MOLAP: Dimensional Modeling

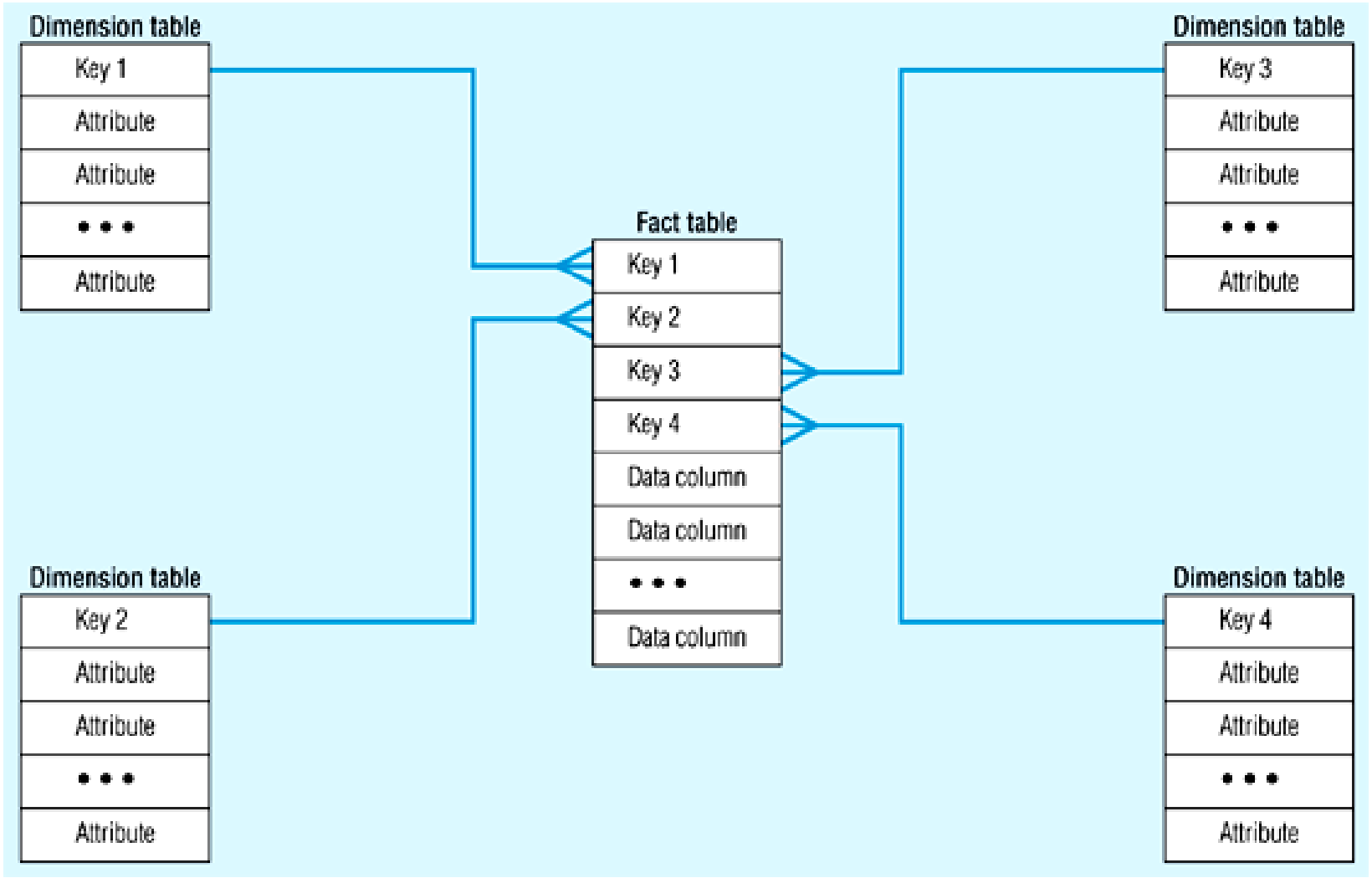
JYOTHISHMATHI INSTITUTE OF TECHNOLOGY AND SCIENCE

Prepared By
Ravindar Mogili

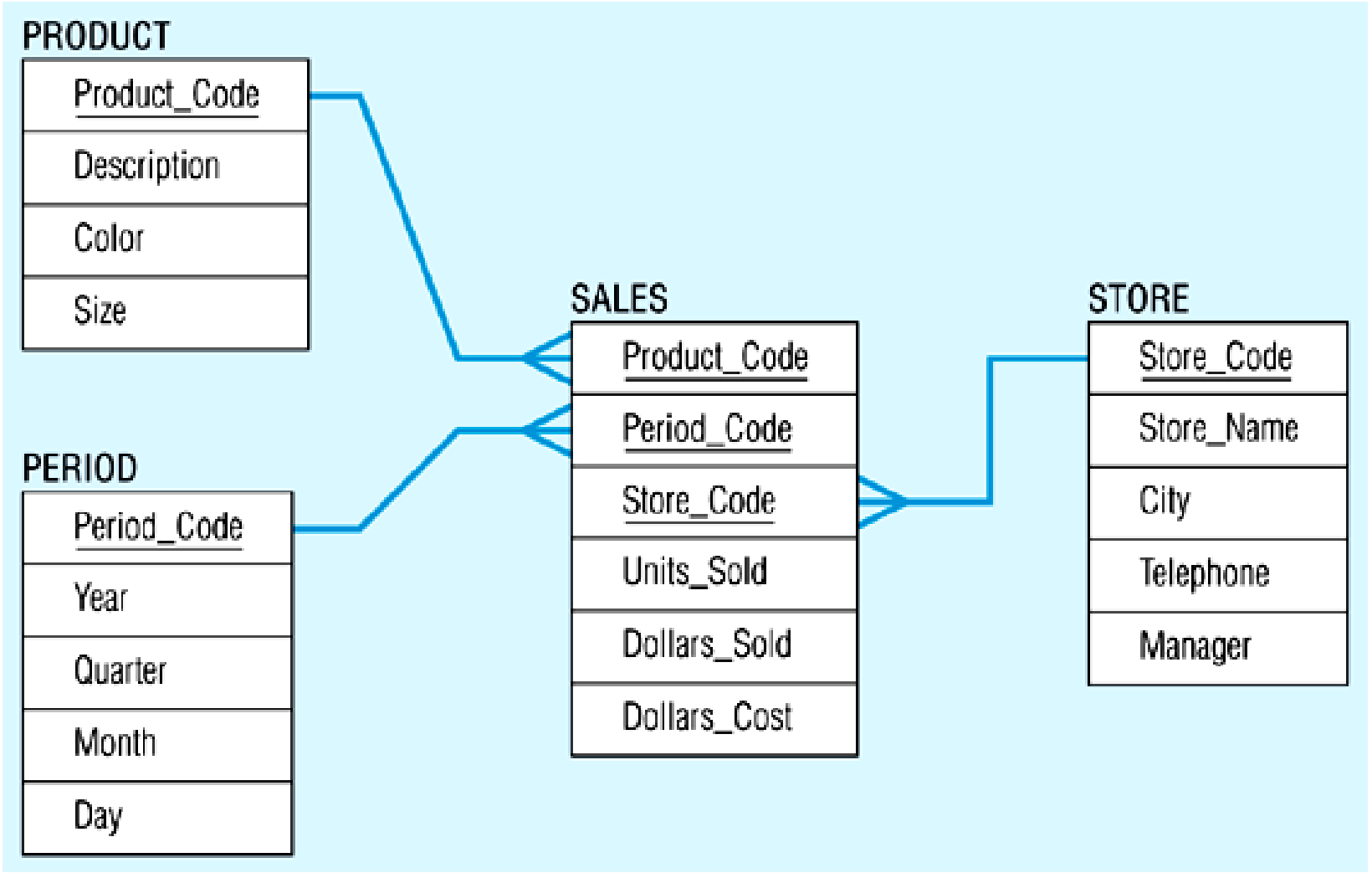
MOLAP: Dimensional Modeling

- MDDB: a special-purpose data model
- Facts stored in multi-dimensional arrays
- Dimensions used to index array
- Sometimes on top of relational DB
- Products
  - Pilot, Arbor Essbase, Gentia

