The Art of Designing a Datawarehouse : the Retail Case Part 2

Retail Case Study : Grocery Chain

- 100 stores spread over a five-state area
- ~60,000 individual products on a store
- 80% come from outside manufacturers

Retail Case Study : Grocery Chain

Data is collected at

- cash registers as customers purchase products
- the back door, where vendors make deliveries

- Sales are much more important than deliveries
  - This is why we treated it first !
  - Now, we can move on.

Inventory

Having products at the right store at the right time :

- minimizes out-of-stocks (sale more)
- reduces overall inventory carrying costs

Inventory Models

- Inventory comes after sales, in terms of importance.

- A company is likely to invest less resources on the analysis of these data.

- **Less resources** = **less detailed information** in the datawarehouse

Inventory Models

1. Periodic Snapshot

2. Transaction-grain

3. Updated Records

PERIODIC SNAPSHOT

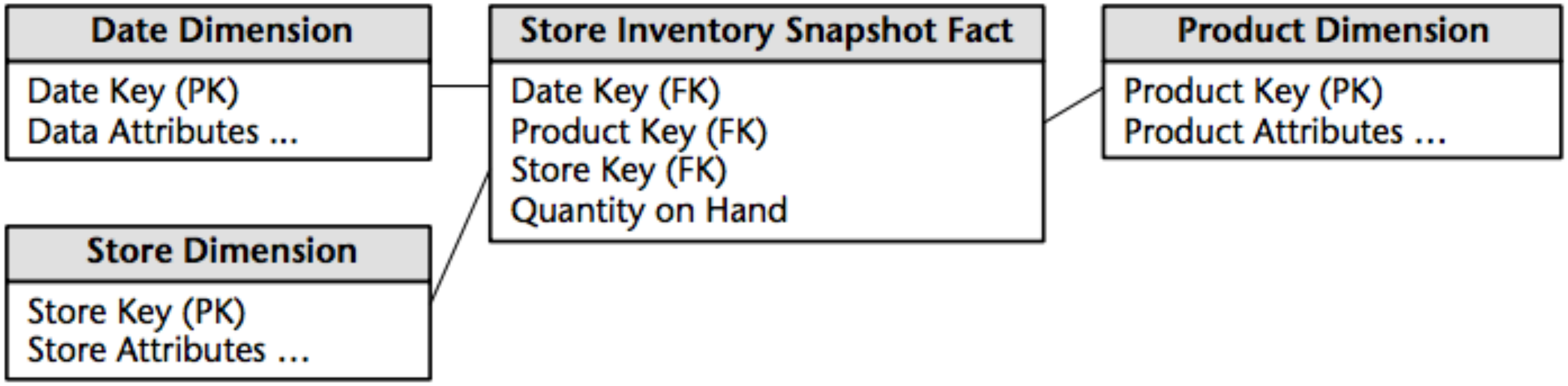
Periodic Snapshot

- **Regularly** record the **full state of the inventory**
  - every day
  - or every week
  - or every 2 hours (etc...)

- Example : record a summary of the status of the inventory at the end of each day

- **Granularity** : higher than real-life actions
  - but should be just right for profitable analysis

Periodic Snapshot Schema



Date Dimension

Date Key (PK)
Data Attributes ...

Store Inventory Snapshot Fact

Date Key (FK)
Product Key (FK)
Store Key (FK)
Quantity on Hand

Product Dimension

Product Key (PK)
Product Attributes ...

Store Dimension

Store Key (PK)
Store Attributes ...

Snapshot Fact Table

date	product	store	quantity
1	21	1	11
1	21	2	65
1	21	3	2332
1	21	4	53

Snapshot Fact Table

date	product	store	quantity
1	21	1	11
1	21	2	65
1	21	3	2332
1	21	4	53
1	31	1	234
1	31	2	23
1	31	3	4332
1	31	4	66

Snapshot Fact Table

date	product	store	quantity
1	21	1	11
1	21	2	65
1	21	3	2332
1	21	4	53
1	31	1	234
1	31	2	23
1	31	3	4332
1	31	4	66
2	21	1	33
2	21	2	234
2	21	3	44
2	21	4	22
2	31	1	44
2	31	2	544
2	31	3	445
2	31	4	22

Snapshot Fact Table

date	product	store	quantity
1	21	1	11
1	21	2	65
1	21	3	2332
1	21	4	53
1	31	1	234
1	31	2	23
1	31	3	4332
1	31	4	66
2	21	1	33
2	21	2	234
2	21	3	44
2	21	4	22
2	31	1	44
2	31	2	544
2	31	3	445
2	31	4	22

Inconvenient : dense snapshot tables

- Dense means: 1 row for each (product,store,day)

  - 60K products * 100 stores = 6M lines

  - assume 1 row (previous table) = 14 bytes
  ➔ 30 GB/year

- Compromise : reduce snapshot frequency
  daily (for last 60 days) weekly (for hystorical data)
  – question : how much storage on 12 months ?
  – always estimate the size of your tables

Now, which analytical queries can we answer ?

« today overall quantity of a given product »

« today overall quantity on a given store »

« overal quantity of a given product in july »

Now, which analytical queries can we answer ?

« today overall quantity of a given product » OK

« today overall quantity on a given store »   OK

« overal quantity of a given product in july » NO

Semiadditive facts

Q : « overal quantity of a given product in july » NO

- Why ? Cannot SUM inventory levels !

July 1st	:	10010	product 21 store 1
July 2nd	:	13016	product 21 store 1
July 3rd	:	19016	product 21 store 1

…

- Semiadditive facts : facts that are additive across some dimensions, but not all
  - store-dim, product-dim are ok, time is not ok!

