Slides credit: Matthias Boehm

SCIENCE
PASSION
TECHNOLOGY

# Data Integration and Large Scale Analysis
## 02 Data Warehousing and ETL

**Shafaq Siddiqi**

Graz University of Technology, Austria



Last update: Oct 13, 2023

ISDS

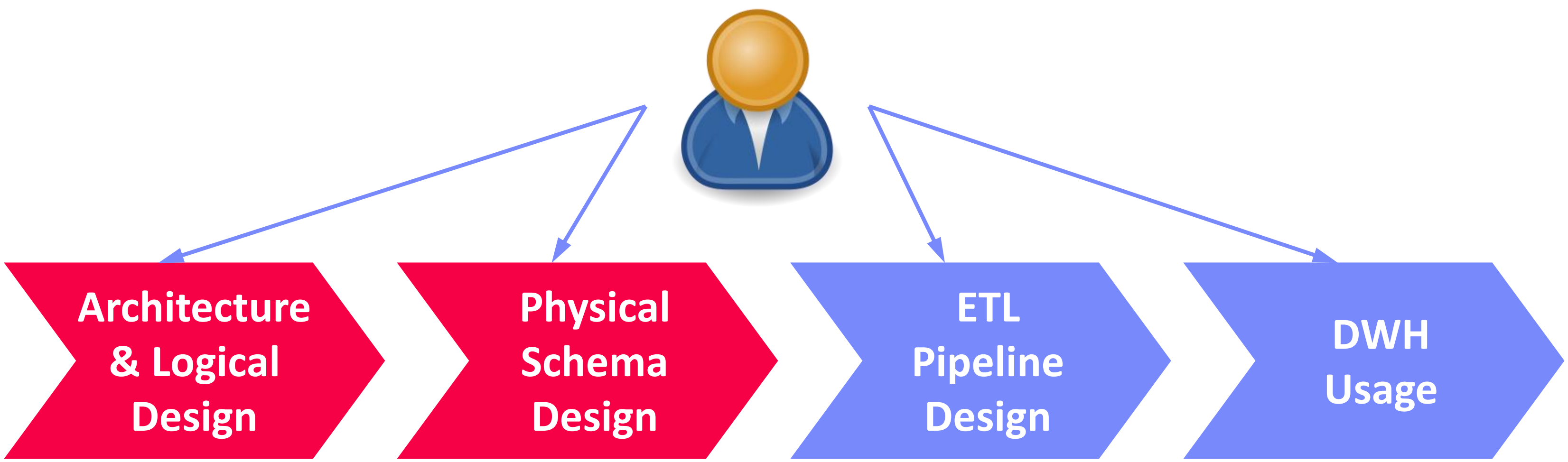
# Announcements/Org

- **#1 Video Recording**
  - Link in **TUbe** & **TeachCenter**
  - Optional attendance
  - In-person and video-recorded lectures
    - **HS i5** or Webex: https://tugraz.webex.com/meet/shafaq.siddiqi

- WKO Research Grants
  - https://www.tugraz.at/en/research/research-at-tu-graz/services-fuer-forschende/foerderprogramme-und-preise-an-der-tu-graz/#c87088
  - Submission deadline October 20, 2023

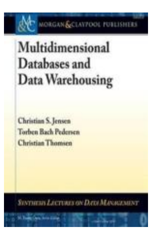
# Agenda

- **Data Warehousing (DWH)**
- **Extraction, Transformation, Loading (ETL)**
- **SQL/OLAP Extensions**

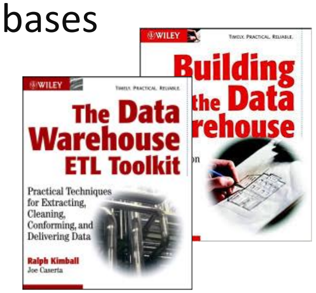



SQL, SQL/OLAP,
Data Mining,
Machine Learning

# Data Warehousing


1. [**Wolfgang Lehner:** Datenbanktechnologie für Data-Warehouse-Systeme. Konzepte und Methoden, Dpunkt Verlag, 1-373, 2003]
2. [**C. S. Jensen, T. B. Pedersen, C. Thomsen.** Multidimensional Databases and Data Warehousing. Morgan and Claypool Publishers. 2010]



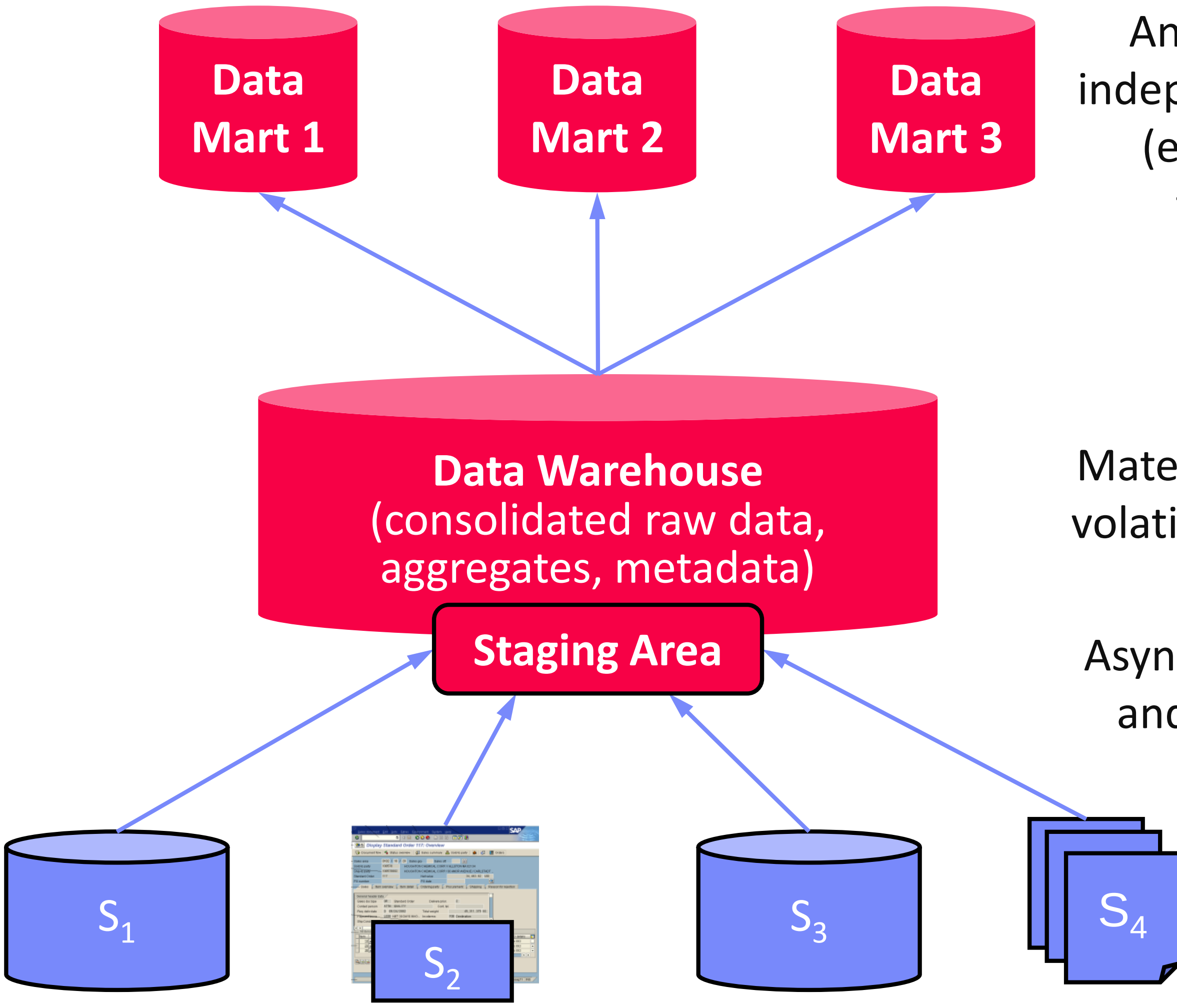
# Motivation and Tradeoffs

- **Goal:** Queries over consolidated and cleaned data of several, potentially heterogeneous, data sources

? **OLTP** (Online Transaction Processing) **vs OLAP** (Online Analytical Processing)


- **Tradeoffs**
  - **Analytical query performance:** write vs read optimized data stores
  - **Virtualization:** overhead of remote access, source systems affected
  - **Consistency:** sync vs async changes, time regime → up-to-date?
  - **Others:** history, **flexibility**, **redundancy**, effort for **data exchange**

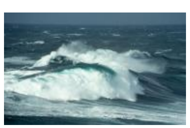
# Data Warehouse Architecture




Analysis-centric independent subsets (e.g., geo, org, functional)

