Data Warehousing and Data Mining

— L2: OLAP —

- Why and What are Data Warehouses?

Data Analysis Problems

- The same data found in many different systems
  - Example: customer data across different departments
  - The same ifferently
- Heterogeneo
  - Relational DBMS, OnLi ction Processing (OLTP)
  - Unstructured data in files (e.g., MS Excel) and documents (e.g., MS Word)

Data Analysis Problems (Cont'd)

- Data is suited for operational systems
  - Accounting,billing,etc.
  - Do not support analysis across business functions
- Data quality i
  - Missing data, imprecise ferent use of systems
- Data are "volatile"
  - Data deleted in operational systems (6months)
  - Data change over time – no historical information

Solution: Data Warehouse

- Defined in many different ways, but not rigorously.
  - A decision support database that is maintained separately from the organization's operational database
  - Support inform g a solid platform of consolidated, h
- "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."—W. H. Inmon
- Data warehousing:
  - The process of constructing and using data warehouses

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales.
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing.
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted.

Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems.
  - Operational database: current value data.
  - Data warehouse: provides information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain "time element".

Data Warehouse—Non-Volatile

1. A physically separate store of data transformed from the operational environment.
2. Operational update of data does not occur in the data warehouse en

   - Does not require transact sing, recovery, and concurrency control s
   - Requires only two operations in data accessing:
     - *initial loading of data* and *access of data*.

Data Warehouse Architecture

- Extract data from operational data sources
  - clean, transform
- Bulk load/refresh
  - warehouse is of
- OLAP-server prov multidimensional view
- Multidimensional-olap (Essbase, oracle express)
- Relational-olap (Redbrick, Informix, Sybase, SQL server)

Data Warehouse Architecture

All subjects, integrated

Advanced analysis

Function-oriented systems

Subject-oriented systems

Why Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—t P queries, multidimensio
- Different functions and different da
  - missing data: Decision support rical data which operational DBs do not typically maintain
  - data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

Why OLAP Servers?

- Different workload:
  - OLTP (on-line transaction processing)
    - Major task of traditional relational DBMS
    - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
  - OLAP (on-line
    - Major task of
    - Data analysis and decision making
- Queries hard/infeasible for OLTP, e.g.,
  - Which **week** we have the largest sales?
  - Does the sales of **dairy products** increase over time?
  - Generate a **spread sheet** of total sales by state and by year.
- Difficult to represent these queries by using SQL ← Why?

OLTP vs. OLAP

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date det iso	historical, zed, multidimensional d, consolidated
usage	repetitive	
access	read/write index/hash on prim. key	
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

Comparisons

	Databases	Data Warehouses
Purpose	Many purposes; Flexible and general	One purpose: Data analysis
Conceptual Model	ER	ltidimensional
Logical Model	(Normalized) Relational	ormalized) Star schema / cube/cuboids
Physical Model	Relational Tables	ROLAP: Relational tables MOLAP: Multidimensional arrays
Query Language	SQL (hard for analytical queries)	MDX (easier for analytical queries)
Query Processing	B+-tree/hash indexes, Multiple join optimization, Materialized	Bitmap/Join indexes, Star join, Materialized data cube

- The Multidimensional Model

The Multidimensional Model

- A data warehouse is based on a multidimensional data model which views data in the form of a data cube, which is a multidimensional generalization of 2D spread sheet.
- Key concepts:
  - **Facts**: the s
    - Typically tr er types includes snapshots, etc.
    - Measures: numbers that can d
    - Dimensions: context of the measure
  - Hierarchies:
    - Provide contexts of different granularities (aka. grains)
- Goals for dimensional modeling:
  - Surround facts with as much relevant context (dimensions) as possible ← Why?

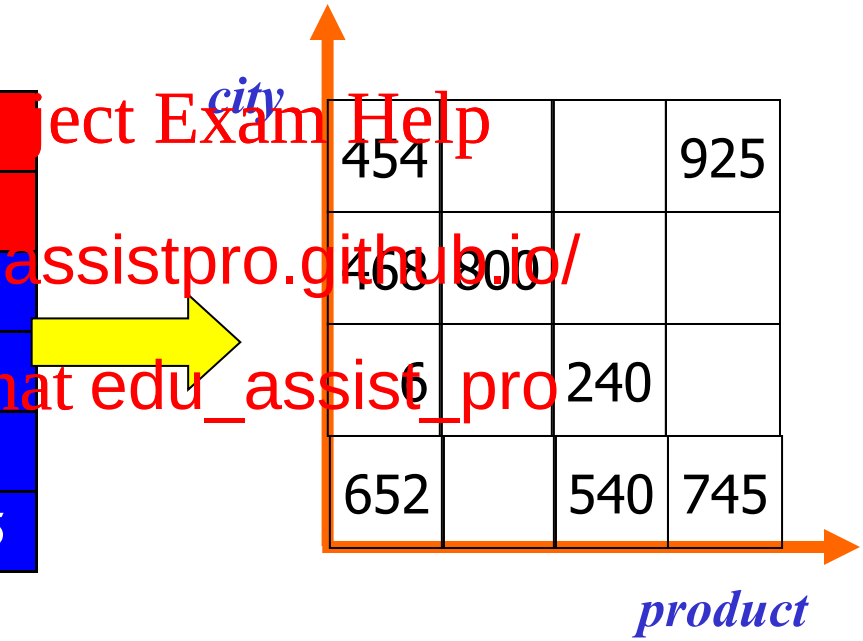
Supermarket Example

- Subject: analyze total sales and profits
- Fact: Each Sales **Transaction**
  - Measure: Dollars_Sold, Amount_Sold, Cost
  - Calculated M
- Dimensions:
  - Store
  - Product
  - Time

Visualizing the Cubes

- A valid instance of the model is a data cube

total Sales		product			
		p1	p2		
city	NY	$454	-		
	LA	$468	$800		-
	SD	$296	-	$240	-
	SF	$652	-	$540	$745



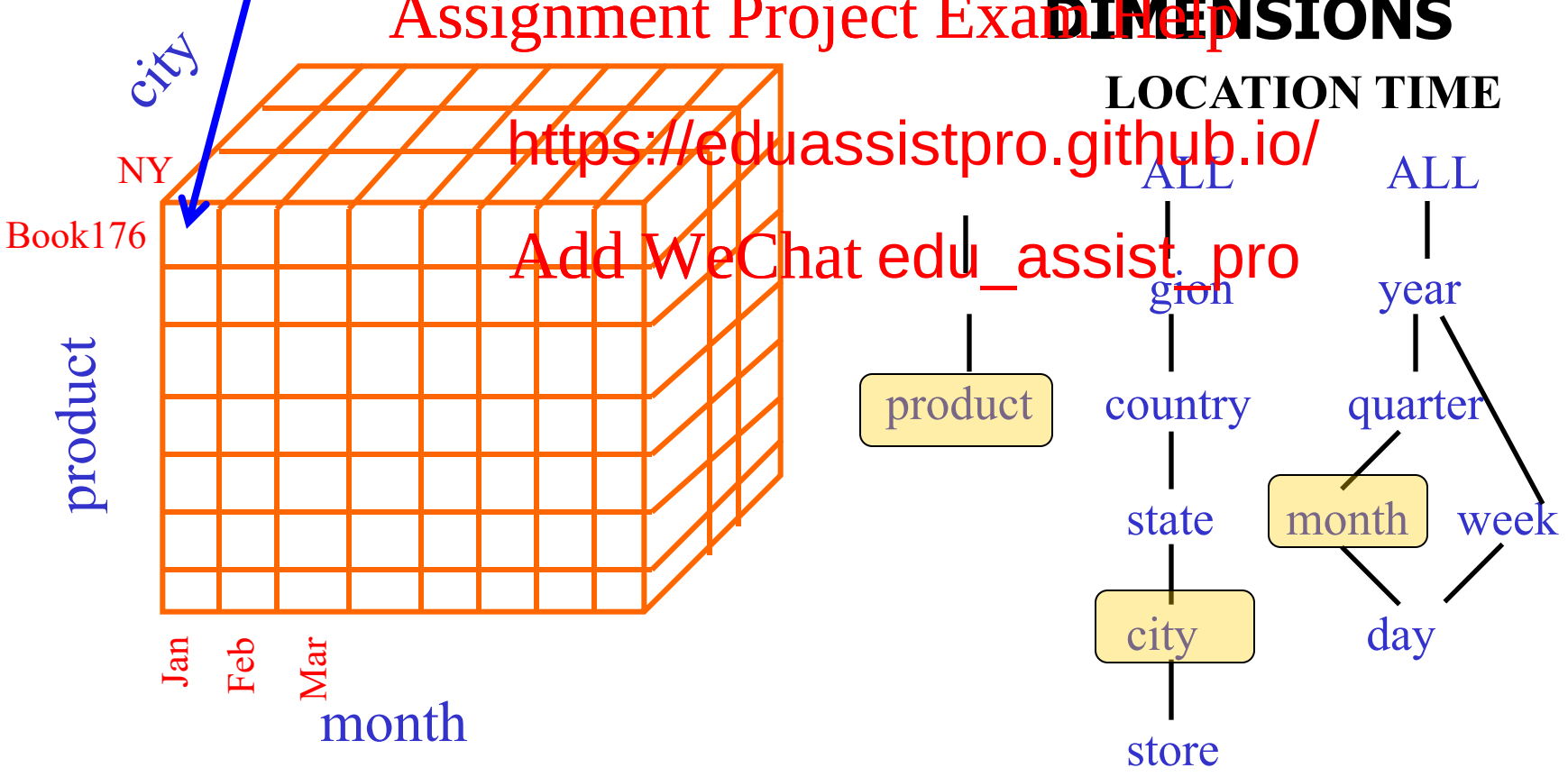
Concepts: cell, fact (=non-empty cell), measure, dimensions

Q: How to generalize it to 3D?

3D Cube and Hierarchies

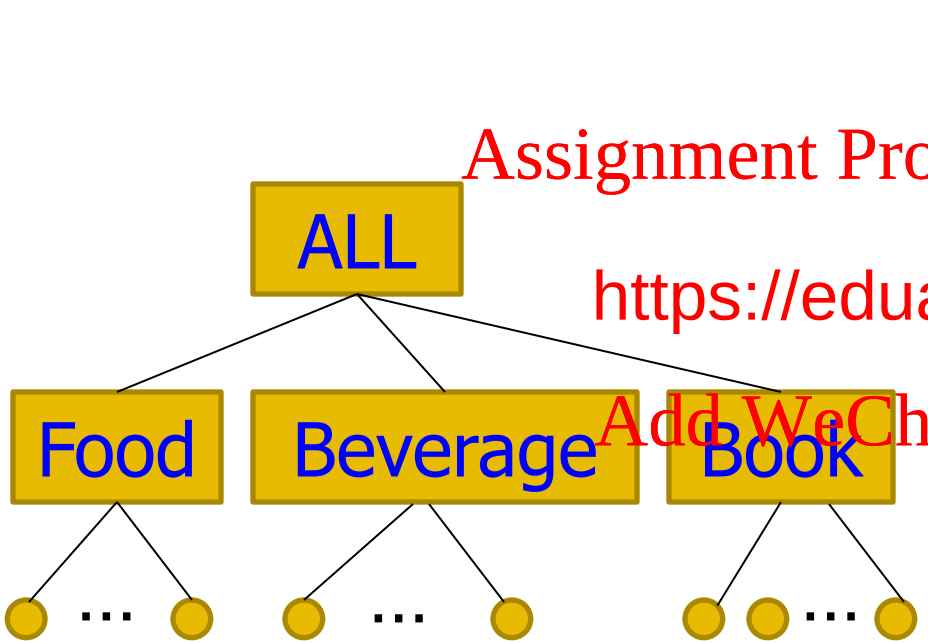
Concepts: hierarchy (a tree of dimension values), level

Sales of book176 in NY in Jan can be found in this cell



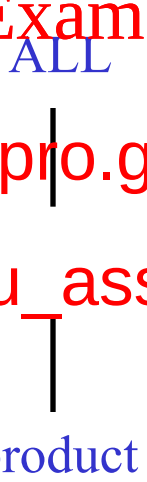
Hierarchies

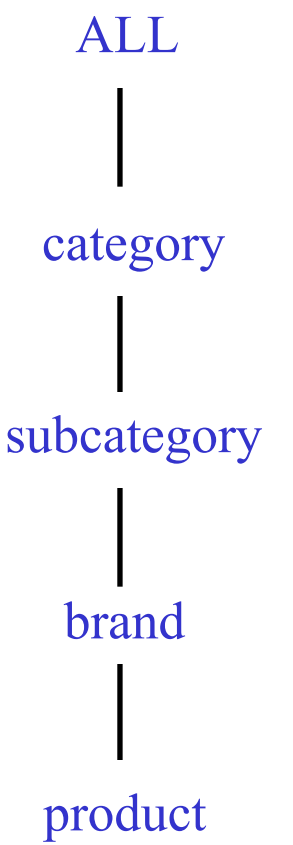
Concepts: hierarchy (a tree of dimension values), level



