Data Warehousing and Data Mining

Assignment Project Exam Help

— L2: Data Warehousing and OLAP —

https://powcoder.com

Add WeChat powcoder

1

- Why and What are Data Warehouses?

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Data Analysis Problems

- The same data found in many different systems
  - Example: customer data across different departments
  - The same concept is defined differently
- Heterogeneous sources
  - Relational DBMS, OnLine Transaction 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 is bad
  - Missing data, imprecise data, different 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 information processing by providing a solid platform of consolidated, historical data for analysis.

- "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

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

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.
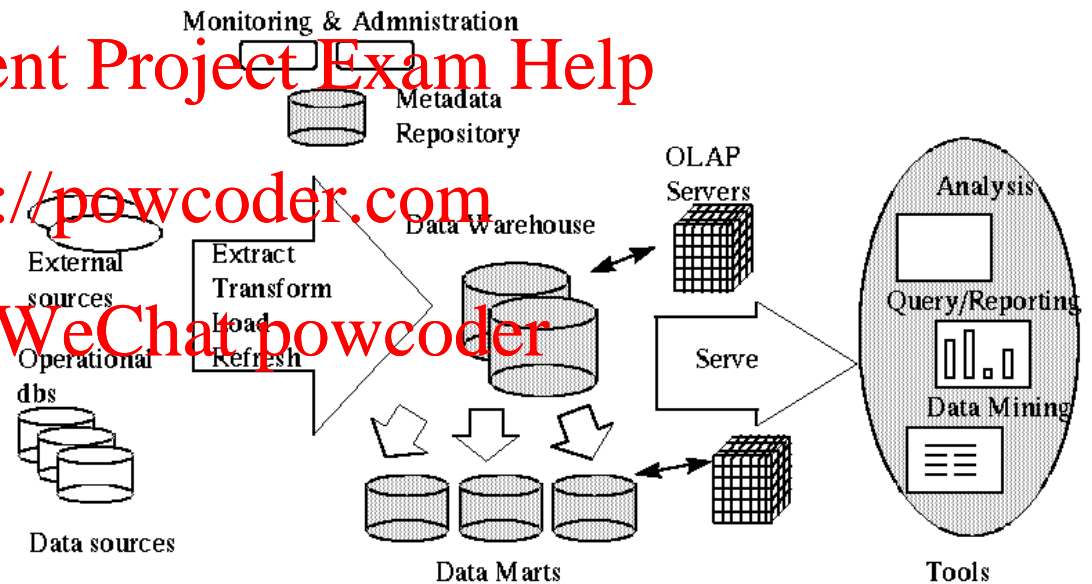
Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems.

  - Operational database: current value data.

  - Data warehouse data: provide 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".
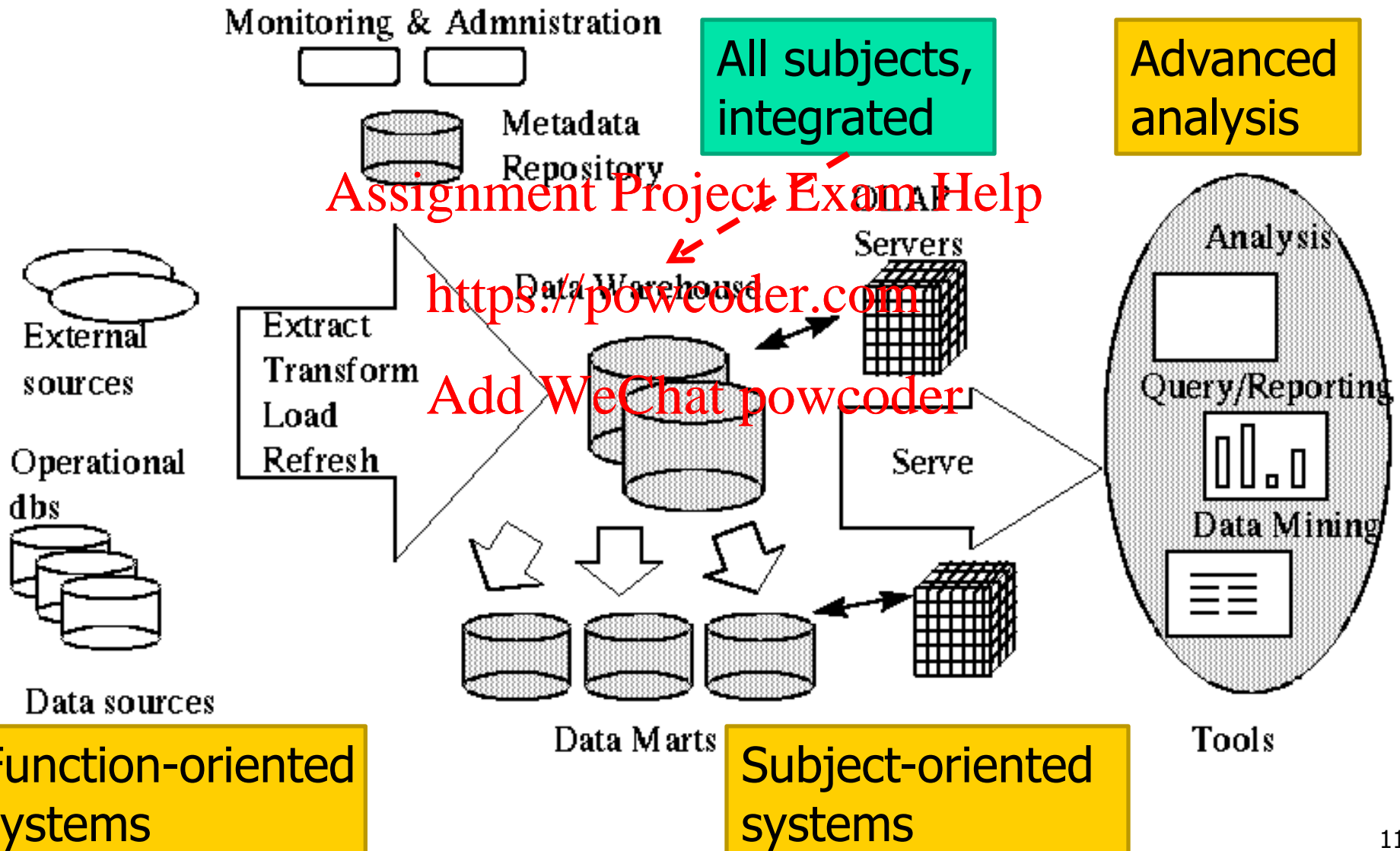
Data Warehouse—Non-Volatile

1. A physically separate store of data transformed from the operational environment.

2. Operational update of data does not occur in the data warehouse environment.

   - Does not require transaction processing, recovery, and concurrency control mechanisms

   - 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 offline
- OLAP-server provides multidimensional view
- Multidimensional-olap
  (Essbase, oracle express)
- Relational-olap
  (Redbrick, Informix, Sybase, SQL server)

Monitoring & Administration

Metadata Repository

External sources

Extract Transform Load Refresh

OLAP Servers

Data Warehouse

Operational dbs

Serve

Analysis

Query/Reporting

Data Mining

Data sources

Data Marts

Tools

Data Warehouse Architecture

Monitoring & Admnistration

Metadata Repository

All subjects, integrated

Advanced analysis

Assignment Project Exam Help

OLAP Servers

https://powcoder.com

Data Warehouse

Add WeChat powcoder

External sources

Operational dbs

Extract Transform Load Refresh

Serve

Analysis

Query/Reporting

Data Mining

Data sources

Data Marts

Tools

Function-oriented systems

Subject-oriented systems

11

Why Separate Data Warehouse?

- High performance for both systems

  - DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery

  - Warehouse—tuned for OLAP: complex OLAP queries, multidimensional view, consolidation.

