



Business Intelligence Breakfast Seminar Series

Data Warehousing
Fundamentals

[Agenda]

- Why Build a Data Warehouse?
- Designing a Data Warehouse
- Loading a Data Warehouse
- Delivering Information

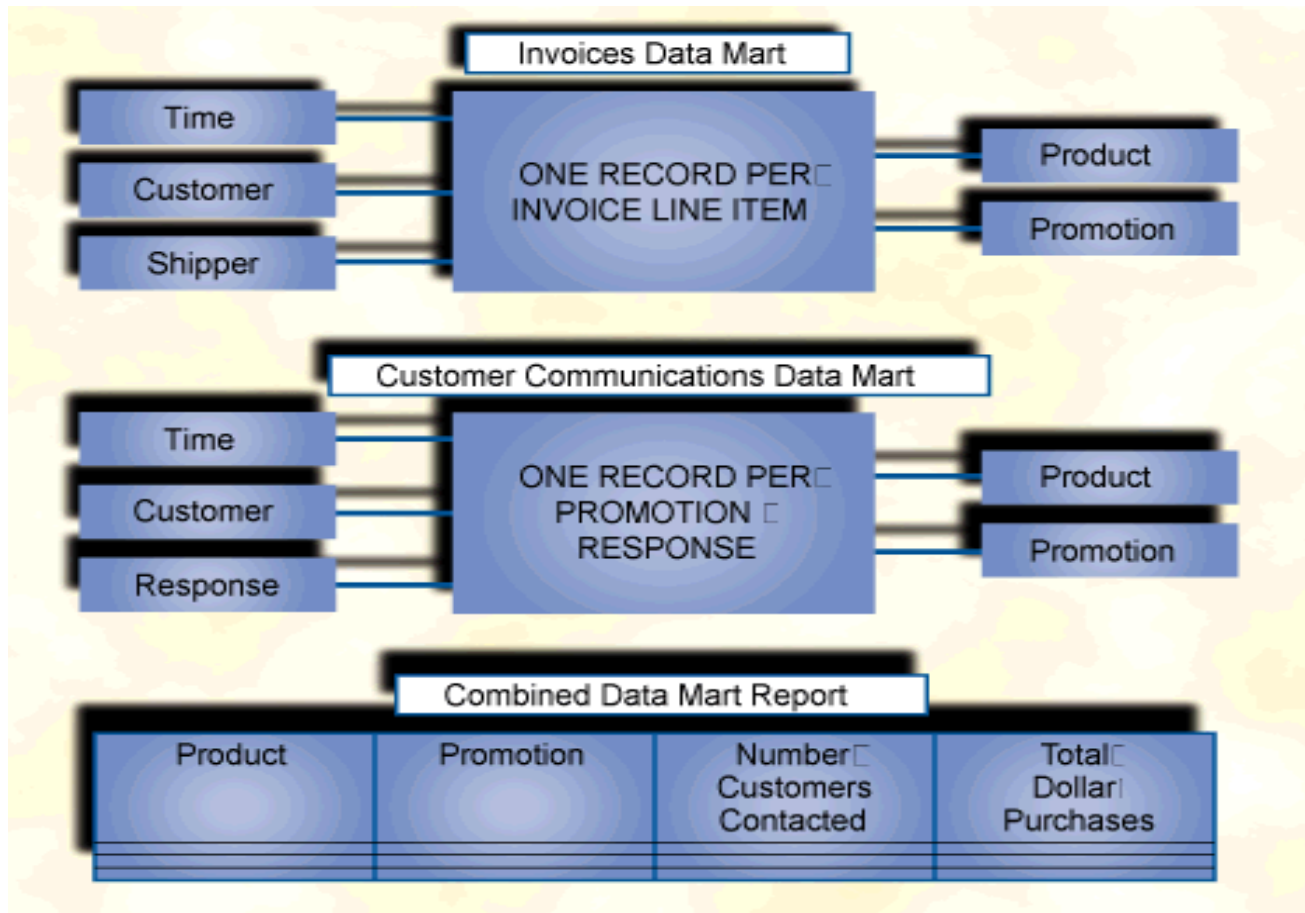
Why Build a Data Warehouse?