Book176

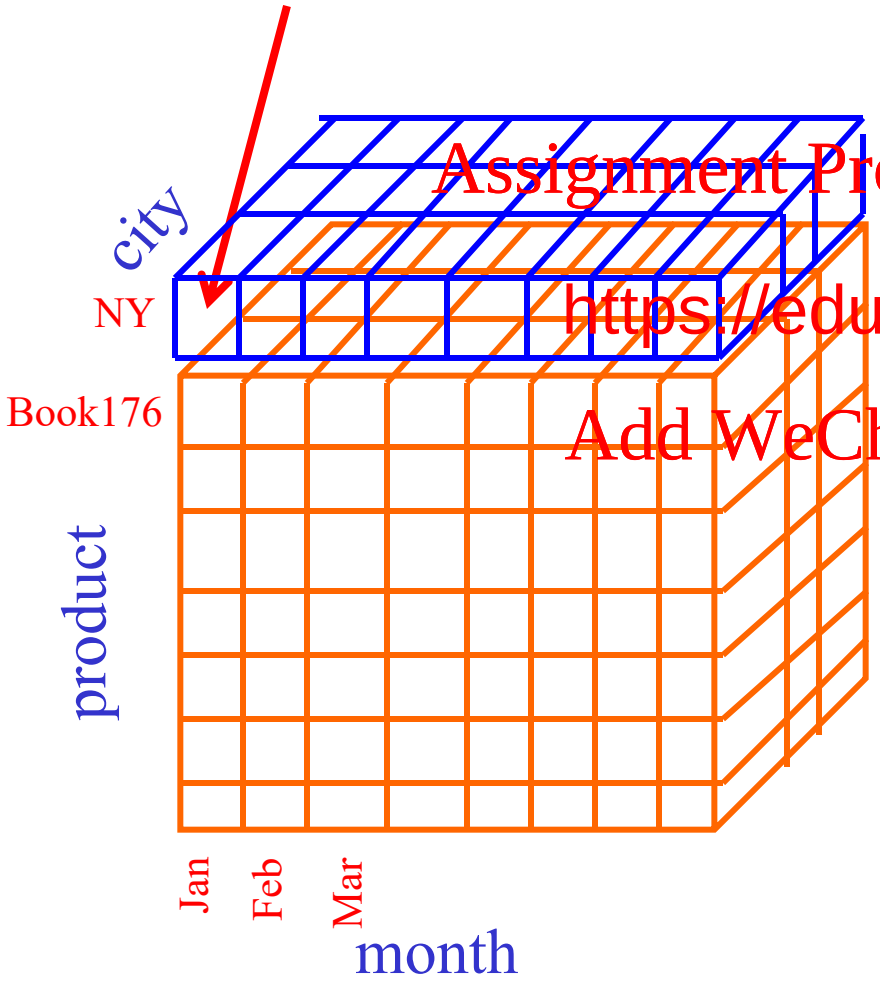
Which design is better? Why?

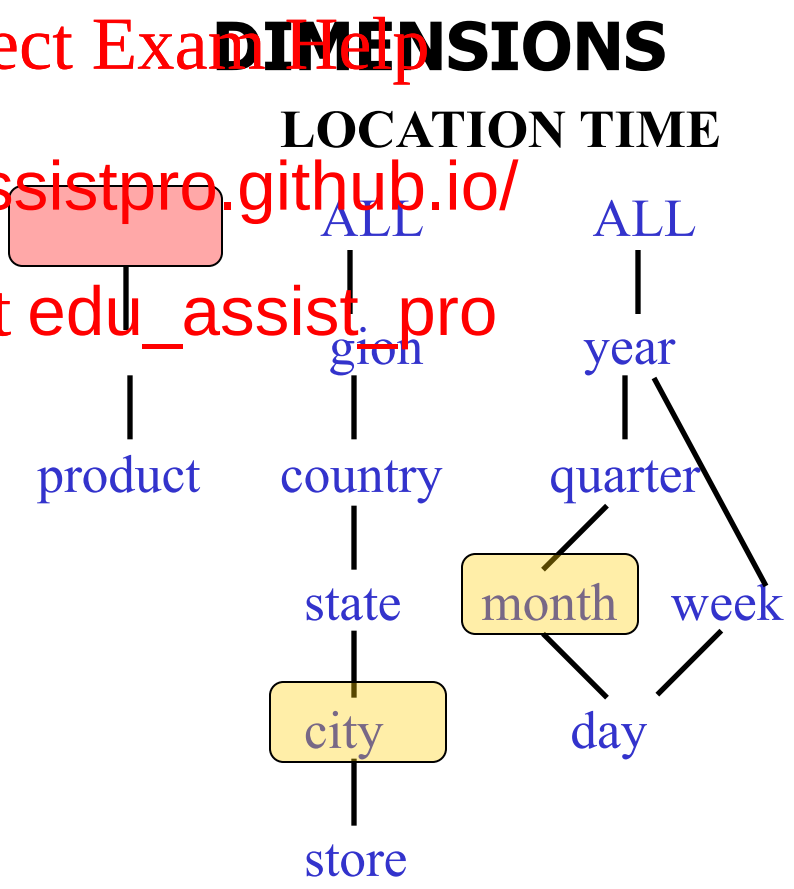




The (city, moth) Cuboid

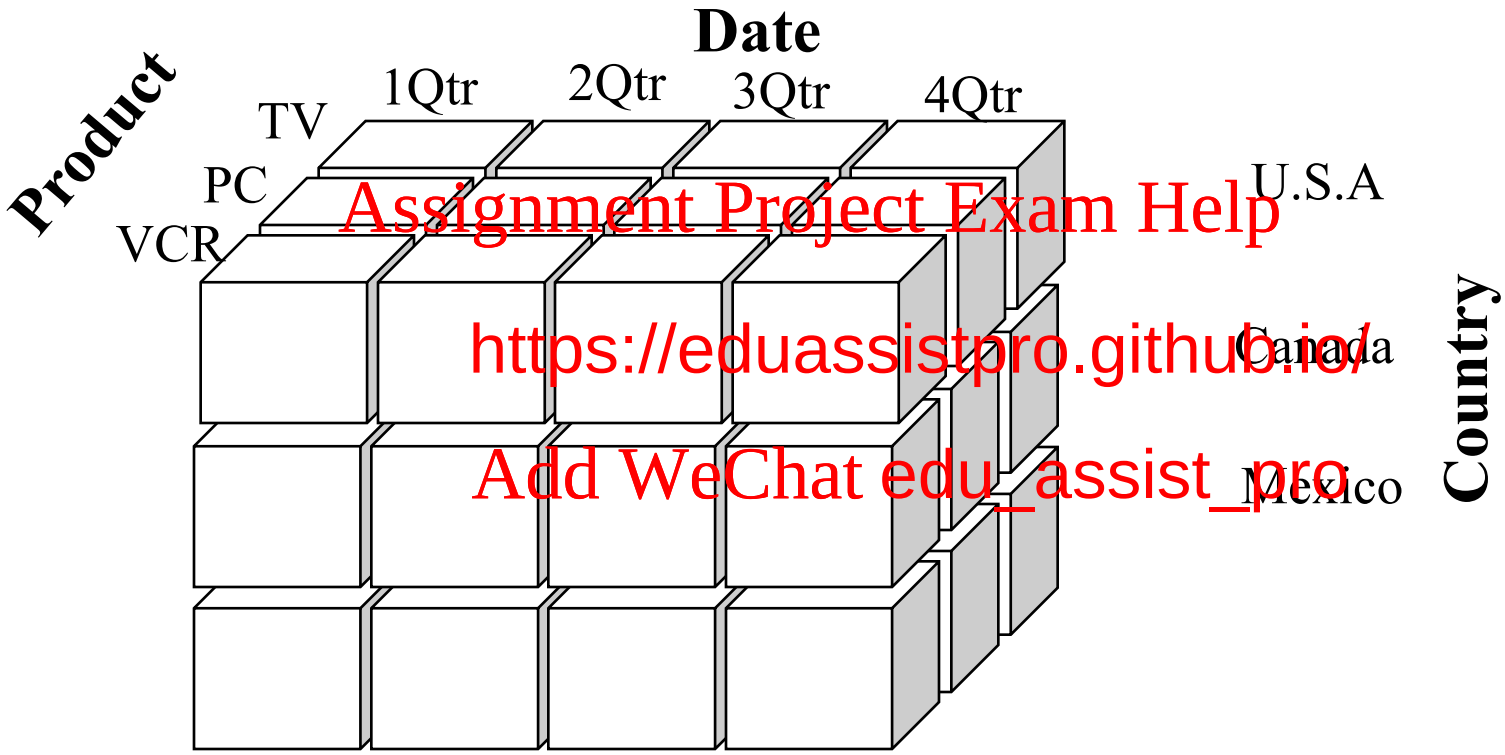
Sales of ALL_PROD in NY in Jan





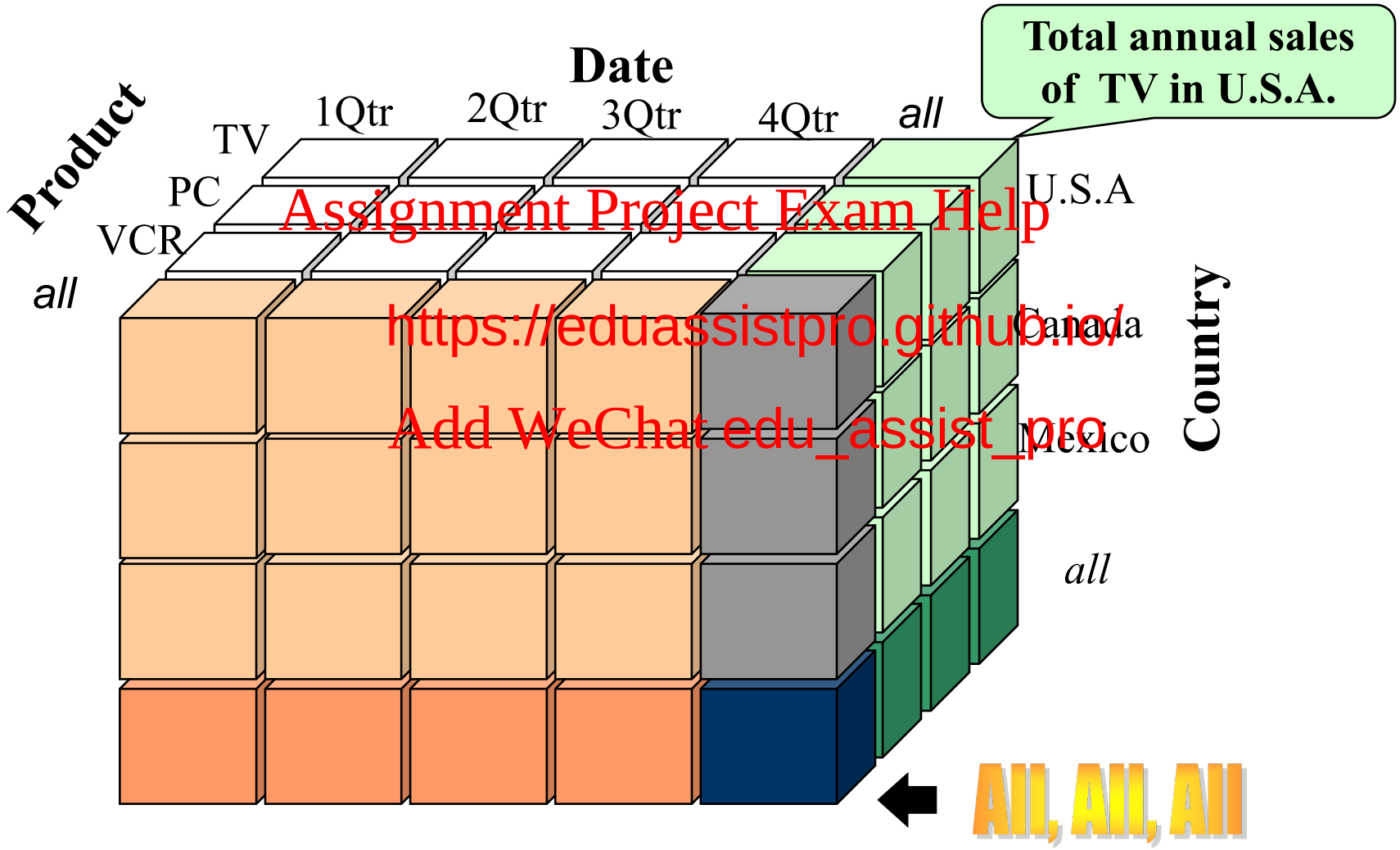
All the Cuboids

Assume: no other non-ALL levels on all dimensions.

