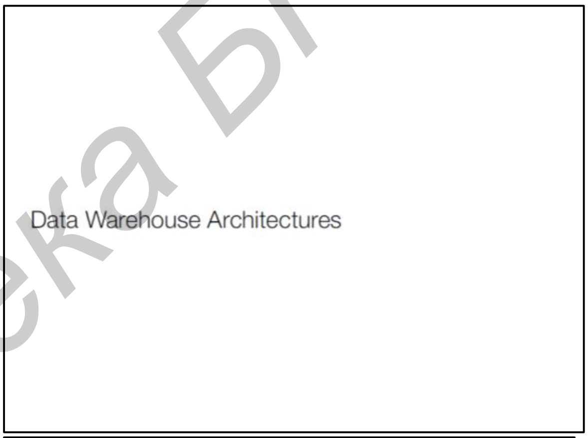
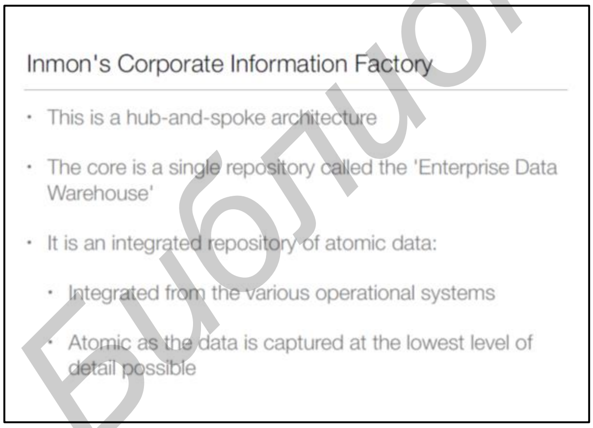
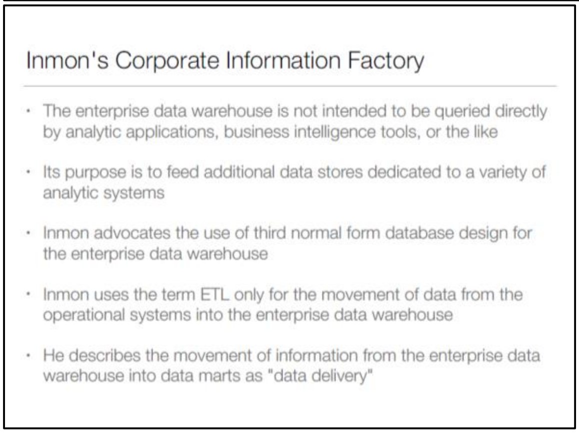
DATA WAREHOUSE ARCHITECTURE AND DESIGN (WORKSHOP LESSEN 2)

M. G. STROO, PhD
Owner of Invisi, Netherlands, Owner of Act On Insight, Belarus, Information Innovation Leader, Business Intelligence Consultant: Royal Agio Cigars, City of Rotterdam, Nuon

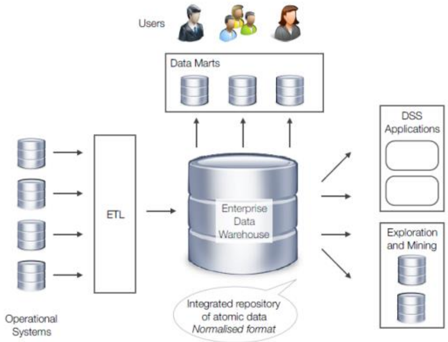
Invisi BV, Netherlands







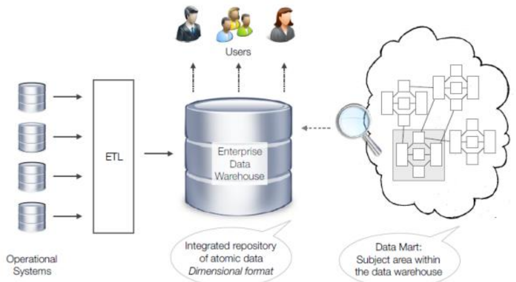
Inmon's Corporate Information Factory



Kimball's Dimensional Data Warehouse

- Kimball is largely responsible for popularising star schema design in the 1990's
- Kimball developed an enterprise architecture for the data warehouse, built on the concept of dimensional design
- Sometimes referred to as the "bus architecture"
- Shares many characteristics of Inmon's Corporate Information Factory

