: This role is responsible for taking the data structure that exists in the enterprise and model it into a schema that is suitable for OLAP analysis.

 **QA Group**: This role is responsible for ensuring the correctness of the data in the data warehouse. This role is more important than it appears, because bad data quality turns away users more than any other reason, and often is the start of the downfall for the data warehousing project.

The two primary modeling constructs are :

1. **Dimensional** modeling are *ease of use* and *query performance* of very large aggregated data with data duplication as an asset.
2. **Relational** modeling (ER) looks at the elimination of duplicate data itself and the relationships between different data elements for integrity and speed.

Steps to perform in the Data Warehouseing Architecture, identification of critical layers to uncover.



**Data Source Layer**

This represents the different data sources that feed data into the data warehouse. The data source can be of any format -- plain text file, relational database, IOT devices, source data from Vendors, REST API data from POS systems, almost anything.

Many different types of data can be a data source:

 Operations -- such as sales data, HR data, product data, inventory data, marketing data, systems data.

 Web server logs with user browsing data.

 Internal market research data.

 Third-party data, such as census data, demographics data, or survey data.

**Data Extraction Layer**

Data gets pulled from the data source into the data warehouse system. There is some data cleansing, in our case we are concerned that the operational data is storeing business event in UNIX EPOCH TIME (Seconds since 1970) and not in a data and time format needed.

**Staging Area**

This is where data sits prior to being scrubbed and transformed into a data warehouse / data mart. Having one common area makes it easier for subsequent data processing / integration.

**ETL Layer**

This is where data gains its "intelligence", as logic is applied to transform the data from a transactional nature to an analytical nature. This layer is also where data cleansing happens. The [ETL design phase](https://www.1keydata.com/datawarehousing/etl.html) is often the most time-consuming phase in a data warehousing project, and an [ETL tool](https://www.1keydata.com/datawarehousing/tooletl.html) is often used in this layer.

**Data Storage Layer**

This is where the transformed and cleansed data sit. Based on scope and functionality, 3 types of entities can be found here: data warehouse, data mart, and operational data store (ODS). In any given system, you may have just one of the three, two of the three, or all three types.

**Data Logic Layer**

This is where business rules are stored. Business rules stored here do not affect the underlying data transformation rules, but do affect what the report looks like.

**Data Presentation Layer**

This refers to the information that reaches the users. This can be in a form of a tabular / graphical report in a browser, an emailed report that gets automatically generated and sent everyday, or an alert that warns users of exceptions, among others. Usually an [OLAP tool](https://www.1keydata.com/datawarehousing/toololap.html) and/or a [reporting tool](https://www.1keydata.com/datawarehousing/toolreporting.html) is used in this layer.

**Metadata Layer**

This is where information about the data stored in the data warehouse system is stored. A logical data model would be an example of something that's in the metadata layer. A [metadata tool](https://www.1keydata.com/datawarehousing/toolmetadata.html) is often used to manage metadata.

**System Operations Layer**

This layer includes information on how the data warehouse system operates, such as ETL job status, system performance, and user access history.

We will discuss requirements for logical data modeling:

A logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Features of a logical data model include:

* Includes all entities and relationships among them.
* All attributes for each entity are specified.
* The primary key for each entity is specified.
* Foreign keys (keys identifying the relationship between different entities) are specified.
* Normalization occurs at this level.

The steps for designing the logical data model are as follows:

1. Specify primary keys for all entities.
2. Find the relationships between different entities.
3. Find all attributes for each entity.
4. Resolve many-to-many relationships.
5. Normalization.

Physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables. Features of a physical data model include:

* Specification all tables and columns.
* Foreign keys are used to identify relationships between tables.
* Denormalization may occur based on user requirements.
* Physical considerations may cause the physical data model to be quite different from the logical data model.
* Physical data model will be different for different RDBMS. For example, data type for a column may be different between MySQL and SQL Server.

The steps for physical data model design are as follows:

1. Convert entities into tables.
2. Convert relationships into foreign keys.
3. Convert attributes into columns.
4. Modify the physical data model based on physical constraints / requirements.

The figure below is an example of a physical data model.

In a data warehouse or a data mart, there are three areas of where data integrity needs to be enforced:

**Database level**

We can enforce data integrity at the database level. Common ways of enforcing data integrity include:

1. Referential integrity

The relationship between the primary key of one table and the foreign key of another table must always be maintained. For example, a primary key cannot be deleted if there is still a foreign key that refers to this primary key.

1. Primary key / Unique constraint

Primary keys and the UNIQUE constraint are used to make sure every row in a table can be uniquely identified.

