CSI4142: Data Science

Topic 3:
Physical Design

(Slides by HL Viktor ©: based on Kimball and Ross, Chapters 2, 15 and 20, as well as Han et. al. Chapter 3

1

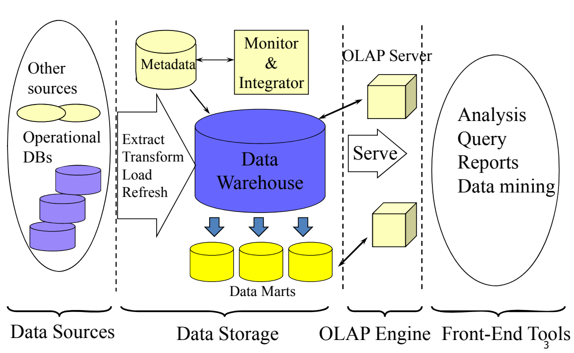
Overview of topic

Creating a data mart:

a. Dimensional (Conceptual) modelling
   i. Star Schemas
   ii. DW Bus Matrix
b. **Physical Design**
   i. **Aggregates, Cubes and Cuboids**
   ii. **Completing the Physical Design**
c. Data staging: extract, transform, load and refresh

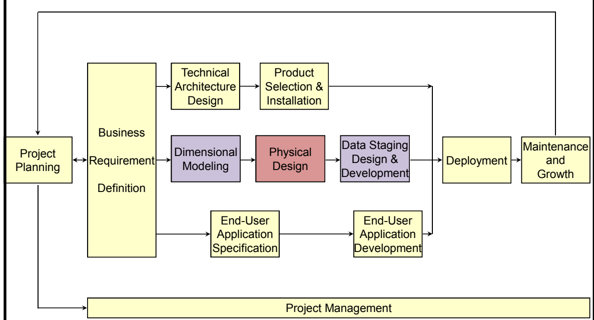
2

Data Warehouse: A Multi-Tiered Architecture



3

Business Dimensional Lifecycle: Kimball and Ross



4

Issues to address

- How do we make sure our system performance is OK?
  - Aggregates (Cubes and Cuboids)
  - A word about Physical Design

5

Learning objectives: Aggregates

- Aggregates are a way to speed up frequent queries
- May be modelled as a lattice of **Cuboids**
- One Cuboid correspond to One Aggregate
- Correspond to **some** pre-stored “materialized views” (results of aggregated queries)

- We aim to design the “optimum set” of aggregates
  - Answer **many** queries faster
  - Using **reasonable** disk space

6

What is an aggregate?

- Data are **SUMMED** using **Concept Hierarchies**
- Pre-calculated and pre-stored summaries that are stored in the data warehouse
- Used for **Query Optimization when doing OLAP operations**
- Aggregates will periodically, dynamically change, since it depends on the frequent queries
  - Frequent business requests
  - Statistical distribution of data

Data Mart = Base Dim. Model + Aggregate Dim Models

7

Why do we need to aggregate?
Example Telephone Call Tracking

- Date dimension: 3 years → 1095 days
- Number of tracked calls per day: 100 million
- Number of base fact records: 109 billion records
- Number of key fields = 5
- Number of fact/measure fields: 3
- Base fact table size (est.): 3490Gb, 3.49TB

8

Why do we need to aggregate?
Typical Business Question in Retail

- How much total business did my newly remodeled stores do compared to the chain average?
- How did leather goods costing less than $40 do with my most frequent shoppers?
- What was the ratio of non-holiday weekend days total revenue to holiday weekend days?

- Detailed information about **one** dimension
- Needed: A set of pre-computed aggregates

9

What to aggregate: The different types of aggregates (Retail)

- Category level **items** aggregates by **location** by **day**
- District level **locations** aggregates by **items** by **day**
- Monthly Sales level by **item** by **location**
- Category-level **product** aggregates by **location** by **day**
- Category-level **product** aggregates by **location city** by **month**

- **Each aggregate occupies its own fact table**

10

Sales Fact Table: Original

Time Dimension
Time Key (PK)
Day of week
Day number in month
Day number overall
Week number in year
Week number overall
Month
Month number overall
Quarter
Fiscal-period
Holiday-flag

Sales Fact Table
Time Key (FK)
Product Key (FK)
Store Key (FK)
Dollars Sold
Units sold
Quantity sold

Product Dimension
Product Key (PK)
SKU_number
Description
Brand
Category
Department
Package size
Other attributes…

Store Dimension
Store Key (PK)
Store-name
Store-number
Store-street-address
City
Store-state
Other attributes ….

11

Sales Category Fact Table: Aggregate by SUM() on Category

Time Dimension
Time Key (PK)
Other attributes…

Sales Fact Table
Time Key (FK)
Product Key (FK)
Store Key (FK)
Dollars Sold
Units sold
Quantity sold

Category Dimension
Product Key (PK)
Category
Department

Store Dimension
Store Key (PK)
Other attributes ….

