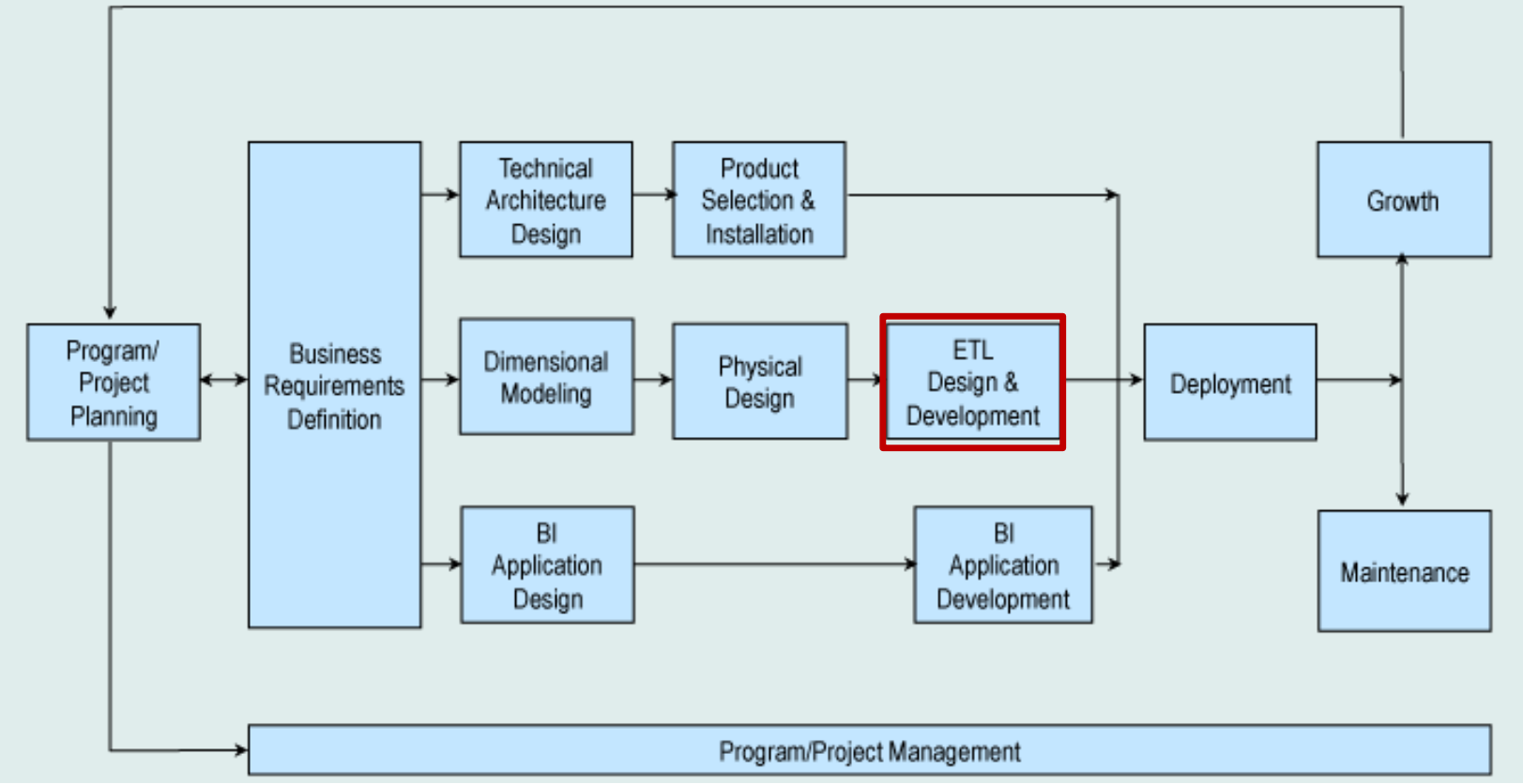
Introduction

School of Information Studies
Syracuse University

Agenda

- Learn about ETL approaches and architectures
- Discuss common subsystems of ETL
- Describe data extraction and staging techniques
- Explain common and advanced ETL patterns

Kimball Lifecycle



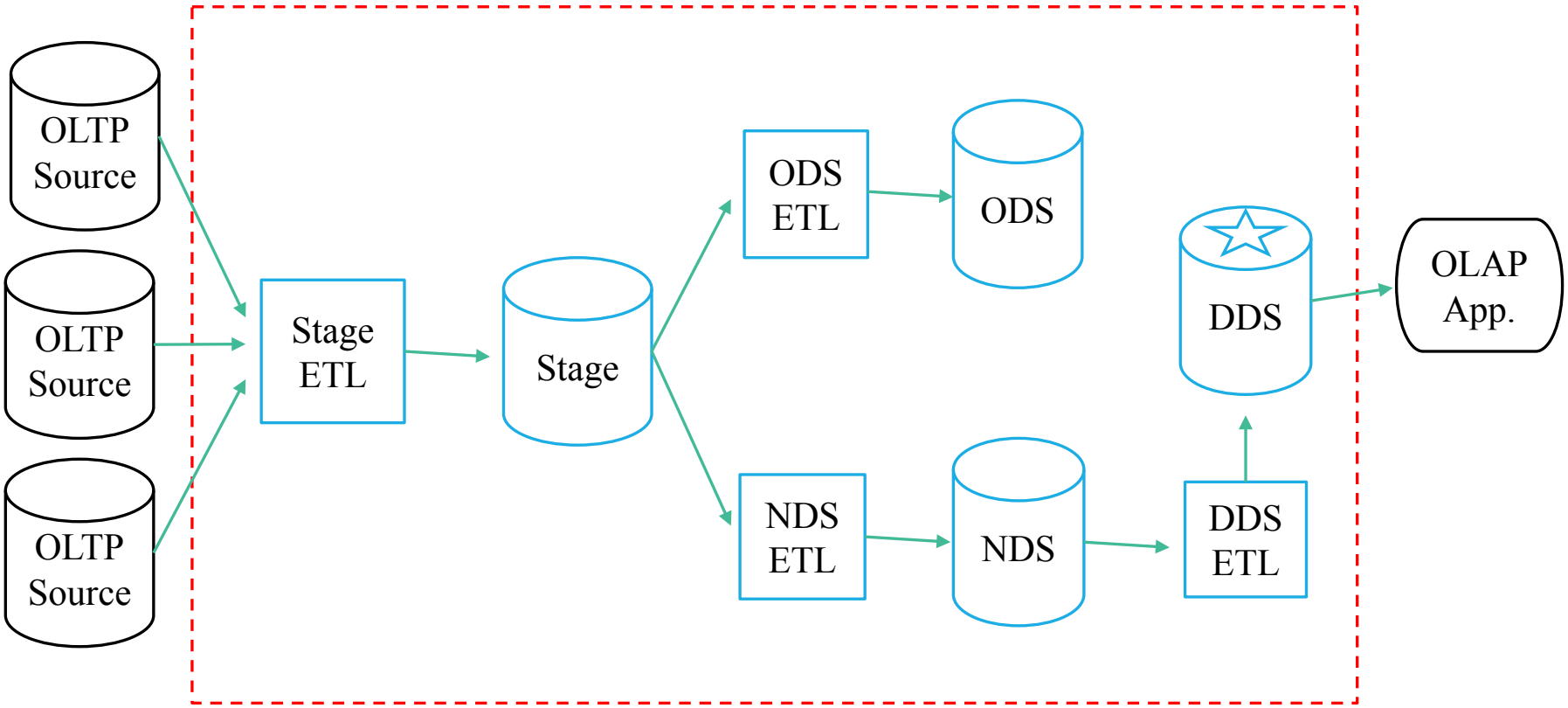
ETL Explained

School of Information Studies
Syracuse University

ETL Explained

- ETL stands for extract, transform, load.
- It's the process of:
  - Retrieving data from the OLTP sources,
  - Transforming it, then
  - Placing it into the data warehouse.
- According to Kimball, ETL is a time-consuming process, consuming up to 70% of your data warehousing effort.
- ETL is code but is not typically written in code. We use tooling to write the code for us.

