ETL: Delivering (Loading)

Subsystem 9: Slowly Changing Dimension Manager

- One of the more important elements of the ETL architecture is the capability to implement slowly changing dimension (SCD) logic.
- The ETL system must determine how to handle an attribute value that has changed from the value already stored in the data warehouse.
- If the revised description is determined to be a legitimate and reliable update to previous information, the appropriate SCD technique must be applied.
- When the data warehouse receives notification that an existing row in a dimension has changed.
- There are three basic responses:
  - type 1 overwrite,
  - type 2 add a new row,
  - type 3 add a new column.

Type 1: Overwrite

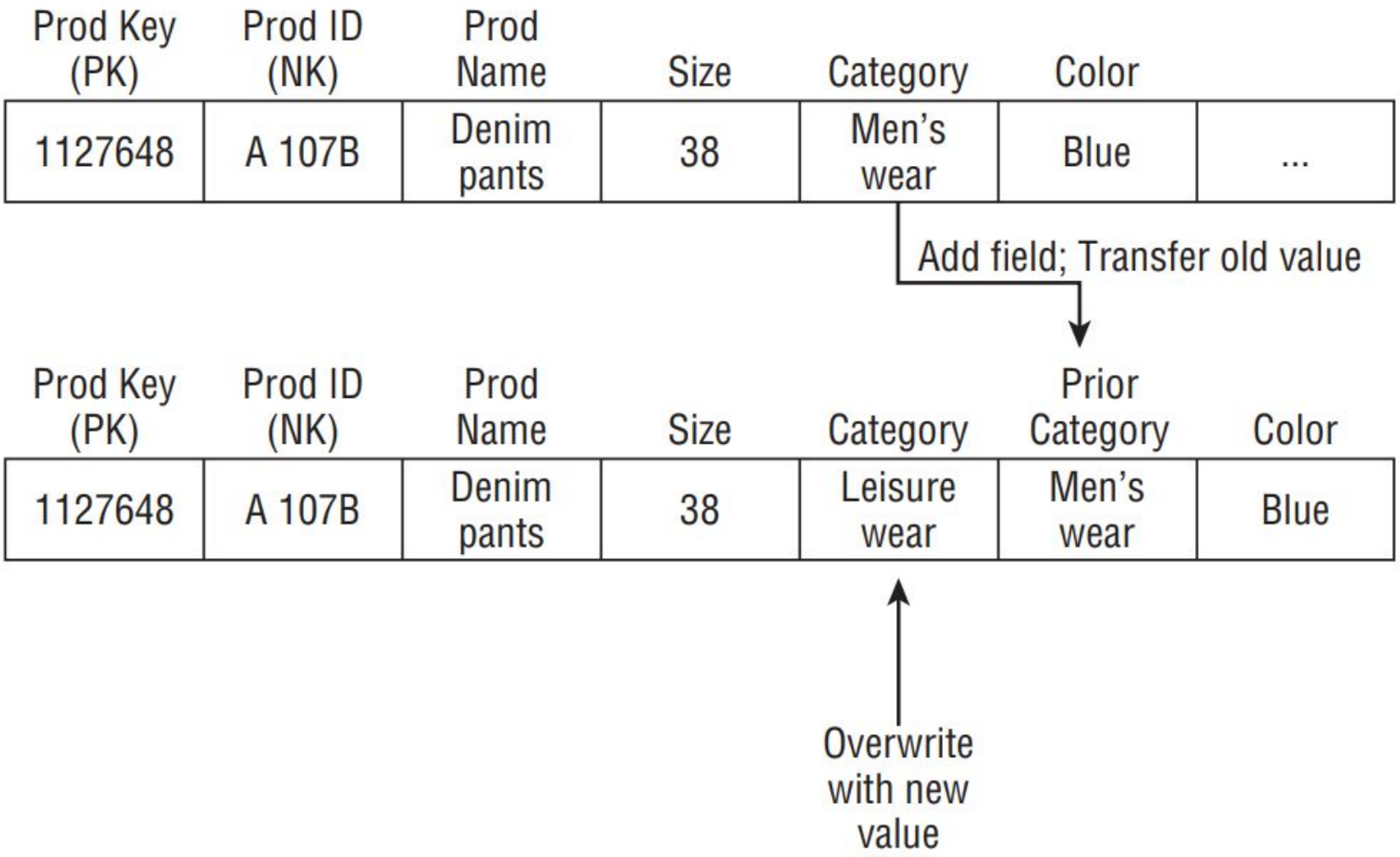
- Type 1 is appropriate when correcting data or when there is no business need to keep the history of previous values.
- For instance, you may receive a corrected customer address. In this case, overwriting is the right choice.
- Type 1 updates invalidate any aggregates built upon the changed column, so the dimension manager (subsystem 17) must notify the affected fact providers (subsystem 18) to drop and rebuild the affected aggregates.