This code :

```
"SELECT  P.category, SUM(F.dollars_sold), SUM(F.units_sold),
         SUM(F.quantity_sold)
FROM     store S, product P, Date D, sales_fact F
WHERE    P.product_key = F.product_key AND
         D.time_key = F.time_key AND
         S.store_key = F.store_key
GROUP BY P.category;"
```

should do the trick!

12

Aggregate Fact Tables

- Dimension tables are "Shrunken versions" of the dimensional tables associated with the base
- Store in own fact tables, a "family of schemas"
- Uses **concept hierarchies to calculate**

TRANSPARENCY:

- End users only know of base cube
- Aggregate Navigator (AN) choose the correct cuboid

Note:

- OLAP Cube engines (if used) precompute some aggregates

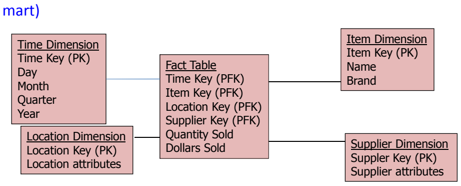
  Pro: Fast queries

  Cons: Slow at Loading and Refresh, Black Box, Vendor Specific

13

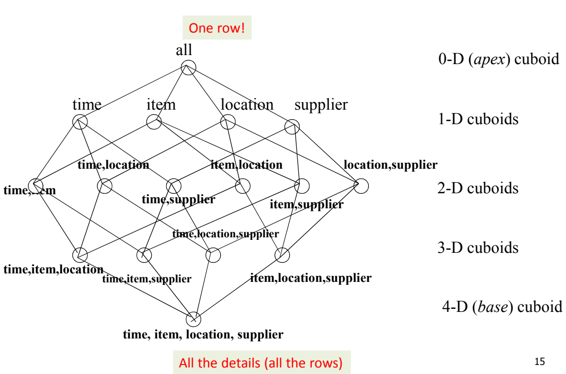
Sales example

- In the multidimensional data model, the (relational) star schema is implemented as a OLAP data cube
- In data warehousing literature, an n-D **base cube** is called a base cuboid.
- The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid.
- The lattice of cuboids forms a OLAP data cube (family of schemas, data mart)



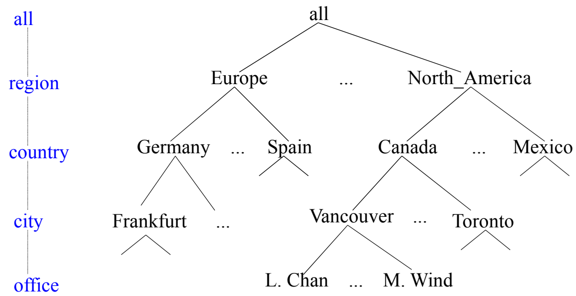
14

Cube: A Lattice of Cuboids



15

A Concept Hierarchy:
Dimension (Location)



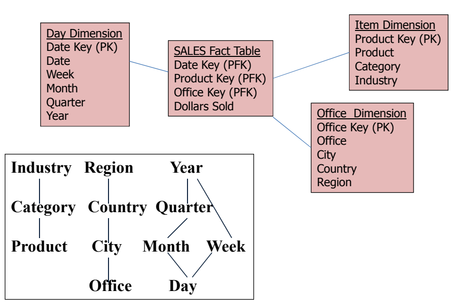
16

Another example

Sales of TVs, VCRs, and PCs, in North America

17

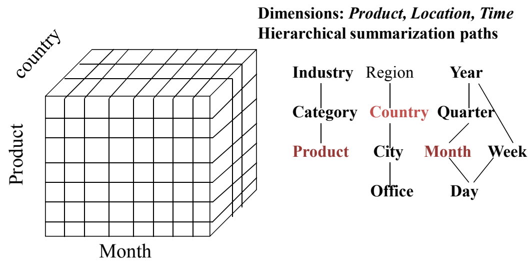
Original Base Star Schema



18

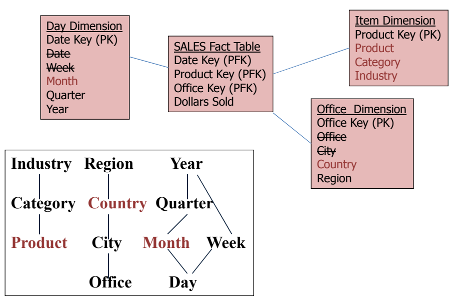
Multidimensional Data: Aggregation

- Frequent user access: Sales volume as a function of product, month, and country



