Advanced Dimensional Modeling

Slowly Changing Dimensions

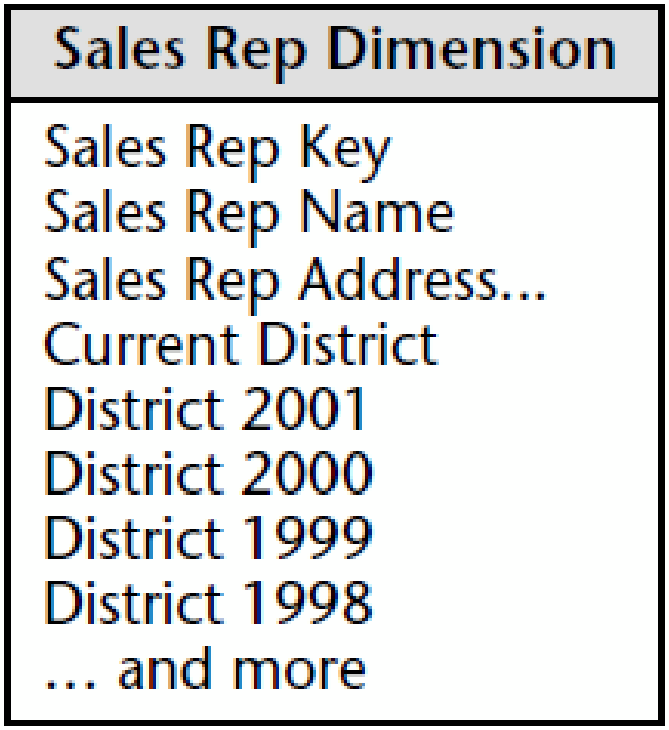
- Hybrid Type: Combination 1, 2 & 3 changes
  - New attribute for predictable series (such as yearly changes)
  - Type 2 changes with prior or original attributes included
  - Expanded dimension table for durable key inclusion in fact
  - Added complexity to users

Hybrid Slowly Changing Dimension Techniques

- Two approaches that combine the basic SCD techniques:
  - Predictable changes with multiple version overlays
  - Unpredictable changes with single-version overlay
- These approaches provide more flexibility at the cost of greater complexity

Predictable Changes with Multiple Version Overlays

- Used in cases of sales organization realignments
- Example: Over a 5-year period the sales organization is reorganized five times.
- At first sight, candidate for Type 2 approach (add dimension row), but more complex business requirements. E.g.,
  - Report each year's sales using the district map for that year
  - Report each year's sales using the district map from an arbitrary different year
  - Report an arbitrary span of years' sales using a single district map from a chosen year.
- Type 3 is also inappropriate because >2 district maps
- Because changes are predictable, an extension of Type 3 is possible → Multiple District columns:
  - Current District; District 2001; District 2002; …


Sales Rep Dimension
Sales Rep Key
Sales Rep Name
Sales Rep Address...
Current District
District 2001
District 2000
District 1999
District 1998
... and more

Unpredictable Changes with Single-Version Overlay

- Preserve historical accuracy surrounding unpredictable attribute changes while supporting the ability to report historical data according to the current values
- Issue a new dimension row (type 2) to capture the change and add a new dimension column to track the historical value (type 3). Also, overwrite "Current Department" value (Type 1).
- See example on page 104.

Rapidly Changing Dimensions

From the previous slides: What is slow?

What if the changes are fast?

Must a different design technique be used?

- Small dimensions:
  - the same technologies as for slowly changing dimensions may be applied
- Large dimensions:
  - the choice of indexing techniques and data design approaches are important
  - find suppress duplicate entries in the dimension
  - do not create additional records to handle the slowly changing dimension problem

Rapidly changing very large dimensions

- Break off some of the attributes into their own separate dimension(s), a demographic dimension(s).
  - force the attributes selected to the demographic dimension to have relatively small number of discrete values
  - build up the demographic dimension with all possible discrete attributes combinations
  - construct a surrogate demographic key for this dimension

NB! The demographic attributes are the one of the heavily used attributes. Their values are often compared in order to identify interesting subsets.