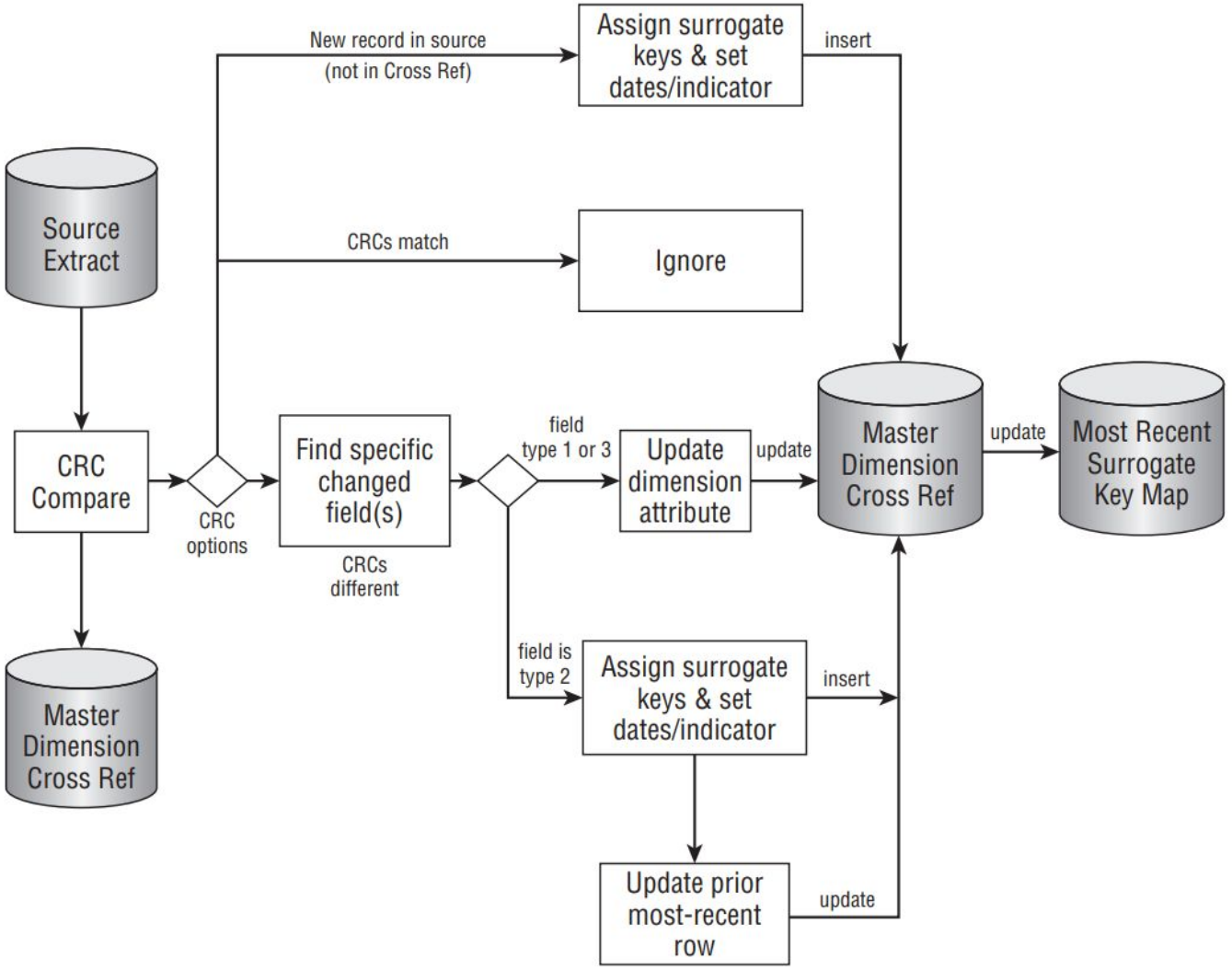
Type 2: Add New Row

- The type 2 SCD is the standard technique for accurately tracking changes in dimensions and associating them correctly with fact rows
- This is the main workhorse technique for handling dimension attribute changes that need to be tracked over time.
- For type 2 updates, copy the previous version of the dimension row and create a new dimension row with a new surrogate key.
- When a new type 2 row is created, you need at least a pair of timestamps, as well as an optional change description attribute. The pair of timestamps defines a span of time from the beginning effective time to the ending effective time when the complete set of dimension attributes is valid.
- Current Flag (current/expired)
- The type 2 process does not change history as the type 1 process does; thus type 2 changes don’t require rebuilding affected aggregate tables.