19

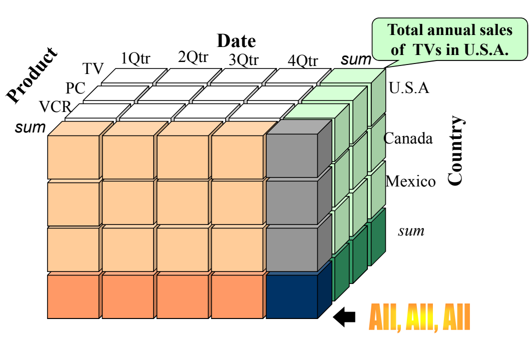
Level for aggregation: User Driven



`How do we implement this in SQL?`

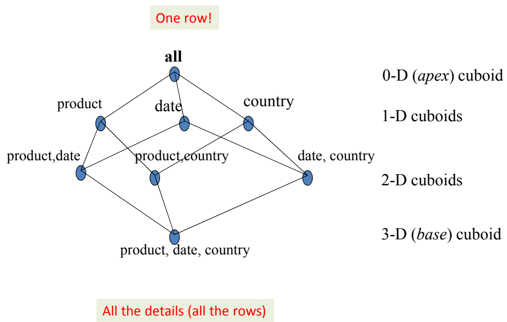
20

A Sample Data Cube



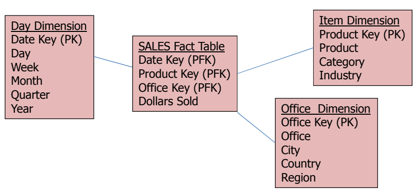
21

Cuboids Corresponding to the Cube



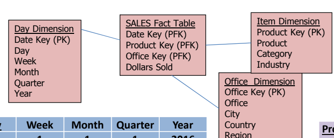
22

Star Schema… Detailed level



23

Star Schema… "Toy" data



Date-key	Day	Week	Month	Quarter	Year
100	12	1	1	1	2016
101	13	1	1	1	2016
102	9	15	5	2	2016
103	15	51	11	4	2016

Product-key	Product	Category	Industry
200	TV	Home	Entertain
201	VCR	Home	Entertain
202	PC	Office	Work
203	PC	Home	Personal

Office-key	Office	City	Country	Region
400	Joe	Ottawa	Canada	NA
401	Ann	Mexico	Cancun	NA
403	Sue	Mexico	Mexico City	NA

Date-key	Product-key	Office-key	Dollars-Sold
100	200	400	620
100	201	400	300
102	203	401	300
102	200	401	100
102	202	401	450
103	200	403	230

24

SQL operations

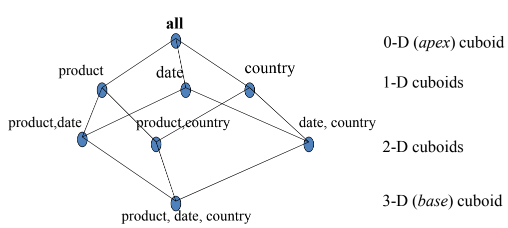
1. Create tables: Date, Office, Item
2. Create table: Sales fact
3. Insert data: Date, Office, Item
4. Insert data: Sales fact
5. SELECT SUM(): OLAP queries

Aggregates:--

1. SELECT SUM() → SELECT: Against pre-computed Aggregates

25

Why Cuboids to Store?



26

Example data for 3-D base cuboid

Office-key	Country
400	Mexico
401	Canada

Date-key	Product-key	Office-key	Dollars-Sold
1000	200	400	6200
1000	202	400	3400
1002	203	400	6000
1002	202	400	1230
1004	200	401	4300
1003	200	401	2300
1003	201	401	4300
1003	200	401	4500

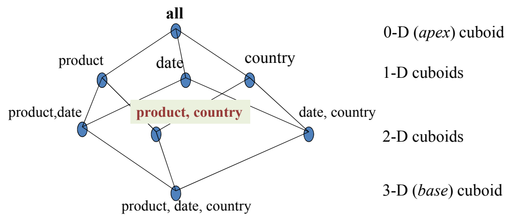
Product-key	Product	Category	Industry
200	TV	Home	Entertain
201	VCR	Home	Entertain
202	PC	Office	Work
203	PC	Home	Personal

Date-key	Month	Quarter	Year
1000	1	1	2016
1001	2	1	2016
1002	3	1	2016
1003	4	2	2016
1004	5	2	2016
1005	6	2	2016

Note I made up new data for the Country, Date and Fact tables …

27

Example data: one of the 2-D cuboids



28

Example data: Products per Country

Office-key	Country
400	Mexico
401	Canada

Office-key	Product-key	Dollars-sold
400	200	6200
400	202	4630
400	203	6000
401	200	11100
401	201	4300

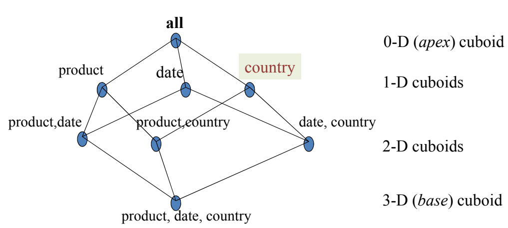
Date-key	Product-key	Office-key	Dollars-Sold
1000	200	400	62
1000	202	400	34
1002	203	400	60
1002	202	400	12
1004	200	401	43
1003	200	401	23
1003	201	401	43
1003	200	401	45