Semiadditive facts

M	T	W	T	F
$50	**$50**	**$100**	**$100**	**$100**

- On Monday have $50, on Tuesday no deposit. Deposit $50 on Wednesday, then no actions.
- Friday night : cannot pretend we have $**400.**

Semiadditive facts

Q: «average bank account weekly balance» YES

M	T	W	T	F
$50	$50	$100	$100	$100

$80

Semiadditivity does not exclude mean

  - At 10am we have 10deg
  - At 11am we have 12deg
  - At 12am we have 15deg

- During last two hours AVG deg temp.was 12.3

Quiz

- Un fait (j, p, c, m, x) existe lorsque
  - un produit p
  - est acheté par un client c
  - le jour j
  - au magasin m
- La mesure x correspond au prix total.

- Fait snapshot ou transactionnel ?

Fait 1 : si Snapshot

- toutes les combinaisons (produit,client,date,magasin)
- intervalles reguliers (chaque jour)

id_produit	id_client	id_date	id_magasin	prix_tot_vente
1	1	**1**	1	6565
1	1	**1**	2	0
1	2	**1**	1	0
1	2	**1**	2	45654
2	1	**1**	1	0
2	1	**1**	2	0
2	2	**1**	1	0
2	2	**1**	2	0

Fait 1 : si Snapshot

- toutes les combinaisons (produit,client,date,magasin)
- intervalles reguliers (chaque jour)

id_pr duit
1
1
1
1
2
2
2
2

id_produit	id_client	id_date	id_magasin	prix_tot_vente
1	1	**2**	1	0
1	1	**2**	2	0
1	2	**2**	1	0
1	2	**2**	2	0
2	1	**2**	1	0
2	1	**2**	2	0
2	2	**2**	1	0
2	2	**2**	2	0

Fait 1 : si Snapshot

- toutes les combinaisons (produit,client,date,magasin)
- intervalles reguliers (chaque jour)

id_pr duit
1
1
1
1
2
2
2
2

id_prc duit
1
1
1
1
2
2
2
2

id_produit	id_client	id_date	id_magasin	prix_tot_vente
1	1	**3**	1	0
1	1	**3**	2	111
1	2	**3**	1	0
1	2	**3**	2	0
2	1	**3**	1	0
2	1	**3**	2	0
2	2	**3**	1	188

Fait 1 : si Transactionnel

- Tous les achats

id_produit	id_client	id_date	id_magasin	prix_tot_vente
1	1	1	1	6565
1	2	1	2	45654
1	1	3	2	111
2	2	3	1	188

Comparaison

- Quelques valeurs possibles
  - 60K produits
  - 10K clients au programme fidélité
  - 30 jours
  - 100 magasins
  - 100 produits achétés par semaine

- Transactionnel = 400**M** lignes * mois
- Snapshot = 1.8**T** lignes * mois (dont +99% à 0)

RECORD TRANSACTIONS

Record Transactions

The «expensive» solution (like, Amazon)

Record every transaction that affects inventory

1. Receive product
2. Place product into inspection hold
3. Release product from inspection hold
4. Return product to vendor due to inspection failure
5. Place product in bin
6. Authorize product for sale

Record Transactions

The «expensive» solution (like, Amazon)

Record every transaction that affects inventory

7. Pick product from bin
8. Package product for shipment
9. Ship product to customer
10. Receive product from customer
11. Return product to inventory from customer return
12. Remove product from inventory

Record Transactions

- Needs a special dimension for **transaction-type**
- Other dimensions : product, order, status, date

Receive

Product_id	Order_id	Transaction_type	Status	Date
1	**No_order**	Receive product	**COMPLETED**	**2015/12/11**

Inspect

Product_id	Order_id	Transaction_type	Status	Date
1	**No_order**	Receive product	**COMPLETED**	**2015/12/11**
1	**No_order**	Inspection Hold	**COMPLETED**	**2015/12/11**

Ask for authorization

Product_id	Order_id	Transaction_type	Status	Date
1	**No_order**	Receive product	**COMPLETED**	**2015/12/11**
1	**No_order**	Inspection Hold	**COMPLETED**	**2015/12/11**
1	**No-order**	Authorized for sale	**PENDING**	**2015/12/11**

Authorize

Product_id	Order_id	Transaction_type	Status	Date
1	**No_order**	Receive product	**COMPLETED**	**2015/12/11**
1	**No_order**	Inspection Hold	**COMPLETED**	**2015/12/11**
1	**No-order**	Authorized for sale	**PENDING**	**2015/12/11**
1	**No-order**	Authorized for sale	**COMPLETED**	**2015/12/12**

Ship

Product_id	Order_id	Transaction_type	Status	Date
1	**No_order**	Receive product	**COMPLETED**	**2015/12/11**
1	**No_order**	Inspection Hold	**COMPLETED**	**2015/12/11**
1	**No-order**	Authorized for sale	**PENDING**	**2015/12/11**
1	**No-order**	Authorized for sale	**COMPLETED**	**2015/12/12**
1	**20**	Ship to Customer	**COMPLETED**	**2016/12/01**

Record Transaction

- When selling thousands of items of the same type : even more cumbersome than before
- But detailed…

Can answer analytical queries that periodic snapshots can't

- How many separate shipments did we receive from a given vendor?
- On which products have we had more than one round of inspection failures ?

UPDATED RECORDS

(previous) Record Transactions

1 fact table row = 1 movement of 1 product

- Movement : receiving, inspection, bin placement, authorization to sell, shipping

Updated Records

1 fact table row = **ALL** movements of 1product

- **UPDATE the fact table row over and over** until the product leaves the warehouse
  - – Eg., shipping can take values : { NO, pending, OK }

Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	INVALID	Package_1	undefined
2	10	INVALID	Package_2	undefined

Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	INVALID	Package_3	undefined
2	10	INVALID	Package_3	undefined

Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	VALIDATED	Package_3	undefined
2	10	VALIDATED	Package_3	undefined

Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	VALIDATED	Package_3	2015/17/12
2	10	VALIDATED	Package_3	2015/17/12

Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	VALIDATED	Package_3	2018/17/12
2	10	VALIDATED	Package_3	2018/17/12
3	20	WAITING	Package_1	undefined
4	20	WAITING	Package_1	undefined