Materialized, non-volatile integration

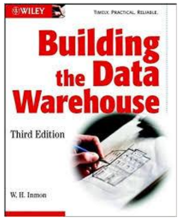
Async replication, and ETL vs ELT



Operational source systems

# Data Warehouse Architecture, cont.

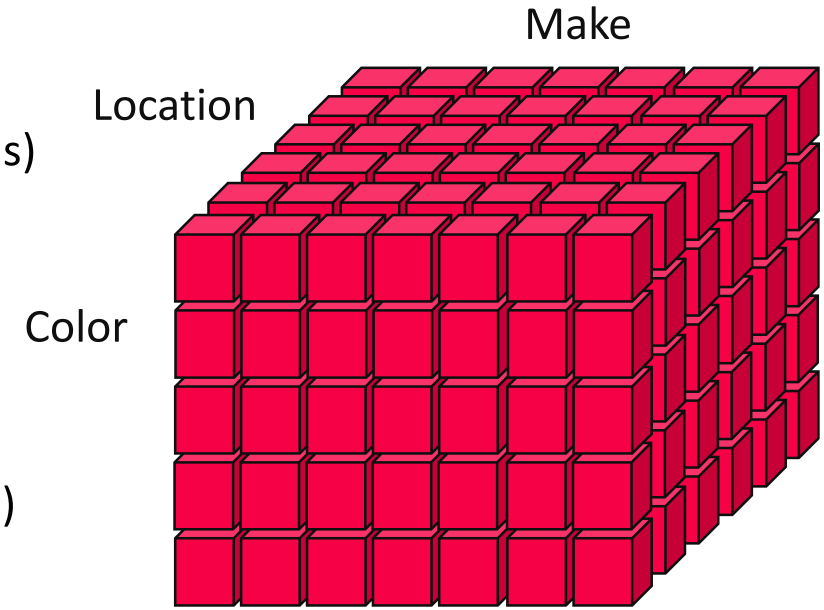
- **Data Warehouse (DWH)**
  - "A *data warehouse is a* ***subject-oriented, integrated, time-varying, non-volatile*** *collection of data in support of the management's decision-making process."* (Bill Inmon)
  - **#1 Subject-oriented:** analysis-centric organization (e.g., sales) → Data Mart
  - **#2 Integrated:** consistent data from different data sources
  - **#3 Time-varying:** History (snapshots of sources), and temporal modelling
  - **#4 Non-volatile:** Read-only access, limited to periodic data loading by admin




- **Different DWH Instantiations**
  - **Single DWH system** with virtual/materialized views for data marts
  - Separate systems for consolidated DWH and aggregates/data marts (**dependent data marts**)
  - Data-Mart-local staging areas and ETL (**independent data marts**)

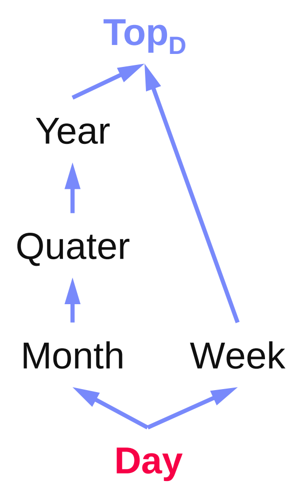
# Multi-dimensional Modeling: Data Cube

- **Central Metaphor: Data Cube**
  - Qualifying data (categories, dimensions)
  - Quantifying data (cells)
  - Often sparse (0 for empty cells)



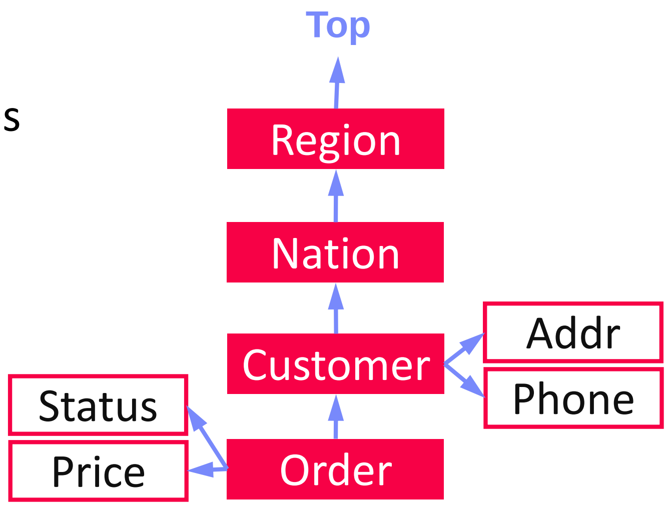

- **Multi-dimensional Schema**
  - Set of **dimension hierarchies** ($D^1$,…, $D^n$)
  - Set of **measures** ($M^1$,…,$M^m$)

- **Dimension Hierarchy**
  - Partially-ordered set D of categorical attributes ($\{D_1,\ldots,D_n, Top_D\};\rightarrow$)
  - Generic **maximum element**
    $\forall i(1 \leq i \leq n): D_i \rightarrow Top_D$
  - Existing **minimum element** (primary attribute)
    $\exists i(1 \leq i \leq n)\forall j(1 \leq i \leq n, i \neq j): D_i \rightarrow D_j$



# Multi-dimensional Modeling: Data Cube, cont.

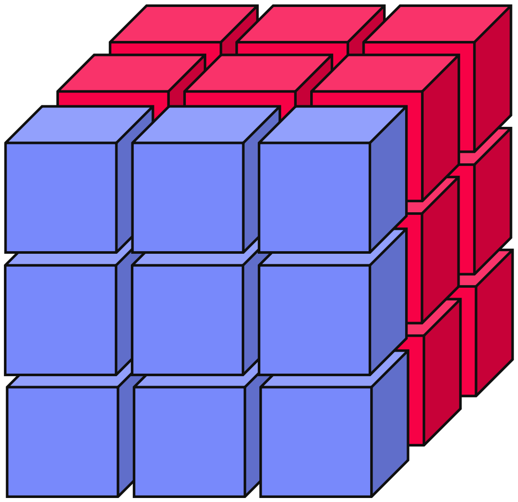
- **Dimension Hierarchy, cont.**
  - Classifying (categorical) vs descriptive attributes
  - **Orthogonal dimensions:** there are no functional dependencies between attributes of different dimensions




- **Fact F**
  - Base tuples w/ measures of summation type
  - Granularity G as subset of categorical attributes
- **Measure M**
  - Computation function over non-empty subset of facts $f(F_1, \ldots, F_k)$ in schema
  - Scalar function vs aggregation function
  - Granularity G as subset of categorical attributes

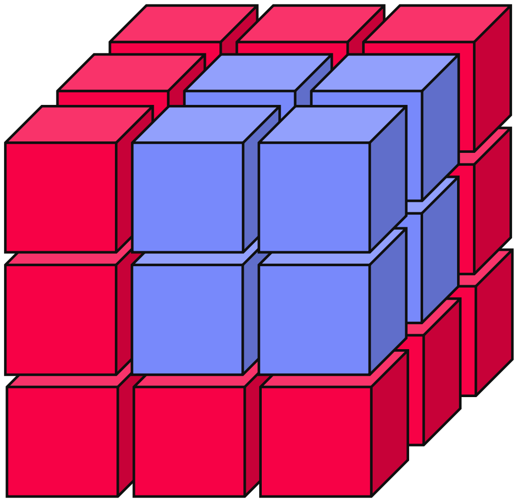
# Multi-dimensional Modeling: Operations

- **Slicing**
  - Select a "slice" of the cube by specifying a filter condition on one of the dimensions (categorical attributes)
  - Same data granularity but subset of dimensions



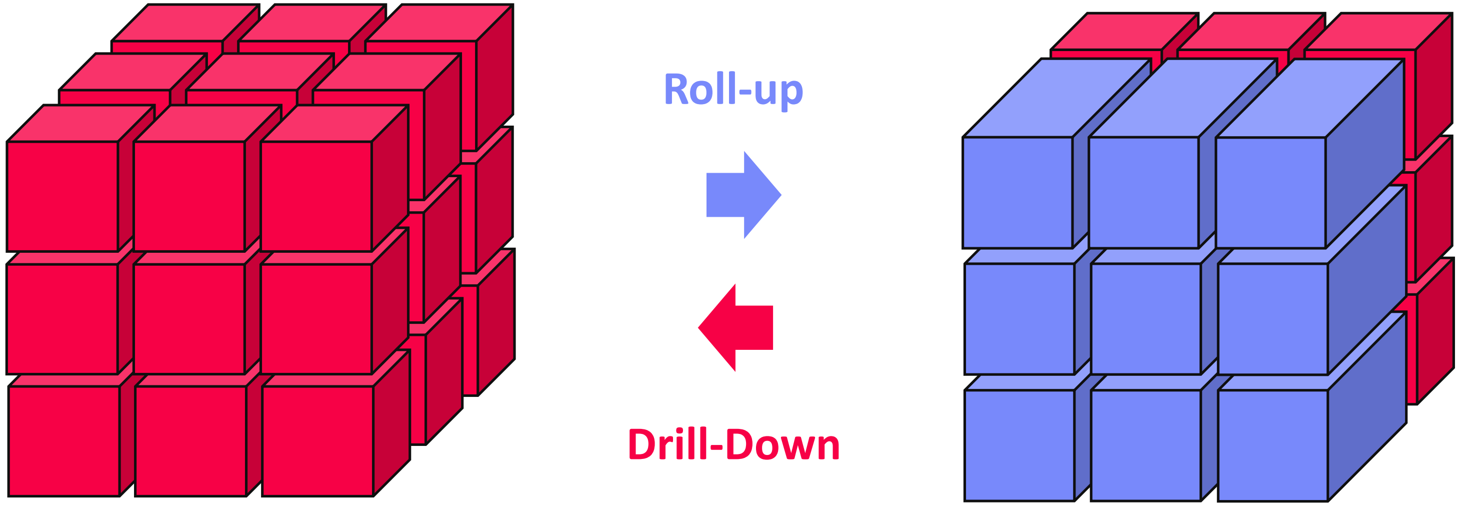
- **Dicing**
  - Select a "sub-cube" by specifying a filter condition on multiple dimensions
  - Complex Boolean expressions possible
  - Sometimes slicing used synonym