- **Deliver consistent and easily accessible information**
 - Reporting is not a high priority in most operational databases, therefore report development is complex and error-prone
- **Maintain history usefully and adapt to changes over time**
 - DW stores transactions as-was, while operational databases typically only present them as-is
 - DW is designed to accommodate changes in source systems
- **Integrate disparate data sources**
 - Complex integration of various data sources is handled during the data load, not during report or query execution
- **Avoid application/reporting performance conflicts**
 - Fundamentally different data access techniques will force prioritization of one function at the expense of the other

Designing a Data Warehouse

- Data Mart or Data Warehouse?
 - NIS philosophy is that a Data Warehouse is a set of conforming Data Marts.
 - Architect a warehouse, build a mart
 - Conforming dimensions
 - Fact tables based on decision-making process
 - Build aggregate fact tables across atomic fact tables

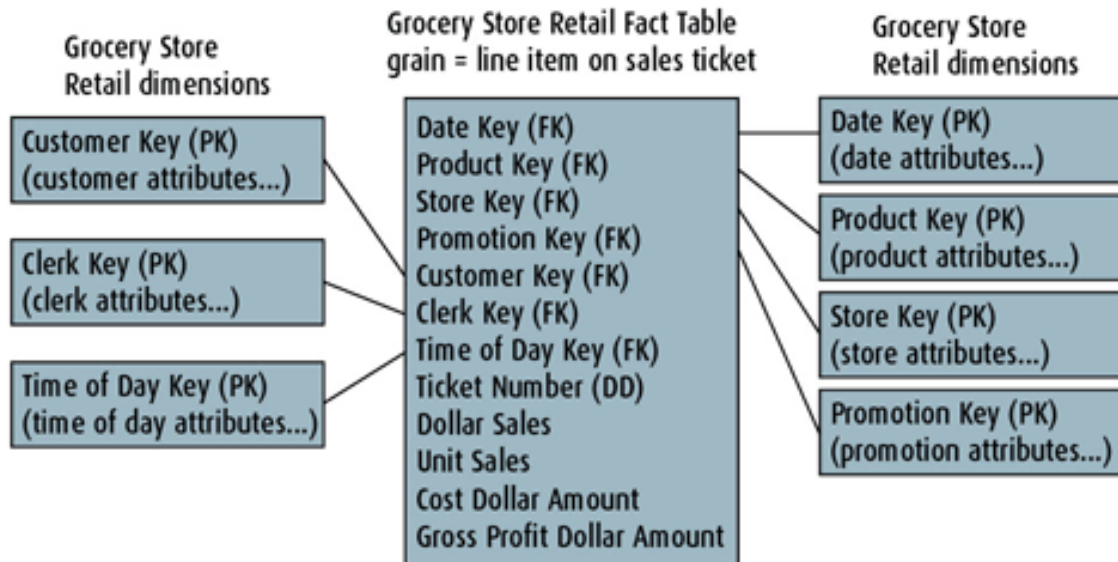
Designing a Data Warehouse



Designing a Data Warehouse

- Dimensional (Star Schema) Model
 - Single join access between attributes to analyze by and measures to analyze
 - Fact table is composed purely of numeric values
 - Can store large quantities of transactions efficiently
 - Highly indexable
 - Textual attributes are stored in smaller dimension tables

Designing a Data Warehouse



Designing a Data Warehouse

- Dimension Tables - Surrogate Keys
 - Independent unique keys generated for each dimension in the Data Warehouse
 - Eliminates dependence on source system keys
 - Accommodates changes in systems over time
 - Enables more granular historical analysis
 - Can generate a new surrogate key based on a change in any dimension attribute
 - Facts are tied to the correct 'version' of the dimension in history.