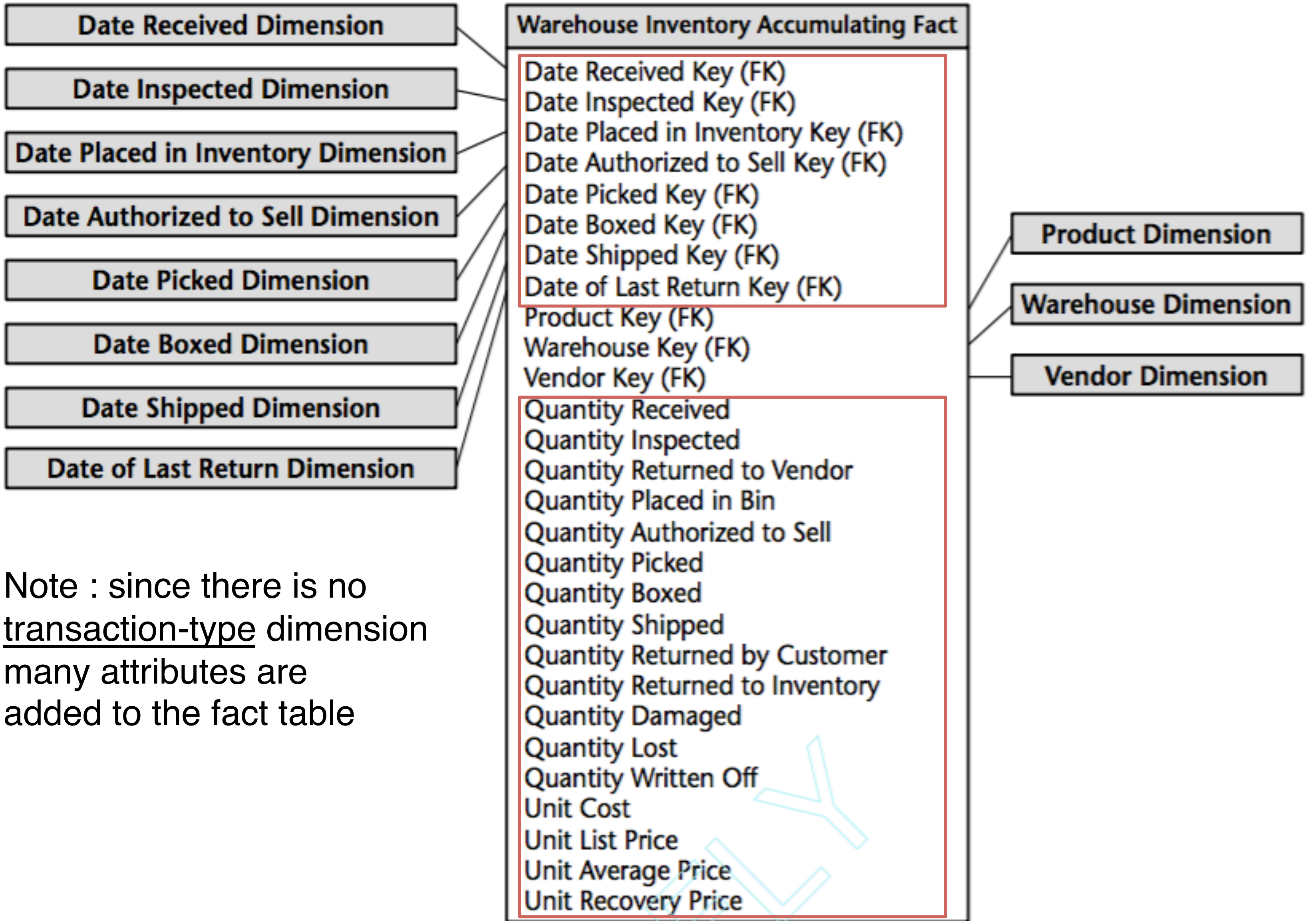
Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	VALIDATED	Package_3	2018/17/12
2	10	VALIDATED	Package_3	2018/17/12
3	20	VALIDATED	Package_1	undefined
4	20	VALIDATED	Package_1	undefined

Updated Records

Product_id	Order_id	Status_id	Boxing_id	Shipping_date_id
1	10	VALIDATED	Package_3	2018/17/12
2	10	VALIDATED	Package_3	2018/17/12
3	20	VALIDATED	Package_1	2019/03/01
4	20	VALIDATED	Package_1	2019/03/01

In reality : a much more complex schema



Date Received Dimension

Date Inspected Dimension

Date Placed in Inventory Dimension

Date Authorized to Sell Dimension

Date Picked Dimension

Date Boxed Dimension

Date Shipped Dimension

Date of Last Return Dimension

Warehouse Inventory Accumulating Fact

Date Received Key (FK)
Date Inspected Key (FK)
Date Placed in Inventory Key (FK)
Date Authorized to Sell Key (FK)
Date Picked Key (FK)
Date Boxed Key (FK)
Date Shipped Key (FK)
Date of Last Return Key (FK)
Product Key (FK)
Warehouse Key (FK)
Vendor Key (FK)
Quantity Received
Quantity Inspected
Quantity Returned to Vendor
Quantity Placed in Bin
Quantity Authorized to Sell
Quantity Picked
Quantity Boxed
Quantity Shipped
Quantity Returned by Customer
Quantity Returned to Inventory
Quantity Damaged
Quantity Lost
Quantity Written Off
Unit Cost
Unit List Price
Unit Average Price
Unit Recovery Price