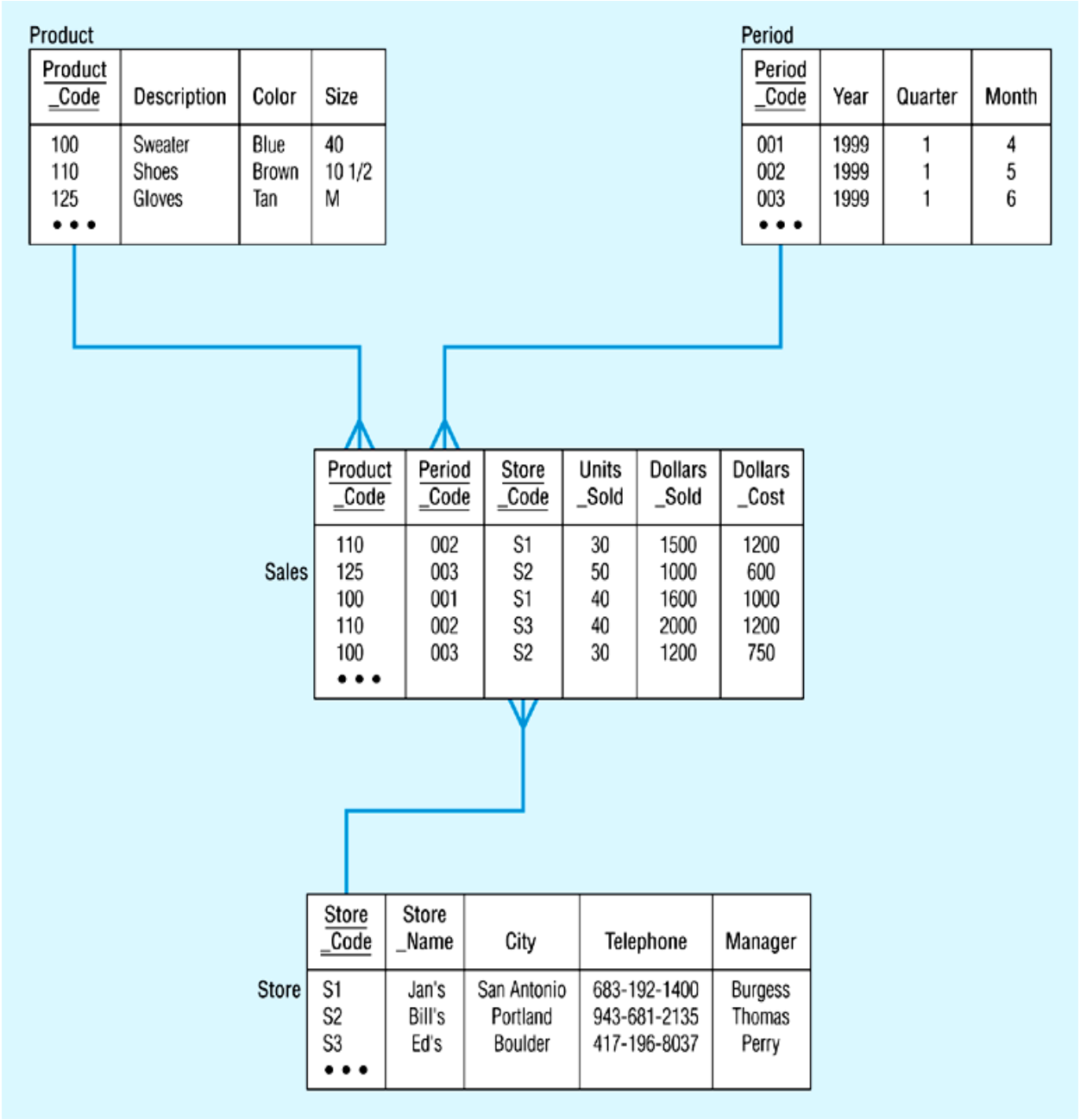
Star Schema (in RDBMS)