More Rapidly Changing Dimensions

- Break off the rapidly changing attributes into one or more separate dimensions
- Two foreign keys in fact table:
  1. Primary dimension table
  2. Rapidly changing attribute(s)

Rapidly Changing Monster Dimensions

- Multi-million customer dimension present unique challenges that warrant special treatment:
  1. Browsing or constraining takes too long
  2. Type-II change not feasible
  3. Business users want to track the myriad of customer attribute changes, e.g., insurance companies want accurate information of customers at the time of approval of a policy or when a claim is made

Mini-Dimensions

- Single technique to handle browsing-performance & change tracking problems
- Separate out frequently analyzed or frequently changing attributes into a separate dimension, called *mini-dimension*

Mini-Dimensions

Demographic Key	AGE	GENDER	INCOME LEVEL
1	20-24	M	< 20000
2	20-24	M	20K-24999
3	20-24	M	25K-29999
18	25-29	M	20K-24999
10	25-29	M	25K-29999

Mini-Dimensions

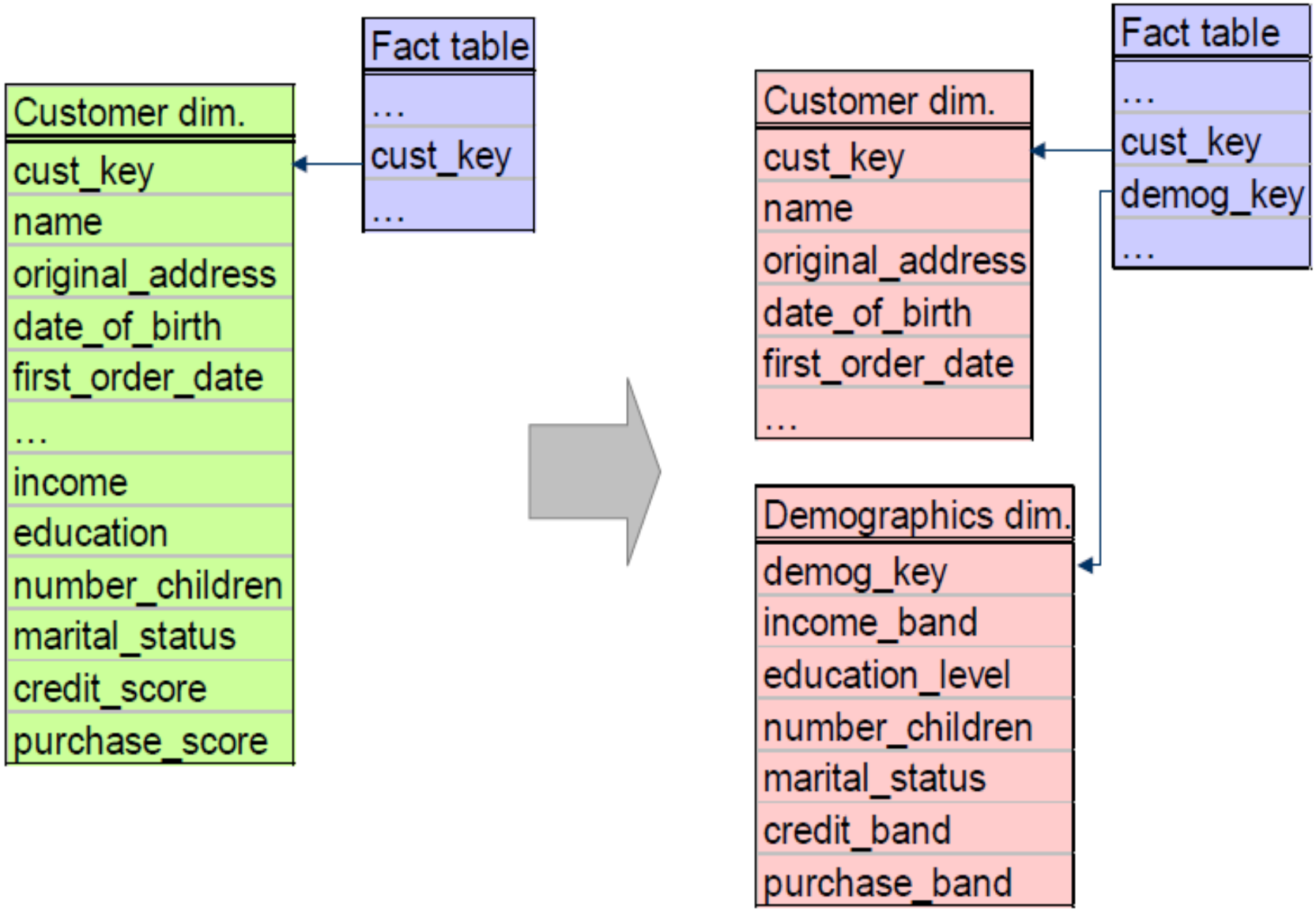
- Mini-dimension can not be itself allowed to grow very large
- 5 demographic attributes
- Each attribute can take 10 distinct values
- How many rows in mini-dimension?