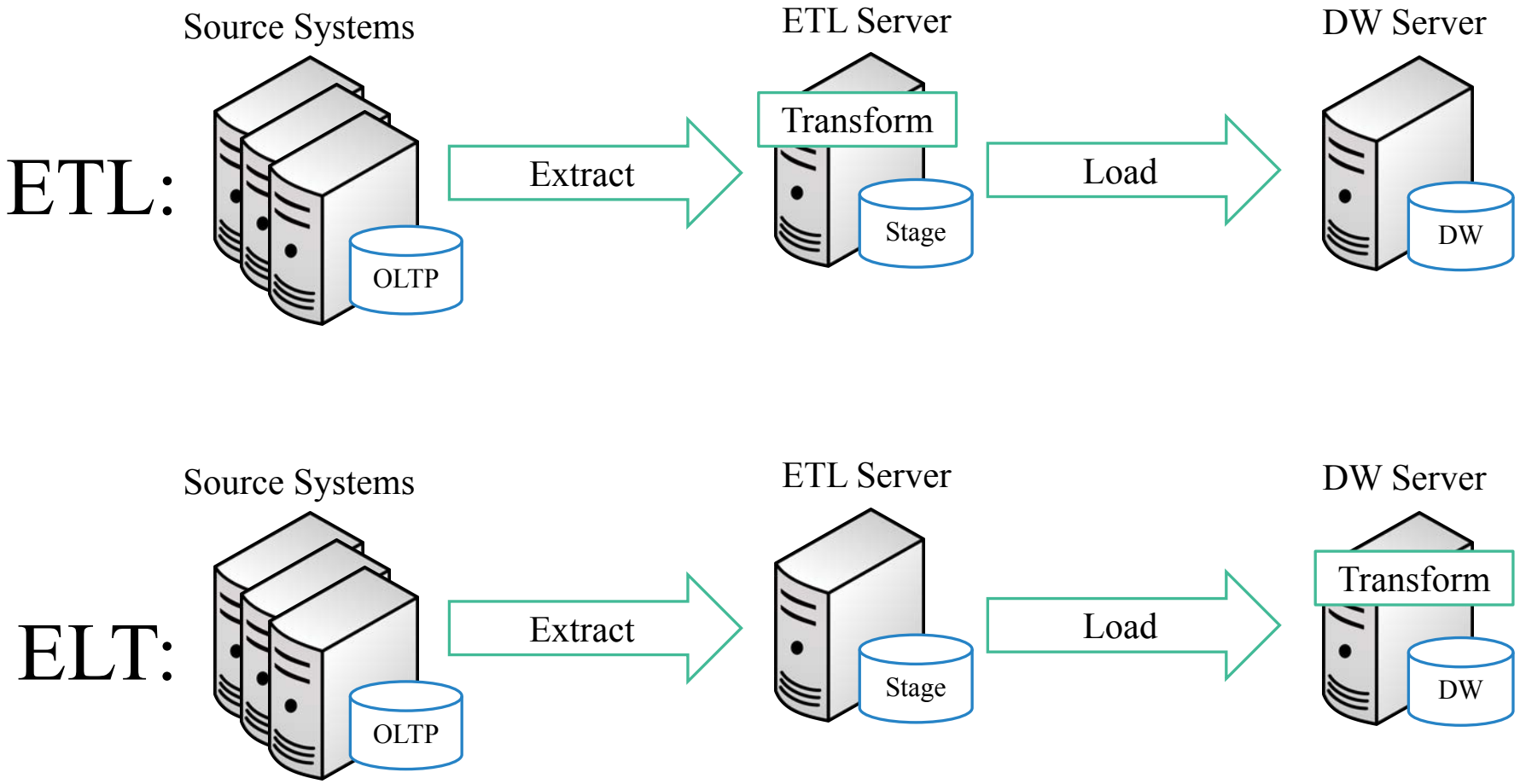
ETL Is for Moving Data Around the Data Warehouse!



ETL vs. ELT

School of Information Studies
Syracuse University

ETL vs. ELT


ETL:
Source Systems
OLTP
Extract
ETL Server
Transform
Stage
Load
DW Server
DW
ELT:
Source Systems
OLTP
Extract
ETL Server
Stage
Load
DW Server
Transform
DW

Four Approaches to Moving Data

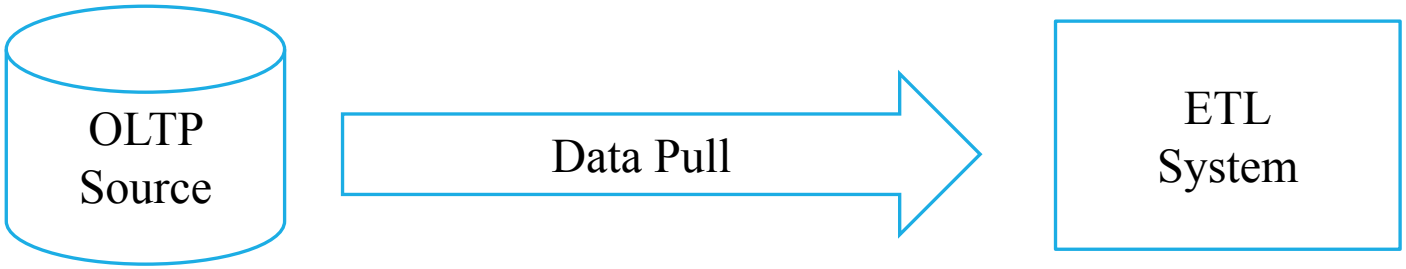
School of Information Studies
Syracuse University

Four Approaches to Moving Data

1. Pull from source
2. Push from source
3. Export and push
4. Pull from log

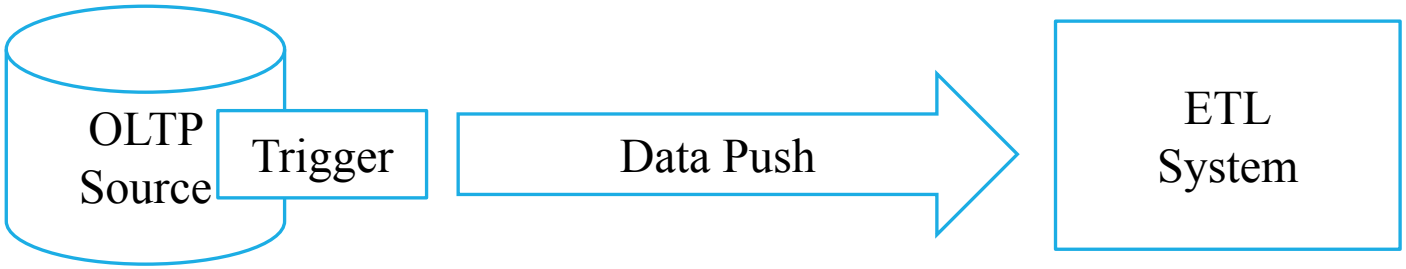
1: Pull From Source

- Most common approach
- ETL system connects directly to OLTP database



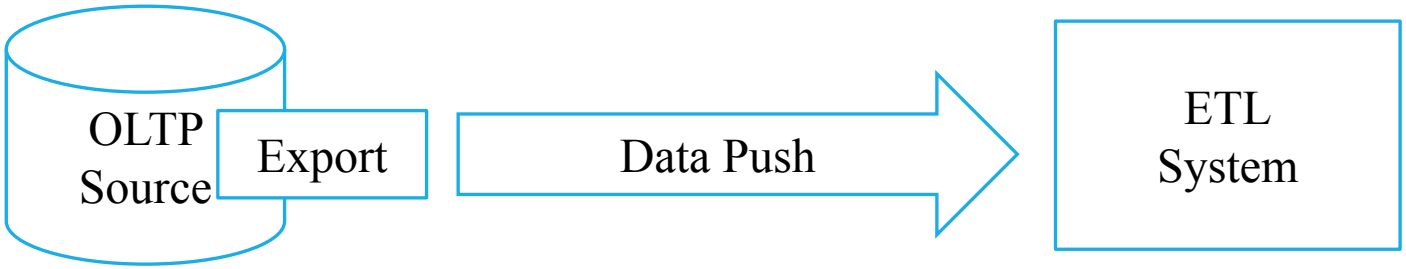
2: Push From Source

- Triggers in the source system push changes out.
- Useful for replaying transactions and changes.



