Modelling a Data Warehouse

Overview

- Steps for modelling a DWH
- Data granularity
- Data storage
- Attribute hierarchies
- Querying a DWH / OLAP
- Frequent mistakes when building a DWH
- Example: grocery store (R. Kimball [KIM96])

Modelling a Data Warehouse

Example: Grocery Store

- Grocery chain with 500 stores spread over 3 states in the US
- Stores: supermarkets with departments like grocery, dairy, meat, frozen food, bakery, liquor, drugs etc.
- About 60.000 products in each store

(R. Kimball, [KIM96])

Modelling a Data Warehouse

Basics

- Data from source systems: OLTP, legacy systems, syndicated data
- Cleaned - within itself consistent data

Data available -> building a Data Warehouse:

- Which business processes to model
- Defining granularity of data to be fed into the DWH
- Modelling the DWH data structure for storing the data
- Transforming data according to DWH structure
- Calculating aggregations and derived attributes

Modelling a Data Warehouse

Which Business Processes to Model

DWH represent a business process view of the underlying data as opposed to the transaction-oriented OLTP systems

The decision which business processes to model has serious effects on the resulting data warehouse

- Problems to be addressed
- Questions to be asked
- Information needed and available
- Central DWH or data marts?

Modelling a Data Warehouse

Example: Business Process

OLTP - data:

- Point-of-Sales data (POS)
- Vendor delivery data
- Accounting data
- Customer data
- Promotions
- ...

Goal: build a **daily item movement** database

Modelling a Data Warehouse

Granularity

- Data is fed into the DWH at a certain level of granularity
- Based on this level of granularity aggregations can be defined
- higher granularity - more data, lower granularity - less data

Questions:

- Which levels of granularity are available?
- Which levels of granularity are reasonable and useful in the DWH (temperature sensor data: per millisecond, second, minute, hour?)

Tendency to store highest-granularity data where possible - once the granularity has been reduced, detailed information is no longer recoverable

Modelling a Data Warehouse

Example: Granularity

Which granularity for POS data?
Possibilities:

- single transactions per customer per product per store
- group transactions per customer per product per store
- group transactions per store per product per day
- group transactions per store per product per week
- group transactions per day per productgroup per region
- ...

Goal: daily item movement database

transactions per day per product per store

Modelling a Data Warehouse

Example: Granularity (2)

Granularity: **transactions per day per product per store**

- Not per customer per product per store because customers cannot be uniquely identified which would be essential for market basket analysis
- Not per week or month because we loose differences in e.g. Monday vs. Saturday sales
- Not per productgroup because of loss of all information relating to special brands, promotion campaigns

Note: these decisions depend a lot on the business process to be modelled and the questions to be answered!

Modelling a Data Warehouse

Ways of Storing the Data

- Data used for OLAP analysis must be stored in some kind of database to be accessed by the OLAP engine
- MOLAP?
- ROLAP?
- HOLAP ?
- Data Marts?

Modelling a Data Warehouse

Example: Storage for Grocery DWH

- Relational databases widely available
- Relational databases used for OLTP systems at companies
- Experienced IT personnel at companies used to relational databases
- ROLAP approach currently most common

Relational database used for storing Grocery DWH data

Modelling a Data Warehouse

Facts and Dimensions

Facts:

- Represent primary business process areas
- Unlikely to change once they are generated
- Stored at a certain level of granularity

Dimensions:

- Reference information by which facts can be structured for analysis
- Define aggregation hierarchies

Modelling a Data Warehouse

Example: Facts and Dimensions

Grocery Store, POS-Data

Facts:

- POS: sales facts

Dimensions:

- Time
- Store
- Promotion
- Product

Modelling a Data Warehouse

ROLAP Storage Schema- Schema

Star Schema:
- Partitioning the data
- Denormalizing tables
- One central fact table is surrounded by several dimension tables
- Queries address the fact table and are structured using the dimension tables
- No need for joins spanning multiple tables
- Most prominent model for DWH

Snowflake Schema:
- Based on star - schema
- Fact table structure identical to star - schema
- Dimension tables normalized (3. NF)
- Clearer structuring of dimensions
- Database people used to 3. NF
- But: necessity to hide the now more complex structure from the user
- Usually not fully normalized dimension tables

Modelling a Data Warehouse

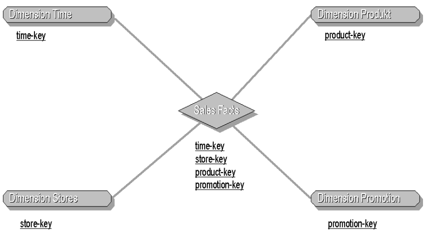
Example: Choosing a Schema for Grocery DWH