Product-key	Product	Category	Industry
200	TV	Home	Entertain
201	VCR	Home	Entertain
202	PC	Office	Work
203	PC	Home	Personal

Date-key	Month	Quarter	Year
1000	1	1	2016
1001	2	1	2016
1002	3	1	2016
1003	4	2	2016
1004	5	2	2016
1005	6	2	2016

29

Example data: one of the 1-D cuboids (country)



30

Example data: Country totals

Office-key	Country
400	Mexico
401	Canada

Office-key	Dollars-sold
400	16830
401	15400

Office-key	Product-key	Dollars-sold
400	200	6200
400	202	4630
400	203	6000
401	200	11100
401	201	4300

Date-key	Product-key	Office-key	Dollars-So
1000	200	400	62
1000	202	400	34
1002	203	400	60
1002	202	400	12
1004	200	401	43
1003	200	401	23
1003	201	401	43
1003	200	401	45

Product-key	Product	Category	Industry
200	TV	Home	Entertain
201	VCR	Home	Entertain
202	PC	Office	Work
203	PC	Home	Personal

Date-key	Month	Quarter	Year
1000	1	1	2016
1001	2	1	2016
1002	3	1	2016
1003	4	2	2016
1004	5	2	2016
1005	6	2	2016

31

Apex Cuboid: a single number

Office-key	Country
400	Mexico
401	Canada

Office-key	Dollars-sold
400	16830
401	15400

Office-key	Product-key	Dollars-sold
400	200	6200
400	202	4630
400	203	6000
401	200	11100
401	201	4300

Date-key	Product-key	Office-key	Dollars-So
1000	200	400	62
1000	202	400	34
1002	203	400	60
1002	202	400	12
1004	200	401	43
1003	200	401	23
1003	201	401	43
1003	200	401	45

Dollars-Sold
32230

Product-key	Product	Category	Industry
200	TV	Home	Entertain
201	VCR	Home	Entertain
202	PC	Office	Work
203	PC	Home	Personal

Date-key	Month	Quarter	Year
1000	1	1	2016
1001	2	1	2016
1002	3	1	2016
1003	4	2	2016
1004	5	2	2016
1005	6	2	2016

Note I made up new data for the Country, Date and Fact tables …

32

Running queries: which level to use?

Office-key	Dollars-sold
400	16830
401	15400

VS

Office-key	Product-key	Dollars-sold
400	200	6200
400	202	4630
400	203	6000
401	200	11100
401	201	4300

VS

Date-key	Product-key	Office-key	Dollars-Sold
1000	200	400	6200
1000	202	400	3400
1002	203	400	6000
1002	202	400	1230
1004	200	401	4300
1003	200	401	2300
1003	201	401	4300
1003	200	401	4500

Product-key	Product	Category	Industry
200	TV	Home	Entertain
201	VCR	Home	Entertain
202	PC	Office	Work
203	PC	Home	Personal

Date-key	Month	Quarter	Year
1000	1	1	2016
1001	2	1	2016
1002	3	1	2016
1003	4	2	2016
1004	5	2	2016
1005	6	2	2016

Office-key	Country
400	Mexico
401	Canada

1. "Give me the total Sales of TVs in Canada, during 2016."
2. "Give me the total Sales in Canada, during Quarter 1 of 2016."

33

Typical OLAP Operations

- Roll up (drill-up): summarize data
  - *by "climbing up" hierarchy*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to more detailed*
- Slice and dice: *project and select*
- Pivot (rotate):
  - *Re-orient the cube, visualization, 3D to series of 2D planes*

- *(more later)*

34

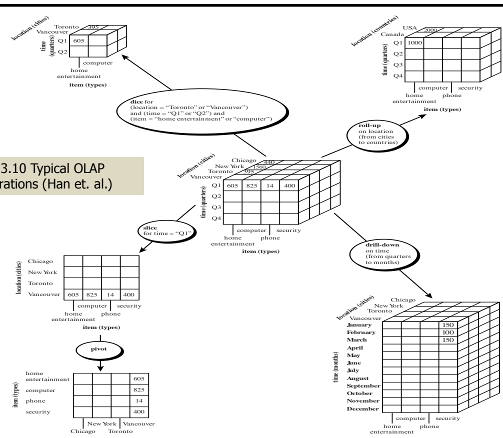


Fig. 3.10 Typical OLAP Operations (Han et. al.)

35

Aggregate goals and risks

Key issue: What aggregate to materialize (store)?

