Lecture 3

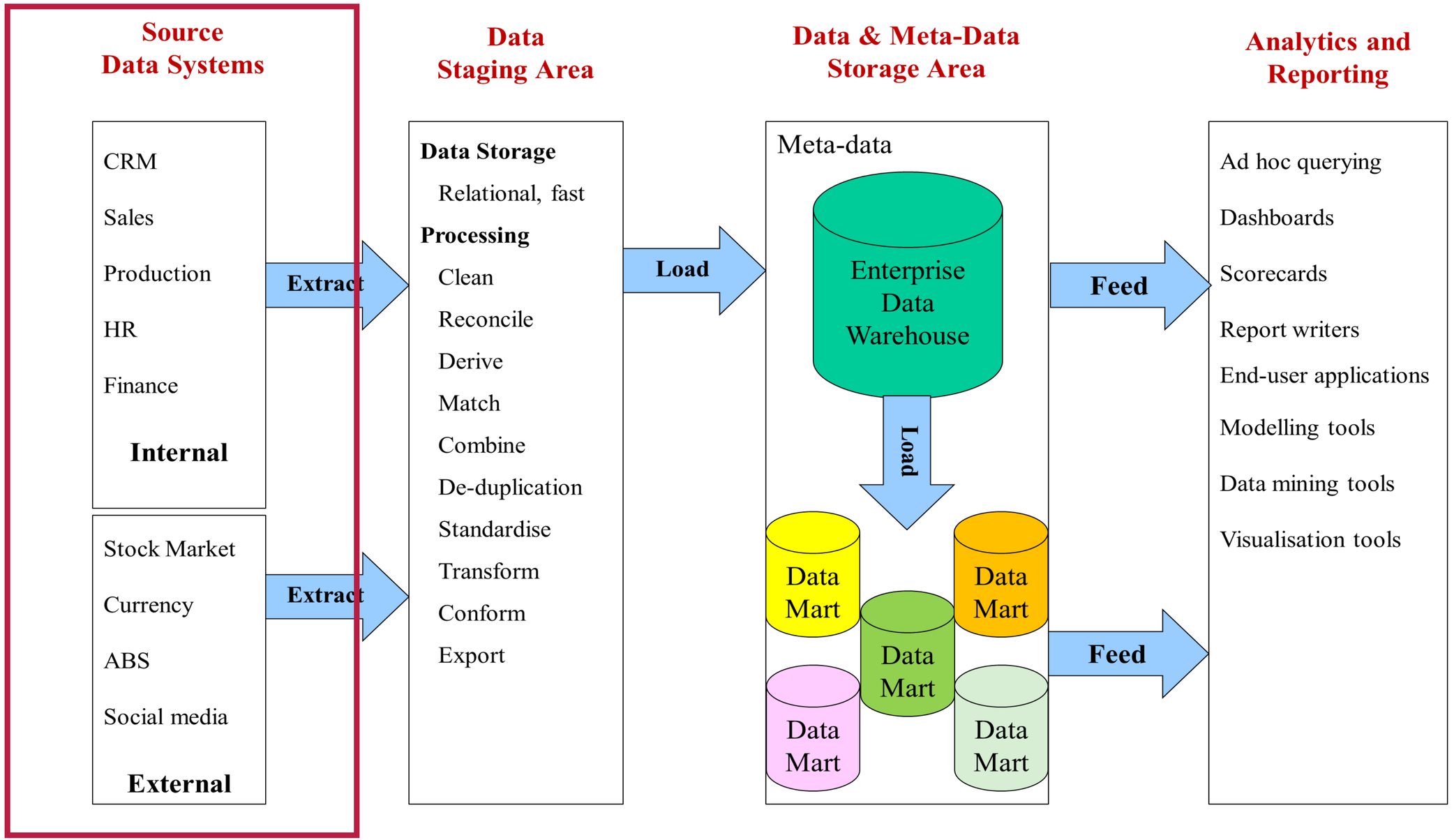
Dimensional Modelling

Bikesh Raj Upreti
E-mail: b.upreti@uq.edu.au
Room: 530 Joyce Ackroyd (37) Building

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

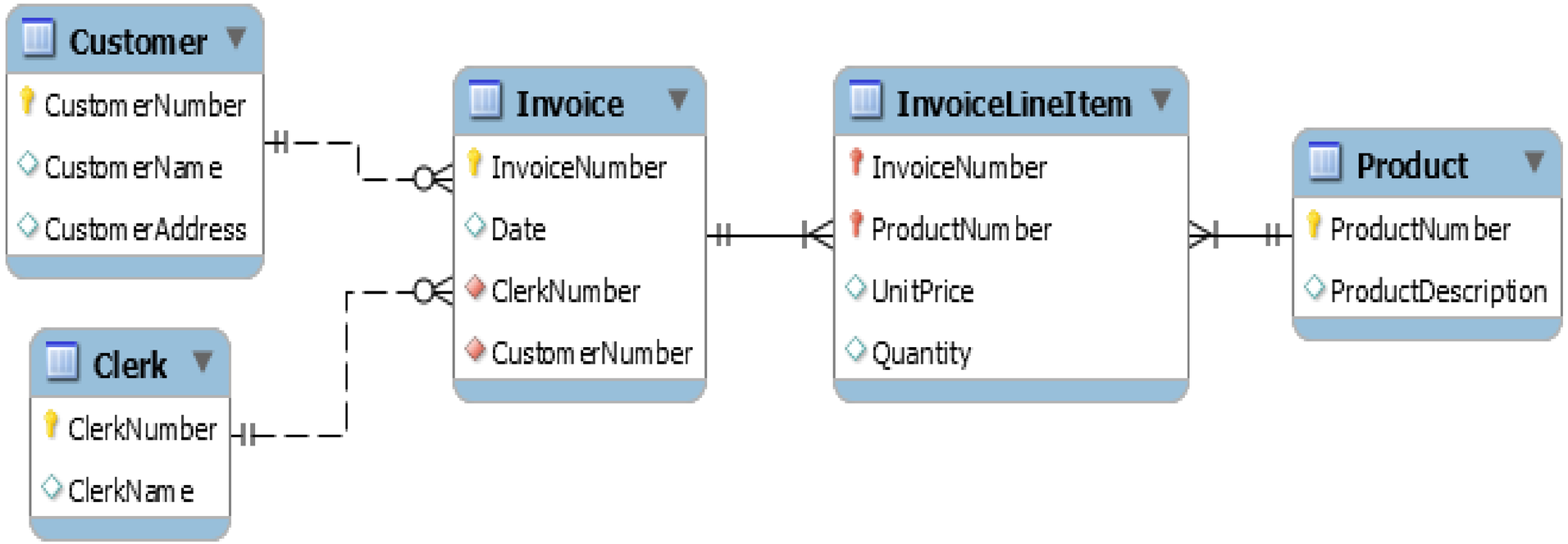
UQ BUSINESS SCHOOL

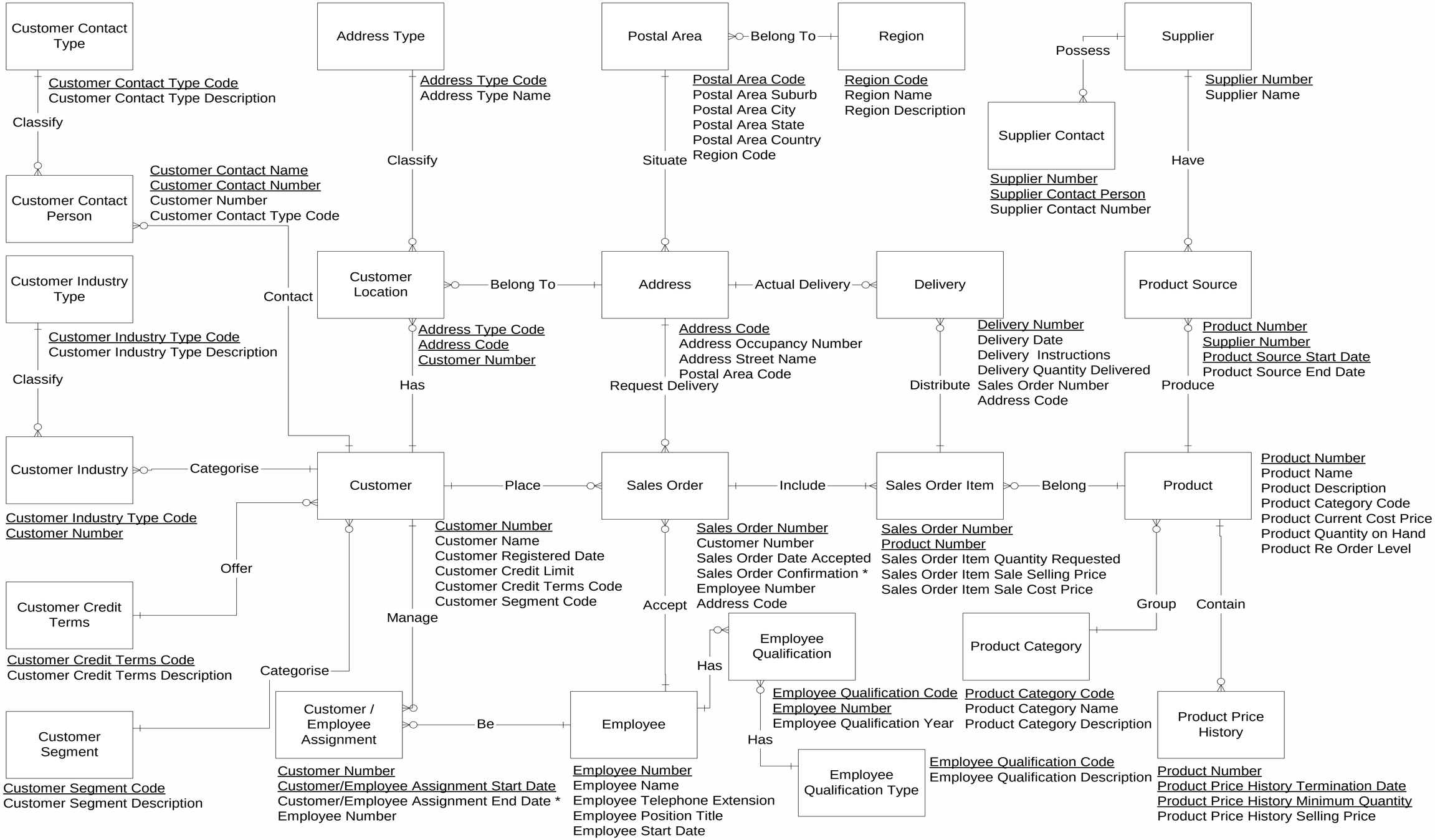
Recap: Business Analytics Framework



Recap: Invoice example (Normalized Relations and ER Diagram)