- Snowflake schema higher normalized
- Uses less disk space
- Browsing by direct access to tables more complicated because of references spanning multiple tables
- Dimension tables rather small -> little disk space benefit compared to size of DWH
- Star schema simpler to administer

Choosing a star schema for the grocery store DWH

Modelling a Data Warehouse

Example: Star Schema for Grocery Store DWH



Modelling a Data Warehouse

Attributes

- Deciding which fields to add to the various dimension tables as well as to the fact table
- Attribute hierarchies
- Aggregation levels
- Considering possible queries and constraints on the tables
- Effects on OLAP operations like drill-down, roll-up
- Separately for each table

Modelling a Data Warehouse

Example: Fact Table

- Stores data relevant for chosen business process area
- Includes key to the attached dimension tables
- Data taken from OLTP system: POS data
- Product sales per store per day
- Defining the place where the aggregation takes place: POS systems calculate the sales for each product and upload to the central DWH

Modelling a Data Warehouse

Example: Fact Table (2)

Fact table attributes for sales data

keys	***facts***
time_key	*dollar_sales*
product_key	*units_sales*
store_key	*dollar_cost*
promotion_key	*customer_count*
	... (tbd) ...

Modelling a Data Warehouse

Example: Fact Table (3)

- Key of fact table is made up of four foreign keys of dimension tables
- Basic facts obtained from the POS system
- Additional derived attributes for analysis purposes to be defined
- Size considerations (estimations): Gross revenue of grocery chain: $4 billion, average price of product $2 -> about 2 billion items sold (ticket lines)
  3 years history -> 6 billion records -> storing single transactions not easily feasible
  2 billion ticket lines divided by 365 days divided by 500 stores -> ~11.000 items data per day per store to be transferred if aggregation is performed at central DWH
  average store 30.000 different products, about 10% sold per day -> transferring ~3000 records per day per store to central DWH if aggregation is performed at POS

Modelling a Data Warehouse

Example: Dimension Time

- Most general dimension
- Present in almost any DWH
- 'Date' attribute enough if only consecutive order of days relevant
- Separate dimension for evaluations concerning days of week, fiscal periods, seasons, holidays, special events, festivals etc.
- Can be built in advance

Modelling a Data Warehouse

Example: Dimension Time (2)

Time dimension for daily data

time_key	*quarter*
day_of_week	*fiscal_period*
day_number_in_month	*holiday_flag*
day_number_overall	*weekday_flag*
week_number_in_year	*last_day_in_month_flag*
week_number_overall	*season*
month	*event*
month_number_overall	...

Modelling a Data Warehouse

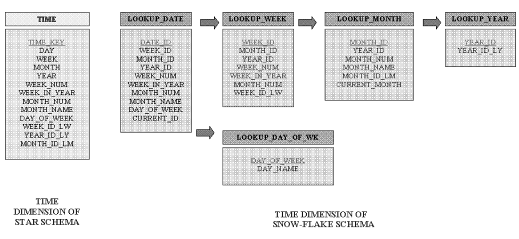
Example: Dimension Time (3)

- Preloaded with data for 5 - 10 years -> ~ 3650 records
- *day_of_week*: allows fast comparison of e.g. Monday to Saturday business
- *day_number_in_month* and *last_day_in_month_flag*: used for daywise comparison, pay-day analysis
- *day_number_overall*: consecutive numbering of days allowing fast arithmetics across month and year boundaries
- Similar flags for weekday - weekend business, holiday - non holiday, quarter comparisons etc.
- event for special events like festivals, strike, catastrophies
- MIND: promotion periods could be handled as time attribute, but since they vary from store to store and from product to product this is not feasible - promotions form separate dimension

Modelling a Data Warehouse

Example: Dimension Time (4)

Star-Schema vs. Snowflake-Schema



Modelling a Data Warehouse

Example: Dimension Product

- Identifies each product by its stock keeping units ID (SKU)
- Based on universal product code (UPC) imprinted as barcode
- Includes special codes for in-store products like fresh meat, groceries, bakery goods etc.
- Stores description of products
- Package size, product groups, brand names, subcategories and categories, department, etc.

