Aggregate Navigation (Ch 15, pages 345++)

From "20 criteria for dimensional friendly systems"

#4. Open Aggregate Navigation.

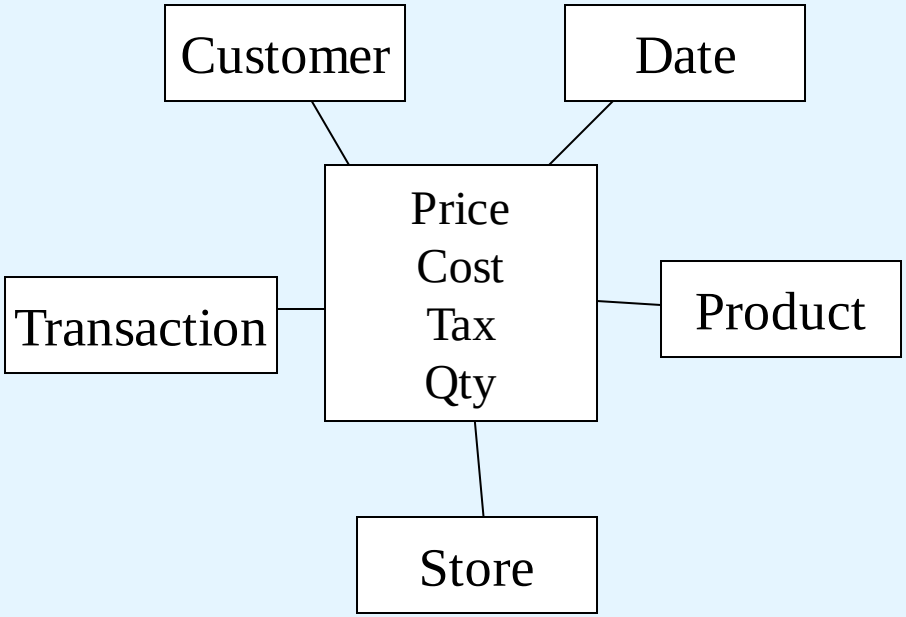
The system uses physically stored aggregates as a way to enhance performance of common queries. These aggregates, like indexes, are chosen silently by the database if they are physically present. End users and application developers do not need to know what aggregates are available at any point in time, and applications are not required to explicitly code the name of an aggregate. All query processes accessing the data, even those from different application vendors, realize the full benefit of aggregate navigation.

Aggregation

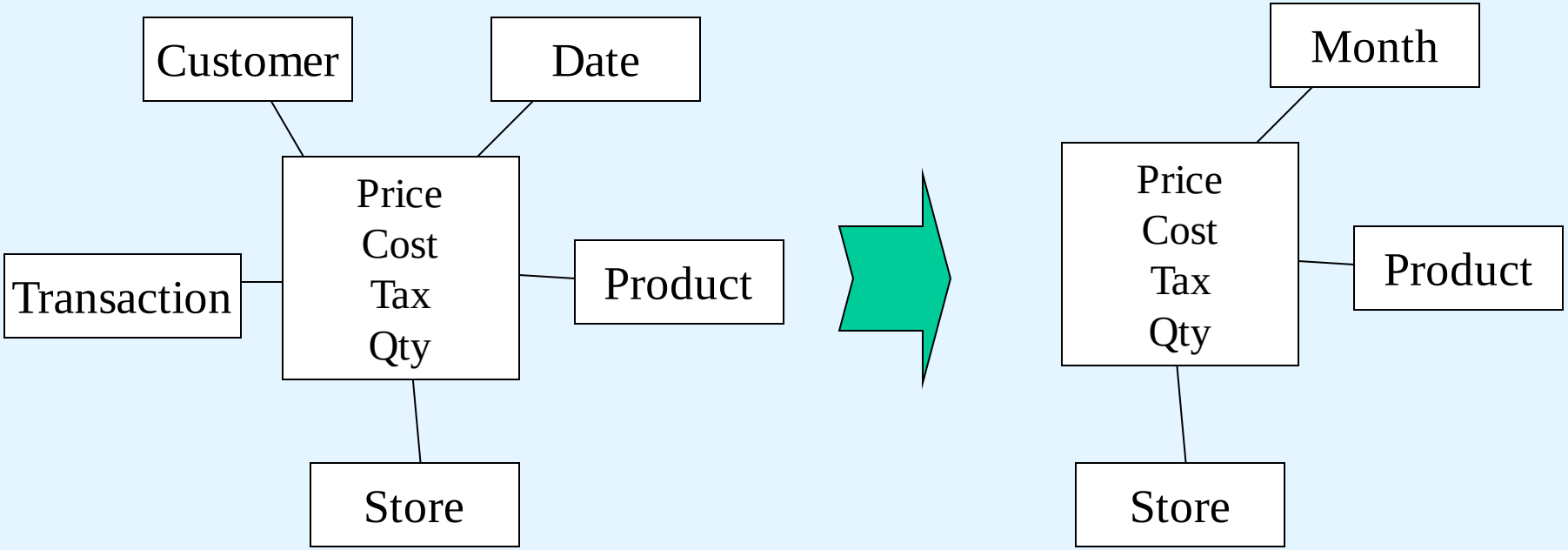
- Created for performance reasons
- Using one **base** star schema, many aggregates can be defined
- Some dimensions could be **lost**, some **shrunken** or **collapsed** (but conforming)

