Dimensional Modeling

QUSER Meeting
April, 2015
Session 2

Designing a Data Warehouse

- **Requires a different set of thinking**
  - In comparison to OLTP design

- **Just like quantum physics is very different to conventional physics, it takes a while to get your head around the conceptual differences**

OLTP Design Principles

- **The primary goals for OLTP database design:**
  - To support the processing of a large number of small, atomic level transactions
  - Ensure transactions are created and processed in a consistent way
  - Ensure all transactions are accounted for
- **This is a microscopic view that is applied primarily at the transaction level**

OLTP Design Principles

- **The ‘rest’ of the OLTP database is designed around the need to support the transactions**
- **We have developed a highly evolved set of practices that dictate this design based on a number of principles:**
  - Use 3rd Normal Form to eliminate data redundancy
  - Minimize the amount of data actually stored
    - Disk is (was) expensive
    - Why use 10 bytes when one will do!

OLTP Design Principles

- **Transactions usually go through a lifecycle**
  - For example, the following steps may apply to sales order
    1. Initial data entry (or data capture from some form of input)
    2. Check inventory, adjust quantities. Create separate back order
    3. Price, based on customer's status/type etc.
    4. Credit check - possible credit hold followed by cancellation or release
    5. Release to warehouse for picking, with possible quantity adjustments
    6. Confirmation & invoicing
    7. Shipping - Update carrier, ship date and delivery date
    8. End-of day updates – costing, postings, date stamping

- **The database design needs to support all of these stages, actions and processes**

Designing a Data Warehouse

- The primary goal for data warehouse database design:

A place where people can easily access their data

- We don't need to process transactions!

Designing a Data Warehouse

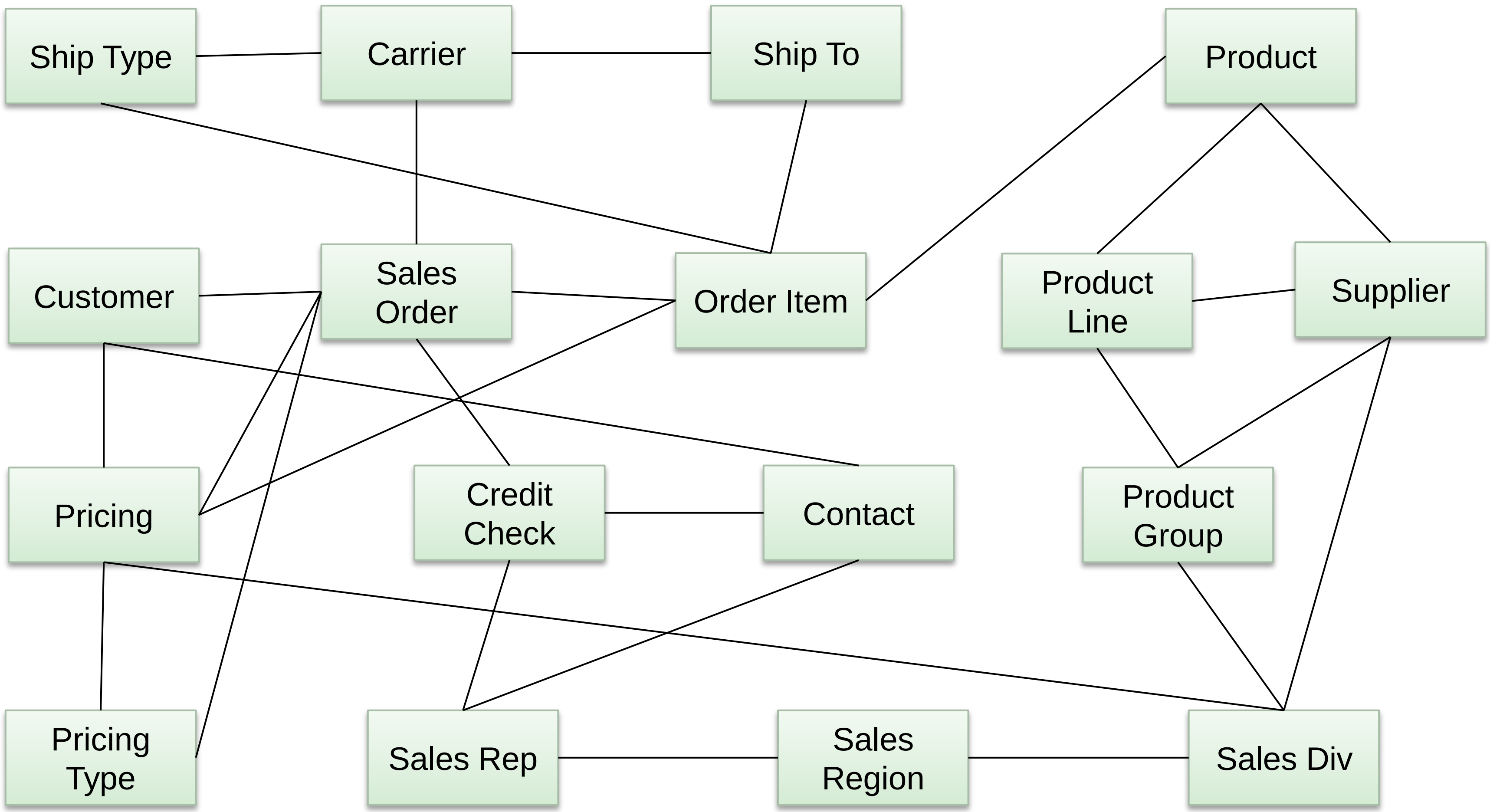
- **What does it need to support?**

- **If we ask our business users:**
    - You know what I want, just do it!
    - I want everything!
    - I need these 5 reports…
    - I want to be able to slice and dice…

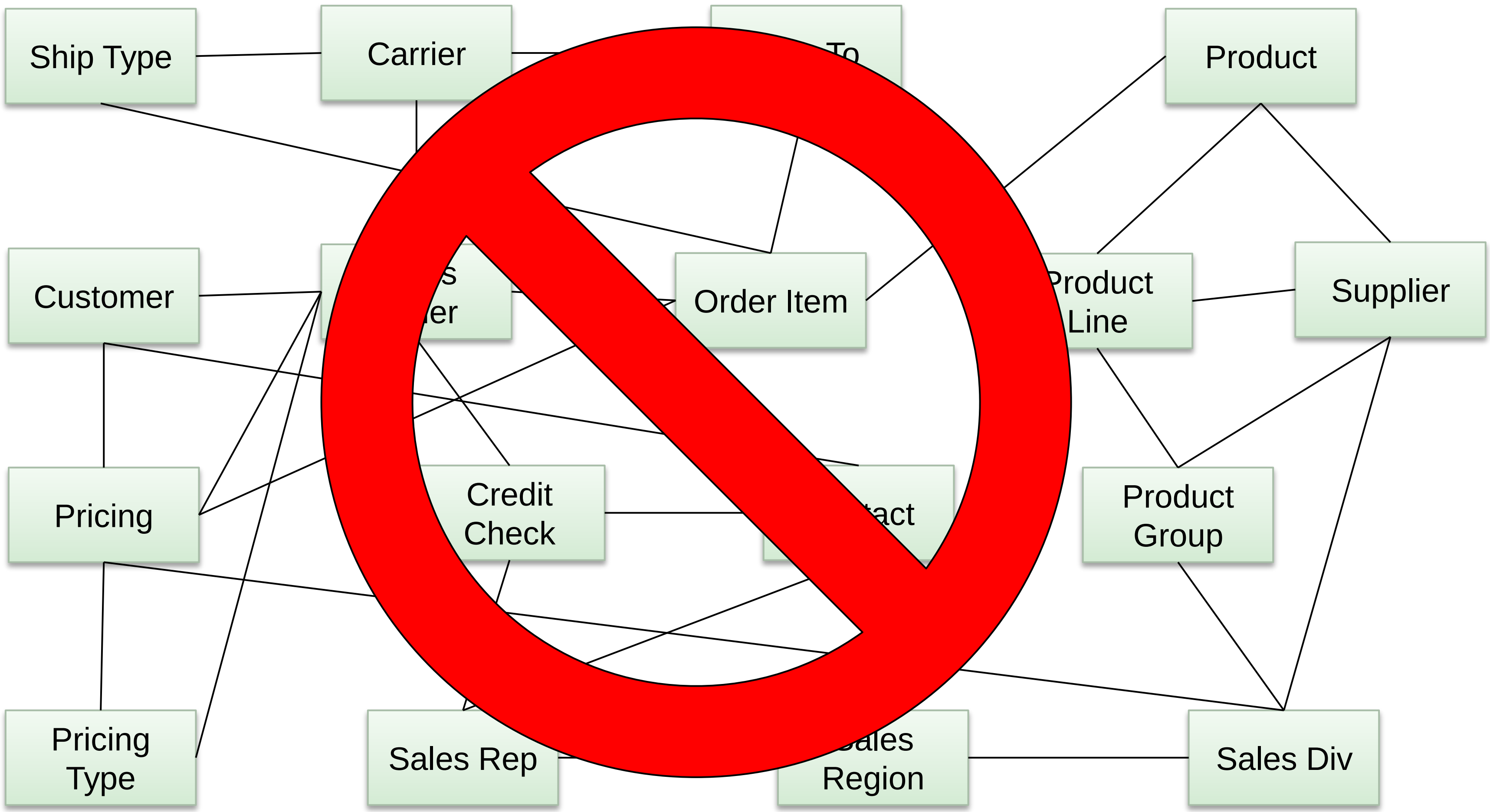
Designing a Data Warehouse

- **How do we start?**
  - Ideas anyone?
- **We have several physical databases, with different designs**
  - Do we pick one of these designs?
- **Maybe we start by looking at the data dependencies in our business**
  - Let’s create an E/R model!
  - This works well for OLTP design

Designing a Data Warehouse



Designing a Data Warehouse



Designing a Data Warehouse

- **This ‘bottom up’ approach is ideal for OLTP but not for Data Warehouses**
- **OLTP design has a very structured requirement**
  - Relationships, processes and dependencies are pre-defined and fairly rigid

Designing a Data Warehouse

- This ‘bottom up’ approach is ideal for OLTP but not for Data Warehouses
- OLTP design has a very structured requirement
    - Relationships, processes and dependencies are pre-defined and fairly rigid

Not always!

- ERP applications are flexible
- Hugely complicates the ER model

Designing a Data Warehouse

- **Let’s go back to our initial question:**
- **What do you need?**
  - You know what I want, just do it
  - I want everything!
  - I need these 5 reports…
  - I want to be able to slice and dice…
- **Maybe we asked the wrong question of the wrong people**

Designing a Data Warehouse

- **If we asked the CEO *what the company does?* He/she might say**

We sell our products in various markets

- **Then if we asked *how can we know how well we do it?* He/she might answer**

We measure our performance over time

Designing a Data Warehouse

We sell our products in various markets

We measure our performance over time

Duh! Obvious - right?