Kimball's Dimensional Data Warehouse



Kimball and Inmon Similarities

- Separation of the operational and analytic systems
- ETL process to consolidate, integrate and load the data into a single repository
- Data goes into an integrated repository of atomic data

Kimball and Inmon differences

- The dimensional data warehouse is designed according to the principles of dimensional modelling. It consists of a series of star schemas or cubes, which capture information at the lowest level of information possible
- The enterprise data warehouse is designed using the principles of ER (entity-relationship) modelling.
- The dimensional data warehouse may be accessed directly by analytic systems. The data mart becomes a logical distinction; it is a subject area within the data warehouse

Kimball and Inmon Variations

- An intermediate step with a set of tables in third normal form to make the ETL easier is acceptable for Kimball
- These are usually staging tables and should be accessed directly only by the ETL process
- This makes the Kimball solution more like the Inmon solution, with a normalised repository of data not accessed by applications
- Another variant is where the dimensional data warehouse is not accessed directly by analytic applications
- New data marts are constructed by extracting data from the dimensional data warehouse
- This increases the resemblance to the Corporate Information Factory, where data marts are separate entities from the integrated repository of atomic data

Stand-Alone Data Marts

- Can achieve rapid and inexpensive results in the short term
- The stand-alone data mart is an analytic data store that has not been designed in an enterprise context
- It is focused exclusively on a subject area
- One or more operational systems feed a database called a data mart
- Analytical tools or applications query it directly, bringing information to end users
- Data marts may be offered as part of packaged (operational) applications
- Sometimes they are built within user organisations, outside of the IT department

Architecture and Dimensional Design

Architecture	Advocate	Also known as	Description	Role of Dimensional Design
Corporate Information Factory	Bill Inmon	• Atomic data warehouse • Enterprise data warehouse	• *Enterprise data warehouse* component is an integrated repository of atomic data • It is *not* accessed directly • Data marts reorganise data for departmental use / analysis	Dimensional design used for data marts only
Dimensional Data Warehouse	Ralph Kimball	• Enterprise data warehouse • Bus architecture • Architected data marts • Virtual data marts	• *Dimensional data warehouse* is an integrated repository of atomic data • It may be accessed directly • Subject areas within the dimensional data warehouse • Data marts not required to be separate databases	All data is organised dimensionally
Stand-Alone Data Mart	No takers, yet common	• Data mart • Silo • Stovepipe • Island	• Subject area implementation without an enterprise context	May employ dimensional design

Operational Data Store

- Contains current or near current integrated data
- Subject oriented
- Limited amount of historical data
- Volatile
- Speed of data updates varies from seconds to a day
- Quick updating limits transformation possibilities
- Comes in different types with different levels of integration and quality

Data Warehouse Architecture Exercise

- You are asked by your company to propose a data warehouse architecture:
  - The director for a company wide solution
  - The manager of a department to give him specific information
  - The Operations Manager to help him to manage his operation
- Propose an architecture and explain your choice

Dimensional Modelling

Purpose of Analytic Databases

- Operational systems support the execution of business processes
- Analytic systems support the evaluation of processes
- Both systems have contrasting usage profiles
- Different principles guide their design
- Interaction with an analytic system takes place exclusively through queries that retrieve data
- These queries can involve large numbers of transactions
- It supports the maintenance of historic data

The Star Schema

- Dimensional design for a relational database
- Contains dimension and fact tables
- Dimension tables contain context for facts
- Dimensions are used to specify how facts will be rolled up
- Dimension values may be used to filter reports
- Dimension tables are not in third normal form