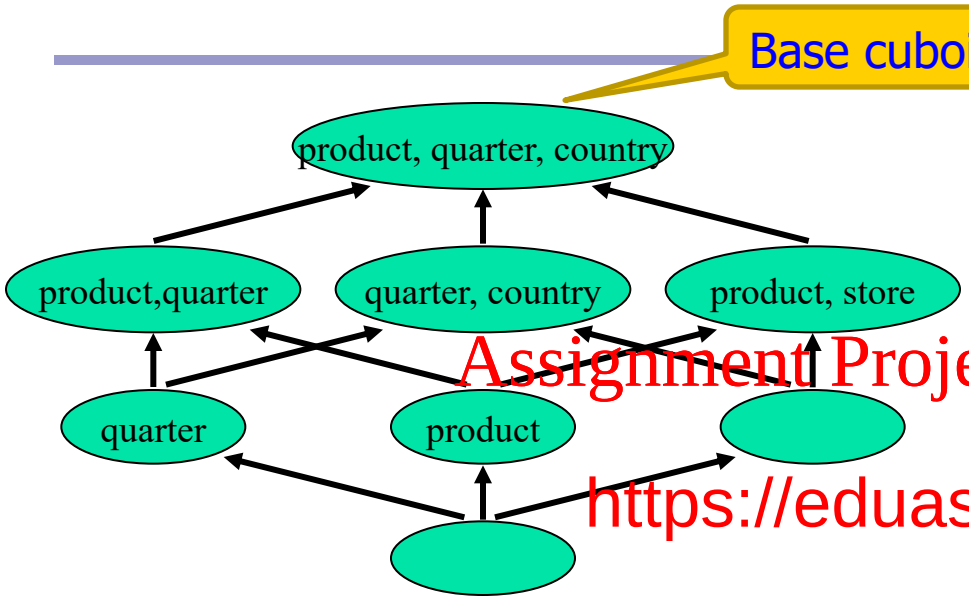


All the Cuboids /2

Assume: no other non-ALL levels on all dimensions.



Lattice of the cuboids



- n-dim cube can be represented as ( ), where D_i is the set of allowed values on the i-th dimen
  - if $D_i = L_i$ (a particular level), then Di = all descendant dimension values of L_i.
  - ALL can be omitted and hence reduces the effective dimensionality
- A complete cube of d-dimensions consists of $\prod_{i=1}^{d}(n_i + 1)$ cuboids, where n_i is the number of levels (excluding ALL) on i-th dimension.
  - They collectively form a lattice.

Properties of Operations

- All operations are closed under the multidimensional model
  - i.e., both input and output of an operation is a cube
- So that they

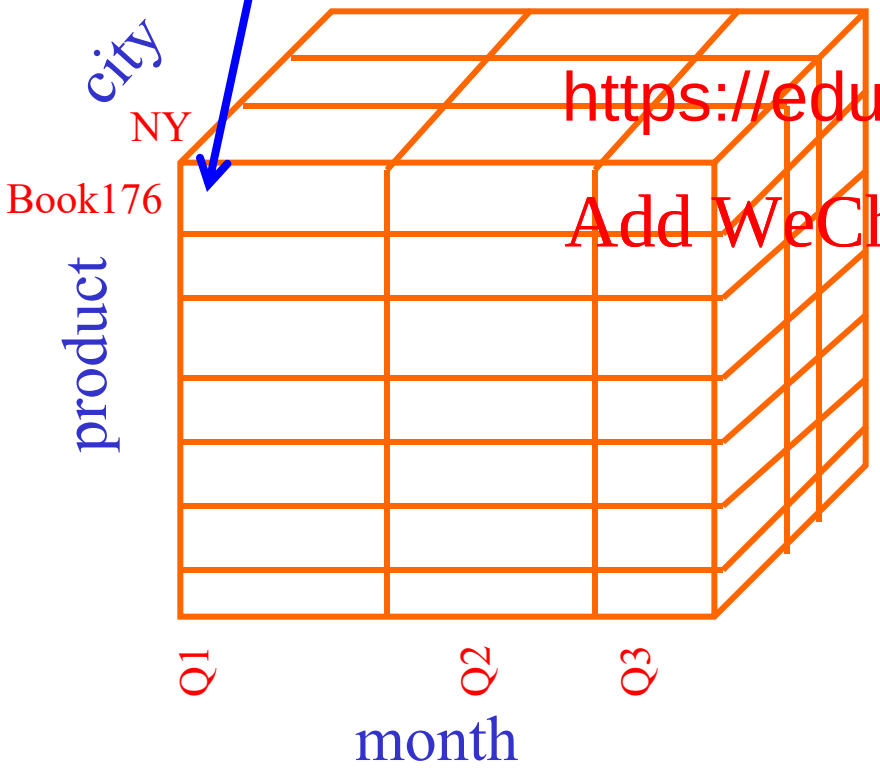
Q: What's the analogy in the Relational Model?

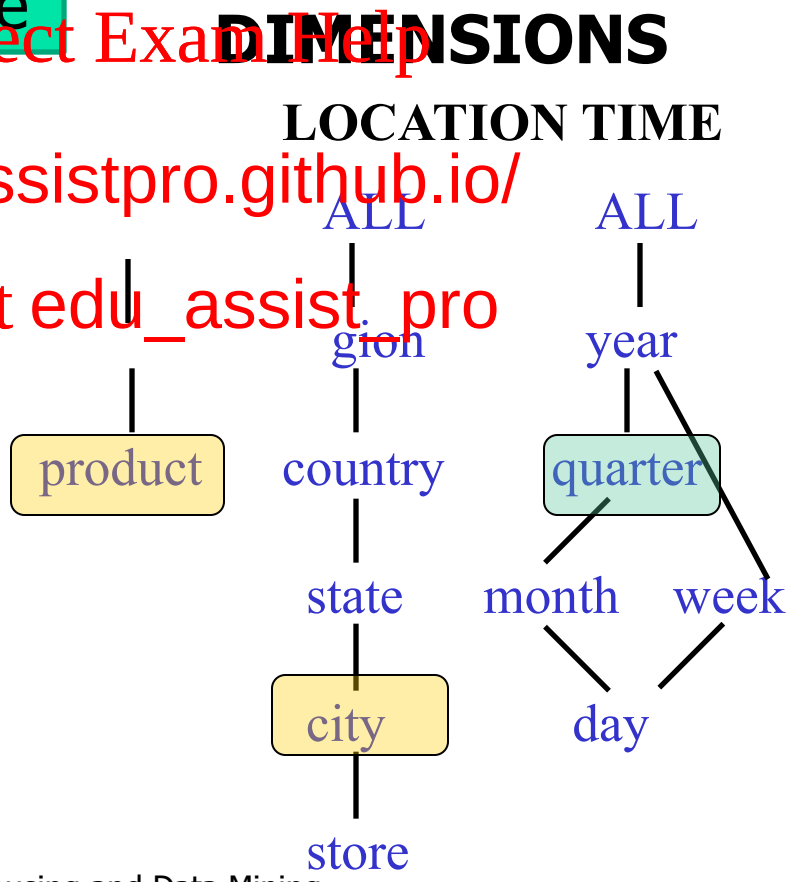
Common OLAP Operations

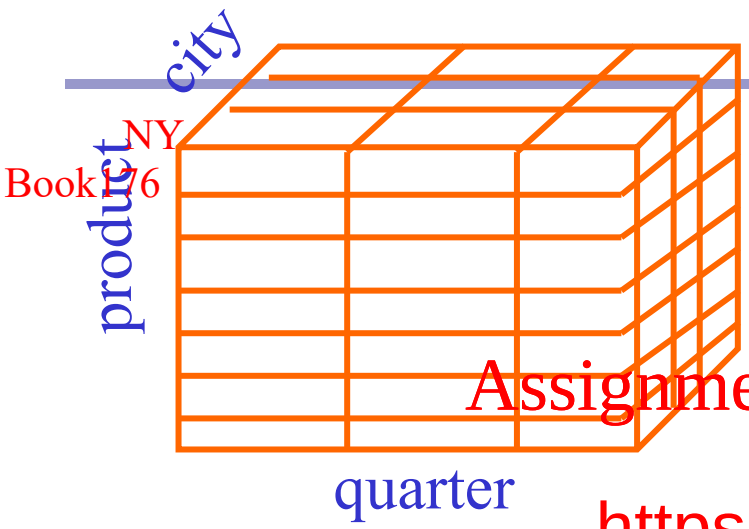
- **Roll-up**: move up the hierarchy

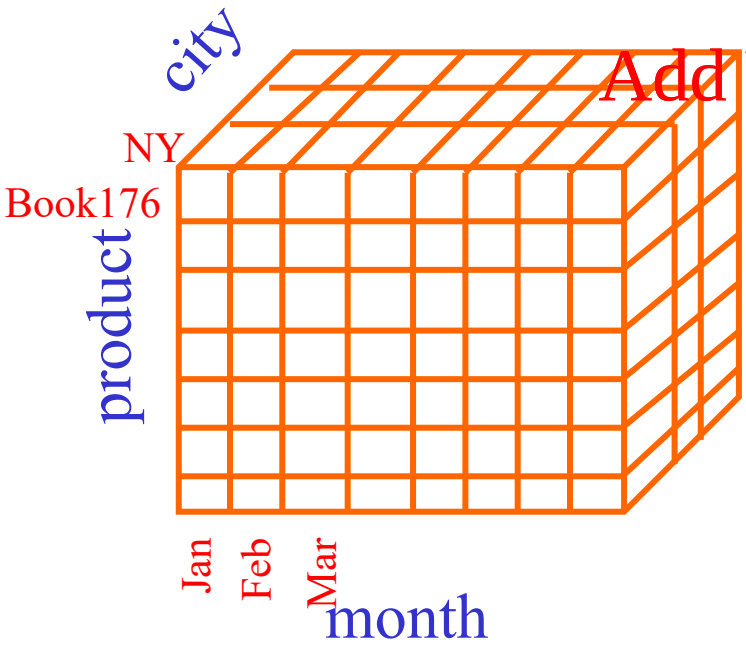
Q: what should be its value?

