Data Integration and Analysis
02 Data Warehousing and ETL

Matthias Boehm
Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMVIT endowed chair for Data Management

Last update: Oct 11, 2019
Announcements/Org

- #1 Video Recording
  - Link in TeachCenter & TUbefiles (lectures will be public)
  - Since 2nd lecture

- #2 Study Abroad Info
  - Oct 17, 10am @ Inffeldgasse
  - Internships, master theses, study courses, summer programs

- #3 Workshop - Focus on FAIR
  - Nov 7, 9.30am – 4pm (all day)
  - Student reach-out BS, MS, PhD
  - Invited speakers from EGI, TU Delft, Uni. Vienna, Uni. Barcelona, UCL, and TU Graz/Know-Center
Agenda

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

[Wolfgang Lehner: Datenbanktechnologie für Data-Warehouse-Systeme. Konzepte und Methoden, Dpunkt Verlag, 1-373, 2003]
Motivation and Tradeoffs

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

- **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

Data Warehouse
(consolidated raw data, aggregates, metadata)

Data Mart 1

Data Mart 2

Data Mart 3

Staging Area

S1

S2

S3

S4

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 \((\{D_1, ..., D_n, \text{Top}_D\}; \rightarrow)\)
  - Generic maximum element
    \[ \forall i (1 \leq i \leq n): D_i \rightarrow \text{Top}_D \]
  - 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 \( f(F_1, ..., F_k) \) 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**
  - Navigate to neighboring cells at same granularity (changed selection)

- **Drill-Through**
  - Drill-Down to smallest granularity of underlying data store (e.g., RDBMS)
  - E.g., find relational tuples

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

---

<table>
<thead>
<tr>
<th>FName</th>
<th>LName</th>
<th>Local</th>
<th>Make</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthias</td>
<td>Boehm</td>
<td>Graz</td>
<td>BMW</td>
<td>White</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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)

- **Beware Summation Types of Measures**
  - Disjointness, completeness, type compatibility
  - **FLOW**: arbitrary aggregation possible
  - **STOCK**: aggregation possible, except over temporal dimension
  - **VPU**: value-per-unit typically cannot be aggregated

<table>
<thead>
<tr>
<th># Stud</th>
<th>16/17</th>
<th>17/18</th>
<th>18/19</th>
<th>19/20</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>1,153</td>
<td>1,283</td>
<td>1,321</td>
<td>(1,184)</td>
<td>?</td>
</tr>
<tr>
<td>SEM</td>
<td>928</td>
<td>970</td>
<td>939</td>
<td>(838)</td>
<td>?</td>
</tr>
<tr>
<td>ICE</td>
<td>804</td>
<td>868</td>
<td>846</td>
<td>(742)</td>
<td>?</td>
</tr>
<tr>
<td>Total</td>
<td>2,885</td>
<td>3,121</td>
<td>3,106</td>
<td>2,764</td>
<td>?</td>
</tr>
</tbody>
</table>
### Aggregation Types, cont.

#### Additivity

<table>
<thead>
<tr>
<th></th>
<th>FLOW</th>
<th>STOCK: Temporal Agg?</th>
<th>VPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MIN/MAX</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>SUM</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>AVG</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>COUNT</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

#### Type Compatibility (addition/subtraction)

<table>
<thead>
<tr>
<th></th>
<th>FLOW</th>
<th>STOCK</th>
<th>VPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW</td>
<td>FLOW</td>
<td>STOCK</td>
<td>✗</td>
</tr>
<tr>
<td>STOCK</td>
<td>STOCK</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>VPU</td>
<td></td>
<td>VPU</td>
<td></td>
</tr>
</tbody>
</table>
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

- **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

**SELECT**
```
{{[Measures].[Sales],
  [Measures].[Tax]} ON COLUMNS,
  {[Date].[Fiscal].[Year].&[2002],
   [Date].[Fiscal].[Year].&[2003]} ON ROWS
FROM [Adventure Works]
WHERE ([Sales Territory].[Southwest])
```
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, ..., A_k\}
  - **Attribute** A_j: value domain D_j = dom(A_j)
  - **Relation**: subset of the Cartesian product over all value domains D_j
    \[ R \subseteq D_1 \times D_2 \times ... \times D_k, \ k \geq 1 \]

- **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**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>A1 INT</th>
<th>A2 INT</th>
<th>A3 BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>7</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7</td>
<td>T</td>
</tr>
</tbody>
</table>

cardinality: 4
rank: 3
ROLAP – Star Schema

Order
- OrderID
- Order Date

Customer
- CustomerID
- Customer Name
- Customer Address
- City

Salesperson
- SalespersonID
- Salesperson Name
- City
- Quota

Product
- ProductID
- ProdName
- ProdDescr
- Category
- CategoryDescr
- UnitPrice

Date
- DateID
- Date

City
- CityID
- City
- State
- Country

Fact Table
- OrderID
- SalespersonID
- CustomerID
- ProdID
- DateID
- CityID
- Quantity
- Total Price

Denormalized Dimension Tables
ROLAP – Snowflake Schema

Order
- OrderID
- Order Date

Salesperson
- SalespersonID
- SalespersonName
- City
- Quota

Customer
- CustomerID
- Customer Name
- Customer Address
- City

Fact Table
- OrderID
- SalespersonID
- CustomerID
- ProdID
- DateID
- CityID
- Quantity
- Total Price