The Star Schema

- Each dimension table is given a surrogate key, typically an integer
- The dimension table key column name usually have the same suffix, like _key
- The dimension tables also contain columns that uniquely identify something in an operational system, like customer_id, salesperson_id, product_code. These are called natural keys
- By having separate surrogate keys and natural keys, you can track changes for dimension values
- Fact tables contain the facts and surrogate keys to the related dimension tables
- Often a fact row can be uniquely identified by these foreign keys, but not always
- The level of detail of the fact table is called the grain
- The information in the fact tables is typically consumed in different levels of details, using aggregation

Main Guiding Design Principles

- These two design principles are at the core of dimensional modelling:
  - accuracy
  - performance
- Accuracy: is it possible that facts can be aggregated in a way that does not make sense? Is there a design alternative that can prevent this?
- Performance: dimensional designs are very good to providing a rapid response to a wide range of unanticipated questions

Dimension Table Features - Keys

- Each dimension table is assigned a surrogate key. It is created especially for the data warehouse or data mart
- Surrogate keys are usually integers, generated and managed as part of the ETL process that loads the star schema
- One or more natural keys will also be present in most dimension tables
- The natural keys are identifiers carried over from source systems
- They identify a corresponding entity in the source system
- The values in natural keys may have meaning to users of the data warehouse
- Even without significant meaning, the presence is needed for the ETL that load fact tables

Dimension Table Features - Rich set of dimensions

- Dimensions can be added to queries in different combinations to answer a wide variety of questions
- The larger the set of dimension attributes, the more ways that facts can be analysed
- Dimension tables with a large number of attributes can be thought of as wide
- Commonly used combinations of attributes may be stored
- Codes may be supplemented with corresponding description values
- Flags are translated from boolean values into descriptive text
- Multi-part field are both preserved and broken down into constituent pieces
- Consider numeric attributes that can serve as dimensions

Dimension Table Features - Common Combinations

- In operational systems, it is common practice data elements down to constituent parts whenever possible
- In the dimensional design, common combinations of these elements are stored as well. Uses:
  - Increases query performance
  - Sort reports
  - Order data
- Example:
  - First name, middle initial, last name
  - Store also full name and Last-name-first format
  - Database administrators can index these columns for efficient query performance

Dimension Table Features - Codes and Flags

- In operational systems it is common to describe values in a domain using codes
- Both the codes and description may be useful dimensions
- Store both in your dimension table so that users can filter, access and organise in whatever way they see fit
- Flags can be stored in source systems in different ways; boolean data type, integer with value 0 or 1, character with "Y" or "N" or two values indicating "True" or "False"
- In a dimensional design, store the descriptive value of the flag options. These are far more useful than 0/1 or Y/N and much clearer when defining a query filter

Grouping Dimensions

- Dimension attributes are grouped into tables that represent major categories of reference.
- Junk dimensions collect miscellaneous attributes that do not share a natural affinity.
- When principles of normalisation are applied to a dimension table, the result is called a snowflake
- Snowflakes may be useful in the presence of specific software tools. Dimensional design fully embraces redundant storage of information (= no snowflakes)

Dimension Table Example

DIM_product
product_key
product_code
product_name
product_group
brand
size
colour
cost_price

Dimension Table Features - Benefits of Redundancy

- The storage of redundant data element specific in dimensional modelling have three advantages in an analytic environment:
  - performance
  - usability
  - consistency
- Precomputing and storing extra columns reduces the burden on the DBMS are query time, optimise performance with indexes and other techniques
- The redundant information makes it also easier for users to interact with the analytic database
- Explicit storage of all dimensions guarantees they are consistent, regardless of the application being used.