- Different functions and different data:

  - <u>missing data</u>: Decision support requires historical data which operational DBs do not typically maintain

  - <u>data consolidation</u>: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources

  - <u>data quality</u>: 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 analytical processing)

    - Major task of data warehouse system

    - 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 detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
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	Multidimensional
Logical Model	(Normalized) Relational Model	(Denormalized) Star schema / Data 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 views	Bitmap/Join indexes, Star join, Materialized data cube

■ The Multidimensional Model

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

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 subject it models
    - Typically transactions in this course; other types includes snapshots, etc.
    - Measures: numbers that can be aggregated
    - 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 Measure: Profit
- Dimensions:
    - Store
    - Product
    - Time

Visualizing the Cubes

- A valid instance of the model is a data cube

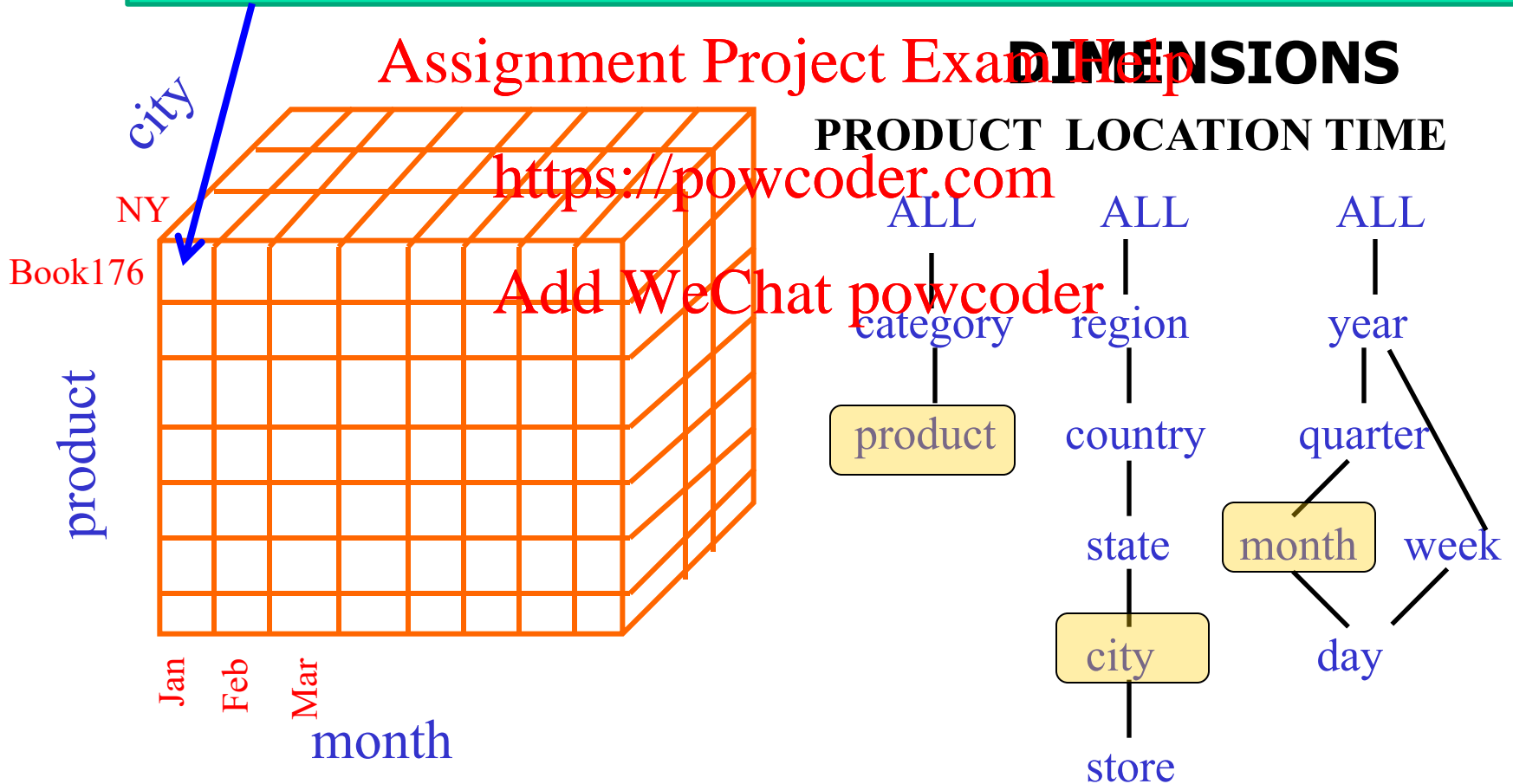| total Sales	product				_city_			
	p1	p2	p3	p4				
NY	$454	-	-	$925	454			925
LA	$468	$800	-	-	468	800		
SD	$296	-	$240	-	296		240	
SF	$652	-	$540	$745	652		540	745

product

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

Q: How to generalize it to 3D?

3D Cube and Hierarchies

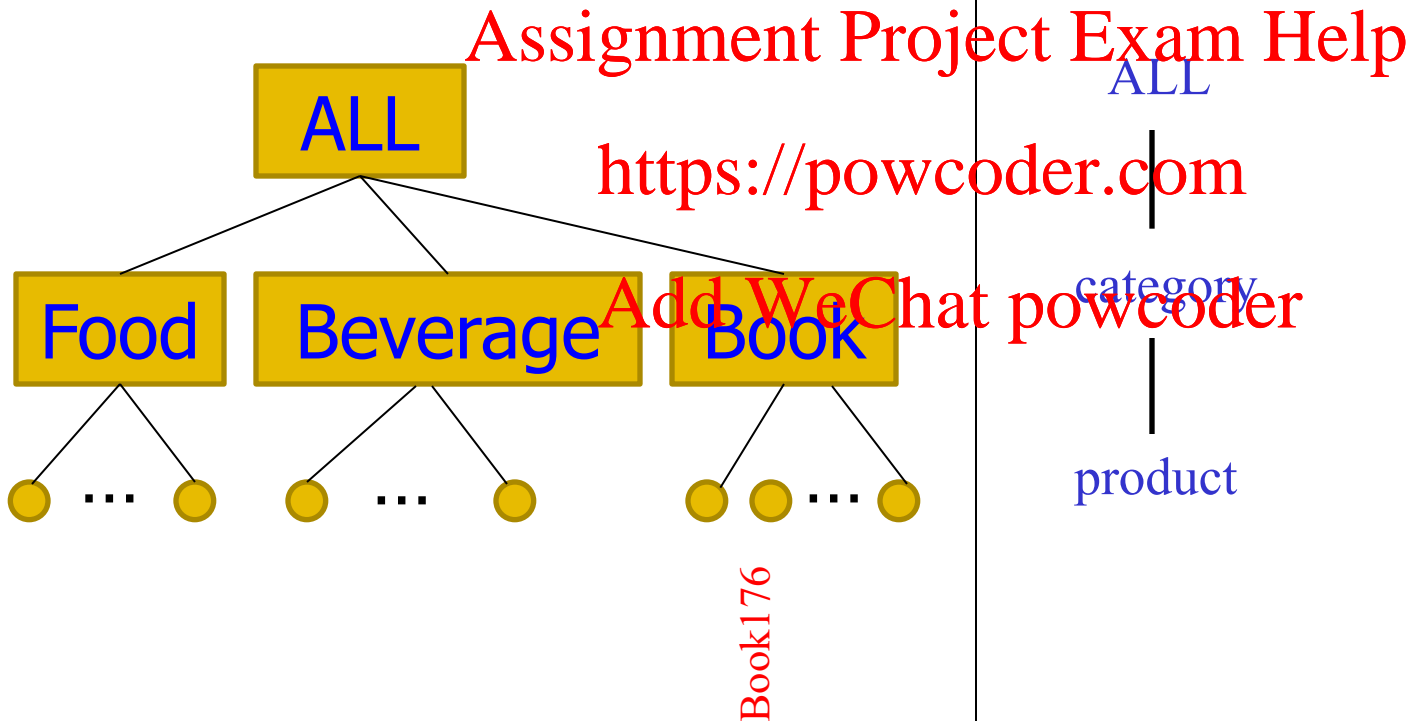**Concepts**: hierarchy (a tree of dimension values), level

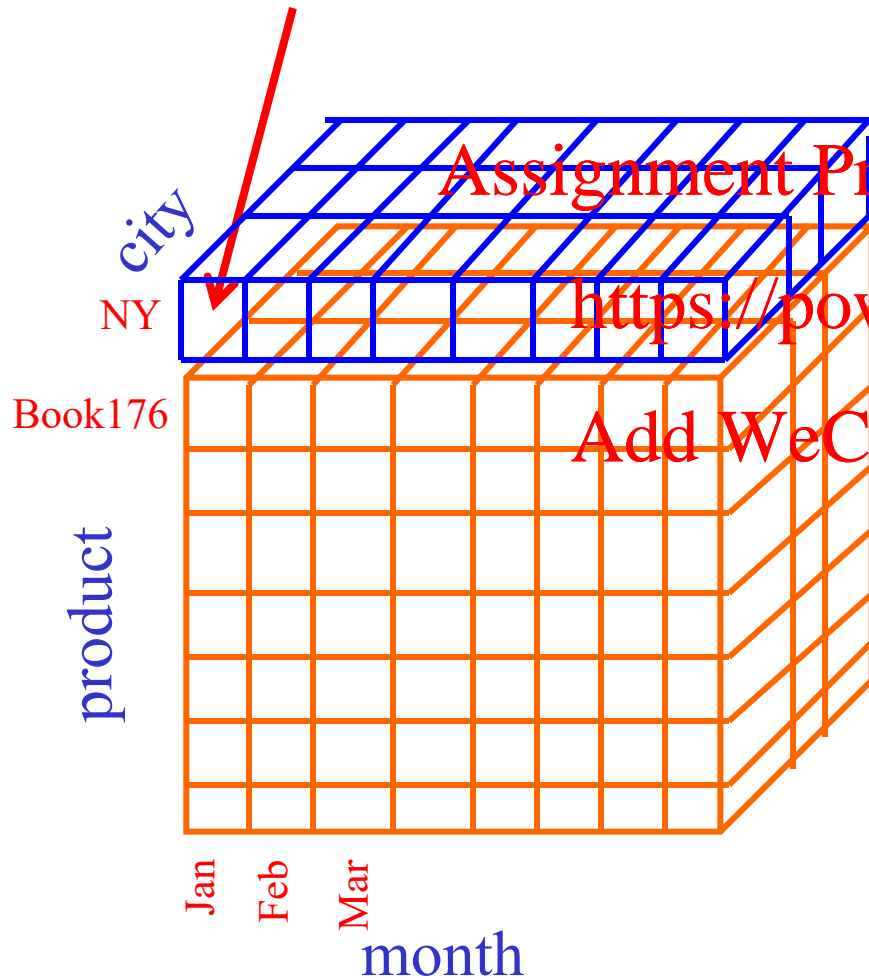Sales of book176 in NY in Jan can be found in this cell

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

DIMENSIONS

PRODUCT	LOCATION	TIME
ALL	ALL	ALL
category	region	year
product	country	quarter
	state	month week
	city	
	store	day

Hierarchies

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

Which design is better? Why?