Type 3: Add New Attribute

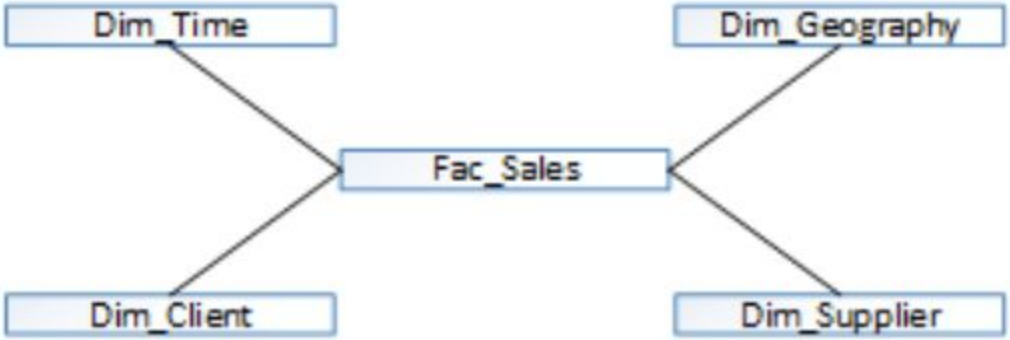
- The type 3 technique is designed to support attribute “soft” changes that allow a user to refer either to the old value of the attribute or the new value.
- For example, if a sales team is assigned to a newly named sales region, there may be a need to track the old region assignment, as well as the new one.
- The type 3 technique requires the ETL system to alter the dimension table to add a new column to the schema, if this situation was not anticipated.
- You then need to push the existing column values into the newly created column and populate the original column with the new values provided to the ETL system.

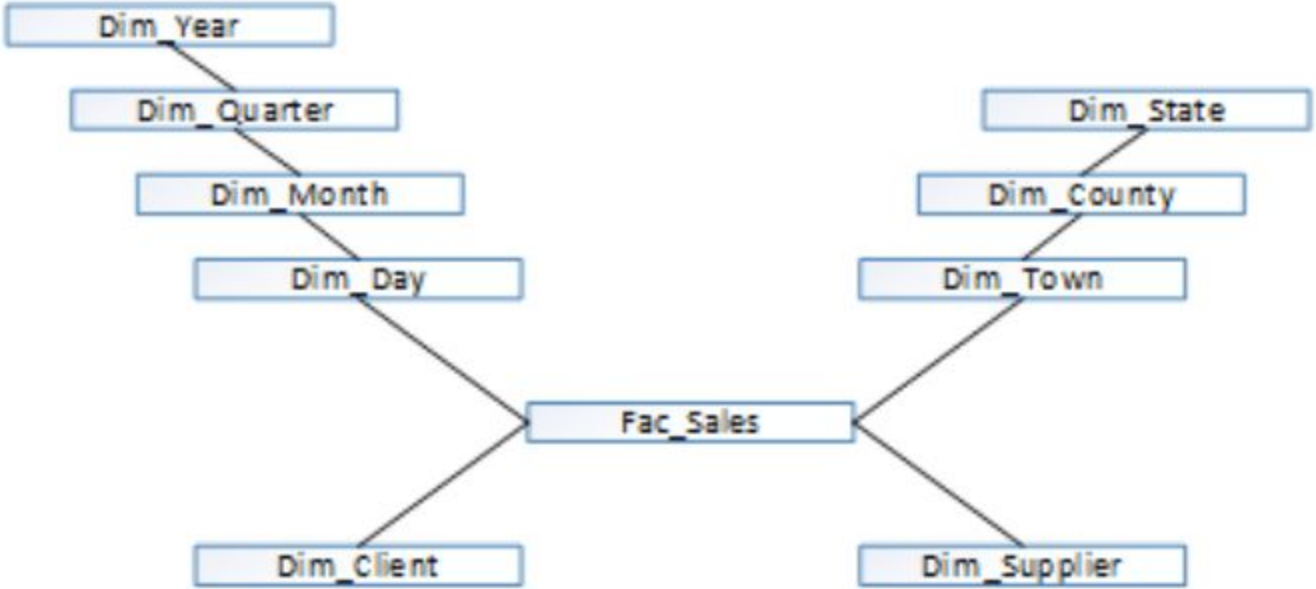

Prod Key (PK)
Prod ID (NK)
Prod Name
Size
Category
Color
1127648
A 107B
Denim pants
38
Men's wear
Blue
...
Add field; Transfer old value
Prod Key (PK)
Prod ID (NK)
Prod Name
Size
Category
Prior Category
Color
1127648
A 107B
Denim pants
38
Leisure wear
Men's wear
Blue
Overwrite with new value