**Example:** `Location=Graz` **AND** `Color=White` **AND** `Make=BMW`



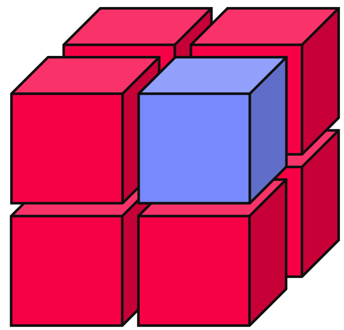
# Multi-dimensional Modeling: Operations, cont.

- **Roll-up** (similar Merge - remove dim)
  - Aggregation of facts or measures into coarser-grained aggregates (measures)
  - Same dimensions but different granularity
- **Drill-Down** (similar Split add dim)
  - Disaggregation of measures into finer-grained measures



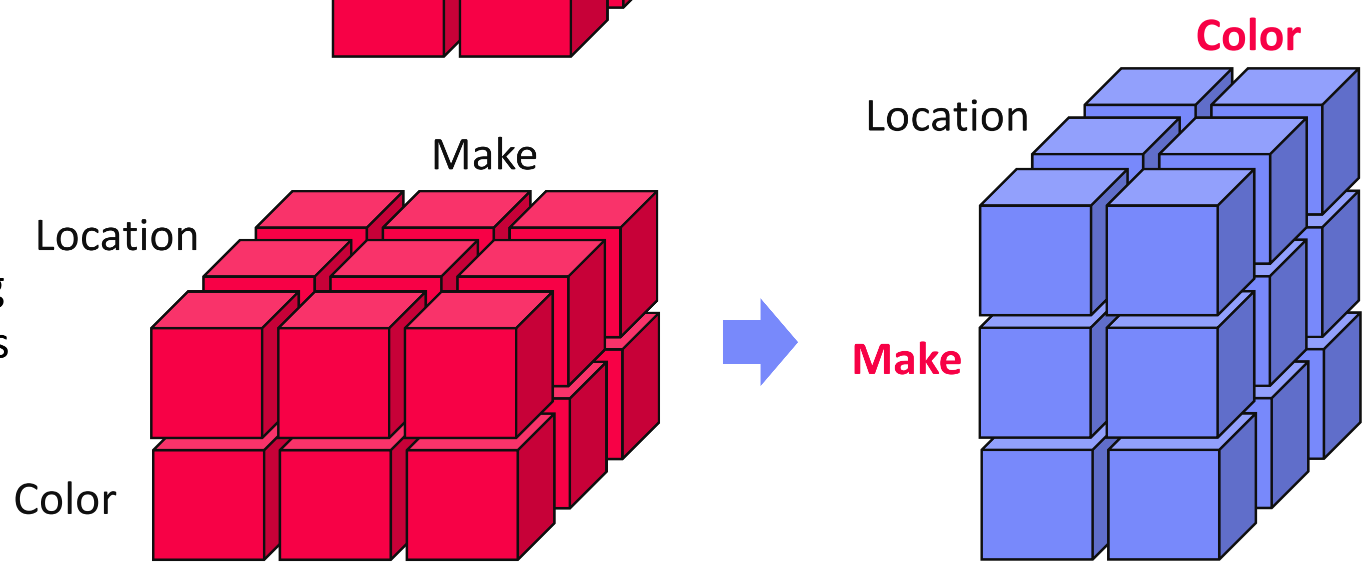
# Multi-dimensional Modeling: Operations, cont.

- **Drill-Across**
  - Change from one cube to another
- **Drill-Through**
  - Drill-Down to smallest granularity of underlying data store (e.g., RDBMS)
  - E.g., find relational tuples



| FName | LName | Local | Make | Color |
|---|---|---|---|---|
| Matthias | Boehm | Graz | BMW | White |
| … | … | … | … | … |

- **Pivot**
  - Rotate cube by exchanging dimensions



# Aggregation Types

- **Recap: Classification of Aggregates**
  - **Additive** aggregation functions (SUM, COUNT)
  - **Semi-additive** aggregation functions (MIN, MAX)
  - **Additively computable** aggregation functions (AVG, STDDEV, VAR)
  - Aggregation functions (MEDIAN, QUANTILES)

- **Summation Types of Measures**
  - **FLOW:** arbitrary aggregation possible
  - **STOCK:** aggregation possible, except over temporal dim
  - **VPU:** value-per-unit typically (e.g., price)

[Hans-Joachim Lenz, Arie Shoshani: Summarizability in OLAP and Statistical Data Bases. SSDBM 1997]

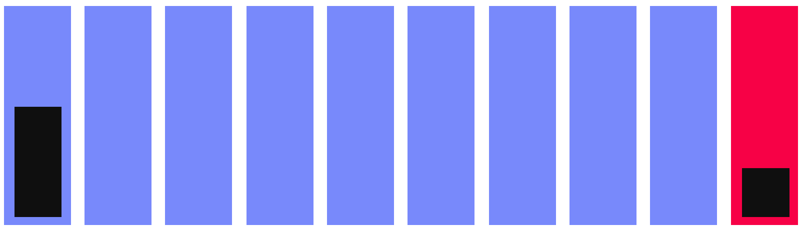
- **Necessary Conditions**
  - Disjoint attribute values
  - Completeness
  - Type compatibility

[TUGraz online]

| # Stud | 16/17 | 17/18 | 18/19 | 19/20 | 20/21 | Total |
|---|---|---|---|---|---|---|
| **CS** | 1,153 | 1,283 | 1,321 | 1,343 | 1368 | ? |
| **SEM** | 928 | 970 | 939 | 944 | 985 | ? |
| **ICE** | 804 | 868 | 846 | 842 | 849 | ? |
| **Total** | 2,885 | 3,121 | 3,106 | 3,129 | 3,202 | ? |

# Excursus: Other Misleading Statistics

- **Problem Setting**
  - 100 people (**90 vaccinated**, **10 non-vaccinated**)
  - **5** infected vaccinated, **2** infected non-vaccinated




[https://twitter.com/howie_hua/status/1421502809862664197]

- **`P(vacc|infected) = 5/7 = 0.71 → misleading`**
- **`P(infected|vacc) = 5/90 = 0.056`**
- **`P(infected|non-vacc) = 2/10 = 0.2`**

[see also Simpson's Paradox in **06 Data Cleaning**]

# Aggregation Types, cont.

- **Additivity**

| | FLOW | STOCK: Temporal Agg? | | VPU |
|---|---|---|---|---|
| | | Yes | No | |
| MIN/MAX | ✓ | ✓ | | ✓ |
| SUM | ✓ | ✗ | ✓ | ✗ |
| AVG | ✓ | ✓ | | ✓ |
| COUNT | ✓ | ✓ | | ✓ |

- **Type Compatibility** (addition/ subtraction)

| | FLOW | STOCK | VPU |
|---|---|---|---|
| **FLOW** | FLOW | STOCK | ✗ |
| **STOCK** | | STOCK | ✗ |
| **VPU** | | | VPU |

# Data Cube Mapping and MDX

- **MOLAP (Multi-Dim. OLAP)**
  - OLAP server with native multi-dimensional data storage
  - Dedicated query language: Multidimensional Expressions (MDX)
  - E.g., IBM Cognos Powerplay, Essbase

[https://docs.microsoft.com/en-us/analysis-services/multidimensional-models/mdx]

```
SELECT
  {[Measures].[Sales],
   [Measures].[Tax]} ON COLUMNS,
  {[Date].[Fiscal].[Year].&[2002],
   [Date].[Fiscal].[Year].&[2003] } ON ROWS
FROM [Adventure Works]
WHERE ([Sales Territory].[Southwest])
```

- **ROLAP (Relation OLAP)**
  - OLAP server w/ storage in RDBMS
  - E.g., all commercial RDBMS vendors
- **HOLAP (Hybrid OLAP)**
  - OLAP server w/ storage in RDBMS and multi-dimensional in-memory caches and data structures

**Requires mapping to relational model**

[**Example systems**: https://en.wikipedia.org/wiki/Comparison_of_OLAP_servers]