Designing a Data Warehouse

- Fact Tables
 - Transaction
 - Individual transactions from one to many source systems (ex. Retail POS system)
 - Snapshot:
 - Point-in-time measurements taken at specified time interval (ex. Monthly account balances)
 - Accumulating:
 - Pipeline measurements of a process (ex. Order fulfillment process)
 - Characterized by revisiting and updating previously loaded fact records with new information

[Loading a Data Warehouse]

- Hand-coding vs. ETL Tools
- Data Staging
- Refresh Intervals

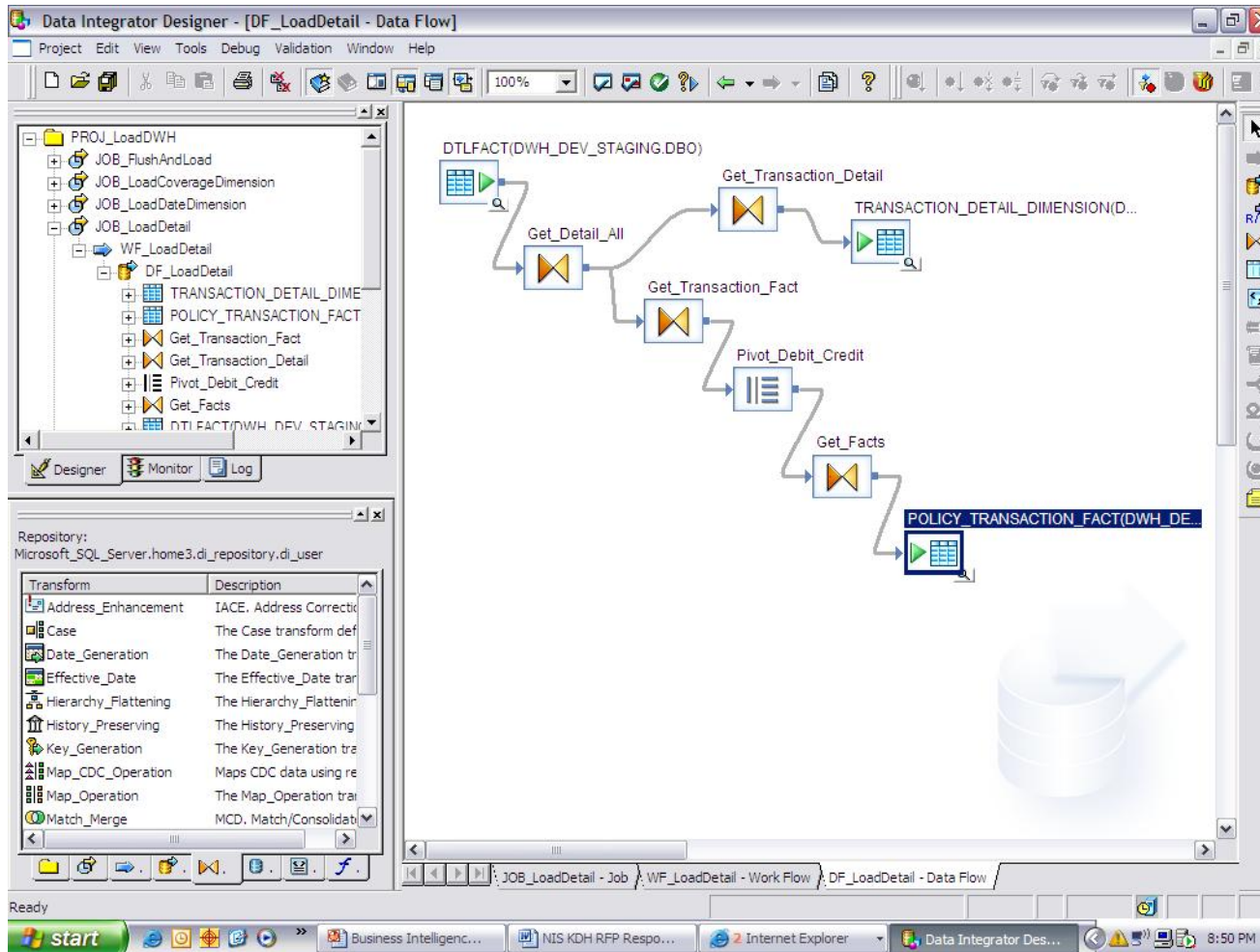
[Loading a Data Warehouse]