10,0000

Mini-Dimensions

- Separate out a package of demographic attributes into a demographic *mini-dimension*
- Age, gender, marital status, no. of children, income level, etc.
- One row in *mini-dimension* for each unique combination of these attibutes

Demographic Minidimension



Demographic Minidimension

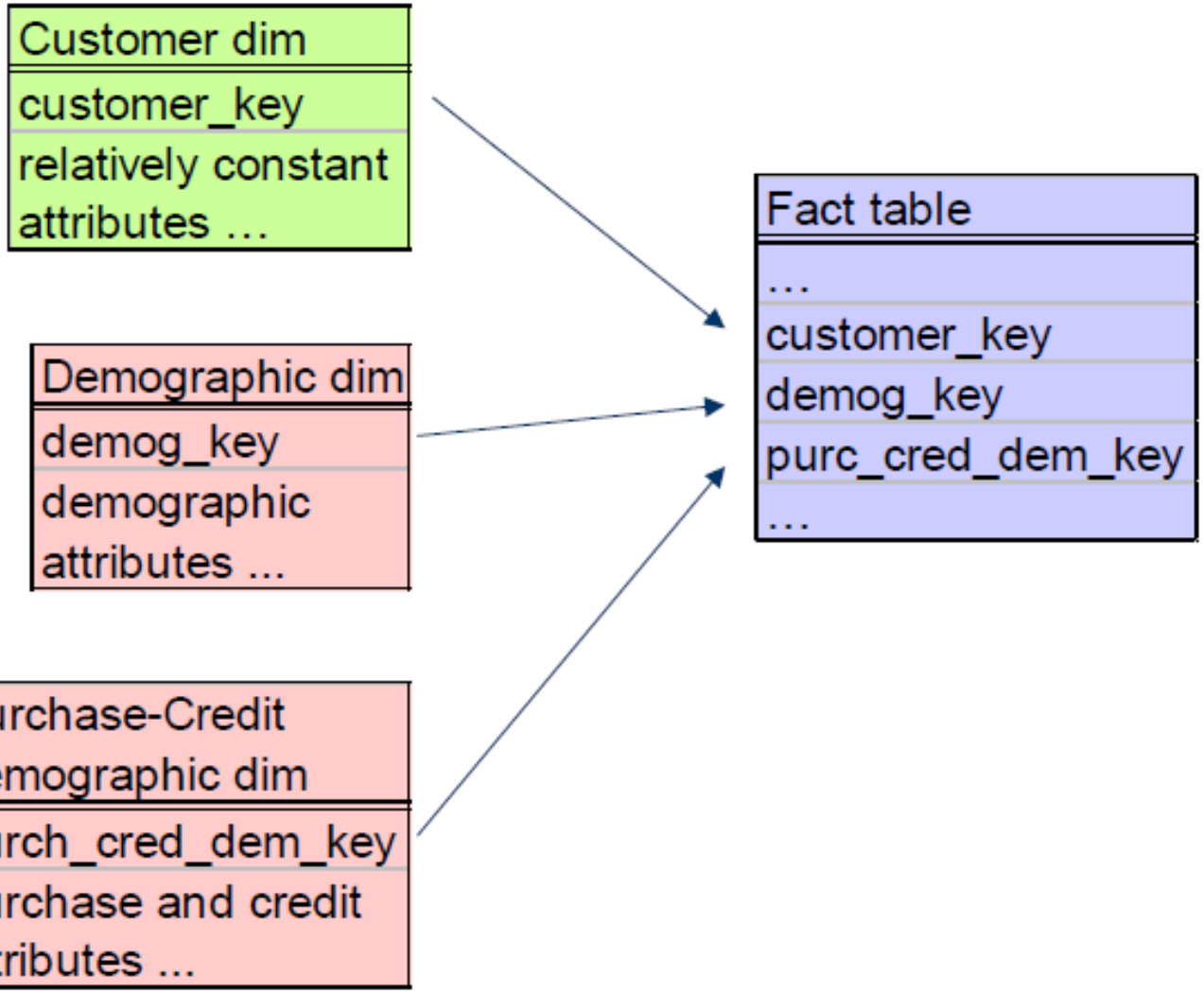
Demographics dim.
demog_key
income_band
education_level
marrital_status

