Enhancing ER Diagrams to View Data Transformations Computed with Queries

Carlos Ordonez, Ladjel Bellatreche

UH (USA), ENSMA (France)

Disclaimer

- Teaching Database Systems courses many years
- Database processing requires understanding data structure before processing
- But I have hardly worked on conceptual modeling or database modeling
- Ladjel gave me guidance

Motivation: Data Sets for Analytics

- Input for Machine Learning or Statistical Models: *n* records, *p* features/variables (dimensions, categorical/discrete)
- Built by many SQL queries: SPJA
- Original database does have some ER diagram behind, maybe denormalized
- Queries, views: disorganized, written independently
- DB populated by queries instead of transactions/ETL
- Data set does have entity (and relational) representation
- In general, no ER diagram exists for temporary tables/views/exports

Our contributions

- Extending existing ER diagram with “data transformation entities”
- Minimal changes to UML diagram notation
- Entity universe: source + transformation
- Tranformed attribute: any expression from SPJA relational algebra
- Fast algorithm to create ER diagram from queries
- Preliminary study of ER diagram properties

Preliminaries

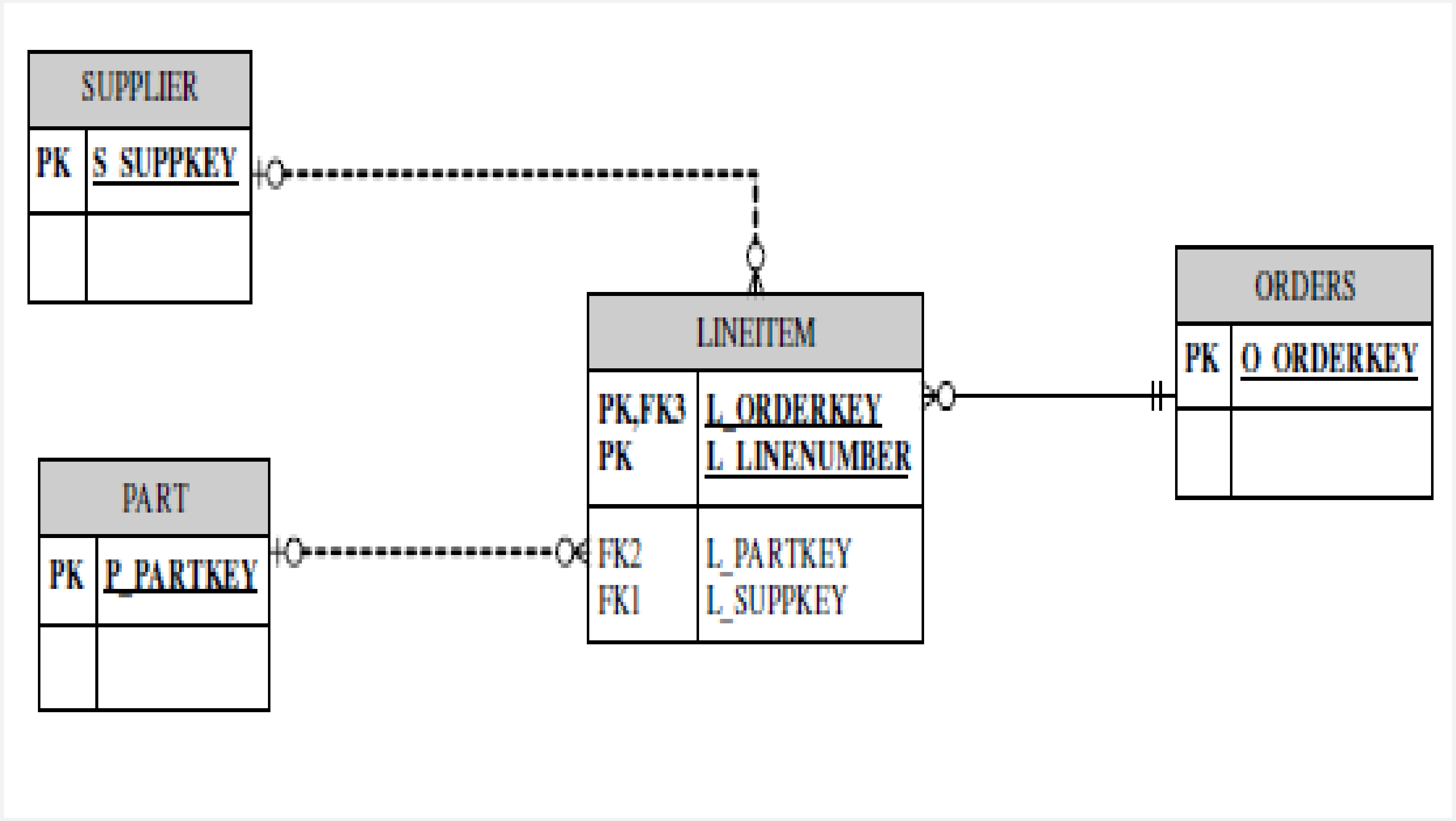
- UML entity notation: scalable, Object-Oriented
- n tables: all linked by 1:N and 1:1 relationships
- Entity and referential integrity: satisfied