Sales of book176 in NY in Q1 here



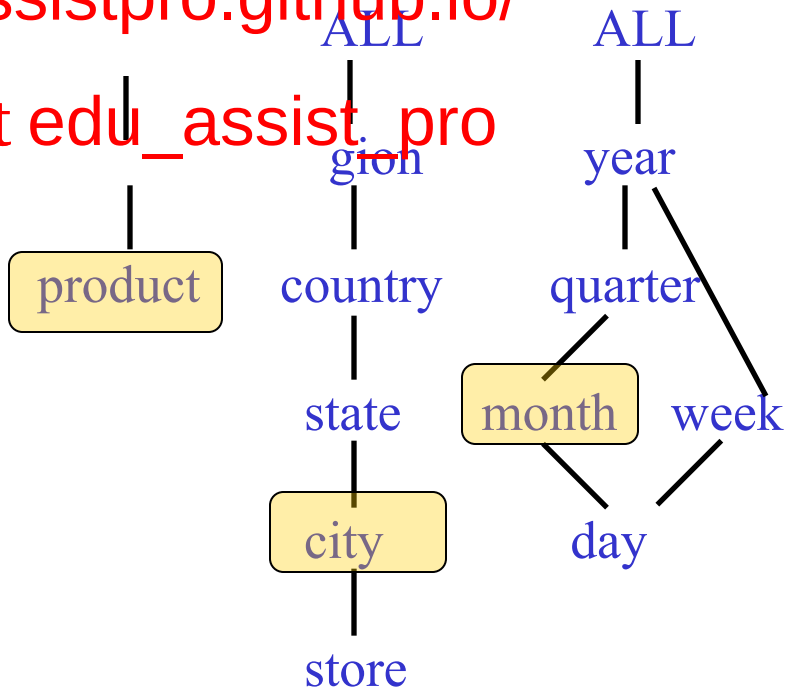






DIMENSIONS

LOCATION TIME



Data Cube Measures: Three Categories

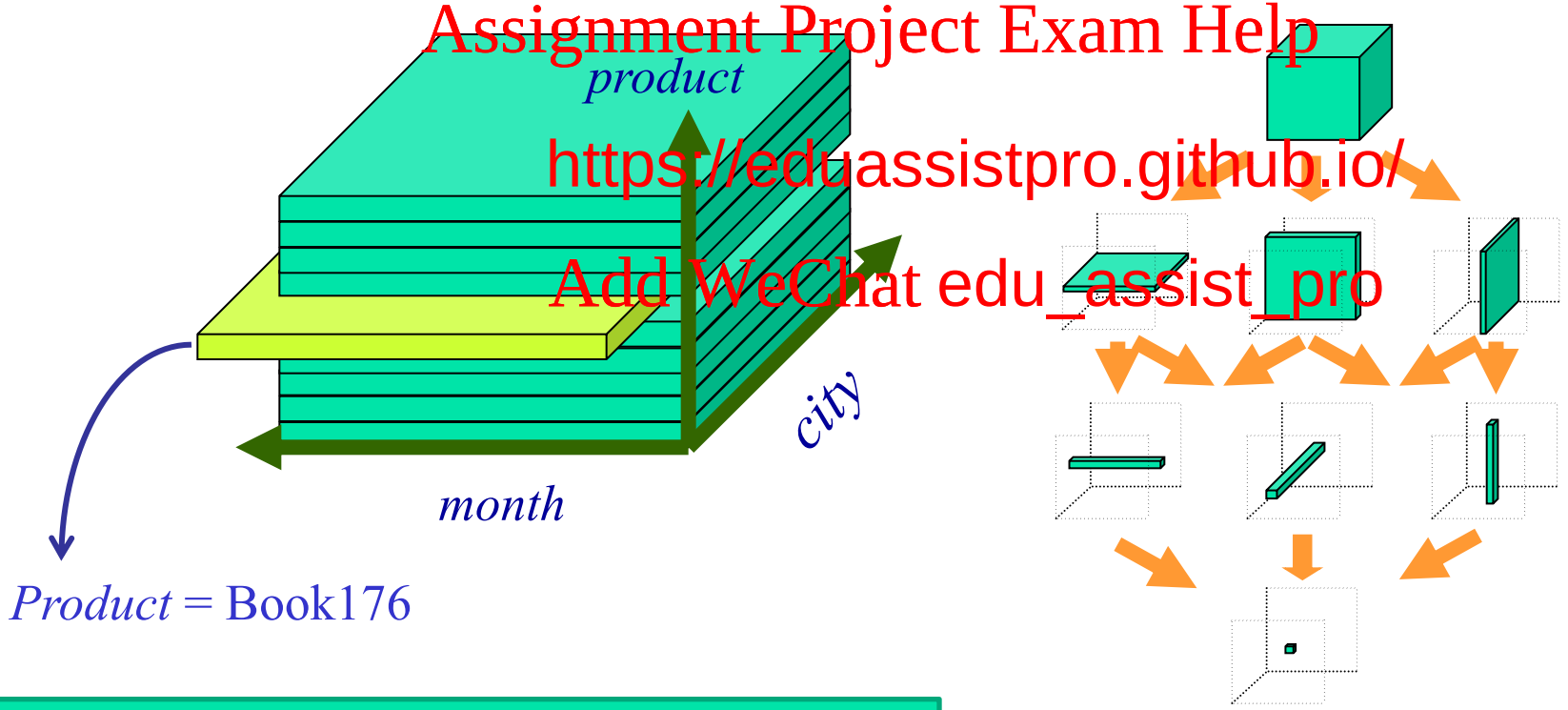
- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()
- Algebraic: if it c [illegible] algebraic function with *M* arguments (where *M* is [illegible] integer), each of which is obtained by applying [illegible] e aggregate function
  - E.g., avg(), min_N(), standard_deviation()
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

Common OLAP Operations

- **Drill-down:** move down the hierarchy
  - more fine-grained aggregation

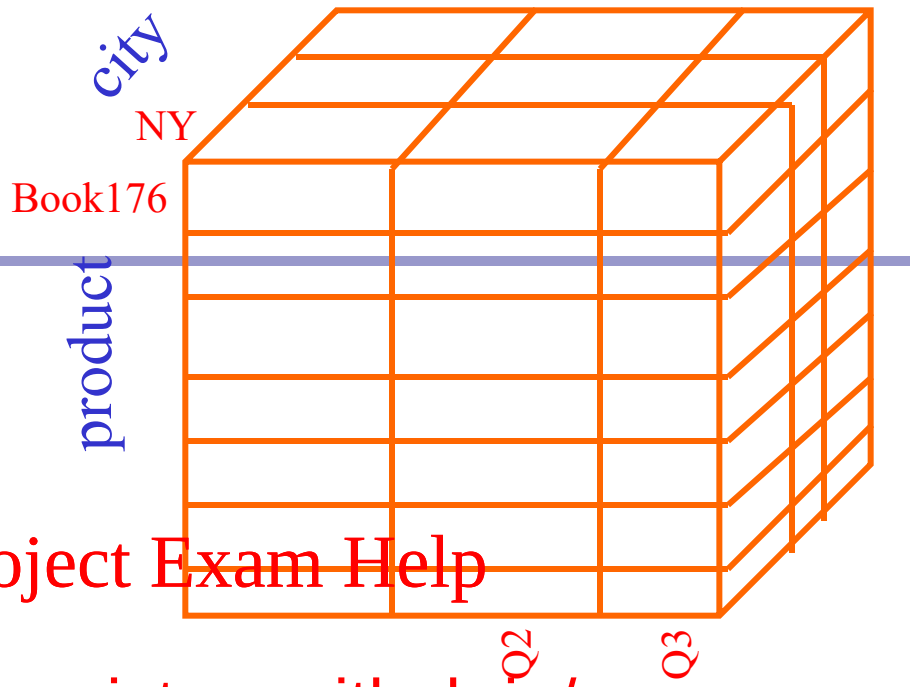
Slice and Dice Queries

- Slice and Dice: select and project on one or more dimension values



The output cube has smaller dimensionality than the input cube

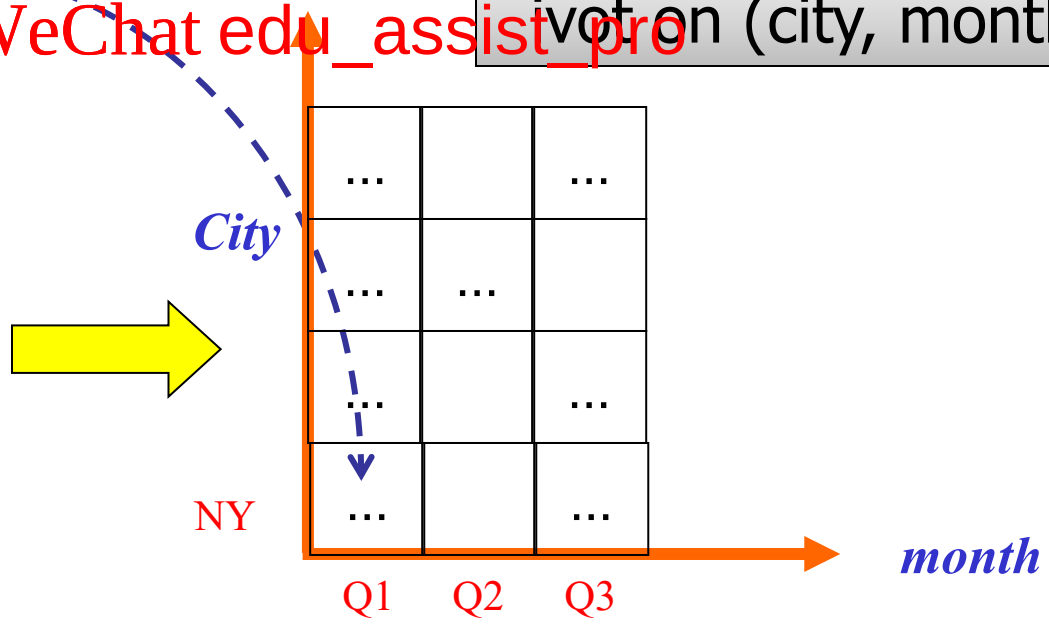
Pivoting



- Pivoting: aggregate on selected dimensions
  - usually 2 dims (cross-tabulation)

Sales (of all produc
NY in Q1
=sum(???)

pivot on (city, month)



A Reflective Pause

- Let's review the definition of data cubes again.

- **Key message**:
  - Disentangle the "object" from its "representation" or "implementation"

Modeling Exercise 1: Monthly Phone Service Billing

Theme: analyze the income/revenue of Telstra

Solution

- FACT

- MEASURE

- DIMENSIONS

- The Logical Model

Logical Models

- Two main approaches:
  - Using relational DB technology:
    - Star schema, Snowflake schema, Fact constellation
  - Using mult logy:
    - Just as m e

Universal Schema ➔ Star Schema

- Many data warehouses adopt a star schema to represent the multidimensional model
- Each dimension is represented by a dimension-table
  - LOCATION (location_key, store, street_address, city, state, country, region
  - dimension tabl
- Transactions are described th t-table
  - each tuple consists of a logical p of the dimension-tables (foreign-key) and a list of measures (e.g. sales $$$)

The universal schema for supermarket

Store	City	State	Prod	Brand	Category	$Sold	#Sold	Cost
S136	Syd	NSW	76Ha	Nestle	Biscuit	40	10	18
S173	Melb	Vic	76Ha	Nestle	Biscuit	20	5	11

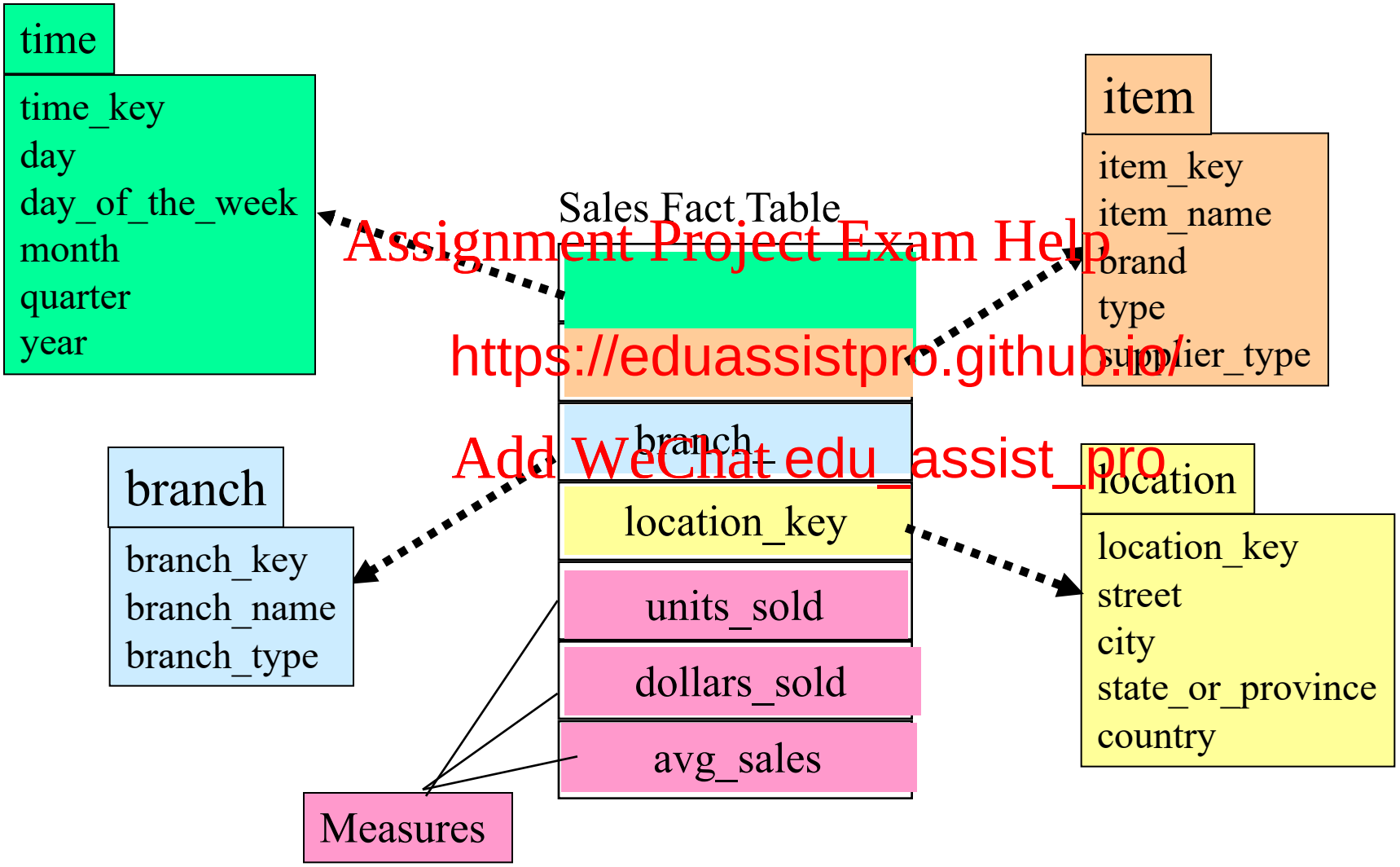
The Star Schema

Think why:
(1) Denormalized once from the universal schema
(2) Controlled redundancy