Three values
Two values
Two values

3*2*2=12 rows

D1	-100 000	Graduate	Married
D2	100 000-200 000	Graduate	Married
D3	200 000-	Graduate	Married
D4	-100 000	Non-graduate	Married
D5	100 000-200 000	Non-graduate	Married
D6	200 000-	Non-graduate	Married
	..cont	*..cont*	*..cont*

Two Demographic Minidimensions



Demographic Minidimension

- Advantages
  - frequent ‘snapshoting’ of customers profiles with no increase in data storage or data complexity
- Drawbacks
  - the demographic attributes are clumped into banded ranges of discrete values (it is impractical to change the set of value bands at a later time)
  - the demographic dimension itself can not be allowed to grow too large
  - slower down the browsing
- What if the fact table (connecting the demographic minidimension with the customer dimension) is sparse?

Demographic Minidimension

- What to do if the fact table (connecting the demographic minidimension with the customer dimension) is sparse?

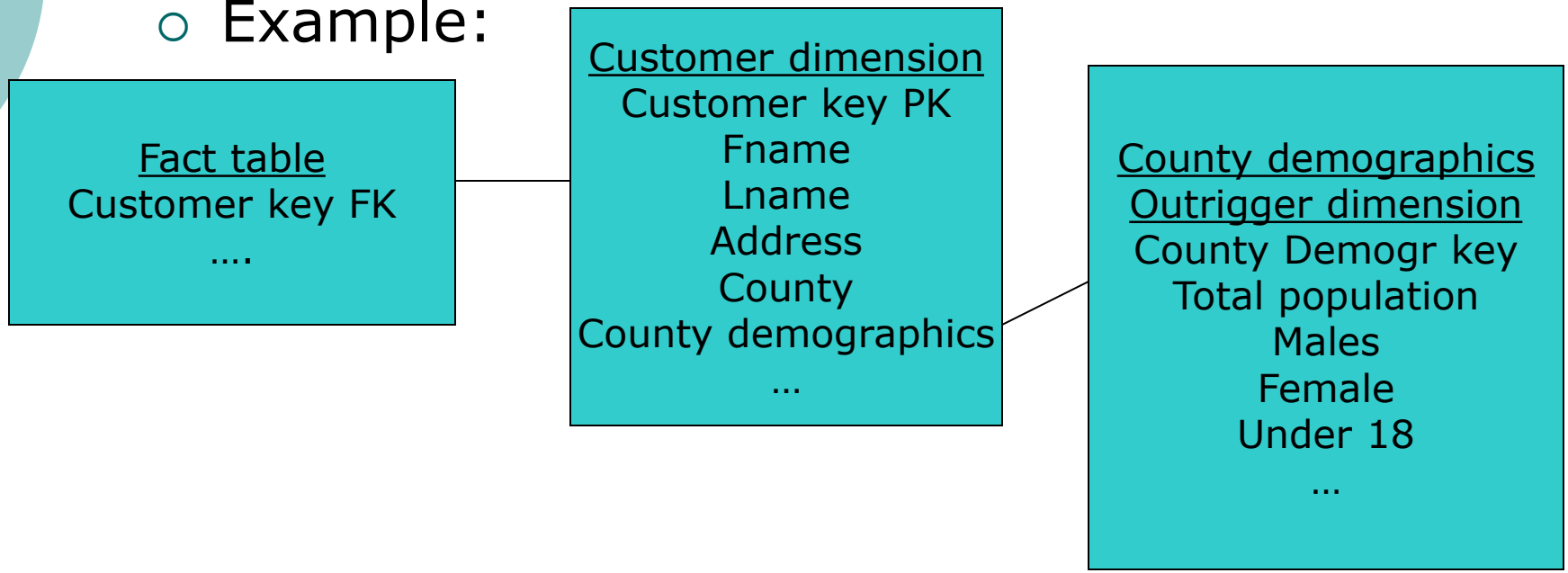
  – Define a demographic transaction event, i.e., introduce a new fact table

  or

  – Add a current demographic key to the customer dimension table

Outrigger Dimensions

- Look like a beginning of a snowflake
- Example:



- Large number of attributes
- Different grain
- Different update frequency

Junk Dimensions

When a number of miscellaneous flags and text attributes exist, the following design alternatives should be avoided:

- Leaving the flags and attributes unchanged in the fact table record
- Making each flag and attribute into its own separate dimension
- Stripping out all of these flags and attributes from the design

A better alternative is to create a junk dimension.

A junk dimension is a convenient grouping of flags and attributes to get them out of a fact table into a useful dimensional framework