# Recap: Relational Data Model

- **Domain D (value domain): e.g., Set S, INT, Char[20]**
- **Relation R**
  - **Relation schema** RS:
    Set of k attributes $\{A_1,\dots,A_k\}$
  - **Attribute** $A_j$: value domain $D_j = \text{dom}(A_j)$
  - **Relation:** subset of the Cartesian product over all value domains $D_j$
    $\mathbf{R \subseteq D_1 \times D_2 \times \dots \times D_k, k \geq 1}$

Attribute

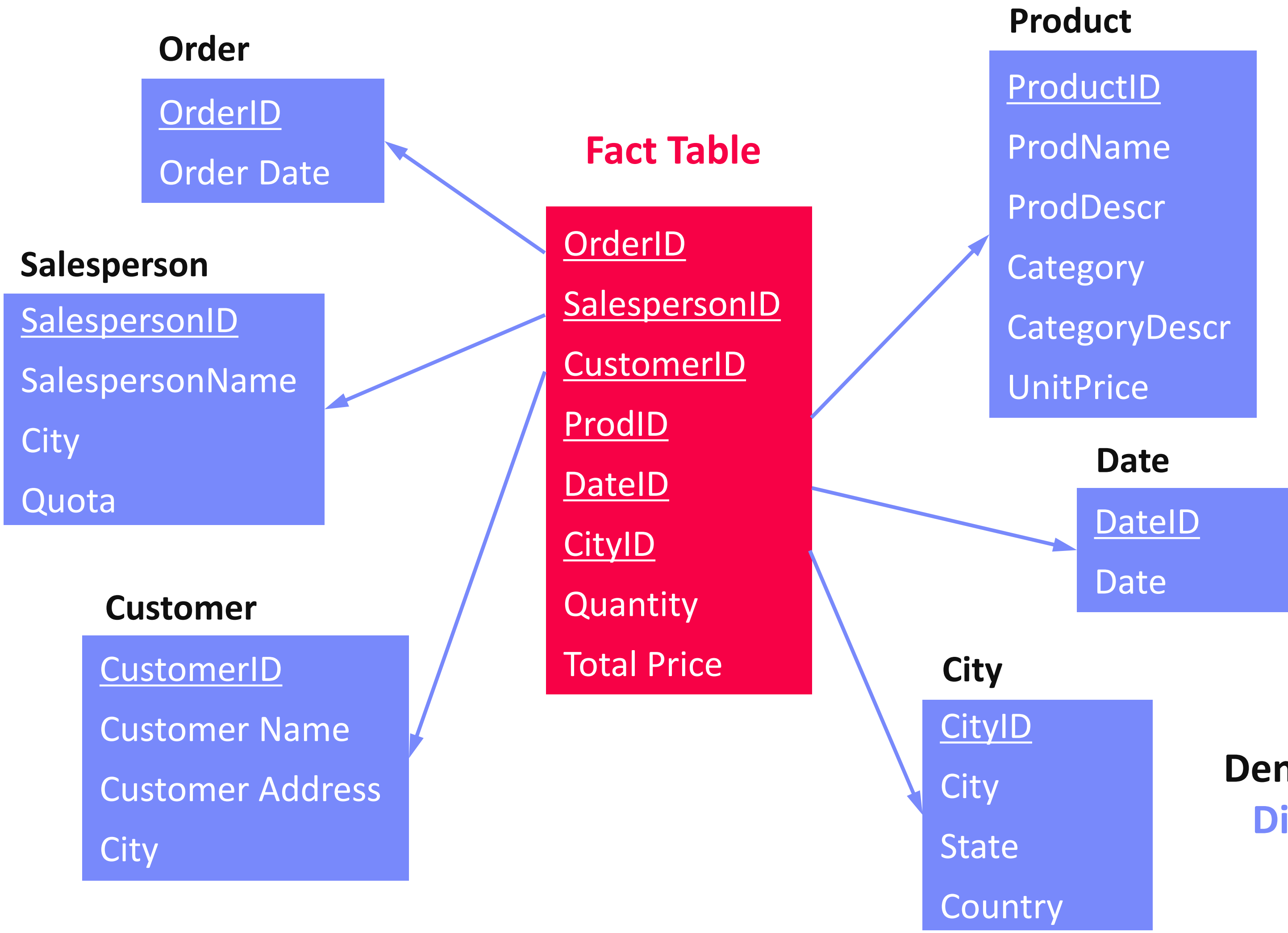
| A1 INT | A2 INT | A3 BOOL |
|---|---|---|
| 3 | 7 | T |
| 1 | 2 | T |
| 3 | 4 | F |
| 1 | 7 | T |

Tuple

cardinality: 4
rank: 3

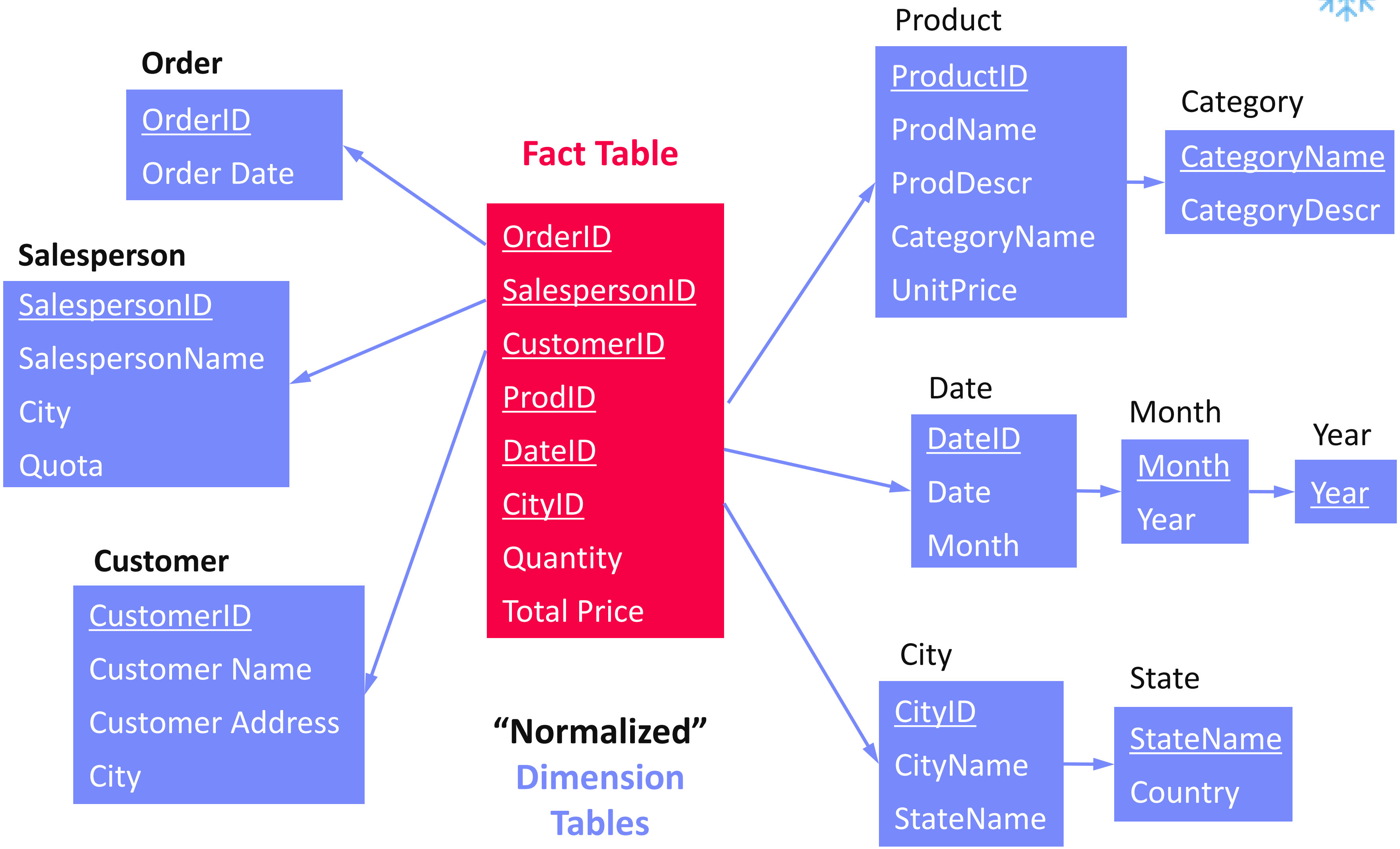
- **Additional Terminology**
  - **Tuple**: row of k elements of a relation
  - **Cardinality** of a relation: number of tuples in the relation
  - **Rank** of a relation: number of attributes
  - **Semantics:** **Set** := no duplicate tuples (in practice: **Bag** := duplicates allowed)
  - **Order of tuples and attributes is irrelevant**

# ROLAP – Star Schema



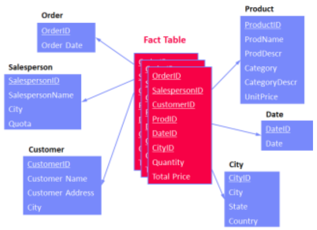

**Denormalized Dimension Tables**

# ROLAP – Snowflake Schema



# ROLAP – Other Schemas

- **Galaxy Schema**
  - Similar to star-schema but with **multiple fact tables** and potentially shared dimension tables
  - Multiple stars → Galaxy




