Aggregation

Dr. Yashvardhan Sharma

Computer Science Department

BITS, Pilani

L13

Aggregation

The single most dramatic way to affect performance in a large data warehouse is to provide a proper set of aggregate (summary) records ... in some cases speeding queries by a factor of 100 or even 1,000. No other means exist to harvest such spectacular gains."

- Ralph kimball

Consider the following questions!

- How much total business did my newly remodelled stores do compared with the chain average?
- How did leather goods items costing less than $5 do with my most frequent shoppers?
- What was the ratio of non-holiday weekend days total revenue to holiday weekend days?

Aggregation

- Aggregations can be created on-the-fly or by the process of pre-aggregation
- An aggregate is a **fact table record representing a** summarization of base-level fact table records
  - Category-level product aggregates by store by day
  - District-level store aggregates by product by day
  - Monthly sales aggregates by product by store
  - Category-level product aggregates by store district by day
  - Category-level product aggregates by store district by month

Aggregation

- Still aggregations is so underused. Why?
- We are still not comfortable with redundancy!
- Requires extra space
- Most of us are not sure of what aggregates to store
- A bizarre phenomena called

  SPARSITY FAILURE

Aggregates & Indexes

- Analogous??
- Indexes duplicate the information content of indexed columns
- We don’t disparage this duplication as redundancy because of the benefits
- Aggregates duplicate the information content of aggregated columns
- Traditional indexes very quickly retrieve a small no. of qualifying records in OLTP systems
- In DW, queries require millions of records to be summarized
- Bypassing indexes and performing table scans

Aggregates

- Need new indexes that can quickly and “logically” get us to millions of records
- Logically because we need only the summarized result and not the individual records
- Aggregates as *Summary Indexes!*

Aggregates

- Aggregates belong to the same DB as the low level atomic data that is indexed (unlike data marts)
- Queries should always target the atomic data
- Aggregate Navigation automatically rewrites queries to access the best presently available aggregates
- Aggregate navigation is a form of query optimization
- Should be offered by DB query optimizers
- Intelligent Middleware

Aggregation: Thumb Rule

The size of the database should not become more than double of its original size

Aggregates: Trade-Offs

- Query performance vs. Costs
- Costs
  - Storing
  - Building
  - Maintaining
  - Administrating
- Imbalance: Retail DW that collapsed under the weight of more that 2500 aggregates and that took more than 24 hours to refresh!!!

Aggregates: Guidelines

- Set an aggregate storage limit (not more than double the original size of the DB)
- Dynamic portfolio of aggregates that change with changing demands
- Define small aggregates: 10 to 20 times smaller than the FT or aggregate on which it is based
  - Monthly product sales aggregate: How many times smaller than daily product sales table?
- If your answer is 30…you are forgiven, but you are likely to be wrong
  - Reason: Sparstiy Failure

Aggregates: Guidelines

- Spread aggregates: Goal should be to accelerate a broad spectrum of queries

Aggregates: Guidelines

- Spread aggregates: Goal should be to accelerate a broad spectrum of queries

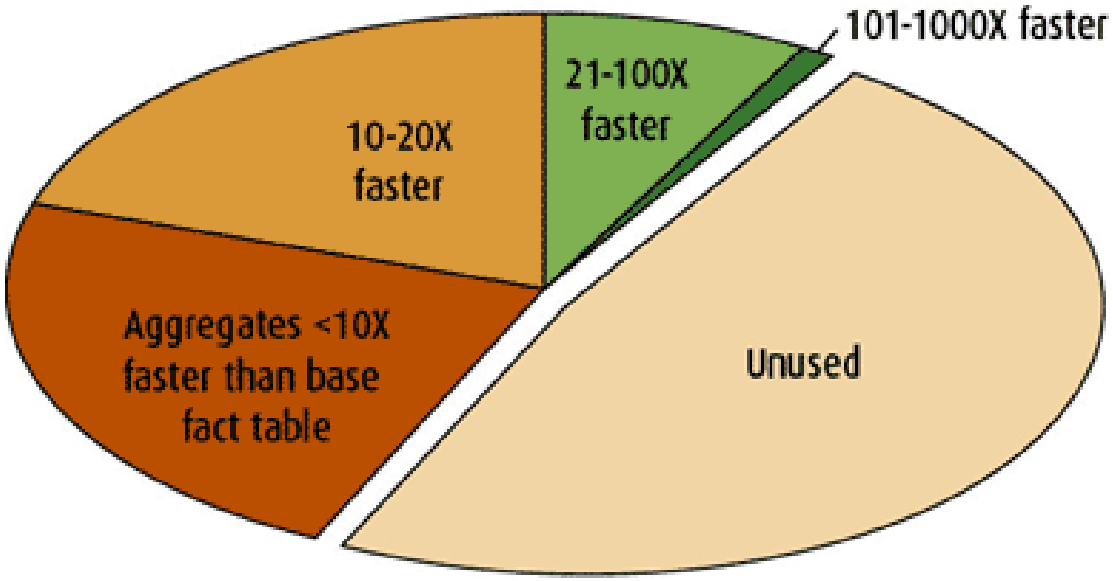


Figure 1 Poor use of the space allocated for aggregates.

Aggregation

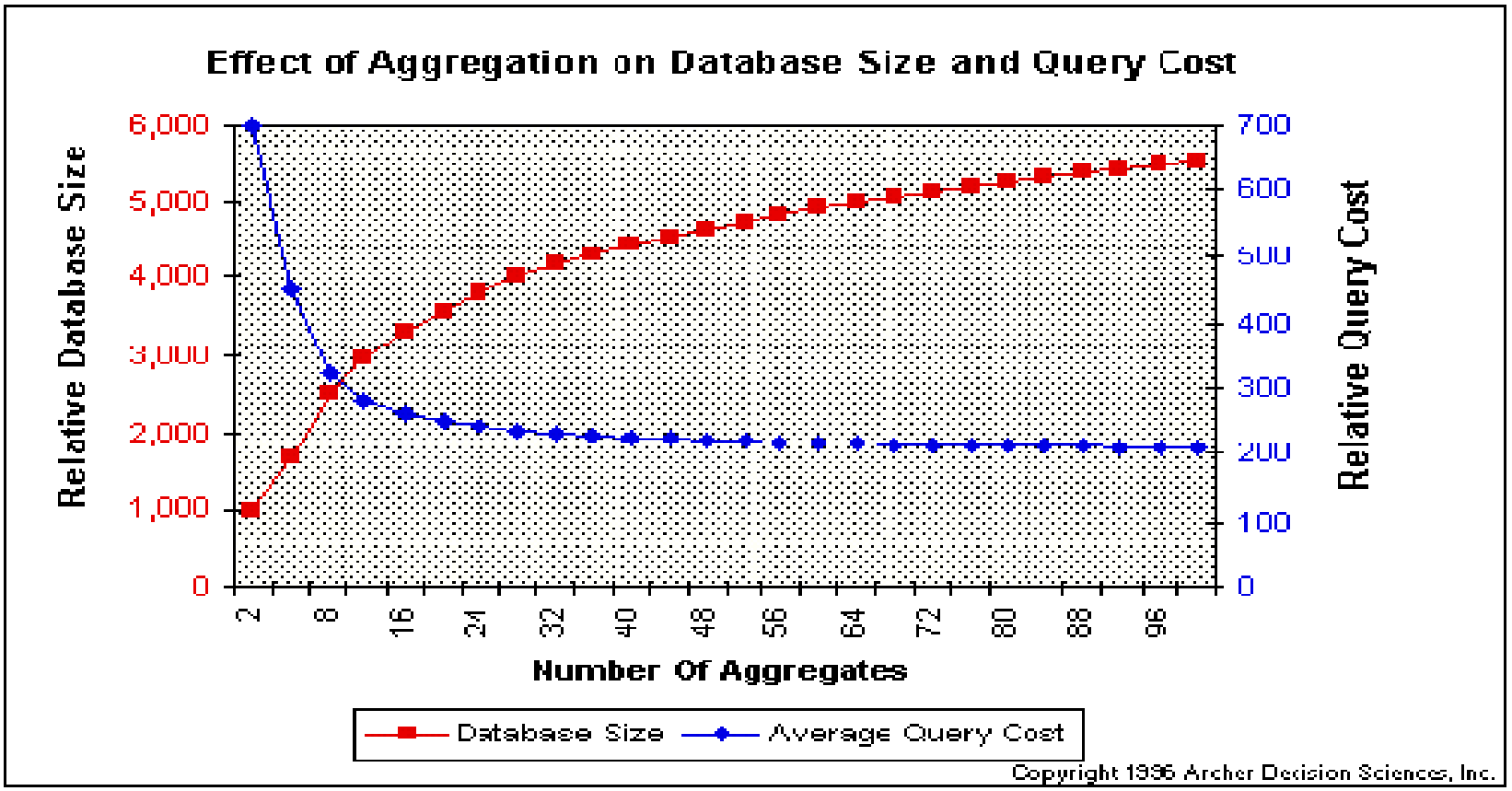


Figure taken from Neil Raden article (www.hiredbrains.com/artic9.html)

Star Schema: Grocery Store

Figure 11-12 STAR schema: grocery chain.