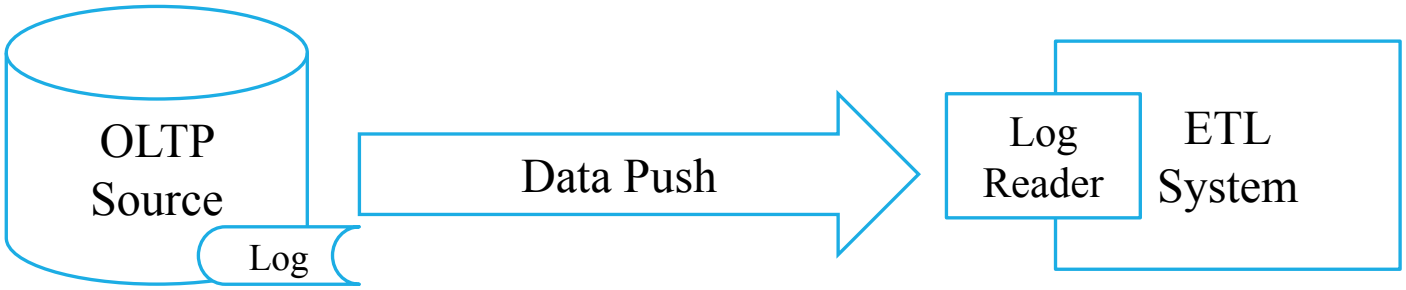
3: Export and Push

- A batch process performs an export from the source.
- Typical when the ETL system cannot query the source because it is not a DBMS.



4: Pull From Log

- The DBMS transaction log records changes. A log reader reads the log.
- Useful for replaying transactions and changes.



Three Approaches to ETL Processing

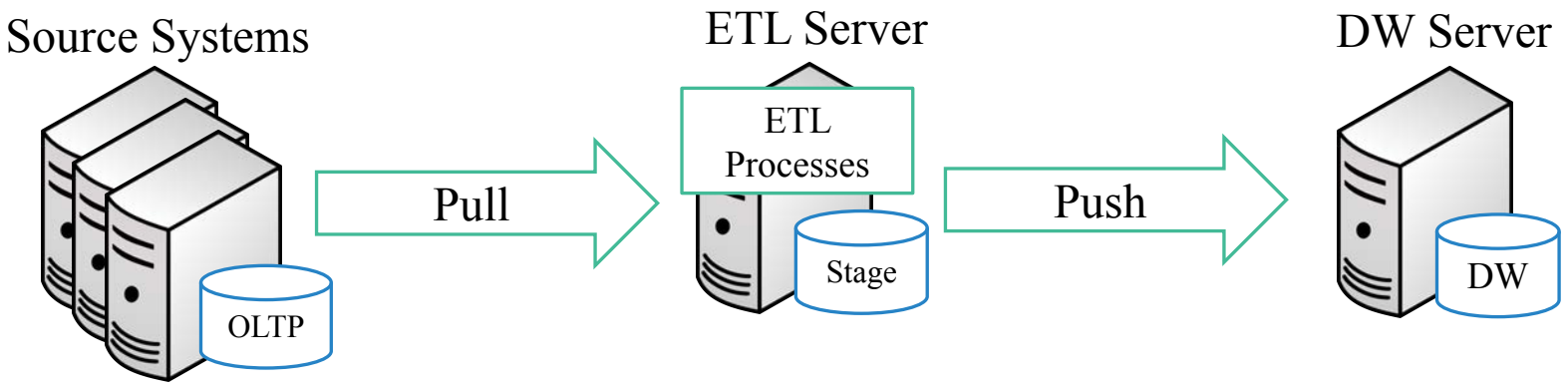
School of Information Studies
Syracuse University

Three Approaches to ETL Processing

1. Processing on ETL server
2. Processing in data warehouse
3. Processing at OLTP source

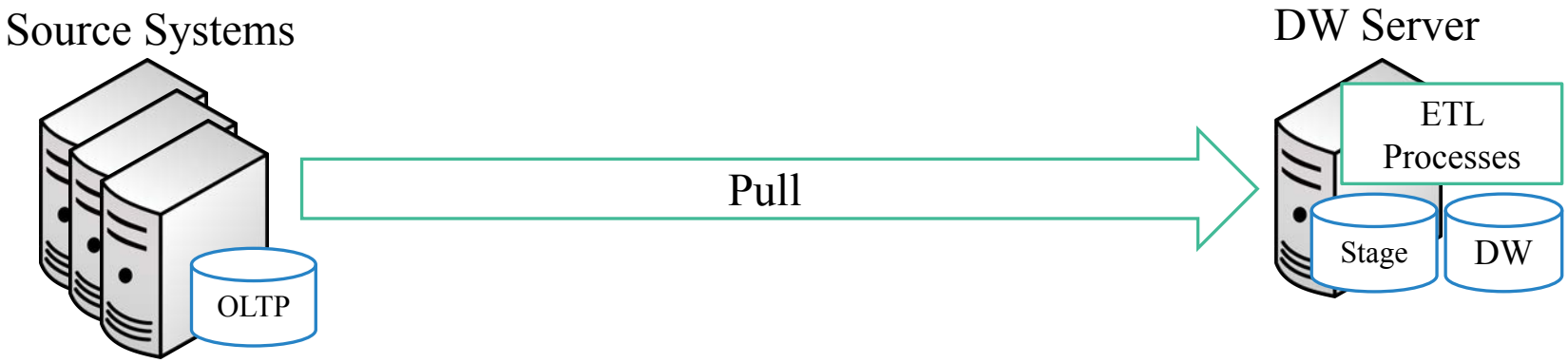
Processing at ETL Server

- Best performance. Does not stress source or target (DW).
- Most common processing framework.



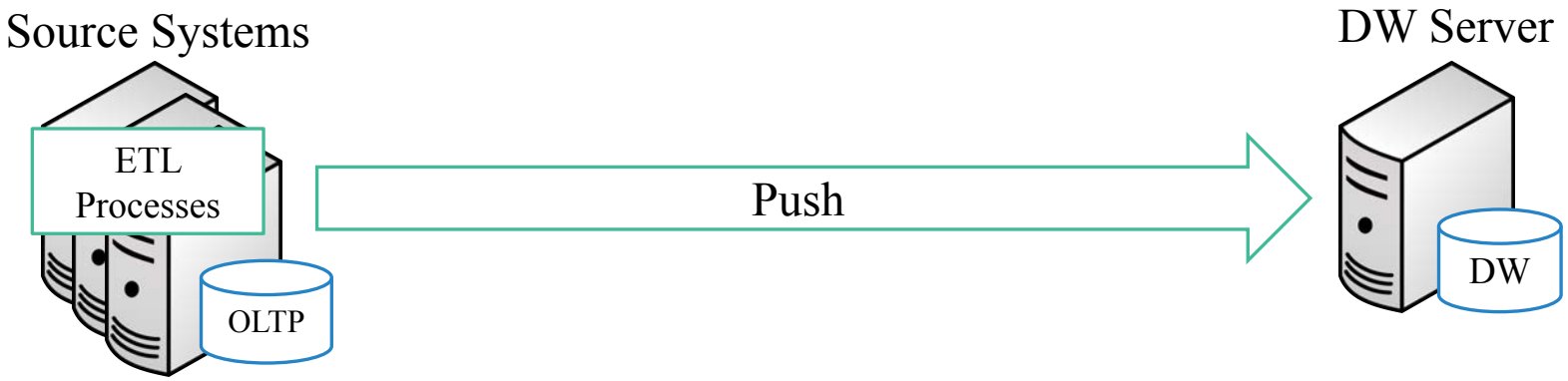
Processing at DW Server

- Saves on additional licensing costs.
- Makes sense in an MPP data warehouse where compute cycles may be available.