$Ti\ (K) \rightarrow Tj\ (K)$

$\Pi_K\ (T\ i\ ) \subseteq \Pi_K\ (T\ j\ )$

- New tables derivedvonly with SPJA queries
- Derived attributes with aggregations, math and string expressions, including CASE statements

Example: Input Database ER Diagram



Our ER diagram extensions

- Logical level: minimal:
  - labeling entity names,
  - same notation for relationships
- Physical level (SQL):
  - zoom in view with relational queries
- Semantics: data analyst point of view

Data Transformations

- Entity (table) level: only relational queries
  - join (denormalization to gather attributes, left outer joins)
  - aggregation/projection (to derive new attributes)
  - selection (filter is important)
- Attribute (column) level:
  - denormalization (expressions, functions, CASE)
  - aggregation (GROUP BY, global)

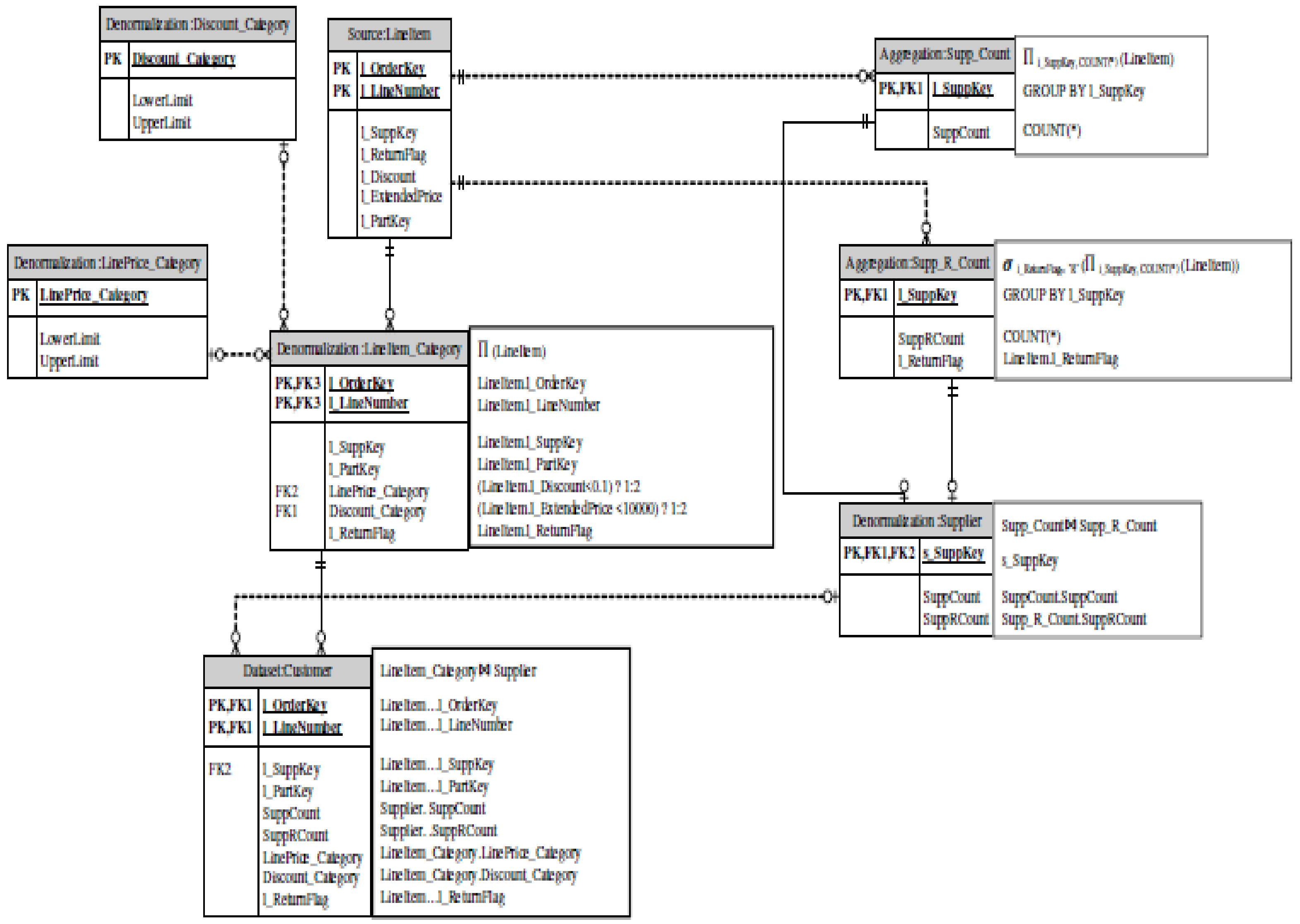
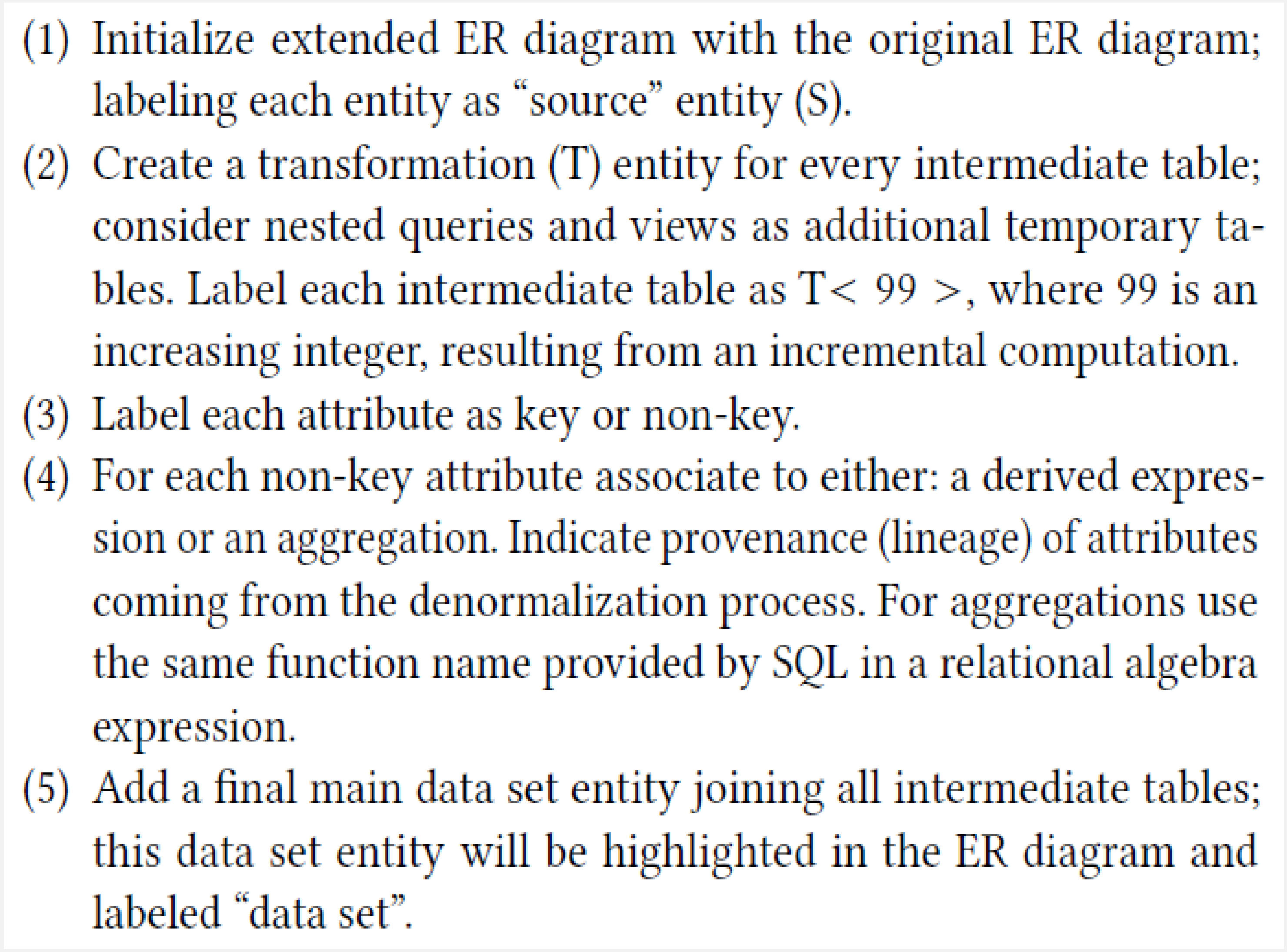