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

ALL

Food   Beverage   Book

…   …   Book176   …

ALL

category

product

ALL

category

subcategory

brand

product

The (city, moth) Cuboid

Sales of ALL_PROD in NY in Jan

city

NY

Book176

product

Jan Feb Mar

month

DIMENSIONS

PRODUCT LOCATION TIME

ALL   ALL   ALL

category   region   year

product   country   quarter

                state   month   week

                city   day

                store

All the Cuboids

Date

Product

1Qtr   2Qtr   3Qtr   4Qtr

TV

PC

VCR

U.S.A

Canada

Mexico

Country

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

All the Cuboids /2

Total annual sales of TV in U.S.A.

Product

Date

TV PC VCR all

1Qtr 2Qtr 3Qtr 4Qtr all

Country

U.S.A Canada Mexico all

All, All, All

Lattice of the cuboids

Base cuboid

product, quarter, country — 3-dim cuboid

product,quarter   quarter, country   product, store — 2-dim cuboid

quarter   product   country — 1-dim cuboid
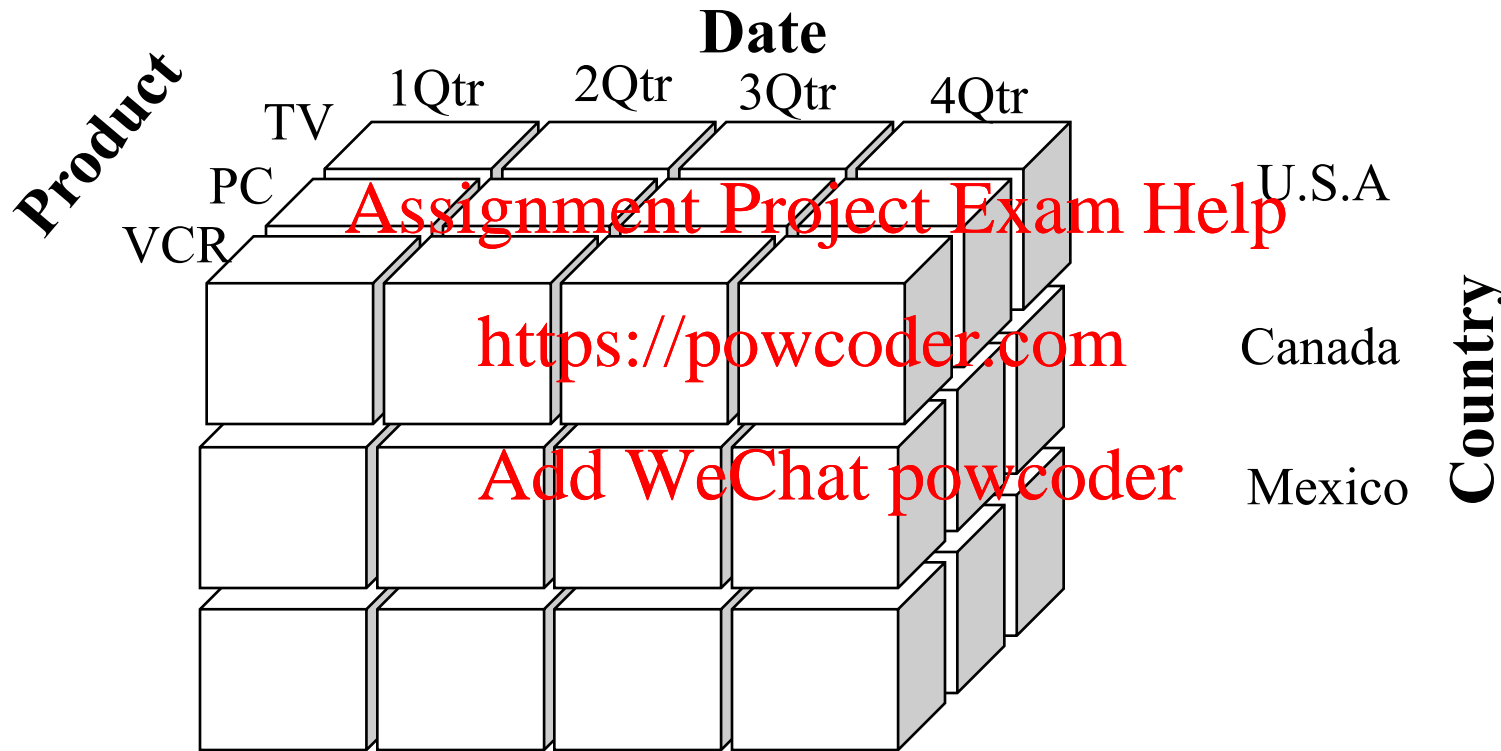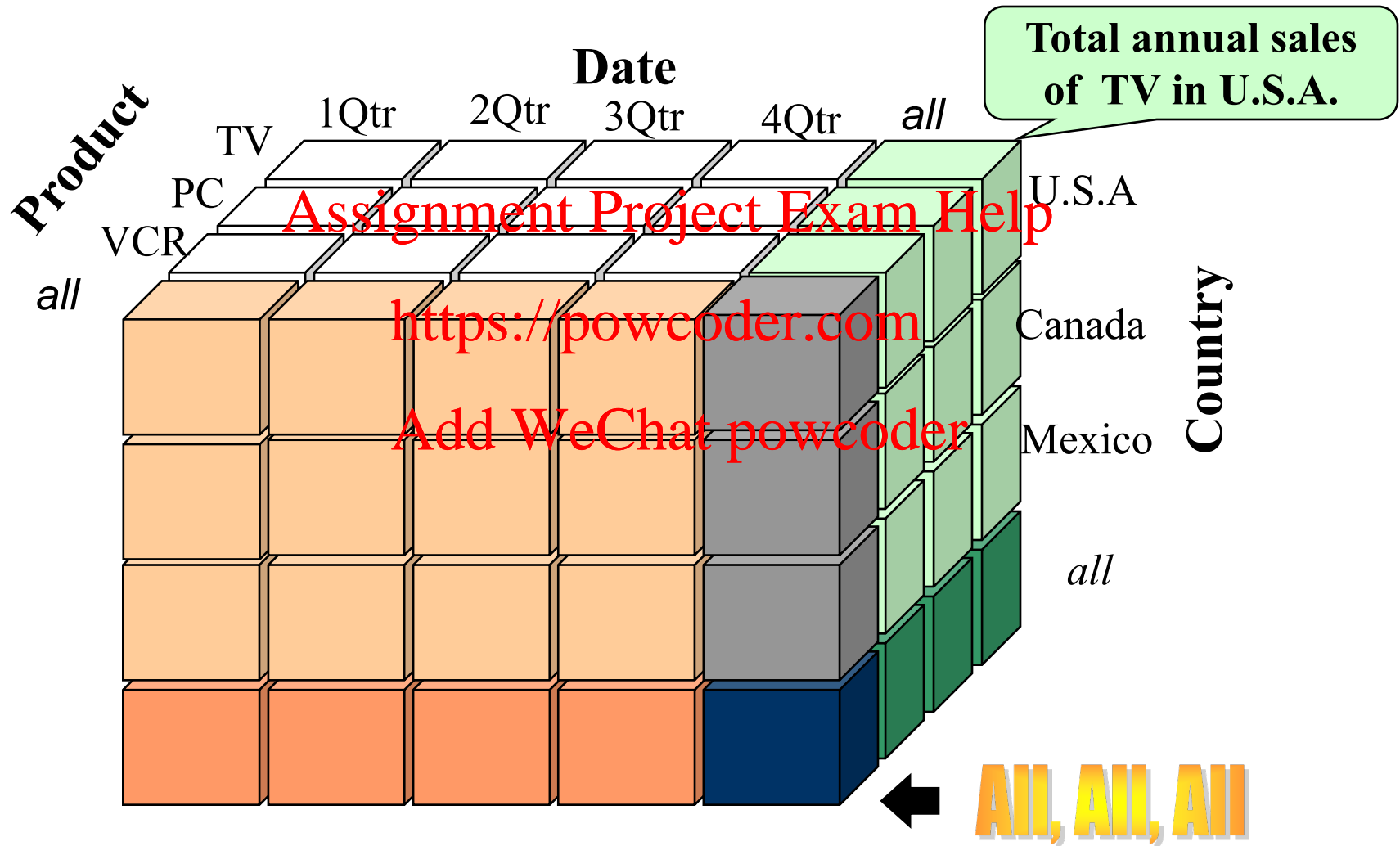
0-dim cuboid
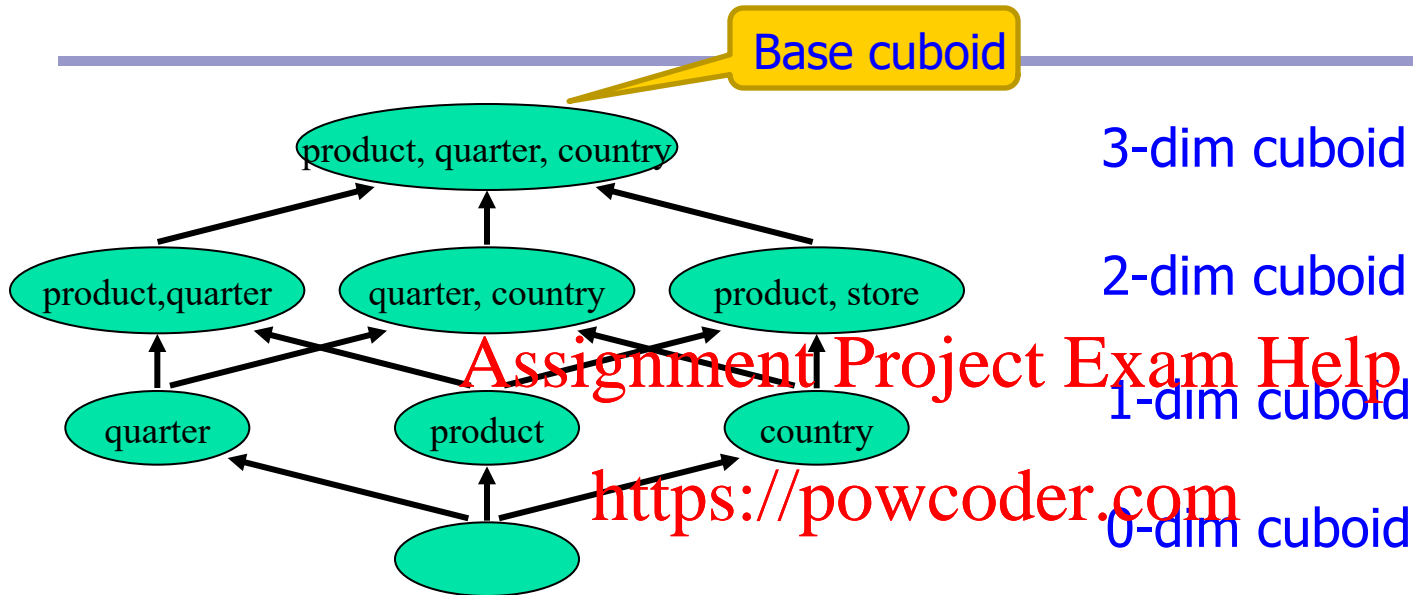
- n-dim cube can be represented as $(D_1, D_2, \ldots, D_d)$, where D_i is the set of allowed values on the i-th dimension.
  - if $D_i = L_i$ (a particular level), then D_i = all descendant dimension values of L_i.
  - ALL can be omitted and hence reduces the effective dimensionality
- A complete cube 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
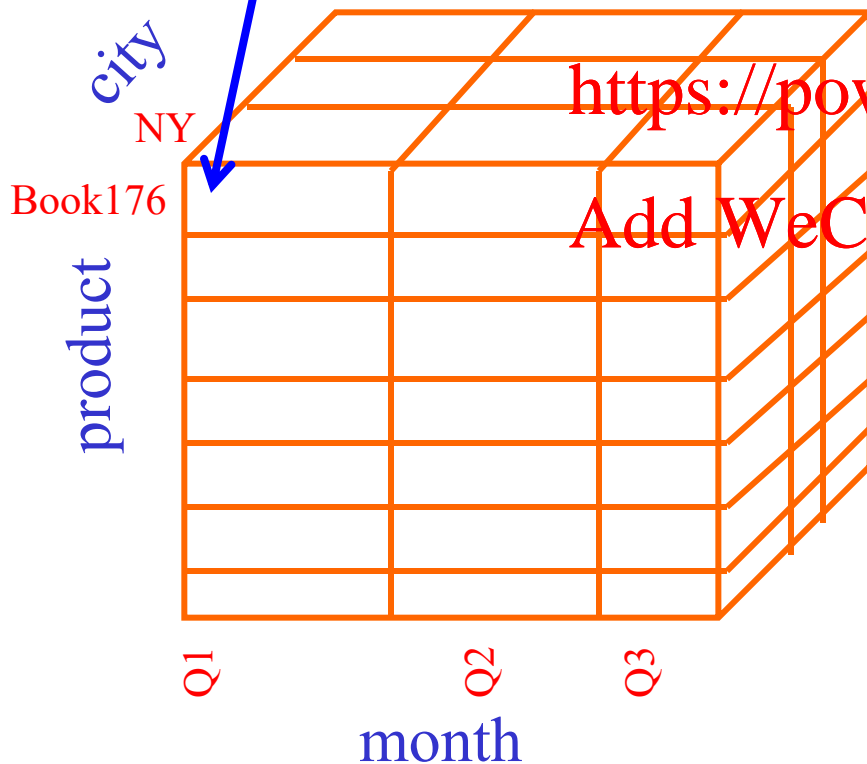- So that they can be composed