1. Not NULL vs. NULL-able

For columns identified as NOT NULL, they may not have a NULL value.

1. Valid Values

Only allowed values are permitted in the database. For example, if a column can only have positive integers, a value of '-1' cannot be allowed.

**ETL process**

For each step of the ETL process, data integrity checks should be put in place to ensure that source data is the same as the data in the destination. Most common checks include record counts or record sums.

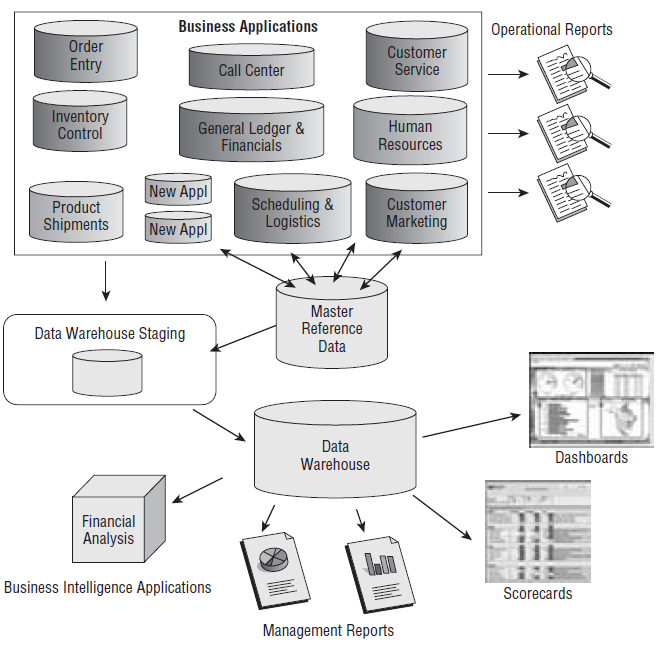
**Access level**

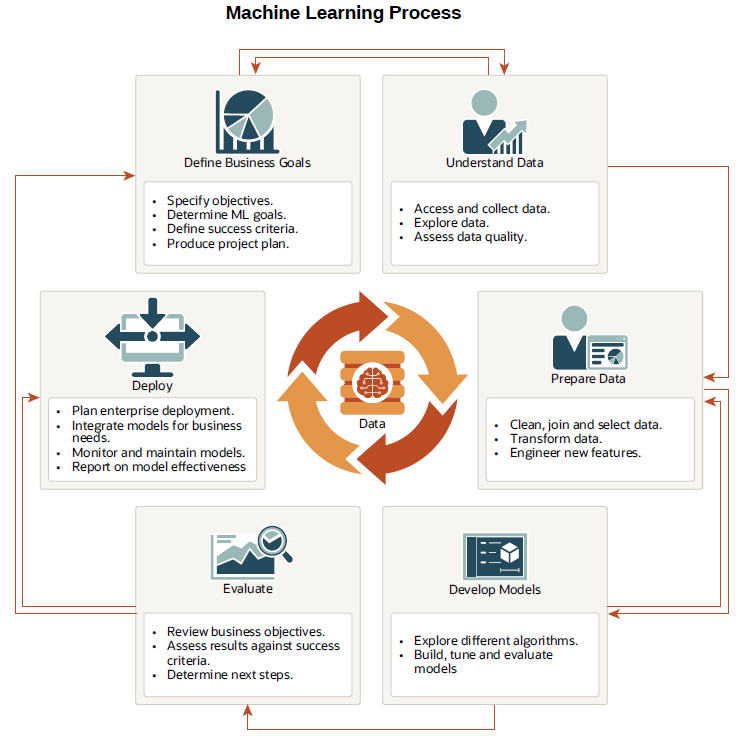
We need to ensure that data is not altered by any unauthorized means either during the ETL process or in the data warehouse. To do this, there needs to be safeguards against unauthorized access to data (including physical access to the servers), as well as logging of all data access history. Data integrity can only ensured if there is no unauthorized access to the data.