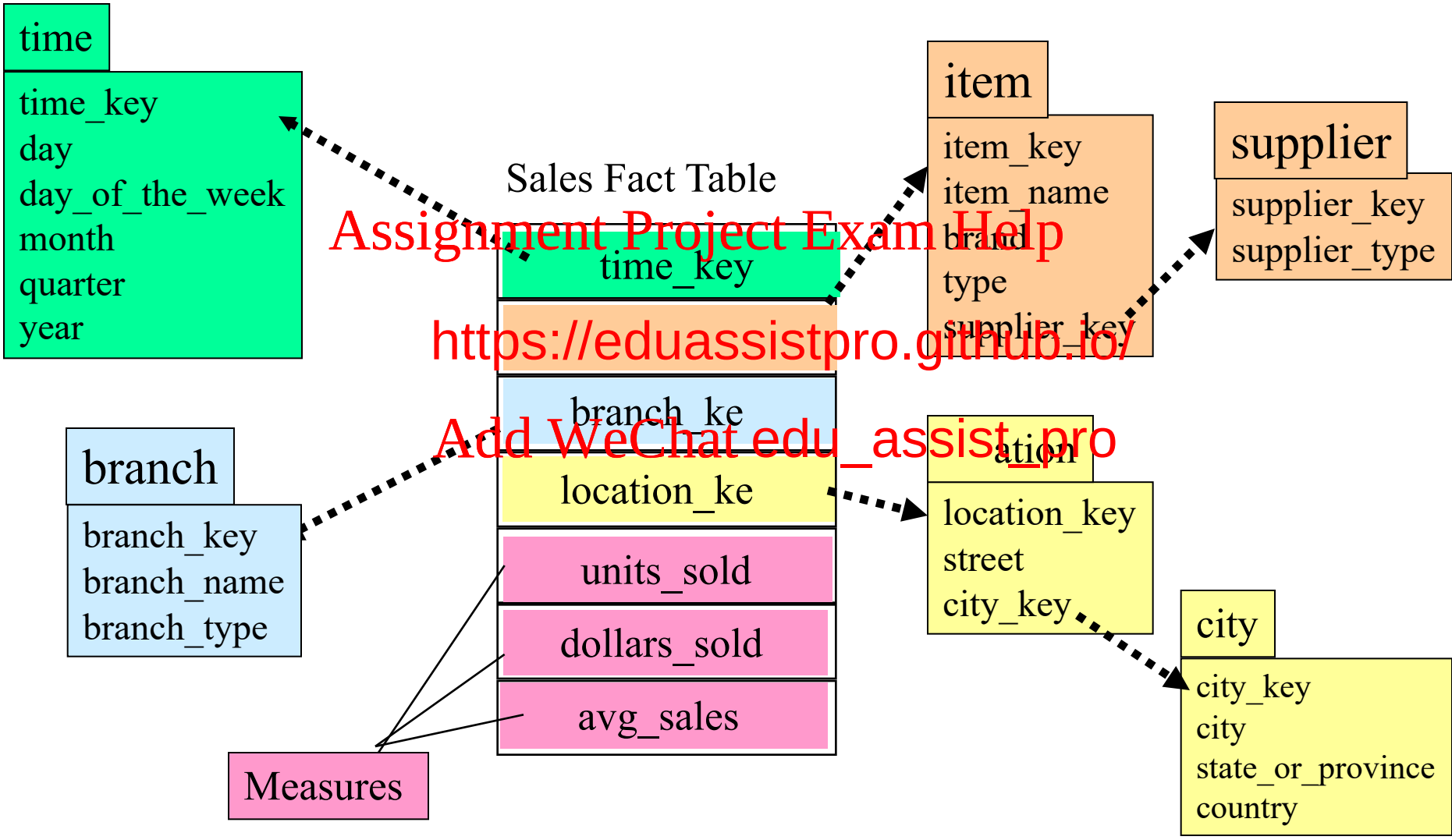
Typical Models for Data Warehouses

- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake s star schema where some dimensional hi normalized into a set of smaller dimension ta ng a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

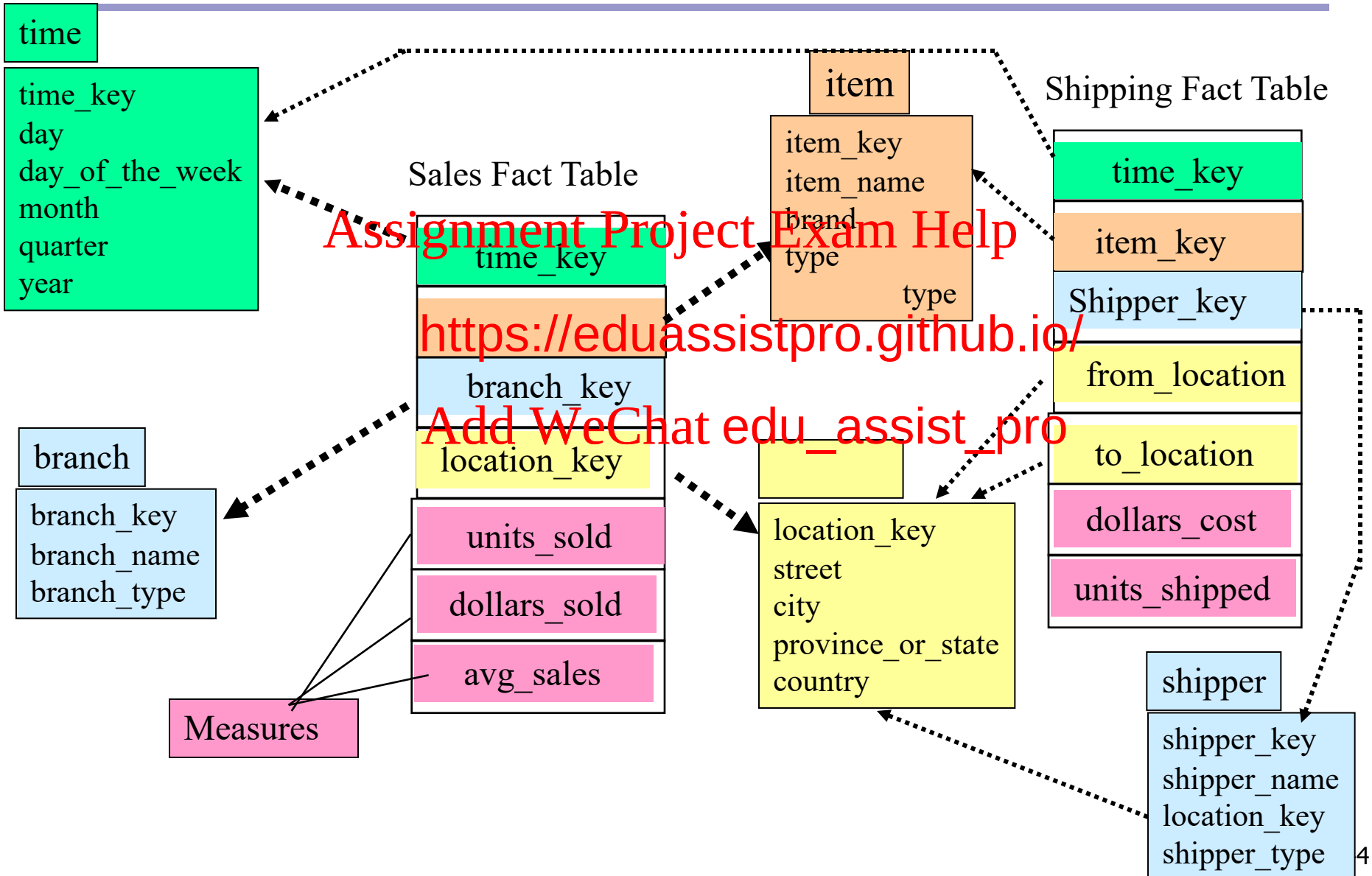
Example of Star Schema



Example of Snowflake Schema



Example of Fact Constellation



Advantages of Star Schema

- Facts and dimensions are clearly depicted
  - dimension tables are relatively static, data is loaded (append mostly) into fact table(s)
  - easy to co queries)

"Find total sales per product categ res in Europe"


```
SELECT PRODUCT.category, SUM(SALES.amount)
FROM   SALES, PRODUCT,LOCATION
WHERE  SALES.product_key = PRODUCT.product_key
AND    SALES.location_key = LOCATION.location_key
AND    LOCATION.region="Europe"
GROUP BY PRODUCT.category
```

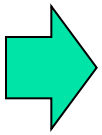
Operations: Slice (Loc.Region.Europe) + Pivot (Prod.category)

- Query Language

Query Language

- Two approaches:
  - Using relational DB technology: SQL (with extensions such as CUBE/PIVOT/UNPIVOT)
  - Using multid


```
SELECT PRODUCT.categ
SUM(SALES.amount)
FROM    SALES, PRODUCT,LOCATION
WHERE   SALES.product_key =
PRODUCT.product_key
AND     SALES.location_key =
LOCATION.location_key
AND     LOCATION.region="Europe"
GROUP BY PRODUCT.category
```



```
{[category]} on ROWS,
{[MEASURES].[amount]} on COLUMNS
FROM    [SALES]
WHERE   ([LOCATION].[region].[Europe])
```

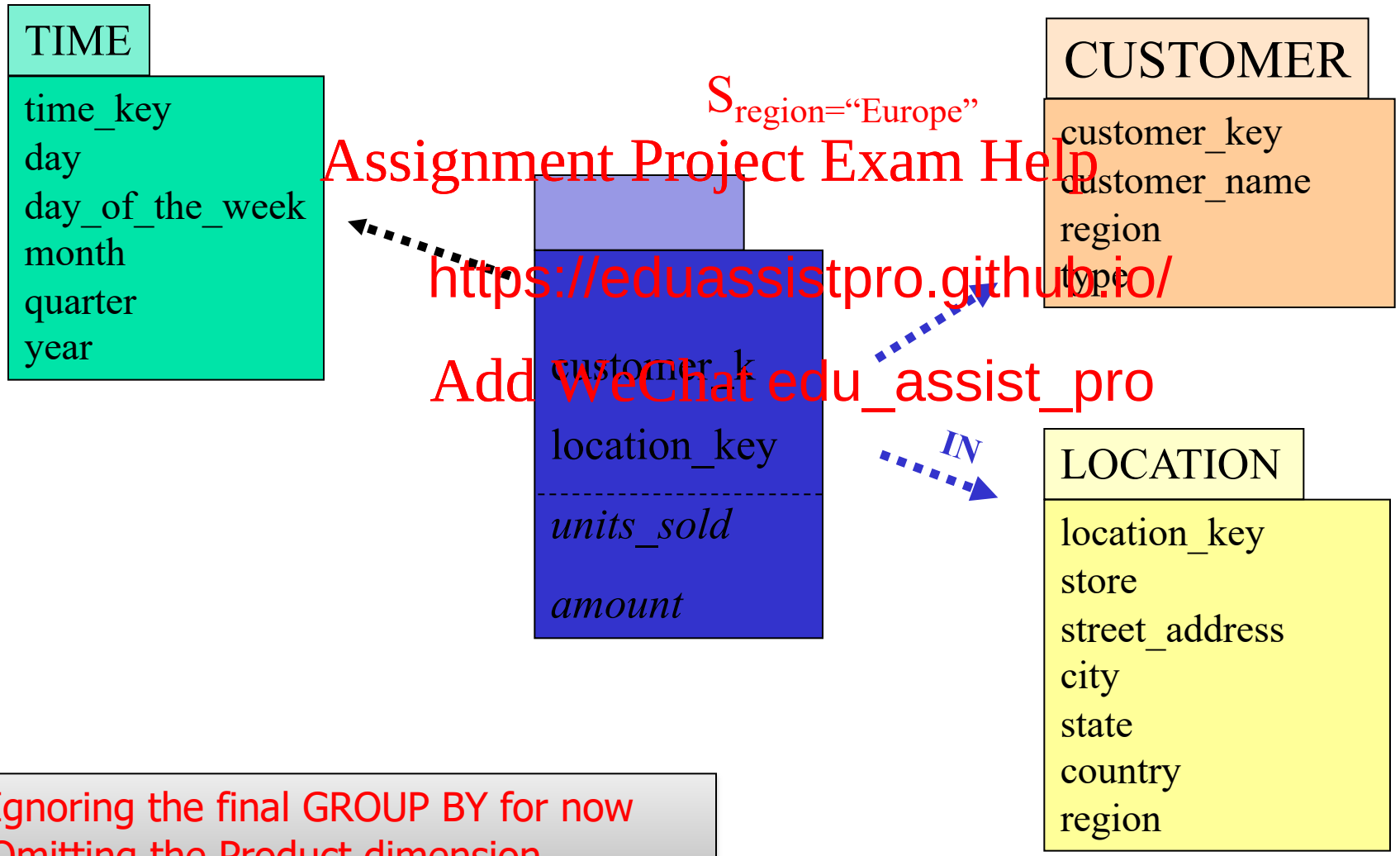
Operations: Slice (Loc.Region.Europe) + Pivot (Prod.category, Measures.amnt)