Modelling a Data Warehouse

Example: Dimension Product (2)

Product dimension

product_key (SKU)	*weight*
SKU_description	*weight_unit_of_measure*
package_size	*units_per_retail_case*
brand	*units_per_shipping_case*
subcategory	*cases_per_pallet*
category	*shel_width*
department	*shelf_height*
package_type	*shelf_depth*
diet_type	*...*

Modelling a Data Warehouse

Example: Dimension Product (3)

- Managed by headquarters and distributed to stores
- Defines a kind of merchandise hierarchy, e.g. *SKUs* roll up to *package_size -> brand -> subcategory -> category -> department* etc.
- Normalization: only about 30 *departments*, repeated many times in table or *package_type* -> could be normalized to save disk space, but not necessary
- Usually many more attributes stored in product dimension table

Modelling a Data Warehouse

Example: Dimension Store

- Describes each store of the grocery chain
- Geographic dimension
- Created at headquarters by collecting information from stores (contrary to product data which is available at headquarters and distributed to stores)
- Two types of hierarchies in store dimension: geographic hierarchy and sales region hierarchy
- Attributes describing stores with respect to relevant analysis queries like store size, location, available departments etc.

Modelling a Data Warehouse

Example: Dimension Store (2)

Product dimension

store_key	*store_fax*
store_name	*floor_plan_type*
store_number	*photo_processing_type*
store_street_address	*finance_services_type*
store_city	*first_opened_date*
store_county	*last_remodel_date*
store_state	*store_sqft*
store_zip	*grocery_sqft*
sales_district	*frozen_sqft*
sales_region	*meat_sqft*
store_manager	*...*
store_phone	

Modelling a Data Warehouse

Example: Dimension Store (3)

- Geographic hierarchy: *store -> store_zip -> store_county -> store_state*
- Sales hierarchy: *store -> sales_district -> sales_region*
- *floor_plan_type, finance_services_type* are textfields filled with standardized descriptors that can be read and interpreted directly -> can be used for generating OLAP queries by interacting with the table
- *first_opened_date, last_remodel_date* are date-type fields either directly filled with date values or linked to copies of the time dimension

Modelling a Data Warehouse

Example: Dimension Promotion

- Describes each promotion condition under which a product is sold, e.g. temporary price reduction, newspaper ads, coupons, etc.
- So-called '*causal dimension*': factors are thought to change product sales
- Causal conditions highly correlated: price reduction or coupons combined with ads -> one record for each combination of promotions
- Can be used to analyze which products experienced an increased sale during the promotion period
- Cannot be used to analyze which products which did *not* sell because they are not present in the fact table storing only products sold (POS - data)

Modelling a Data Warehouse

Example: Dimension Promotion (2)

Product dimension

promotion_key	*ad_media_name*
promotion_name	*display_provider*
price_reduction_type	*promo_cost*
ad_type	*promo_begin_date*
display_type	*promo_end_date*
coupon_type	...

Modelling a Data Warehouse

Example: Dimension Promotion (3)

Used for evaluation:

- Gain in sales during the promotional period (*lift*) - only possible if a kind of baseline sales can be defined if the product had not been promoted
- Whether products under promotions showed a drop in sales after the promotion period thus cancelling the gain (*time shift*)
- Whether products under promotion showed a gain in sales while other products in the same product category showed a decrease in sales (*cannibalization*)
- Whether the products under promotion (plus additional products of the same brand or product category) showed a net overall gain in sales comparing periods before, during and after the promotion (*growing the market*)

- Whether the promotion was *profitable* taking into account the *lift, time shift, cannibalization* as well as the *costs* for the promotion.

Modelling a Data Warehouse

Example: Fact Table (4)

Current Fact Table Attributes for Sales Data

keys	***facts***
time_key	*dollar_sales*
product_key	*units_sales*
store_key	*dollar_cost*
promotion_key	*customer_count*
	... (tbd) ...

Additional attributes to be defined for further analysis

Modelling a Data Warehouse

Example: Fact Table (5)

Additivity:

- *dollar_sales, units_sales* and *dollar_cost* are additive across all dimensions

- It is possible to calculate aggregates in all dimensions
  e.g. sales or costs per week, per month, per product group, per sales region etc.

- *customer_count* is not additive across all dimensions -> semi-additive attribute

- It is not possible to calculate e.g. customer count per product group

- Information is lost during the aggregation process

Modelling a Data Warehouse

Example: Fact Table (6)

Example for *customer_count*:

- *customer_count* per week per product per store can be calculated
- customer count per week per product per sales region can be calculated
- *customer_count per week per product group per store cannot be calculated:*
  sales for *product* A in *store* 1 has a *customer_count* of 20
  sales for *product* B in *store* 1 has a *customer_count* of 60
  sales for product group containing products A and B for store 1 has a *customer_count* between 20 an 80

If required ways for obtaining correct customer counts for other dimensions must be identified

Modelling a Data Warehouse

Example: Fact Table (7)

Making *customer_count* additive:

- Changing granularity by storing single transaction lines per customer -> customer counts can be calculated by grouping transactions and calculating sums on the fly
- Expensive solution