Processing at Source

- A solution for real-time data warehousing.
- This is not a total solution; it would only be used in specific cases.



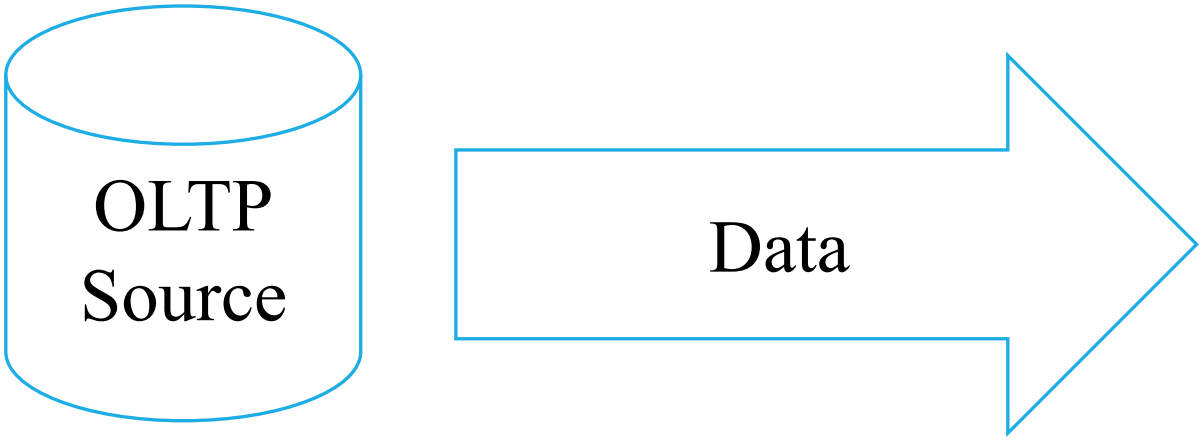
Data Extraction

School of Information Studies
Syracuse University

Kimball: Four Major ETL Operations

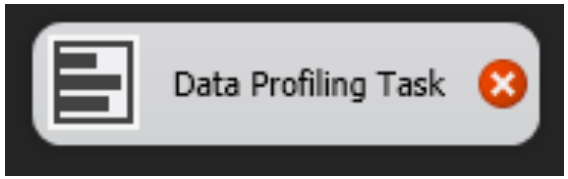
1. **Extract** the data from its source.
2. **Cleanse and Conform** to improve data accuracy and quality (transform).
3. **Deliver** the data into the presentation server (load).
4. **Manage** the ETL process itself.

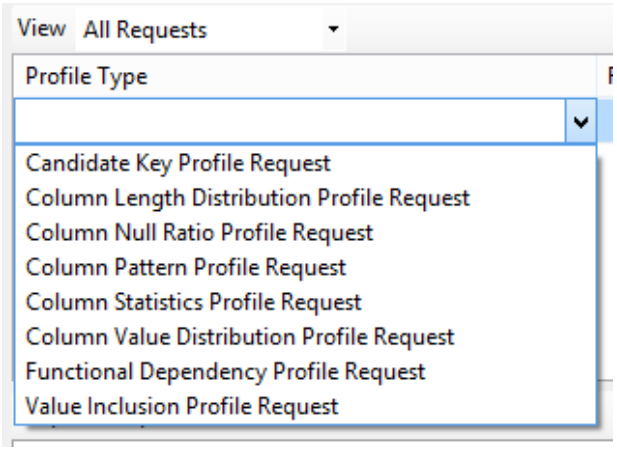
Data Extraction Subsystems



Data Profiling

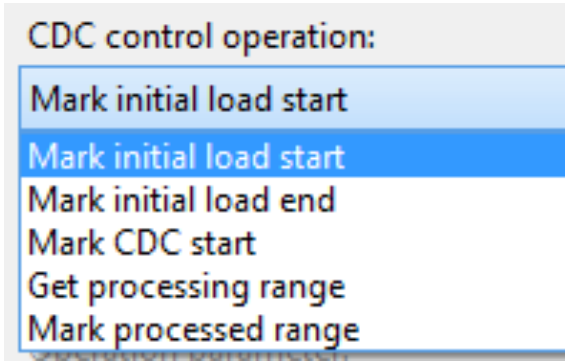
- Helps you to understand the source data.
  - Identify candidate business keys
  - Functional dependencies
  - Nulls
  - Etc.
- Helps us figure out the facts, dimensions, and source-to-target mapping.
- Valuable tool when you do not have the SQL chops to query the source data.





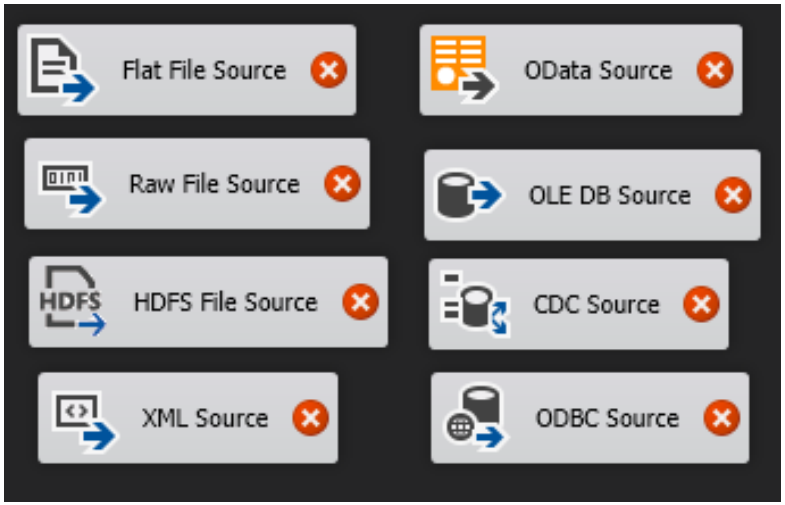
Change Data Capture System

- A means to detect which data are part of the **incremental load** (selective processing).
- Difficult to get right, needs a lot of testing.
- Common approaches:
  - **Audit columns** in source data (last update)
  - **Timed extracts** (e.g., yesterday's records)
  - **Diff compare** with CRC /Hash
  - **Database transactions logs**
  - **Triggers/message queues**



Extract System

- Getting data from the source system—a fundamental component!
- Two methods:
  - **File:** extracted output from a source system. Useful with third parties/legacy systems.
  - **Stream:** initiated data flows out of a system: middleware query, web service.
- Files are useful because they provide restart points without **requerying the source**.