- Physical Model + Query Processing Techniques

Physical Model + Query Processing Techniques

- Two main approaches:
  - Using relational DB technology: ROLAP
  - Using multidimensional technology: MOLAP
- Hybrid: HOL
  - Base cuboid: ROLAP
  - Other cuboids: MOLAP

Q1: Selection on low-cardinality attributes



- Ignoring the final GROUP BY for now
- Omitting the Product dimension

Indexing OLAP Data: Bitmap Index

(1) BI on dimension tables

- Index on an attribute (column) with low distinct values
- Each distinct values, v, is associated with a n-bit vector (n = #rows)
  - The i-th bit is set if the i-th row of the table has the value v for the inde
- Multiple BIs ca enable optimized scan of the table

Custom

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

BI on Customer.Region

v	bitmap
Asia	1 0 1 0 0
Europe	0 1 0 0 1
America	0 0 0 1 0

Indexing OLAP Data: Bitmap Index /2

(1) Bitmap join index (BI on Fact Table Joined with Dimension tables)

- Conceptually, perform a join, map each dimension value to the bitmap of corresponding fact table rows.


```
-- ORACLE SYNTAX –

CREATE BITMAP INDEX sales_cust_region_bjix
 ON    sales(customer.cust_region)
FROM sales, customer
WHERE sales.cust_id = customers.cust_id;
```

Indexing OLAP Data: Bitmap Index /3

Sales

time	customer	loc	Sale
101	C1	100	1
173	C1	200	2
208	C2	10	
863	C3	20	
991	C1	100	8
1001	C2	200	13
1966	C4	100	21
2017	C5	200	34

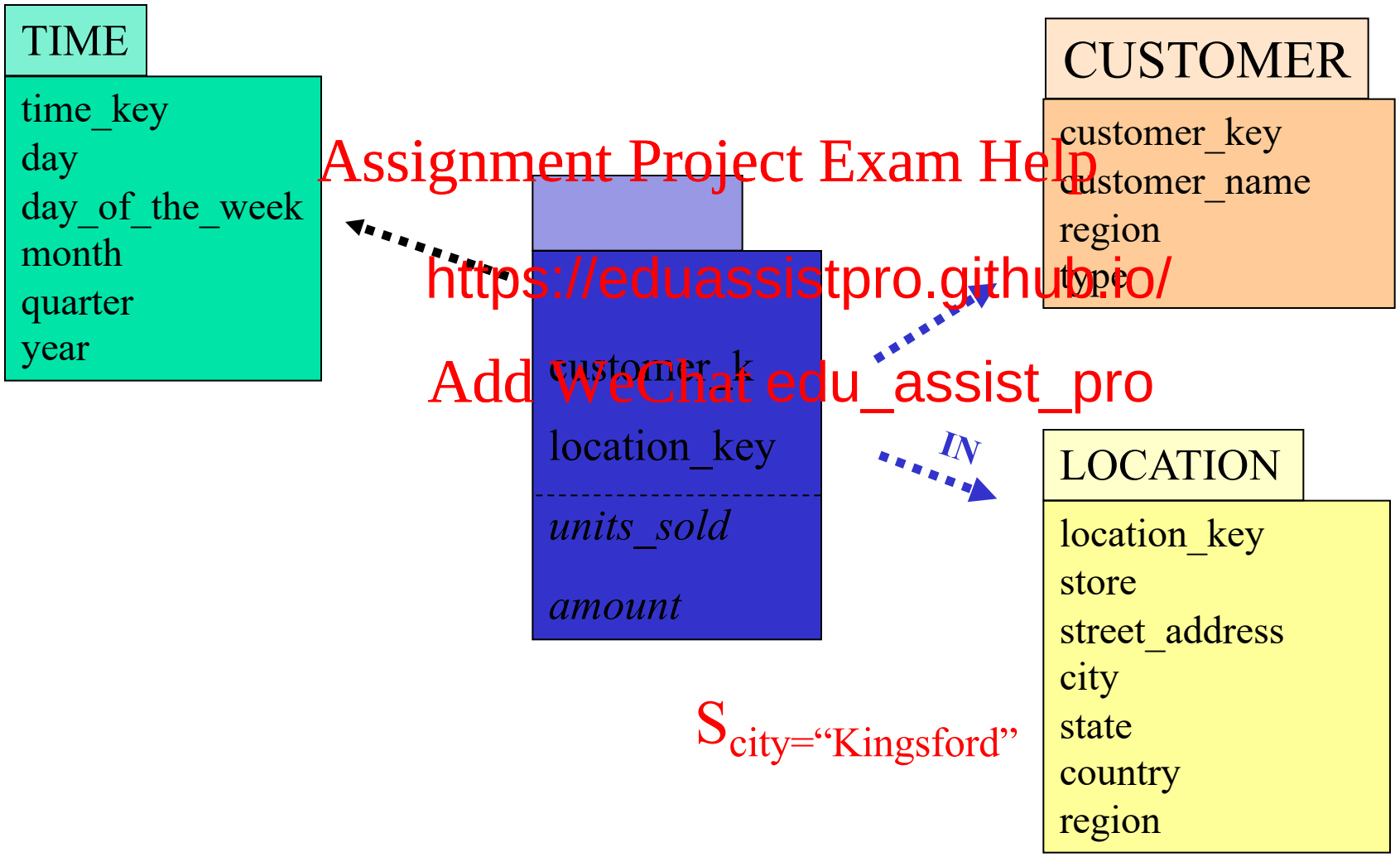
Customer

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
	sia	Dealer
	merica	Retail
	pe	Dealer

BI on Sales(Customer.Region)

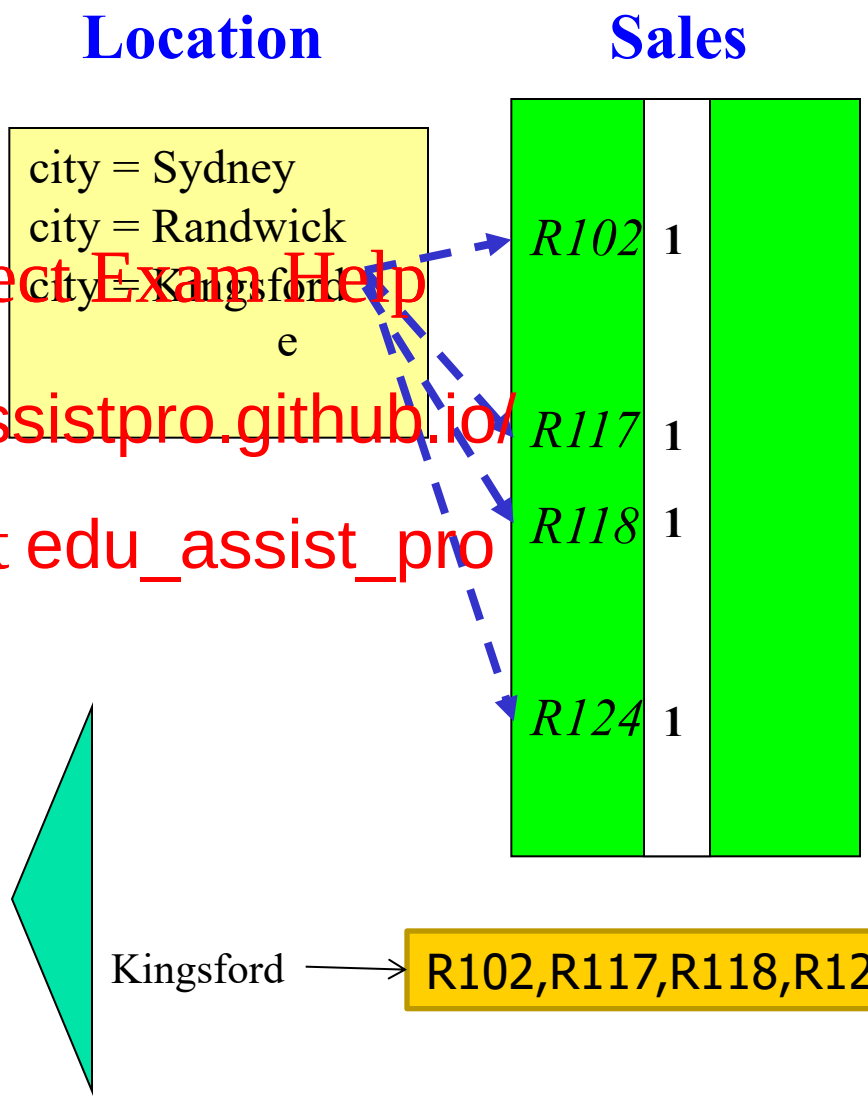
v	bitmap
Asia	11011000
Europe	00100101
America	00000010

Q2: Selection on high-cardinality attributes

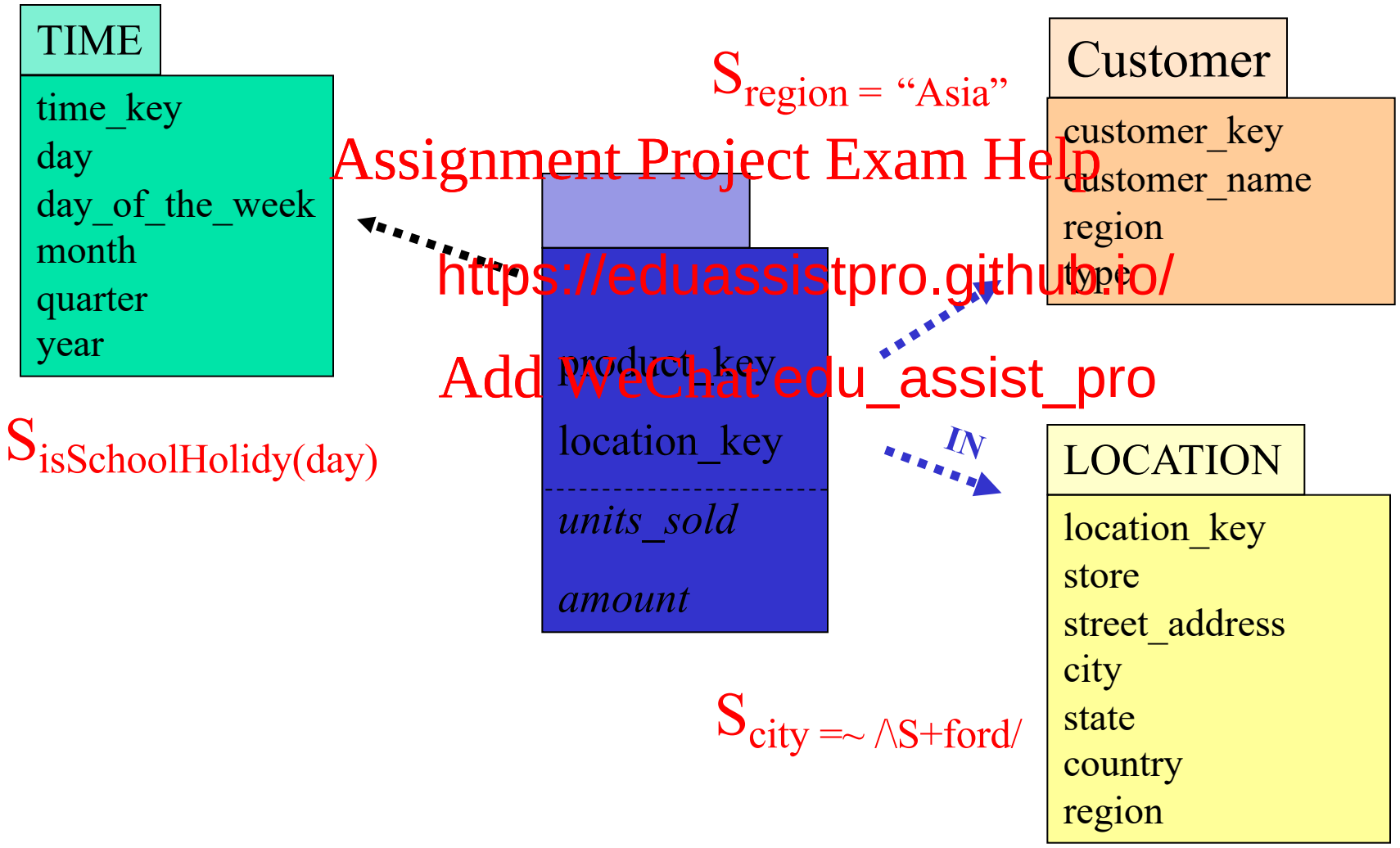


Indexing OLAP Data: Join Indices

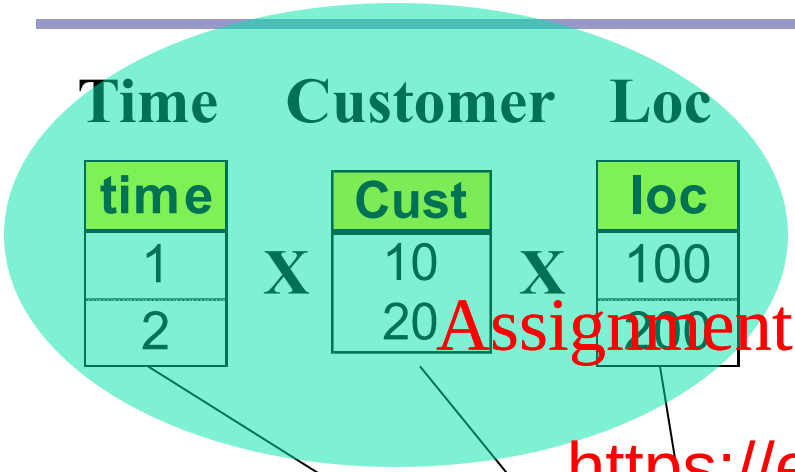
- Join index relates the values of the dimensions of a star schema to rows in the fact table.
  - a join index on *city* maintains for city a list of R the tuples recording the sales in the city
- Join indices can span multiple dimensions OR
  - can be implemented as bitmap-indexes (per dimension)
  - use bit-op for multiple-joins



Q3: Arbitrary selections on Dimensions



Star Query and Star Join (Cont.)