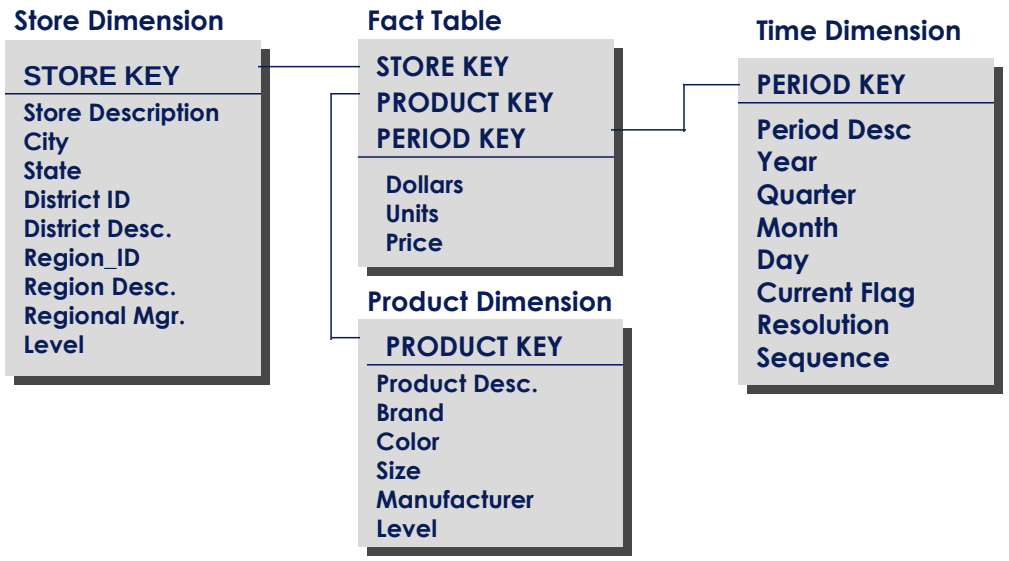
Star Schema Example



Star Schema with Sample Data



The "Classic" Star Schema

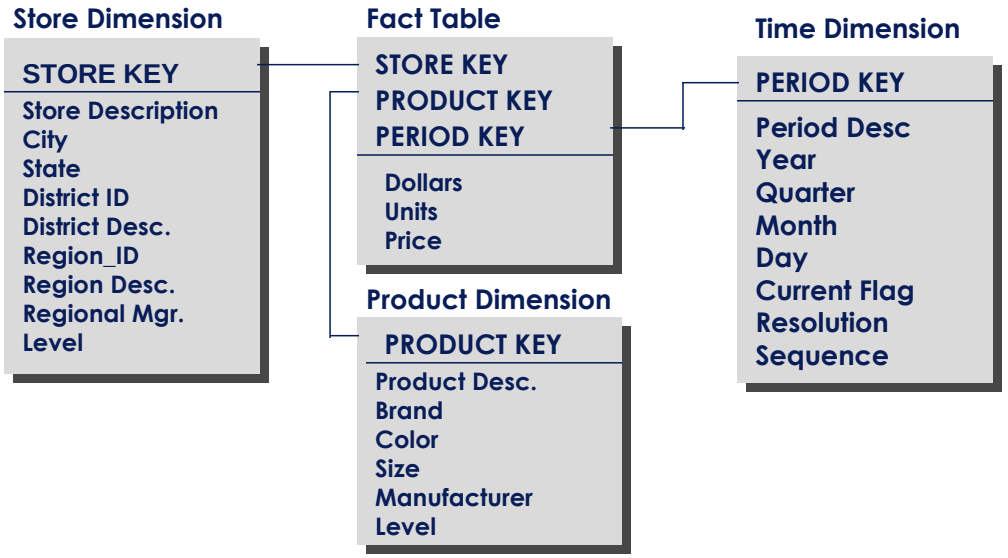


- A single fact table, with detail and summary data
- Fact table primary key has only one key column per dimension
- Each key is generated
- Each dimension is a single table, highly denormalized

Benefits: Easy to understand, easy to define hierarchies, reduces # of physical joins, low maintenance, very simple metadata

Drawbacks: Summary data in the fact table yields poorer performance for summary levels, huge dimension tables a problem

The "Classic" Star Schema



The biggest drawback: dimension tables must carry a *level* indicator for every record and every query must use it. In the example below, without the level constraint, keys for all stores in the NORTH region, including aggregates for region and district will be pulled from the fact table, resulting in error.