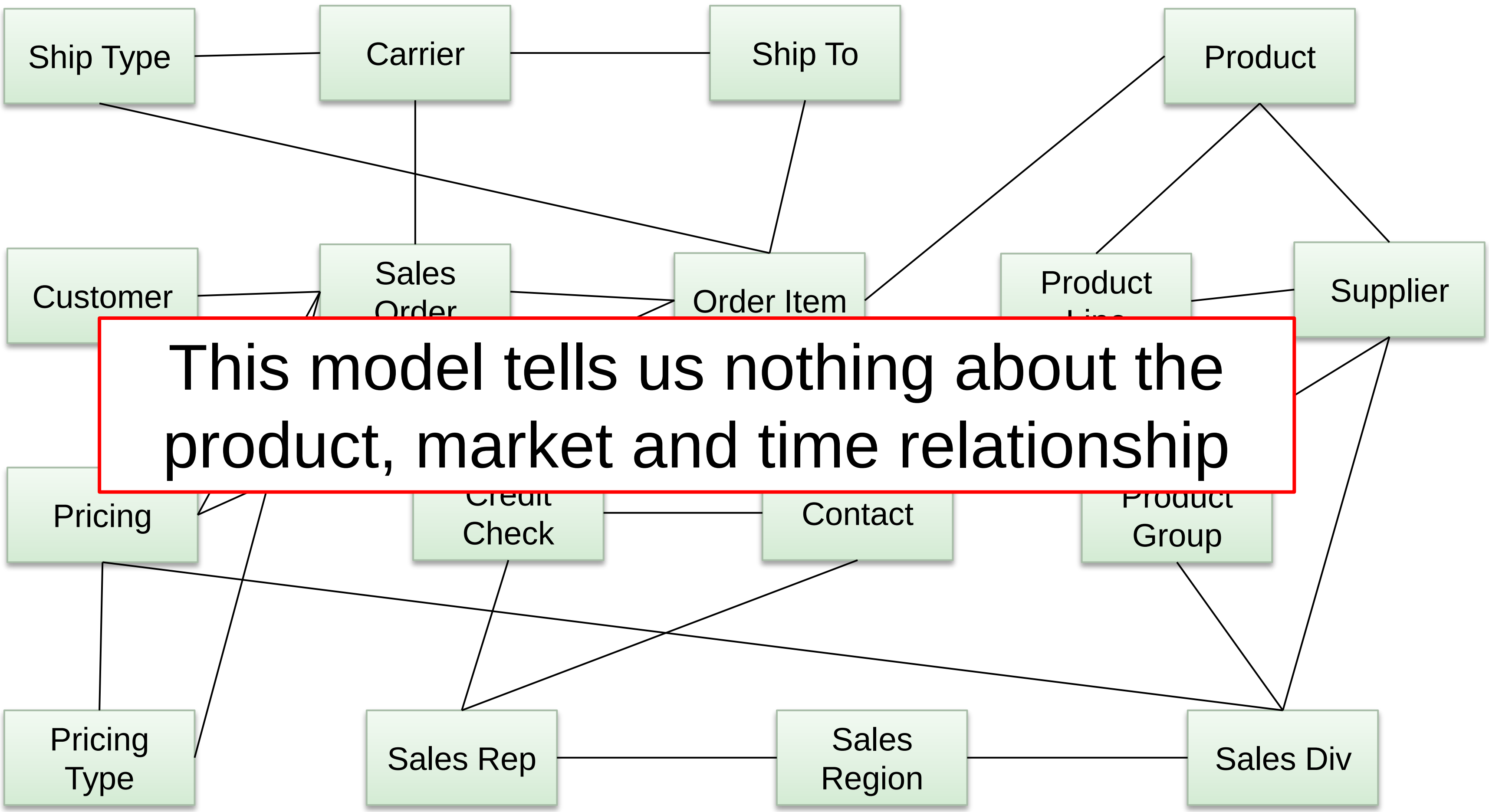
Designing a Data Warehouse

We sell our products in various markets

We measure our performance over time

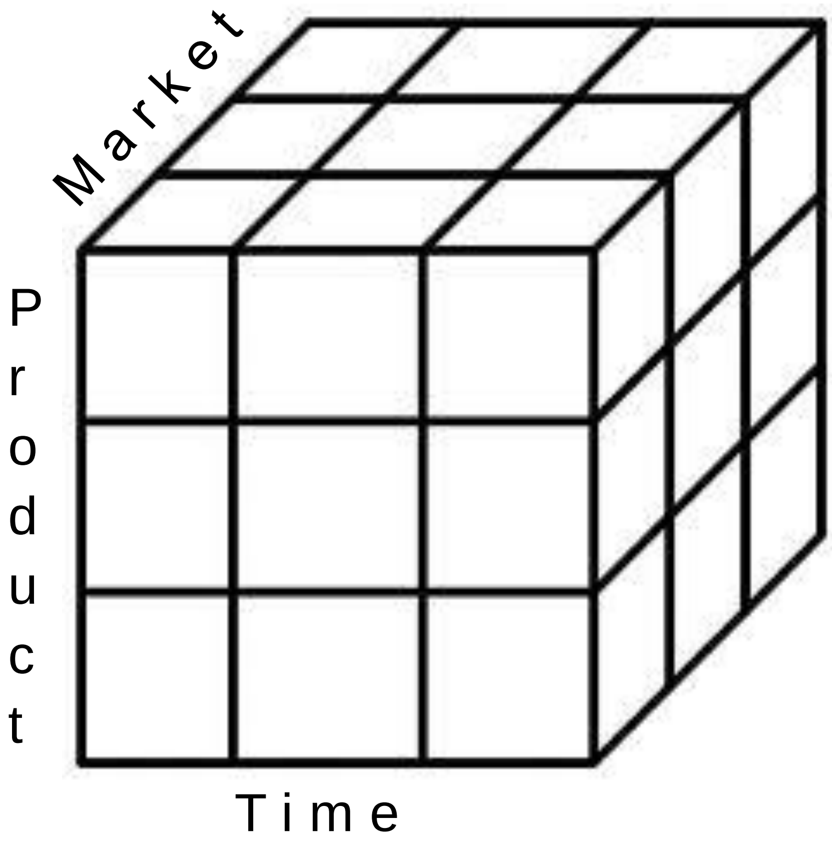
Three fundamentals that almost always apply to any business!

Relational Model



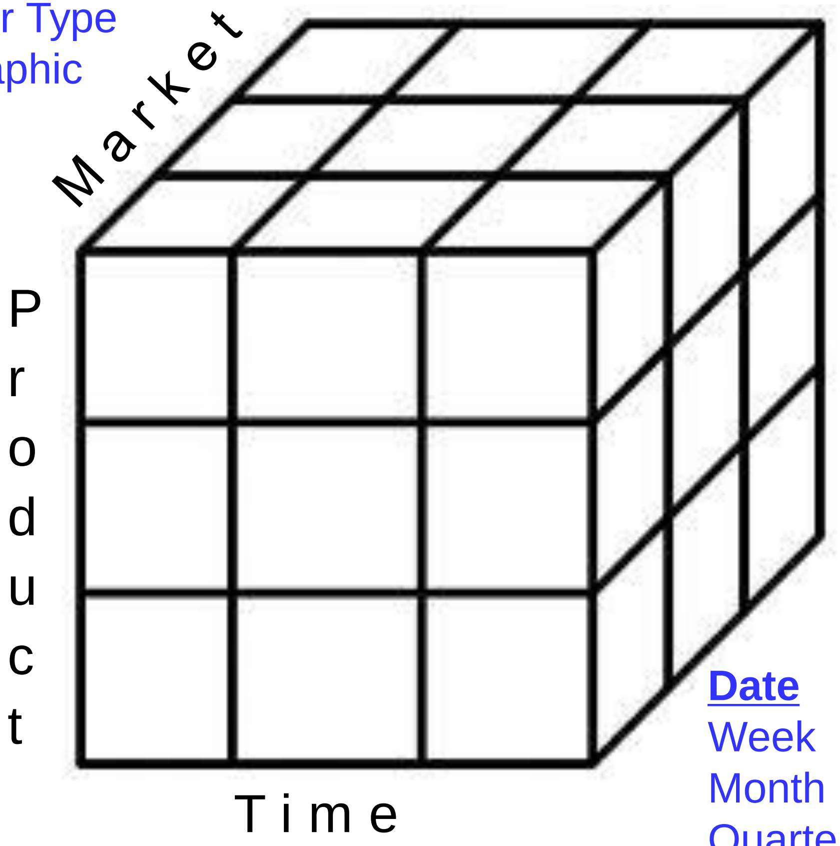
This model tells us nothing about the product, market and time relationship

Dimensional Model



Dimensional Model

Customer
Customer Type
Demographic
Location



(SKU)
Product Line
Brand
Supplier

Date
Week
Month
Quarter
Year

Each of these Major Dimensions have

attributes
or
sub-dimensions

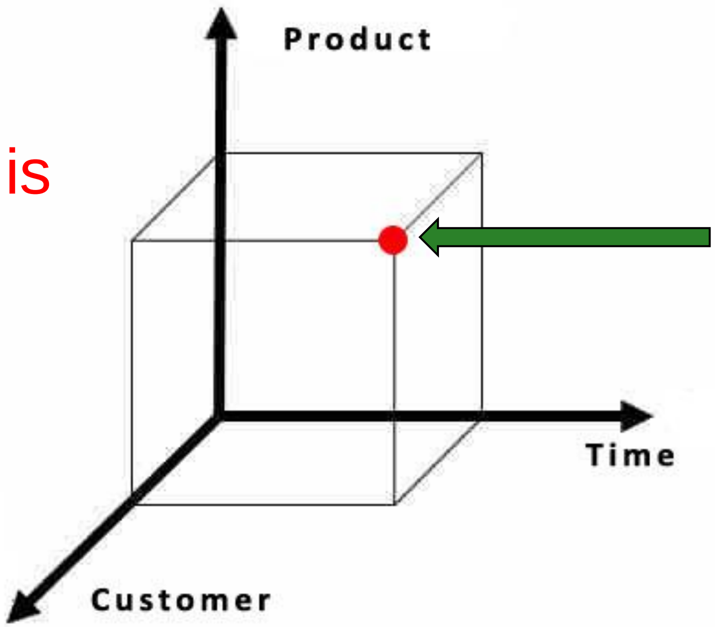
BI Reporting & Analytics Tools

- **OLAP Tools**
  - The structures built by OLAP tools are often referred to as 'cubes', suggesting 3 axes.

Flashback

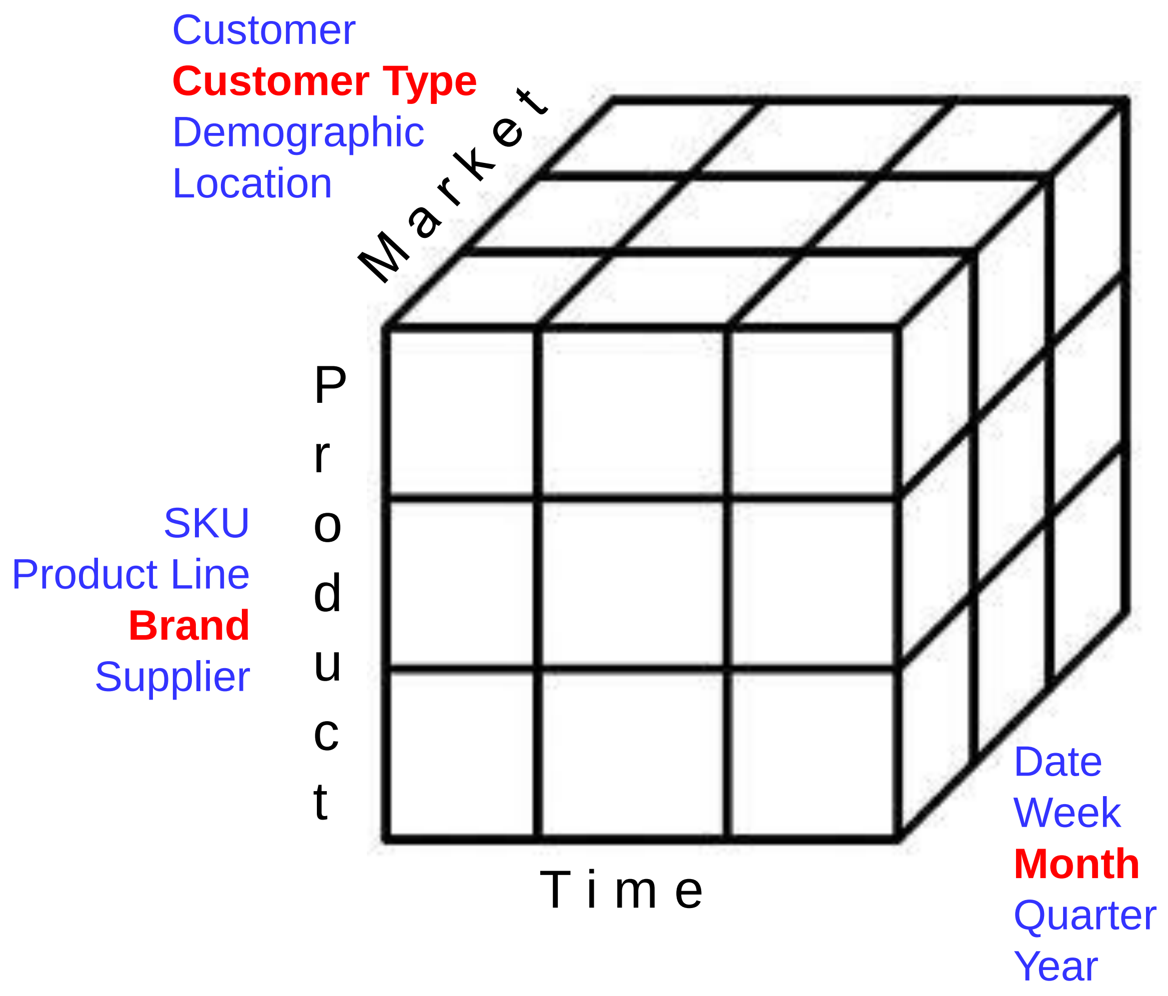
A 3-dimensional structure (cube) is easy for us to visualize.