- Dramatic performance gains
- Reasonable extra data storage
- Transparent to users → aggregate navigation
- Benefit all users
- Low impact on data staging
- Low impact on DBA's workload

36

Main issue:
Deciding WHAT to aggregate

Choice will change periodically → different user needs

- Business needs, queries
  - What attributes are frequently used for grouping?
  - Which attributes are used together?
  - Beware of too many aggregates!
- Statistical distribution of data
  - 3 attributes & 4 dimensions → 256 possible aggregates

37

The aggregate table plan:
Number of rows (Date and Product)

Level	Count
Day	1,826
Month	60
Quarter	20
Year	5
Total	1

Level	Count
SKU	2,023
Product	723
Brand	44
Category	15
Total	1

38

The aggregate table plan:
Find high impact aggregates

- What about e.g. Month and Brand?
  - Month cuts about 1/30 of the detail size
  - Brand cuts to about 1/50 of the detail size
    - E.g. select 2,640 rows for aggregate instead of 3,693,998 from detail
  - Product aggregate useful if reporting on product level
  - etc.

39

Application Issues: The Aggregate Navigator

- GOAL: Transparently intercepts the end user code and uses the best aggregate possible
- Often part of OLAP engines' query optimization

e.g. Food, Drink, Stationary, Homeware, etc.

Partial pseudo code

```
Select   Category, Sum(Sales-dollars)
From     Sales_fact, dim-tables
Where    Date = Jan 2, 2002 AND
         City = "Ottawa" and {other PK joins}
Group by Category;
```

```
Select   Category, Sales-dollars
From     Category_Sales_fact, dim-tables
Where    Date = Jan 2, 2002 and
         City = "Ottawa" AND {other PK joins}
Group by Category;
```

40

The Aggregate Navigation Strategy: How does it work? (VERY high level)

1. Rank order all the aggregate fact tables for the smallest to the largest. (Cuboids)
2. Find the smallest aggregate fact table and proceed to step 2.
3. For the smallest, see if all the dimensional attributes of the query can be found.
   1. If yes, we are done.
   2. If not, find the next smallest aggregate fact table and retry step 2.
4. Execute the altered SQL. (If no aggregate fact tables found, use the Base Cuboid.)

```
Select   Category, Sum(Sales-dollars)
From     Sales_fact, dim-tables
Where    Product = "Milk" AND
         City = "Ottawa" and {other PK joins}
Group by Category;
```

41

Aggregates: A Recipe

1. Identify set of frequent queries
2. Identify concept hierarchies used (in queries in 1)
3. Determine levels in concept hierarchies to be used to speed up the queries (month, year)?
4. Decide on initial set of aggregates
5. If your system allows:

a) Implement aggregate strategy and aggregate navigator (e.g. write the code) (*or*)

b) Verify appropriateness of actual aggregates used in OLAP cube engine (if allowed by system)

6. Monitor and adapt

42

The bottom line

- Aggregates are “behind the query usage scenes”
- As important as indexes
- Transparent to end users and application developers
- DBA adds or remove aggregates, even on hourly basis
  - Uses query usage statistics
  - E.g. if a group of queries are slow; build a new aggregate
- A good aggregate strategy make life simple for the DBA; no more “fighting with aggregates”

43

In summary… A good aggregate strategy: The benefits

- Speed up queries by factor 100 → 1000
- Use a reasonable amount of extra disk space
- Completely transparent to users
- Benefit all users
- Low impact on data extract system
- Low impact on DBA

44

Next: A word about indexing

“Completing the physical design”

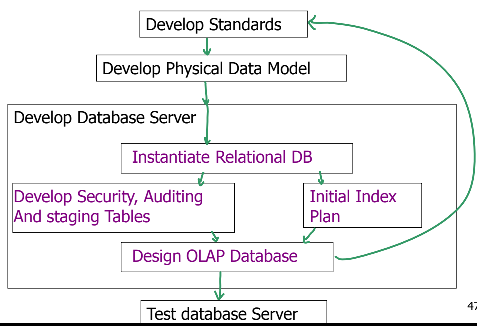
45

Completing the Physical design

- Steps to convert a logical design to a physical design
  1. Develop naming and database standards
  2. Create physical model
  3. Review aggregate table plan
  4. Create initial index strategy
  5. Create database instance
  6. Create storage structure
  7. Monitor the usage

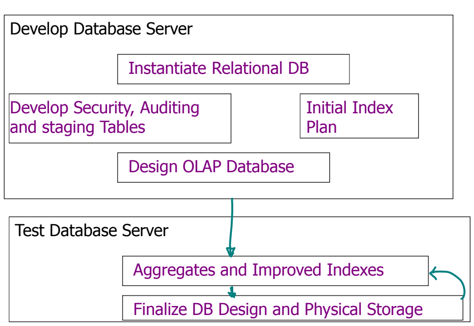
46

The high-level physical design process



47

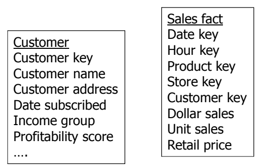
The high-level physical design process (cont.)