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

Common OLAP Operations

■ **Roll-up**: move up the hierarchy

Q: what should be its value?
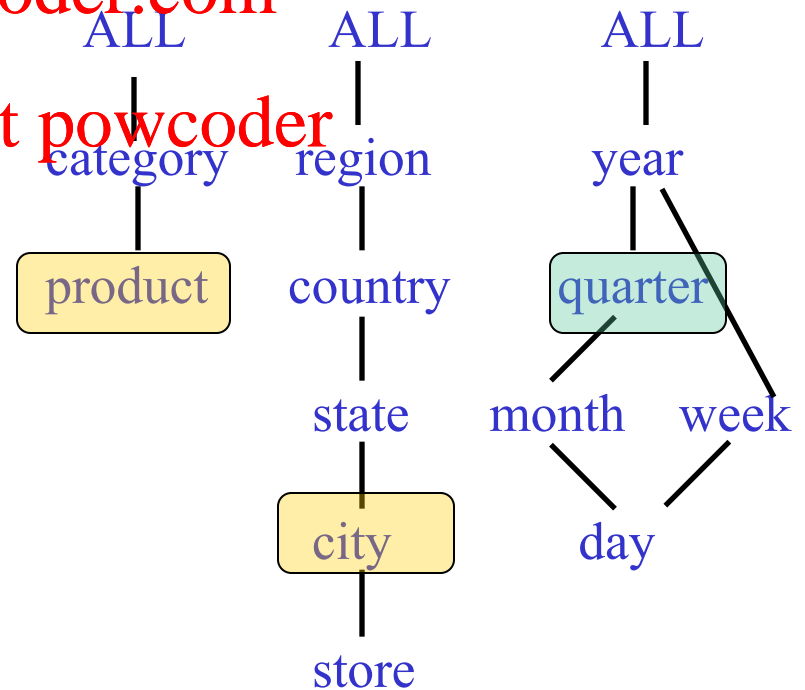
Sales of book176 in NY in Q1 here
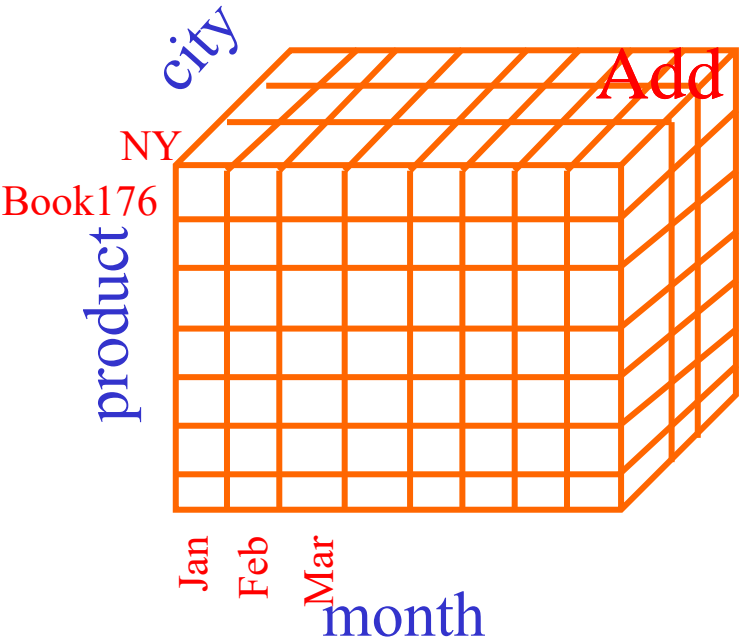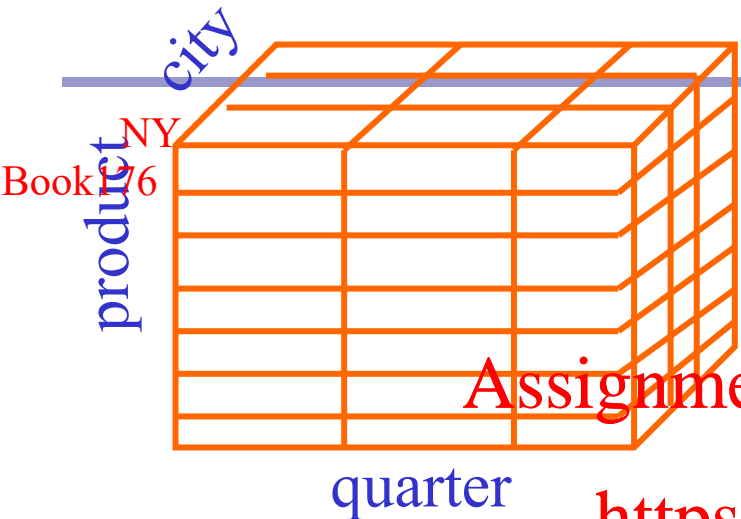
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