Some Queries

- Query: Total sales for customer number 12345678 during the first week of December 2003 for product Widget-1

- Find and sum the sales quantity and sales dollars for all fact table rows where the customer key relates to customer number 12345678, the product key relates to product Widget-1, and the time key relates to the seven days in the first week of December 2003.
  - Assuming a customer can make a single purchase on a given day, only seven rows of the fact table will be summed

Some Queries

- Query: total sales for all customers in the south-central territory for the first two quarters of 2003 for product category Bigtools
- All fact table rows where the customer key relates to all customers in the south-central territory, the product key relates to all products in the product category Bigtools and the time key relates to about 180 days in the first two quarters of 2003.
  - In this query, clearly a large number of fact table rows participate the summation
- How can we reduce the execution time?

Fact Table Size

- Credit card transaction tracking

- Time dimension: 5 years = 60 months
- Number of credit card accounts: 150 million
- Av. number of monthly transaction/account: 20
- Max. number of base fact table records: 180 billion

Aggregating Fact Tables

- Typically, queries require detailed data on some dimensions, while only summary data is needed for the other dimensions
- Example: assume one sale per product per store per week. Estimate the number of fact table rows required:
  - Query involves 1 product, 1 store, 1 week
  - Query involves 1 product, all stores, 1 week
  - Query involves 1 brand, 1 store, 1 week
  - Query involves 1 brand, all stores, 1 year
  - Suppose now you have an aggregate fact table where each row summarizes the totals for a brand, a store, and a week. Now estimate the number of fact table rows required.

Multi-way Aggregates

- Utilize hierarchies in dimensions to create appropriate aggregate fact tables
- Single-way aggregate fact table aggregates along one dimension only; multi-way have more than one dimension aggregated

Multi-way Aggregates

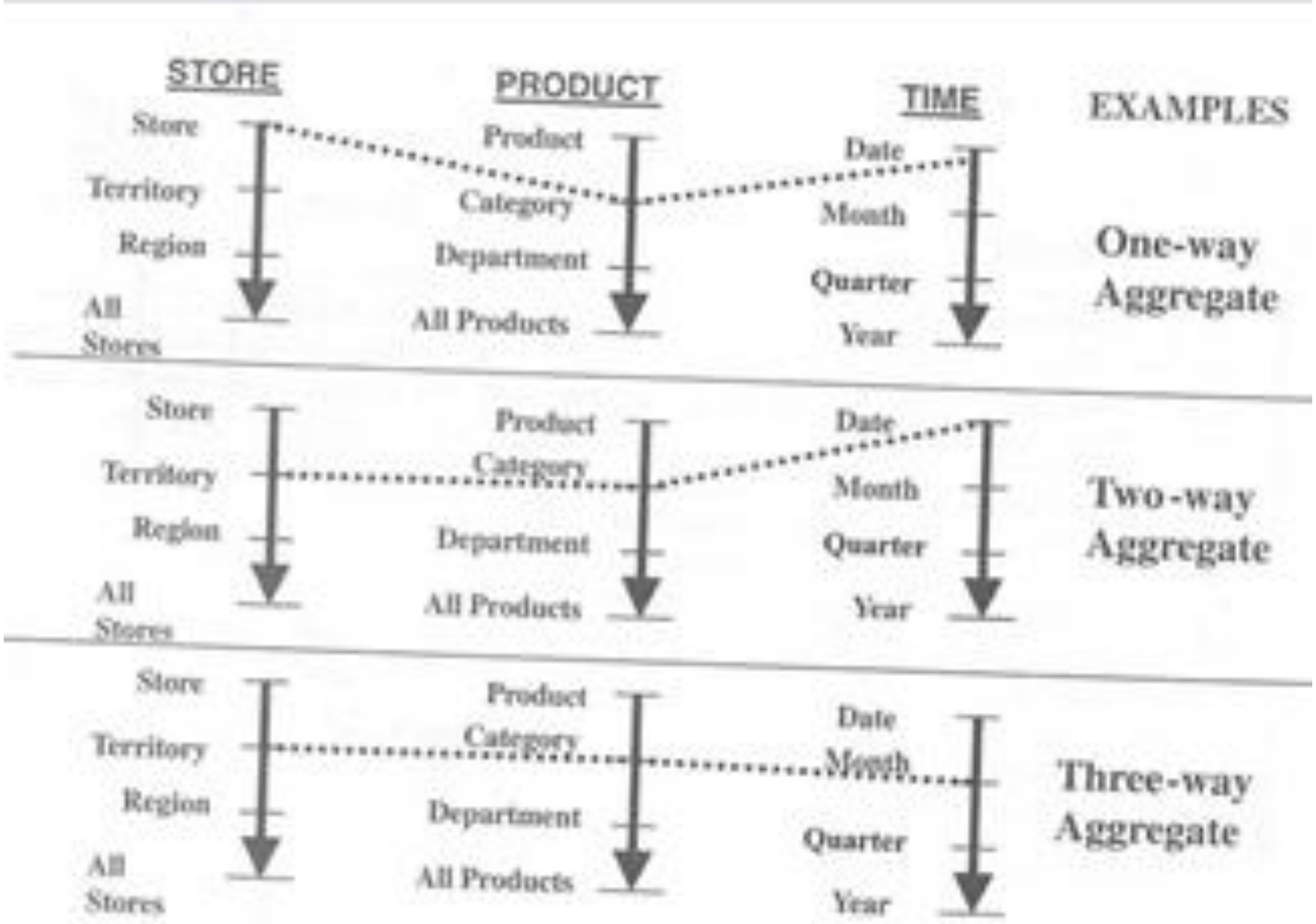


Figure 11-14 Forming aggregate fact tables.