Product
- ProductID
- ProdName
- ProdDescr
- CategoryName
- UnitPrice

Category
- CategoryName
- CategoryDescr

Date
- DateID
- Date
- Month
- Year

Month
- Month

Year

City
- CityID
- CityName
- StateName

State
- StateName
- Country

“Normalized” Dimension Tables
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

**Step 1:** Reporting
- What did happen?
- Create reports with pre-defined queries.

**Step 2:** Analysis
- Why did it happen?
- Increasing number of ad-hoc queries.

**Step 3:** Forecasting
- What will happen?
- Extension of the analytical model (advanced analytics).

**Step 4:** Operational BI
- What happens right now?
- Continuous streams of ad-hoc queries and propagated updates.

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

- **In the days of Kings and Priests**
  - Computers and Data: Crown Jewels
  - Executives depend on computers
    - But cannot work with them directly
  - The DBA “Priesthood”
    - And their Acronymia: EDW, BI, OLAP

- **The architected Enterprise DWH**
  - Rational behavior ... *for a bygone era*
  - “There is no point in bringing data ... into the data warehouse environment without integrating it.”
    —Bill Inmon, Building the Data Warehouse, 2005
Exкурсус: MAD Skills, cont.

**Magnetic**
- „Attract data and practitioners“
- Use all available data, irrespective of data quality

**Agile**
- „Rapid iteration: ingest, analyze, productionalize“
- Continuous and fast evolution of physical and logical structures (ELT)

**Deep**
- „Sophisticated analytics in Big Data“
- Ad-hoc advanced analytics and statistics

Trend: Cloud Data Warehousing

- **#1 Google Big Query**

- **#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]
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

1. Synonyms/Homonyms
2. Simple Mapping (mathematical)
3. Union Types
4. Complex Mappings
5. Language Expressions
6. Nulls (Missing Values)
7. Virtual Columns
8. Semantic Incompatibility
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

### Table

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

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

**Credit:** Felix Naumann

<table>
<thead>
<tr>
<th>Zip</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>98120</td>
<td>San Jose</td>
</tr>
<tr>
<td>90001</td>
<td>Lost Angeles</td>
</tr>
</tbody>
</table>

**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)

- **Other Tools**
  - IBM IS, Informatica, SAP BO, MS Integration Services
  - Open Source: Pentaho Data Integration, Scriptella ETL, CloverETL, Talend
SQL/OLAP Extensions
Recap: SQL Standard (ANSI/ISO/IEC)

**SQL/OLAP Extensions**

1: Framework

3: CLI
   - Call Level Interface

4: PSM
   - Persistent Stored Modules

9: MED
   - Management of External Data

10: OLB
   - Object Language Bindings

13: JRT
   - Java Routines and Types

14: XML
   - Extensible Markup Language

11: Schemata

2: Foundation

(1) Enhanced Date/Time Fac.

(2) Enhanced Integrity Management

(6) Basic Objects

(7) Enhanced Objects

(8) Active Databases

(10) OLAP

Core SQL (all SQL:92 entry, some extended SQL:92/SQl:99)
Overview Multi-Groupings

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

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

- Grouping Extensions

- GROUPING SETS
  - ROLLUP
  - CUBE
  - GROUPING
Grouping Sets

- **Semantics**
  - Grouping by multiple group-by attribute lists with 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))
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>90</td>
</tr>
<tr>
<td>2004</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>2005</td>
<td>-</td>
<td>30</td>
</tr>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>
Rollup (see also multi-dim ops)

### Semantics
- Hierarchical grouping along dimension hierarchy
- \( \text{GROUP BY ROLLUP} \ (A1,A2,A3) \)
  \[ := \text{GROUP BY GROUPING SETS}((),(A1),(A1,A2),(A1,A2,A3)) \]

### Example

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>-</td>
<td>90</td>
</tr>
<tr>
<td>2004</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2005</td>
<td>4</td>
<td>20</td>
</tr>
</tbody>
</table>
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**
  - Used with ROLLUP or CUBE to identify computed tuples
  - Example

```
SELECT Year, Quarter, SUM(Revenue),
     GROUPING(Quarter) AS Flag
FROM R
GROUP BY ROLLUP (Year, Quarter)
```
Cube

**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)
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>-</td>
<td>90</td>
</tr>
<tr>
<td>2004</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>2005</td>
<td>-</td>
<td>30</td>
</tr>
<tr>
<td>2004</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

GROUP BY CUBE(<attribute-list>)
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

```
Function(arg) OVER (optional)
  partition by clause
  order by clause
  window aggr. group

Partitioning (similar to GROUP BY)
Sorting order per partition
Window definition wrt current tuple
```
RF – Aggregation Function

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

- **Example**

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Rank1</th>
<th>DRank1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

OVER() represents all tuples
RF – Partitioning

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

- **Example**

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>
RF – Partition Sorting

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

- **Example**

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>
RF – Windowing

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

- Example

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Revenue</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>2004</td>
<td>4</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>2005</td>
<td>1</td>
<td>30</td>
<td>25</td>
</tr>
</tbody>
</table>
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

- Next Lectures (Data Integration Architectures)
  - 03 Message-oriented Middleware, EAI, and Replication [Oct 18]
  - 05 Entity Linking and Deduplication [Nov 08]
  - 06 Data Cleaning and Data Fusion [Nov 15]

“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