48

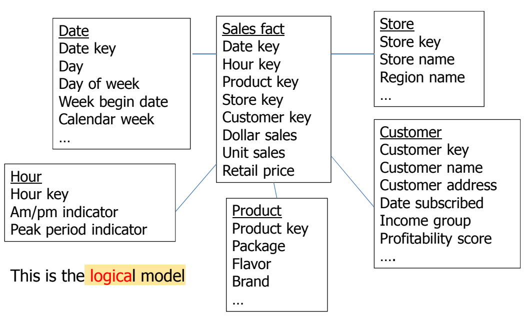
Developing the Physical Model (and Reviewing the Aggregate table plan)

- Starting point: dimensional (logical) model
- What is the major difference between the logical and physical models?
  - Detailed specs of physical DB characteristics:
    - Data types
    - Table segmentation
    - Table organization
    - Table storage parameters
    - Disk page size
    - Buffer size
    - Etc.



49

Developing the physical model: Beverage store example



This is the logical model

50

Developing the physical model: Beverage store example (cont): `create tables`

Table/column name	**Data type**	**Nulls ?**	**PK**	**Comment**
Date				**Date dimension**
Date_key	integer	n	1	Surrogate key B+ index
Day_date	date	n		Date, can be used for date arithmetic
Day_of_week	varchar(9)	n		Weekday, e.g. "Monday"
Week_begin_date	Date	n		Date of this week's Monday
Calender_week	integer	n		Takes values 1..52. Week 1 begins first Monday of year

51

Developing the physical model: Beverage store example (cont)

Table/column name	Data type	Nulls?	PK	Comment
Customer				**Customer dimension**
Customer_ key	integer	n	1	Surrogate key; most occur with = 0 (no Reward Card) B+ index
Customer_ last_name	Varchar(50)	n		Customer last name; limited user access
Customer_ first_name	Varchar(50)	n		Customer first name; limited user access
Gender	Char(1)	y		Values M, F, U Use bitmap index

52

Developing the physical model: Beverage store example (cont)

Table/column name	Data type	Nulls?	PK	Comment
Sales_Fact				**Fact table, with sales by store, day, hour, customer (if known) and product**
Date_key	integer	n	1	Foreign key to date.date_key
Customer_key	integer	n	2	Foreign key to customer.customer_key
Product_key	integer	n	3	Foreign key to product.product_key

53

Developing the physical model: Beverage store example (cont)

Table/column name	Data type	Nulls?	PK	Comment
Hour_key	integer	n	4	Foreign key to hour.hour_key
Store_key	integer	n	5	Foreign key to store.store_key
Dollar_sales_ amount	Number(11,2)	n		Canadian Dollar amount of sold item, value > 0
Quantity sold	integer	n		Quantity sold, value > 0
Retail_price	Number(11,2)	n		Price per item, value > 0

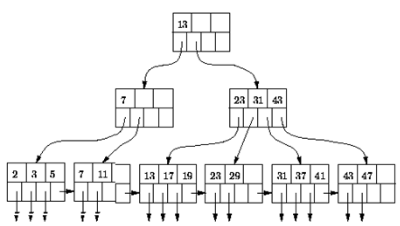
54

Indexing…

- B+ tree indexes on primary keys
- Clustered versus unclustered
- Bitmap indexes
- Indexes for n-way joins (star joins)

55

Recall from CSI2132: B+ tree

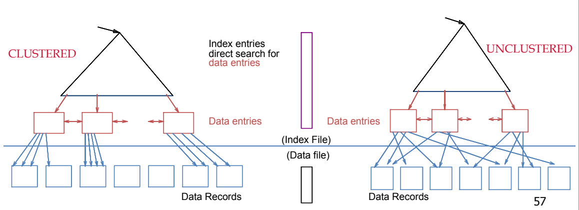


Used for Primary Keys

56

Recall from CSI2132: Clustered vs. Unclustered Indexes

- Question: What primary key to cluster on, if any, in the FACT table…
- Date usually first!



57

Bitmap indexing for Gender field

Records	1	50,000,000
Female	0 1 1 0 0 0 0 0 0 1 0 0 0 1	
Male	1 0 0 1 1 0 0 0 0 1 0 1 0	
Undisclosed	0 0 0 0 0 1 1 1 0 0 1 0 0	

For columns with low cardinality

58

Using Bitmap Indexes

Query:
Get product-key of products with size=2 and name ='Coke'

1. Bit map for size = 2: 11011000000000000000
2. Bit map for name='Coke': 01000110000000000010
3. Answer is intersection: 01000000000000000000