- We can name the relations now
  - Customer (CustomerNumber, CustomerName, CustomerAddress)
  - Clerk (ClerkNumber, ClerkName)
  - Product (ProductNumber, ProductDescription)
  - Invoice (InvoiceNumber, Date, CustomerNumber, ClerkNumber)
  - InvoiceLineItem (InvoiceNumber, ProductNumber, UniPrice, Quantity)




Customer Contact Type
Customer Contact Type Code
Customer Contact Type Description
Classify
Customer Contact Person
Customer Contact Name
Customer Contact Number
Customer Number
Customer Contact Type Code
Customer Industry Type
Customer Industry Type Code
Customer Industry Type Description
Classify
Customer Industry
Categorise
Customer Industry Type Code
Customer Number
Contact
Address Type
Address Type Code
Address Type Name
Classify
Customer Location
Belong To
Address Type Code
Address Code
Customer Number
Has
Customer
Place
Customer Number
Customer Name
Customer Registered Date
Customer Credit Limit
Customer Credit Terms Code
Customer Segment Code
Manage
Offer
Customer Credit Terms
Customer Credit Terms Code
Customer Credit Terms Description
Categorise
Customer Segment
Customer Segment Code
Customer Segment Description
Customer / Employee Assignment
Be
Customer Number
Customer/Employee Assignment Start Date
Customer/Employee Assignment End Date *
Employee Number
Postal Area
Belong To
Postal Area Code
Postal Area Suburb
Postal Area City
Postal Area State
Postal Area Country
Region Code
Situate
Address
Address Code
Address Occupancy Number
Address Street Name
Postal Area Code
Request Delivery
Sales Order
Sales Order Number
Customer Number
Sales Order Date Accepted
Sales Order Confirmation *
Employee Number
Address Code
Accept
Include
Employee
Employee Number
Employee Name
Employee Telephone Extension
Employee Position Title
Employee Start Date
Has
Employee Qualification
Employee Qualification Code
Employee Number
Employee Qualification Year
Has
Employee Qualification Type
Employee Qualification Code
Employee Qualification Description
Region
Region Code
Region Name
Region Description
Actual Delivery
Delivery
Delivery Number
Delivery Date
Delivery Instructions
Delivery Quantity Delivered
Sales Order Number
Address Code
Distribute
Sales Order Item
Sales Order Number
Product Number
Sales Order Item Quantity Requested
Sales Order Item Sale Selling Price
Sales Order Item Sale Cost Price
Belong
Possess
Supplier Contact
Supplier Number
Supplier Contact Person
Supplier Contact Number
Supplier
Supplier Number
Supplier Name
Have
Product Source
Product Number
Supplier Number
Product Source Start Date
Product Source End Date
Produce
Product
Product Number
Product Name
Product Description
Product Category Code
Product Current Cost Price
Product Quantity on Hand
Product Re Order Level
Group
Product Category
Product Category Code
Product Category Name
Product Category Description
Contain
Product Price History
Product Number
Product Price History Termination Date
Product Price History Minimum Quantity
Product Price History Selling Price

Recap: Querying a Database (Four Primary Operations of SQL)

Operation	SQL Command
Create	INSERT
Read	SELECT
Update	UPDATE
Delete	DELETE

Recap: SELECT statement

SELECT [ALL | DISTINCT] *select_expr* [, *select_expr* ...]
List the columns (and expressions) that are returned from the query

[FROM *table_references*
Indicate the table(s) or view(s) from where the data is obtained

[WHERE *where_condition*]
Indicate the conditions on whether a particular row will be in the result

GROUP BY {*col_name* | *expr* } [ASC | DESC], ...]
Indicate categorisation of results

HAVING *where_condition*]
Indicate the conditions under which a particular category (group) is included in the result

ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...]
Sort the result based on the criteria

[LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]
Limit which rows are returned by their return order (ie 5 rows, 5 rows from row 2)**]**

Recap: SELECT Example 1

Customer	
PK	**CustomerID**
	CustFirstName CustMiddleName **CustLastName** BusinessName **CustType**

SQL →

```
SELECT CustLastName FROM Customer;
```

RESULT →

CustLastName
Smith
Jones
Smithies
Smythe
Smart
Lam
Unila
Jones
Samson

The names of the attributes that we want data from in the table

The TABLE (name) we want to query

Recap: SELECT Example 2

Customer	
PK	**CustomerID**
	CustFirstName CustMiddleName **CustLastName** BusinessName **CustType**

SQL →

```
SELECT * FROM Customer;
```

RESULT →

Edit: Export: Autosize:

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType
1	Peter	NULL	Smith	NULL	Personal
2	James	NULL	Jones	JJ Enterprises	Company
3	Akin	NULL	Smithies	Bay Wart	Company
4	Julie	Anne	Smythe	Konks	Company
5	Jen	NULL	Smart	BRU	Company
6	Lim	NULL	Lam	NULL	Personal
7	Kim	NULL	Unila	Saps	Company
8	James	Jay	Jones	JJ's	Company
9	Keith	NULL	Samson	NULL	Personal
NULL	NULL	NULL	NULL	NULL	NULL

Recap: An SQL Primer: GROUP BY

Aggregating data by particular attribute

Customer	
PK	**CustomerID**
	CustFirstName CustMiddleName **CustLastName** BusinessName **CustType**

SQL →

```
SELECT CustType, Count(CustomerID)
    FROM Customer
    GROUP BY CustType;
```

RESULT →

Export: Autosize:

CustType	Count(CustomerID)
Personal	3
Company	6

Logic: Count (Customer ID) will return the number of customers,
Group BY CustType will group the result based on CustType

Recap: SQL Joins – Inner JOIN

- Inner Join the tables with foreign keys!

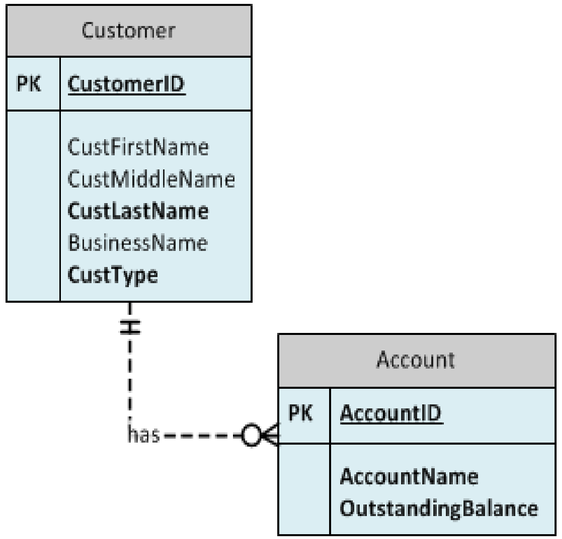
SQL →

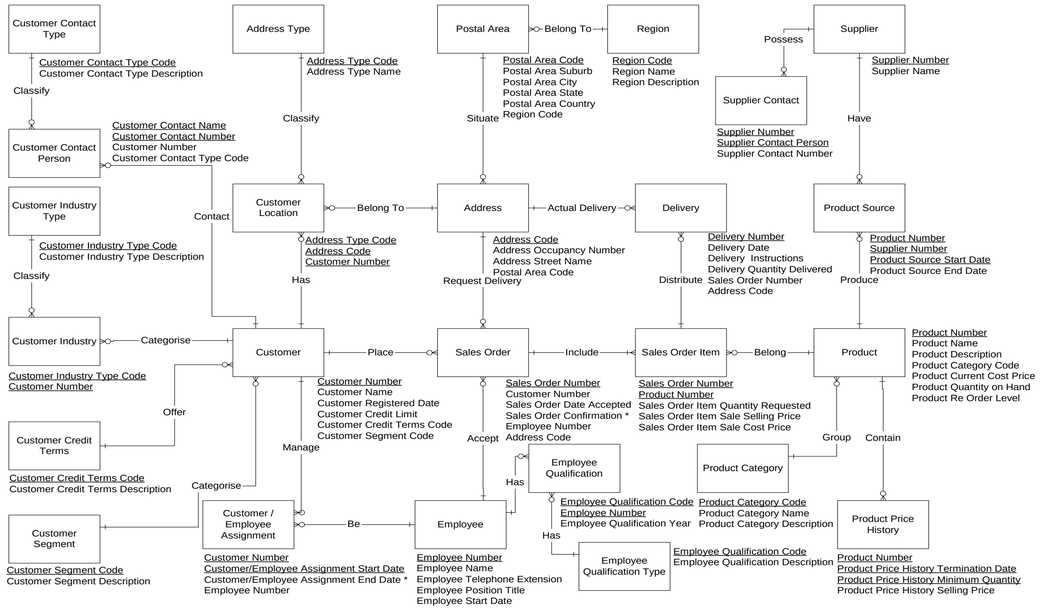
```
SELECT * FROM Customer INNER JOIN Account
    ON Customer.CustomerID = Account.CustomerID;
```

RESULT →

Export: Autosize:

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance	CustomerID
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25	1
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ ENt.	552.39	2
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ ENt. Mgr	10.25	2





ER Modelling Task

The Brisbane Movie Library purchases movies on various formats and loans them to its members for a charge in order to make a profit. The business is designing a new information system.

The proposed new system will include an accurate catalogue to inform members of movies held in each store by a number of different categories (eg. action, comedy, etc.) or which movies are held featuring their favourite actors. The catalogue will also show if a particular movie is available that day at a particular store.