Consider the

Transaction-level schema:



Aggregation



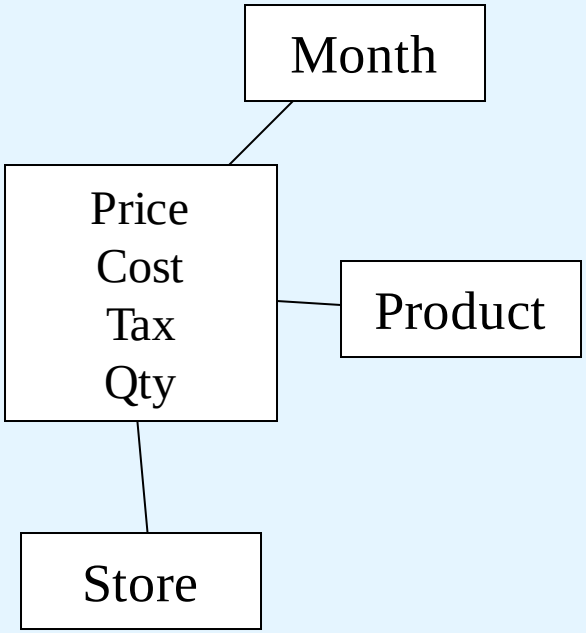
Transaction-level schema

Aggregate schema with:

- lost dimensions (transaction, customer)
- shrunken dimension (month)
- metrics are accumulated over the transactions, customers, month
- How to create the fact table?

Aggregation

- Suppose you must create the schema:



- What are the steps to do so?
  - Product and Store already exist, but the others must be created (if they don't already exist)