- Good if domain cardinality is small
- Bit vectors can be compressed

Joining relations

Recall from CSI3130 that JOINs are expensive (in terms of I/Os)
Toy example with only two relations

```
Select *
From Reserve R, Sailor S
Where R.sid = S.sid;

Sailor (SID, Name, Rating)
Reserve (SID, BID, Day)
```

60

Toy Example

SID	Sname	Rating
102	Ann	3
105	James	5
100	Joe	1
103	John	4
101	Sue	5
104	Sue	5

SID	BID	Date
105	Boat 1	13-Jan-17
100	Boat1	01-Jan-17
103	Boat1	14-Feb-17
105	Boat10	01-Jan-17
105	Boat10	09-Jan-17
100	Boat2	10-Jan-17
104	Boat2	13-Feb-17
105	Boat3	20-Jan-17
100	Boat3	14-Feb-17
104	Boat5	10-Jan-17

- What is the estimated cost (in terms of I/Os)?

61

Sort Merge Join (R ⋈ $_{i=j}$ S)

- Sort R and S on the join column
- Scan R and S to do a ``merge'' (on join col.), and output result tuples.
  - Advance scan of R until (current R-tuple >= current S) tuple,
    Then advance scan of S until (current S-tuple >= current R) tuple;
    Do this until (current R tuple = current S) tuple.
  - At this point, all R tuples with same value in Ri (*current R group*) and all S tuples with same value in Sj (*current S group*) *match*;
    Output <r, s> for all pairs of such tuples.
  - Then resume scanning R and S.
- R is scanned once; each S group is scanned once per matching R tuple.
- Multiple scans of an S group are likely to find needed pages in buffer.

(Covered in CSI3130)

62

Sorted

SID	Sname	Rating
100	Joe	1
101	Sue	5
102	Ann	3
103	John	4
104	Sue	5
105	James	5

SID	BID	Date
100	Boat1	01-Jan-17
100	Boat2	10-Jan-17
100	Boat3	14-Feb-17
103	Boat1	14-Feb-17
104	Boat5	10-Jan-17
104	Boat2	13-Feb-17
105	Boat10	01-Jan-17
105	Boat10	09-Jan-17
105	Boat 1	13-Jan-17
105	Boat3	20-Jan-17

- Cost for sorting each relation: N log N, where is the # of tuples
- Cost for scanning: M+N, where M is the size of R and N is the size of S

63

Scanning

SID	Sname	Rating
100	Joe	1
101	Sue	5
102	Ann	3
103	John	4
104	Sue	5
105	James	5

SID	BID	Date
100	Boat1	01-Jan-17
100	Boat2	10-Jan-17
100	Boat3	14-Feb-17
103	Boat1	14-Feb-17
104	Boat5	10-Jan-17
104	Boat2	13-Feb-17
105	Boat10	01-Jan-17
105	Boat10	09-Jan-17
105	Boat 1	13-Jan-17
105	Boat3	20-Jan-17

- Total cost: N log N + M log M + (M + N) I/Os
- Recall that Disk to Buffer Transfer Costs are high; we also may have to do some External Sorting!

64

Total Cost for Sort-Merge join

- Total cost = N log N + M log M + (M + N) I/Os
- Recall the Scan involves disk to buffer transfers: (M+N) I/O s (may approach M*N but very unlikely if the buffer is large enough)!
- Imagine a real-world (snowflake) where:

  M (number of suppliers) = 100,000,000 and

  N (number of products) = 5,000,000,000

  That is, a supplier supplies on average 50 items

  Cost: approximately 54,400,000,000 I/Os

item
item_key
item_name
brand
type
supplier_key

supplier
supplier_key
supplier_type

65

Total cost for n-way Sort-Merge Join: Huge!

Total = (F log F + P log P + (F + P)) + (F log F + D log D + (F + D)) + (F log F + S log S + (F + S))

where F = size of Sales Fact, P = size of Product, D = size of Date and S = size of Store

Date Dimension
Date Key (PK)
Date attributes

Daily Sales Fact Table
Date Key (PFK)
Product Key (PFK)
Store Key (PFK)
Promotion Key (PFK)
Quantity Sold
Dollars Sold

Product Dimension
Product Key (PK)
Product attributes

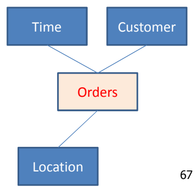
Store Dimension
Store Key (PK)
Store attributes

66

Star Join Optimization

- Attacks the n-way join problem in a star join
- Idea:
  - Start with the dimensions with conditions on them
  - Create list of key combinations that meet this condition
  - Extract the appropriate data from the Fact

```
Select sum(totalorders)
From <tables>
Where date = today
And city = 'Ottawa'
And <foreign key links>
```



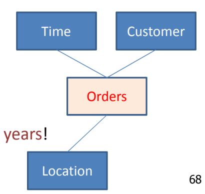
67

Star Join Optimization

- Attacks the n-way join problem in a star join
- Intuition:
  - Queries are selective
  - We need to reduce the number of rows we need to join
  - Push down the selects

```
Select sum(totalorders)
From <tables>
Where date = today
And city = 'Ottawa'
And <foreign key links>
```

Consider Date = today versus joining 10 years!