DIMENSIONS

PRODUCT LOCATION TIME

city

NY

Book176

product

Q1   Q2   Q3

month

PRODUCT	LOCATION	TIME
ALL	ALL	ALL
category	region	year
product	country	quarter
	state	month week
	city	day
	store	

DIMENSIONS

PRODUCT LOCATION TIME

- city
- NY
- Book176
- product
- quarter

- city
- NY
- Book176
- product
- Jan Feb Mar
- month

- category → product
- ALL
- region → country → state → city → store
- ALL
- year → quarter → month, week → day
- ALL
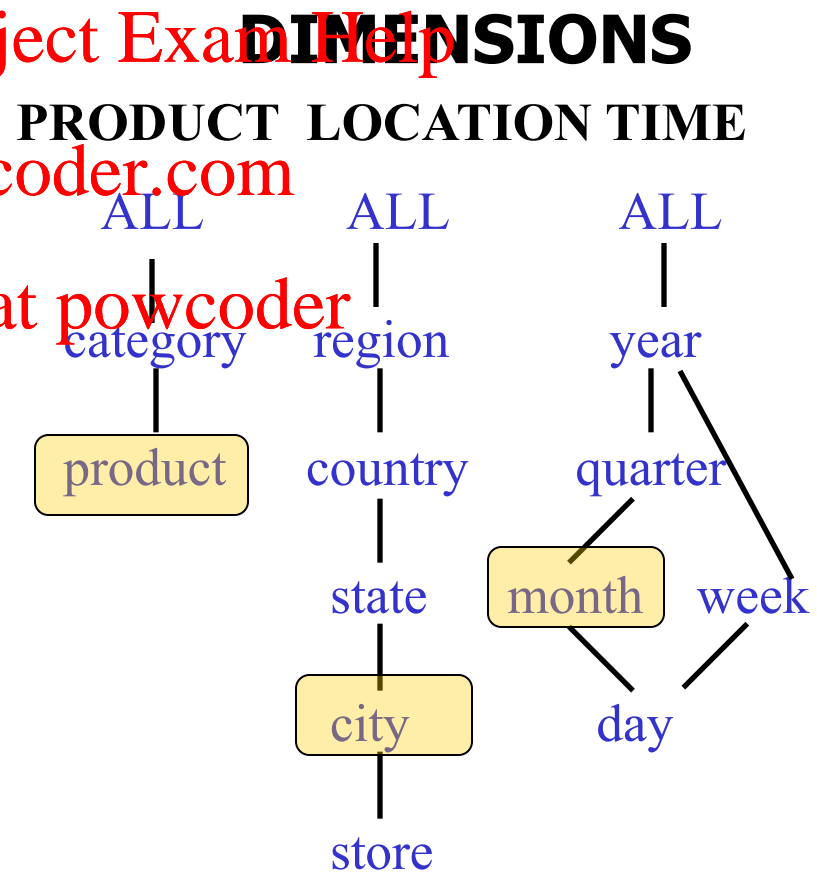
Data Cube Measures: Three Categories

- <u>Distributive</u>: 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()
- <u>Algebraic</u>: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., avg(), min_N(), standard_deviation()
- <u>Holistic:</u> if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

Common OLAP Operations

- **<u>Drill-down:</u>** move down the hierarchy
  - more fine-grained aggregation

Assignment Project Exam Help

https://powcoder.com
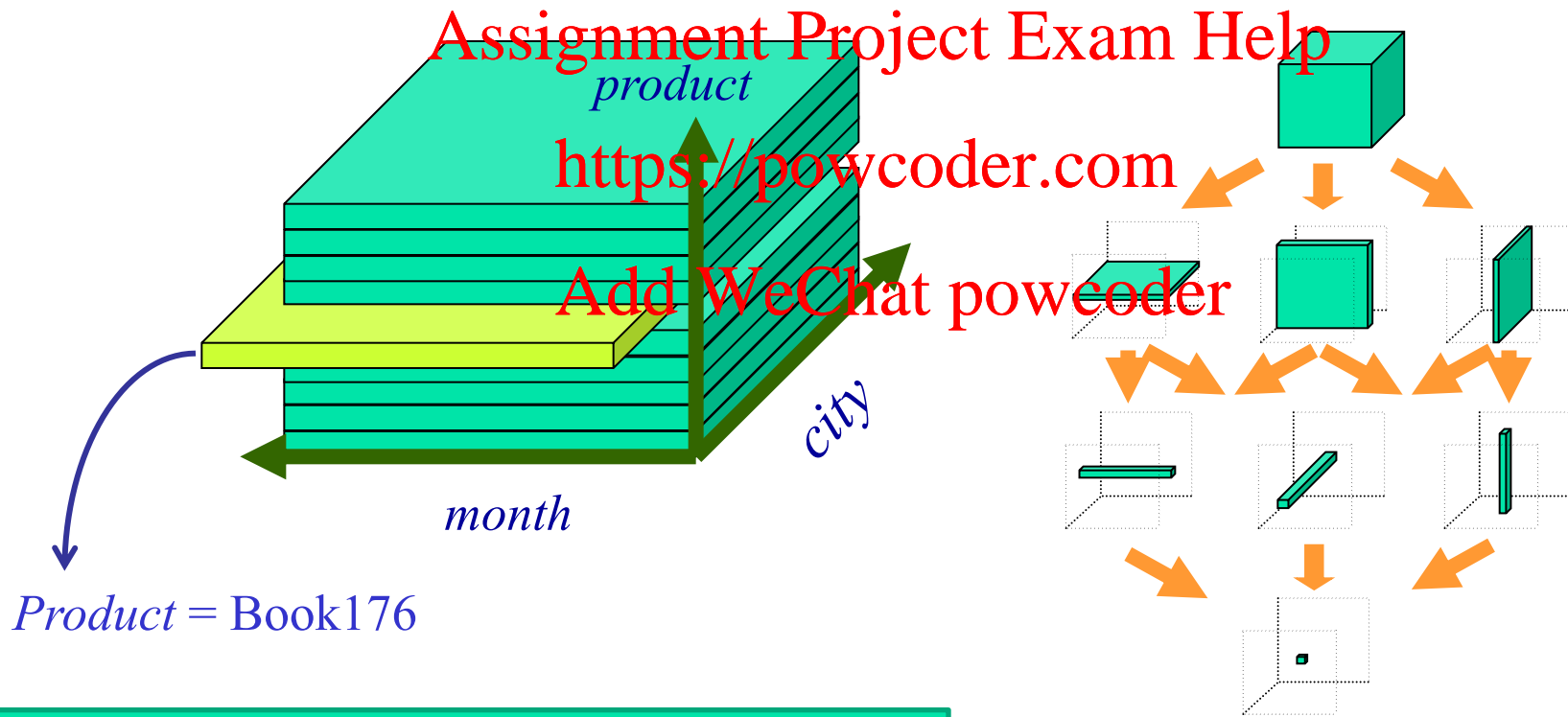
Add WeChat powcoder