- Store brand, subcategory, category, department etc. customer counts in explicitly stored aggregates
- Computed while aggregating the data for single customer ticket at POS
- Individual customer ticket aggregates are additive to allow computation of daily item movement records

Modelling a Data Warehouse

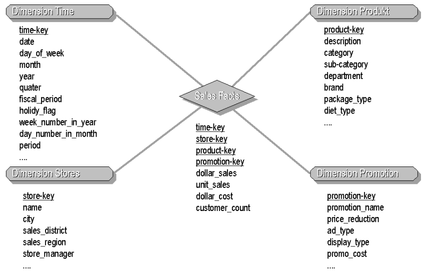
Example: Fact Table (8)

***customer_count* and Additivity in Other Dimensions?**

- Additivity in time dimension?
- Additivity in stores dimension?
- Additivity in promotion dimension?

Modelling a Data Warehouse

Example: Star Schema for Grocery Store (2)



Modelling a Data Warehouse

Aggregate Tables

Most queries address detailed data on one dimension and use summarized data on other dimensions

Examples:

- How much total business did the newly remodelled stores do compared with the chain average?
  Specific stores, summarized over all products
- How did leather goods items costing less than $5 do with my most frequent shoppers?
  Specific products, summarized over all stores
- What was the ration of nonholiday weekend days total revenue to holiday weekend days?
  Specific days, summarized over stores and products

Modelling a Data Warehouse

Aggregate Tables (2)

Goal: Accelerating the most frequent queries

Steps:

- Identify the most frequent queries
- Identify dimensions and aggregates that are most relevant to the respective business areas
- Define aggregate hierarchy
- Create selected pre-calculated aggregate fact tables
- Create corresponding aggregate dimension tables

Modelling a Data Warehouse

Aggregate Tables (3)

The use of prestored summaries (aggregates) is the single most effective too the data warehouse designer has to control performance

(R. Kimball, [KIM96])

Modelling a Data Warehouse

Identify Most Frequent Queries

- Creating a list of possibly most frequent queries
- Performed during design phase (needed for DWH design anyway)
- Based on existing OLTP system and reports
- **But**: monitored and performed during operation of DWH:
  watch what users are doing!
- Behaviour of users changes with given possibilities!
- List of queries

Modelling a Data Warehouse

Example: Identifying Queries

- Sales of bakery goods during holiday periods compared to non-holiday periods
- Sales in the western sales district compared to the eastern sales district
- Sales of low-fat food products in the last 24 months
- Profitability of newspaper ads compared to radio commercials, effects of combinations of both
- ...

Modelling a Data Warehouse

Identify Dimensions

- Select dimensions most frequently involved in list of relevant queries
- Mind: select only the most relevant dimensions
- Consider size of aggregate tables: sparsity failure!

Sparsity: size explosion:

even if only 10% of a stores' different products are being sold per day:

- not only 10% of all brands being sold in that store on that day
- more than 10% of all different products being sold over all stores

Modelling a Data Warehouse

Example: Identifying Dimensions

Dimensions:

- Product ?
- Stores ?
- Time ?
- Promotion ?

Modelling a Data Warehouse

Identify Hierarchies

- For each dimension create hierarchy based on available attributes
- Consider relevant queries
- Consider available data
- Consider additivity of fact table attributes
  (e.g. customer count per product group?)

Modelling a Data Warehouse

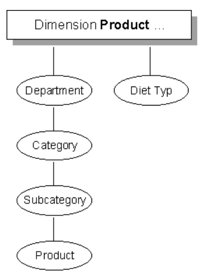
Example: Hierarchy for Product

Attributes - Dimension Hierarchy :

product_key (SKU) ?	*weight ?*
SKU_description ?	*weight_unit_of_measure ?*
package_size ?	*units_per_retail_case ?*
brand ?	*units_per_shipping_case ?*
subcategory ?	*cases_per_pallet ?*
category ?	*shel_width ?*
department ?	*shelf_height ?*
package_type ?	*shelf_depth ?*
diet_type ?	*...*

Modelling a Data Warehouse

Example: Hierarchy for Product (2)



Modelling a Data Warehouse

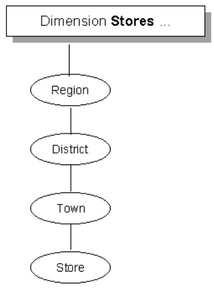
Example: Aggregates for Stores

Attributes - Dimension Hierarchy :

store_key ?
store_name ?
store_number ?
store_street_address ?
store_city ?
store_county ?
store_state ?
store_zip ?
sales_district ?
sales_region ?
store_manager ?
store_phone ?
store_fax ?
floor_plan_type ?
photo_processing_type ?
finance_services_type ?
first_opened_date ?
last_remodel_date ?
store_sqft ?
grocery_sqft ?
frozen_sqft ?
meat_sqft ?
...