- **Snow-Storm Schema**
  - Similar to snow-flake-schema but with **multiple fact tables** and potentially shared dimension tables
  - Multiple snow flakes → snow storm

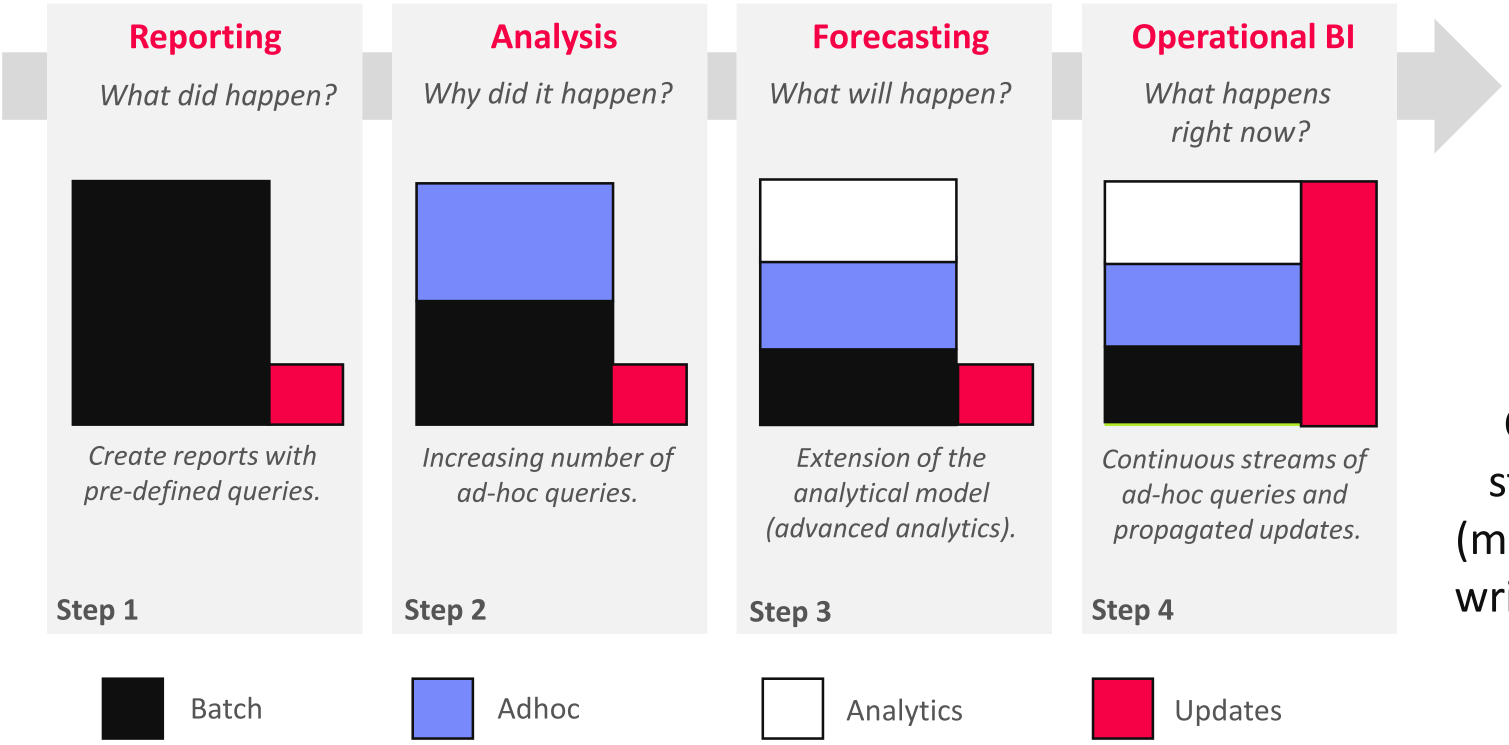
- **OLAP Benchmark Schemas**
  - **TPC-H** (8 tables, normalized schema)
  - **SSB** (5 tables, star schema, simplified TPC-H)
  - **TPC-DS** (24 tables, snow-storm schema)

  *"TPC-D and its successors, TPC-H and TPC-R assumed a 3rd Normal Form (3NF) schema. However, over the years the industry has expanded towards star schema approaches."*

[Raghunath Othayoth Nambiar, Meikel Poess: The Making of TPC- DS. **VLDB 2006**]

# Evolution of DWH/OLAP Workloads

- **Goals: Advanced analytics and Operational BI**

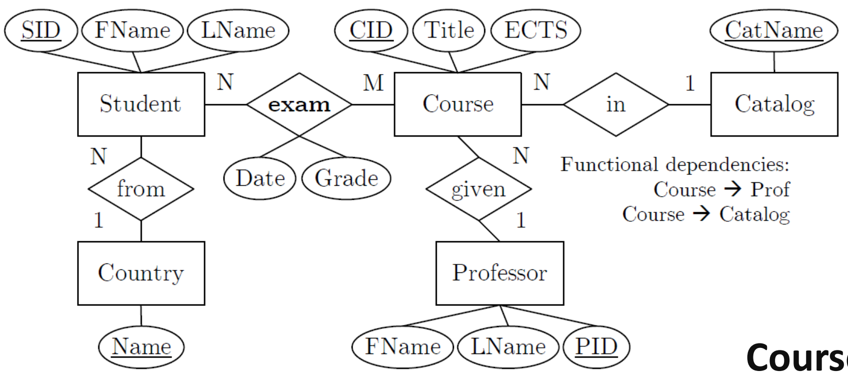


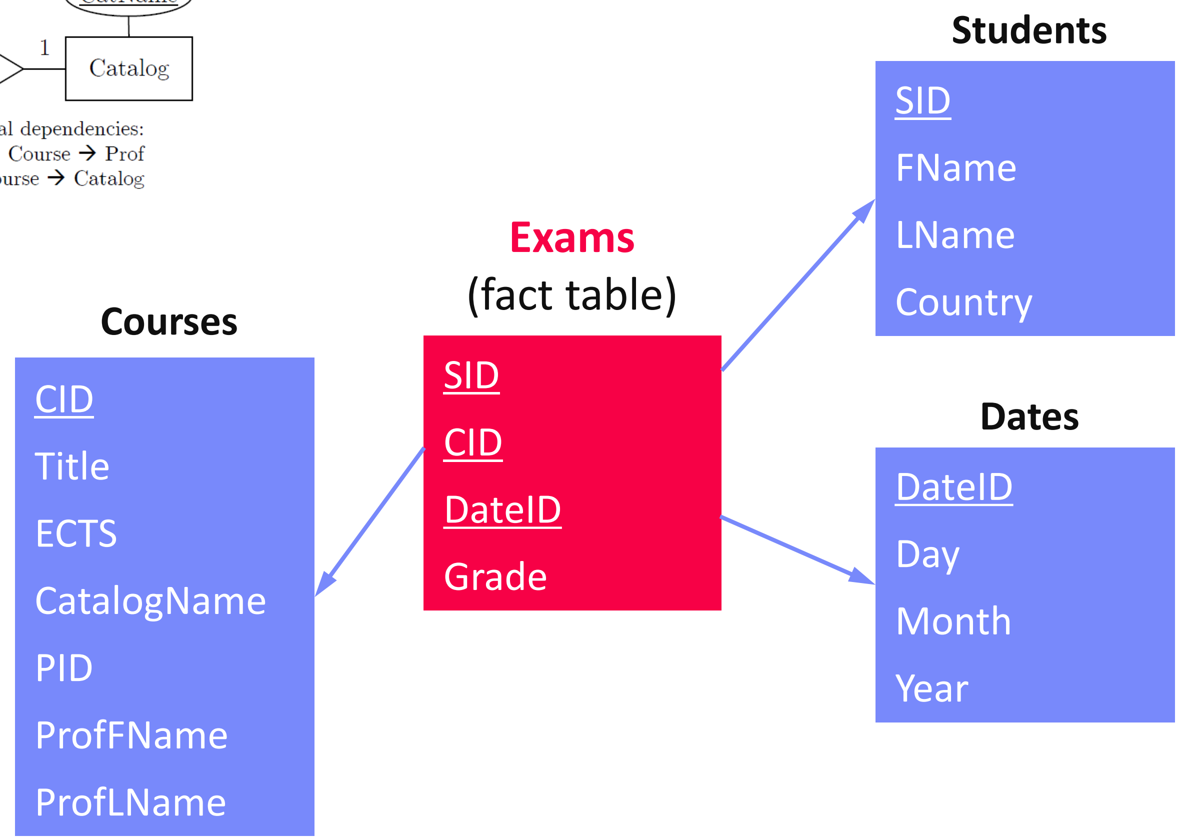

**SotA:** Column stores w/ (multi-stage) write buffers

# BREAK (and Test Yourself)

**[Exam Feb 08, 2021]**

- **Task: Given below ER diagram, create a ROLAP star schema. Data types can be ignored, but indicate PK and FK constraints. (9/100 points)**