Multi-way Aggregates

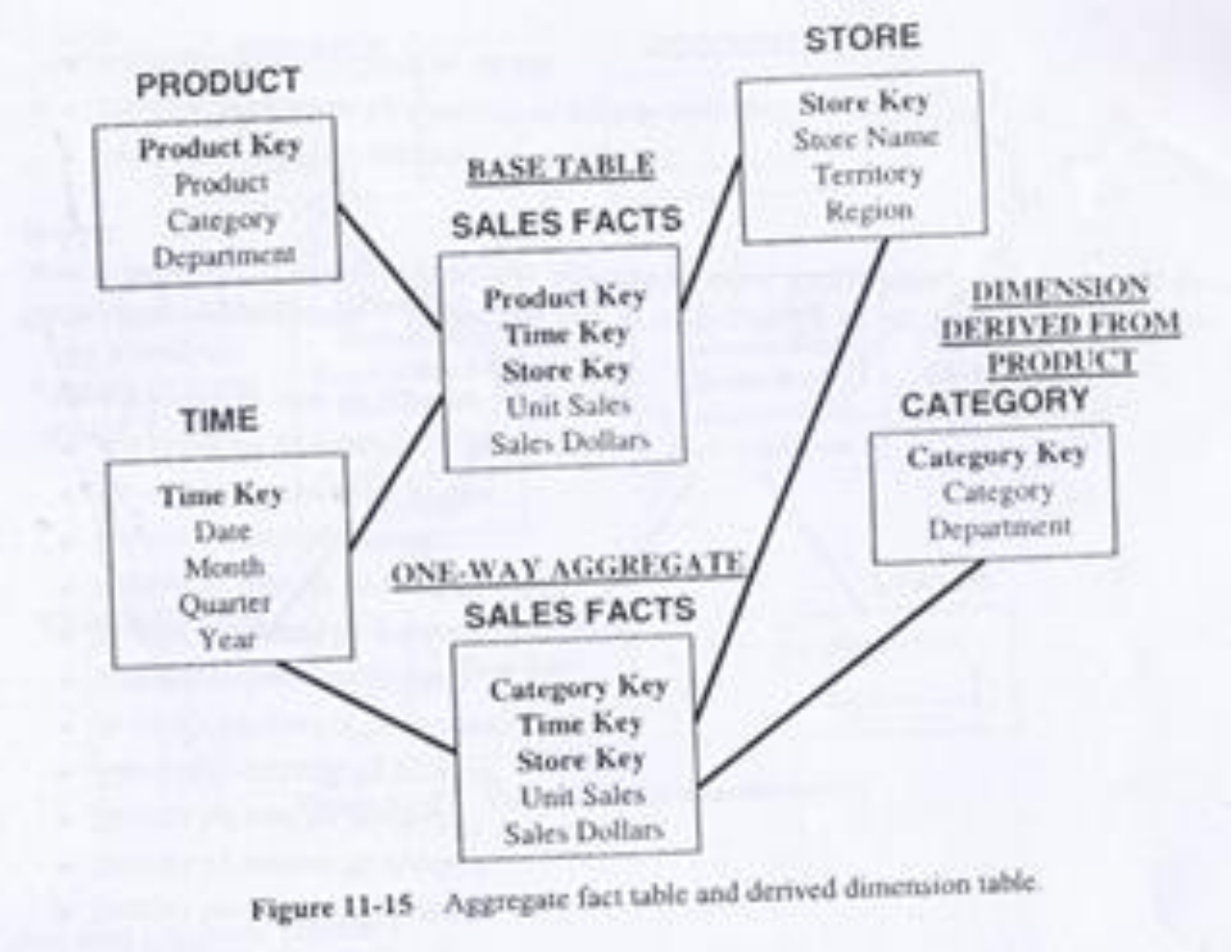


Figure 11-15 Aggregate fact table and derived dimension table

Goals of Aggregation

- Primary goal: improve overall DW performance
- Do not get bogged down with too many aggregates. Remember you have to create addition derived dimensions to support the aggregates
- Try to cater to a wide range of user groups
- Go for aggregates that do not unduly increase the overall usage of storage
- Keep the aggregates hidden from the end-users

Aggregation

- Aggregations can be created on-the-fly or by the process of pre-aggregation
- An aggregate is a **fact table record representing a** summarisation of base-level fact table records
  – Category-level product aggregates by store by day
  – District-level store aggregates by product by day
  – Monthly sales aggregates by product by store
  – Category-level product aggregates by store district by day
  – Category-level product aggregates by store district by month

Aggregates

Issues

- Which aggregates to create?
- How to guard against sparsity failure?
- How to store them?
  - New Fact Table approach
  - New Level Field approach
- How queries are directed to appropriate aggregates?

Aggregates: Example

- Grocery Store
- 3 dimensions – Product, Location, & Time
- 10000 products
- 1000 stores
- 100 time periods
- 10% Sparsity

Total no. of records = 100 million

Aggregates: Example

Hierarchies

- 10000 products in 2000 categories
- 1000 stores in 100 districts
- 30 aggregates in 100 time periods

Aggregates: Example

How many aggregates are possible?

- 1-way: Category by Store by Day
- 1-way: Product by District by Day
- 1-way: Product by Store by Month
- 2-way: Category by District by Day
- 2-way: Category by Store by Month
- 2-way: Product by District by Month
- 3-way: Category by District by Month

Aggregates: Example

What is Sparsity?

- Fact tables are sparse in their keys!
- 10% sparsity at base level means that only 10% of the products are sold on any given day (average)
- As we move from base level to 1-way the sparsity _ _ _ _ _ _ _ _ **Increases!**
- What affect sparsity will have on the size of the aggregate fact table?

Sparsity Failure

- The planning of aggregate table sizes can be tricky because of the phenomenon called sparsity failure
- This phenomenon appears when we build aggregates on sparse tables.
- For example: In the grocery store item movement database, only about 10% of the products in the store are actually sold in a given store on a given day. Even disregarding the promotion dimension, the database is only occupied 10% in the primary keys of product, store, and time. However when we build aggregates, the occupancy rate shoots up dramatically.

Date_key	P_Key	$ sold
1-May	P12	100
1-May	P11	200
1-May	P25	300
2-May	P12	250
3-May	P12	100
4-May	P13	50
4-May	P24	150

Sparsity Failure

- Assume we have:
- 10,000 products within 2,000 different brands
- 1,000 stores within 100 districts
- 100 weeks within 30 months

Table	Product	Store	Time	Sparsity	# Records
Base: product by store by week	10,000	1,000	100	10%	100 milion
One-way: brand by store by week	2,000	1,000	100	50%	100 milion
One-way: product by district by week	10,000	100	100	50%	50 milion
One-way: product by store by month	10,000	1,000	30	50%	150 milion
Two-way: brand by district by week	2,000	100	100	80%	16 milion
Two-way: brand by store by month	2,000	1,000	30	80%	48 milion
Two-way: product by district by month	10,000	100	30	80%	24 milion
Three-way: brand by district by month	2,000	100	30	100%	6 milion
				Grand Total	494 milion

Aggregates: Example

- Let us assume that sparsity for 1-way aggregates is 50%
- For 2-way 80%
- For 3-way 100%
- Do you agree with this?
- Is it logical?

Aggregates: Example

Table	Prod.	Store	Time	Sparsity	# Records
Base	10000	1000	100	10%	100 million
1-way	2000	1000	100	50%	100 million
1-way	10000	100	100	50%	50 million
1-way	10000	1000	30	50%	150 million
2-way	2000	100	100	80%	16 million
2-way	2000	1000	30	80%	48 million
2-way	10000	100	30	80%	24 million
3-way	2000	100	30	100%	6 million
				Grand Total	494 million

Aggregates: Example

- An increase of almost 400%
- Why it happened?
- Look at the aggregates involving Location and Time!
- How can we control this aggregate explosion?
- Do the calculations again with 500 categories and 5 time aggregates

Aggregates: Example

Table	Prod.	Store	Time	Sparsity	# Records
Base	10000	1000	100	10%	100 million
1-way	500	1000	100	50%	25 million
1-way	10000	100	100	50%	50 million
1-way	10000	1000	5	50%	25 million
2-way	500	100	100	80%	4 million
2-way	500	1000	5	80%	2 million
2-way	10000	100	5	80%	4 million
3-way	500	100	5	100%	0.25 million
				Grand Total	210.25 million