Slice and Dice Queries

- <u>Slice and Dice</u>: select and project on one or more dimension values

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

product

city

month

Product = Book176

The output cube has smaller dimensionality than the input cube

Pivoting

- Pivoting: aggregate on selected dimensions
  - usually 2 dims (cross-tabulation)
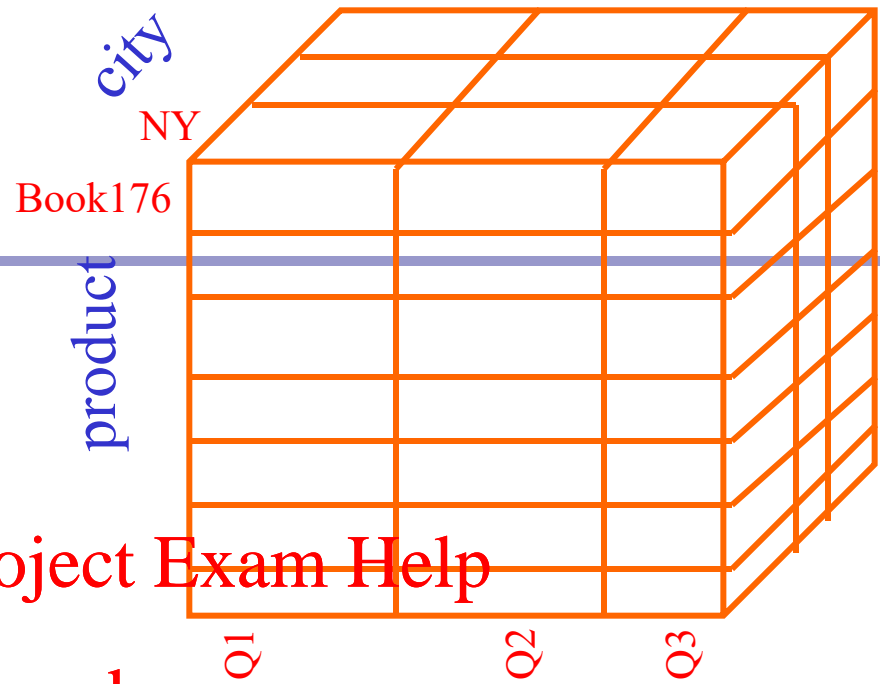
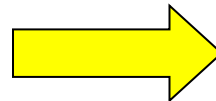Sales (of all products) in NY in Q1
=sum(???   )

pivot on (city, month)

city

NY

Book176

product

Q1 Q2 Q3

month

City

...		...
...	...	
...		...
NY | ... | | ... |

Q1 Q2 Q3

month

A Reflective Pause

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

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- **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

Assignment Project Exam Help

- MEASURE   https://powcoder.com

Add WeChat powcoder

- DIMENSIONS

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- **The Logical Model**

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Logical Models

- Two main approaches:
  - Using relational DB technology:
    - Star schema, Snowflake schema, Fact constellation
  - Using multidimensional technology:
    - Just as multidimensional data cube

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 tables are not normalized

- Transactions are described through a fact-table
  - each tuple consists of a logical pointer to each 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

TIME

time_key
day
day_of_the_week
month
quarter
year

SALES

time_key

product_key

location_key

----
units_sold

amount

PRODUCT

product_key
product_name
category
brand
color
supplier_name

LOCATION

location_key
store
street_address
city
state
country
region

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 schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming 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

41

Example of Star Schema

time
- time_key
- day
- day_of_the_week
- month
- quarter
- year

item
- item_key
- item_name
- brand
- type
- supplier_type

Sales Fact Table

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

| time_key |
| item_key |
| branch_key |
| location_key |
| units_sold |
| dollars_sold |
| avg_sales |

branch
- branch_key
- branch_name
- branch_type

location
- location_key
- street
- city
- state_or_province
- country

Measures

Example of Snowflake Schema

time

time_key
day
day_of_the_week
month
quarter
year

item

item_key
item_name
brand
type
supplier_key

supplier

supplier_key
supplier_type

Sales Fact Table

time_key

item_key

branch_key

location_key

units_sold

dollars_sold

avg_sales

Measures

branch

branch_key
branch_name
branch_type

location

location_key
street
city_key

city

city_key
city
state_or_province
country

Example of Fact Constellation

time

time_key
day
day_of_the_week
month
quarter
year

Sales Fact Table

time_key

items_key

branch_key

location_key

units_sold

dollars_sold

avg_sales

Measures

item

item_key
item_name
brand
type
supplier_type

Shipping Fact Table

time_key

item_key

Shipper_key

from_location

to_location

dollars_cost

units_shipped

branch

branch_key
branch_name
branch_type

location

location_key
street
city
province_or_state
country

shipper

shipper_key
shipper_name
location_key
shipper_type

Advantages of Star Schema

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

"Find total sales per product category in our stores 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**

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Query Language

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

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

SELECT   {[PRODUCT].[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

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Physical Model + Query Processing Techniques

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

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Q1: Selection on low-cardinality attributes

TIME

time_key
day
day_of_the_week
month
quarter
year

$S_{region=\text{"Europe"}}$

CUSTOMER

customer_key
customer_name
region
type

SALES

time_key

customer_key

location_key

- - - - - - - - - -

units_sold

amount

JOIN

JOIN

LOCATION

location_key
store
street_address
city
state
country
region

- 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 indexed column

- Multiple BIs can be efficiently combined to enable optimized scan of the table

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

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.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
-- 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	100	3
863	C3	200	5
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
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

BI on Sales(Customer.Region)

v	bitmap
Asia	11011000
Europe	00100101
America	00000010

Q2: Selection on high-cardinality attributes

TIME

time_key
day
day_of_the_week
month
quarter
year

SALES

time_key

customer_key

location_key

_units_sold_

amount

CUSTOMER

customer_key
customer_name
region
type

JOIN

JOIN

LOCATION

location_key
store
street_address
city
state
country
region

$S_{city=\text{``Kingsford''}}$

Indexing OLAP Data: Join Indices

- Join index relates the values of the <u>dimensions</u> of a star schema to <u>rows</u> in the fact table.

  - a join index on city maintains for each distinct city a list of ROW-IDs for 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

Location

city = Sydney
city = Randwick
city = Kingsford
city = Coogee

Sales

R102   1

R117   1

R118   1

R124   1

Kingsford → R102,R117,R118,R124

Q3: Arbitrary selections on Dimensions

TIME

time_key
day
day_of_the_week
month
quarter
year

$S_{isSchoolHolidy(day)}$

$S_{region = \text{"Asia"}}$

Customer

customer_key
customer_name
region
type

SALES

time_key

product_key

location_key

- - - - - - - - - - - -

units_sold

amount

JOIN

JOIN

LOCATION

location_key
store
street_address
city
state
country
region

$S_{city =\sim /\backslash S+ford/}$

Star Query and Star Join (Cont.)
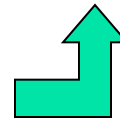
Usually only part of the dim tables because of the selection predicates

Time **Customer** **Loc**

time
1
2

X

Cust
10
20

X

loc
100

millions of tuples

Sales

time	cust	loc	sold
1	10	100	7
1	10	150	13
1	20	150	2
2	20	200	16
…	…	…	…
1000	2000	500	86

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

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

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

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

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 unclustered

- High-level aggregations are expensive!!!!!

  ⇒Long Query Response Times

  ⇒Pre-computation is necessary

  ⇒Pre-computation is most beneficial
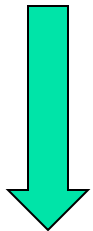
LOCATON
|
region
|
country
|
state
|
city
|
store