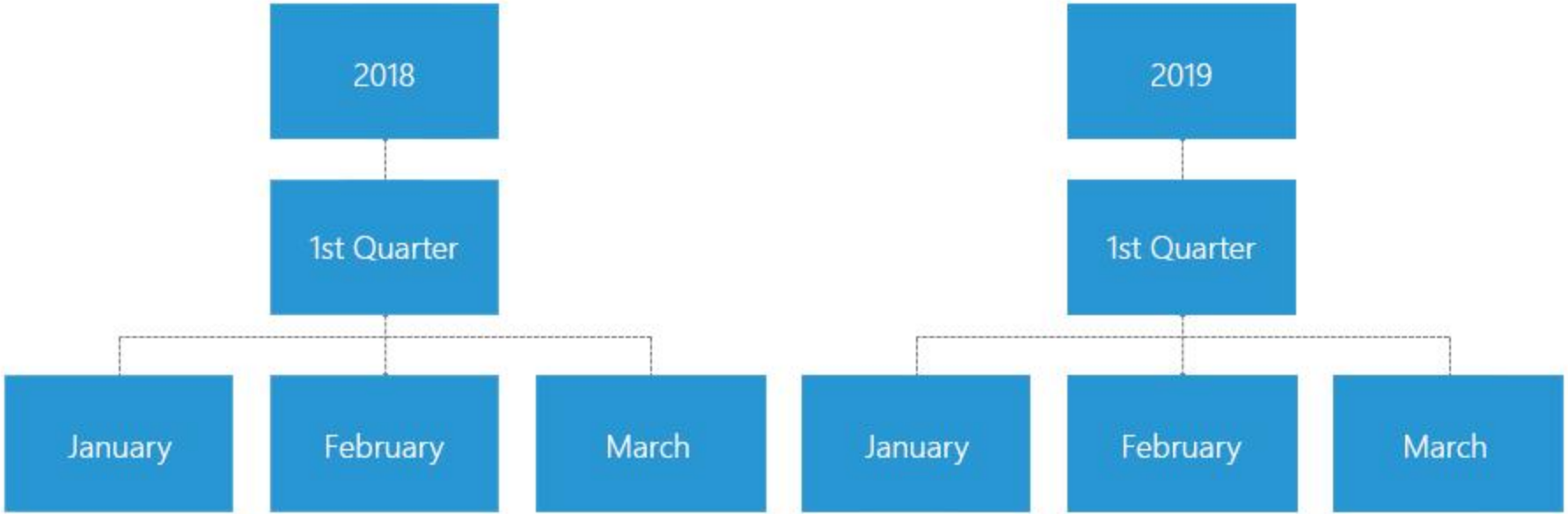

New record in source
(not in Cross Ref)
Assign surrogate keys & set dates/indicator
insert
Source Extract
CRCs match
Ignore
CRC Compare
CRC options
Find specific changed field(s)
CRCs different
field type 1 or 3
Update dimension attribute
update
Master Dimension Cross Ref
update
Most Recent Surrogate Key Map
Master Dimension Cross Ref
field is type 2
Assign surrogate keys & set dates/indicator
insert
Update prior most-recent row
update

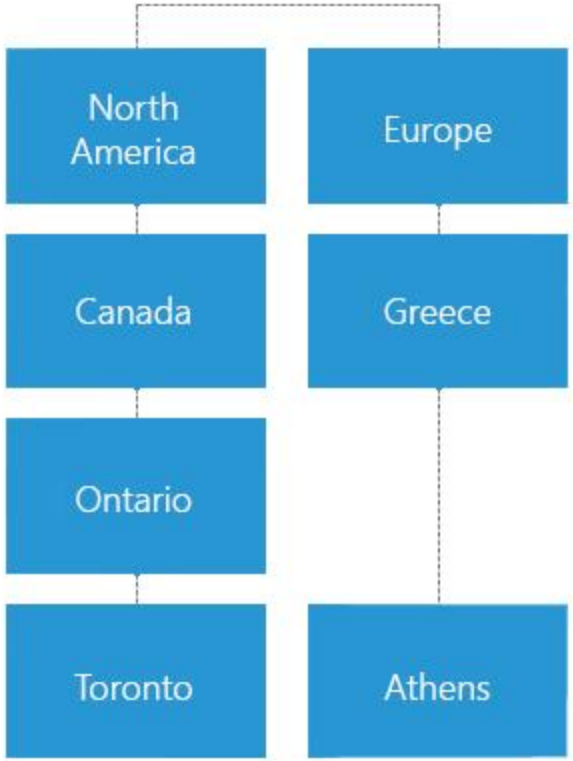
Subsystem 10: Surrogate Key Generator

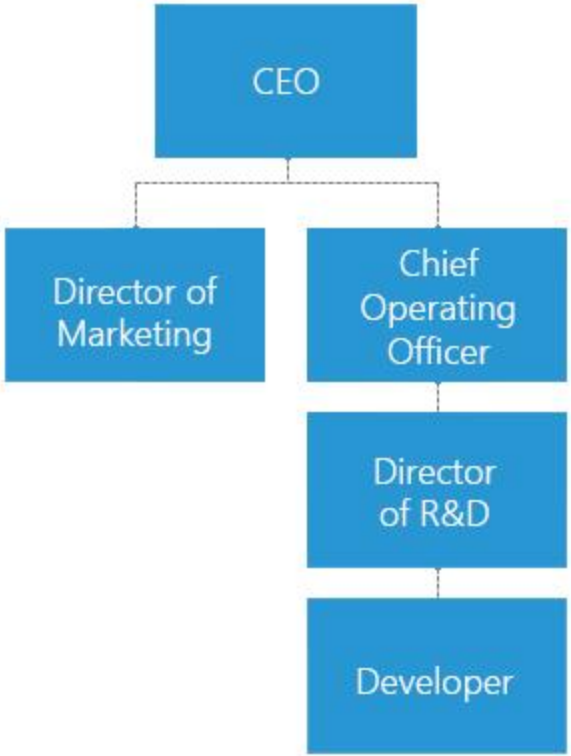
- The surrogate key generator should independently generate surrogate keys for every dimension; it should be independent of database instance.
- The goal of the surrogate key generator is to generate a meaningless key, typically an integer, to serve as the primary key for a dimension row.
- If the DBMS is used to assign surrogate keys, it is preferable for the ETL process to directly call the database sequence generator.
- For improved efficiency, consider having the ETL tool generate and maintain the surrogate keys.