Usually only part of the dim tables because of the selection predicates

Sales

millions

time	cust	loc	sold
	10	100	7
	10	150	13
	20	150	2
	20	200	16
	…	…	…
1000	2000	500	86

thousands of tuples

time	cust	loc
1	10	100
1	10	200
1	20	100
1	20	200
2	10	100
2	10	200
2	20	100
2	20	200

$$\text{Sales} \bowtie \sigma_1(\text{Time}) \bowtie \sigma_2(\text{Cust}) \bowtie \sigma_3(\text{Loc}) \rightarrow \text{Sales} \bowtie (\sigma_1(\text{Time}) \times \sigma_2(\text{Cust}) \times \sigma_3(\text{Loc}))$$

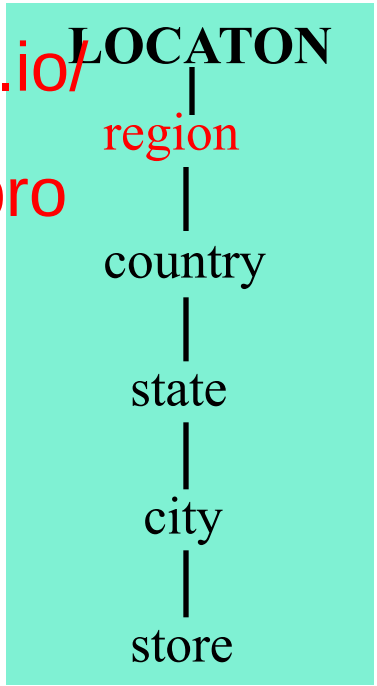
Q4: Coarse-grain Aggregations

- *"Find total sales per customer type in our stores in Europe"*
  - Join-index will prune ¾ of the data (uniform sales), but the remaining ¼ is still large (several millions transactions)
    - Index is un
- High-level aggregations are e !

⇒Long Query Response Ti

⇒Pre-computation is necessary

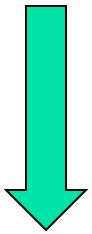
⇒Pre-computation is most beneficial



Cuboids = GROUP BYs

- Multidimensional aggregation = selection on corresponding cuboid

$GB_{(type,\ city)}(\ Sales \bowtie \sigma_1(Time) \bowtie \sigma_2(Customer) \bowtie \sigma_3(Loc))\ )$



σ_1 selects some Brands,

σ_3 selects some Cities

$GB_{(type,\ city)}(\ \sigma_{1,2,3}(\mathbf{Cuboid}(\mathit{Year}, \mathit{Type}, \mathit{City}))\ )$

- Materialize some/all of the cuboids
  - A complex decision involving cuboid sizes, query workload, and physical organization

Two Issues

- How to store the materialized cuboids?
- How to compute the cuboids efficiently?

CUBE BY in ROLAP

Sales		Product				
		1	2	3	4	ALL
Store	1	454	-	-	925	1379
	2	468	800	-	-	1268
	3	296	-	240	-	536
	4	652	-	540		
	ALL	1870	800	780		

Store	Product_key	sum(amout)
1	1	454
1	4	925
2	1	468
2	2	800
3	1	296
3	3	240
4	1	625
	3	240
	4	745
	ALL	1379
	ALL	1268
	ALL	536
	ALL	1937
ALL	1	1870
ALL	2	800
ALL	3	780
ALL	4	1670
ALL	ALL	5120

4 Group-bys here:
(store,product)
(store)
(product)
()

- Need to write 4 queries!!!
- Compute them independently

SELECT LOCATION.store, SALES.product_key, SUM (amount)

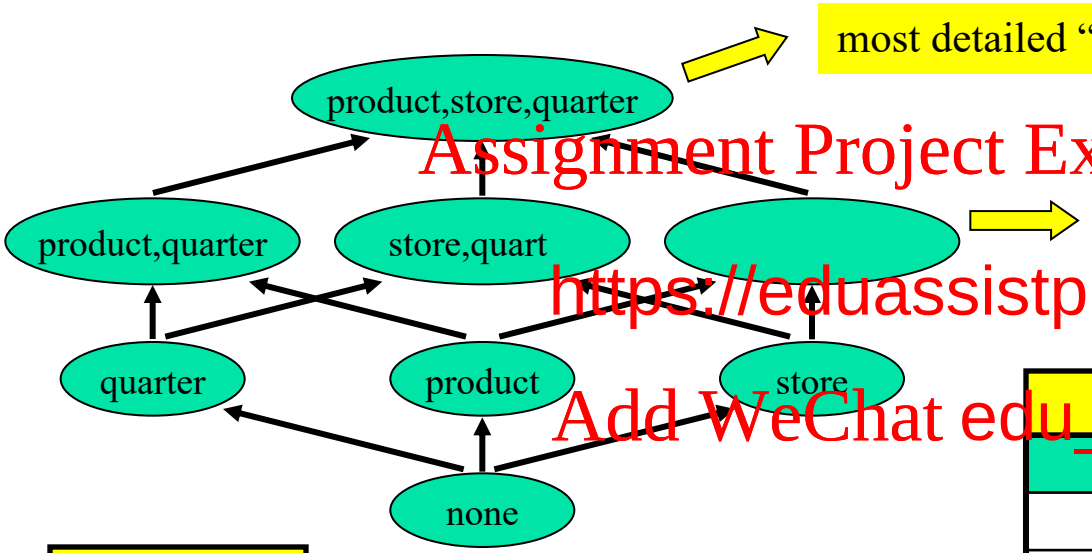
FROM SALES, LOCATION

WHERE SALES.location_key=LOCATION.location_key

CUBE BY SALES.product_key, LOCATION.store

Top-down Approach

- Model dependencies among the aggregates:



Cuboid AB

A	B	M
1	1	10
1	3	20
1	2	30
1	1	40
2	4	50

	B	M
1	(all)	???
2	(all)	???

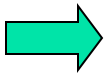
Cuboid B

A	B	M
(all)	1	???
(all)	2	???
(all)	3	???
(all)	4	???

Bottom-Up Approach (BUC)

- BUC (Beyer & Ramakrishnan, SIGMOD'99)
- Ideas
  - Compute th
    bottom up
  - Divide-and-conquer
- A simpler recursive version:
  - BUC-SR

A	B	...
1	1	...
1	3	...
1	2	...
1	1	...
2	...	...



...

Understanding Recursion /1

- Powerful computing/problem-solving techniques
- Examples
  - Factorial:
    - f(n) = 1, if n = 1
    - f(n) = f(n-1) * n, if n ≥ 1
  - Quick sort:
    - Sort([x]) = [x]
    - Sort([x1, …, *pivot*, … xn]) = sort[ys] ++ sort[zs]), where

      ys = [x | x ***in*** xi, x ≤ *pivot*]

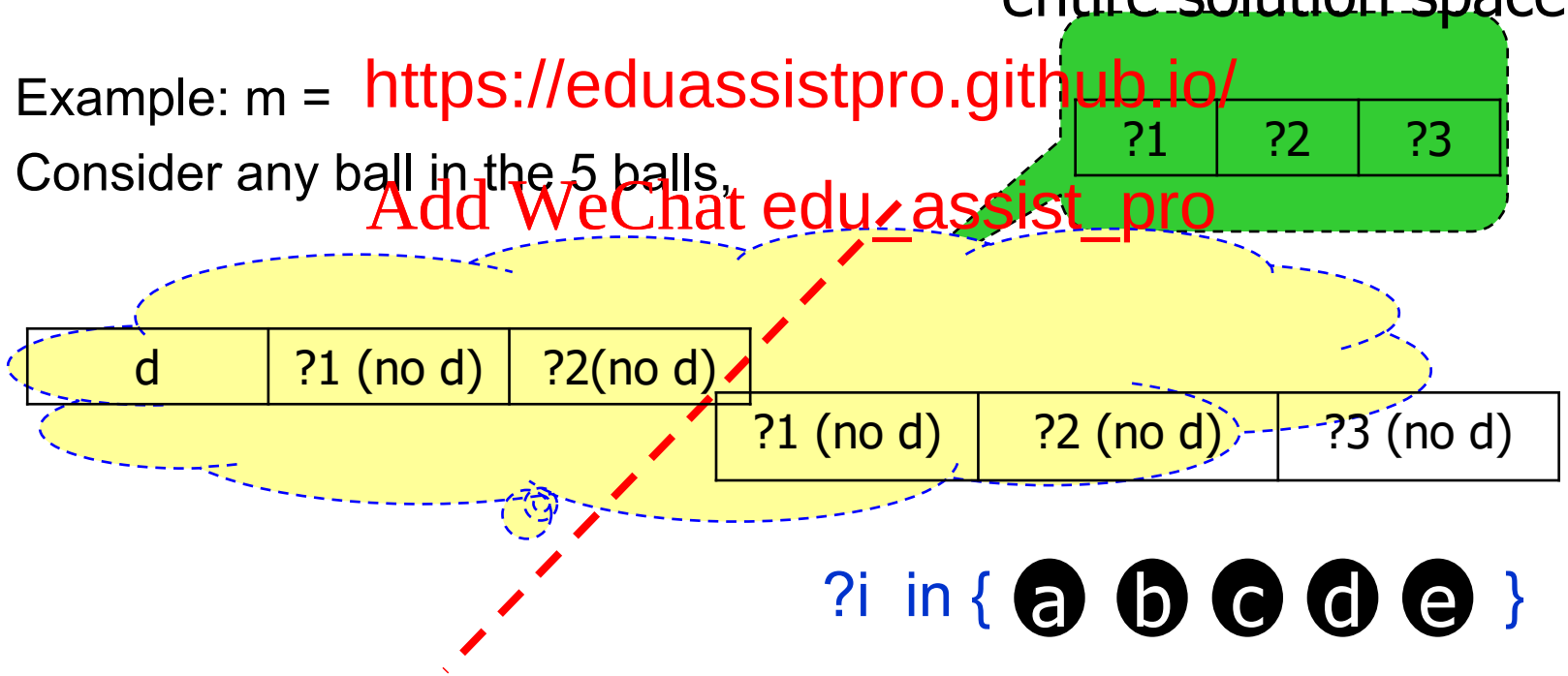
      zs = [x | x **<-** xi, x > *pivot*]

f(0) = 0! = ???