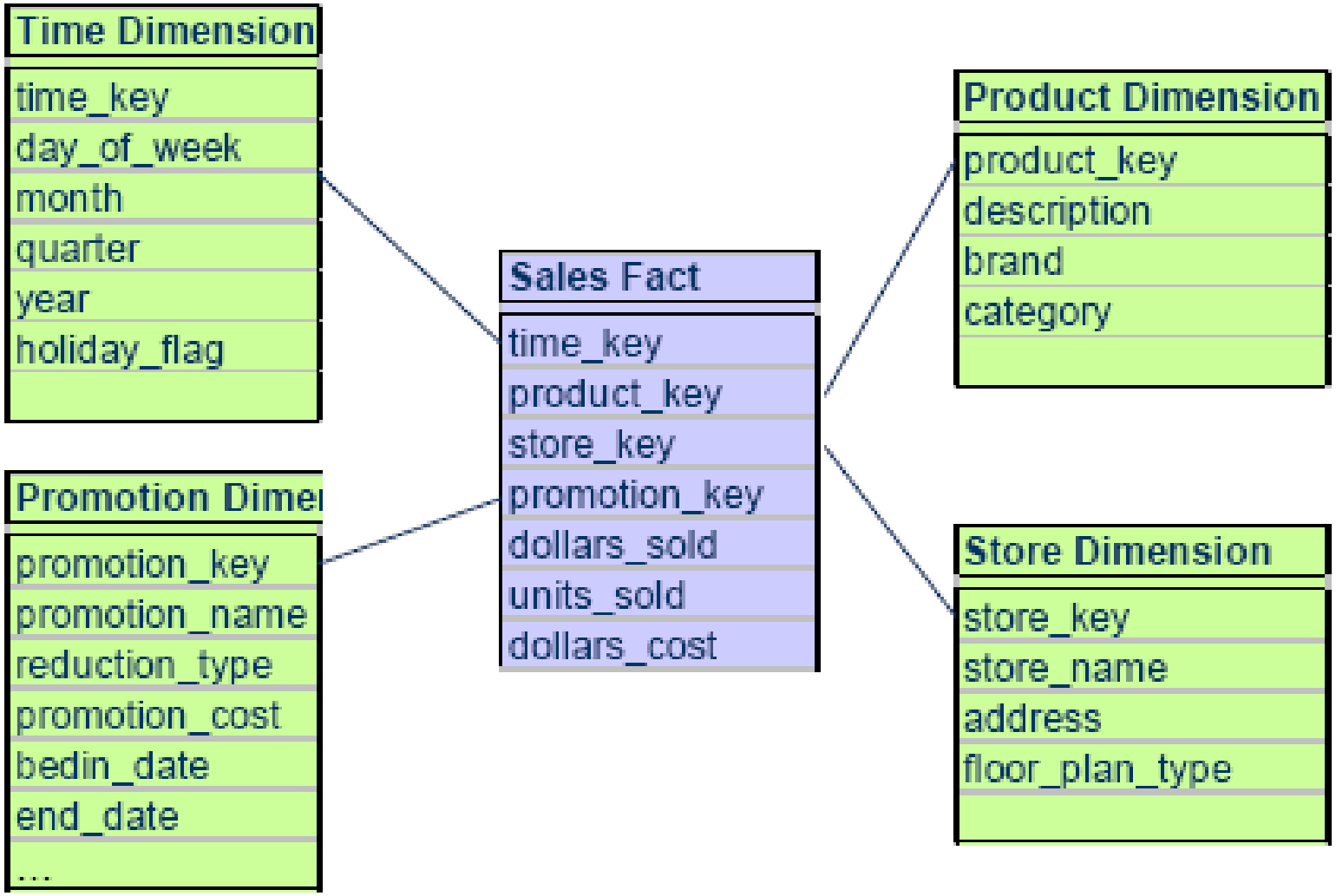
Aggregate Design Principle

Each aggregate must summarize at least 10 and preferably 20 or more lower-level items

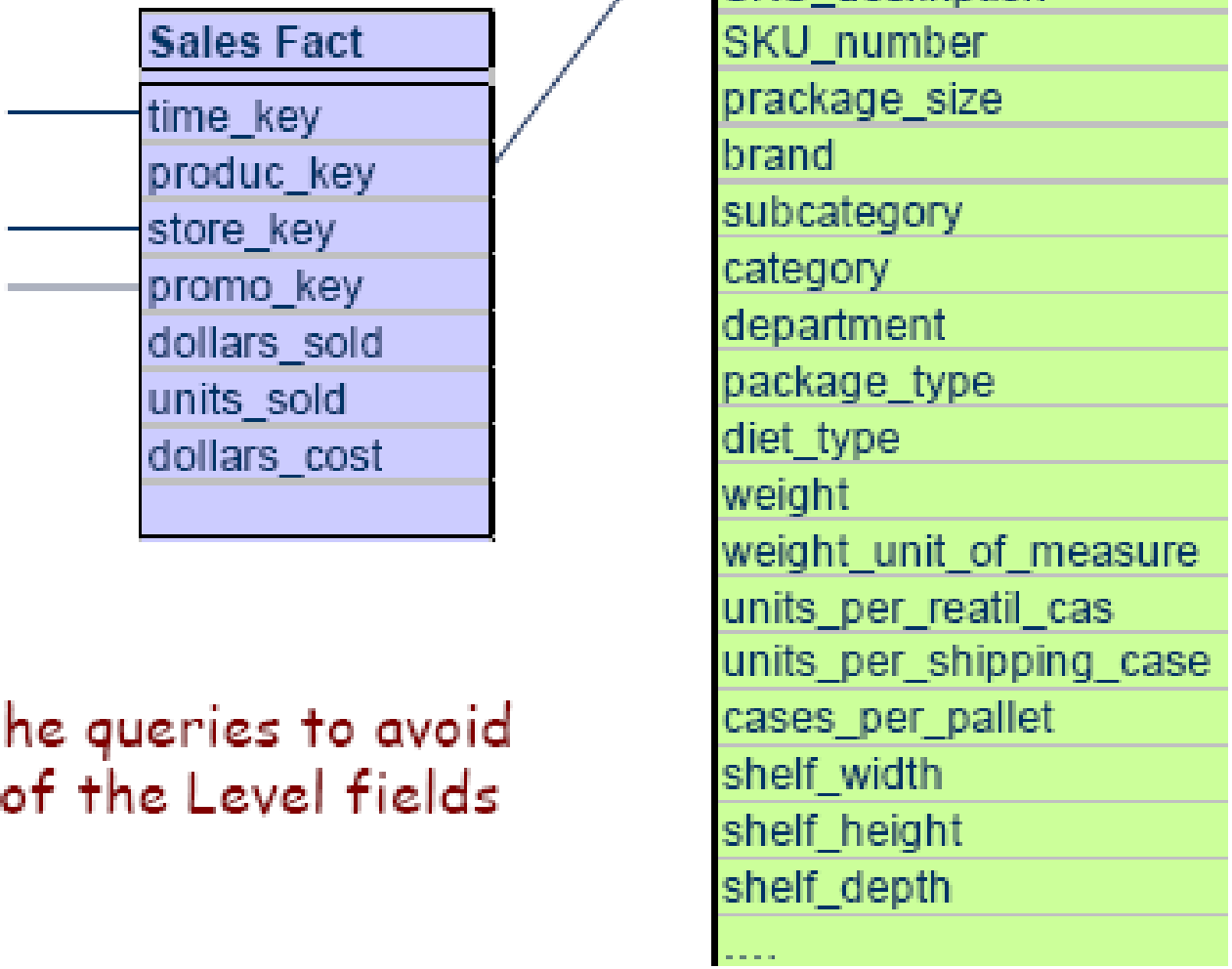
How to store aggregates?

- as new Level fields in an already existing Fact table
- as new fact tables

Example of a Dimensional Model



New Level field for Aggregates



NB! Constraint the queries to avoid double counting of the Level fields

An example

Key	Produc name	Subcategory	Category
P11	white napkin	napkin	paper
P12	pink napkin	napkin	paper
P13	red napkin	napkin	paper
P24	Eko tissue	tissue	paper
P25	Lambi tissue	tissue	paper

Date_key	P_Key	$ sold
1-May	P12	100
1-May	P11	200
1-May	P25	300
2-May	P12	250
3-May	P12	100
4-May	P13	50
4-May	P24	150
1-May	P10	300

? $sold napkin/day
? $sold tissue/day
? $sold paper/day

New Level field for Aggregates - an example

P_Key	Produc name	Subcategory	Category	LEVEL
P11	white napkin	napkin	paper	base
P12	pink napkin	napkin	paper	base
P13	red napkin	napkin	paper	base
P24	Eko tissue	tissue	paper	base
P25	Leni tissue	tissue	paper	base

Date_key	P_Key	$ sold
1-May	P12	100
1-May	P11	200
1-May	P25	300
2-May	P12	250
3-May	P12	100
4-May	P13	50
4-May	P24	150

? $sold napkin/day
? $sold tissue/day
? $sold paper/day

Well, is this a solution you would chose?

New Tables

Product

P_Key	Produc name	Subcategory	Category	LEVEL
P11	white napkin	napkin	paper	base
P12	pink napkin	napkin	paper	base
P13	red napkin	napkin	paper	base
P24	Eko tissue	tissue	paper	base
P25	Leni tissue	tissue	paper	base

Subcategory_product

SK	Subcat	Category
P10	napkin	paper
P20	tissue	paper

Category_product

CK	Category
P100	paper

Base_sales_fact

Date_key	P_Key	$ sold
1-May	P12	100
1-May	P11	200
1-May	P25	300
2-May	P12	250
3-May	P12	100
4-May	P13	50
4-May	P24	150
1-May	P10	300