Dim_Time
Dim_Geography
Fac_Sales
Dim_Client
Dim_Supplier


Dim_Year
Dim_Quarter
Dim_Month
Dim_Day
Dim_Client
Fac_Sales
Dim_State
Dim_County
Dim_Town
Dim_Supplier


2018
1st Quarter
January
February
March
2019
1st Quarter
January
February
March


North America
Europe
Canada
Greece
Ontario
Toronto
Athens


CEO
Director of Marketing
Chief Operating Officer
Director of R&D
Developer

Subsystem 11: Hierarchy Manager

- For intermediate tables and ETL staging tables storing dimensions with hierarchies, often normalized structures work best, to have the database assist in pre-verifying the many-to-one hierarchy relationships through referential integrity.
- Hierarchies are either **fixed** or **ragged**.
- A fixed depth hierarchy has a consistent number of levels and is simply modeled and populated as separate dimension attributes for each of the levels.
- Slightly ragged hierarchies like postal addresses are most often modeled as a fixed hierarchy.
- Profoundly ragged hierarchies are typically found with organization structures that are unbalanced and of indeterminate depth.
- The data model and ETL solution required to support these needs require the use of a bridge table containing the organization map.

Subsystem 12: Special Dimensions Manager

- **Date/Time Dimensions**
- The date and time dimensions are unique in that they are completely specified at the beginning of the data warehouse project, and they don’t have a conventional source.
- Typically, these dimensions are easy to build, but in a global enterprise environment, even this dimension can be challenging when taking into account multiple financial reporting periods or multiple cultural calendars.
- **Small Static Dimensions**
- A few dimensions are created entirely by the ETL system without a real outside source.
- These are usually small lookup dimensions where an operational code is translated into words.
- In these cases, there is no real ETL processing. The lookup dimension is simply created directly by the ETL team as a relational table in its final form.

Subsystem 13: Fact Table Builders

1. Transaction Fact Table Loader

- The transaction grain represents a measurement event defined at a particular instant.
- A line item on an invoice is an example of a transaction event.
- A scanner event at a cash register is another.
- Transaction grain fact tables are the largest and most detailed of the three types of fact tables.
- The transaction fact table loader receives data from the changed data capture system and loads it with the proper dimensional foreign keys.

1. Transaction Fact Table Loader (contd.)

- Addition of the most current records is the easiest case: simply bulk loading new rows into the fact table.
- In most cases, the target fact table should be partitioned by time to ease the administration and speed the performance of the table.
- An audit key, sequential ID, or date/timestamp column should be included to allow backup or restart of the load job.
- The addition of late arriving data is more difficult, requiring additional processing capabilities described in subsystem 16.
- In the event it is necessary to update existing rows, this process should be handled in two phases.
- The first step is to insert the corrected rows without overwriting or deleting the original rows, and then delete the old rows in a second step.
- Using a sequentially assigned single surrogate key for the fact table makes it possible to perform the two steps of insertion followed by deletion.

2. Periodic Snapshot Fact Table Loader (contd.)

- The periodic snapshot grain represents a regular repeating measurement or set of measurements, like a bank account monthly statement.
- This fact table also has a single date column, representing the overall period.
- The facts in this periodic snapshot table must be true to the grain and should describe only measures appropriate to the timespan defined by the period.
- Periodic snapshots are a common fact table type and are frequently used for account balances, monthly financial reporting, and inventory balances.
- The periodicity of a periodic snapshot is typically daily,weekly, or monthly.

2. Periodic Snapshot Fact Table Loader (contd.)

- Periodic snapshots have similar loading characteristics to those of transaction grain fact tables.
- The same processing applies for inserts and updates.
- Assuming data is promptly delivered to the ETL system, all records for each periodic load can cluster in the most recent time partition.
- Traditionally, periodic snapshots have been loaded en masse at the end of the appropriate period.

2. Periodic Snapshot Fact Table Loader (contd.)

- For example, a credit card company might load a monthly account snapshot table with the balances in effect at the end of the month.
- More frequently, organizations will populate a hot rolling periodic snapshot.
- In addition to the rows loaded at the end of every month, there are special rows loaded with the most current balances in effect as of the previous day.
- As the month progresses, the current month rows are continually updated with the most current information and continue in this manner rolling through the month.
- Note that the hot rolling snapshot can sometimes be difficult to implement if the business rules for calculating the balances at the period end are complex.
- Often these complex calculations are dependent on other periodic processing outside the data warehouse, and there is not enough information available to the ETL system to perform these complex calculations on a more frequent basis.