Try visualizing a 12-dimensional structure!



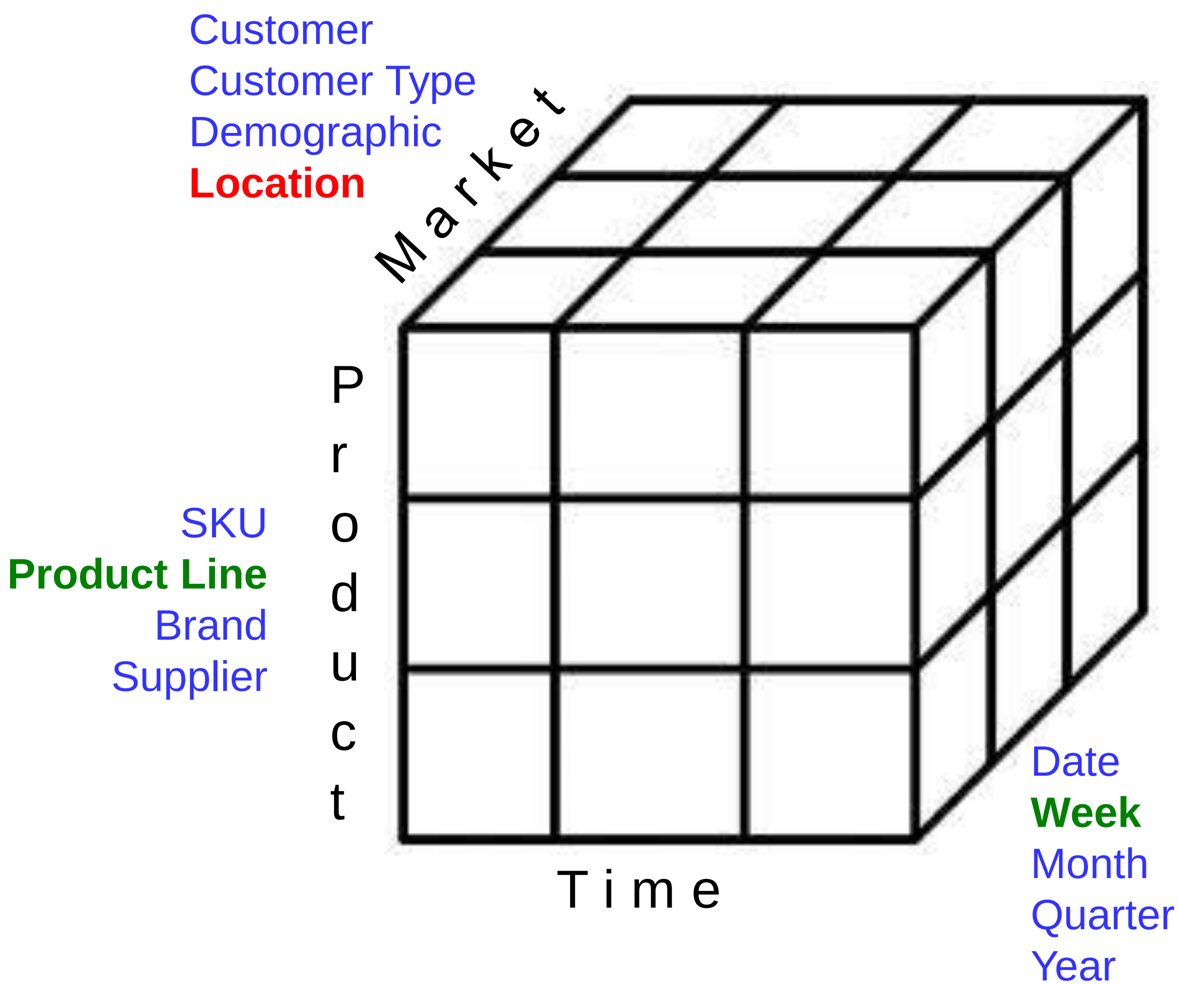
The intersection of the axes (dimensions) contains a data point (fact)

Dimensional Model



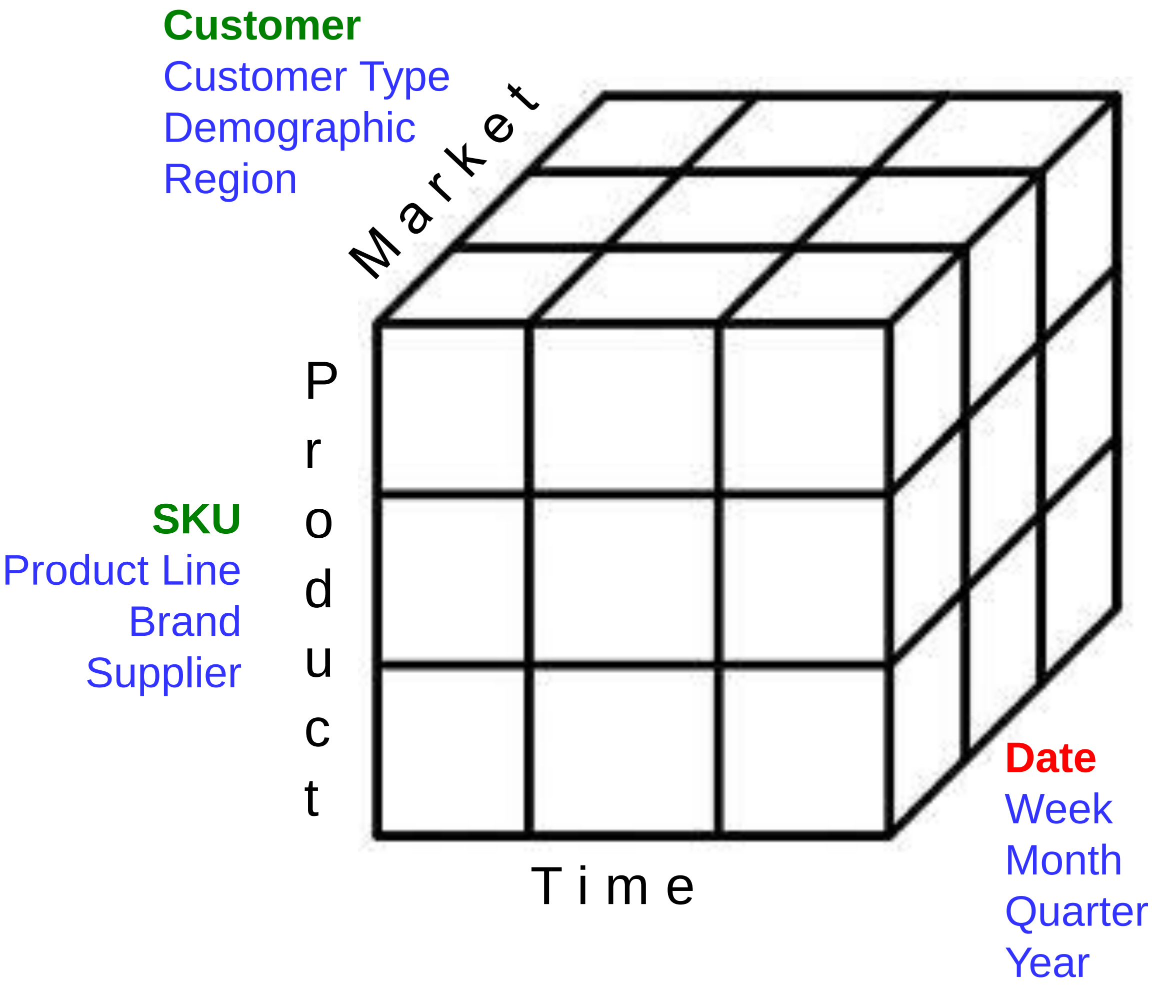
We can drill-down (slice & dice) by any combination of attributes

Dimensional Model



We can drill-down (slice & dice) by any combination of attributes

Dimensional Model



Unlike an OLAP cube, each data point is not pre-calculated.

If there are no 'facts' for a combination of dimensions, there is no data stored!

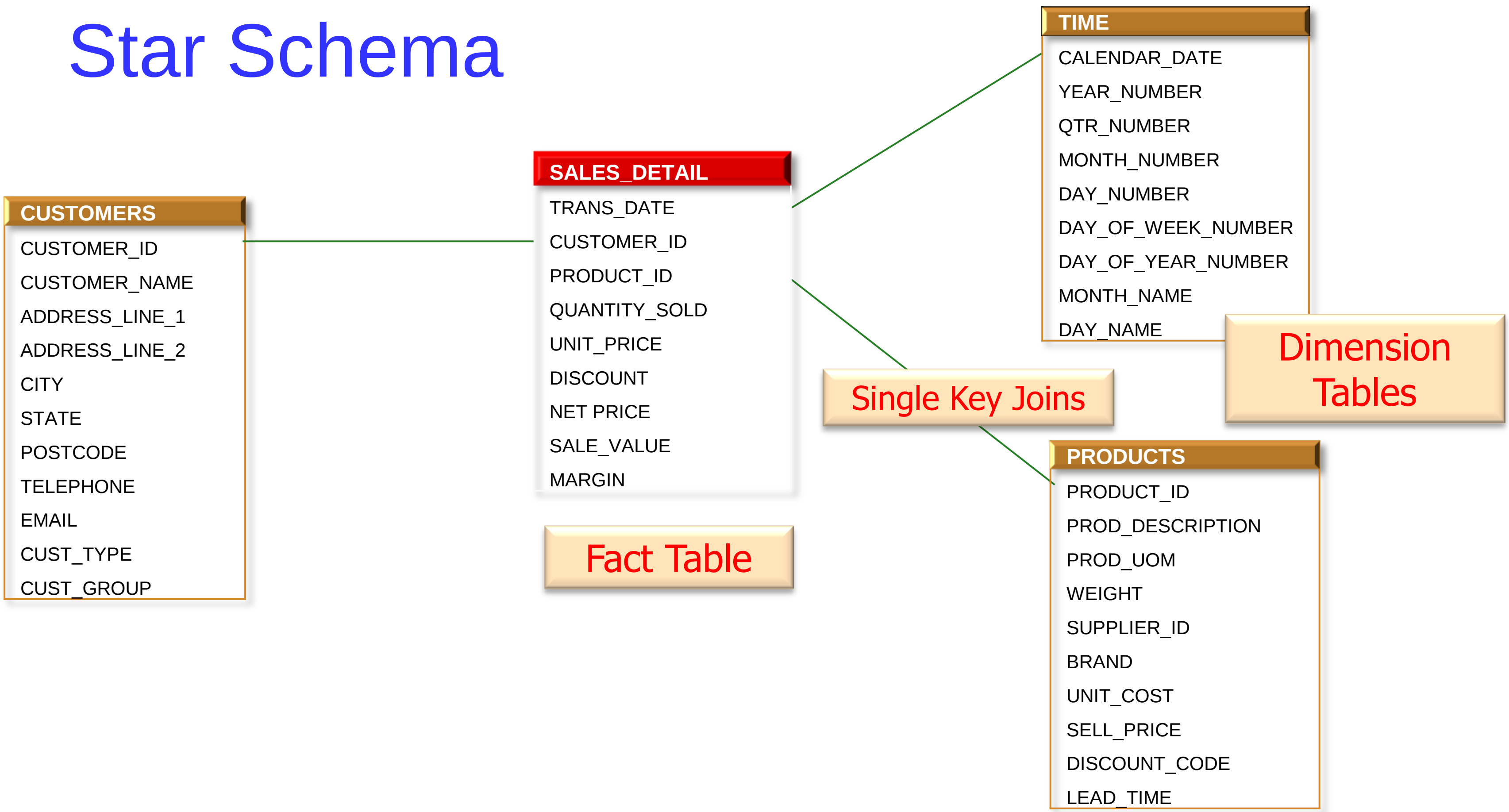
Dimensional Modeling

OK, we have the **logical** model

What does the **physical** model look like?