Product Dimension

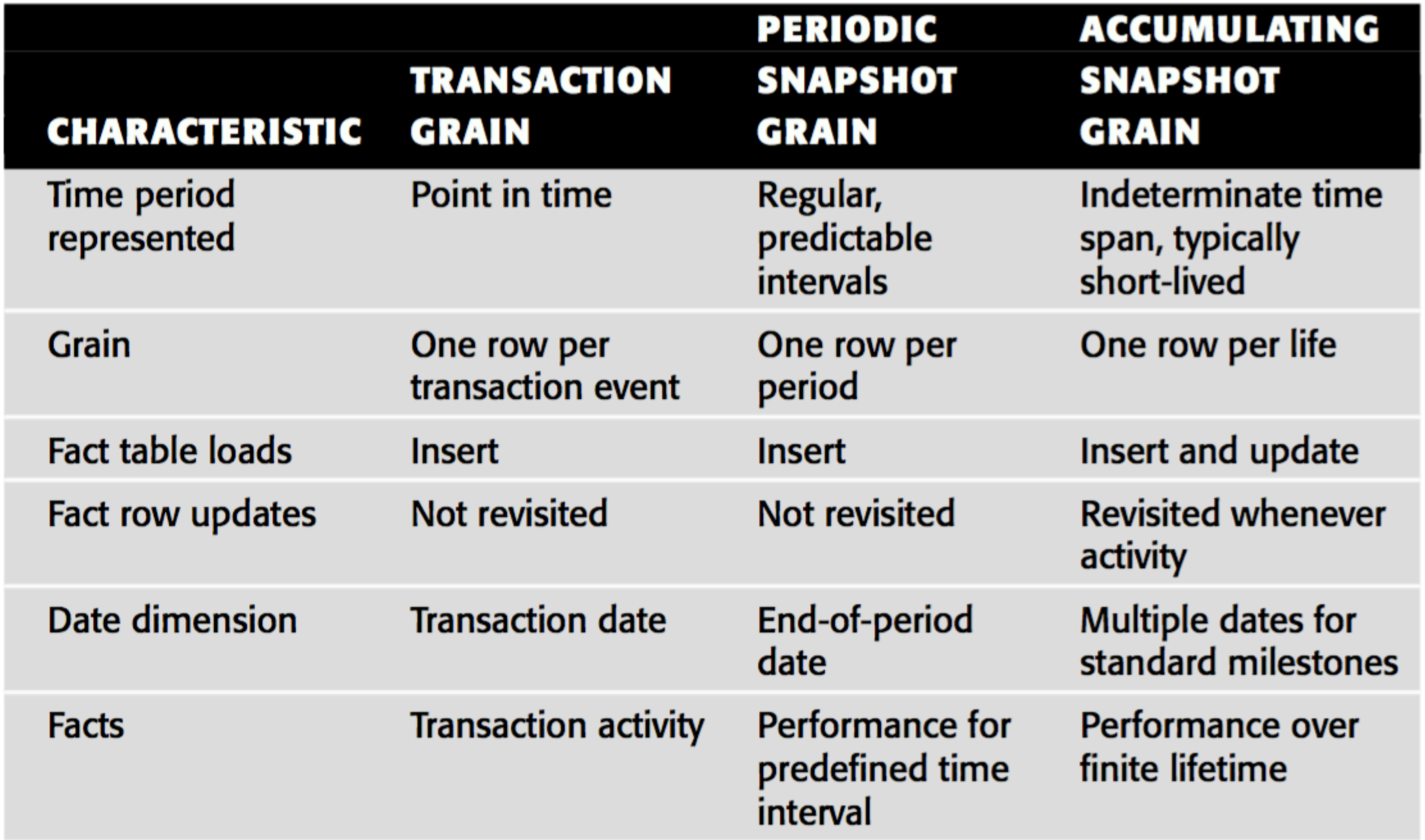
Warehouse Dimension

Vendor Dimension

Note : since there is no transaction-type dimension many attributes are added to the fact table