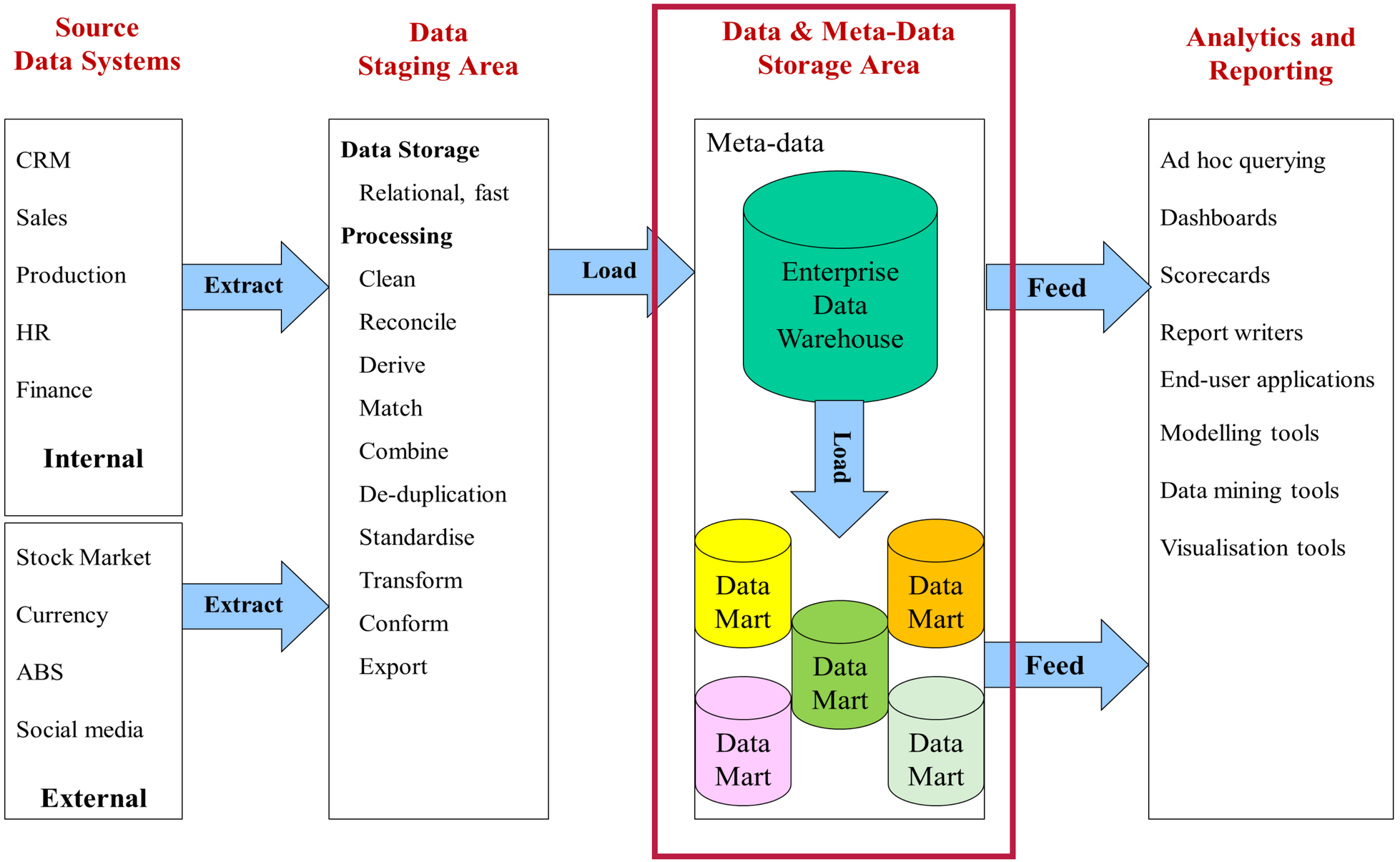
Accurate information about which members have borrowed which movies, and when movies are due to be returned will also be available. This should encourage borrowers to return their movies promptly. Keeping track of loans using the current membership system has proven to be slow and prone to error. Improved turnaround of movies should increase profit.

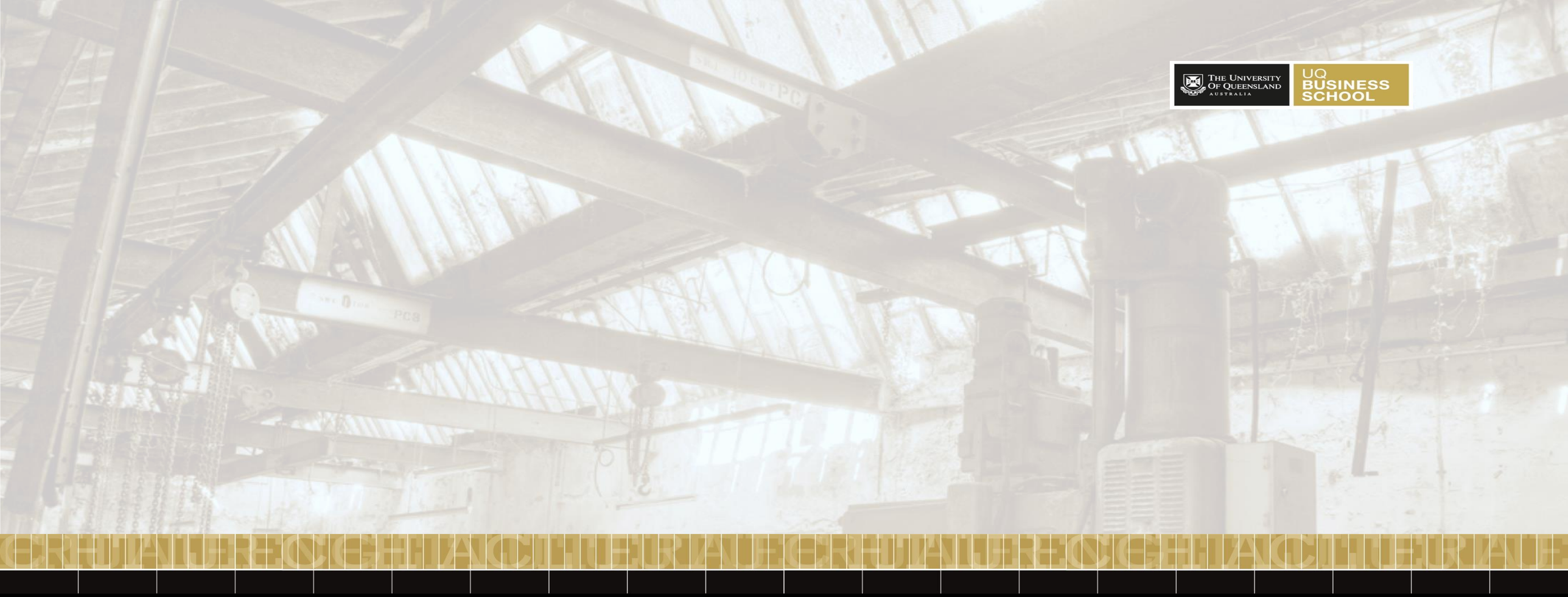
In order to keep track of the costs involved in purchasing movies, details of purchase orders will be stored for all movies. This information will help to select suppliers, negotiate cheaper prices for future purchases, and help with auditing.

Each movie is allocated a rental charge and all loans are for one day (24 hour period). Occasionally, a special member may be given a longer loan period. All overdue movies incur an excess charge of $2 per day for each day they are late. While members will be encouraged to return movies to the store from which they borrowed them, the new system should also make it easier to keep track of movies returned to other stores.

Brisbane Movie Library – ER Model

Recap: Business Analytics Framework





Transactional vs Informational Databases

Agenda and Learning Objectives for today

By the end of this class you should be able to:

- Identify difference between **informational** and **transactional questions**
- Explain the differences between **transactional** and **informational databases**
- Define and develop **dimensional data models** for data-driven decision

Transactional Databases

- Support operations of an organization (running transactions)
  - Selling a products, shipping, hiring, supplying
- Store data from every-day transactions
- Highly normalised to avoid redundancy of data
- Optimised to write new data in as transactions happen (because of normalised structure)

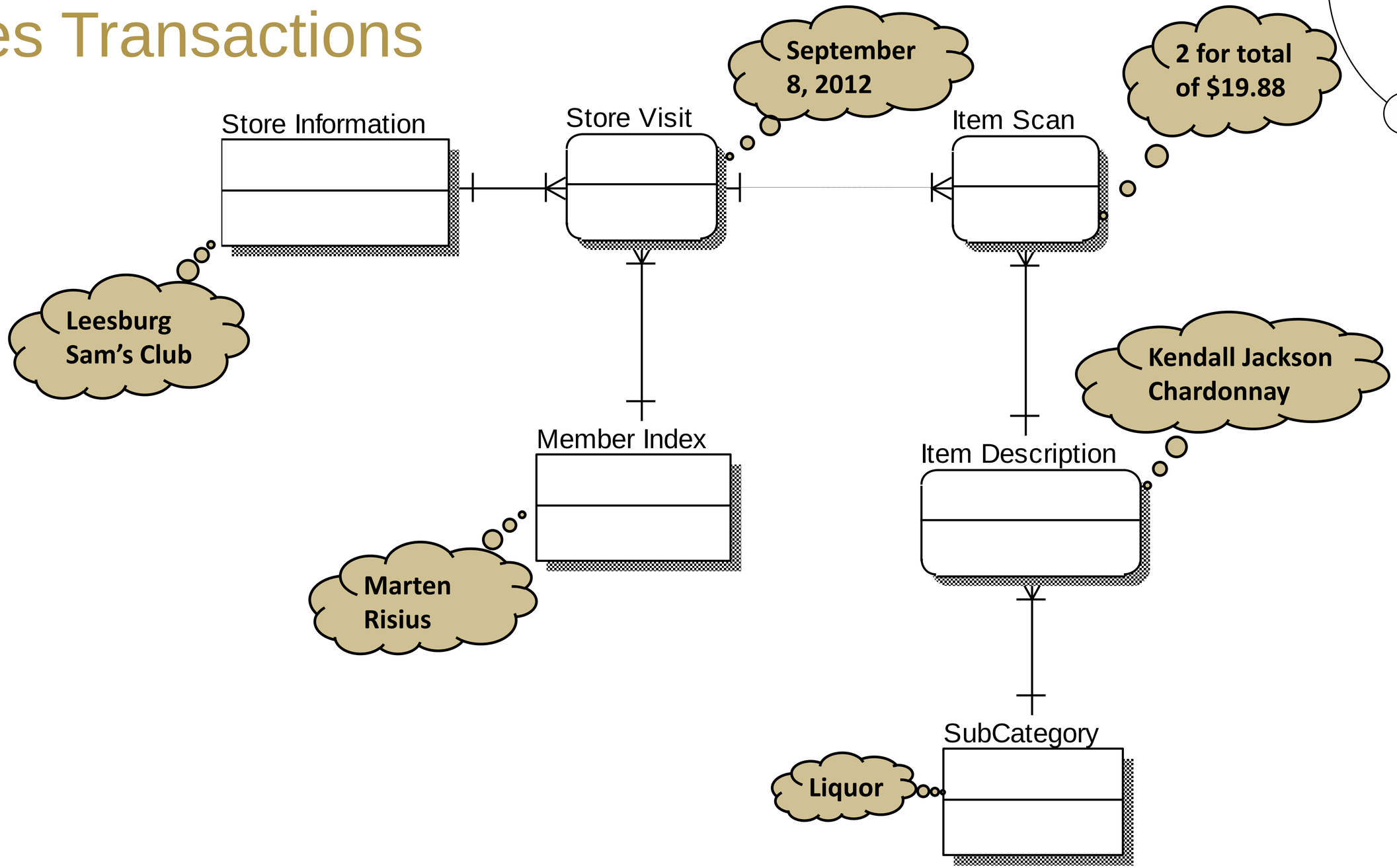
Is normalization good for analytical decision-making purposes?