Dimensional Modeling

Star Schema



Dimensional Modeling

- **Fact Table**
  - Is a Highlander
  - Is the central table in the star schema design
  - Include all of the base Facts of a Transaction

SALES_DETAIL
TRANS_DATE
CUSTOMER_ID
PRODUCT_ID
QUANTITY_SOLD
UNIT_PRICE
DISCOUNT
NET PRICE
SALE_VALUE
MARGIN

Dimensional Modeling

- **Dimension Tables**
  - Have a single Unique ID (Primary Key)
  - Include all of the useful attributes of the dimension
  - 1 to n Dimension Tables can exist in a star schema

PRODUCTS
PRODUCT_ID
PROD_DESCRIPTION
PROD_UOM
WEIGHT
SUPPLIER_ID
BRAND
UNIT_COST
SELL_PRICE
DISCOUNT_CODE
LEAD_TIME

CUSTOMERS
CUSTOMER_ID
CUSTOMER_NAME
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
EMAIL
CUST_TYPE
CUST_GROUP

TIME
CALENDAR_DATE
YEAR_NUMBER
QTR_NUMBER
MONTH_NUMBER
DAY_NUMBER
DAY_OF_WEEK_NUMBER
DAY_OF_YEAR_NUMBER
MONTH_NAME
DAY_NAME

Dimensional Modeling

- **Dimension Tables**
  - Have a single Unique ID (Primary Key)
  - Include all of the useful attributes of the dimension
  - 1 to n Dimension Tables can exist in a star schema

PRODUCTS
PRODUCT_ID
PROD_DESCRIPTION
PROD_UOM
WEIGHT
SUPPLIER_ID
BRAND
UNIT_COST
SELL_PRICE
DISCOUNT_CODE
LEAD_TIME

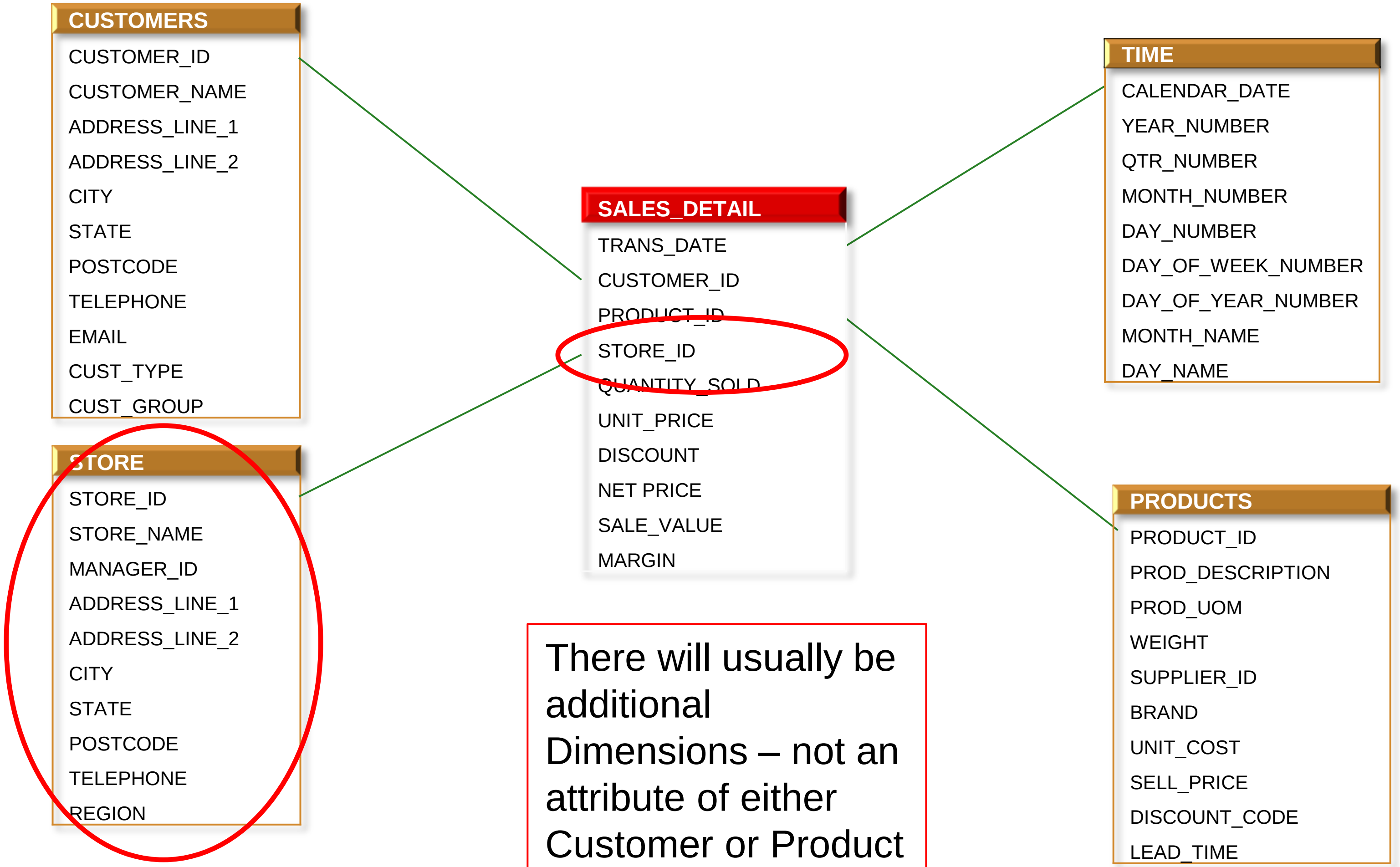
CUSTOMERS
CUSTOMER_ID
CUSTOMER_NAME
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
EMAIL
CUST_TYPE
CUST_GROUP

TIME
CALENDAR_DATE
YEAR_NUMBER
QTR_NUMBER
MONTH_NUMBER
DAY_NUMBER
DAY_OF_WEEK_NUMBER
DAY_OF_YEAR_NUMBER
MONTH_NAME
DAY_NAME

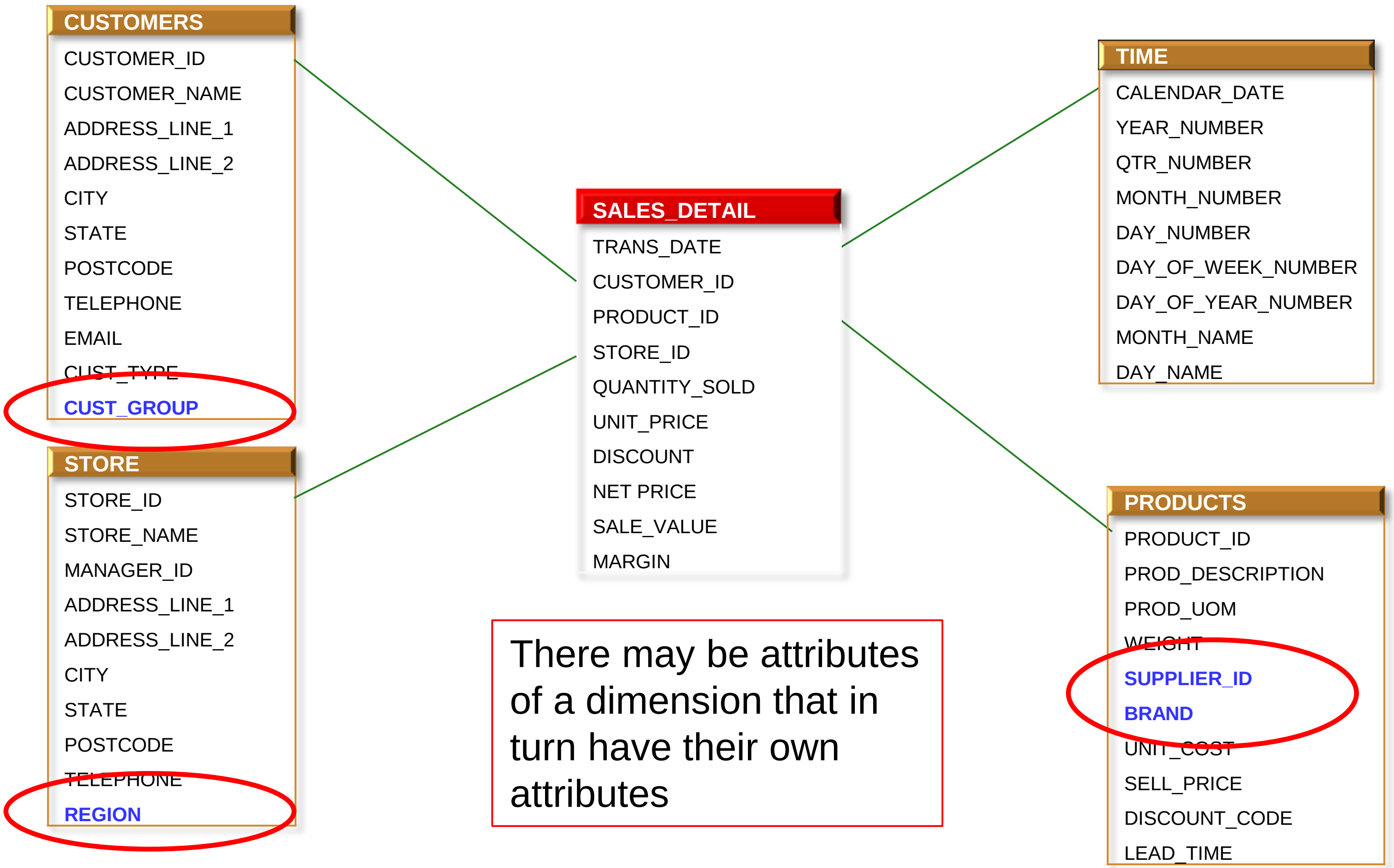
and what about

- Store
- Salesperson
- Sale Type
- Referrer
- ??