Summing up : types of fact tables

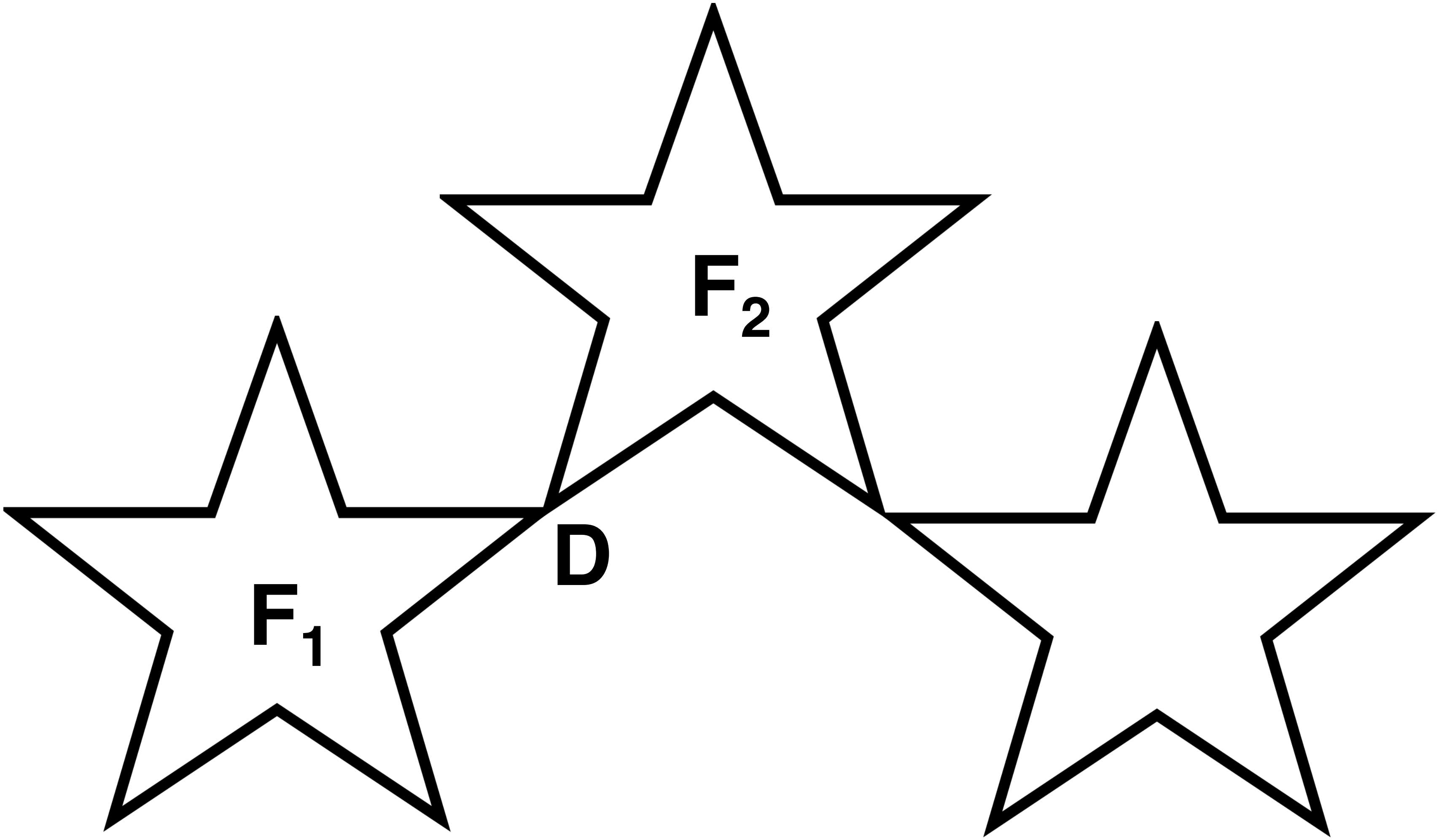
updated records



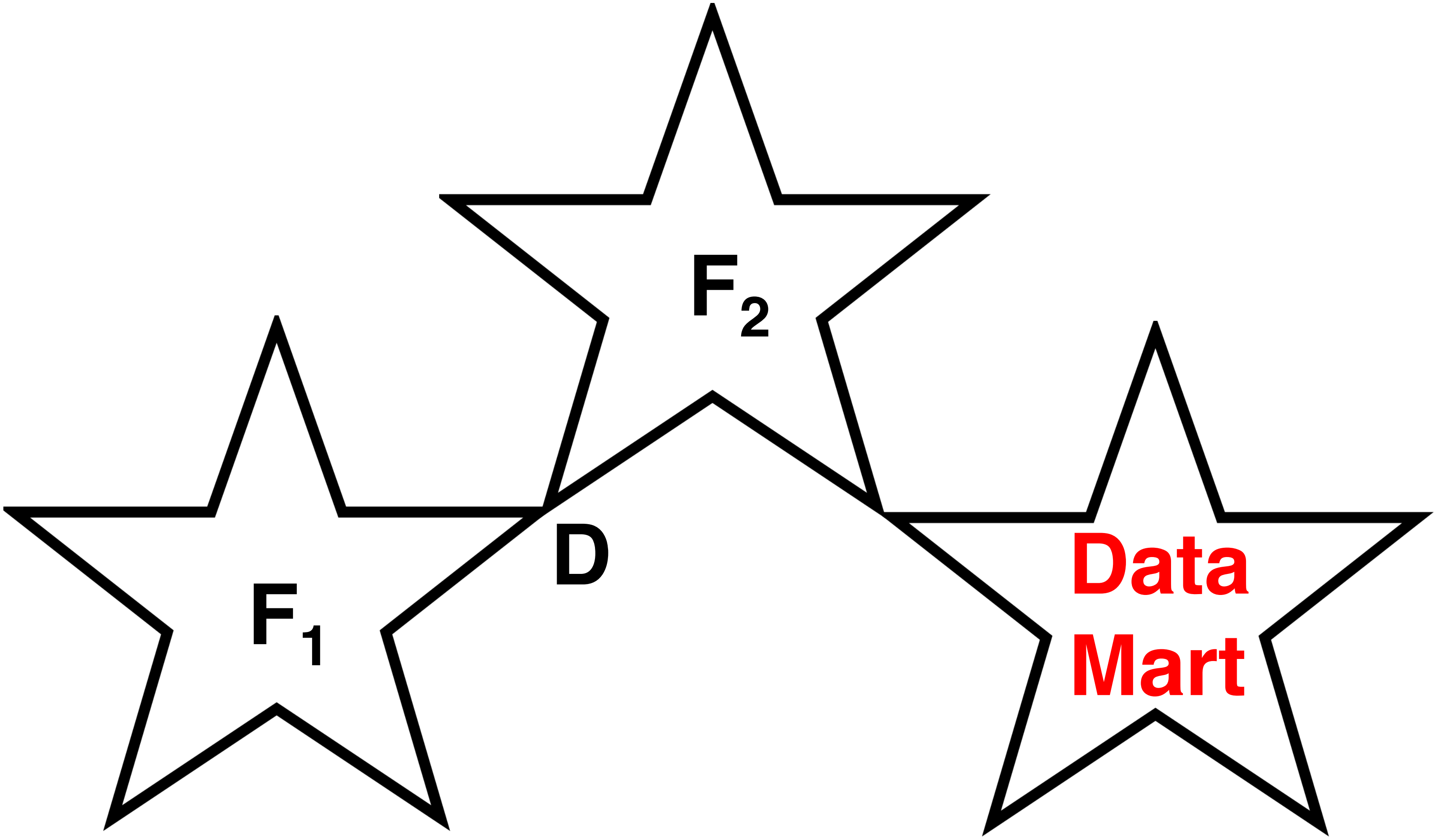
		PERIODIC	ACCUMULATING
	TRANSACTION	SNAPSHOT	SNAPSHOT
CHARACTERISTIC	GRAIN	GRAIN	GRAIN
Time period represented	Point in time	Regular, predictable intervals	Indeterminate time span, typically short-lived
Grain	One row per transaction event	One row per period	One row per life
Fact table loads	Insert	Insert	Insert and update
Fact row updates	Not revisited	Not revisited	Revisited whenever activity
Date dimension	Transaction date	End-of-period date	Multiple dates for standard milestones
Facts	Transaction activity	Performance for predefined time interval	Performance over finite lifetime