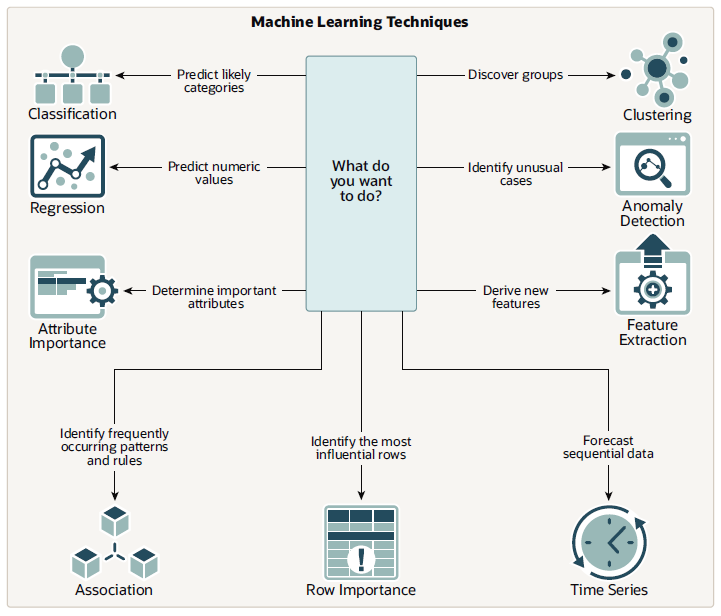
We will not implement MOLAP, ROLAP or HOLAP

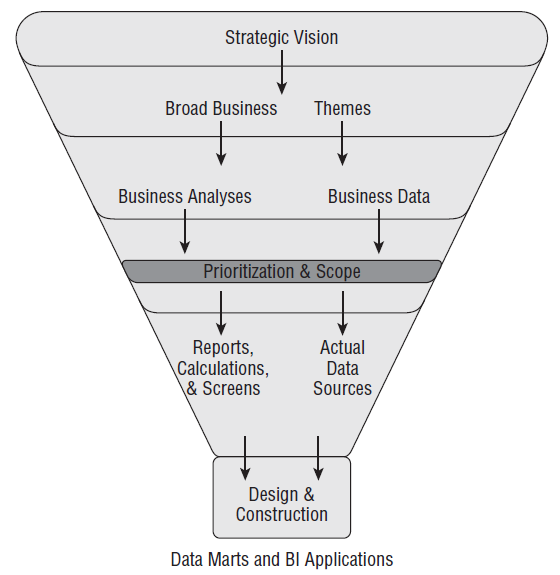
We are in a design phase The following are the typical steps involved in the data warehousing project cycle.

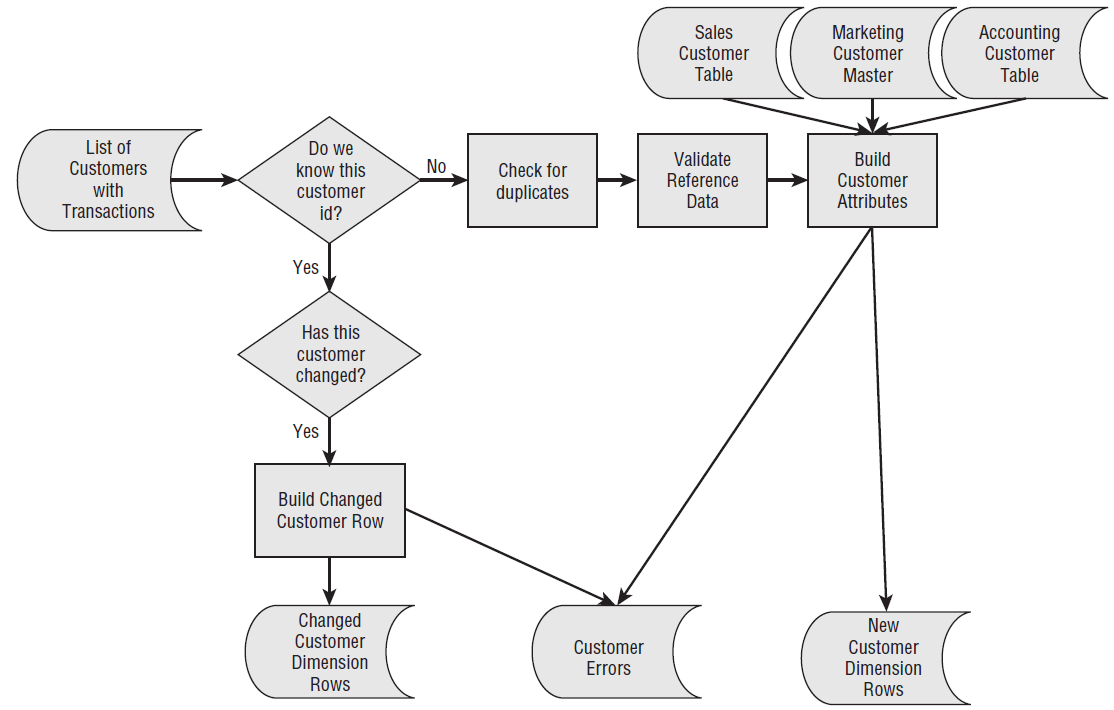
* Requirement Gathering - The primary goal of this phase is to identify what constitutes as a success for this particular phase of the data warehouse project.
* Physical Environment Setup Oracle ATP Cloud
* Data Modeling – Identification of data sources
* ETL
* OLAP Cube Design
* Front End Development
* Report Development
* Performance Tuning
* Query Optimization
* Quality Assurance
* Rolling out to Production
* Production Maintenance
* Incremental Enhancements