Data Cleansing and Conforming

School of Information Studies
Syracuse University

Cleanse and Conform Systems

Data-Cleansing System

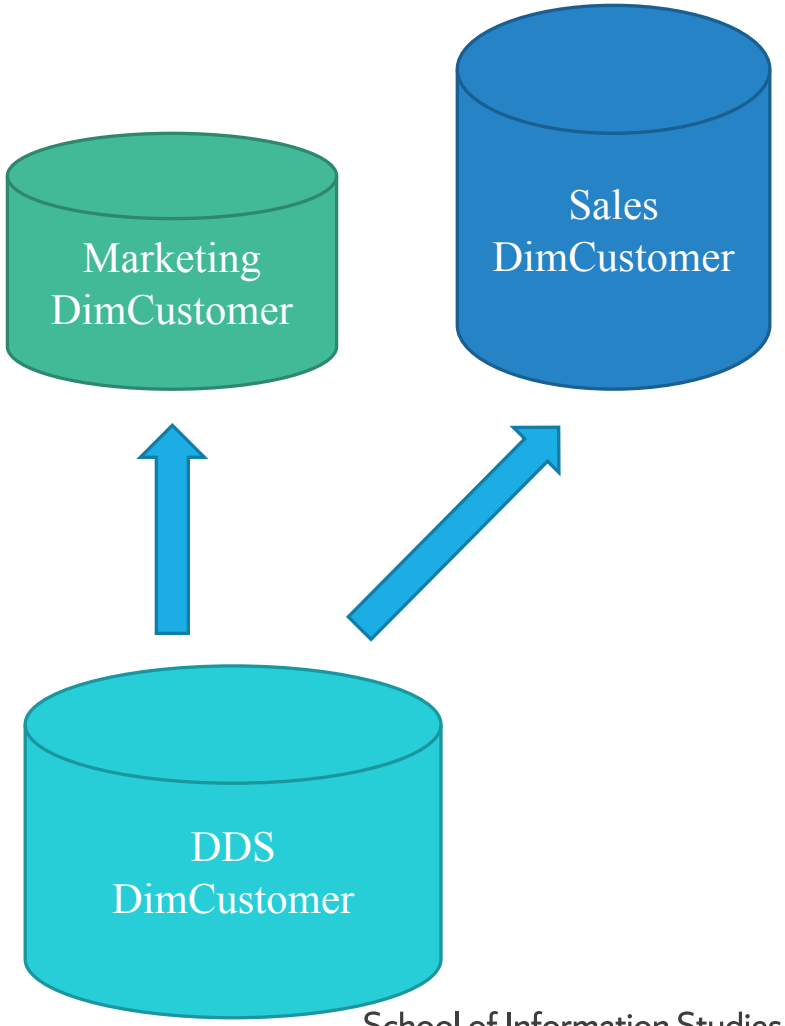
- Balance these conflicting goals:
  - Fix dirty data yet maintain data accuracy.
- Quality screens act as diagnostic filters:
  - **Column screens:** test data in fields
  - **Structure screens:** test data relationships, lookups
  - **Business rule screens:** test business logic
- Responding to quality events:
  - Fix (e.g., replace NULL w/value)
  - Log error and continue or abort (depending on severity)

Deduplication System

- When dimensions are derived from several sources
  - E.g., customer information merges from several lines of business
- **Survivorship**: the process of combining a set of matched records into unified image of authoritative data
- **Master data management:** centralized facilities to store master copies of data

Conforming System

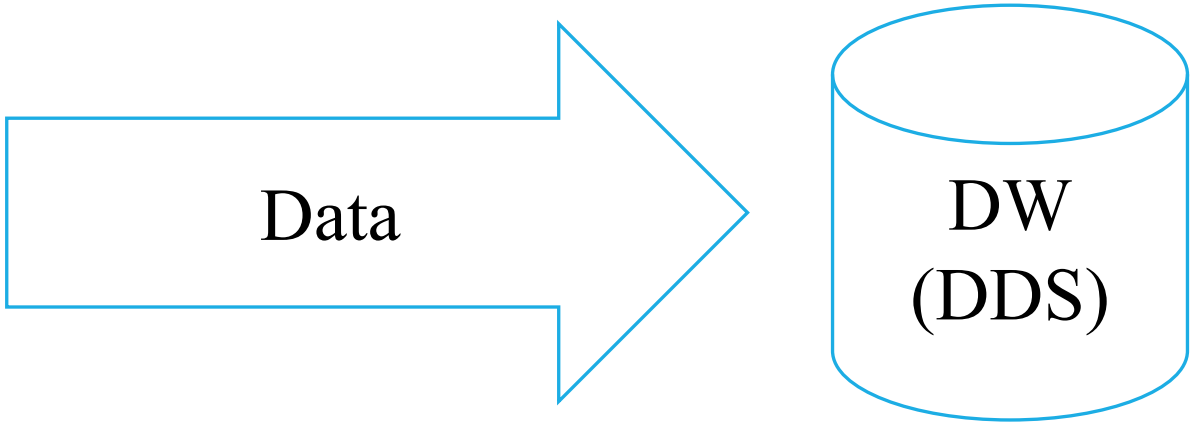
- Responsible for creating conformed dimensions and facts.
- Typically conformed dimensions are managed in one place and distributed as a copy into the required dimensional model.



Data Delivery

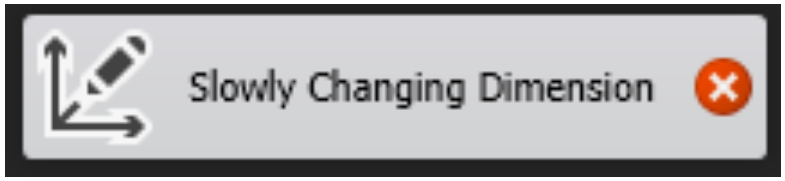
School of Information Studies
Syracuse University

Presentation Systems



Slowly Changing Dimension Manager

- The ETL system must determine how to handle a dimension attribute value that has changed from what is already in the warehouse.
  - Type 1: Overwrite
  - Type 2: New row
  - Type 3 = New column

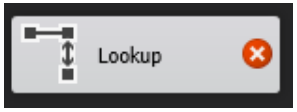


Surrogate Key Manager

- Surrogate keys are recommended for PKs of your dimension tables.
- In SQL Server, use **`IDENTITY`**.
- In other DBMSs a sequence with a database trigger can be used in place of identity.
- The ETL system can also manage them.

Surrogate Key Pipeline

- A system for replacing **operational natural keys** in the incoming **fact table record** with appropriate **dimension surrogate keys**.
- Approaches to handling **referential integrity** errors:
  - Throw away fact rows: bad idea
  - Write bad rows to an error table: most common
  - Insert placeholder row into the dimension: most complex
  - Fail the package and abort: draconian



Aggregate Builder

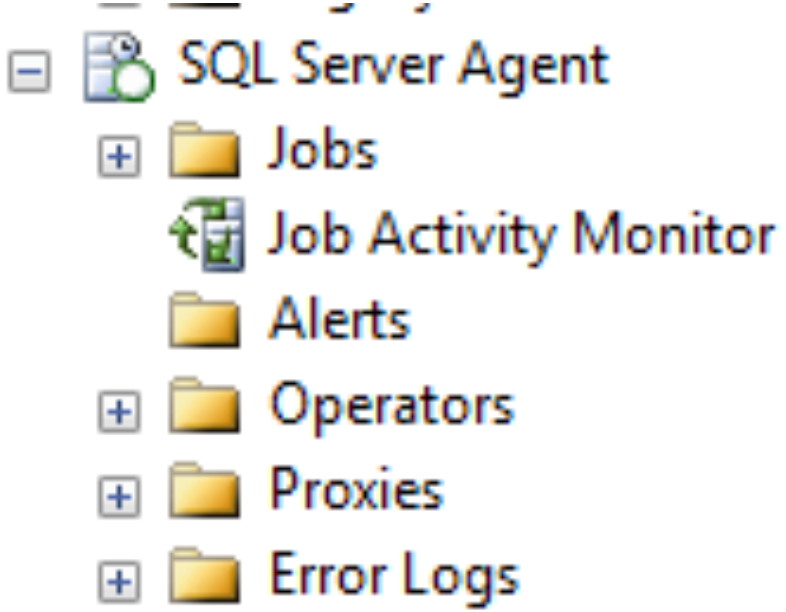
- Aggregates are specific data structures created to improve performance.
- Aggregates must be chosen carefully: Overaggregation is just as problematic as not enough.
- **Summary tables** and **subset dimensions** are generated from the base facts/dimensions.