Example:

```
Select A.STORE_KEY, A.PERIOD_KEY, A.dollars from
Fact_Table A
        where A.STORE_KEY in (select STORE_KEY
                from Store_Dimension B
                where region = "North" and Level = 2)
```

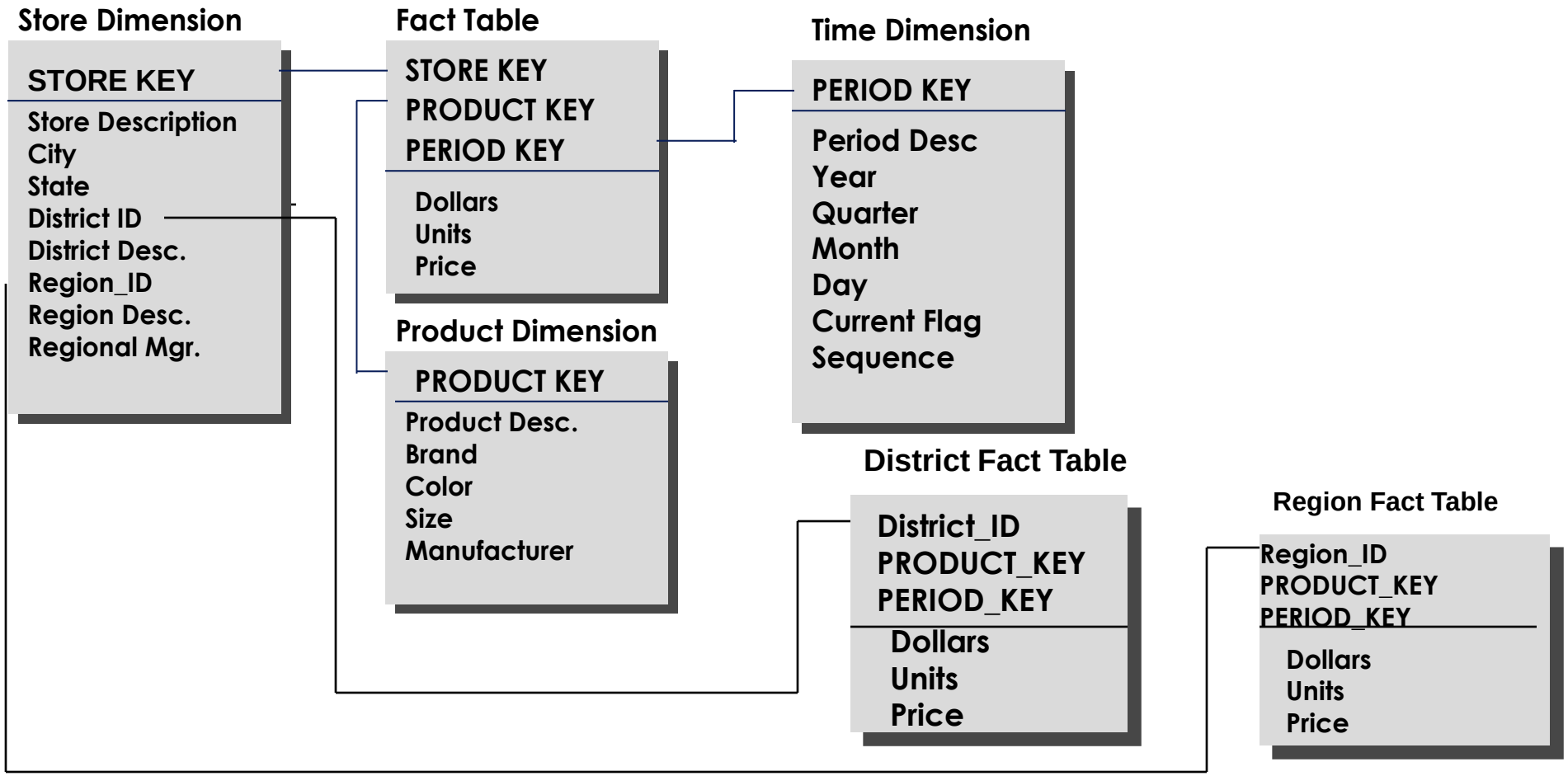
and *etc*...