Degenerate dimensions

- Sometimes some dimensions associated with a business don't fit into a neat set of tables
- It may be appropriate to store one or more dimensions in the fact table. It is then called a degenerate dimension
- Although stored in the fact table, the column is still considered a dimension
- Consider if the attribute is really a degenerate dimension. Often such dimensions are better placed in junk dimensions.
- Transaction identifiers are commonly used as degenerate dimensions

Degenerate Dimension Example

FCT_order_line
order_date_key
customer_key
product_key
order_number
order_line
quantity_ordered
unit_price
discount_given

Slowly Changing Dimensions

- Information in a dimension table may change in the operational source over time, through correction of errors or updates.
- Because the dimension tables have surrogate keys as the primary key , it can handle changes different from the source systems
- How changes in source data are represented in dimension tables is referred to as slowly changing dimensions

Slowly Changing Dimensions - Type 1

- When the source of a dimension value changes, and it is not necessary to preserve its history in the star schema, type 1 is used
- The dimension (attribute) is simply overwritten with the new value
- The star carries no hint that the column ever contained a different value
- Any associated facts from before the change have their historic context altered
- Type 1 typically used for dimensions where a change is usually because of an error that is corrected (like birth date for a person)

Slowly Changing Dimensions - Type 2

- Type 2 preserves the history of facts:
  - Facts that describe events before the change are associated with the old value
  - Facts that describe events after the change are associated with the new value
- With type 2, a new row is inserted in the dimension table when there is a change in the source data
- This creates the effect of "versions" of a single dimension value in the dimension table
- These versions have the same natural key, but a different surrogate key value
- You can add a "current" flag to indicate the current row of a given natural key value
- To know when a version of a dimension row was valid, a date stamp is added

Choosing and Implementing Response Types

- A single dimension may have a type 1 response to some changes and type 2 response to other changes
- Most of the time a type 2 response is most appropriate
- There are situations in which the change of a source element may result in either type of response. When the source system records the reason for a change, you may choose to treat a change as type 1 in the case of an "error correction" or type 2 otherwise
- When a dimension contains multiple response types, ETL developers must factor in a variety of possible situations

Grouping Dimensions into Tables

- A dimensional model does not expose every relationship between attributes as a join
- Contextual relationships tend to pass through fact tables
- Natural affinities are represented by putting attributes in the same dimension table
- Dimensions are entities that can be related in multiple contexts (in different stars)
- Dimensions are grouped into tables based on natural affinity

Breaking Up Large Dimensions

- It is not uncommon for large dimensions to contain well over 100 attributes
- A dimension table may become so wide that it may have an effect on the database, like allocation of space or block size
- Large dimensions can be a concern for ETL developers. With many type 2 attributes, updates can become a tremendous bottleneck
- You may solve this by splitting dimensions arbitrarily
- An overwhelmingly large dimension may also be a sign that there are two distinct dimensions. Put these in two tables
- You can relocate free-form text fields to an outrigger

Dimension Roles and Aliasing

- Measurement of a business process can involve more than one instance of a dimension
- These roles are represented in a fact table by multiple foreign key references to the same dimension table
- This is very common to happen with the date dimension

Avoiding the NULL

- NULL can fail in WHERE clauses that lack a condition specifically for the NULL
- Never allow the storage of NULL in dimension columns. Instead, choose a value that will be used when data is not available (e.g. "Unknown")
- When a fact can't be associated with a row in a dimension table, we will use a special row in the dimension table
- You may have special rows for different situations, like invalid data or late-arriving data

Fact Table Features

- The fact table is the engine for business process measurement
- Where dimension tables are wide, fact tables are deep. They contain many more rows than dimension tables
- They contain foreign keys to the dimension tables, usually integers
- The facts themselves are usually integers or floating point decimal numbers
- The fact table should contain every fact relevant to the process it describes, even if some of the facts can be derived from others
- Some facts are nonadditive, like percentages or account balances

Fact Tables and Business Processes

- Dimensional models describe how people measure their world
- To be studied individually, each process should have its own fact table
- To determine if facts belong to one process, ask:
  - Do these facts occur simultaneously?
  - Are these facts at the same level of detail (or grain)?
- Multiple-process fact tables can be useful when *comparing* processes