3. Accumulating Snapshot Fact Table Loader

- The accumulating snapshot grain represents the current evolving status of a process that has a well defined beginning and end.
- Usually, these processes are of short duration and therefore don’t lend themselves to the periodic snapshot.
- Order processing is the classic example of an accumulating snapshot. The order is placed, shipped, and paid for within one reporting period.
- All accumulating snapshot fact tables have a set of dates which describe the typical process workflow.

3. Accumulating Snapshot Fact Table Loader (cont'd)

- An order might have an order date, actual ship date, delivery date, final payment date, and return date.
- In this example, these five dates appear as five separate date-valued foreign surrogate keys.
- When the order row is first created, the first of these dates is well defined, but perhaps none of the others have yet happened.
- This same fact row is subsequently revisited as the order winds its way through the order pipeline.
- Each time something happens, the accumulating snapshot fact row is destructively modified.
- The date foreign keys are overwritten, and various facts are updated.
- Often the first date remains inviolate because it describes when the row was created, but all the other dates may well be overwritten, sometimes more than once.
- Many RDBMSs utilize variable row lengths.

Fact table remarks

- Often it makes sense to utilize all three fact table types to meet various needs.
- Periodic history can be captured with periodic extracts, and all the infinite details involved in the process can be captured in an associated transaction grain fact table.
- The presence of many situations that violate standard scenarios or involve repeated looping through the process would prohibit the use of an accumulating snapshot.

Subsystem 14: Surrogate Key Pipeline

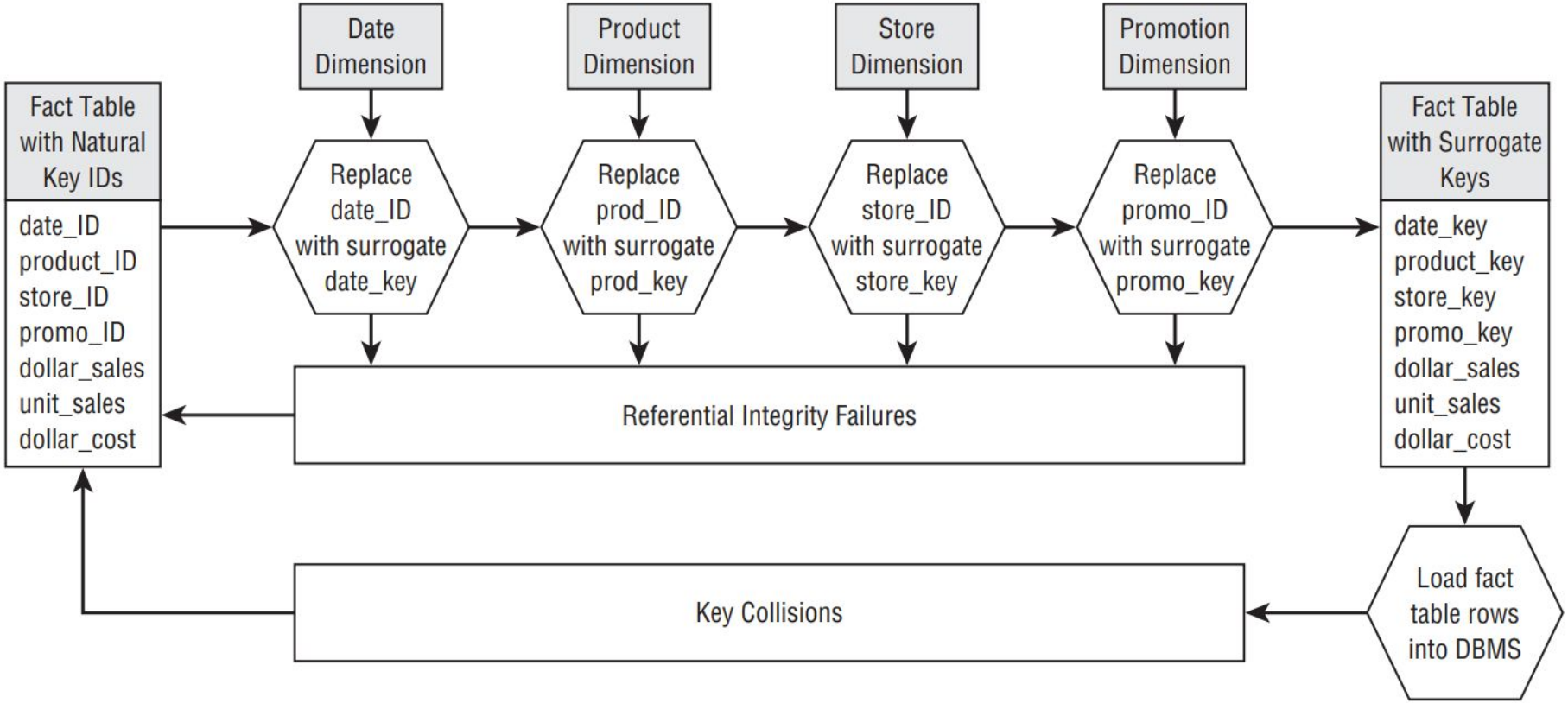
- Every ETL system must include a step for replacing the operational natural keys in the incoming fact table row with the appropriate dimension surrogate keys.
- An important requirement for loading fact tables is maintaining referential integrity with the associated dimension tables.
- If there’s a row in a sales fact table for product surrogate key 323442, you need to have a row in the product dimension table with the same key, or you won’t know what you’ve sold.
- You have a sale for what appears to be a nonexistent product.
- Even worse, without the product key in the dimension, a business user can easily construct a query that will omit this sale without even realizing it.