Let’s look at the two types of databases:

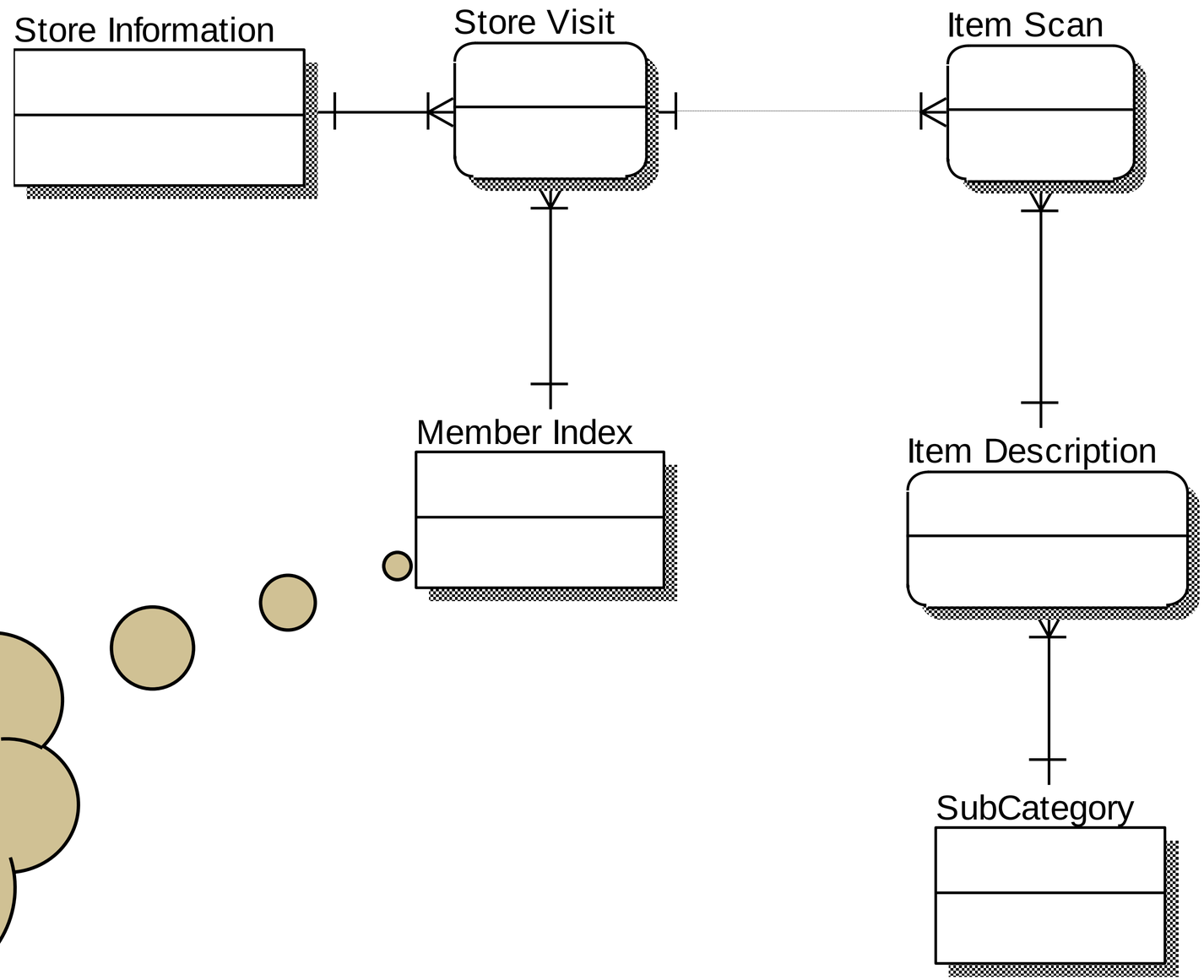
- Transactional databases
  - used to answer operational questions

- Informational (Analytical) databases
  - used to answer strategic questions

Sales Transactions



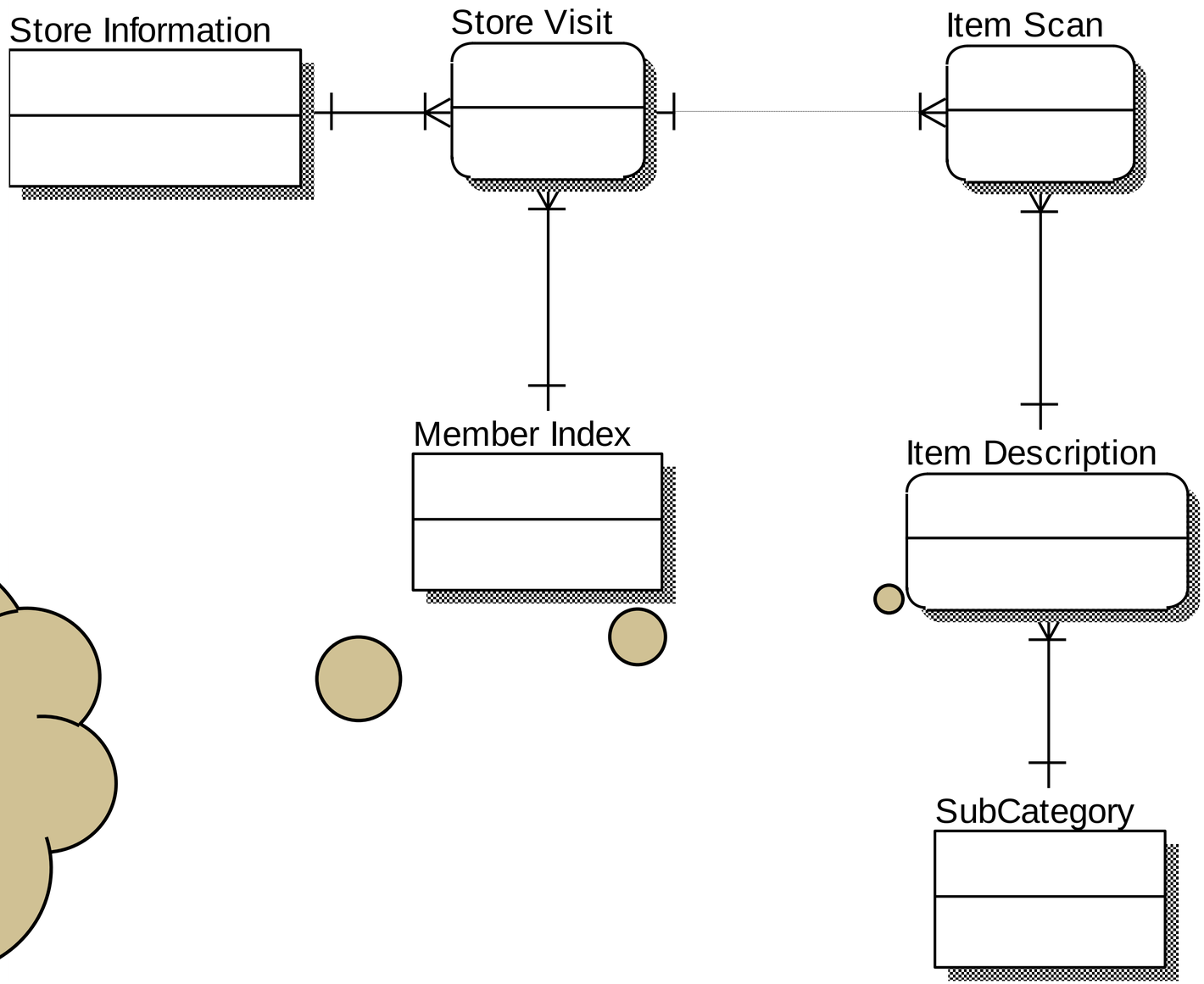
Transactional (Operational) Questions



Customer Service: Help! I forgot my membership card!

Select membership_nbr from MEMBER_INDEX where phone_num = ‘555-1212’

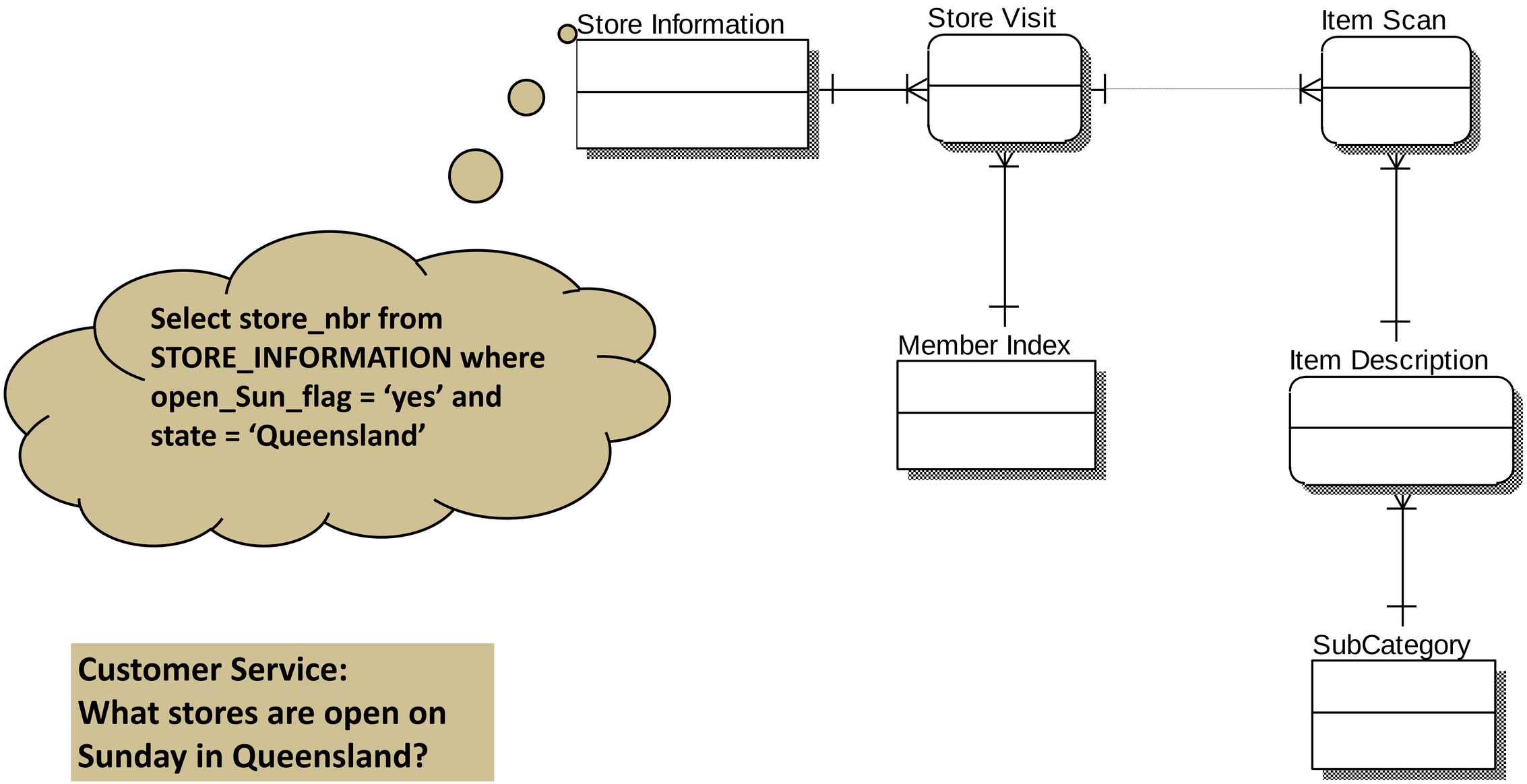
Transactional (Operational)