Level is needed whenever aggregates are stored with detail facts.

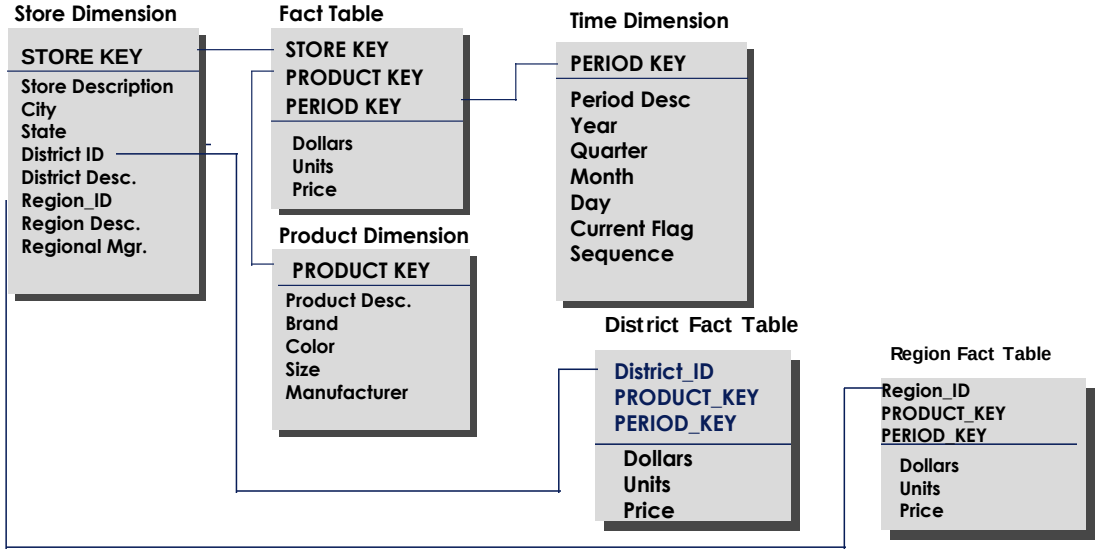
The “Level” Problem

- Level is a problem because because it causes potential for error. If the query builder, human or program, forgets about it, perfectly reasonable looking WRONG answers can occur.
- One alternative: the FACT CONSTELLATION model...

The "Fact Constellation" Schema



The "Fact Constellation" Schema



In the Fact Constellations, aggregate tables are created separately from the detail, therefor
it is impossible to pick up, for example, Store detail when querying
the District Fact Table.