“Junk” Dimensions

- Miscellaneous flags and text attributes that cannot be placed into one of existing dimension tables
- Store them in a “junk” dimension
  - Store as unique combinations
  - Example:

Invoice indicator key	Payment terms	Order mode	Ship mode
1	Net 10	Web	Freight
2	Net 10	Web	Air
3	Net 10	phone	Freight
4	Net 10	Phone	Air
5	Net 30	Web	Freight
6	Net 30	Web	Air
7	Net 30	Phone	Freight
8	Net 30	Phone	Air

  - Data profiling is useful in identifying junk dimension candidates

Dimension Role Playing

- A single table that plays multiple roles (using views) to create synonym dimension attributes.
- Most common role playing dimension is the Date Dimension. i.e. separate role playing dimensions for order date and ship date.

Role-Playing Dimension

- What to do when a single dimension appears several times in the same fact table?
- Consider a fact table to record the status and final disposition of a customer order
- Dimensions of this table could be Order Date, Packaging Date, Shipping Date, Delivery Date, Payment Date, Return Date, Refer to Collection Date, Order Status, Customer, Product, Warehouse, and Promotion.

Role-Playing Dimension

- Note that the first 7 dimensions are all time
- 7 FKs from the FT to the time dimension!!
- We can not join these 7 FKs to the same table
- SQL would interpret such a seven-way simultaneous join as requiring that all of the dates be the same
- Is this what we want?

Role-Playing Dimension

- We need to make SQL believe that there are 7 independent time dimension tables
- The column labels in each of these tables should also be different!
- WHY?
- We will not be able to tell the columns apart if several of them have been dragged into a report
- How can we do this?