- Hand-coding vs. ETL Tools
 - 50% of Data Warehouse requirements surface after the first project is complete
 - Data Warehouse projects are complex and analysis-intensive, therefore application development needs to be rapid to avoid distractions
 - Data Warehouses are intended to survive decades therefore maintainability is key

[Loading a Data Warehouse]

- Hand-coding vs. ETL Tools (Cont'd)
 - Metadata: Information about information
 - Where a data element originated from
 - How a data element was transformed
 - What other names or aliases exist for a data element
 - Building and maintaining Metadata by hand is time-intensive
 - Current ETL tools will generate and maintain metadata, and store it in a standard metadata format that report and query tools can access

Loading a Data Warehouse



Loading a Data Warehouse

The screenshot displays the Data Integrator Designer interface for a transform editor. The main window is titled "Data Integrator Designer - [History_Preserving - Transform Editor]".

Schema In: POLICY_DIMENSION

- POLICYID
- RECORDEFFECTIVEDATE
- RECORDEXPIRATIONDATE
- CURRENTRECORDINDICATOR
- POLICYNUMBER
- ACCOUNTNUMBER
- VERSIONNUMBER
- VERSIONSTARTDATE
- VERSIONSTARTREASON
- VERSIONSTOPDATE
- VERSIONSTOPREASON
- VERSIONREPORTSTOPDATE

Schema Out: History_Preserving

- POLICYID
- RECORDEFFECTIVEDATE
- RECORDEXPIRATIONDATE
- CURRENTRECORDINDICATOR
- POLICYNUMBER
- ACCOUNTNUMBER
- VERSIONNUMBER
- VERSIONSTARTDATE
- VERSIONSTARTREASON
- VERSIONSTOPDATE
- VERSIONSTOPREASON
- VERSIONREPORTSTOPDATE

History Preserving Settings:

- Date columns:** Valid from: RECORDEFFECTIVEDATE, Valid to: RECORDEXPIRATIONDATE, Valid to date value: 9999.12.31 (yyyy.mm.dd)
- Current flag:** Column: CURRENTRECORDINDICATOR, Set value: 1, Reset value: 0
- Preserve delete row(s) as update row(s)
- Compare columns:** VERSIONSTARTREASON, VERSIONSTARTDATE, VERSIONSTOPDATE, VERSIONSTOPREASON, VERSIONREPORTSTOPDATE, COMPANY, POLICYPREFIX, POLICYSUFFIX, ISSUEDSTATE, POLICYSTATUSCODE, ORIGINALPOLICYEFFECTIVEDAT, CURRENTPOLICYISSUEDATE, POLICYSTATUS, ANNUALPREMIUMAMOUNT

The interface also shows a project tree on the left with various data flows and a repository list at the bottom left. The taskbar at the bottom indicates the system is ready and shows the time as 8:55 PM.

[Loading a Data Warehouse]

- Data Staging
 - Data Warehouse work area
 - Minimize impact on source systems by extracting data in its original form
 - Utilize for change detection between current data extracts and previous extracts
 - Identifying and loading only changed data will improve data warehouse load performance and avoid headaches

[Loading a Data Warehouse]

- Data Warehouse Refresh Intervals
 - Dependent on the type of fact table and the source systems
 - Dimensions must always be refreshed immediately prior to refreshing a related fact table
 - Exception: Static dimensions such as Time
 - Real-time is attractive for the operational reporting benefits, but can cause the appearance of inconsistency to end-users

[Next Time]

- Delivering Information
 - Server-based Report Execution
 - Centralized Report Delivery
 - Historical Report Repository