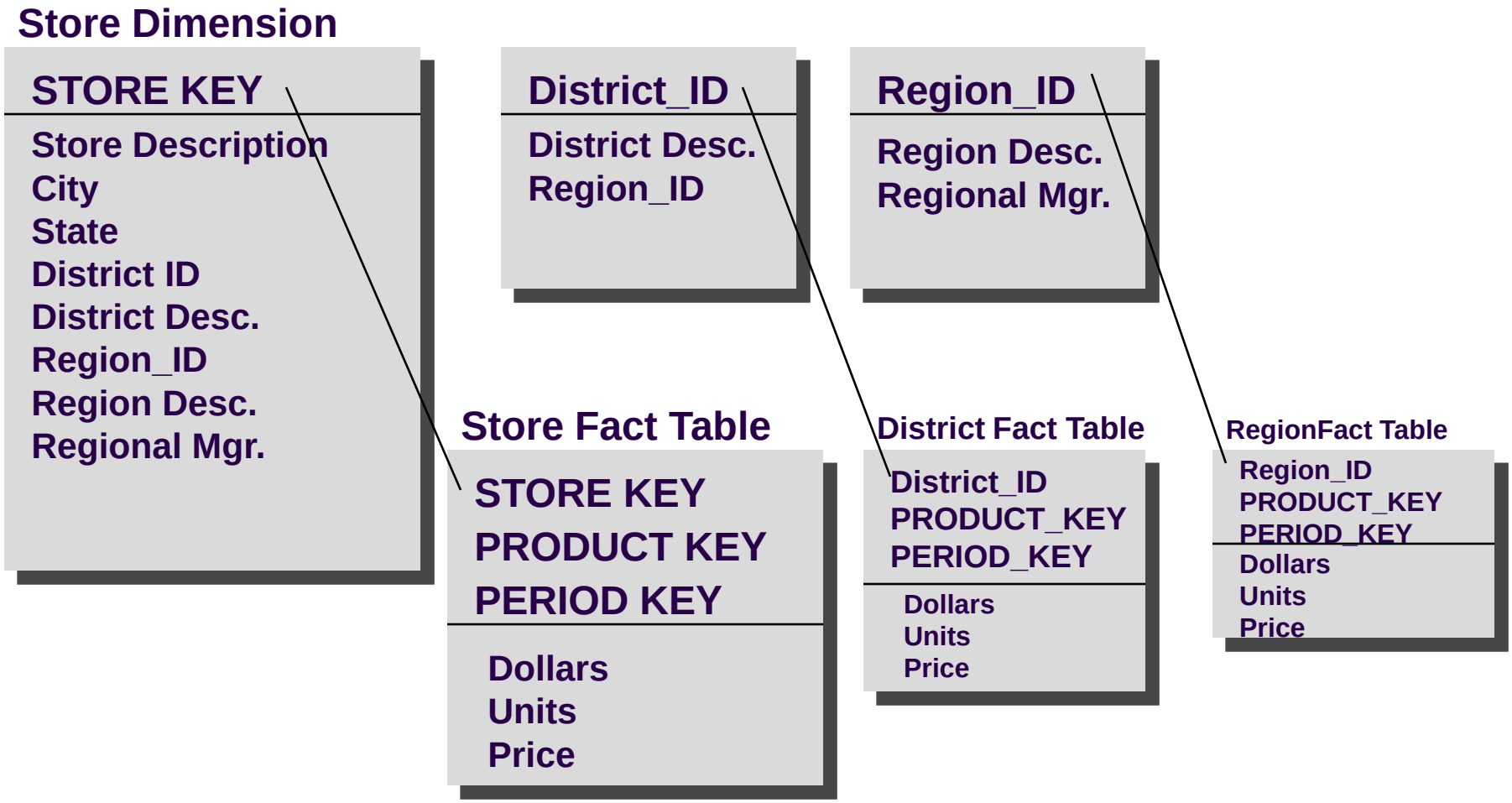
Major Advantage: No need for the "Level" indicator in the dimension tables, since no aggregated data is stored with lower-level detail

Disadvantage: Dimension tables are still very large in some cases, which can slow performance; front-end must be able to detect existence of aggregate facts, which requires more extensive metadata