Cuboids = GROUP BYs

- Multidimensional aggregation = selection on corresponding cuboid

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

σ_1 selects some Years, σ_2 selects some Brands, σ_3 selects some Cities

$$GB_{(type,\ city)}(\ \sigma_{1'2'3'}(Cuboid(Year,\ Type,\ City)))\ )$$
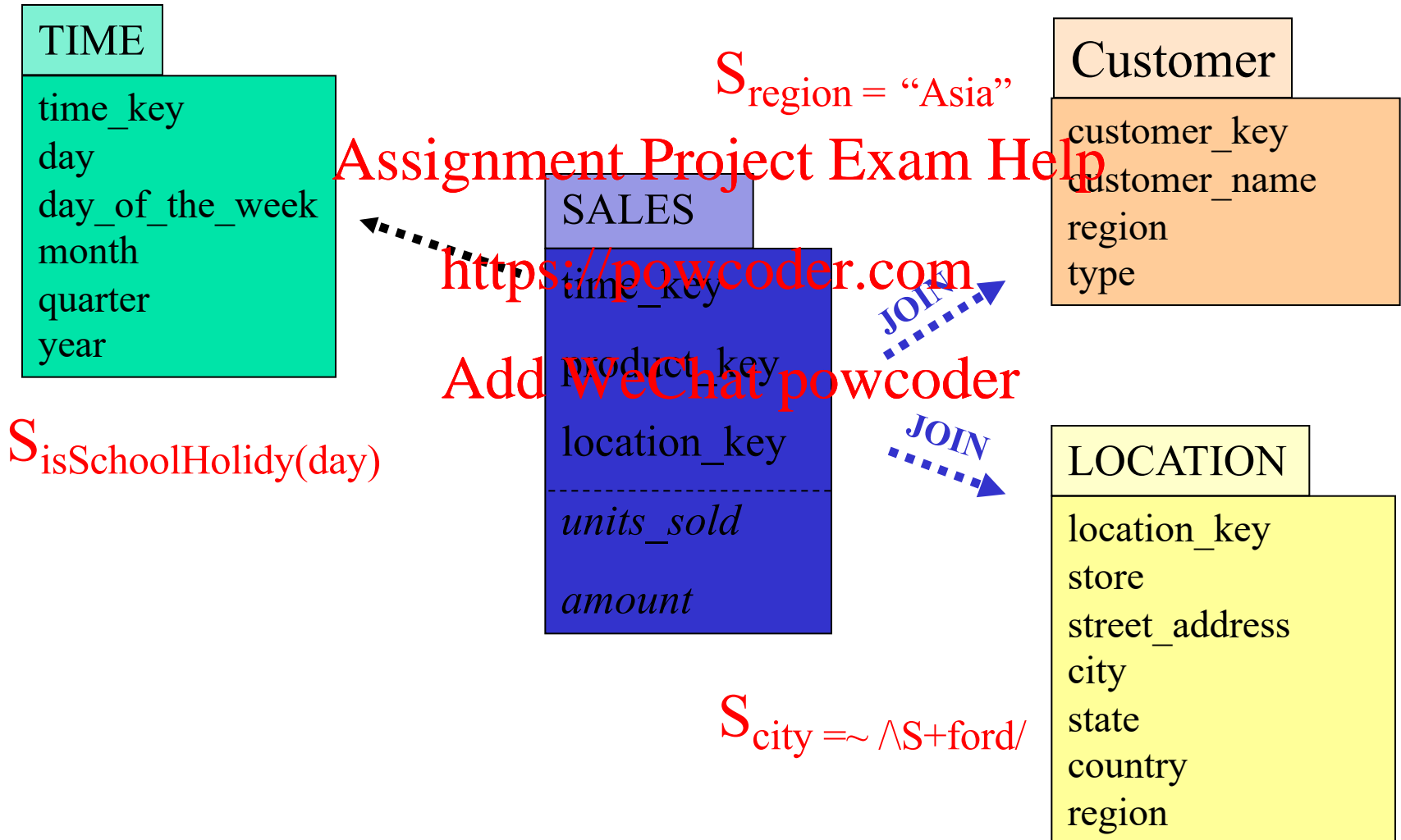
- Materialize some/all of the cuboids

  - A complex decision involving cuboid sizes, query workload, and physical organization

59

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	745	1937
	ALL	1870	800	780	1670	5120

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
4	3	240
4	4	745
1	ALL	1379
2	ALL	1268
3	ALL	536
4	ALL	1937
ALL	1	1870
ALL	2	800
ALL	3	780
ALL	4	1670
ALL	ALL	5120

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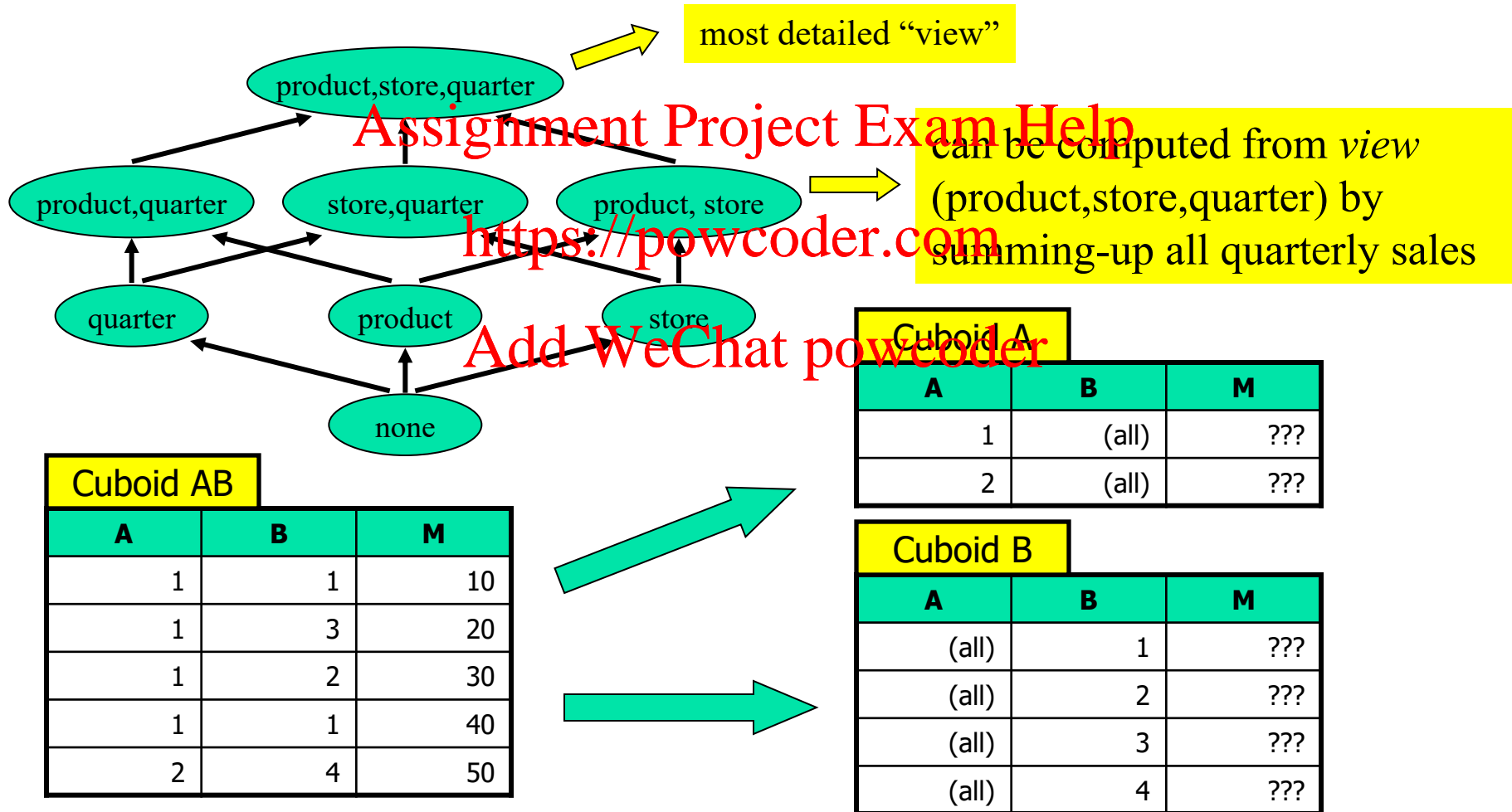**(store,product)**
(store)
(product)
()

- 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:

most detailed "view"

product,store,quarter

product,quarter   store,quarter   product, store

can be computed from *view* (product,store,quarter) by summing-up all quarterly sales

quarter   product   store

Cuboid A

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

Cuboid B

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

Cuboid AB

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

Bottom-Up Approach (BUC)

- BUC (Beyer & Ramakrishnan, SIGMOD'99)

- Ideas
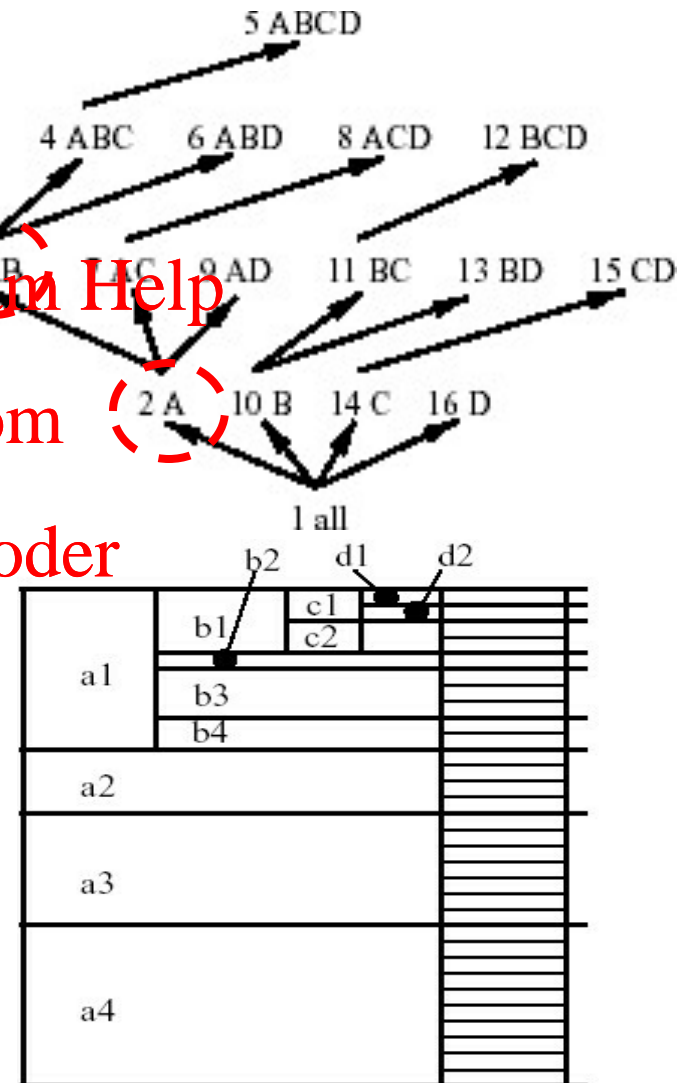 - Compute the cube from bottom up
  - Divide-and-conquer

- A simpler recursive version:
  - BUC-SR

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

5 ABCD

4 ABC   6 ABD   8 ACD   12 BCD

3 AB   7 AC   9 AD   11 BC   13 BD   15 CD

2 A   10 B   14 C   16 D

1 all

a1   b1   b2   c1   c2   d1   d2   b3   b4
a2
a3
a4
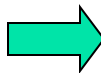
Understanding Recursion /1

- Powerful computing/problem-solving techniques

- Examples
    - Factorial:
        - f(n) = 1, if n = 1
        - f(n) = f(n-1) * n, if n ≥ 1

    > f(0) = 0! = ???