Modelling a Data Warehouse

Example: Hierarchy for Stores (2)



Modelling a Data Warehouse

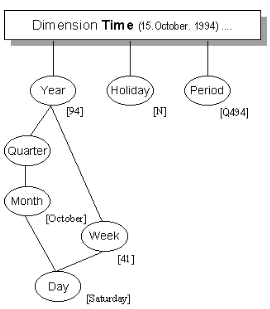
Example: Hierarchy for Time

Attributes - Dimension Hierarchy :

time_key ?
day_of_week ?
day_number_in_month ?
day_number_overall ?
week_number_in_year ?
week_number_overall ?
month ?
month_number_overall ?
quarter ?
fiscal_period ?
holiday_flag ?
weekday_flag ?
last_day_in_month_flag ?
season ?
event ?
...

Modelling a Data Warehouse

Example: Hierarchy for Time (2)



Modelling a Data Warehouse

Aggregated Fact Tables

- Identify required / desired fact tables
- Higher-order aggregates can be calculated using lower-order aggregates e.g. sales per department can be based on sales per category
- Estimate their number and size
- Check availability of data
- Check needed dimension aggregate tables

Modelling a Data Warehouse

Example: Identifying Aggregated Fact Tables

Desired aggregate fact tables:

- One-way aggregate: category totals by store by day
- One-way aggregate: district totals by store by day
- One-way aggregate: monthly totals by product by store
- Two-way aggregate: category totals by district totals by day
- Two-way aggregate: category totals by month totals by store
- Two-way aggregate: district totals by month totals by product
- Three-way aggregate: category totals by district totals by month totals

Modelling a Data Warehouse

Example: Identifying Aggregated Fact Tables

- 7 aggregated fact tables
- Aggregated fact tables are derivatives of basic fact table
- Checking additivity of fact table attributes:
  dollar_sales ?
  units_sales ?
  dollar_cost ?
  customer_count ?
- Aggregates of *customer_count* must be created at POS (point of sales)

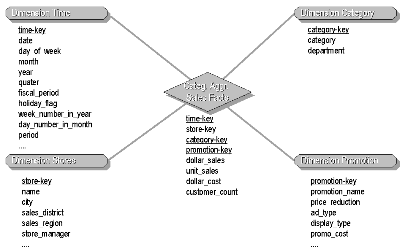
Modelling a Data Warehouse

Linking Fact with Dimension Tables

- Keys for linking fact to dimension tables in starschema
- Keys for linking aggregated fact tables to aggregated dimension tables
- Defining attributes in aggregated dimension tables

Modelling a Data Warehouse

Example: Linking Category Aggreg. Sales Facts



Modelling a Data Warehouse

Aggregated Fact Tables (2)

Additional fact tables can be added as necessity arises

Dimension tables tend to get very small as we move up the hierarchy
e.g. 1 entry in dimension table for all_stores, 1 for all_products, 1-10 for all years, ...

Number and size of aggregate fact tables explodes !

Check which aggregate fact tables are needed and which can be calculated

Modelling a Data Warehouse

Example: Possible Aggregated Fact Tables

e.g. for dimensions

- **Product**: per category, all merchandise total
- **Store**: per district, per region, all stores total
- **Time**: month, year

We end up with 7 aggregate dimension tables (small) and 35 aggregate fact tables

Modelling a Data Warehouse

Example: Possible Aggregated Fact Tables

One way aggregates:

category by store by day
district by product by day
region by product by day
all stores by product by day
month by product by store
year by product by store
all merchandise by store by day

Modelling a Data Warehouse

Example: Possible Aggregated Fact Tables (2)

Two way aggregates:

category by district by day
category by region by day
category by all stores by day
category by month by store
category by year by store
district by month by product
region by month by product
all stores by month by product
all merchandise by district by day
all merchandise by region by day
all merchandise by all stores by day
all merchandise by month by store
all merchandise by year by store
district by year by product
region by year by product
all stores by year by product

Modelling a Data Warehouse

Example: Possible Aggregated Fact Tables (3)

Three way aggregates:

category by district by month
category by region by month
category by all stores by month
category by district by year
category by region by year
category by all stores by year
all merchandise by district by month
all merchandise by region by month
all merchandise by all stores by month
all merchandise by district by year
all merchandise by region by year
all merchandise by all stores by year

Modelling a Data Warehouse

Sparsity Failure

- 10.000 products with 2.000 aggregates (e.g. brands)
- 1.000 stores with 100 aggregates (e.g. districts)
- 100 time periods with 30 aggregates (e.g. financial period based on weeks)
- 10% of all products sold per day

Additional size

- Product dimension: 20%
- Store dimension: 10%
- Time dimension: 30%

Modelling a Data Warehouse

Sparsity Failure (2)

Considering the 7 aggregates identified before:

- Category totals by store by day
- District totals by store by day
- Monthly totals by product by store
- Category totals by district totals by day
- Category totals by month totals by store
- District totals by month totals by product
- Category totals by district totals by month totals

By how much will the size of the database increase as opposed to the size of the base fact table ?