Dimensional Modeling



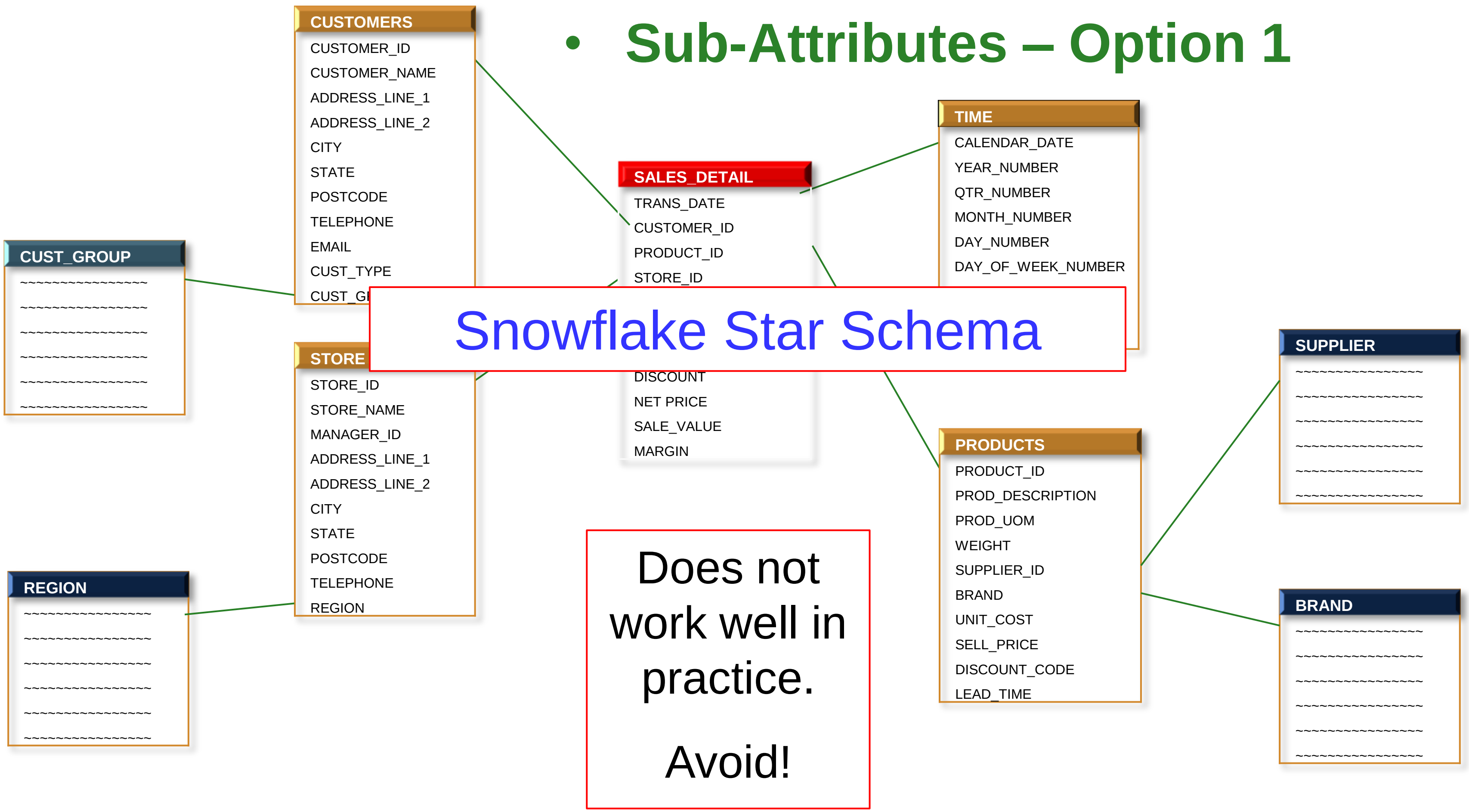
Dimensional Modeling



There may be attributes of a dimension that in turn have their own attributes

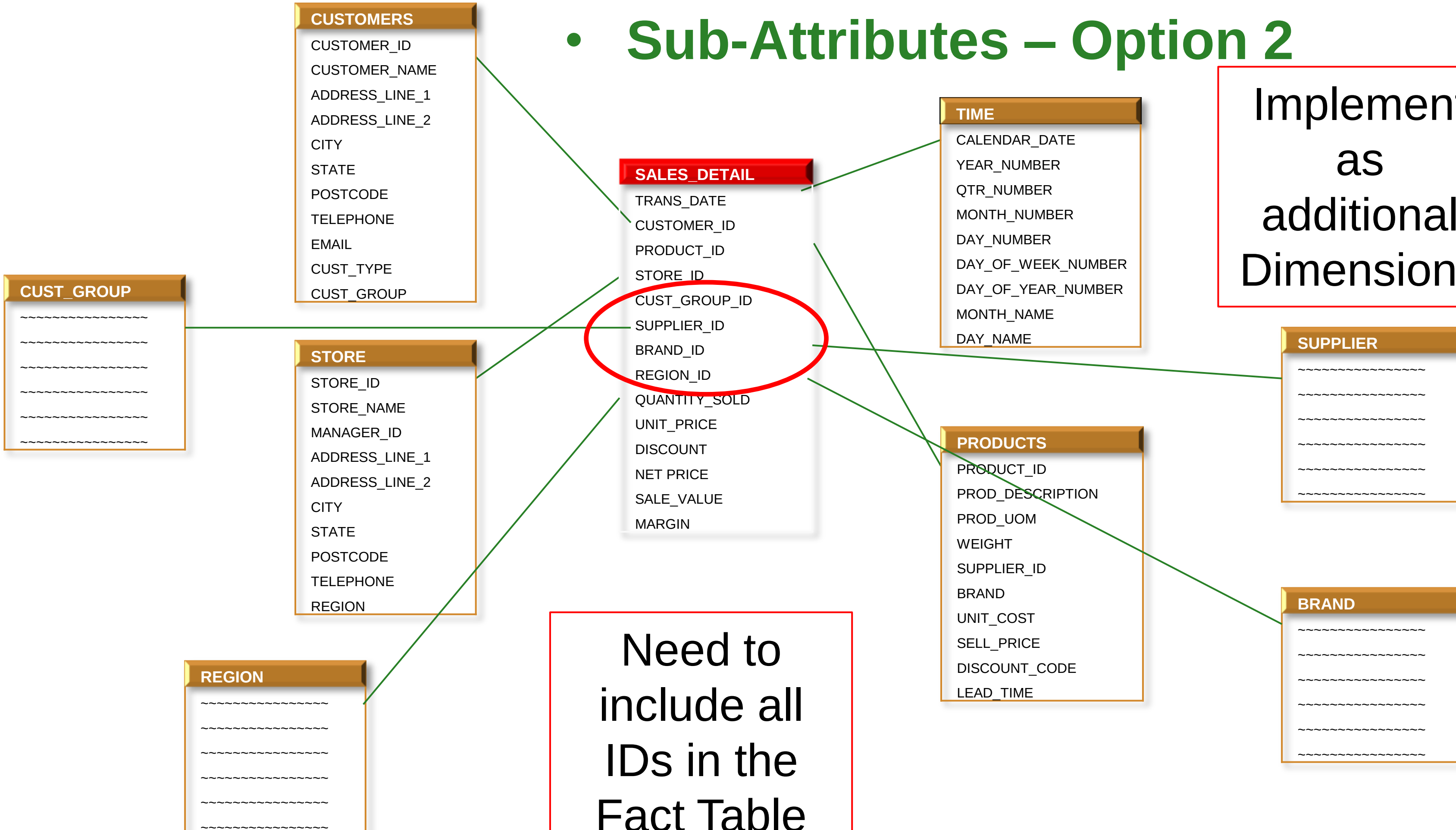
Dimensional Modeling

- ## Sub-Attributes – Option 1



Dimensional Modeling

- **Sub-Attributes – Option 2**



Implement as additional Dimensions

Need to include all IDs in the Fact Table

Dimensional Modeling

- **Sub-Attributes – Option 3**
  - Include the sub-attributes in the same Dimension table
  - i.e. denormalize (horror of horrors!)
  - This is a good approach when there are just a few sub-attributes
    - description, type for example

PRODUCTS
PRODUCT_ID
PROD_DESCRIPTION
PROD_UOM
WEIGHT
SUPPLIER_ID
SUPPLIER_NAME
SUPPLIER_TYPE
BRAND_ID
BRAND_DESCRIPTION
UNIT_COST
SELL_PRICE
DISCOUNT_CODE
LEAD_TIME

Dimensional Modeling

- **Fact or Attribute?**
  - Sometimes it may be difficult to determine whether a numeric value is a Fact or an Attribute.
  - Often we can make that decision by asking "*can this value vary by transaction*". If so it is a Fact.
    - SALE_QUANTITY is obviously a Fact.
    - ITEM_WEIGHT is most likely an Attribute (of Product)
    - COST may not be so obvious. It may remain constant for many weeks, but then will change. It's probably best treated as a Fact, rather than an Attribute of Product.

Dimensional Modeling

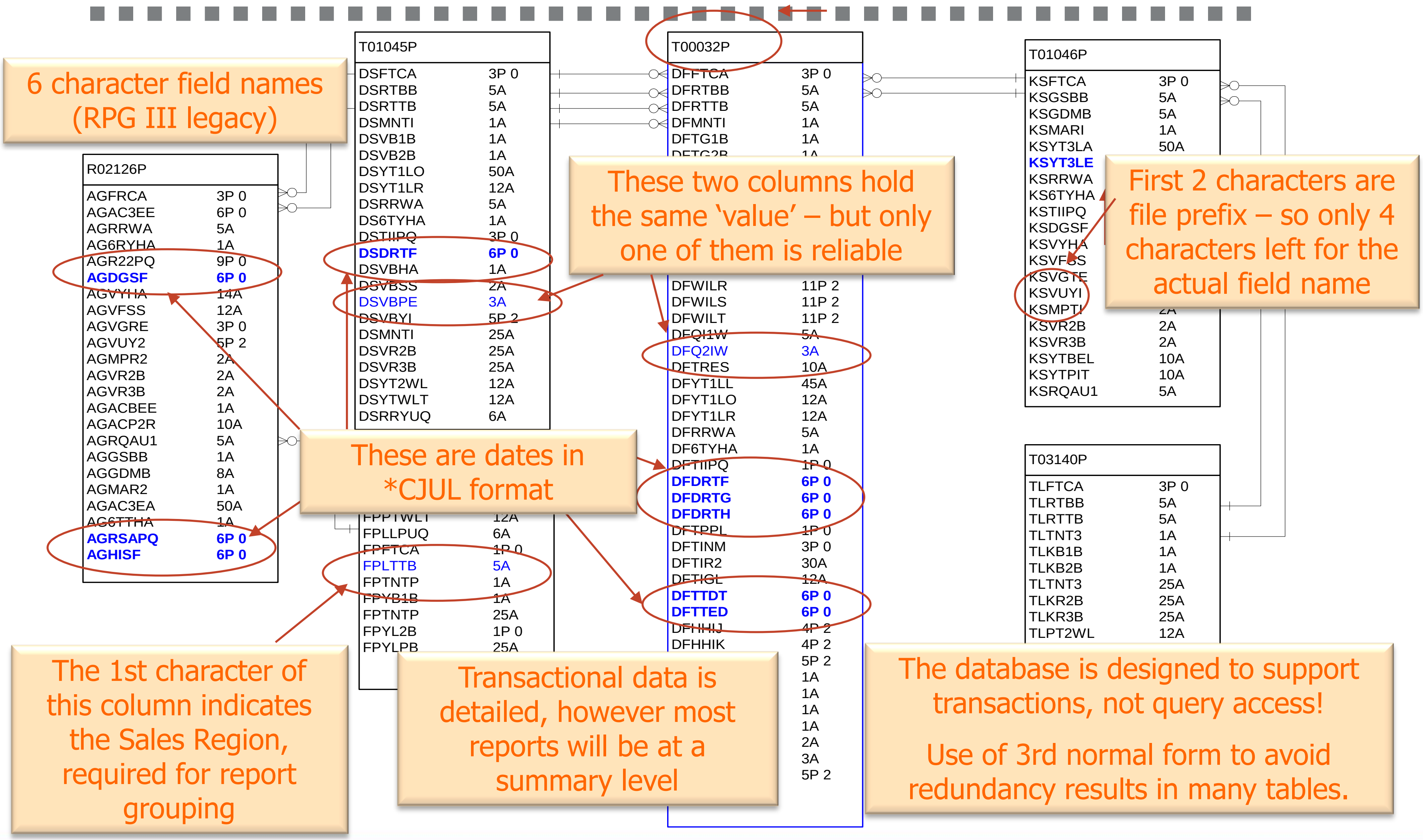
- **The TIME Dimension**
  - Primary key is the smallest unit of time we measure
  - Most common is DATE
  - Even if our reporting is (currently) always at (e.g.) MONTH level or higher, we still use DATE

TIME
CALENDAR_DATE
YEAR_NUMBER
QTR_NUMBER
MONTH_NUMBER
DAY_NUMBER
DAY_OF_WEEK_NUMBER
DAY_OF_YEAR_NUMBER
MONTH_NAME
DAY_NAME