Facts That Have Different Timing

- Events may share the same dimensions and seem related, but take place at different times. Then they are different processes and should have separate fact tables
- When a fact table for example can contain shipments and/or orders, the "and/or" in the statement of grain is usually a sign of problems to come
- Querying on such a table may get unexpected result rows that will confuse users
- Working around poor schema design may end up in an example of boiling the frog

Facts That Have Different Timing - Example

day_key	customer_key	product_key	quantity_ordered	quantity_shipped
123	777	111	100	0
123	777	222	200	0
123	777	333	50	0
456	777	111	0	100
456	777	222	0	75
789	777	222	0	125

These zeros will cause trouble

Facts That Have Different Timing - Example

Shipment Report - January 2008 - Customer 777

Product	Quantity shipped
Product 111	100
Product 222	200
Product 333	0

Page 1 of 1

A zero appears because there was an order

Facts That Have Different Grain

- When two or more facts describe events with different grain, they describe different processes
- Different grain can be caused by a different number of related dimensions or different level of hierarchy in a dimension (e.g. months versus days)

Fact Table Types

- The transaction fact table tracks individual activities or events that define a business process
- The snapshot fact table periodically samples status measurements such as balances or levels
- The accumulating snapshot table is used to track the progress of an individual item through a series of steps

Transaction Fact Tables

- Examples:
  - Booking of an order
  - Shipment of a product
  - Payment on a policy
- Each individual row describes the occurrence of an event
- By storing facts and associated dimensional detail, they allow activities to be studied individually and in aggregate

Transaction Fact Table Grain

- May be defined by referencing an actual transaction identifier, such as an order line
- May be specified in purely dimensional terms, as in "orders by day, customer, product and salesperson"
- Sometimes the grain is already a summary instead of an individual transaction, for instance because detail is available elsewhere or because the transaction volume is too large
- Despite a clearly defined grain, also an optional relationship is possible. Then the dimension contains a special row to represent this missing relation, like "not applicable"

Transaction Fact Tables Are Sparse

- Rows are only recorded for activities that take place, not for every combination of dimension values
- For instance rows are only created for those days when there are orders, only those products that are ordered and customers that place the orders

Transaction Fact Tables Contain Additive Facts

- Most nonadditive measurements, like ratios, can and should be broken down into fully additive components
- This allows the granular data in the fact table to be aggregated to any desired level of detail
- If you can use the sum of each measurement in the fact table in an aggregation, the fact is additive
- Storing fully additive facts provide the most flexible analytic solution

Transaction Fact Table Example

FCT_order_line
order_date_key
customer_key
product_key
order_number
order_line
quantity_ordered
unit_price
discount_given

Snapshot Fact Tables

- Are used to describe the effect of a series of transactions. These effects are called status measurements
- Some status measurements cannot be described as the effect of a series of transactions, for example the water level in a reservoir, the oxygen level in the air
- The snapshot fact table samples the measurement in question at a predetermined interval
- A snapshot fact eliminates the need to aggregate a long chain of transaction history

Snapshot Fact Example

- To know the balance of a bank account it is possible to calculate this from the full transaction history
- Over time this may involve thousands of transactions per bank account
- The account balance may be used to compute interest fees for example

When Transaction Data Is Not Stored

- It is possible that transactions reach further back into the past than is recorded in the data warehouse. For example a bank account that has been active for 50 years
- The volume of transaction detail may be too large to store in the data warehouse. For example the quality of train tracks every 20 cm
- A measurement may be status-oriented. For example budgets, temperature readings, reservoir levels

Don't Store the Balance with Each Transaction

- The transaction fact table is sparse. When there is no activity on a certain day, the balance will not be recorded when stored with transactions
- When there is more than one transaction, there will be double-counting in queries

The Snapshot Model

- Snapshots are dense
- A snapshot model contains at least one fact that is semi-addiditive
- The grain of a snapshot must include the periodicity at which status will be sampled and a definition of what is being sampled
- The grain of a snapshot fact table is usually declared in dimensional terms (definition of what is being sampled)