Modelling a Data Warehouse

Sparsity Failure (3)

A suggestion / estimation:

Table	Sparsity	Factor
Category totals by store by day	0.2	0.2
District totals by store by day	0.1	0.1
Monthly totals by product by store	0.3	0.3
Category totals by district totals by day	0.2*0.1	0.02
Category totals by month totals by store	0.2*0.3	0.06
District totals by month totals by product	0.1*0.3	0.03
Category totals by district totals by month totals	0.2*0.1*0.3	0.006
TOTAL		**0.716**

Baseline: 10.000 products * 1.000 stores * 100 time periods * 10% -> 100 mil.
Plus aggregated fact tables: ~72% of baseline = 0.72 * 100 mil. -> 172 mil.

Modelling a Data Warehouse

Sparsity Failure (4)

e.g. brands total by store by day:

- Assumption: only 10% of all products sold in given store on a given day -> 10% of 10.000 products = 1.000 products being sold -> 1.000 products in daily store data
- If every product sold belongs to a different brand -> 1.000 different brands = 50%
- There may be 50% of all brands in the daily store data as opposed to 10% of the individual products -> 1.000 brands in daily store data
- Aggregated fact table may have same size as basic fact table

Modelling a Data Warehouse

Sparsity Failure (5)

Table	**Product**	**Store**	**Time**	**Sparsity**	**# Records**
Base: Products by store by week	10.000	1.000	100	10%	100 mil.
Brand by store by week	2.000	1.000	100	50%	100 mil.
Product by district by week	10.000	100	100	50%	50 mil.
Product by store by period	10.000	1.000	30	50%	150 mil.
Brand by district by week	2.000	100	80	80%	16 mil.
brand by store by period	2.000	1.000	30	80%	48 mil.
Product by district by period	10.000	100	30	80%	24 mil.
Brand by district by period	2.000	100	30	100%	6 mil.
TOTAL					**494 mill.**

Database may grow up 394% !

Modelling a Data Warehouse

Sparsity Failure (6)

- Take care when designing aggregate tables
- Make sure that each aggregate summarizes at least about 10 - 20 records on the average

Example:

- Product dimension summarized only 5 lower level products on the average
- Time dimension summarized only about 3 periods on the average
- One way aggregates of time and product contributed 250 mil. records
- If each had summarized about 20 lower level items on average -> only about 70 mil. new records even with sparsity growing to 70%

Modelling a Data Warehouse

Querying Aggregate Tables

Queries transformed into SQL statements

e.g.: Show total sales by category in Cincinnati stores on New Year's Day 1998 for base fact table::

select category_description, sum(sales_dolars)
from base_sales_fact, product, store, time
where base_sales_fact.product_key = product.product_key
and base_sales_fact.store_key=store.store_key
and base_sales_fact.time_key = time.time_key
and store.city = 'Cincinnati'
and time.day = 'January 1, 1996'
group by category_description

Modelling a Data Warehouse

Querying Aggregate Tables (2)

Same query if category totals aggregate table exists:

select category_description, sum(sales_dolars)
from ***category_sales_fact***, ***category_product***, *store, time*
where ***category_sales_fact***.*product_key =* ***category_product***.*product_key*
and ***category_sales_fact***.*store_key=store.store_key*
and ***category_sales_fact***.*time_key = time.time_key*
and store.city = 'Cincinnati'
and time.day = 'January 1, 1996'
group by category_description

Category_sales_fact and corresponding category_product dimension table replace base sales fact and basic product dimension table

Modelling a Data Warehouse

Querying Aggregate Tables (3)

- Navigator
- Reads users query and transforms it into best available aggregate query
- Metadata descriptions provide information about existing aggregate tables
- Existence of aggregate tables is transparent to the user
- Can be used to build statistics on user queries, aggregate table usage and the need for additional aggregates
- Allows incremental rollout of new aggregation tables

Modelling a Data Warehouse

Querying Aggregate Tables (4)