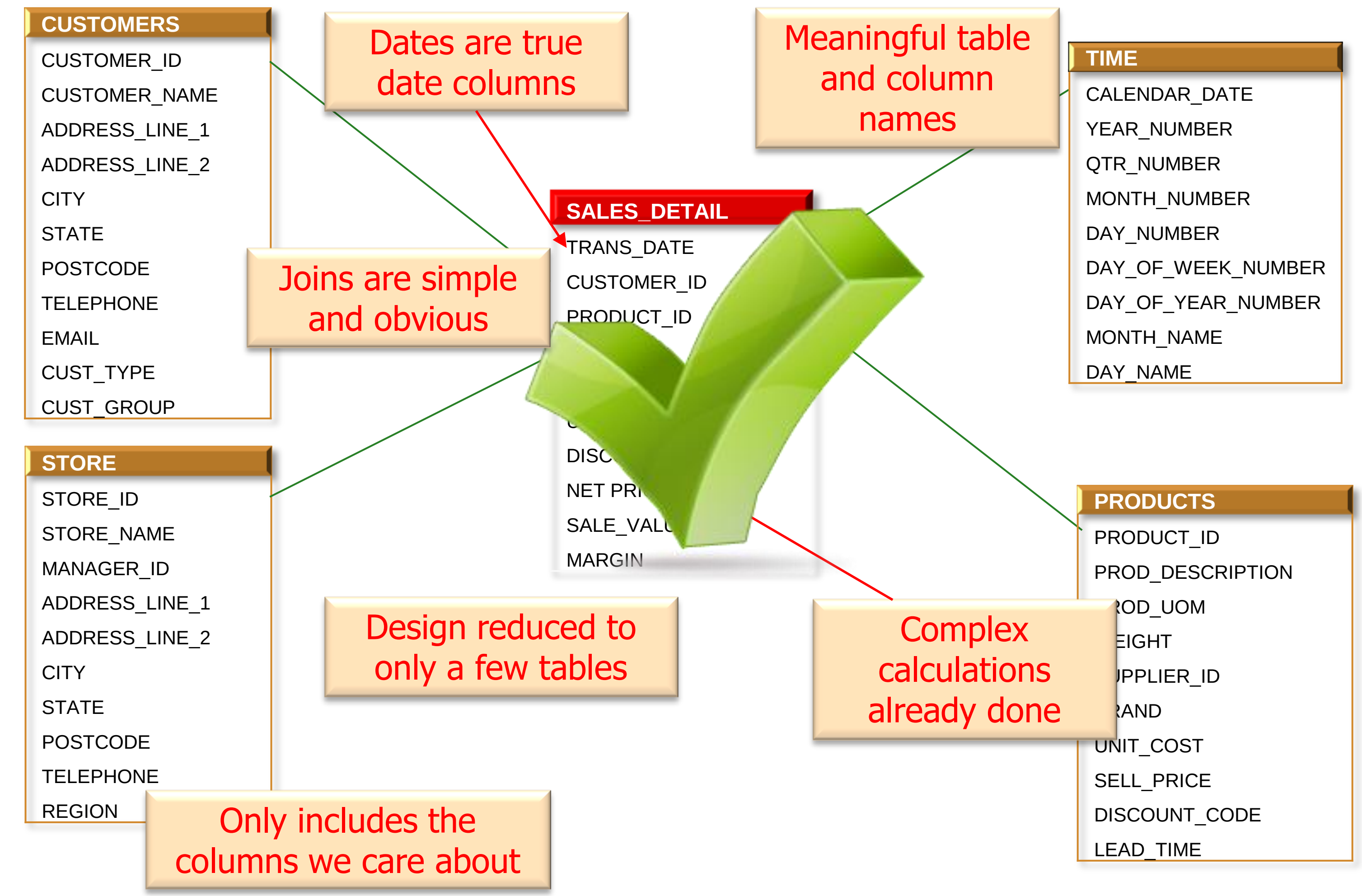
Dimensional Modeling

- **Let’s review some common issues**
  - Are they solved by Dimensional Modeling?

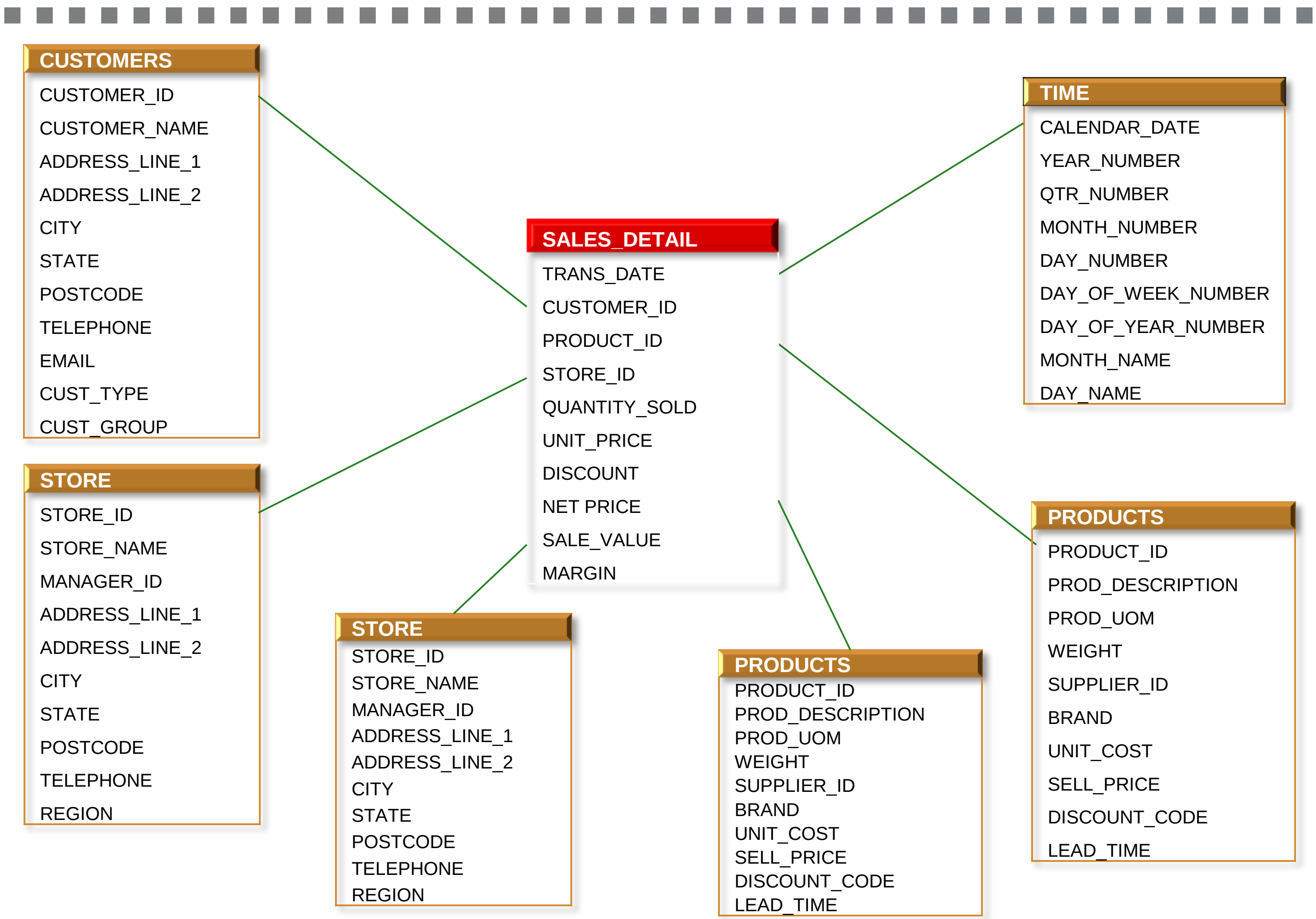
Complex Operational Data



Star Schema


CUSTOMERS
CUSTOMER_ID
CUSTOMER_NAME
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
EMAIL
CUST_TYPE
CUST_GROUP
Dates are true date columns
Meaningful table and column names
TIME
CALENDAR_DATE
YEAR_NUMBER
QTR_NUMBER
MONTH_NUMBER
DAY_NUMBER
DAY_OF_WEEK_NUMBER
DAY_OF_YEAR_NUMBER
MONTH_NAME
DAY_NAME
SALES_DETAIL
TRANS_DATE
CUSTOMER_ID
PRODUCT_ID
MARGIN
Joins are simple and obvious
STORE
STORE_ID
STORE_NAME
MANAGER_ID
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
REGION
PRODUCTS
PRODUCT_ID
PROD_DESCRIPTION
UNIT_COST
SELL_PRICE
DISCOUNT_CODE
LEAD_TIME
Design reduced to only a few tables
Complex calculations already done
Only includes the columns we care about

Star Schema


CUSTOMERS
CUSTOMER_ID
CUSTOMER_NAME
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
EMAIL
CUST_TYPE
CUST_GROUP
STORE
STORE_ID
STORE_NAME
MANAGER_ID
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
REGION
SALES_DETAIL
TRANS_DATE
CUSTOMER_ID
PRODUCT_ID
STORE_ID
QUANTITY_SOLD
UNIT_PRICE
DISCOUNT
NET PRICE
SALE_VALUE
MARGIN
STORE
STORE_ID
STORE_NAME
MANAGER_ID
ADDRESS_LINE_1
ADDRESS_LINE_2
CITY
STATE
POSTCODE
TELEPHONE
REGION
PRODUCTS
PRODUCT_ID
PROD_DESCRIPTION
PROD_UOM
WEIGHT
SUPPLIER_ID
BRAND
UNIT_COST
SELL_PRICE
DISCOUNT_CODE
LEAD_TIME
TIME
CALENDAR_DATE
YEAR_NUMBER
QTR_NUMBER
MONTH_NUMBER
DAY_NUMBER
DAY_OF_WEEK_NUMBER
DAY_OF_YEAR_NUMBER
MONTH_NAME
DAY_NAME
PRODUCTS
PRODUCT_ID
PROD_DESCRIPTION
PROD_UOM
WEIGHT
SUPPLIER_ID
BRAND
UNIT_COST
SELL_PRICE
DISCOUNT_CODE
LEAD_TIME

Issues with Operational Data

Data Quality Example

2005: Valparaiso, Indiana

Somehow a propert ses nt value for the home shown above was i ctly changed to $400M in the property tax dat

The expected prop tax re ue was included in the county budget - but the $8M property tax bill on the

Must be addressed in the ETL

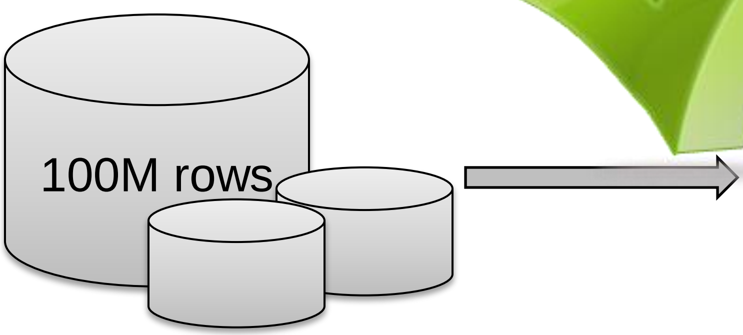
The school district was force to return $2.7M

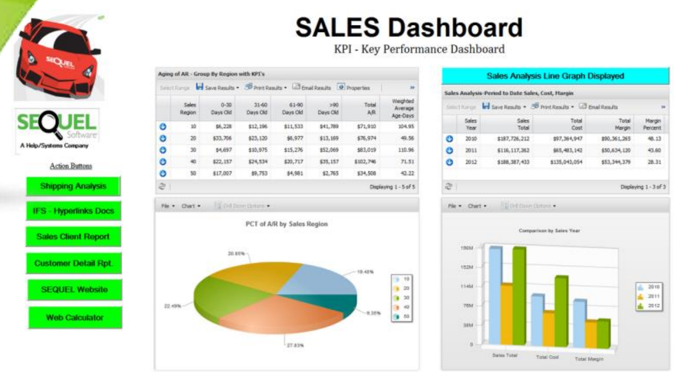
All extracurricular activities and sports were cancelled that year

Issues with Operational Data

Poor Performance

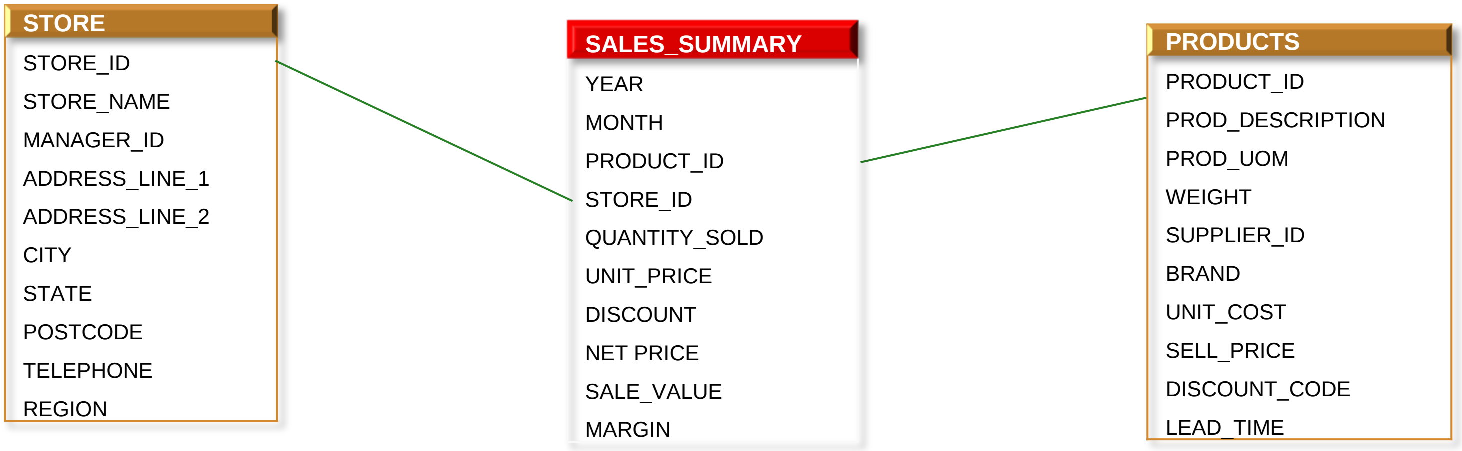
- Large transaction table
- Many related tables
- Most reports are at a summary le
- Reports and queries are long r d consume significant system resources





Star Schema

- **Example: Monthly Product Sales by Store**
  - Summary Fact table aggregated to YEAR/MONTH/PRODUCT/STORE
    - Customer and Time dimension dropped
  - Also supports queries by Brand, Region etc.