Sharing Dimensions : beyond star schemas

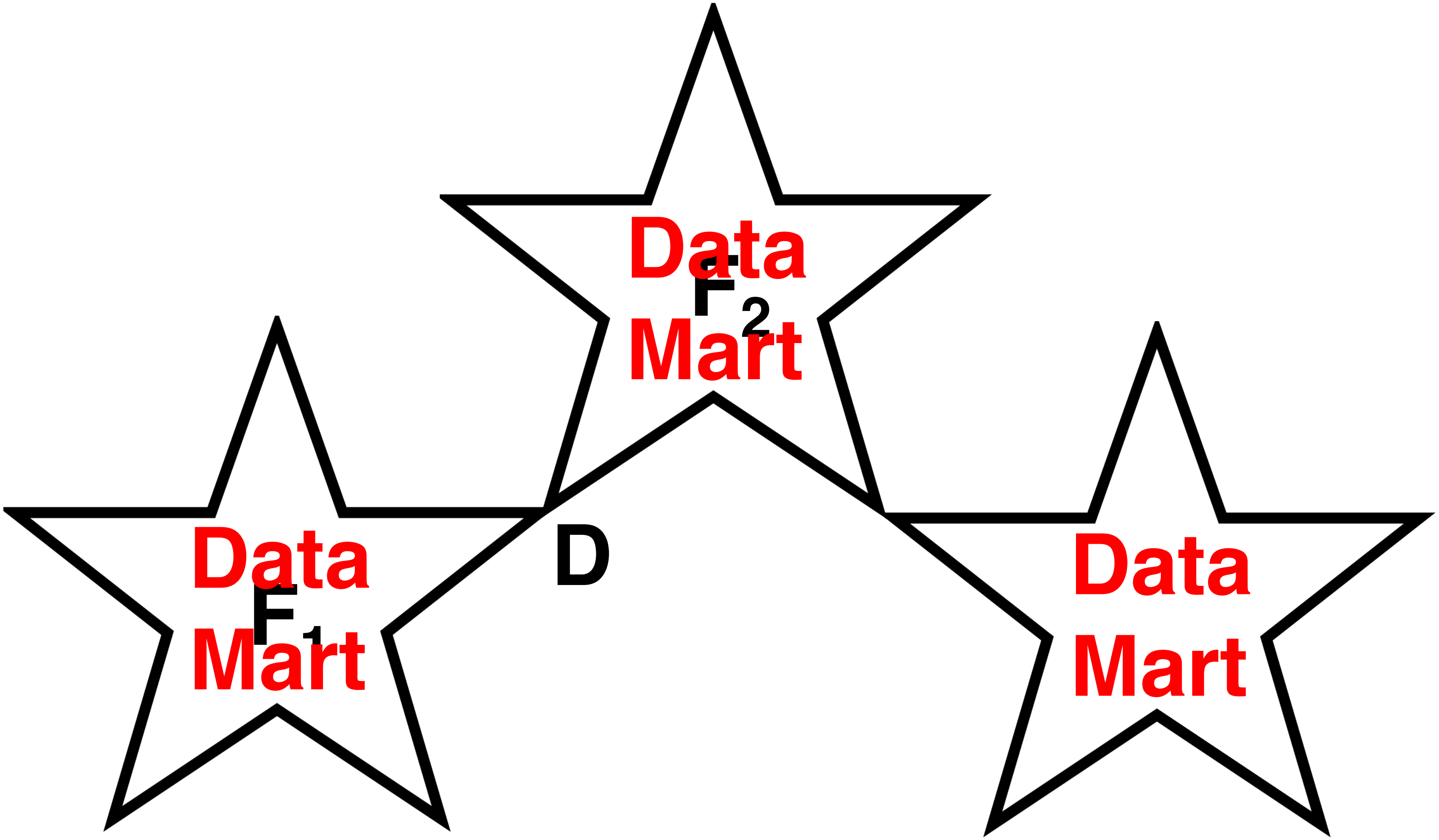
DW Model : Constellation Schemas



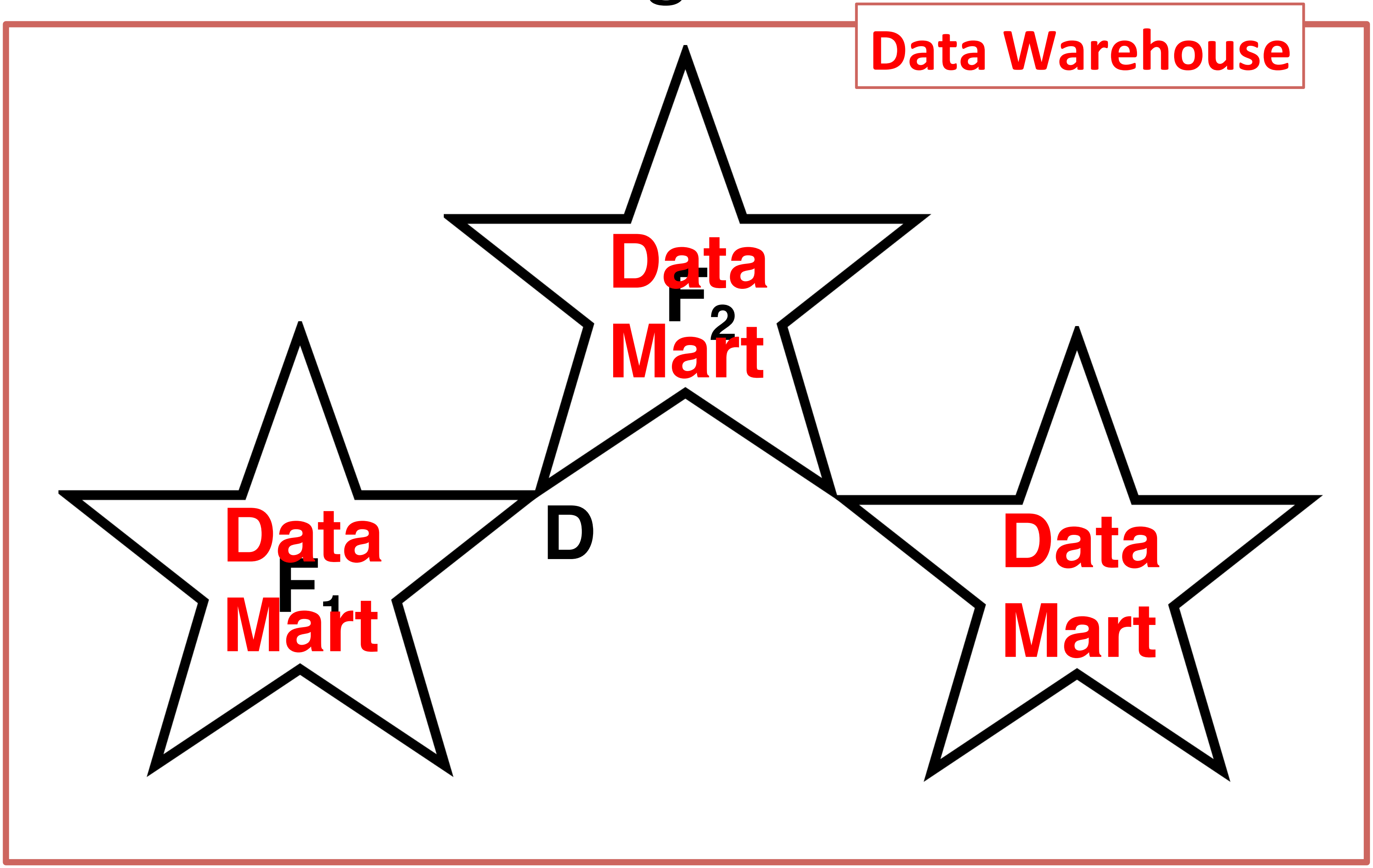
Data Mart : Single Fact-table DW



Data Mart : Single Fact-table DW