Inventory:
Where do you carry Kendall Jackson chardonnay?

Select item_location from ITEM_DESCRIP where item_name = ‘Kendall Jackson chardonnay’

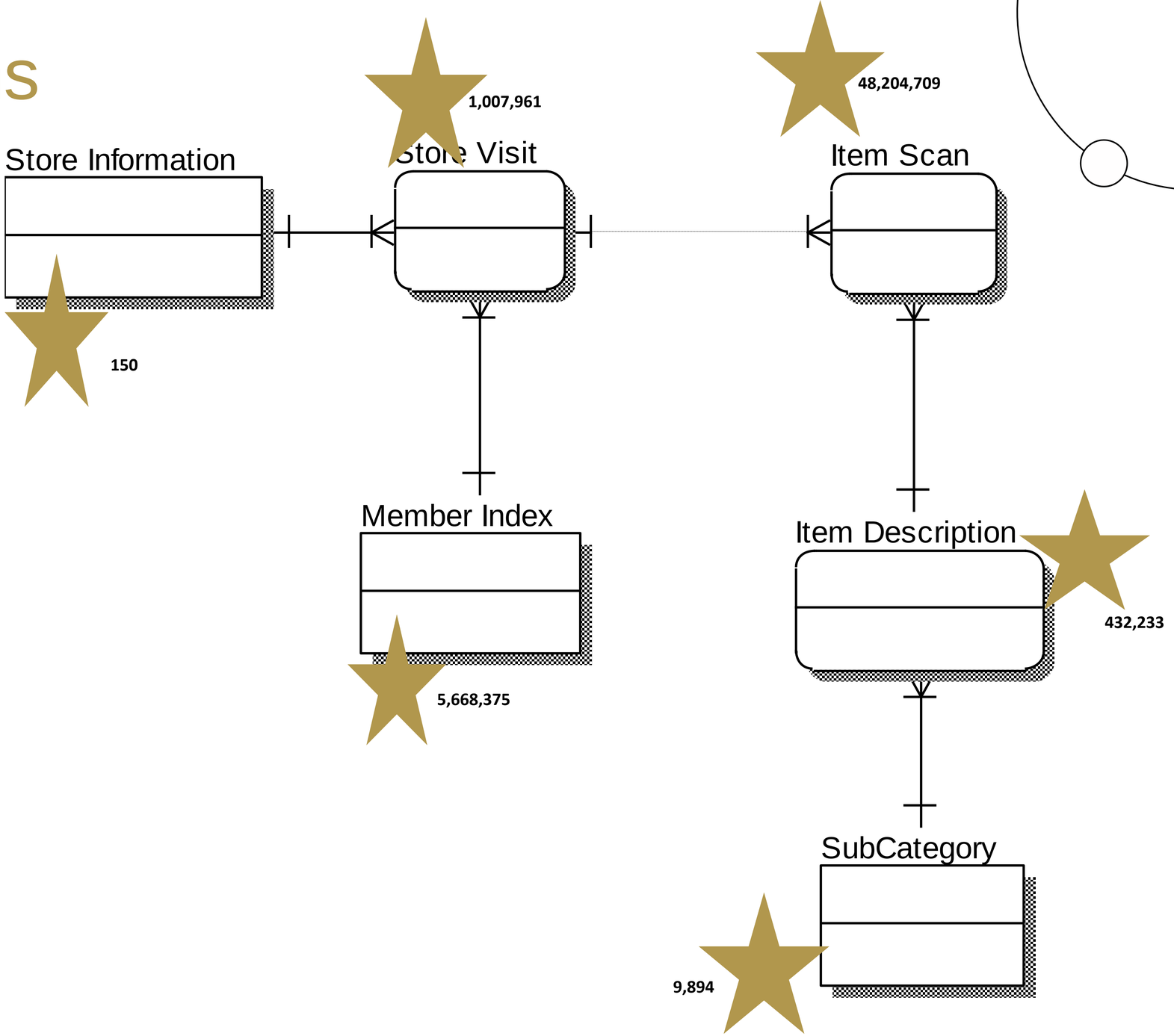
Transactional (Operational)



Customer Service:
What stores are open on Sunday in Queensland?

Analytical Questions

Campaign Management: How many customers purchased more than $500 worth of alcohol in our Brisbane stores this year?



With business analytics, we are interested in analytical queries

- One is interested in numerical aggregations
  - How many?
  - What is the average?
  - What is the total cost?

- One is interested in understanding dimensions
  - Sales by state by customer type
  - Sales by product by store by quarter

Transactional vs Informational Databases

Transactional Databases

- Focus is on supporting day to day operations
  - Recording orders
  - Processing claims
  - Making shipments
  - Generating invoices
  - Receiving cash
  - Reserving airline seats

Informational Databases

- Have a different scope & different purpose
  - Show me the top products
  - Show me problem regions
  - Tell me why (drill down)
  - View other data (drill across)
  - Show the highest margins
  - Alert me if calls are high
- Focus is on getting information at a higher level suitable for decision-making

Transactional vs Informational Databases

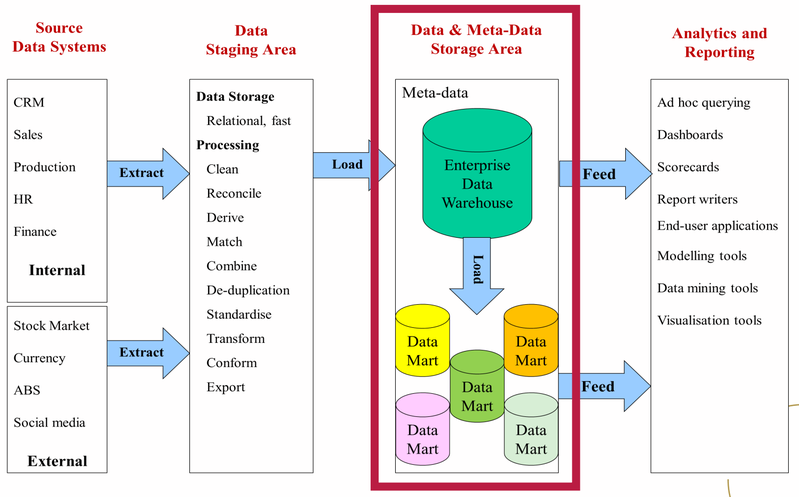
	Transactional	Informational
Data Content	Current Values	Archives, derived, summarised
Data Structure	Optimised for transactions (lots of writes)	Optimised for complex queries
Access Frequency	Very High	Medium
Access Type	Read, update, delete	Read
Usage	Predictable, repetitive	Ad hoc, random, heuristic
Response Time	Sub-seconds	Seconds to Minutes
Users	Many	Relatively few

So for decision-making purposes, we need an Informational Database

- Designed for analytic tasks
- Gets data from multiple locations
  - Internal / external
- Intuitive and easy to use
  - Allows direct access by users without IT support
- Conducive to long analysis sessions
- Read intensive
- Updated at known intervals and is stable
- Storing historical data also
- Able to allow users to run queries and get results online
- Able to allow users to initiate reports

Our Solution

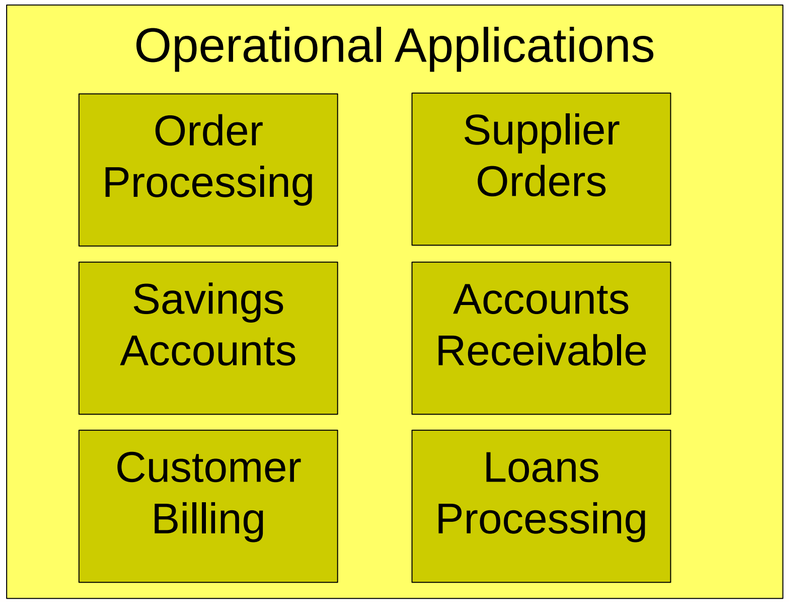
- The Data Warehouse!
- So, what's a data warehouse?
  - A single repository of organisational data
    - Current and historical
  - Integrates data from multiple sources
    - Internal and external
    - Extracts data from source systems, transforms, loads into the warehouse
    - "Single version of truth" – a holistic integrated view of organization data
  - Makes data available to managers/users
    - Without hindering day to day transactional work
  - It's a database! But it is denormalised…