Managing the ETL Environment

School of Information Studies
Syracuse University

Subsystems for Managing the ETL Environment

- Job Scheduler
- Recovery and Restart
- Error Logging
- Job Monitoring
- Notification and Problem Escalation



Cardinal Rules for Data Extraction

School of Information Studies
Syracuse University

Rules of OLTP DATA Extraction

Reminds me of "the rules" of Thanksgiving at my grandparents' house.

(Now that I think about it . . . any dinner at my grandparents'!)

Rules of OLTP DATA Extraction

1. Be respectful of the source.
   Extract as is. Transform on your own dime.
2. Ask for permission first.
   Despite having access, work with DBAs.
3. Don't take more data than you need.
   Be smart about the I/O you place on the source.
4. Don't take what you already have.
   If you're requerying the same data, you're being inefficient.
5. Timestamp!
   Mark the extracted data with a Timestamp column.

Extracting From a DBMS: Whole Table

School of Information Studies
Syracuse University

Extracting From DBMS

1. Whole table every time
2. Incremental by CET/LSET (current execution time/last successful execution time)
3. Incremental by OLTP surrogate key or date
4. Fixed range

Whole Table

- The entire DBMS table is extracted each time.
- Inefficient but simple.
- Sometimes the only way to extract data if there is no way to detect changes or additions.
- When to use this approach:
  - This is the approach for periodic snapshots.
  - Can be used on master data/dimensions.
  - Should not be used on transaction fact tables.

Extracting From a DBMS: Incremental

School of Information Studies
Syracuse University

Incremental CET LSET

- You extract only what you have not extracted previously.
- CET →Current extraction timestamp.
- LSET → Last successful extraction timestamp.
- CET and LSET are stored in a metadata table.
- Fault tolerant. If it fails, it can be rerun to pick up data it missed.
- When extraction is complete LSET is set to CET.
- When to use this approach:
  - For any OLTP sources that include metadata columns indicating when the row was created or last updated. Cannot be used otherwise.

Example: CET and LSET

OLTP source customers

Customer ID	Customer Name	Customer Credit	Created On	Last Update On
1001	Robin Banks	$4000	3/2/2015	7/11/2016
1002	Jean Poole	$1500	5/25/2016	7/12/2016
1003	Max Emum	$3200	7/13/2016	7/13/2016

Metadata: incr_extract

Extract ID	Source	Table Name	LSET	CET
1	fudgemart	customers	7/11/2016	7/13/2016

```
SELECT * FROM fudgemart.customers
WHERE [Created On] > LSET and [Created On] <= CET
AND [Last Update On] > LSET and [Last Update On] <= CET
```

Which row(s) will be extracted? Which row(s) were extracted already?

Incremental Other Source Column

- Use a PK, date column, or business key of the source system for incremental loads.
- Metadata table used to keep track of current extraction ID (CEID) and last successful extraction ID (LSEID).
- Fault tolerant.
- When extraction is complete, LSEID is set to CET.
- When to use this approach:
  - Works for transaction fact tables; cannot track updates to dimensions or master data this way.

Example: Incremental Other

OLTP source customers

Customer ID	Customer Name	Customer Credit
1001	Robin Banks	$4000
1002	Jean Poole	$1500
1003	Max Emum	$3200

Metadata: incr_extract

Extract ID	Source	Table Name	LSEID	CEID
1	fudgemart	customers	1001	1003

```
SELECT * FROM fudgemart.customers
WHERE [Customer ID] > LSET and [Customer ID] <= CET
```

Which row(s) will be extracted? Which row(s) were extracted already?

Fixed Range

- Useful for very large source tables that take a very long time to query.
- Extract data in batches by year or month, for example.
- A good strategy for one-time extracts such as transactions and completed accumulating snapshots with millions of rows.

From Files and Web Services

School of Information Studies
Syracuse University

Extraction From File Systems

- Not all OLTP sources are RDBMS or can be extracted via query.
- Use metadata columns in stage table to keep track of which files were processed when.

	Order No	Order Date	Order Amt	Processed Date	Processed File
o9203.dat	101425	12/14/2016	$450.02	1/1/2017	o9203.dat
	101426	12/20/2016	$380.55	1/1/2017	o9203.dat
	101427	12/30/2016	$1,968.90	1/1/2017	o9203.dat
o9203.dat	101428	1/5/2017	$192.40	2/1/2017	o9204.dat
	101429	1/12/2017	$500.25	2/1/2017	o9204.dat

Extract From Web Services or Web Scraping

- Web has a lot of useful data for the data warehouse, especially for data mining and machine learning.
  - Weather: store visits impacted by weather?
  - Product data of competitors
  - Social media: what are people saying about us?
- Custom programs to extract the data.
- Store on file system first. Web is notorious for being here today and gone tomorrow.
- Then use the file system approach.

Staging Patterns

School of Information Studies
Syracuse University

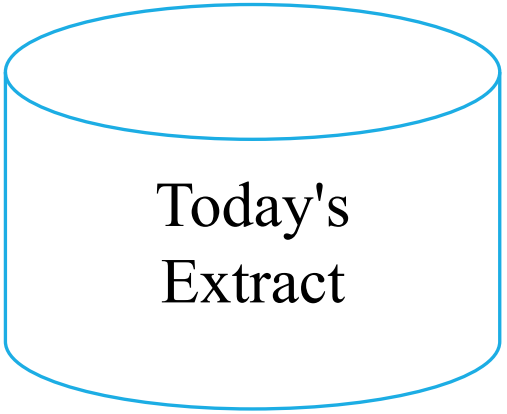
Staging Patterns

- Data extractions should be saved in stage, but how? Three common patterns:

1. Truncate and load
2. Append
3. New table each time

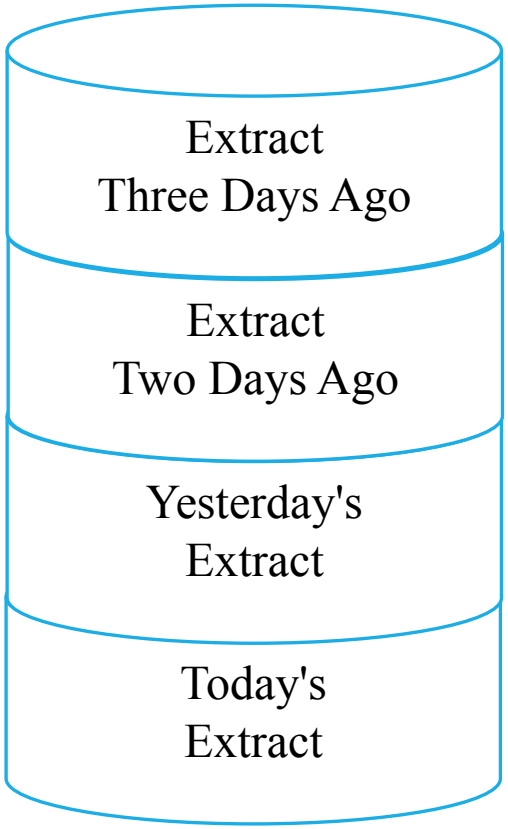
Truncate and Load

- Before extract, the previous extract is truncated.
- Easy to implement, but there is no convenient access to previous extracts.
- If you need a previous extract, you must rely on database backup.