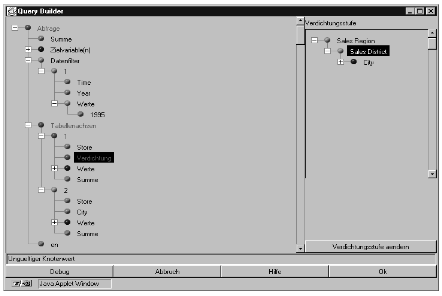
- Navigator: Replacing base-level fact and dimension tables with aggregated fact and dimension tables

1.) Rank order all aggregate tables

2.) Starting from the smallest, verify whether all of the dimensional attributes in the query can be found.
If so, replace base tables in query with corresponding aggregate tables.
If not continue with the next bigger aggregate table until finally reaching the base fact table

Modelling a Data Warehouse

Example: Navigator



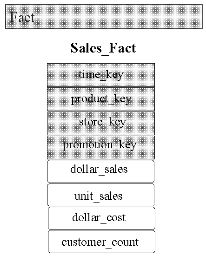
Modelling a Data Warehouse

Metadata

- Describes the data in the DWH
- Technical metadata
- Business metadata
- Operational metadata

Modelling a Data Warehouse

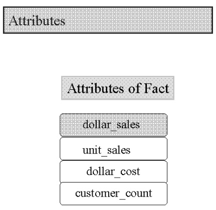
Example: Metadata Fact Table




```
Destination = ( ( Database =
( Identifier      = “Grocery_id”
  Description   = “Grocery Database”
  ServerName  = “String”
  Name            = “String”
  Type  = “MULTIDIMENSIONAL”
  Facts = ( Identifier = “Sales_Fact_id”
    Description = “Sales Fact”
    SqlName = “Sales_Fact”
    Field = ( ( Identifier = “time_id”
     Description = “Time Dimension”
     SqlName = “time_key”
     Key = “1” Type = “NUMERIC”
     NullValue = “T  | F”
      ) /* end of first field */
    ) /* end of Field */
```


Modelling a Data Warehouse

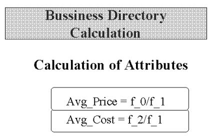
Example: Metadata - Attributes




```
Attributes = (
( Identifier    = “dollar_sales_id”
  Description = “Dollar Sales”
  SqlName     = “dollar_sales”
  Type           = “NUMERIC”
  NullValue  = “T | F”
  Format        = “#,##0”
) /* end of first attribute
) ) /* end of Attributes & Facts */
```


Modelling a Data Warehouse

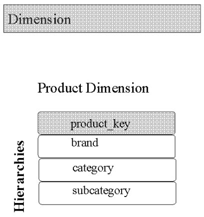
Example: Metadata - Derived Attributes




```
Calculation = (
( Identifier    = “Avg_Price_id”
  Description = “Average Price”
  SqlName     = “Avg_Price”
  Type           = “NUMERIC”
  NullValue   = “T | F”
  Rules       = “ / f_0 f_1”
  Format        = “#,##0”
  CalcType     = “DERIVED”
) /* end of Calculation */
```

Modelling a Data Warehouse

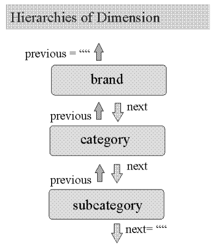
Example: Metadata - Dimensions




```
Dimension = (
( Identifier    = "product_id"
  Description = "Product Dimension"
  SqlName    = "Product"
  Fields = (
  ( Identifier    = "product_key_id"
    Description = "Product"
    SqlName     = "product_key"
    Key            = "1"
    Type          = "NUMERIC"
    MissValues = "M | N"
  ) ) /* end of first field & Fields */
```


Modelling a Data Warehouse

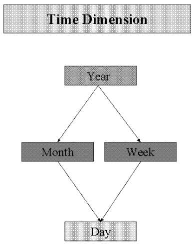
Example: Metadata - Dimension Hierarchies




```
Hierarchies = (
(  Identifier = "brand_id"
   Description = "Brand"
   SqlName = "brand"
   Type = "VARCHAR"
   Previous = ""
   Next = "category"
) ) /* end of first hierarchy &
    Hierarchies */
) )  /* end of first dimension &
     Dimension */
```


Modelling a Data Warehouse

Example: Metadata - Time Dimension




```
Dimension = ( Id = "TimeId"
    Desc = "Time dimension"
    Sql   = "Time"
    Aggr = (
    ( Sql = "Year" Prev = ""
      Next = ( Month Week ) )
    ( Sql = "Month" Prev = "Year"
      Next = "Day"
      Security =
      ( User =
        ( Id = "user1" value = ("1 3 ") ) )
    ( Sql = "Week" Prev = "Year"
      Next = "Day")
    ( Sql = "Day" Prev = ( Month Week )
      Next = "") )
```