Subsystem 14: Surrogate Key Pipeline (cont'd)

- The key lookup process should result in a match for every incoming natural key or a default value.
- In the event there is an unresolved referential integrity failure during the lookup process, you need to feed these failures back to the responsible ETL process for resolution.
- Likewise, the ETL process needs to resolve any key collisions that might be encountered during the key lookup process.
- After the fact table data has been processed and just before loading into the presentation layer, a surrogate key lookup needs to occur to substitute the operational natural keys in the fact table record with the proper current surrogate key.

Subsystem 14: Surrogate Key Pipeline (cont'd)

- To preserve referential integrity, always complete the updating of the dimension tables first.
- In that way, the dimension tables are always the legitimate source of primary keys you must replace in the fact table.
- Each time you need the current surrogate key, look up all the rows in the dimension with the natural key equal to the desired value, and then select the surrogate key that aligns with the historical context of the fact row using the current row indicator or begin and end effect dates.
- In the event a key collision is recognized, the surrogate key pipeline process halts, either discards or corrects and loads the row, and writes an explanatory row into the error event schema.


Date Dimension
Product Dimension
Store Dimension
Promotion Dimension
Fact Table with Natural Key IDs
date_ID
product_ID
store_ID
promo_ID
dollar_sales
unit_sales
dollar_cost
Replace date_ID with surrogate date_key
Replace prod_ID with surrogate prod_key
Replace store_ID with surrogate store_key
Replace promo_ID with surrogate promo_key
Fact Table with Surrogate Keys
date_key
product_key
store_key
promo_key
dollar_sales
unit_sales
dollar_cost
Referential Integrity Failures
Key Collisions
Load fact table rows into DBMS

Multivalued Dimensions and Bridge Tables

- In a classic dimensional schema, each dimension attached to a fact table has a single value consistent with the fact table grain.
- But there are a number of situations in which a dimension is legitimately multivalued.
- For example, a patient receiving a healthcare treatment may have multiple simultaneous diagnoses.
- In these cases, the multivalued dimension must be attached to the fact table through a group dimension key to a **bridge table** with one row for each simultaneous diagnosis in a group.

Multivalued Dimensions and Bridge Tables (cont’d)

- A multivalued bridge table may need to be based on a type 2 slowly changing dimension.
- For example, the bridge table that implements the many-to-many relationship between bank accounts and individual customers usually must be based on type 2 account and customer dimensions.
- In this case, to prevent incorrect linkages between accounts and customers, the bridge table must include effective and expiration date/time stamps, and the requesting application must constrain the bridge table to a specific moment in time.

ER_Admittance_Transactions

ER_Admittance_ID	Diagnosis_ Code
27	T41.201
27	Z77.22
28	K35.2
28	B58.09
28	I13.10
29	T41.201
29	Z77.22

Diagnosis_Group

Diagnosis_ Group_Key	Diagnosis_Code_List
1	B58.09, I13.10, K35.2
2	T41.201, Z77.22

Subsystem 15: Multivalued Dimension Bridge Table Builder

- The challenge for the ETL team is building and maintaining the bridge table.
- As multivalued relationships to the fact row are encountered, the ETL system has the choice of either making each set of observations a unique group or reusing groups when an identical set of observations occurs.

Subsystem 16: Late Arriving Data Handler

- Data warehouses are usually built around the ideal assumption that measured activity (fact records) arrive in the data warehouse at the same time as the context of the activity (dimension records).
- When you have both the fact records and the correct contemporary dimension rows, you have the luxury of first maintaining the dimension keys and then using these up-to-date keys in the accompanying fact rows.
- However, for a variety of reasons, the ETL system may need to process late arriving fact or dimension data.

Subsystem 16: Late Arriving Data Handler (cont'd)

- In some environments, there may need to be special modifications to the standard processing procedures to deal with late arriving facts, namely fact records that come into the warehouse very much delayed.
- This is a messy situation because you have to search back in history to decide which dimension keys were in effect when the activity occurred.
- In addition, you may need to adjust any semi-additive balances in subsequent fact rows.
- In a heavily compliant environment, it is also necessary to interface with the compliance subsystem because you are about to change history.

Subsystem 16: Late Arriving Data Handler (cont'd)

- Eg. Customer dimension data arrives late.
- the ETL system needs to support two situations.
- The first situation is to support late arriving type 2 dimension updates.
- In this situation, you need to add the revised customer row to the dimension with a new surrogate key and then destructively modify the respective fact rows' foreign key to the customer table.
- The effective dates for the affected dimension rows also need to be reset.