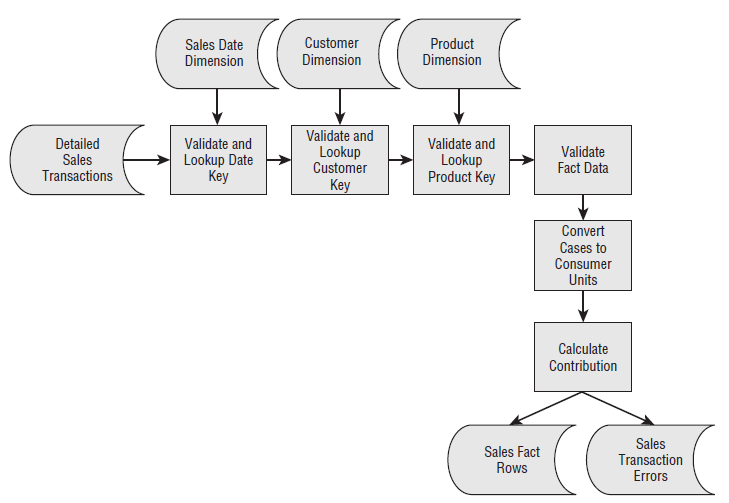












Diagram

Description automatically generated

**Glossary**

**Aggregation**: One way of speeding up query performance. Facts are summed up for selected dimensions from the original [fact table](https://www.1keydata.com/datawarehousing/fact-table-types.html). The resulting aggregate table will have fewer rows, thus making queries that can use them go faster.

**Attribute**: Attributes represent a single type of information in a dimension. For example, year is an attribute in the Time dimension.

**Conformed Dimension**: A dimension that has exactly the same meaning and content when being referred to from different fact tables.

**Data Mart**: Data marts have the same definition as the data warehouse (see below), but data marts have a more limited audience and/or data content.

**Data Warehouse**: A warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process (as defined by Bill Inmon).

**Data Warehousing**: The process of designing, building, and maintaining a data warehouse system.

**Dimension**: The same category of information. For example, year, month, day, and week are all part of the Time Dimension.

**Dimensional Model**: A type of data modeling suited for data warehousing. In a dimensional model, there are two types of tables: dimensional tables and fact tables. Dimensional table records information on each dimension, and fact table records all the "fact", or measures.

**Dimensional Table**: Dimension tables store records related to this particular dimension. No facts are stored in a dimensional table.

**Drill Across**: Data analysis across dimensions.

**Drill Down**: Data analysis to a child attribute.

**Drill Through**: Data analysis that goes from an OLAP cube into the relational database.

**Drill Up**: Data analysis to a parent attribute.

**ETL**: Stands for Extraction, Transformation, and Loading. The movement of data from one area to another.

[**Fact Table**](https://www.1keydata.com/datawarehousing/fact-table-types.html): A type of table in the dimensional model. A fact table typically includes two types of columns: fact columns and foreign keys to the dimensions.

**Hierarchy**: A hierarchy defines the navigating path for drilling up and drilling down. All attributes in a hierarchy belong to the same dimension.

**Metadata**: Data about data. For example, the number of tables in the database is a type of metadata.

**Metric**: A measured value. For example, "Total Sales" is a metric.

**MOLAP**: Multidimensional OLAP. MOLAP systems store data in the multidimensional cubes.

**OLAP**: On-Line Analytical Processing. OLAP should be designed to provide end users a quick way of slicing and dicing the data.

**ROLAP**: Relational OLAP. ROLAP systems store data in the relational database.

[**Snowflake Schema**](https://www.1keydata.com/datawarehousing/snowflake-schema.html): A common form of dimensional model. In a snowflake schema, different hierarchies in a dimension can be extended into their own dimensional tables. Therefore, a dimension can have more than a single dimension table.

[**Star Schema**](https://www.1keydata.com/datawarehousing/star-schema.html): A common form of dimensional model. In a star schema, each dimension is represented by a single dimension table.