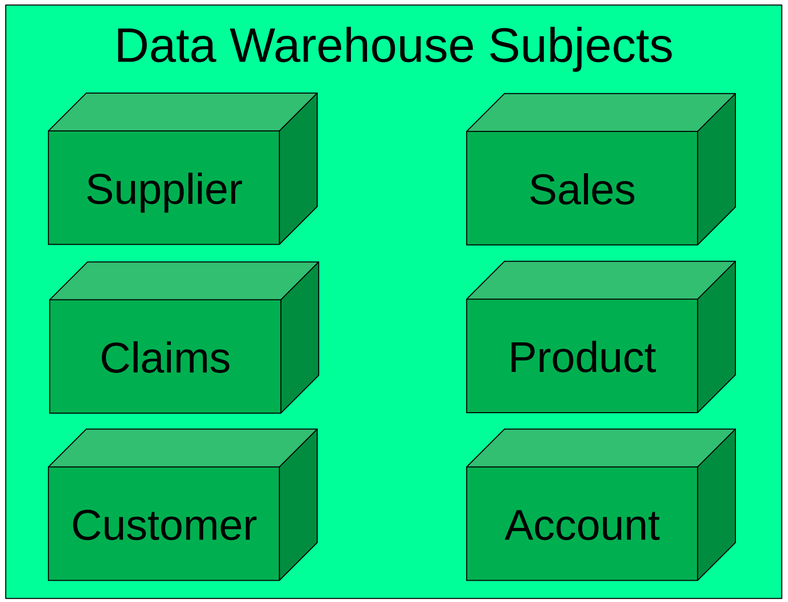


Data Warehouse Features

Defining Features

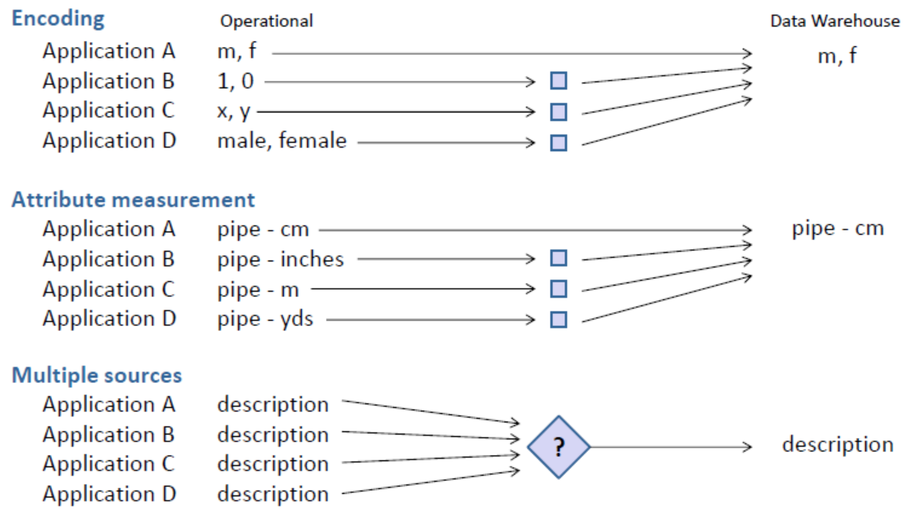
- Subject Oriented Data
  - Data warehouses are organised around particular subjects
    - Data is integrated across functions
      - sales, customers, products
    - Data in a DW cuts across Application requirements





Defining Features

- Integrated Data
  - Data from different systems
    - Can be from different applications, operating systems, etc
    - File layouts, field naming conventions could be different
    - Local information could be different
  - Need to convert to a common format
    - allows comparison and consolidation of data from different sources
  - Data from various sources are validated before storing them in a data warehouse.
  - Data quality is crucial to the credibility of the warehouse

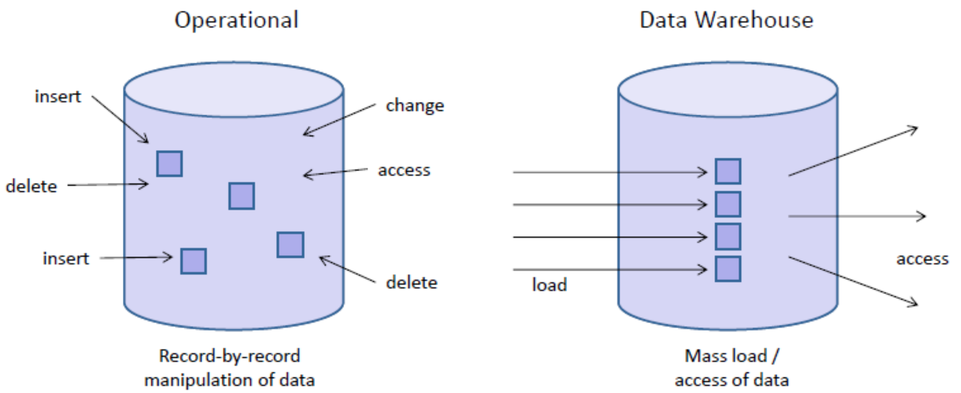


Defining Features

- Time-Variant Data
  - In application systems the data is current
    - i.e. The current true (or correct) value
  - In a Data Warehouse
    - Data used for analysis and decision making
    - Need current and past data = Historical data
      - Otherwise can’t answer many analytical questions
  - Data is stored as snapshots of the current values
    - Snapshots are time stamped
    - Data changes stored over time
  - Allows
    - Analysis of the past
    - Relation of data to the present
    - Forecasting for the future

Defining Features

- Non-Volatile Data
  - Unlike transaction systems the DW doesn’t get updated every time the data changes
  - Store extracted data snapshots over time
  - Data is periodically updated
    - That could be every second, hour, day, week or even month
    - Different data items updated with different frequencies
  - Users have read access only
    - all updating done automatically by ETL process and periodically by DB Administrator



- Regular data accesses
- Regular data updates

- Data can not be manipulated (regular snapshots are loaded to the system)

→ read-only access

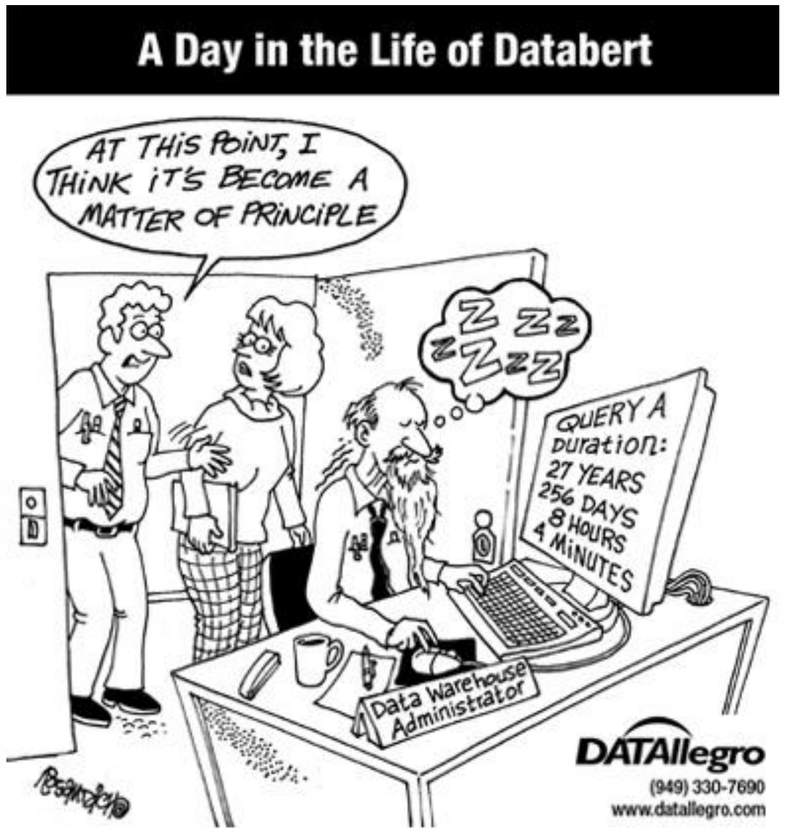
Inmon 2002, p. 33-34

Defining Features

- Data Granularity
  - Operational systems
    - Data kept at lowest level of detail
    - Summary data created by adding up the numbers
      - Its not stored
  - Informational systems
    - Queries usually start with summary data
      - Then as analysis occurs more detailed levels of data are needed
    - Data usually stored at various levels for efficiency
  - Data granularity is the level of detail
    - The finer the granularity the lower the level of detail
    - The lowest level of granularity is called “the grain”

Defining Features

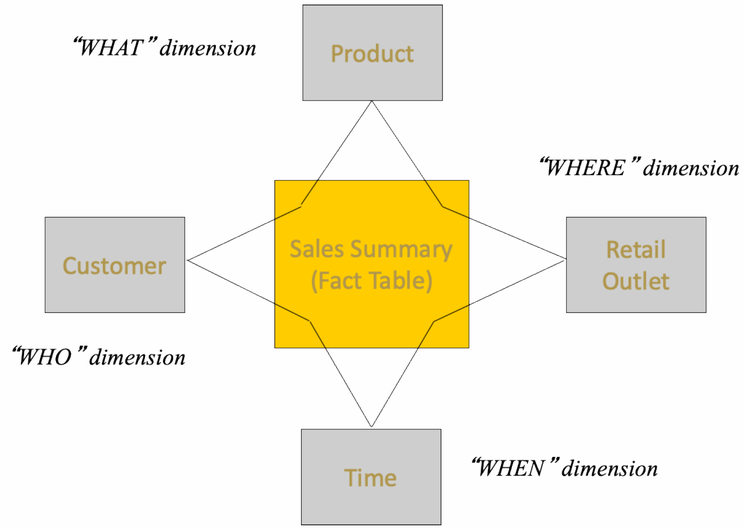
- Supports management needs
  - Used by end users
    - Data warehouses require a simple and easy to navigate structure
    - Responses to queries should be “timely”



Data Warehouse Design: Dimensional Modelling (Kimball)

Business Analyst World

- How much **revenue** did the **product G** generate in the **last three months**, broken down by month for the south eastern sales **region**, by individual **stores**, broken down by **promotions**, compared to estimates and to the previous version of the product
  - Analysis starts usually with a single indication of something strange, then goes deep into the data, left to a new dimension, right to another, up to the summary, back down and left and right again, until the problem is identified…