Semi-Additivity

- A semi-additive fact cannot be summed meaningfully across the time (date) dimension
- The fact can be additive across other dimensions
- The semi-additive fact can be summarised across periods in other ways, like minimum, maximum and average
- Some status measurements are not additive at all. For example water level or ambient temperature

Snapshot Fact Table Example

FCT_bank_balance
period_key
bank_account_key
branch_key
account_balance

Pairing Transaction and Snapshot Designs

- Many processes can be modelled both in a transaction and a snapshot fact
- When a design will include both a transaction fact table and a periodic snapshot, the snapshot can and should be designed to use the transaction fact table as a source
- This eliminates duplicative ETL processing of the source data
- It ensures that dimensional data will be identified and loaded consistently

Accumulating Fact Tables

- Focuses on time between events in a process
- The grain is a unit that goes through the business process, like a loan application
- The fact table will have exactly one row for each unit
- It will have multiple keys to the Date dimension for completion of each stage of the process
- Each row has a group of facts that measure the number of days spent on each stage

Accumulating Fact Tables

- The active rows are updated regularly
- Fact for the duration of the active step is incremented at each load
- Each time a stage is completed, the appropriate end date key is set
- When the design for a business process includes both a transactional star and an accumulating snapshot, the accumulating snapshot should use the transaction star as its source

Dimensional Modelling Exercise

- You are approached by one department of your company to create a data mart for one of their processes:
  - Accounting - bookkeeping
  - Sales - product sales
  - Human Resources - employees
  - Specific to company:
    - Production
    - Client product development
    - Customer activity (telecom)

Querying Dimensional Models

Using a Star Schema

- Most queries against a star schema follow a consistent pattern:
  - One or more facts are requested, along with the dimensional attributes that provide the desired context
  - The facts will be summarised in accordance with the dimensions present in the query
  - Dimension values are used to limit the scope of the query (filter)
- The star schema can be used in this way with any combination of facts and dimensions (in the star)
- Note that the ability to report facts is primarily limited by the level of detail at which they are stored.
- Various aggregations are sum, average, count

Typical Star Schema Query Example

```
SELECT store_location, month_name, SUM(sales_price) AS
total_sales, SUM(discount) AS total_discount
FROM fact_sales fs
JOIN dim_date dd
ON dd.date_key = fs.date_key
JOIN dim_sales_people dp
ON dp.sales_people_key = fs.sales_people_key
WHERE year = 2015
AND country = 'Belarus'
GROUP BY store_location, month_name
ORDER BY month_number
```

see: aggregation, relate fact table to dimension tables, filters, order

Typical Star Schema Query Example Alternative

```
SELECT store_location, month_name, SUM(sales_price) AS
total_sales, SUM(discount) AS total_discount
FROM fact_sales fs, dim_date dd, dim_sales_people dp
WHERE dd.date_key = fs.date_key
AND dp.sales_people_key = fs.sales_people_key
AND country = 'Belarus'
AND year = 2015
GROUP BY store_location, month_name
ORDER BY month_number
```

Analysing Facts From More Than One Fact Table

- When comparing facts from different fact tables, it is important to collect them from separate SELECT clauses
- When you use a single SELECT, there is risk of double counting, or worse
- The two-step process used is called *drilling across*, stepping from one star to another

Drill-across Procedure

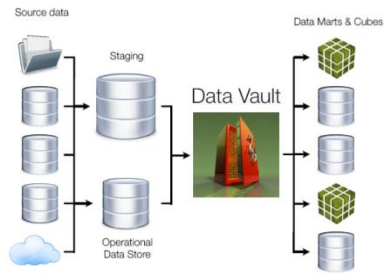
- Phase 1: retrieve facts from each fact table, applying appropriate filters, outputted in desired level of dimensional detail
- Phase 2: merge the intermediate results together
- This process can be done with any amount of fact tables
- This can also be done across different databases, as long as the dimensions involved have the same structure and content