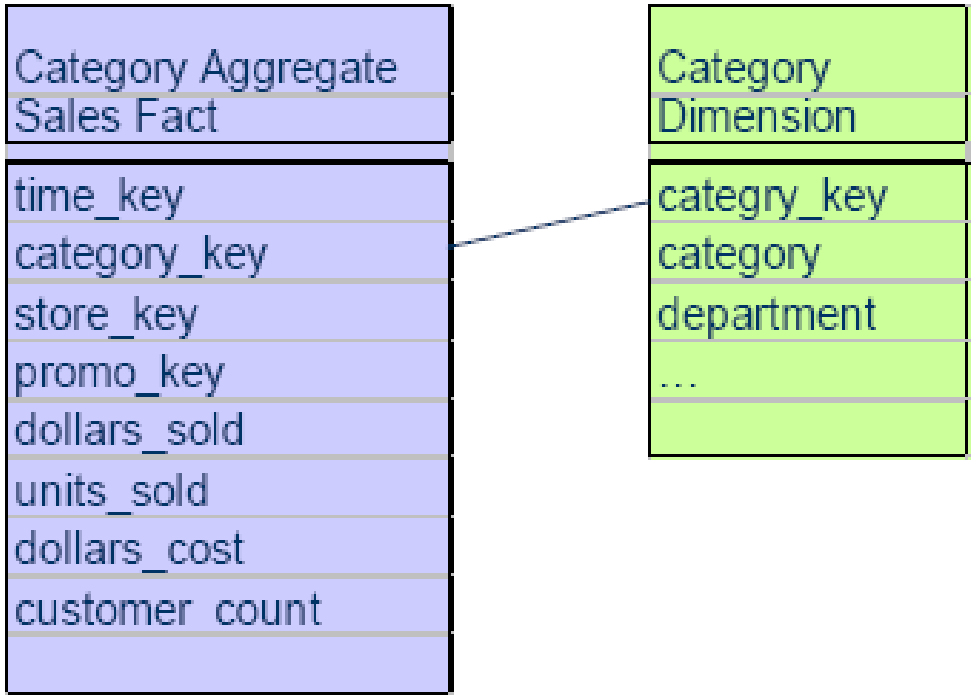
Subcategory_sales_fact

Date_key	SK	$ sold
1-May	P20	300
2-May	P10	250
3-May	P10	100
4-May	P10	50
4-May	P20	150

Category_sales_fact

Date_key	CK	$ sold
1-May	P100	600
2-May	P100	250
3-May	P100	100
4-May	P100	200

New Fact Tables for Aggregates



The creation of aggregate fact table requires the creation of:

- a derivative dimension
- an artificial key for each new derivative dimension

How to store aggregates

- as new Level fields in an already existing Fact table
  - problems with double count
  - visible for the users
- as new fact tables

  + no problems with double count

  + invisible for the users

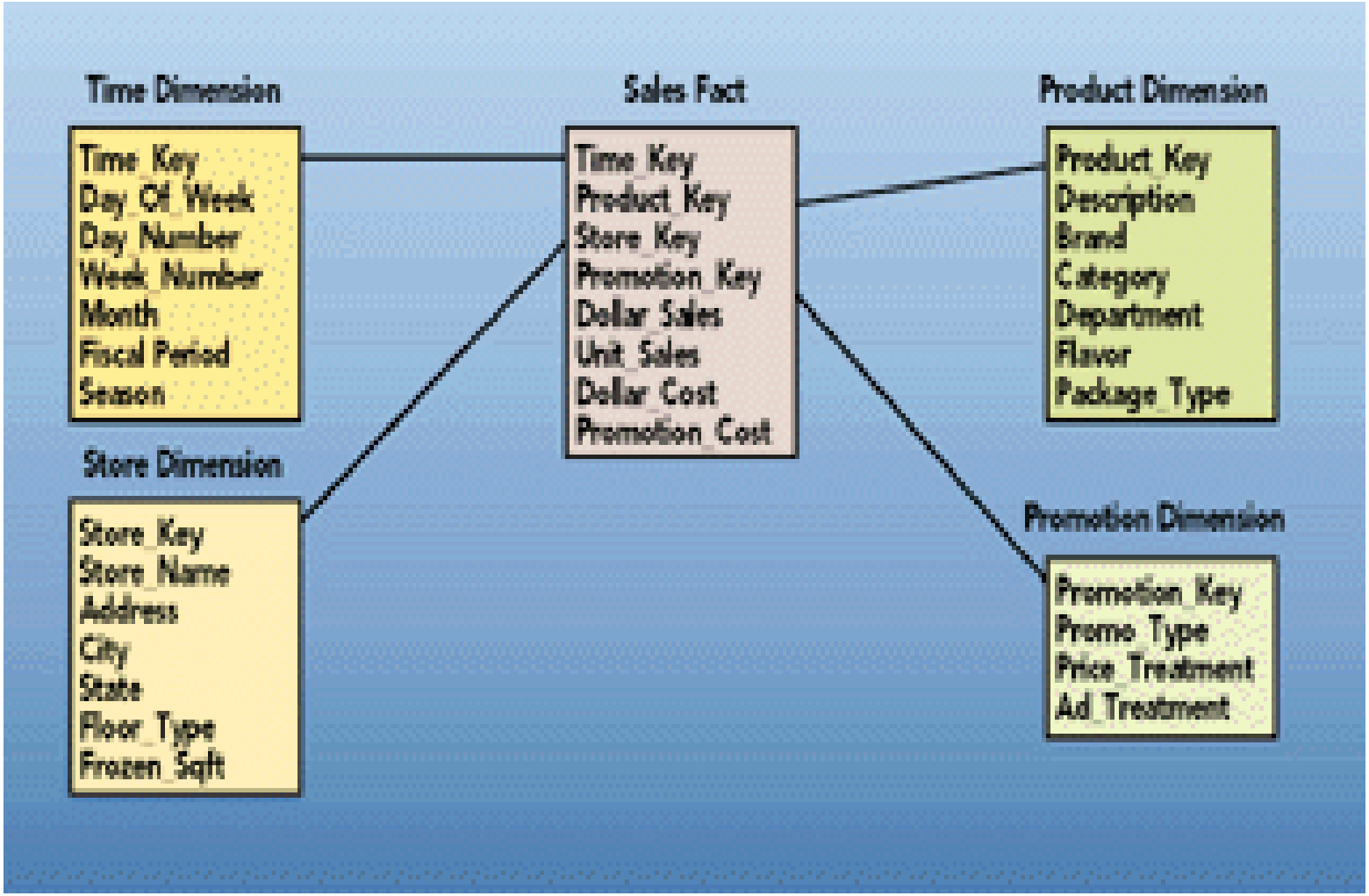
  + are easily introduced and/or reduced at different points in time

  + simpler metadata

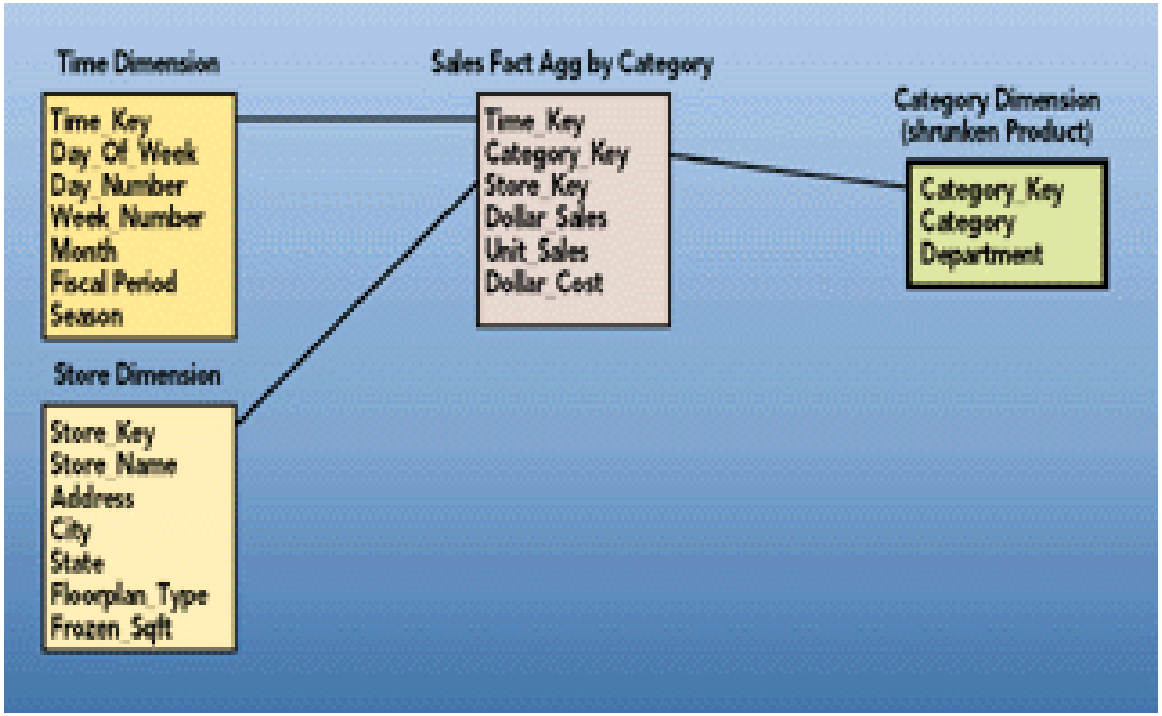
  + simpler choice of key

  + the size of the field for the summarised data does not increase the size of the field for the basic data