    - 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*]

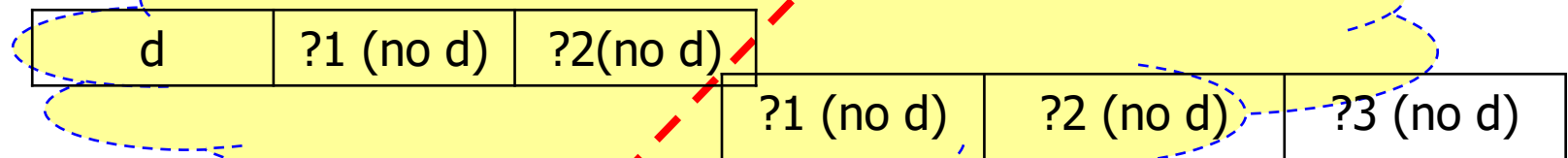   > List comprehension in Haskell or python

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 = 3, n = 5

- Consider any ball in the 5 balls, e.g., 'd'

entire solution space

?1	?2	?3

d	?1 (no d)	?2(no d)

			?1 (no d)	?2 (no d)	?3 (no d)

?i in { a b c d e }

Key Points

- Sub-problems need to be "**smaller**", so that a simple/trivial boundary case can be reached

- Divide-and-conquer
  - There may be multiple ways the entire solution space can be divided into **disjoint** sub-spaces, each of which can be conquered **recursively**.

Geometric Intuition /1

■ Reduce Cube(in 2D) to Cube(in 1D)

	b1	b2	b3	
a1	M11	M12	M13	[Step 1]
a2	M21	M22	M23	[Step 1]
	[Step 2]	[Step 2]	[Step 2]	[Step 3]

	b1	b2	b3	*
[a1] ×	M11	M12	M13	[Step 1]
[a2] ×	M21	M22	M23	[Step 1]
[*] ×	[Step 2]	[Step 2]	[Step 2]	[Step 3]

Geometric Intuition /2

- Reduce Cube(in 3D) to Cube(in 2D)
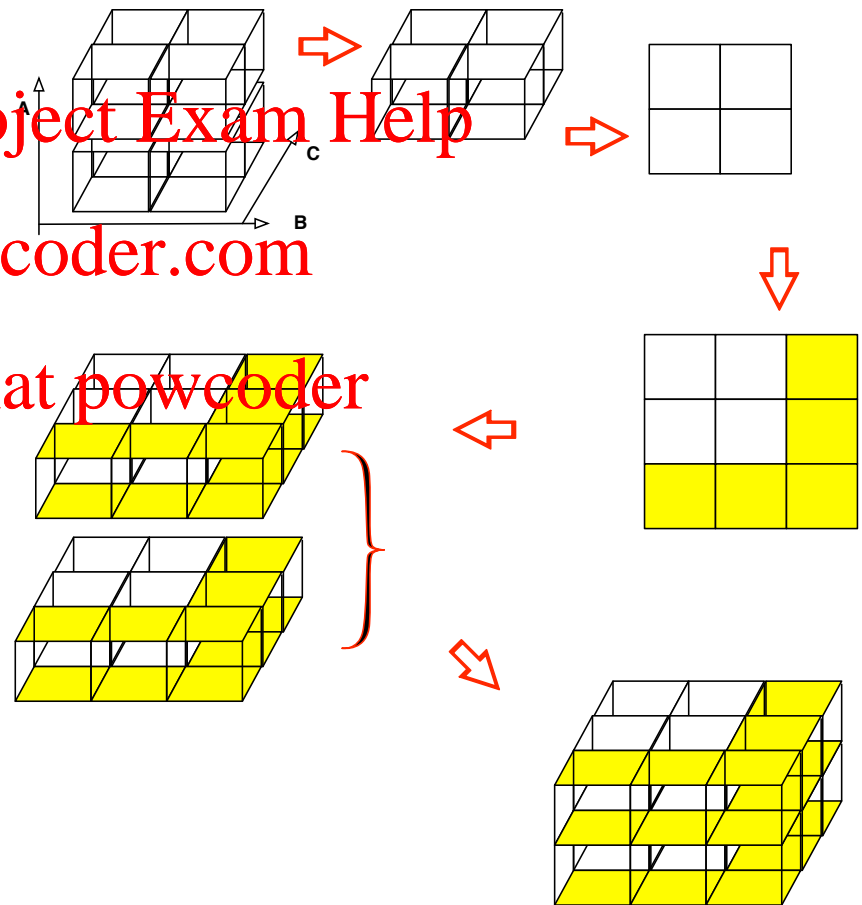
Geometric Intuition /3
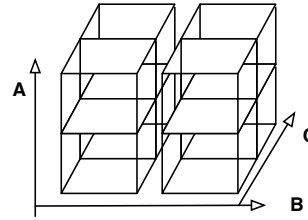
- Reduce Cube(in 3D) to Cube(in 2D)
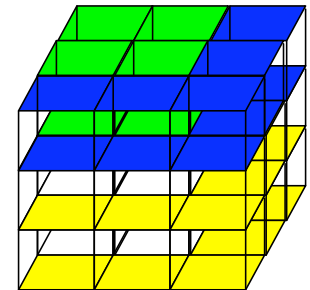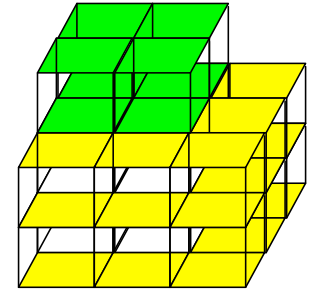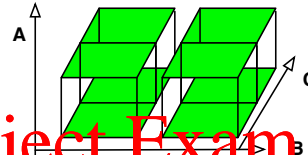
Algebraic Derivation

- How to compute *n*-dim cube on (*n*+1)-dim base cuboid (array)?

  - What do output tuples look like?

- How to compute (*n*+1)-dim cube on (*n*+1)-dim base cuboid (array)?

  - What else do we need?

[{r1–r5}, **ABC**]

[{r1–r5}, **BC**]

	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

BUC-SR (Simple Recursion)*

- BUC-SR(data, dims)

  - If (dims is empty)

    - Output (sum(data))

  - Else

    - Dims = [dim1, rest_of_dims]

    - 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

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Example

[{r1-r4}, B]

B	M
1	10
1	20
2	30
3	40

Internal Output:

B	M
1	30
2	30
3	40
*	100

Output:

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

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

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

[{r5}, B]

B	M
1	50

Internal Output:

B	M
1	50
*	50

Output:

A	B	M
2	1	50
2	*	50

[{r1-r5}, AB]

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

r1
r2
r3
r4
r5

[{r1'-r5'}, B]

B	M
1	80
2	30
3	40

Internal Output:

B	M
1	80
2	30
3	40
*	150

Output:

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 (esp. for dense cubes)
  - fast in indexing and query processing
- Cons:
  - scalability
  - conversion from relational data

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	927
Q1	computer	825
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
Q1	0
Q2	1
Q3	2
Q4	3

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
2	1	1023	9
3	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

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
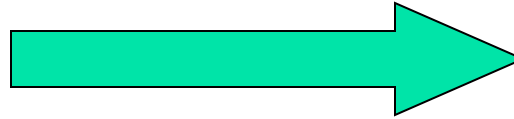
■ Think: how to decode a slot?

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

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

The Sparse Case

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 */

Step 1

Mappings

time	*value*
Q1	0
Q2	1
Q3	2
Q4	3

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

Q4	security	580

time	item	dollars_sold	offset
0	0	605	0

/* same table but with many rows deleted to make 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
  - Use sparse array (i.e., offset + value)
  - Could use chunk to further reduce the space
- Space usage:
  - (d+1)*n*4 vs 2*n*4

- HOLAP:
  - Store all non-base cuboid in MD array
  - Assign a value for ALL