  - Dimension tables can be included in multiple star schemas

Issues with Operational Data

Challenge

Multiple instances of same table, with duplicate key values

Customer File - US	
CUSTNO	CUSTNAME
1001	John Smith
1002	Mary Jones
1003	Chris Anderson
1004	David Perry

Customer File - Canada	
[illegible]	CUSTNAME
[illegible]	Harry Potter
[illegible]	Jeremy Carr
[illegible]	Penny Hayes
1004	Debbie Thornton

or different versions of sam[illegible]y

- Incompatible data types
- Duplicates

Customer File - US	
CUSTNO	CUSTNAME
1001	**John Smith**
1002	Mary Jones
1003	Chris Anderson
1004	David Perry

Customer File - Canada	
CUSTID	CUSTNAM
AA234	Julie Johnson
AA235	Fred Hunter
AB670	**John Smith**
BD309	Alan Jordan

Surrogate Key Concepts

A surrogate key is a key value (i.e., a dimension) that is used as a substitute key in place of the natural key(s) of the data.

- It does not exist in the source systems (otherwise it would be a normal key/dimension value)
- It is generated when loading the data warehouse tables.

Surrogate keys are ALWAYS numeric values

- Value is assigned to new entities (ie rows in a table) on a sequential basis
- Each new row is assigned the next available number
- No “intelligence” is used to decide the value

A table with a Surrogate key will ALWAYS have a secondary index based on the original key of the data.

- This is used as the ‘cross-reference’ between the old key(s) and the new surrogate key.

Issues with Operational Data

Surrogate Keys are generated and assigned in the ETL process

Customer File - US	
CUSTNO	CUSTNAME
1001	John Smith
1002	Mary Jones
1003	Chris Anderson
1004	David Perry

Customer File - Canada	
CUSTNO	CUSTNAME
1001	Harry Potter
1002	Jeremy Carr
1003	Penny Hayes
1004	Debbie Thornton

	CUSTOMER DIMENSION		
CUST_ID	CUSTNO	COUNTRY	CUSTNAME
1	1001	USA	John Smith
2	1002	USA	Mary Jones
2	1003	USA	Chris Anderson
4	1004	USA	David Perry
5	1001	CAN	Harry Potter
6	1002	CAN	Jeremy Carr

Secondary Index

Dimensional Modeling

Issues with Operational Data

Challenge

Changing attributes

2011	100	Acme Flooring	Small Retailer	Jenny Brown
2013	100	Acme Flooring	…etailer	Jenny Brown
2014	100	Acme Floorin…	…or Retailer	Rob McAdam

2011 Report

2011 Sales by Sales Rep/Customer Group

Acme Flooring	250,000
Regal Rugs	150,000
Total Small Retailer	**400,000**
Carpet Warehouse	2,500,000
Hardwood Hank	2,100,000
Total Major Retailer	**4,600,000**
Total Jenny Brown	**5,000,000**

Same report, re-run in 2014

2011 Sales by Sales Rep/Customer Group

Regal Rugs	150,000
Total Small Retailer	**150,000**
Carpet Warehouse	2,500,000
Hardwood Hank	2,100,000
Total Major Retailer	**4,600,000**
Total Jenny Brown	**4,750,000**

Slowly Changing Dimensions

The Slowly Changing Dimension (SCD) concept provides for tracking of historical changes to important attributes of an entity, such as a Customer.

Slowly Changing Dimensions

There are three types of Slowly Changing Dimensions:

1. A change to an attribute, which is defined as **Type 1**, requires that the value is simply updated with the new value in the existing record. This is the default behavior where updates occur to an entity. Attributes that are not used for grouping or reporting (e.g. Telephone Number) can safely be implemented as Type 1.

2. A change to an attribute, which is defined as **Type 2**, requires that a new record for the entity be created, using a new surrogate key value. The old record retains the previous values, and the new record includes the current values.

3. **Type 3** dimensions are less commonly used. In this case, the record includes a 'previous value' column for the attribute. When the attribute changes, the old attribute value is moved to the 'previous' column, and the new value takes its place. Therefore only the current and the most recent previous value are available (in the same record).

Slowly Changing Dimensions

There are three types of Slowly Changing Dimensions:

1. A change to an attribute, which is defined as **Type 1**, requires that the value is simply updated with the new value in the existing record. This is the default behavior where updates occur to an entity. Attributes that are not used for grouping or reporting (e.g. Telephone Number) can safely be implemented as Type 1.

2. A change to an attribute, which is defined as **Type 2**, requires that a new record for the entity be created, using a new surrogate key value. The old record retains the previous values, and the new record includes the current values.

3. **Type 3** dimensions are less commonly used. In this case, the record includes a 'previous value' column for the attribute. When the attribute changes, the old attribute value is moved to the 'previous' column, and the new value takes its place. Therefore only the current and the most recent previous value are available (in the same record).

Slowly Changing Dimensions

Design requirements necessary to enable Type II SCD:

1. The dimension table must use a surrogate key. This is because there will (eventually) be several rows for the same entity, and each must have its own unique identifier.

2. As with all surrogate key tables, there must be a secondary index that includes the natural key(s) of the entity. However, this secondary index must also include an **Effective To Date** column. This indicates the date range for which this instance of the entity is applicable.

3. The current row for the entity must be identifiable. This will have a future (unknown) Effective to Date.

Slowly Changing Dimensions

This following example shows how a Customers Table might look, when tracking Sales Region as a Slowly Changing Dimension.

CUSTOMER_ID	OLD_CUSNO	CUST_NAME	SALES_REGION	EFF_TO_DATE
11003	406060	Acme Tools	NORTH	2011-04-20
12104	523043	Johnson Controls	EAST	2012-05-17
16563	650555	Bilson Construction	SOUTH	9999-12-31
19369	406060	Acme Tools	WEST	9999-12-31
22531	532043	Johnson Controls	WEST	9999-12-31

Surrogate Key

Original Customer #

Defines the date range for which this row is active. The current row has an unknown future date (9999-12-31)

In this example, the Sales Region for Acme Tools changed on April 20, 2011.

Slowly Changing Dimensions

ETL - Loading the Dimension Table (e.g. CUSTOMER)

- Determine if new or existing customer
- If New Customer
  - Get next surrogate key
  - Set Effective-to-Date to 9999-12-31
  - Insert row
- If Existing Customer
  - Get current Dimension Table row (by original customer number and 9999-12-31 date) and compare all type II SCD
  - If changed, get next surrogate key and insert new row with current customer attributes and 9999-12-31 date
    - Update the 'old' current row, changing the effective-to-date to today
  - If unchanged, update the existing row (maybe telephone number has changed)

Slowly Changing Dimensions

ETL - Loading the Fact Table

- When loading Transactions into the Fact Table we need to associate the Transaction with the row for the entity that was **current at the time of the transaction.**
- A look-up is performed on the Dimension Table, using the original key and the transaction date, to retrieve the surrogate key for that row
  - Remember we have a secondary index over the table that uses these values
- If using RPG the lookup would look like this

```
C     CUSKEY        KLIST
C                   KFLD                    CUSNO
C                   KFLD                    TRANDATE
 * Position using full key of index 2
C     CUSKEY        SETLL     CUST_02
 * Read using Customer number only
C     CUSNO         READE     CUST_02
 * We now have the row effective as at TRANDATE
```

- Include the retrieved surrogate key in the Fact table row

Slowly Changing Dimensions

Fact Table

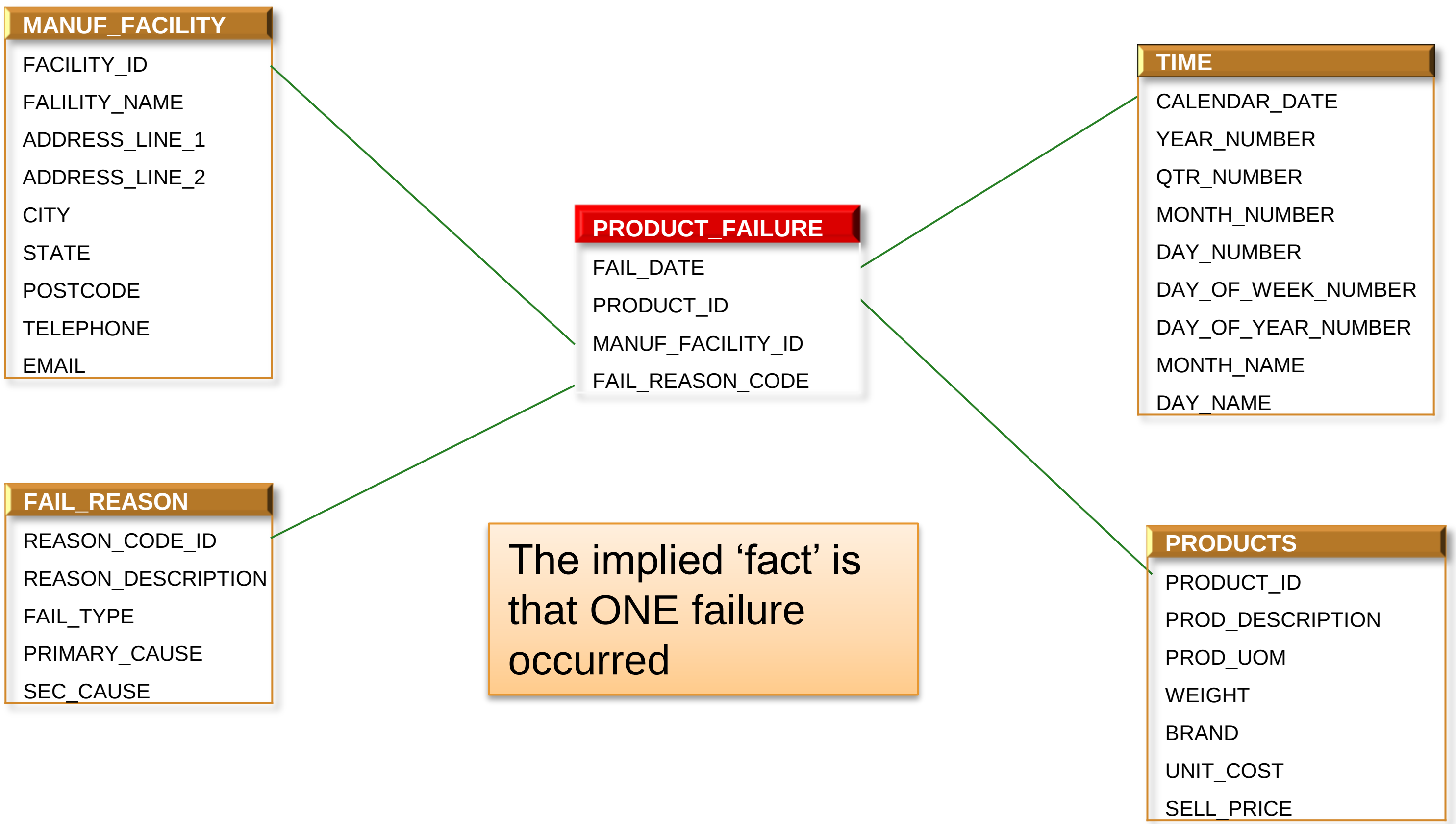
TRAN_DATE	CUSTOMER_ID	ITEM_ID	QUANTITY	PRICE
2010-02-19	11003	2039495	3	987.55
2011-03-29	11003	3404904	12	1,230.00
2012-01-17	19369	4093932	1	120.00
2013-09-02	19369	2049383	5	250.00