Role-Playing Dimension

- We cannot literally use a single time table
- But we still want to build & maintain single time table behind the scenes
- Create an illusion for the user
  - Make 7 identical physical copies of the time table
  - Make 7 “virtual” copies using the SQL’s SYNONYM command
- Once these clones are in place, we still need to define a SQL view on each copy in order to make the field names uniquely different.

```
CREATE VIEW ORDER_DATE (ORDER_DATE_KEY, ORDER_DAY_OF_WEEK,
ORDER_MONTH...)
AS SELECT DATE_KEY, DAY_OF_WEEK, MONTH, . . . FROM DATE
```

and

```
CREATE VIEW REQ_SHIP_DATE (REQ_SHIP_DATE_KEY, REQ_SHIP_DAY_OF_WEEK,
REQ_SHIP_MONTH ...)
AS SELECT DATE_KEY, DAY_OF_WEEK, MONTH, . . . FROM DATE
```

Role-Playing Dimension

- Now that we have seven differently described Time dimensions, they can be used as if they were independent
- Can have completely unrelated constraints, and they can play different roles in a report

Role-Playing Dimension

Other Example

- Frequent Flyer flight segment FT need to include Flight Date, Segment Origin Airport, Segment Destination Airport, Trip Origin Airport, Trip Destination Airport, Flight, Fare Class, and Customer.
- The 4 Airport dimensions are 4 different roles played by a single underlying Airport table

Date/Time Dimensions

- Standard date dimension table at a daily grain

- Rationale: remove association with calendar from BI applications
- Use numeric surrogate keys for date dimension tables

Date/Time Dimensions

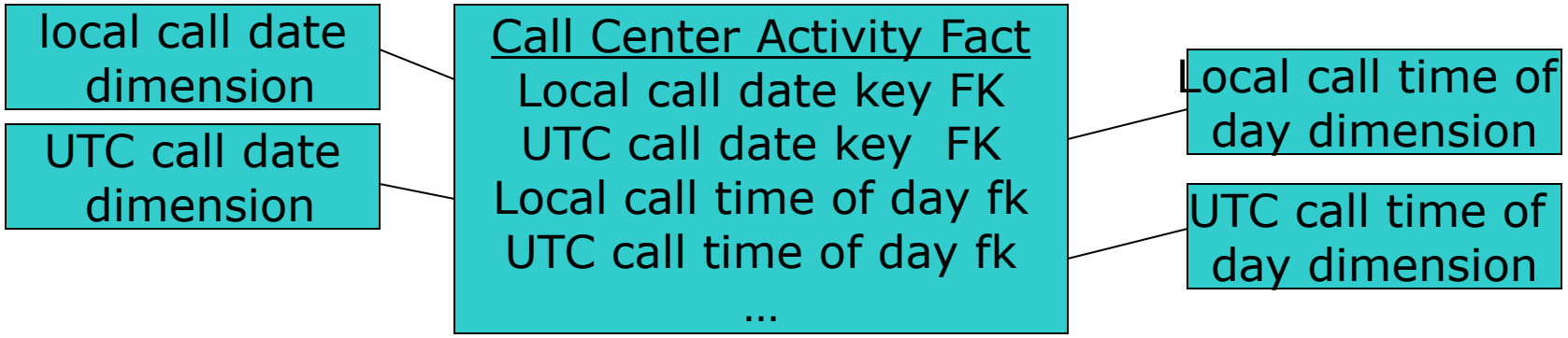
- Time of day should be treated as dimension only if there are meaningful textual descriptions for periods within the day
  - Example; lunch hour, rush hours, etc.
- Otherwise, time of day needs to be represented as a simple non-additive fact or a date/timestamp

Date/Timestamp

- Used in the fact table to support precise time interval calculated across fact rows
  - Calculations to be performed by ETL system
  - Example: elapsed time between original claim date and first payment date

Multiple Time Zones

- Express time in coordinated universal time (UTC)
- Additionally, may be expressed in local time
- Other options: use a single time zone (for example, ET) to express all times in this zone