Data Mart : Single Fact-table DW



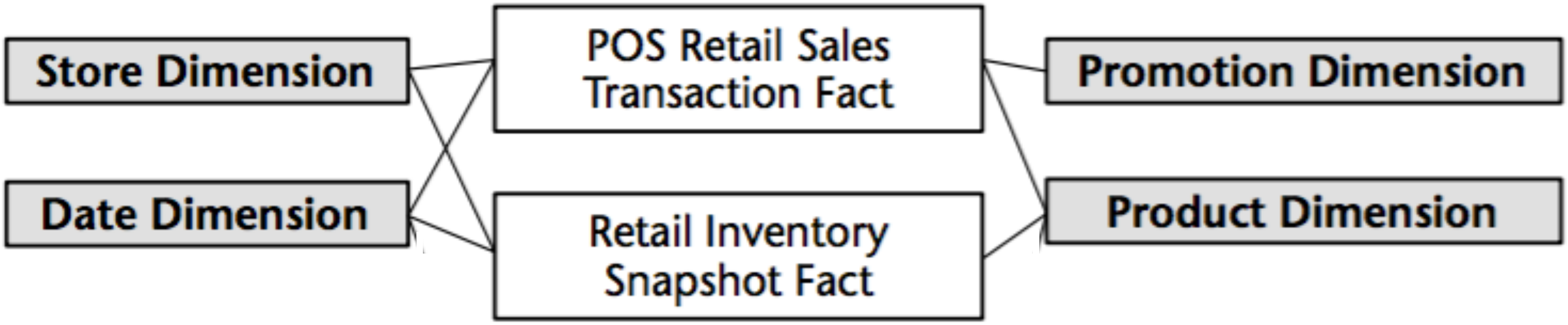
Data mart

- Business oriented star-schema (or few of them)

- Virtual (users have views on data) or Materialized (different database instances; more expensive)