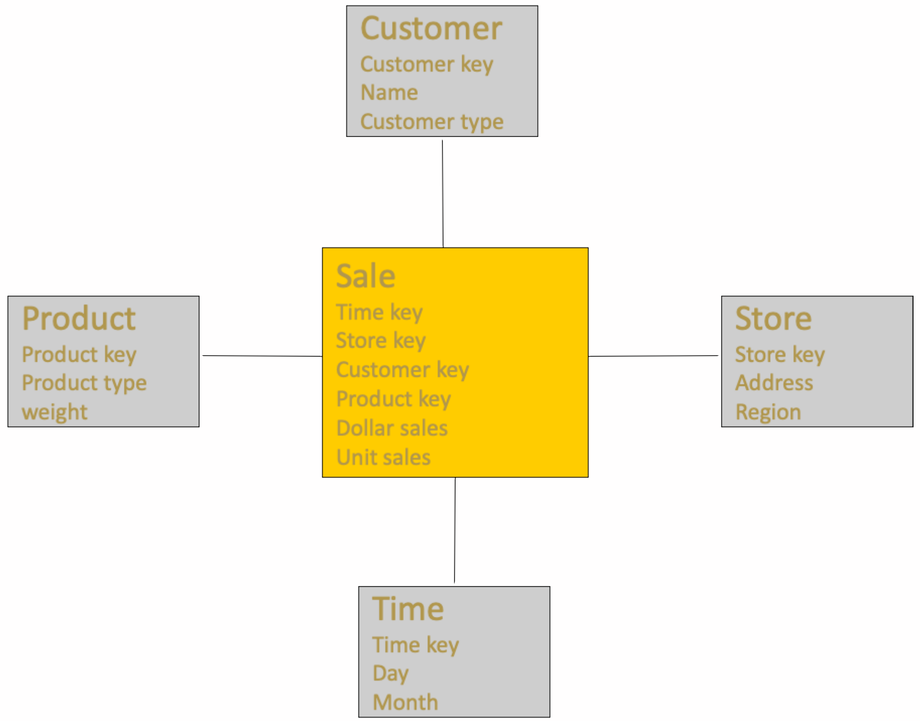


Introduction to Dimensional Modelling

- Popularised by Ralph Kimball in the 1990s
- Based on the multi-dimensional model of data and designed for retrieval-only databases
- Very simple, intuitive, and easily-understood structure
- Also known as star schema design

Dimensional Modelling- Objectives

- Produce database structures that are easy for end users to understand and write queries against
- Optimise query performance (as opposed to update performance)



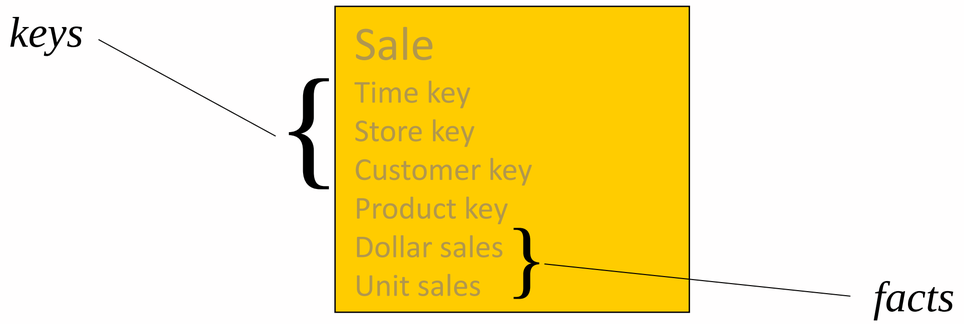
Dimensional Modelling

- A dimensional model consists of
  - a fact table
  - several dimensional tables
  - hierarchies in the dimensions

- Essentially a simple and restricted type of ER model

Fact Table

- A fact table contains the actual business measures (additive), called facts
- Also contain foreign keys for dimensions



Fact Table - example

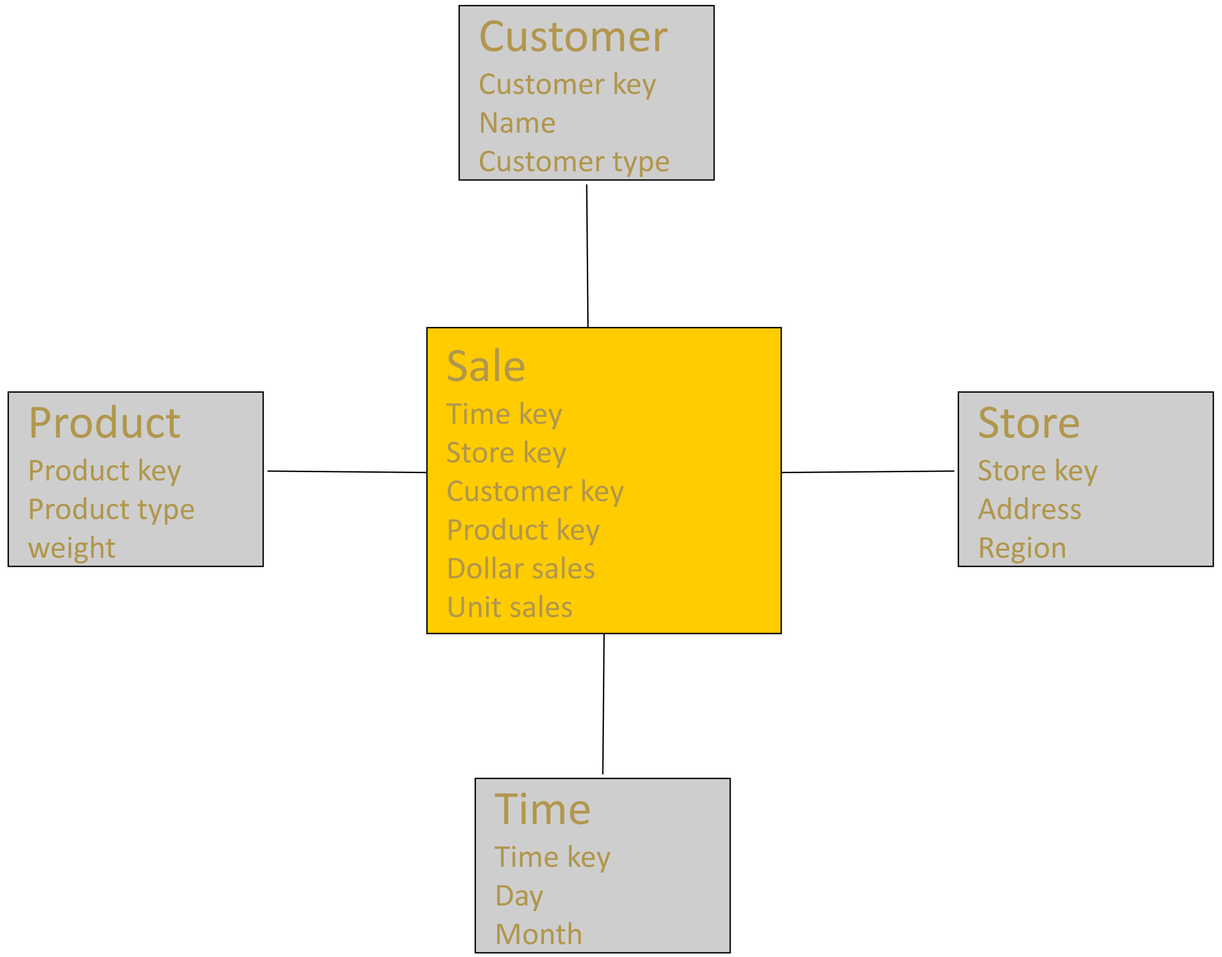
- Actual data might look like this
- Granularity, or level of detail, is a key issue
  - Finest level of detail for a fact table
  - Determined by the finest level of each dimension

Time-id	*Store-id*	*Cust-id*	*Prod-id*	*Dollar sales*	*Unit Sales*
T100	S303	C101	P98	$120,000	5,000
T101	S303	C256	P98	$240000	10,000
T102	S387	C101	P10	$456,000	27,899
T100	S234	C400	P56	$100,200	5,600

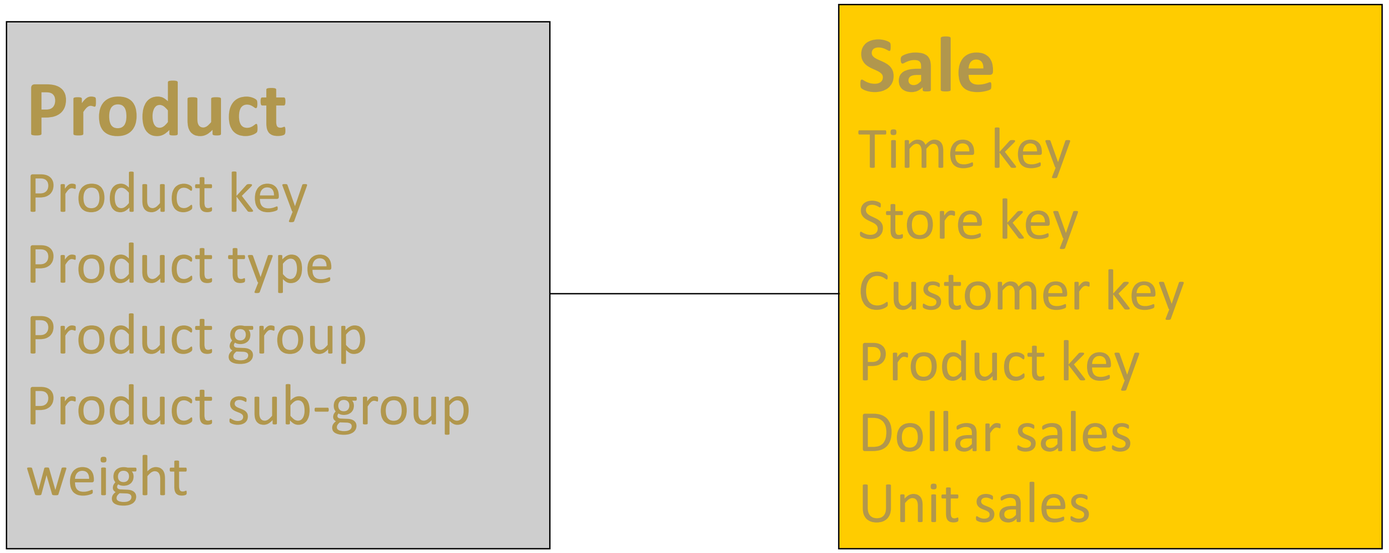
Grain Example

- Rough: customer postal codes (5,000), product type (200), store (300), week (52)
- Detailed: individual customer (200,000), individual product (2,000), store (200), day (365)
- Impact
  - Higher storage requirements for fine grain
  - More reporting flexibility for fine grain