Modelling a Data Warehouse

Example: Metadata - EIS Page

```
// Example EIS page described in MetaScript
eispage = (
title = "Cross Dollar sales over all marketing regions"
.....
// Definition of default OLAP DWH cube request
mql = (
    id = query1
    presentation = report
    ....
    measure = ( fact = "Sales_Fact" attr="dollar_sales")
    // Definition of title dimensions
    selection_criteria = (
            (dim = "Time"  agg = "year" val = ("1995")
    ) // end_of_selctive clause
    display = (
            (dim = "Store" agg="sales_region" val =())
            (dim = "Product" agg="subcategory" val=())
            (dim = "Promotion" agg="promotion_name"
val=())
    ) // end of display clause
) // end_of_default_query
// Page template
frame0 = (
            type = frameset
            subframes = (
            (frame1 horiz 5%)
            (frame2 horiz *)
            (frame3 horiz 15%)
            ) // end_of_subframes
) // end_of_frame description

frame1 = (
    type = frame
    desc = "Toolbar"
    buttons = (
      (type = navigator img = "navigator.gif" params = (mode =
       wizard))
      (type = forwd img = "forwd.gif" dim = time
      (type = show img = "barchart.gif" target = chart1)
) // end_of_buttons
) // end_of_frame1
// Visualisierungsobjekte
frame2 = (
     type = frame
     ....
      // BarChart
    chart1 = (
            type = barchart-grouped
            visible = on
            query = query1
            ....
            xaxis = (dim = "Time" label = short ....)
            yaxis = (dim = "Product" label = short ....)
            .......
     ) // end_of_bar chart 1
) // end_of_display area
) // end_of_eis_page
```

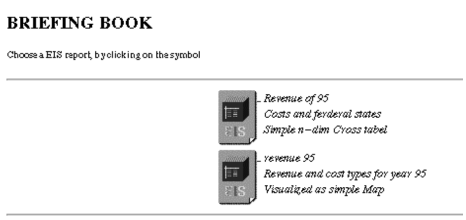

Modelling a Data Warehouse

Executive Information Systems

- Briefing books
- Tables
- Charts

Modelling a Data Warehouse

Example: Briefing Book



Modelling a Data Warehouse

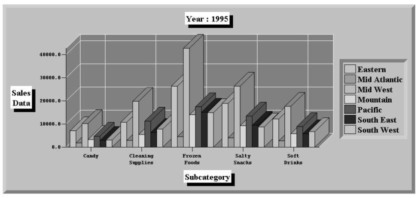
Example: Table

	Dimension Product *Department*					
Dimension Store *Sales Region*	Grocery		Household		Sum	
	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales	Dollar Sales	Unit Sales
Eastern	131,313.03	95540	23,724.09	15660	155,037.13	111200
Mid Atlantic	29,748.22	22819	7,786.74	4777	37,534.96	27596
Mid West	194,411.88	140037	38,723.01	25231	233,134.89	165268
Mountain	62,636.52	43853	11,382.74	7422	74,019.26	51275
Pacific	95,898.20	69775	21,062.78	12852	116,960.98	82627
South East	68,829.44	46728	12,970.05	8363	81,799.49	55091
South West	69,072.26	48749	13,844.63	8914	82,916.89	57663
Sum	651,909.56	467501	129,494.04	83219	781,403.56	550720

Modelling a Data Warehouse

Example: Chart

- Business measures **Sales Data** , additive attribute *Dollar Sales* given in [1 Us-Dollar]
- selection: for Dimension **Time** aggregated *Year* in ('1995')
- arrange: for Dimension **Store** aggregated *Sales Region* Sum out ('Eastern' , 'Mid Atlantic' , 'Mid West' , 'Mountain' , 'Pacific' , 'South East' , 'South West') and Dimension **Product** aggregated |R| *Subcategory* Sum out ('Candy' , 'Cleaning Supplies' , 'Frozen Foods' , 'Salty Snacks' , 'Soft Drinks')
- "GROCERY" Data Warehouse given in 1 Us-Dollar. Timestamp of last update: Friday, April 17, 1997 10:54:00



Modelling a Data Warehouse

Pitfalls

- Not knowing what you really want
- Thinking that DWH-design is the same as transactional DB design - a DWH is not simply a big database!
- Loading the warehouse with data simply because it is available
- Underestimating the complexity of a DWH project
- Getting caught by technological gadgets
- Focusing on internal data and ignoring the use of external data
- Using data with overlapping or confusing definitions / semantics
- Believing performance and scalability promises
- Believing that once a DWH is running the work is done