# Extraction, Transformation, Loading (ETL)

# Extract-Transform-Load (ETL) Overview

- **Overview**
  - ETL process refers to the overall process of obtaining data from the source systems, cleaning and transforming it, and loading it into the DWH
  - Subsumes many integration and cleaning techniques

- **#1 ETL**
  - Extract data from heterogeneous sources
  - Transform data via dedicated data flows or in staging area
  - Load cleaned and transformed data into DWH

- **#2 ELT**
  - Extract data from heterogeneous sources
  - Load raw data directly into DWH
  - Perform data transformations inside the DWH via SQL

  → allows for **automatic optimization of execution plans**

# Types of Heterogeneity

[J. Hammer, M. Stonebraker, and O. Topsakal: THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches. U Florida, TR05-001, **2005**]

- **Heterogeneity**
  - **Semantic Heterogeneity**
    - **Attribute Heterogeneity**
      1. Synonyms/Homonyms (Instructor vs lecturer)
      2. Simple Mapping (mathematical 12 hr vs 12 hr)
      3. Union Types (String, urls)
      4. Complex Mappings (Units/Credits)
      5. Language Expressions (Databse vs Datenbank)
    - **Missing Data**
      6. Nulls (Missing Values)
      7. Virtual Columns
      8. Semantic Incompatibility (Concept does not exist)
  - **Structural Heterogeneity**
    9. Same Attribute in different structure
    10. Handling Sets
    11. Attribute name does not define semantics
    12. Attribute composition

# Corrupted Data

- **Heterogeneity of Data Sources**
  - Update anomalies on denormalized data / eventual consistency
  - Changes of app/preprocessing over time (US vs us) → inconsistencies
- **Human Error**
  - Errors in semi-manual data collection, laziness (see default values), bias
  - Errors in data labeling (especially if large-scale: crowd workers / users)
- **Measurement/Processing Errors**
  - Unreliable HW/SW and measurement equipment (e.g., batteries)
  - Harsh environments (temperature, movement) → aging

**Uniqueness & duplicates** | **Contradictions & wrong values** | **Missing Values** | **Ref. Integrity**

[**Credit:** Felix Naumann]

| <u>ID</u> | Name | BDay | Age | Sex | Phone | Zip |
|---|---|---|---|---|---|---|
| 3 | Smith, Jane | 05/06/1975 | 44 | F | 999-9999 | 98120 |
| 3 | John Smith | 38/12/1963 | 55 | M | 867-4511 | 11111 |
| 7 | Jane Smith | 05/06/1975 | 24 | F | 567-3211 | 98120 |

| Zip | City |
|---|---|
| 98120 | San Jose |
| 90001 | Lost Angeles |

**Typos**

# ETL – Planning and Design Phase

- **Architecture, Flows, and Schemas**
  - #1 Plan requirements, architecture, tools
  - #2 Design high-level integration flows (systems, integration jobs)
  - #3 Data understanding (copy/code books, meta data)
  - #4 Design dimension loading (static, dynamic incl keys)
  - #5 Design fact table loading
- **Data Integration and Cleaning**
  - #5 Types of data sources (snapshot, APIs, query language, logs)
  - #6 Prepare schema mappings → see **04 Schema Matching and Mapping**
  - #7 Change data capture and incremental loading (diff, aggregates)
  - #8 Transformations, enrichments, and deduplication → **05 Entity Linking**
  - #9 Data validation and cleansing → see **06 Data Cleaning and Data Fusion**
- **Optimization**
  - #10 Partitioning schemes for loaded data (e.g., per month)
  - #11 Materialized views and incremental maintenance

# Events and Change Data Capture

- **Goal: Monitoring operations of data sources for detecting changes**

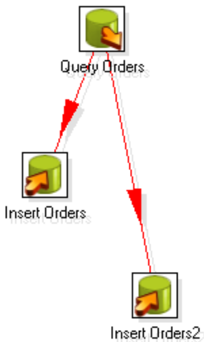
- **#1 Explicit Messages/Triggers**
  - Setup update propagation from the source systems to middleware
  - Asynchronously propagate the updates into the DWH

- **#2 Log-based Capture**
  - Parse system logs / provenance to retrieve changes since last loading
  - Sometimes combined w/ replication → **03 MoM, EAI, and Replication**
  - Leverage explicit audit columns or internal timestamps

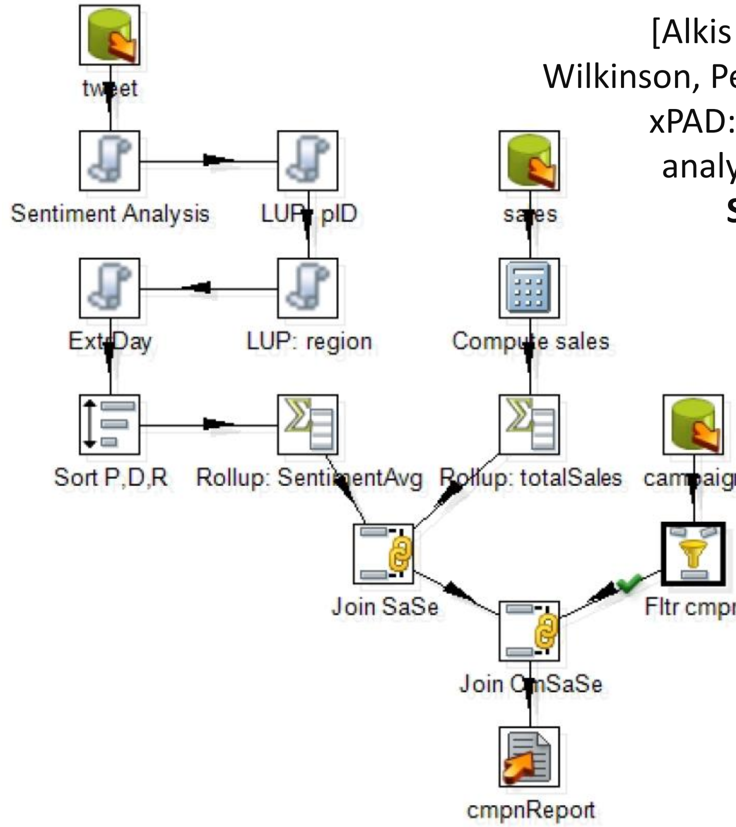
- **#3 Snapshot Differences**
  - Compute difference between old and new snapshot (e.g., files) before loading
  - Broadly applicable but more expensive

# Example ETL Flow

- **Example Flows** (Pentaho Data Integration, since 2015 Hitachi)




[Matthias Boehm, Uwe Wloka, Dirk Habich, Wolfgang Lehner: GCIP: exploiting the generation and optimization of integration processes. **EDBT 2009**]




[Alkis Simitsis, Kevin Wilkinson, Petar Jovanovic: xPAD: a platform for analytic data flows. **SIGMOD 2013**]

- **Other Tools**
  - IBM IS, Informatica, SAP BO, MS Integration Services
  - Open Source: Pentaho Data Integration, Scriptella ETL, CloverETL, Talend

# ETL via Apache Spark

- **Example**
  - Distributed ETL pipeline processing

[Xiao Li: Building Robust ETL Pipelines with Apache Spark, **Spark Summit 2017**]


```
//load csv and postgres tables
val csvTable = spark.read.csv("/source/path")
val jdbcTable = spark.read.format("jdbc")
  .option("url", "jdbc:postgresql:...")
  .option("dbtable", "TEST.PEOPLE")
  .load()

//join tables, filter and write as parquet
csvTable
  .join(jdbcTable, Seq("name"), "outer")
  .filter("id <= 2999")
  .write.mode("overwrite")
  .format("parquet")
  .saveAsTable("outputTableName")
```

**11 Distributed, Data-Parallel Computation**

# SQL/OLAP Extensions

# Overview Multi-Groupings

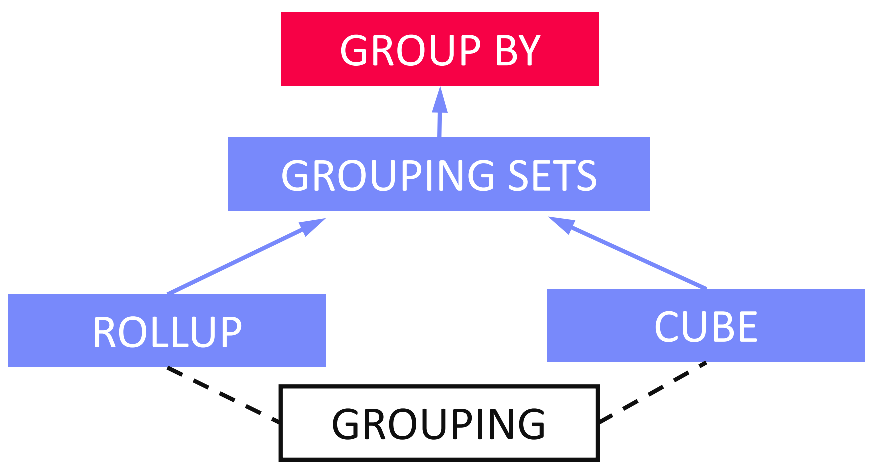
- **Recap: GROUP BY**
  - Group tuples by categorical variables
  - Aggregate per group

| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

```
SELECT Year, SUM(Revenue)
  FROM Sales
  GROUP BY Year
```

| Year | SUM |
|---|---|
| 2004 | 60 |
| 2005 | 30 |

- **Grouping Extensions**



# Grouping Sets

```
GROUP BY GROUPING SETS
  ((<attribute-list>), …)
```

- **Semantics**
  - Grouping by multiple group-by attribute lists w/ consistent agg function
  - Equivalent to multiple GROUP BY, connected by UNION ALL