CUSTOMER_ID	OLD_CUSNO	CUST_NAME	SALES_REGION	EFF_TO_DATE
11003	406060	Acme Tools	NORTH	**2011-04-20**
12104	523043	Johnson Controls	EAST	2012-05-17
16563	650555	Bilson Construction	SOUTH	9999-12-31
19369	406060	Acme Tools	WEST	**9999-12-31**
22531	532043	Johnson Controls	WEST	9999-12-31

Dimension Table

Factless Fact Tables

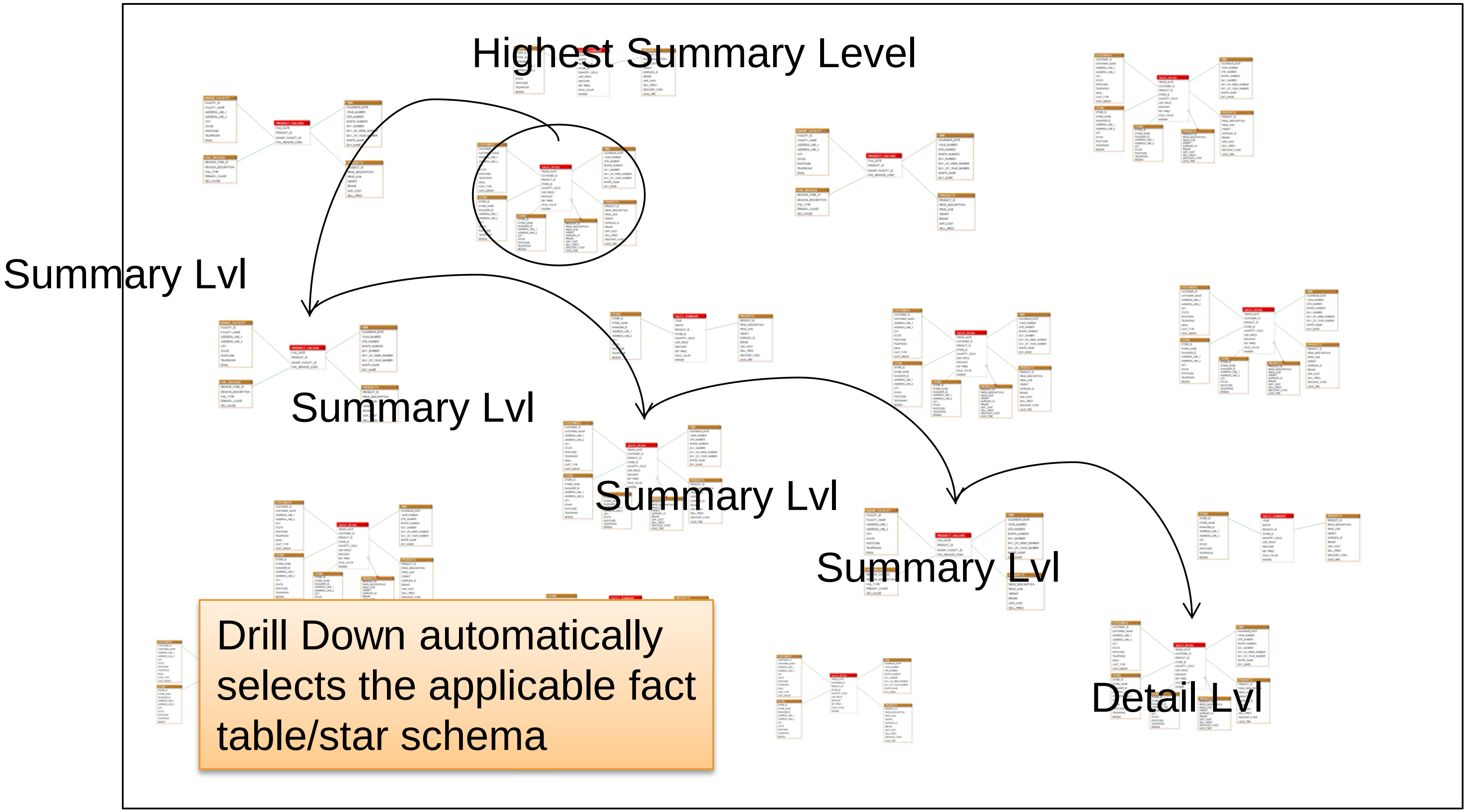
A common example: Event Tracking



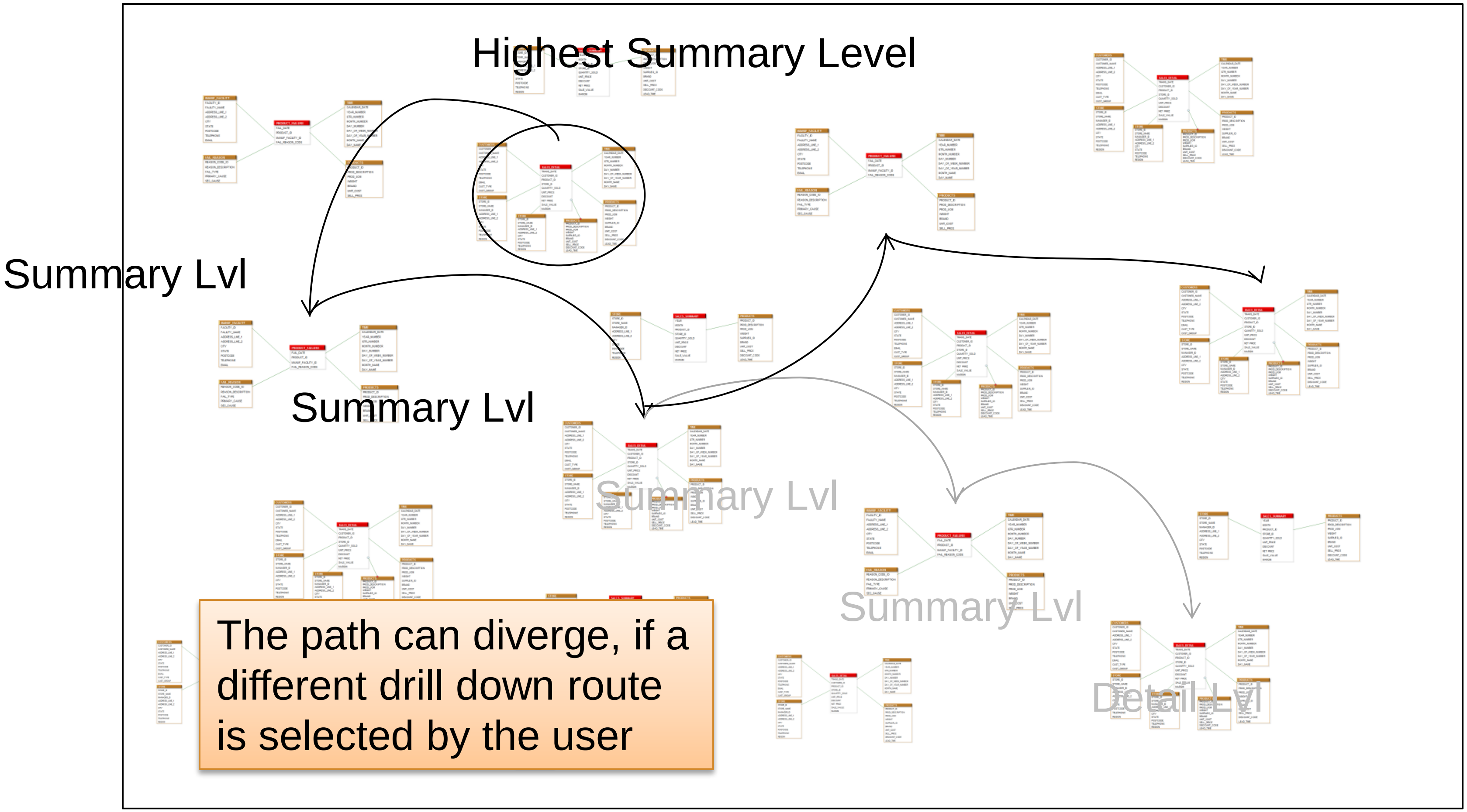
The implied ‘fact’ is that ONE failure occurred

Dimensional Data Warehouse

Dimensional Data Warehouse

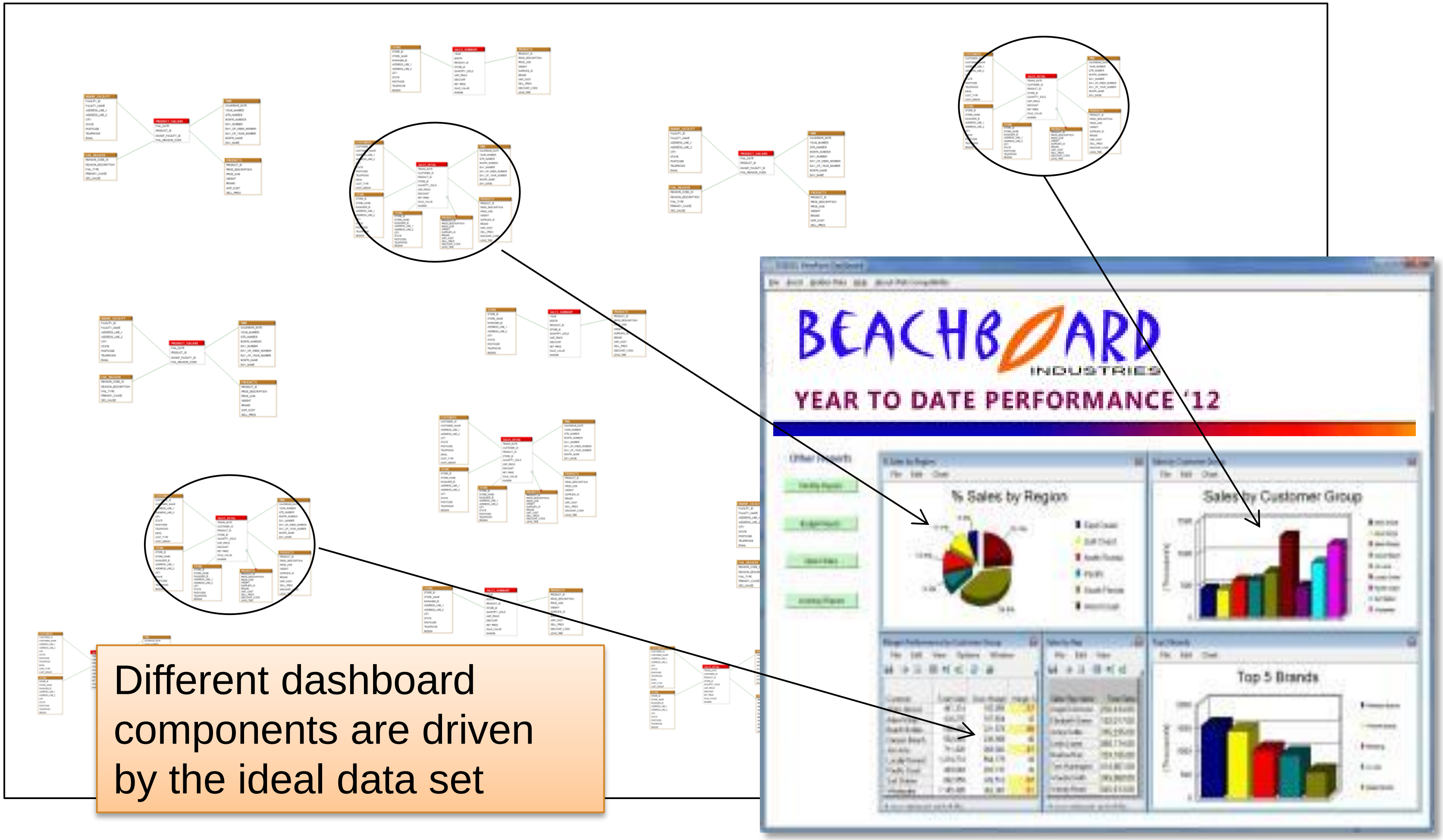


Dimensional Data Warehouse



The path can diverge, if a different drill down route is selected by the user

Dimensional Data Warehouse



Different dashboard components are driven by the ideal data set

Questions & Answers