How to Query Semi-Additive Facts

- When summing the semi-additive fact, the query must be constrained (filtered) by a unique row in the nonadditive dimension, or grouped by rows in the nonadditive dimension
- Consider the grain of the snapshot fact table to see if the SQL AVG function can be used

Data Vault

Data Vault in a Data Warehouse Architecture



Data Vault Fundamentals - Hub

- The Hub represents a core business concept as Customer, Vendor, Sale, Employee
- The hub table is formed around the Business Key of this concept
- A hub row is created the first time a specific business key is introduced to the Enterprise Data Warehouse
- The hub contains no descriptive information and no foreign keys
- The hub contains only the business key, a data warehouse ID, a load date-timestamp and a record source

Data Vault Fundamentals - Hub

H_customer
h_customer_sid
h_customer_code
h_customer_ldts
h_customer_record_source

Data Vault Fundamentals - Link

- A Link represents a natural business relationship between two or more business keys
- Just like the hub, it contains no descriptive information
- A link row is created the first time a unique association between business keys is introduced to the Enterprise Data Warehouse
- The link consists of the data warehouse IDs from the hubs that it is relating, with a data warehouse ID, a load date-timestamp and a record source

Data Vault Fundamentals - Link

L_customer_product_sale
lnk_cps_sid
lnk_customer_sid
h_product_sid
h_sale_sid
lnk_cps_ldts
lnk_cps_record_source

Data Vault Fundamentals - Satellite

- The Satellite contains the descriptive information or context for a business key
- There can be several satellites to describe a single business key (hub) or association of keys (link)
- A satellite can describe only one key (hub or link)
- The satellite is connected to a hub or link with the data warehouse ID of the hub or link
- The key of a satellite row is the hub or link key and the date-timestamp
- The satellite is the only construct that manages data warehouse history using various rows with date-timestamps to record the validity of each row
- A satellite has no foreign key constraints

Data Vault Fundamentals - Satellite

S_customer
h_customer_sid
s_customer_ldts
s_customer_ledts
customer_name
customer_address
s_customer_record_source

Choosing Satellites

- There are different reasons to put attributes or context in various satellites:
  - subject area
  - rate of change (do values change often or seldom)
  - source system (and arrival time of data)

Modelling With The Data Vault

- Identify business concepts
- Establish the enterprise wide business keys for hubs
- Model the hubs
- Identify natural business relationships
- Analyse relationships Unit of Work (relationships formed from a business perspective)
- Model the links
- Gather context attributes to keys
- Establish criteria and design satellites
- Model the satellites

Data Vault Modelling Challenges - Business Keys

- A business key is a unique identifier according to a business person
- Some business concepts may lack a visible identifier

Data Vault Load Order

- First load the hubs, so new keys are appointed to new rows in the hubs
- Secondly load the links, so new keys are appointed to new rows and the correct hub data vault keys can be assigned to each row
- Lastly load the satellites, so the correct hub or link data vault keys can be assigned to each row

Data Vault 1 or 2 - ID

- The original Data Vault uses a meaningless sequence (integer) per hub or link as an ID
- The new Data Vault uses a hash key derived from the business key
- The hash key has the advantage that parallel loading of hubs, links and satellites is possible
- The hash key ID can cause key collisions (identical keys), although the chance of this is tiny

Data Vault Advantages

- Uses mainly fast inserts into the database instead of slower updates
- Restarting a load again after an error can be done safely
- Using many-to-many relationships by default means no rework when the relationship type changes
- Traceability with the load date-timestamp and record source columns
- Use of various satellites offers flexibility and means no rework when new attributes are added or source systems change
- System of 'facts' as there is (almost) no application of business rules, cleansing or other transformations

Data Vault Considerations

- Data Vault is bad for querying, it is no substitute for data marts
- The amount of tables is higher due to the separation in hubs, links and satellites

Data Vault Exercise

- Create a Data Vault model that fits the dimensional model you created earlier
- Do it step by step:
  - Hubs
  - Links
  - Satellites
- Present and discuss results after each step