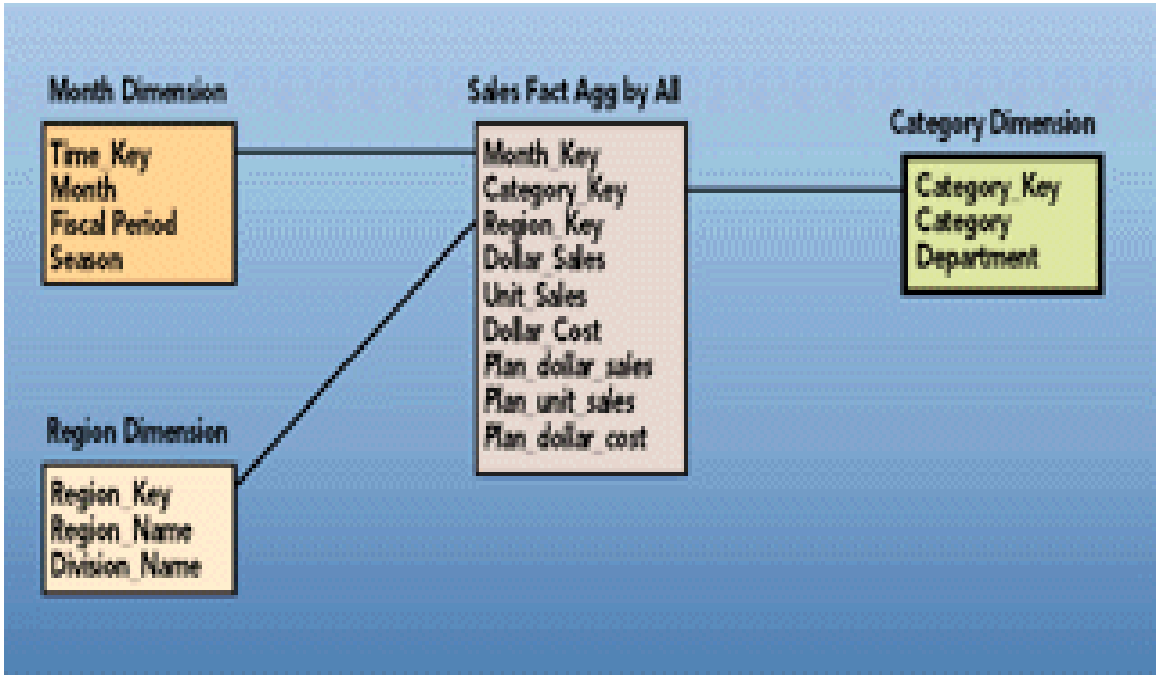
Aggregates


Time Dimension
Time_Key
Day_Of_Week
Day_Number
Week_Number
Month
Fiscal Period
Season
Sales Fact
Time_Key
Product_Key
Store_Key
Promotion_Key
Dollar_Sales
Unit_Sales
Dollar_Cost
Promotion_Cost
Product Dimension
Product_Key
Description
Brand
Category
Department
Flavor
Package_Type
Store Dimension
Store_Key
Store_Name
Address
City
State
Floor_Type
Frozen_Sqft
Promotion Dimension
Promotion_Key
Promo_Type
Price_Treatment
Ad_Treatment