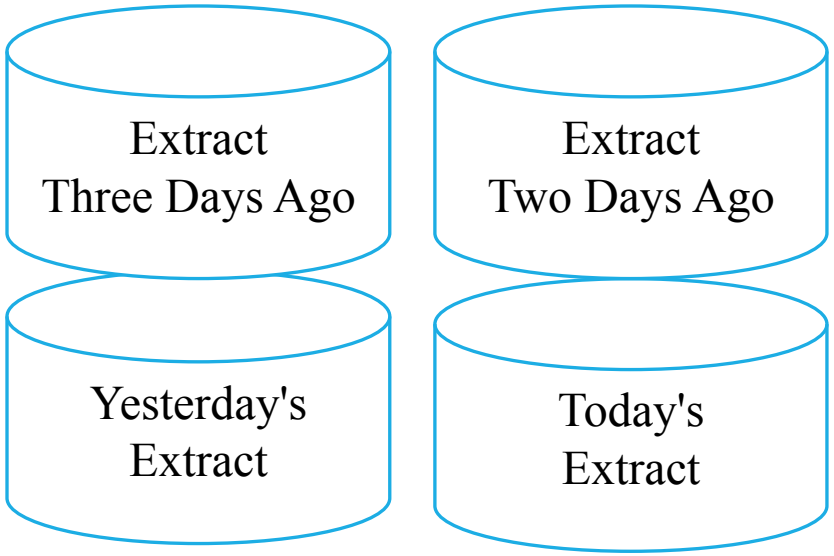
Append

- Each extract is appended to the stage table.
- Easy to implement; convenient access to previous extracts.
- Performance issues with large amounts of data. Should be partitioned by date and clustered index.



New Table Each Time

- Each extract creates a new table.
- Not as easy to implement, but convenient access to previous extracts.
- Better performance than append since row set is smaller.



Example: Snapshotting to Add Time Variance to Current Data

School of Information Studies
Syracuse University

Example: Snapshotting to Add Time Variance to Current Data

Orders Extracted on 12/6/2016

Order ID	Order Date	Order Status	Created On	Last Update On
4452	12/4/2016	Processed	12/4/2016	12/4/2016
4453	12/6/2016	Processed	12/6/2016	12/6/2016

Orders Extracted on 12/7/2016

Order ID	Order Date	Order Status	Created On	Last Update On
4452	12/4/2016	Shipped	12/4/2016	12/7/2016
4453	12/6/2016	Cancelled	12/6/2016	12/7/2016

Data in Stage

Extracted On	Order ID	Order Date	Order Status	Created On	Last Update On
12/6/2016	4452	12/4/2016	Processed	12/4/2016	12/4/2016
12/6/2016	4453	12/6/2016	Processed	12/6/2016	12/6/2016
12/7/2016	4452	12/4/2016	Shipped	12/4/2016	12/7/2016
12/7/2016	4453	12/6/2016	Cancelled	12/6/2016	12/7/2016

Truncate and Load

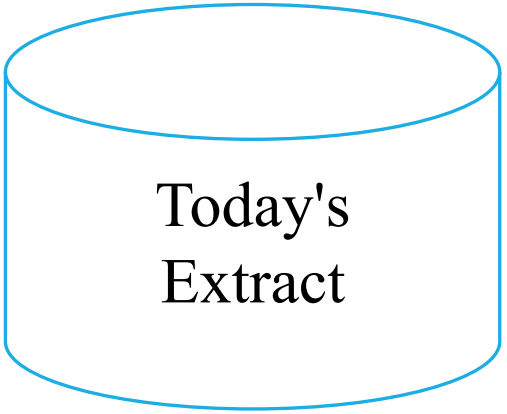
School of Information Studies
Syracuse University

Common ETL Patterns

1. Truncate and load
2. Insert if not exists
3. Upsert (Type 1 SCD)
4. Type 2 SCD

Truncate and Load

- Before extract, the previous extract is truncated.
- Easy to implement, but there is no convenient access to previous extracts.
- If you need a previous extract, you must rely on database backup.

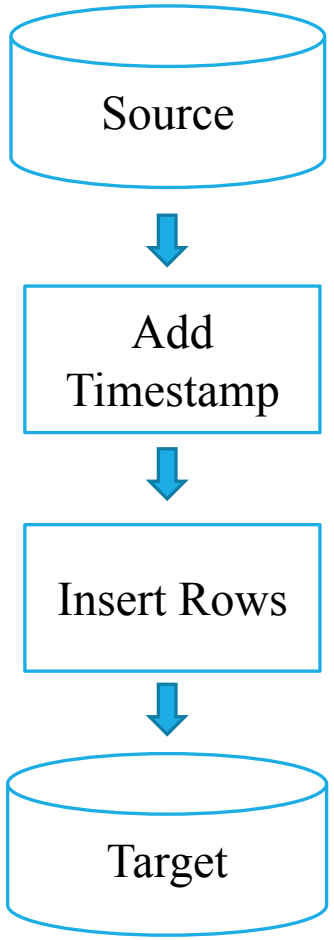


Append

School of Information Studies
Syracuse University

Append

- Simple
- Another common staging pattern
- Provides convenient access to prior extracts

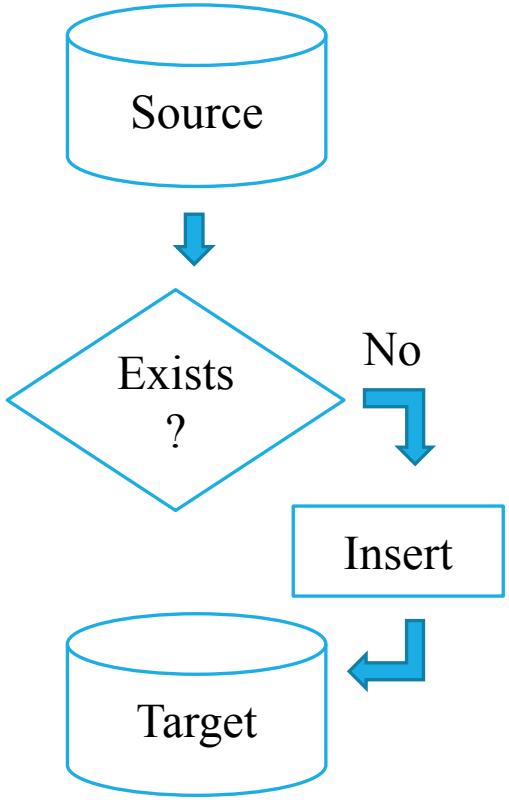


Insert If Not Exists

School of Information Studies
Syracuse University

Insert If Not Exists

- Insert if not exists. If exists, then it is ignored.
- Uses business key (natural key) to check for exist.
- Commonly used for fact table loads to ensure same fact row not readded.

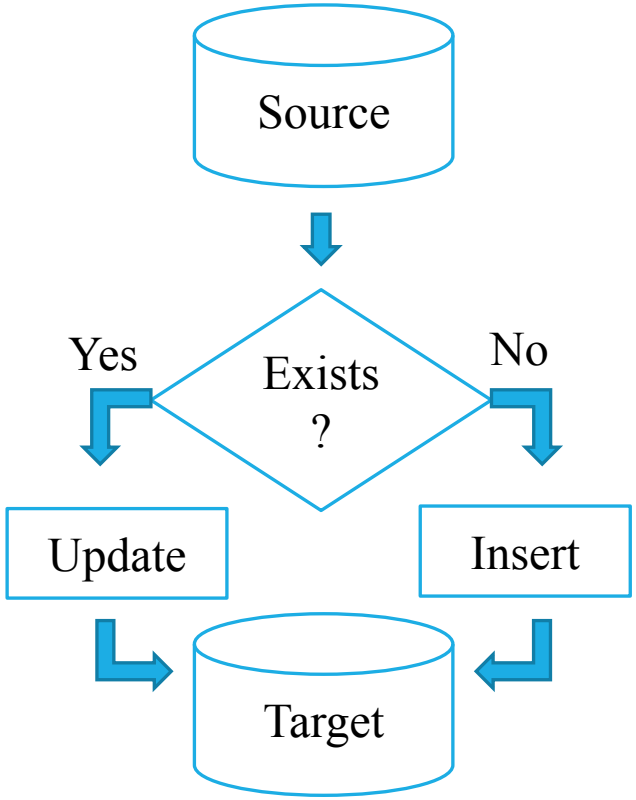


Upsert (SCD Type 1)

School of Information Studies
Syracuse University

Upsert (Type 1 SCD)

- Insert if not exists; update if exists.
- Uses business key (natural key) to check for exist.
- Type 1 SCD pattern.
- Useful for fact table loads as well.