- **Example**

```
SELECT Year, Quarter, SUM(Revenue)
  FROM R
  GROUP BY GROUPING SETS
    ((), (Year), (Year,Quarter))
```

| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

→

| Year | Quarter | SUM |
|---|---|---|
| - | - | 90 |
| 2004 | - | 60 |
| 2005 | - | 30 |
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

# Rollup (see also multi-dim ops)

```
GROUP BY ROLLUP
  (<attribute-list>)
```

- **Semantics**
  - Hierarchical grouping along dimension hierarchy
  - `GROUP BY ROLLUP (A1,A2,A3)`
    `:= GROUP BY GROUPING SETS((),(A1),(A1,A2),(A1,A2,A3))`
- **Example**

```
SELECT Year, Quarter, SUM(Revenue)
  FROM R
  GROUP BY ROLLUP(Year,Quarter)
```

| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

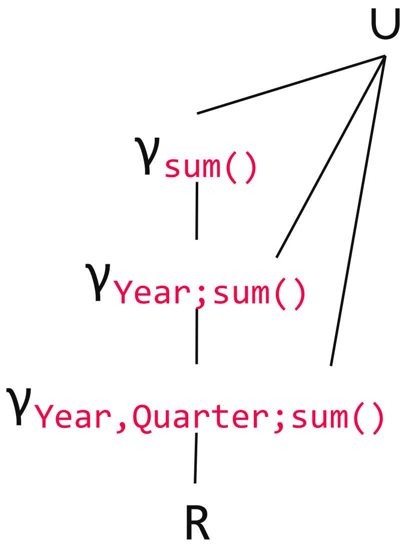
→

| Year | Quarter | SUM |
|---|---|---|
| - | - | 90 |
| 2004 | - | 60 |
| 2005 | - | 30 |
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

# Rollup, cont. and Grouping

- **Operator Implementation**
  - Aggregation towers for (semi-)additive aggregation functions
  - Example

```
SELECT Year, Quarter, SUM(Revenue)
  FROM R
  GROUP BY ROLLUP(Year,Quarter)
```




- **GROUPING Semantics**
  - With ROLLUP or CUBE to identify aggregates
  - NULL group vs NULL due to aggregation
  - Example

```
SELECT Team, SUM(Revenue),
    GROUPING(Team) AS Agg
  FROM R
  GROUP BY ROLLUP (Team)
```

| Team | Revenue | Agg |
|---|---|---|
| **NULL** | 10 | 0 |
| Sales | 40 | 0 |
| Tech | 20 | 0 |
| **NULL** | 70 | 1 |

# Cube

`GROUP BY CUBE(<attribute-list>)`

- **Semantics**
  - Computes aggregate for all $2^n$ combinations for n grouping attributes
  - Equivalent to enumeration via `GROUPING SETS`
- **Example**

```
SELECT Year, Quarter, SUM(Revenue)
  FROM R
  GROUP BY CUBE(Year,Quarter)
```

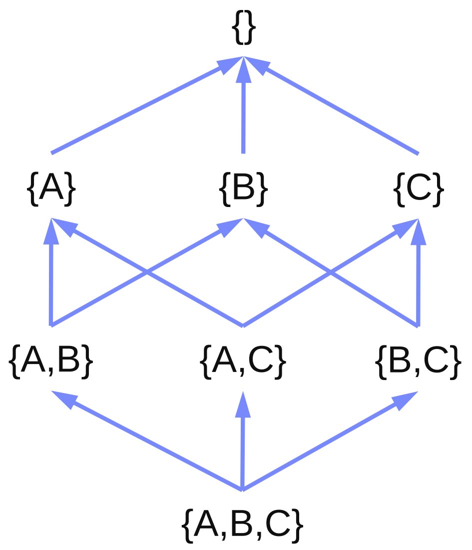
| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

→

| Year | Quarter | SUM |
|---|---|---|
| - | - | 90 |
| 2004 | - | 60 |
| 2005 | - | 30 |
| - | 1 | 40 |
| - | 2 | 20 |
| - | 3 | 10 |
| - | 4 | 20 |
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

# Cube, cont.

- **Operator Implementation**
  - Aggregation lattice for (semi-)additive aggregation functions
  - **But: multiple alternative paths**
    → how to select the cheapest?



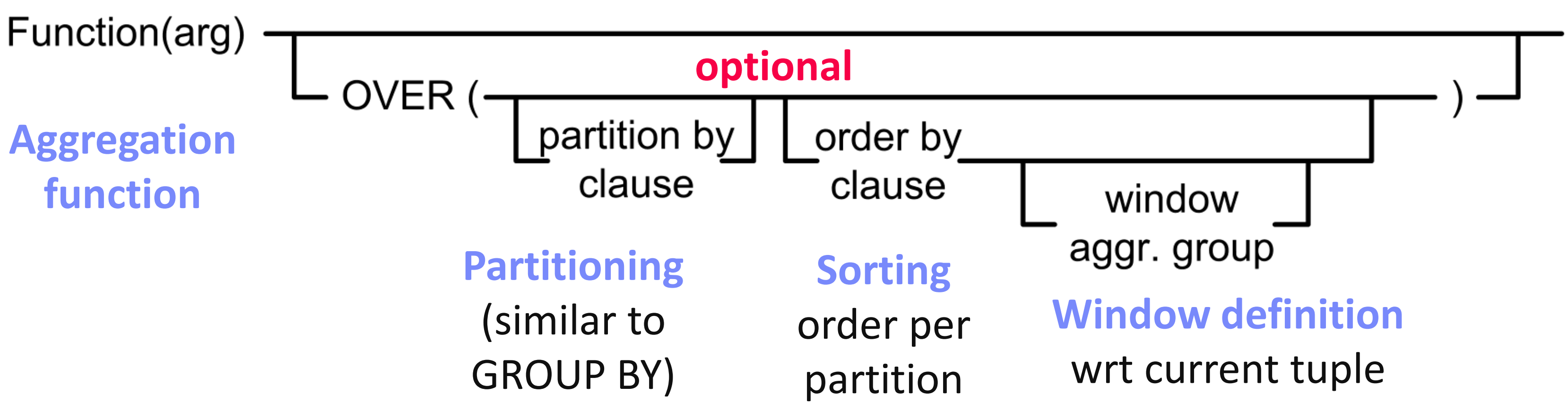

- **Recap: Physical Group-By Operators**
  - SortGroupBy / -Aggregate
  - HashGroupBy / -Aggregate

- **Cube Implementation Strategies**
  - #1: Some operators can share sorted order (e.g., {A,B} -> {A})
  - #2: Subsets with different cardinality → pick smallest intermediates

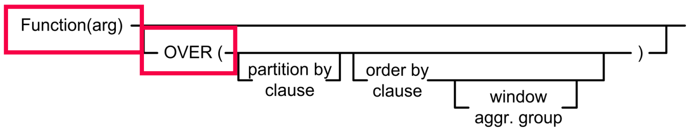
# Overview Reporting Functions

- **Motivation and Problem**
  - Scalar functions as well as grouping + aggregation
  - For many advanced use cases **not flexible enough**

- **Reporting Functions**
  - Separate partitioning (grouping) and aggregation via OVER
  - Allows local partitioning via windows and ranking/numbering



# RF – Aggregation Function




- **Semantics**
  - Operates over window and returns value for every tuple
  - `RANK(), DENSE_RANK(), PERCENT_RANK(), CUME_DIST(), ROW_NUMBER()`
- **Example**

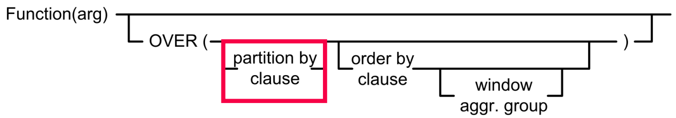
```
SELECT Year, Quarter,
    RANK() OVER (ORDER BY Revenue ASC) AS Rank1,
    DENSE_RANK() OVER (ORDER BY Revenue ASC) AS DRank1,
  FROM R
```

| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

OVER() represents all tuples

| Year | Quarter | Rank1 | DRank1 |
|---|---|---|---|
| 2004 | 1 | 1 | 1 |
| 2004 | 3 | 1 | 1 |
| 2004 | 2 | 3 | 2 |
| 2004 | 4 | 3 | 2 |
| 2005 | 1 | 5 | 3 |