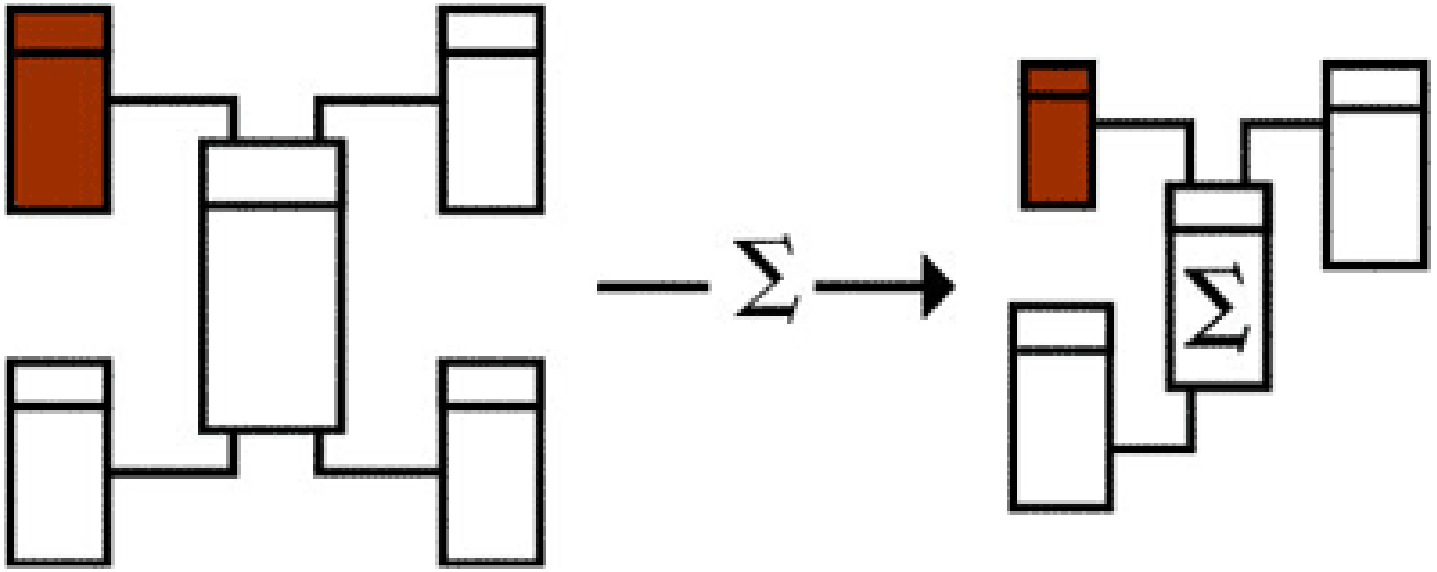
Aggregates:
Shrunken Dimensions



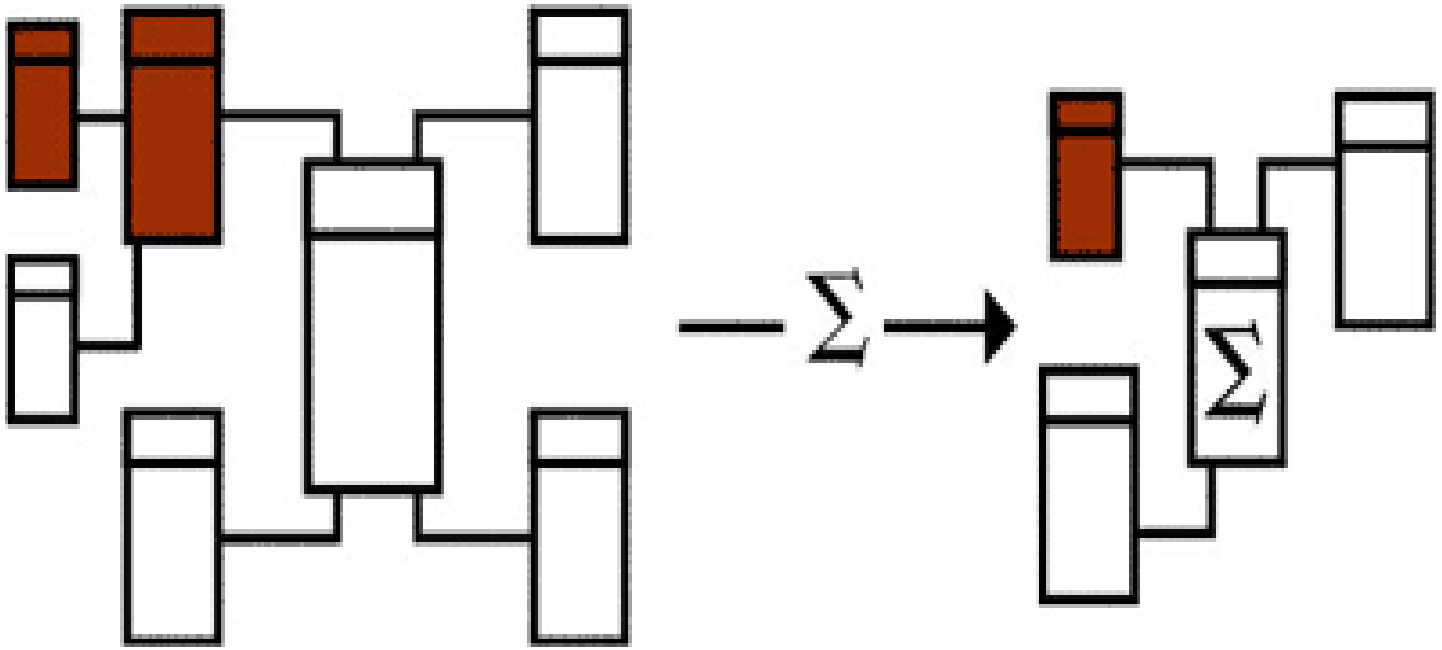
Aggregates:
Shrunken Dimensions



Aggregates:
Shrunken Dimensions



Aggregates:
Shrunken Dimensions



Aggregate Navigator

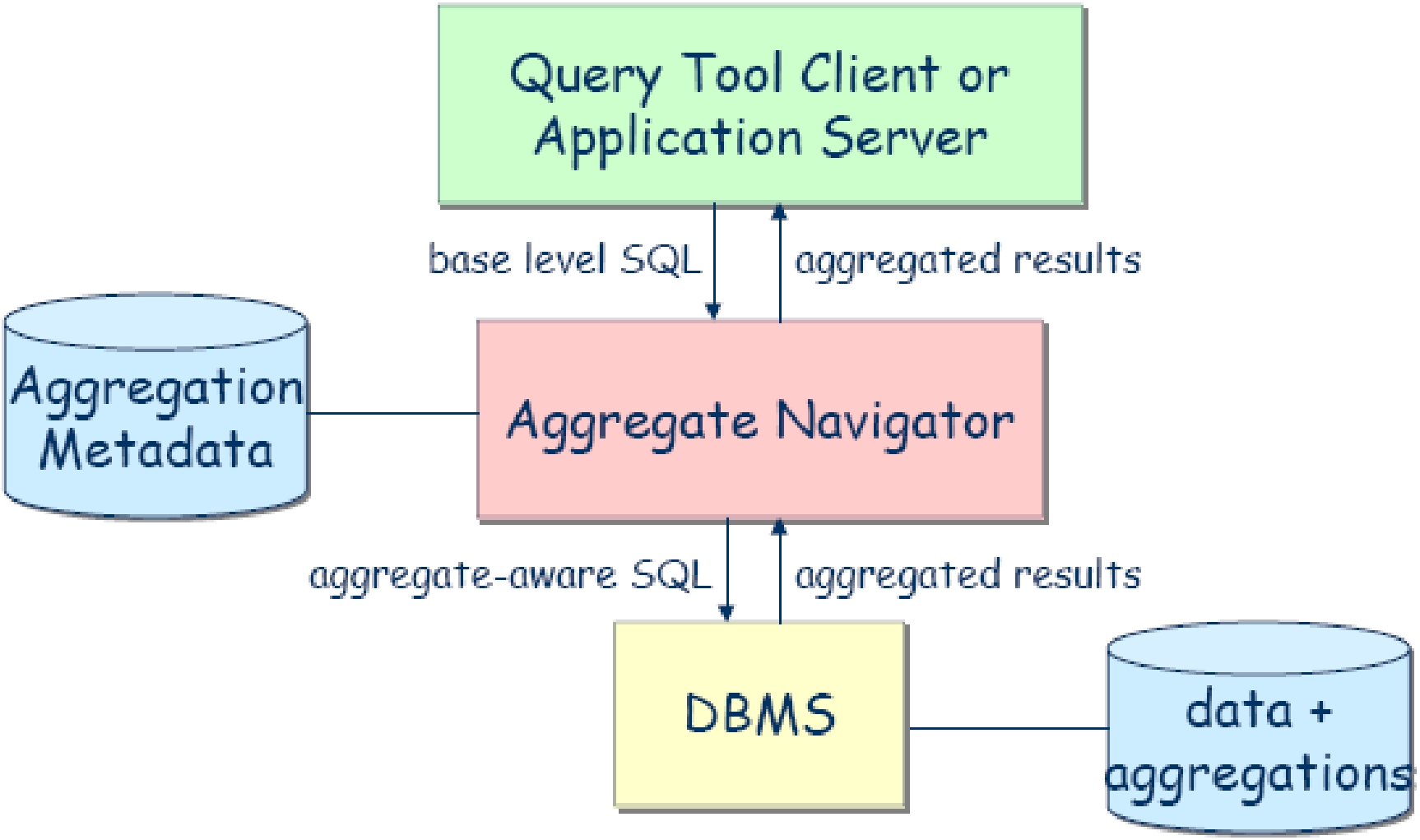
- How queries are directed to the appropriate aggregates?
- Do end user query tools have to be “hardcoded” to take advantage of aggregates?
- If DBA changed the aggregates all end user applications have to be recoded

How do we overcome this problem?

Aggregate Navigator

- Aggregate Navigator (AN) is the solution
- So what is an AN?
- A middleware sitting between user queries and DBMS
- With AN, user applications speak just base level SQL
- AN uses metadata to transform base level SQL into “Aggregate Aware” SQL

Aggregation Navigator



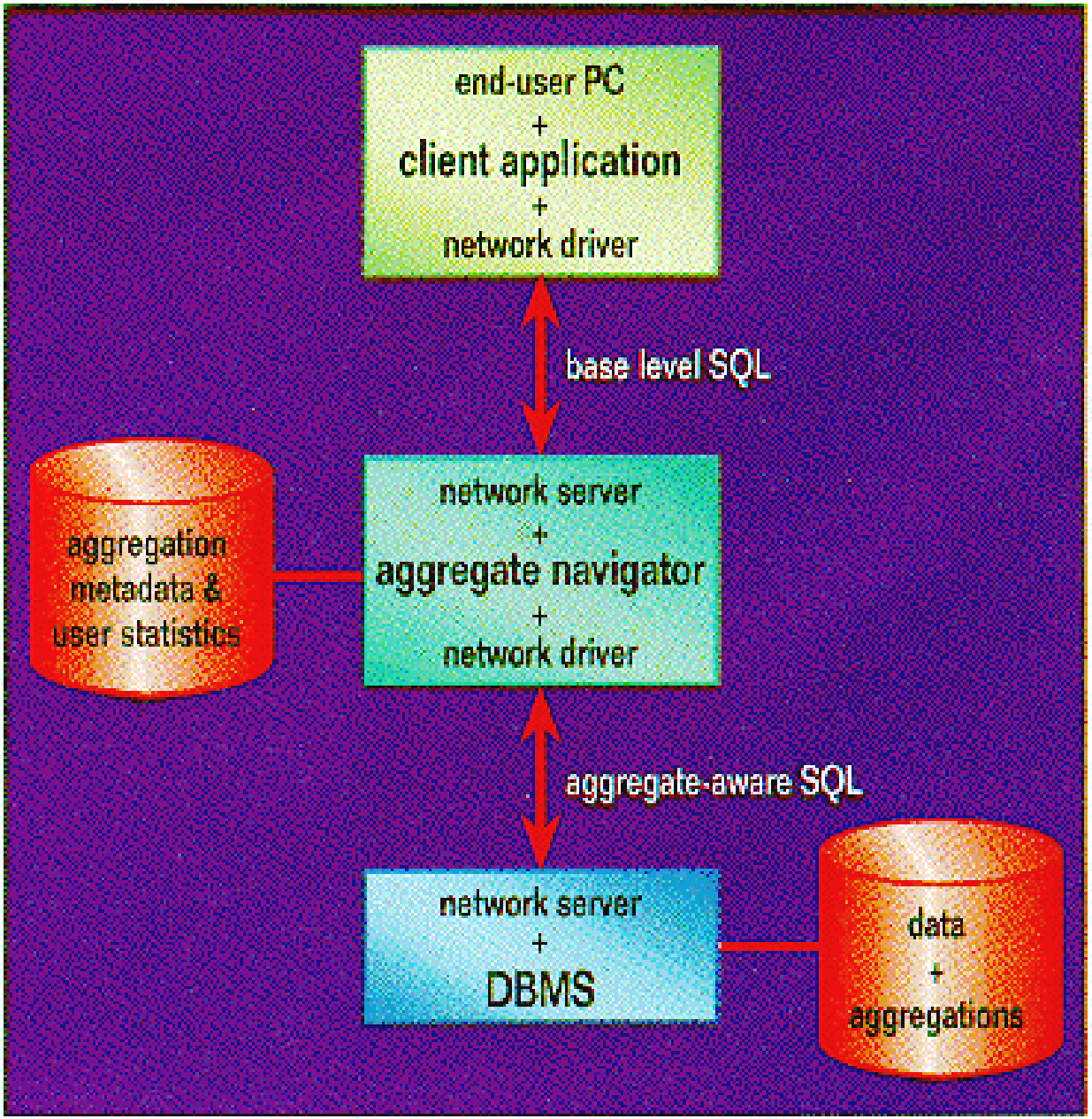


FIGURE 1 The aggregate navigator sits between the application and DBMS, and intercepts the end-user's SQL.