List comprehension in Haskell or python

Understanding Recursion /2

- Let C(n, m) be the number of ways to select m balls from n numbered balls
- Show that C(n, m) = C(n-1, m-1) + C(n-1, m)
- Example: m =
- Consider any ball in the 5 balls,

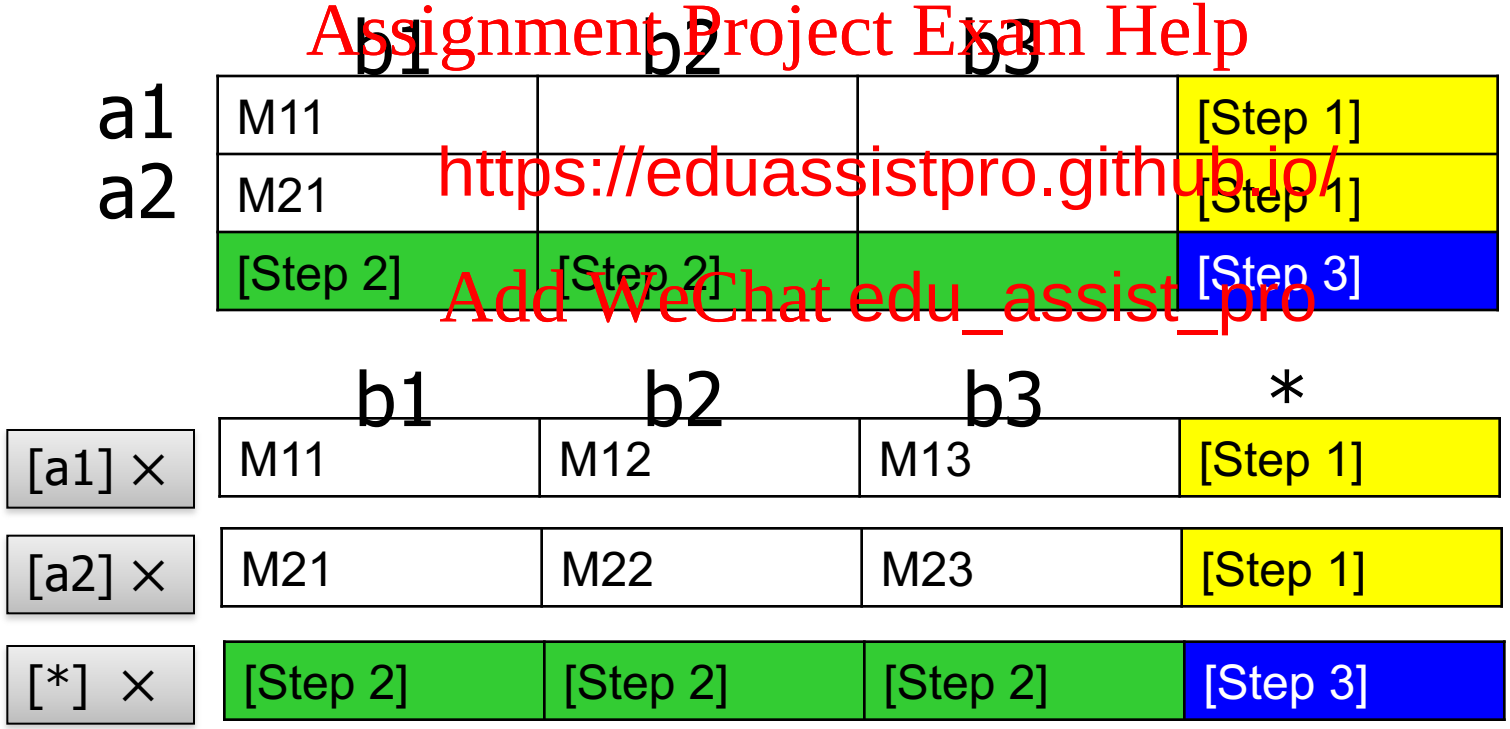


Key Points

- Sub-problems need to be "**smaller**", so that a simple/trivial boundary case can be reached
- Divide-and
  - There ma ntire solution space can be divided int sub-spaces, each of which can be co **cursively**.

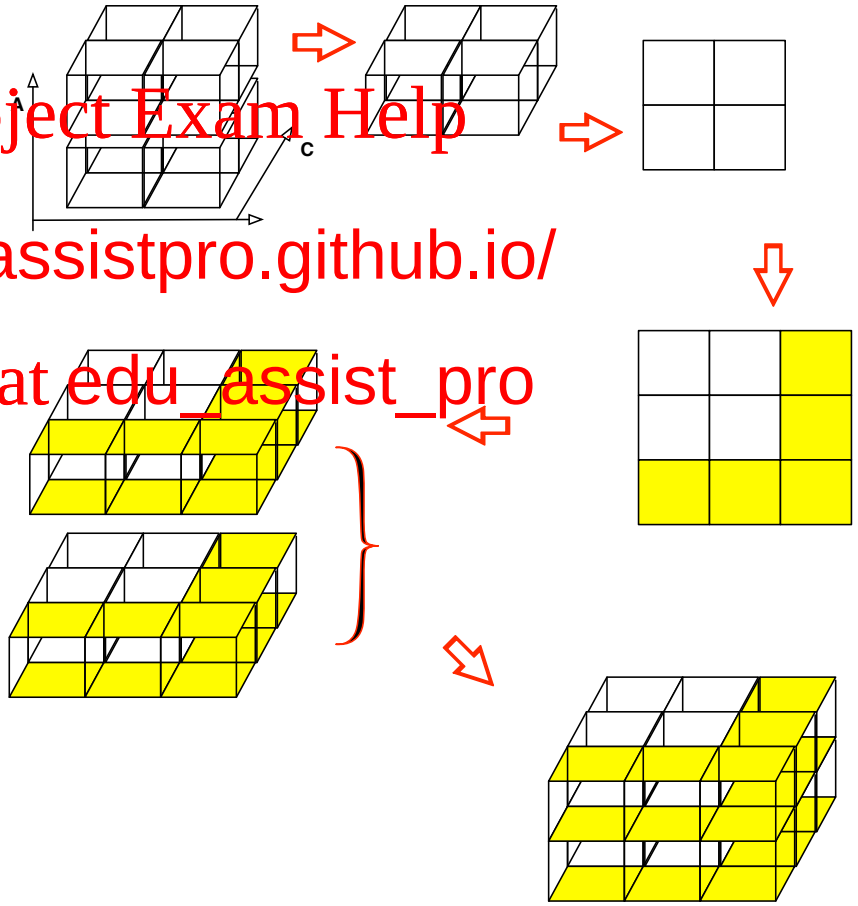
Geometric Intuition /1

- Reduce Cube(in 2D) to Cube(in 1D)

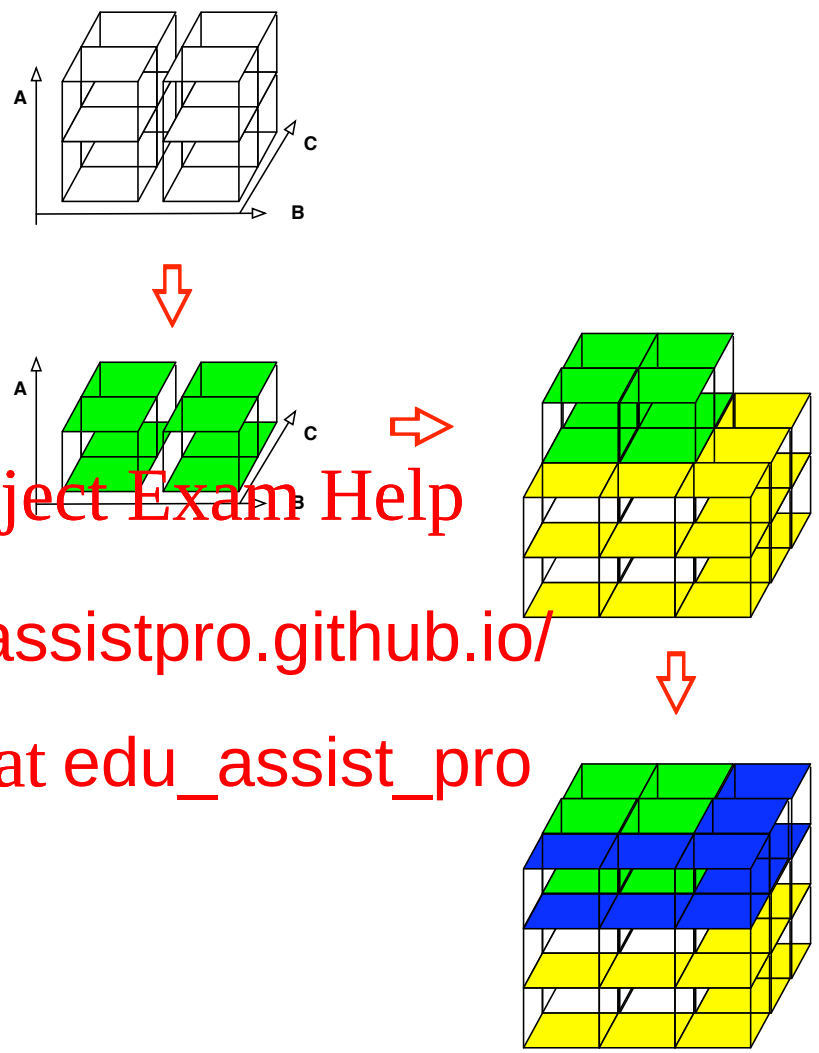


Geometric Intuition /2

- Reduce Cube(in 3D) to Cube(in 2D)



Geometric Intuition /3



- Reduce Cube(in 3D) to Cube(in 2D)

Algebraic Derivation

- How to compute *n*-dim cube on (*n*+1)-dim base cuboid (array)?
  - What do output tuples look like?
- How to comp on (*n*+1)-dim base cuboid (
  - What else do we need?

[{r1-r5}, **ABC**]

[{r1-r5}, **BC**]

r1
r2
r3
r4
r5

A	B	C	M
1	1	1	10
1	1	2	20
1	2	1	30
1	3	1	40
2	1	1	50

You need to fix the incomplete output part of the algorithm!

BUC-SR (Simple Recursion)*

- BUC-SR(data, dims)
  - If (dims is empty)
    - Output (su
  - Else
    - Dims = [dim1, rest_of_di
    - For each distinct value v of dim1
      - slice_v = slice of data on "dim1 = v"
      - BUC-SR(slice_v, rest_of_dims)
    - data' = Project(data, rest_of_dims)
    - BUC-SR(data', rest_of_dims)

Boundary case:
data is essentially a list of measure values

General case:
1)Slice on dim1. Call BUC-SR recursively for each slice

2)Project out dim1, and call BUC-SR on it recursively

Example

Input | Internal Output | Output

	1	2	3	*
1	30	30	40	100
2	50			50
*	80	30	40	150

[{r1-r4}, B]

B	M
1	10
1	20
2	30
3	40

B	M
1	30
2	30
3	40
*	100

A	B	M
1	1	30
1	2	30
1	3	40
1	*	100

[{r1-r5}, AB]

	A	B	M
r1	1	1	10
r2	1	1	20
r3	1	2	30
r4	1	3	40
r5	2	1	50

[{r1-r5}, B]

B	M
1	50

B	M
	50
	50

A	B	M
2	1	50
2	*	50

[{r1'-r5'}, B]

B	M
1	**80**
2	30
3	40

B	M
1	80
2	30
3	40
*	150

A	B	M
*	1	80
*	2	30
*	3	40
*	*	150

Try a 3D-Cube by Yourself

[{r1-r5}, ABC]

	A	B	C	M
r1	1	1	1	10
r2	1	1	2	20
r3	1	2	1	30
r4	1	3	1	40
r5	2	1	1	50

MOLAP

- (Sparse) array-based multidimensional storage engine
- Pros:
  - small size
  - fast in ind
- Cons:
  - scalability
  - conversion from relational data

Step 2: An **injective** map from cell to offset

Multidimensional Array

f(time, item) = 4*time + item

time	item	dollars_sold
Q1	home entertainment	605
Q2	home entertainment	680
Q3	home entertainment	812
Q4	home entertainment	
Q1	computer	
Q2	computer	952
Q3	computer	1023
Q4	computer	1038
Q1	phone	14
Q2	phone	31
Q3	phone	30
Q4	phone	38
Q1	security	400
Q2	security	512
Q3	security	501
Q4	security	580

Step 1

Mappings

time	*value*
Q3	2
Q4	

item	*value*
home entertainment	0
computer	1
phone	2
security	3

time	item	dollars_sold	offset
0	0	605	0
1	0	680	4
2	0	812	8
3	0	927	12
0	1	825	1
1	1	952	5
	1	1023	9
	1	1038	13
0	2	14	2
1	2	31	6
2	2	30	10
3	2	38	14
0	3	400	3
1	3	512	7
2	3	501	11
3	3	580	15

Multidimensional Array

Step 3: If dense, only need to store sorted slots

offset	dollars_sold
0	605
1	825
2	14
3	400
4	680
5	952
6	31
7	512
8	812
9	1023
10	30
11	501
12	927
13	1038
14	38
15	580

Dense MD array
605
825
14
400
680
952
31
512
812
1023
30
501
927
1038
38
580

- Think: how to decode a slot?

The Sparse Case

Step 2: An **injective** map from cell to offset

f(time, item) = 4*time + item

time	item	dollars_sold
Q1	home entertainment	605
/* same table but with many rows deleted to make it sparse */		
Q4	security	580

Step 1

Mappings

time	value
Q3	2
Q4	

item	value
home entertainment	0
computer	1
phone	2
security	3

time	item	dollars_sold	offset
0	0	605	0
/* same table but with many rows deleted to e it sparse */			
3	3	580	15

Multidimensional Array

Choice 1

offset	dollars_sold
0	605
15	580

Choice 2

Dense MD array
605
-
-
-
-
-
-
-
-
-
-
-
-
-
-
580

- Think: how to decode a slot?
- Multidimensional array is typically sparse
  - Could use further re e
- Space usage:
  - (d+1)*n*4 vs 2*n*4
- HOLAP:
  - Store all non-base cuboid in MD array
  - Assign a value for ALL