Aggregation

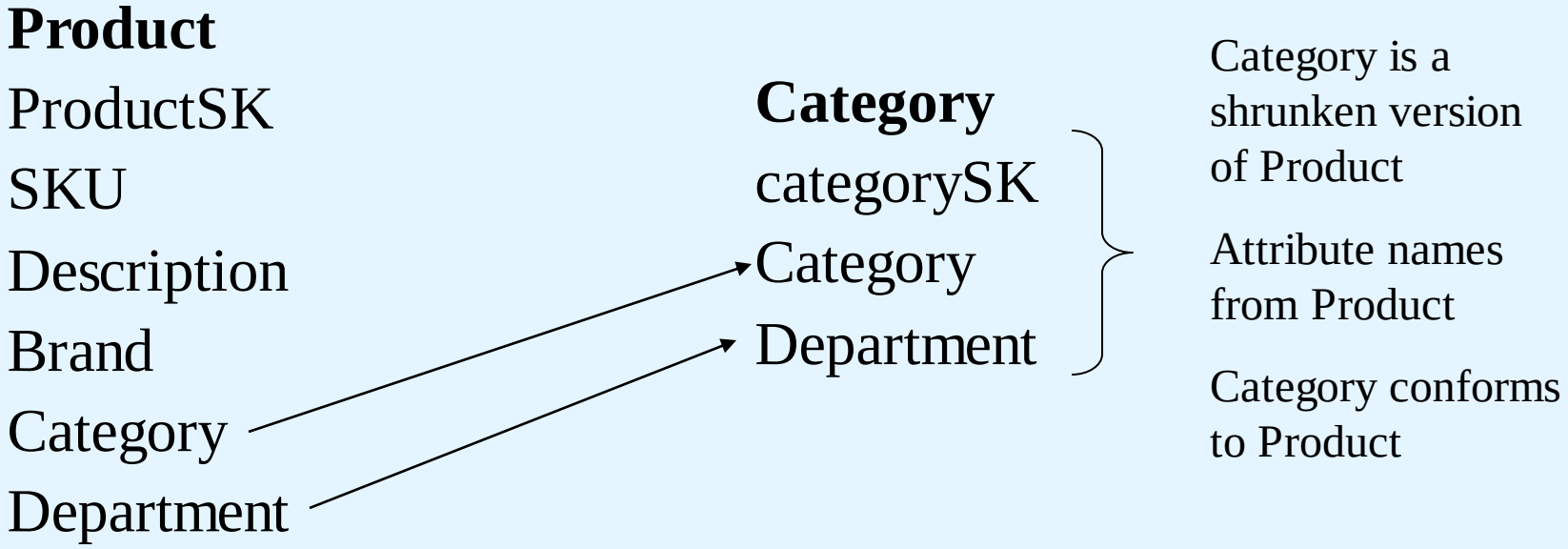
Design Goal 1

- Aggregates must be stored in their own fact tables; each distinct aggregation level must occupy its own unique fact table

Aggregation

Design Goal 2

- If not already existing, dimension tables attached to the aggregate fact tables must be shrunken versions of the dimension tables associated with the base fact table



Aggregation

Design Goal 3

- The base atomic fact table and all of its related aggregate fact tables must be associated together as a "family of schemas" so that the aggregate navigator knows which tables are related to one another.
  - Kimball's algorithm/technique for mapping an SQL statement from a base schema to an aggregate schema → slide 9

Aggregation

Design Goal 4

- Force all SQL created by any end-user data access tool or application to refer exclusively to the base fact table and its associated full-size dimension tables.

Kimball's Aggregate Navigation Algorithm

1. For any given SQL statement presented to the DBMS, find the smallest fact table that has not yet been examined in the family of schemas referenced by the query. "Smallest" in this case means the least number of rows. Choose the smallest schema and proceed to step 2.
2. Compare the table fields in the SQL statement to the table fields in the particular Fact and Dimension tables being examined.
   - This is a series of lookups in the DBMS system catalog.
   - If all of the fields in the SQL statement can be found in the Fact and Dimension tables being examined, alter the original SQL by simply substituting destination table names for original table names. No field names need to change.
   - If any field in the SQL statement cannot be found in the current Fact and Dimension tables, then go back to step 1 and find the next larger Fact table.
3. Run the altered SQL. It is guaranteed to return the correct answer because all of the fields in the SQL statement are present in the chosen schema.

Materialized View (MV)

An MV is a view where the result set is pre-computed and stored in the database

Periodically an MV must be re-computed to account for updates to its source tables

MVs represent a performance enhancement to a DBMS as the result set is immediately available when the view is referenced in a SQL statement

MVs are used in some cases to provide summary or aggregates for data warehousing transparently to the end-user

Query optimization is a DBMS feature that may re-write an SQL statement supplied by a user.

Aggregates

Oracle provides a database-specific means for navigating user queries to the optimum aggregated tables

The query rewrite mechanism in the Oracle server automatically rewrites an SQL query to use the summary tables.

https://docs.oracle.com/cd/B28359_01/server.111/b28313/qrbasic.htm#i1006201

https://docs.oracle.com/cd/B10501_01/server.920/a96520/physical.htm#97989

Materialized Views

Oracle example:

```
CREATE MATERIALIZED VIEW hr.mview_employees AS
    SELECT employees.employee_id, employees.email
    FROM employees
UNION ALL
    SELECT new_employees.employee_id, new_employees.email
    FROM new_employees;
```

Materialized Views

SQL Server has “indexed views” … let’s examine this article

http://www.databasejournal.com/features/mssql/article.php/2119721