Dimension Tables



Dimension Hierarchies



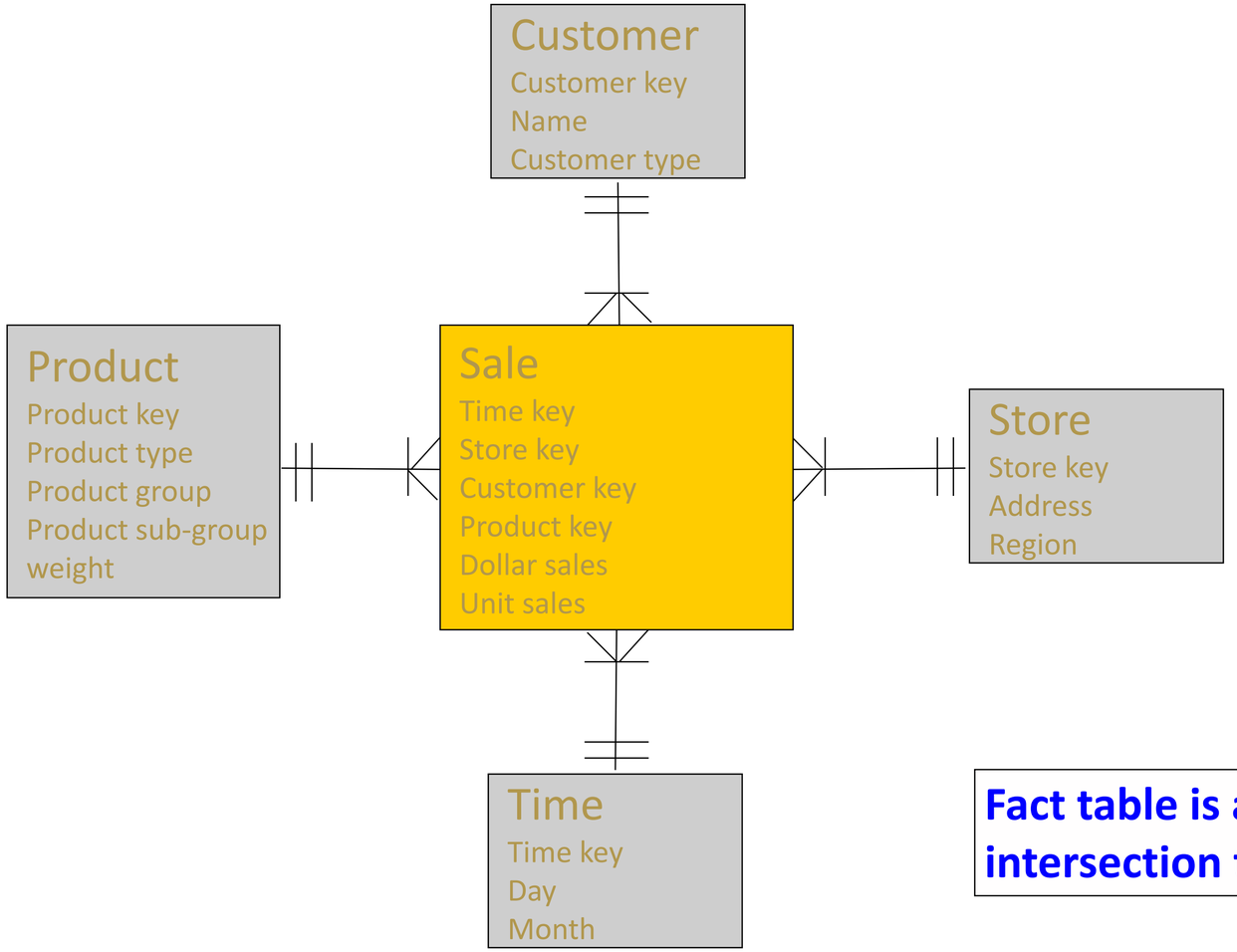
Product group e.g. Hardware
- Product type e.g. Tool
  - Product e.g. Hammer

Dimension Table - example

- Actual data might look like this
- Hierarchy evident in data

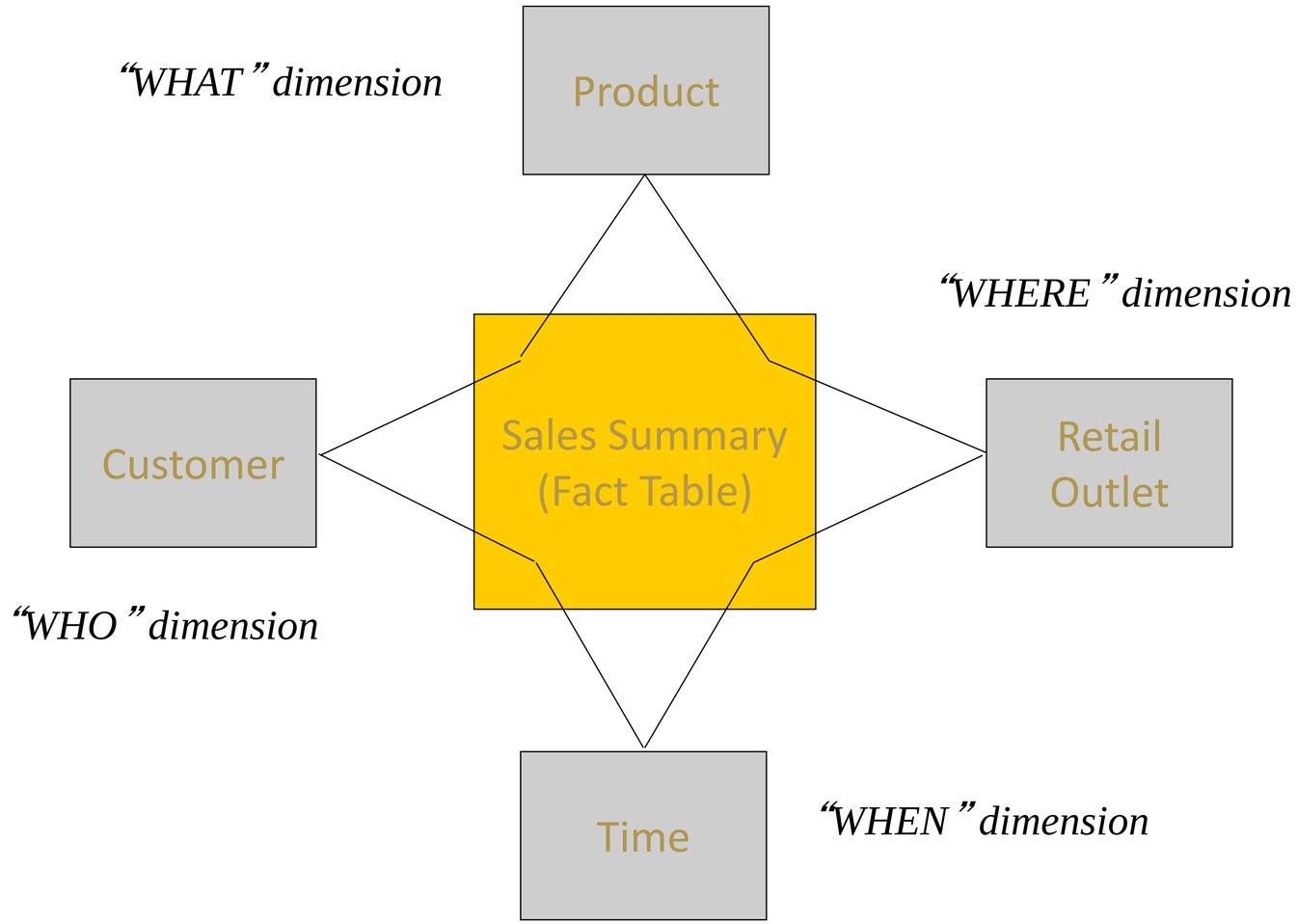
Prod-id	*Prod-Name*	*Prod-Group*	*Prod-Subgroup*	*Weight*
P10	Hammer	Hardware	Tool	5kg
P56	10cm Nails	Hardware	Nails	1kg
P98	Plastic Pipe	Plumbing	Pipe	1kg

Dimensional model as an ER model



Fact table is an intersection table

Star Schema


"WHAT" dimension
Product
"WHERE" dimension
Customer
Sales Summary
(Fact Table)
Retail
Outlet
"WHO" dimension
Time
"WHEN" dimension

Designing a Dimensional Model

- Choose a Business Process
- Choose the grain of the fact table
- Choose the dimensions
- Choose the measured facts (usually numeric, additive quantities)
- Complete the dimension tables

(Kimball, 1996)

Dimensional Modelling Task

- Design a dimensional model for LOANS

The Brisbane Movie Library purchases movies and loans them to its members for a charge in order to make a profit. The business is designing a data mart and decision support system.

Management wants to analyse the borrowing patterns of members in order to better identify the key members (most revenue per quarter). They can then focus on providing service to these members.

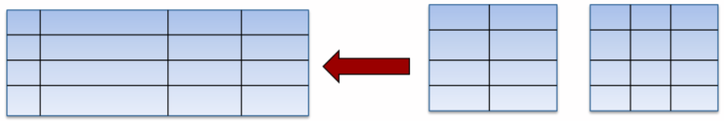
Management needs to analyse the value of their movies. They want to know which movies generate the most revenue per quarter. They don’t want to keep movies which are never (or rarely) borrowed. If a movie has not been borrowed for 3 months, it will be sold. Movies which generate a total rental return over 6 months which is less than their purchase price should also be sold to help keep inventory levels down.

Management wants to analyse the performance of each store to understand which are the most successful (in terms of profit = revenue – cost).

Design Outcomes: Normalised or Denormalised?

- Normalisation
  - Eliminates redundancy
  - Storage efficiency
  - Referential Integrity

- Denormalisation
  - Fewer tables (fewer joins)
  - Fast querying
  - Design is tuned for end-user analysis (tools & cognition)



Let’s Summarise!!

- Transactional databases suitable for running transactions
  - Store data in normalized structure
- Informational databases suitable for decision-making
  - It is not highly normalized

What is Examinable:

- Differenced between informational and transactional databases/questions
- DW Features
- Developing dimensional models

Next Seminar

Next Seminar

- More Dimensional Modelling