# RF – Partitioning



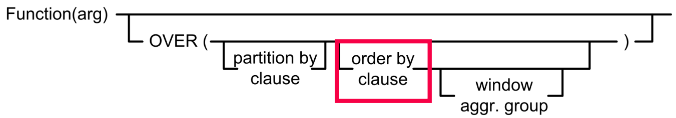

- **Semantics**
  - Select tuples for aggregation via **`PARTITON BY`** `<attribute-list>`
- **Example**

```
SELECT Year, Quarter, Revenue,
    SUM(Revenue) OVER(PARTITION BY Year)
  FROM R
```

| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

| Year | Quarter | Revenue | SUM |
|---|---|---|---|
| 2004 | 1 | 10 | 60 |
| 2004 | 2 | 20 | 60 |
| 2004 | 3 | 10 | 60 |
| 2004 | 4 | 20 | 60 |
| 2005 | 1 | 30 | 30 |

# RF – Partition Sorting



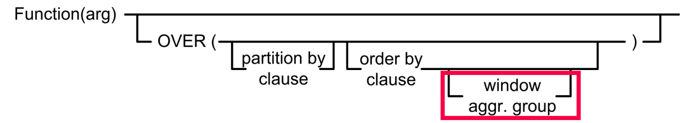

- **Semantics**
  - Define computation per partition via **ORDER BY** <attribute-list>
  - Note: ORDER BY allows cumulative computation → cumsum()
- **Example**

```
SELECT Year, Quarter, Revenue,
    SUM(Revenue) OVER(PARTITION BY Year ORDER BY Quarter)
  FROM R
```

| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

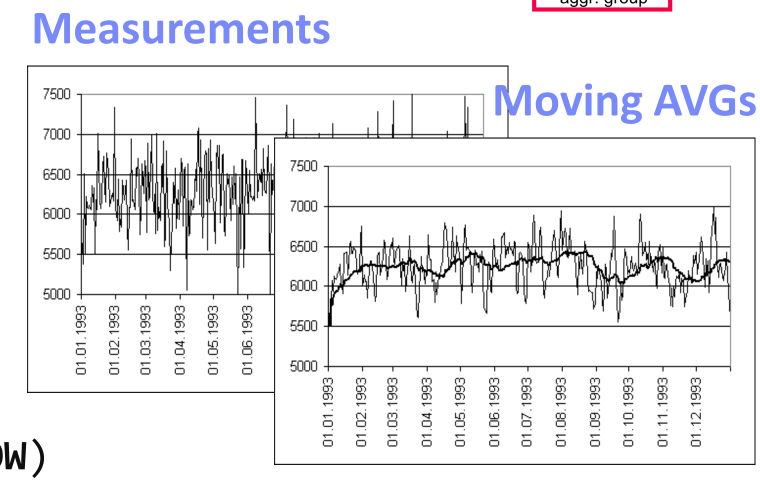
| Year | Quarter | Revenue | SUM |
|---|---|---|---|
| 2004 | 1 | 10 | 10 |
| 2004 | 2 | 20 | 30 |
| 2004 | 3 | 10 | 40 |
| 2004 | 4 | 20 | 60 |
| 2005 | 1 | 30 | 30 |

# RF – Windowing




- **Semantics**
  - Define window for computation (e.g., for moving average, cumsum)
- **Example**

```
SELECT Year, Quarter, Revenue, AVG(Revenue)
  OVER (ORDER BY Year, Quarter
    ROWS BETWEEN 1 PRECEDING AND CURRENT ROW)
  FROM R
```




| Year | Quarter | Revenue |
|---|---|---|
| 2004 | 1 | 10 |
| 2004 | 2 | 20 |
| 2004 | 3 | 10 |
| 2004 | 4 | 20 |
| 2005 | 1 | 30 |

| Year | Quarter | Revenue | AVG |
|---|---|---|---|
| 2004 | 1 | 10 | 10 |
| 2004 | 2 | 20 | 15 |
| 2004 | 3 | 10 | 15 |
| 2004 | 4 | 20 | 15 |
| 2005 | 1 | 30 | 25 |

# Trend: Cloud Data Warehousing

**10 Distributed Data Storage**

- **#1 Google Big Query**

[Google, Kazunori Sato: An Inside Look at Google BigQuery, Google **White Paper 2012**]

- **#2 Amazon Redshift**

[Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, Vidhya Srinivasan: Amazon Redshift and the Case for Simpler Data Warehouses. **SIGMOD 2015**]

- **#3 Microsoft Azure Data Warehouse**

- **#4 IBM BlueMix dashDB**

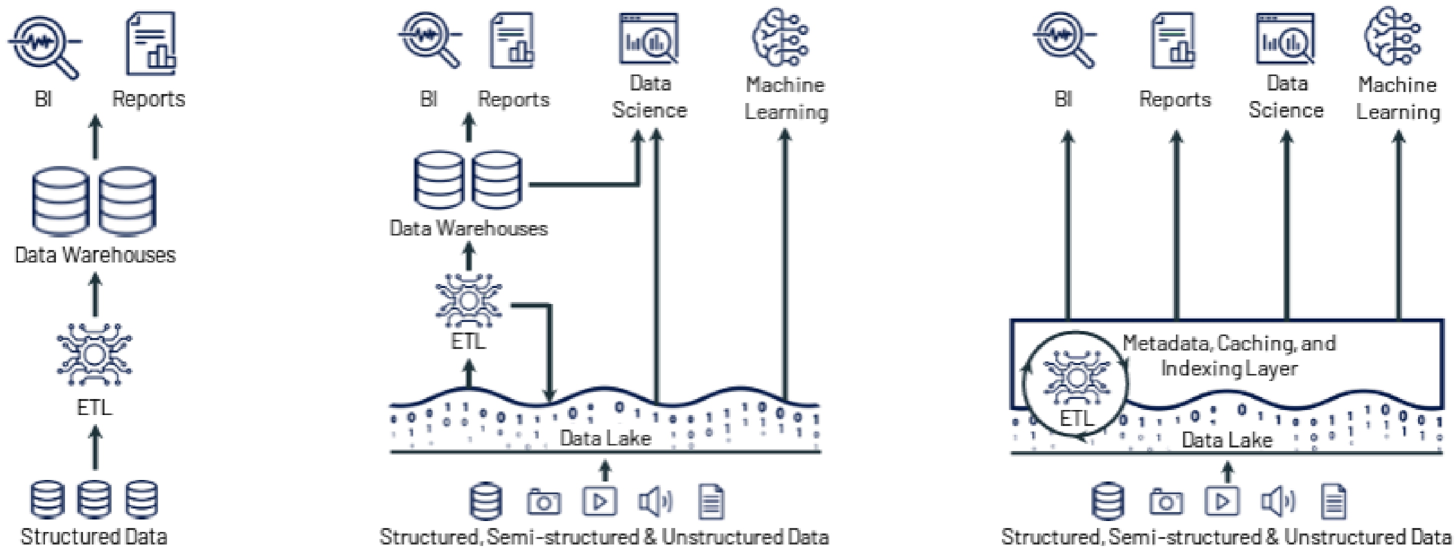
[IBM: IBM dashDB - Cloud-based data warehousing as-a-service, built for analytics, IBM **White Paper 2015**]

- **#5 Snowflake Data Warehouse**

[Benoît Dageville et al.: The Snowflake Elastic Data Warehouse. **SIGMOD 2016**]

# Trend: Data Lakes and Lakehouse

**10 Distributed Data Storage**

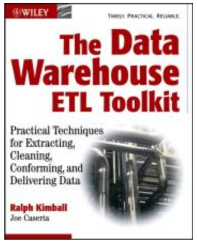



(a) First-generation platforms. (b) Current two-tier architectures. (c) Lakehouse platforms.

[Matei Zahari et. al, Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. **CIDR 2021**]

[Alkis Simitsis et. al., The History, Present, and Future of ETL Technology, **DOLAP 2023]**

# Summary and Q&A




- **Data Warehousing (DWH)**
  - DWH architecture
  - Multidimensional modeling
- **Extraction, Transformation, Loading (ETL)**
  - ETL process, errors, and data flows
- **SQL/OLAP Extensions**
  - Multi-grouping operations
  - Reporting functions

"There is a profound cultural assumption in the business world that *if only we could see all of our data, we could manage our businesses more effectively*. This cultural assumption is so deeply rooted that we take it for granted. Yet this is the mission of the data warehouse, and this is why the data warehouse is a permanent entity […] even as it morphs and changes its shape."
-- Ralph Kimball, Joe Caserta; **2004**

- **Next Lectures (Data Integration Architectures)**
  - **03 Message-oriented Middleware, EAI, and Replication** [Oct 20]
  - **04 Schema Matching and Mapping** [Oct 27]
  - **05 Entity Linking and Deduplication** [Nov 03]
  - **06 Data Cleaning and Data Fusion** [Nov 10]