Figure 2: Transformation ER diagram for TPC-H database (low level).

Properties of our ER diagram

- Logical and physical level come closer, but still separate
- PKs and Fks remain the glue
- Queries take the role of insert/delete/update in traditional DB
- Complete (no table left out) and consistent (every piece of data derived via queries)
- Transformation entities are weak entities
- Provenance can be tracked; flow can be embedded in entity labels

Algorithm



(1) Initialize extended ER diagram with the original ER diagram; labeling each entity as "source" entity (S).

(2) Create a transformation (T) entity for every intermediate table; consider nested queries and views as additional temporary tables. Label each intermediate table as T< 99 >, where 99 is an increasing integer, resulting from an incremental computation.

(3) Label each attribute as key or non-key.

(4) For each non-key attribute associate to either: a derived expression or an aggregation. Indicate provenance (lineage) of attributes coming from the denormalization process. For aggregations use the same function name provided by SQL in a relational algebra expression.

(5) Add a final main data set entity joining all intermediate tables; this data set entity will be highlighted in the ER diagram and labeled "data set".

Conclusions

- Any diagram helps analysts, but there will not be an ER model in the traditional sense
- A first step to have a DB ER diagram of data transformations
- Complements flow diagrams
- Relational, but can be later extended to non-relational data (text, semistructured)
- DB state: As of now, source refreshed via transactions/ETL. Versioning: future (temporal & stream databases)