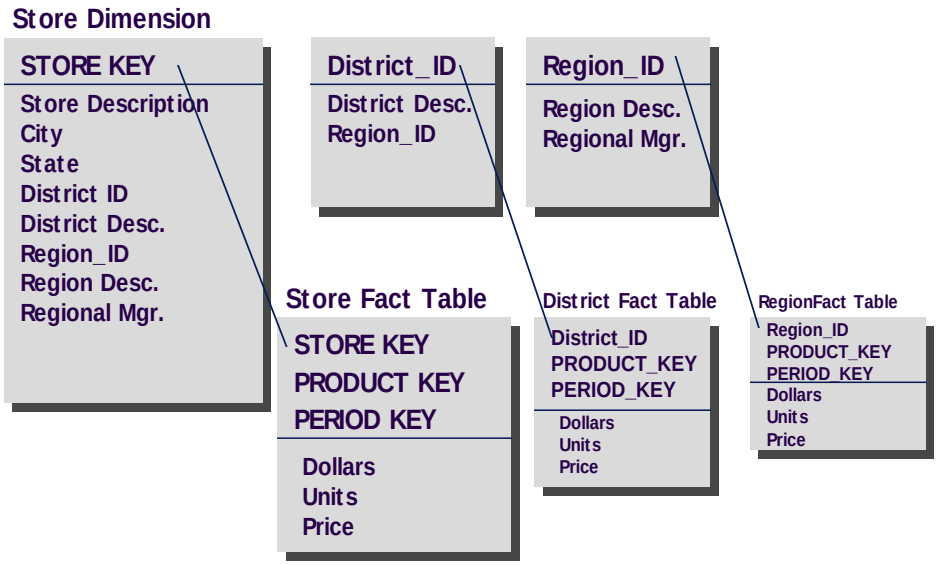
Another Alternative to "Level"

- Fact Constellation is a good alternative to the Star, but when dimensions have very high cardinality, the sub-selects in the dimension tables can be a source of delay.
- An alternative is to normalize the dimension tables by attribute level, with each smaller dimension table pointing to an appropriate aggregated fact table, the "Snowflake Schema" ...

The "Snowflake" Schema



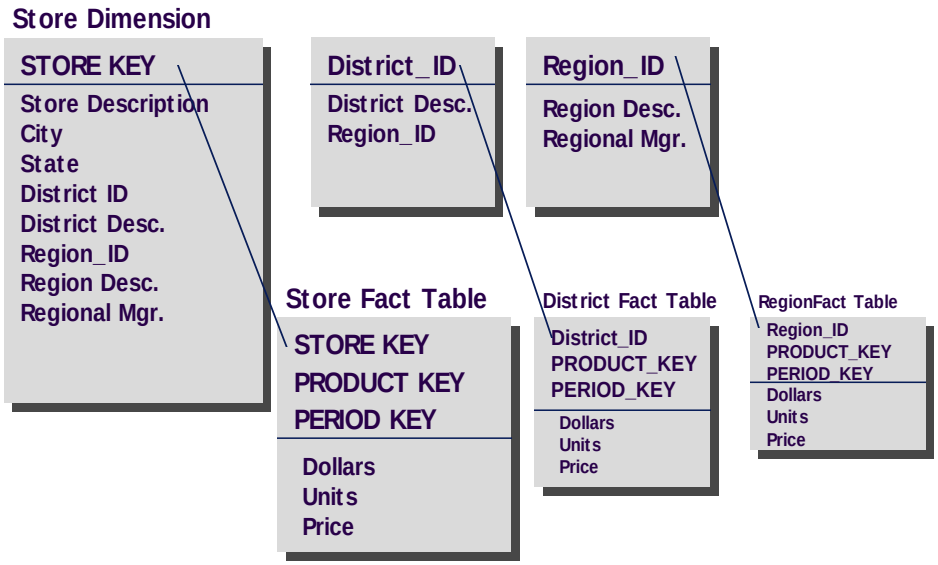
The “Snowflake” Schema



- No LEVEL in dimension tables
- Dimension tables are normalized by decomposing at the attribute level
- Each dimension table has one key for each level of the dimensionís hierarchy
- The lowest level key joins the dimension table to both the fact table and the lower level attribute table

How does it work? The best way is for the query to be built by understanding which summary levels exist, and finding the proper snowflaked attribute tables, constraining there for keys, then selecting from the fact table.

The "Snowflake" Schema



- Additional features: The original Store Dimension table, completely de-normalized, is kept intact, since certain queries can benefit by its all-encompassing content.
- In practice, start with a Star Schema and create the "snowflakes" with queries. This eliminates the need to create separate extracts for each table, and referential integrity is inherited from the dimension table.

Advantage: Best performance when queries involve aggregation

Disadvantage: Complicated maintenance and metadata, explosion in the number of tables in the database