AN Algorithm

1. Rank Order all the aggregate fact tables from the smallest to the largest
2. For the smallest FT, look in associated DTs to verify that all the dimensional attributes of the current query can be found.If found, we are through. Replace the base-level FT with the aggregate fact and aggregate DTs
3. If step 2 fails, find the next smallest aggregate FT and try step 2 again. If we run out of aggregate FTs, then we must use base tables

Design Requirements

#1 Aggregates must be stored in their own fact tables, separate from the base-level data. In addition, each distinct aggregation level must occupy its own unique fact table

#2 The dimension tables attached to the aggregate fact tables must, wherever possible, be shrunken versions of the dimension tables associated with the base fact table.

Design Requirements

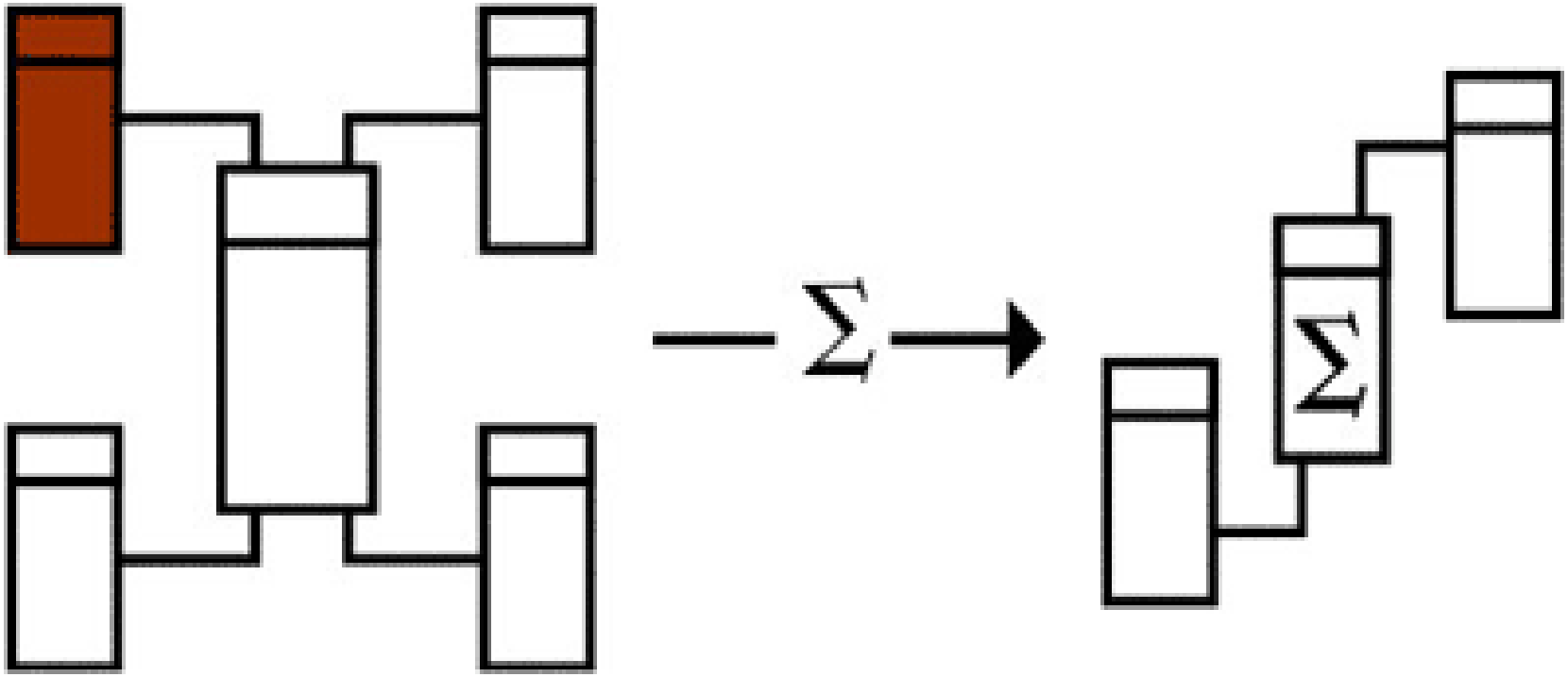
#3 The base Fact table and all of its related aggregate Fact tables must be associated together as a "family of schemas" so that the aggregate navigator knows which tables are related to one another.

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

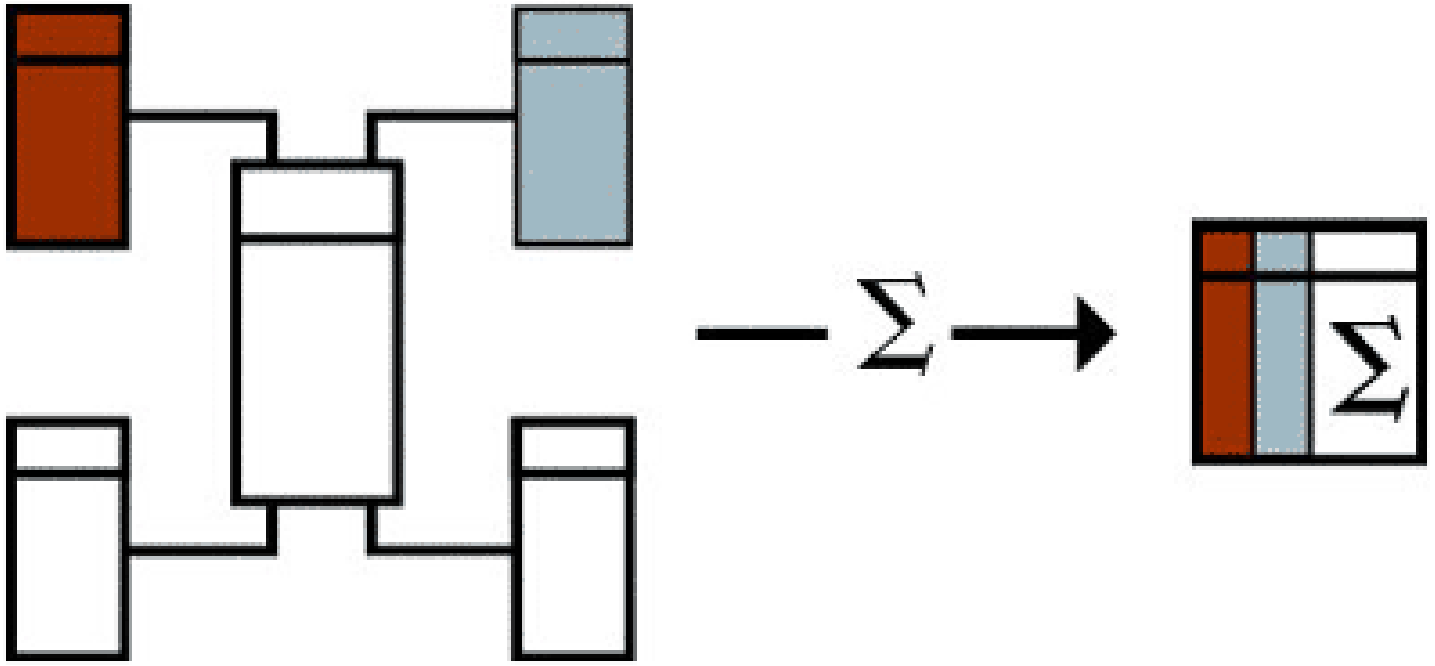
Storing Aggregates

- New fact and dimension table approach (Approach 1)
- New Level Field approach (Approach 2)
- Both require same space?
- Approach 1 is recommended
- Reasons?

Lost Dimensions



Collapsed Dimensions



Aggregations

- Effective way to augment the performance of the data warehouse if you augment basic measurements with aggregate information
- Aggregates speed queries by a factor of 100 or even 1000
- The whole theory of dimensional modeling was born out of the need of storing multiple sets of aggregates at various grouping levels within the key dimensions
- You can store aggregates right into fact tables in the Data Warehouse or (more appropriately) the Data Mart