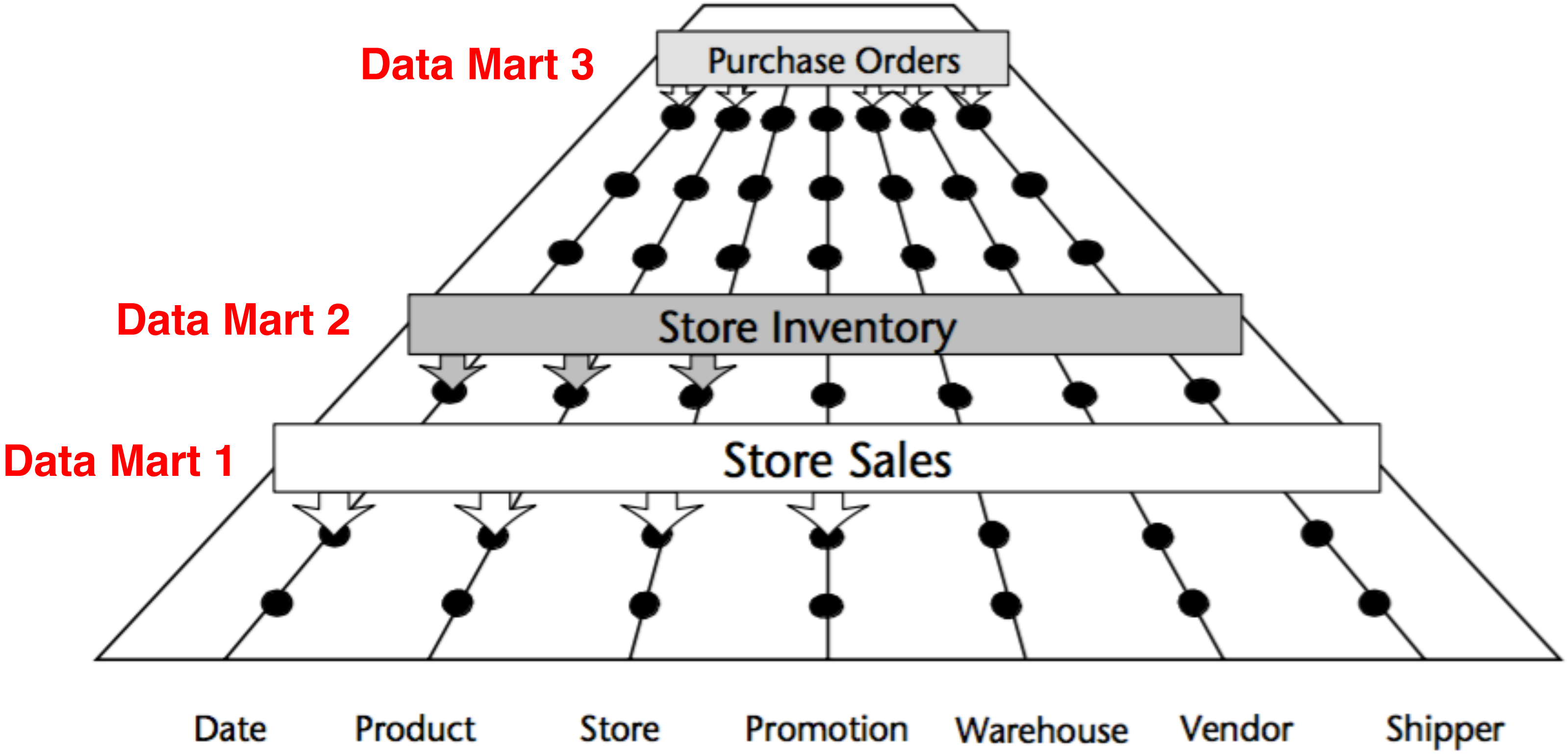
- Important notion also for privacy reasons : data should not be visible to any user !

Sharing Dimensions



- Star-Constellation Schema **!=** Snowflake Schema

Shared Dimensions



Sharing Dimensions : Which Level of Detail ?

- *Date* dimension identical for **retail** and **inventory**
  - also *product* and *store*

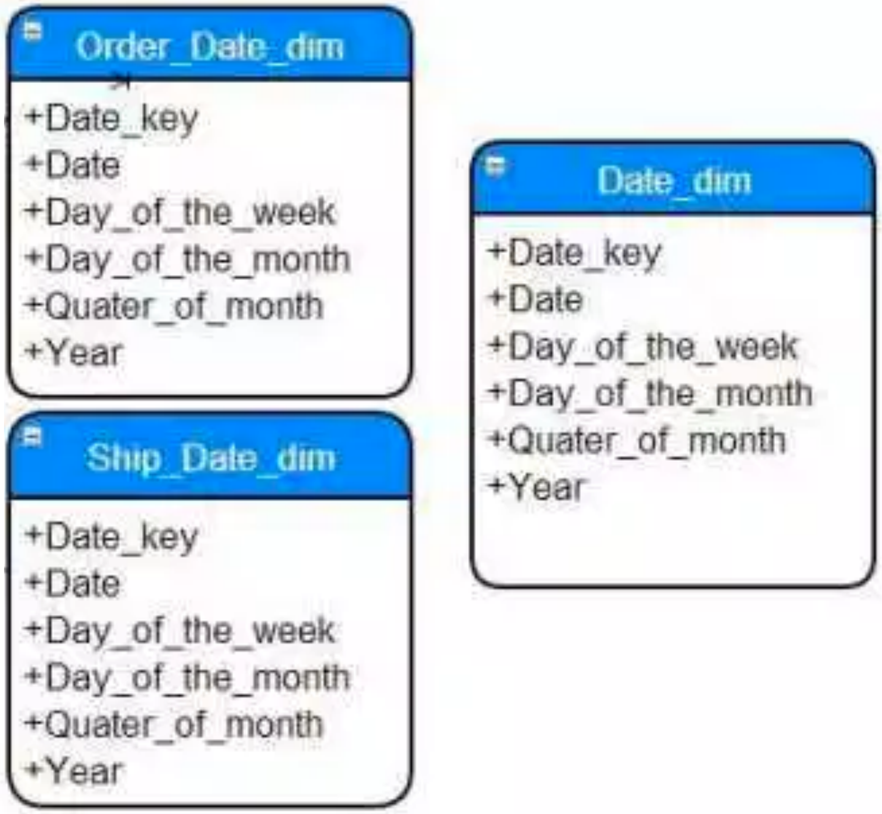
- Use the **more detailed** version of the dimension !

  - Previously defined tables for retail **may be not enough detailed** and lack of attributes useful for inventory analysis

  - Eg product dimension: <u>minimum reorder quantity</u>
  - Eg store dimension : <u>storage square footages</u>

Sharing Date Dimension

- Problem : order date and shipping date **both** dates
- Better to specify order and ship date dimension



Order_Date_dim

+Date_key
+Date
+Day_of_the_week
+Day_of_the_month
+Quater_of_month
+Year

Ship_Date_dim

+Date_key
+Date
+Day_of_the_week
+Day_of_the_month
+Quater_of_month
+Year

Date_dim

+Date_key
+Date
+Day_of_the_week
+Day_of_the_month
+Quater_of_month
+Year

Create illusion of independent date-tables using (virtual) views

-
  ```
  CREATE VIEW ORDER_DATE
  AS SELECT * FROM DATE
  ```

-
  ```
  CREATE VIEW SHIP_DATE
  AS SELECT * FROM DATE
  ```

Create illusion of independent date-tables using (virtual) views

- CREATE VIEW ORDER_DATE
  AS SELECT $X_1,\dots,X_n$ FROM DATE

  more detailed

- CREATE VIEW SHIP_DATE
  AS SELECT $X_1,\dots,X_{m<n}$ FROM DATE

  less detailed