Degenerate Dimensions

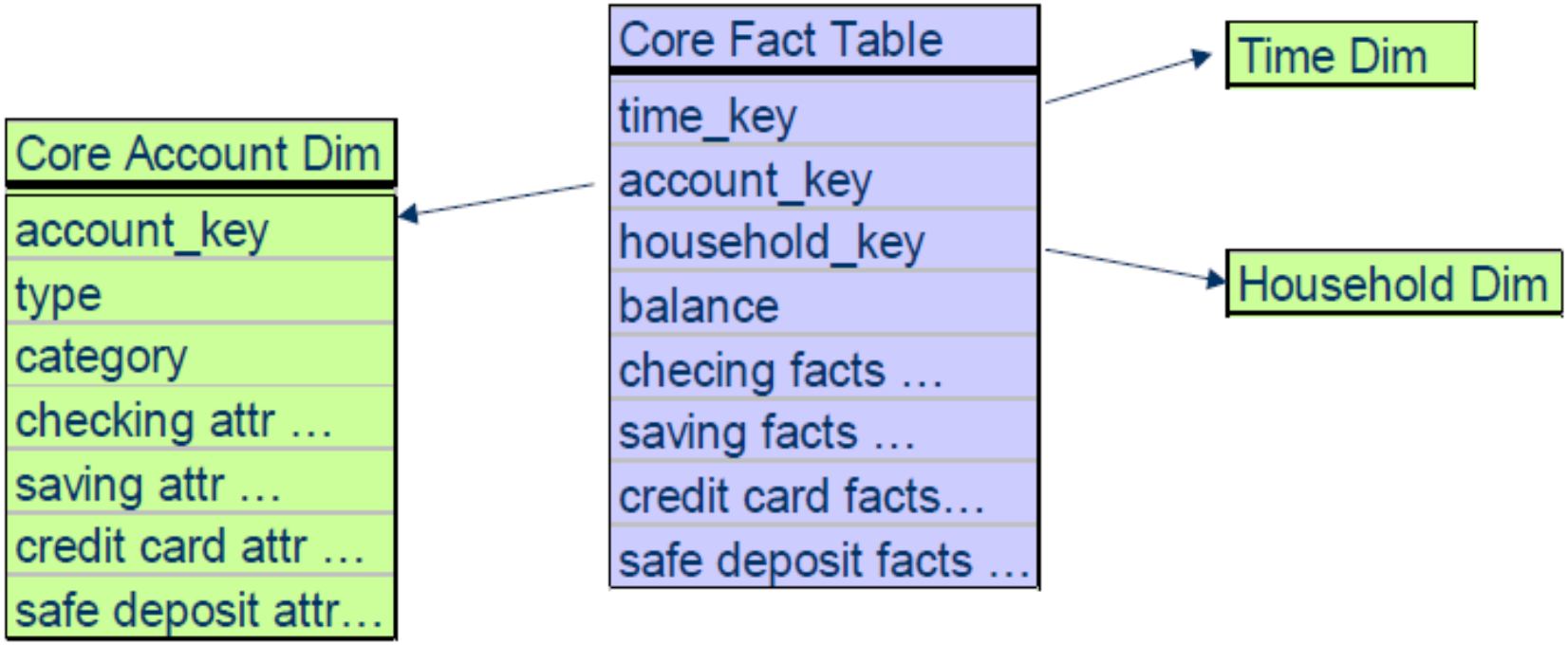
- Occur in transaction fact tables that have a natural parent-child structure
- Key remains the only attribute left after other attributes got separated into dimensions
- Key should be the actual transaction number
- Stored in a fact table - do not create a corresponding dimension table

Heterogeneous Products

Some products have many, many distinguishing attributes and many possible permutations (usually on the basis of some customised offer). This results in immense product dimensions and bad browsing performance

- In order to deal with this, fact tables with accompanying product dimensions can be created for each product type - these are known as custom fact tables
- Primary core facts on the products types are kept in a core fact table (but can also be copied to the conformed fact tables)

Heterogeneous Products



Heterogeneous Products