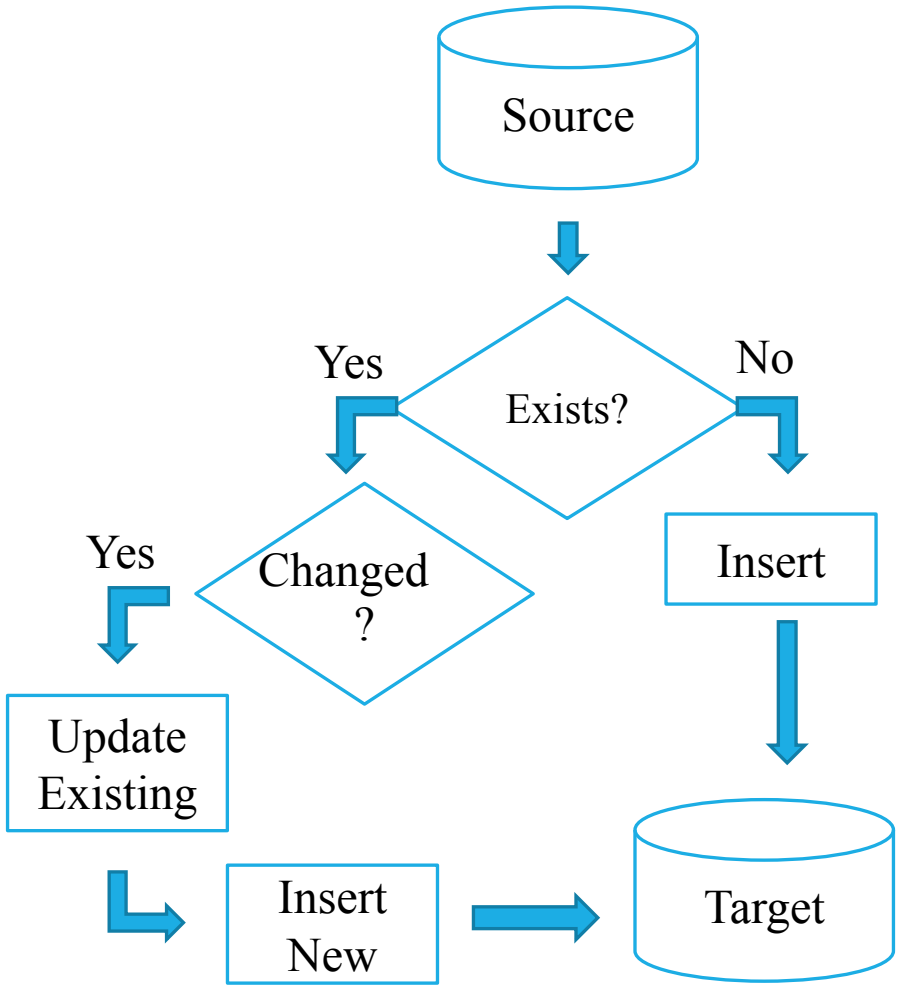


Type 2 SCD

School of Information Studies
Syracuse University

Type 2 SCD

- Preserves history by expiring existing row on change, then inserting new row.
- Uses business key (natural key) to check for exist.
- Uses metadata to determine which columns should be checked for change.
- Used for dimension processing.

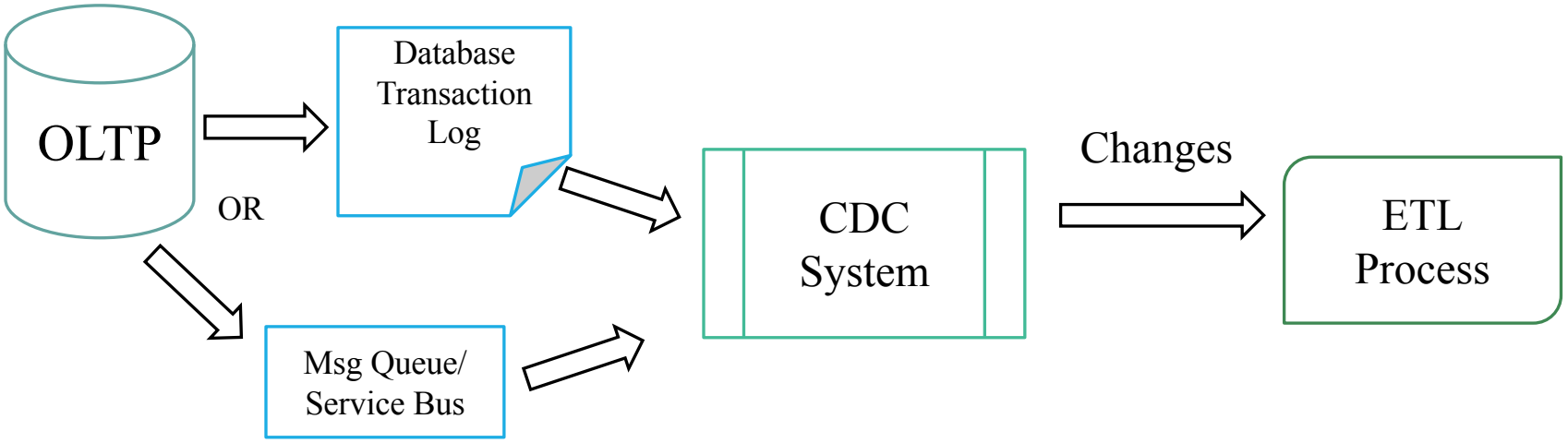


Change Data Capture

School of Information Studies
Syracuse University

CDC: Change Data Capture

- Data change events (create, update, delete) are passed to the CDC system.
- The system acts as a source for the ETL process.
- Ideal for tracking incremental changes to source data.



Late Arriving Dimensions

School of Information Studies
Syracuse University

Late-Arriving Data

- Believing all data will arrive at the same time in the data warehouse is wishful thinking.
- Why? Performance. OLTP access windows. Disparate sources of OLTP data.
- Examples:
  - Orders are updated hourly, but the dimensions that rely upon it (salesperson, customer, product) are updated weekly because they come from several sources.
  - A customer changes address while order is being processed.

Late-Arriving Dimensions

- Fact comes in with a natural/business key that is not yet in the dimension.
- Placeholder technique:
  1. Insert a new dimension row with just natural key and placeholder attributes.
  2. At a later time, dimension processing should update the dimension attributes in a Type 1 fashion.

Customer Key	Customer ID	Customer Name	Customer Credit	Placeholder
5502	1001	Robin Banks	$4000	N
5503	1002	Jean Poole	$1500	N
5504	1003	Customer TBD	$0	Y

Late-Arriving Facts

School of Information Studies
Syracuse University

Late-Arriving Facts

- Fact being inserted is old and should not be referenced by the current values in the current dimension.
- We must rely on Type 2 metadata to find the correct row in for that point in time.
- Rare, but can happen.

Order ID	Order Date	Order Status	Cust Id	Amount
4402	12/31/2015	Complete	1001	$4500

Is this order over the customer's credit limit?

Cust Key (PK)	Cust ID	Customer Name	Cust Credit	Effective Date	Expiration Date
5502	1001	Robin Banks	$4000	4/1/2015	7/15/2016
5506	1001	Robin Banks	$7500	7/15/2016	12/31/9999

Early-Arriving Facts

School of Information Studies
Syracuse University

Early-Arriving Facts

- Accumulating snapshots require fact rows to be updated.
- Fact row comes in, but not all facts known at initial write.
- Null written in place of facts; unknown members used for dimension keys.
- Updated when known.

Order ID	Order Date	Order Status	Customer Id	Amount	Shipped Date
5590	5/20/2017	Processing	1001	$4500	<NULL>

Order ID	Order Date Key	Order Status	Customer Key	Amount	Shipped Date	Days To Ship
5590	20170520	Processing	5506	$4500	-1	<NULL>

When an updated fact comes in with shipped date, we update this row, replacing the unknown member and calculating the days to ship.