Subsystem 16: Late Arriving Data Handler (cont'd)

- The second situation occurs when you receive a fact row with what appears to be a valid customer natural key, but you have not yet loaded this customer in the customer dimension.
- If the customer is valid, but not yet processed customer, you should assign a new customer surrogate key with a set of dummy attribute values in a new customer dimension row.
- You then return to this dummy dimension row at a later time and make type 1 overwrite changes to its attributes when you get complete information on the new customer.
- At least this step avoids destructively changing any fact table keys.

Subsystem 17: Dimension Manager System

- The dimension manager is a centralized authority who prepares and publishes conformed dimensions to the data warehouse community.
- A conformed dimension is by necessity a centrally managed resource: Each conformed dimension must have a single, consistent source.
- There may be multiple dimension managers in an organization, each responsible for a dimension. The dimension manager's responsibilities include the following ETL processing:
    - Implement the common descriptive labels agreed to by the data stewards and stakeholders during the dimension design.
    - Add new rows to the conformed dimension for new source data, generating new surrogate keys.
    - Add new rows for type 2 changes to existing dimension entries, generating new surrogate keys.
    - Modify rows in place for type 1 changes and type 3 changes, without changing the surrogate keys
    - Update the version number of the dimension if any type 1 or type 3 changes are made.
    - Replicate the revised dimension simultaneously to all fact table providers.

Subsystem 17: Dimension Manager System

- It is easier to manage conformed dimensions in a single tablespace DBMS on a single machine because there is only one copy of the dimension table.
- However, managing conformed dimensions becomes more difficult in multiple tablespace,multiple DBMS, or multi-machine distributed environments.
- In these situations, the dimension manager must carefully manage the simultaneous release of new versions of the dimension to every fact provider.
- Each conformed dimension should have a version number column in each row that is overwritten in every row whenever the dimension manager releases the dimension.
- This version number should be utilized to support any drill-across queries to assure that the same release of the dimension is being utilized.

Subsystem 18: Fact Provider System

- The fact provider is responsible for receiving conformed dimensions from the dimension managers.
- The fact provider is responsible creation, maintenance, and use of fact tables.
- If fact tables are used in any drill-across applications, then by definition the fact provider must be using conformed dimensions provided by the dimension manager.

Subsystem 18: Fact Provider System

- The fact provider's responsibilities are more complex and include:
  - Receive or download replicated dimension from the dimension manager.
  - In an environment in which the dimension cannot simply be replicated but must be locally updated, the fact provider must process dimension records marked as new and current to update current key maps in the surrogate key pipeline
  - Add all new rows to fact tables after replacing their natural keys with correct surrogate keys.
  - Check the quality of base and aggregate fact tables.
  - Modify rows in all fact tables for error correction, accumulating snapshots, and late arriving dimension changes.
  - Remove aggregates that have become invalidated.
  - Recalculate affected aggregates. (Subsystem 19)
  - Bring updated fact and dimension tables online.
  - Inform users that the DW has been updated. Tell them if major changes have been made, including dimension version changes, postdated records being added, and changes to historical aggregates.

Subsystem 19: Aggregate Builder

- Aggregations are like indexes; they are specific data structures created to improve performance.
- Aggregates can have a significant impact on performance.
- The ETL system needs to effectively build and use aggregates without causing significant distraction or consuming extraordinary resources and processing cycles.
- Although you can depend on informal feedback to some extent, a log of frequently attempted slow-running queries should be captured.
- You should also try to identify the nonexistent slow-running queries that never made it into the log because they never run to completion, or aren’t even attempted due to known performance challenges.

Subsystem 20: OLAP Cube Builder

- OLAP servers present dimensional data in an intuitive way, enabling a range of analytic users to slice and dice data.
- Because many OLAP systems do not directly address referential integrity or data cleaning, the preferred architecture is to load OLAP cubes after the completion of conventional ETL processes.
- Note that some OLAP tools are more sensitive to hierarchies than relational schemas.
- It is important to strongly enforce the integrity of hierarchies within dimensions before loading an OLAP cube.
- In case of Type 2 SCDs, a new surrogate key is just treated as a new member.
- Type1 SCDs that restate history do not fit OLAP well.
- Overwrites to an attribute value can cause all the cubes using that dimension to be reprocessed in the background, become corrupted, or be dropped.

Subsystem 21: Data Propagation Manager

- Many organizations need to extract data from the DW to share with business partners, customers, and/or vendors for strategic purposes.
- Similarly, some organizations are required to submit data to various government organizations for reimbursement purposes, such as healthcare organizations that participate in the Medicare program.
- Many organizations have acquired package analytic applications. Typically, these applications cannot be pointed directly against the existing data warehouse tables, so data needs to be extracted from the DW and loaded into proprietary data structures required by the analytic applications.
- Finally, most data mining tools need data extracted from the data warehouse and fed to the data mining tool in a specific format.
- All the situations previously described require extraction from the DW/BI presentation server, possibly some light transformation, and loading into a target format—in other words ETL.
- Data propagation should be considered a part of the ETL system; ETL tools should be leveraged to provide this capability.