68

Star Join Optimization (an example)

- Semijoins
  - return the row-identifiers that match the query (in each dimension)
- Use a bitmap index to AND the results
- Complete the query
- Used in DB2, Oracle and MS SQL Server

69

Schematic… the general idea

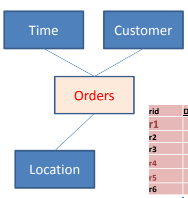
- P1 is not frequent; used to PRUNE the space prior to joining

Product	Id	Name	Price	jindex
	P1	Bolt	10	r1,r3,r5,r6
	P2	Nut	5	r2,r4

Sale	rid	Prodid	Storeid	Date	Amt
	r1	P1	C1	1	12
	r2	P2	C1	1	11
	r3	P1	C3	1	50
	r4	P2	C2	1	8
	r5	P1	C1	2	44
	r6	P1	C2	2	4

70

Example: Semi-joins and bitmap indexes



Cust-key	Name	Age	I-rid
100	Joe	20	r1
101	Ann	22	r2
102	Sue	34	r6
103	Ann	12	r5
104	Mark	17	r3
105	Joey	71	r4

rid	Day-key	Loc-key	Cust-key	$amount
r1	3012	200	100	450
r2	3013	201	101	400
r3	3013	201	103	200
r4	3012	200	105	200
r5	3017	200	103	300
r6	3014	202	102	100

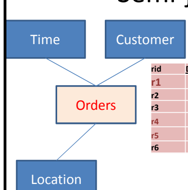
Day-key	Day	Month	Year	I-rid
3012	12	October	2016	r1, r4
3013	13	October	2016	r1
3014	14	October	2016	r3, r6
3015	15	October	2016	r5, r10
3016	16	October	2016	…
3017	17	October	2016	
3018	18	October	2016	
3019	19	October	2016	
3020	20	October	2016	
3021	21	October	2016	
3022	22	October	2016	
3023	23	October	2016	
3024	24	October	2016	

Loc-key	City	I--rid
200	Ottawa	r1, r4, r5
201	Toronto	r2
202	Vancounver	r3, r6, r7, r8

Find the total $ of orders for customers shopping in **Ottawa**, for 12 October 2013

71

Example: Semi-joins and bitmap indexes



Cust-key	Name	Age	I-rid
100	Joe	20	r1
101	Ann	22	r2
102	Sue	34	r6
103	Ann	12	r5
104	Mark	17	r3
105	Joey	71	r4

rid	Day-key	Loc-key	Cust-key	$amount
r1	3012	200	100	450
r2	3013	201	101	400
r3	3013	201	103	200
r4	3012	200	105	200
r5	3017	200	103	300
r6	3014	202	102	100

Loc-key	City	I--rid
200	Ottawa	r1, r4, r5
201	Toronto	r2
202	Vancounver	r3, r6, r7, r8

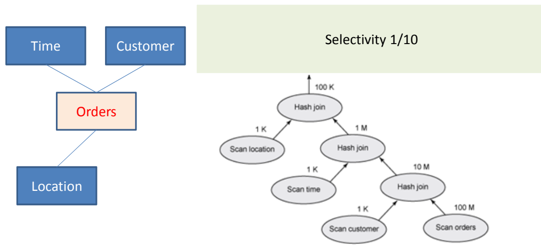
Day-key	Day	Month	Year	I-rid
3012	12	October	2016	r1, r4
3013	13	October	2016	r1
3014	14	October	2016	r3, r6
3015	15	October	2016	r5, r10
3016	16	October	2016	…
3017	17	October	2016	
3018	18	October	2016	
3019	19	October	2016	
3020	20	October	2016	
3021	21	October	2016	
3022	22	October	2016	
3023	23	October	2016	
3024	24	October	2016	

Bitmap AND yields: R1, R4

Find the total $ of orders for customers shopping in **Ottawa**, for 12 October 2013

72

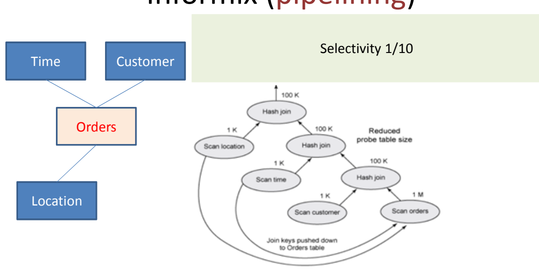
Star join optimization using hash join: Informix




```
SELECT t.quarter, sum(o.quantity)
FROM orders o, location l, customer c, time t
WHERE o-cust_num = c.cust_num  AND o.date_time = t.date_time AND c.type <= 3
     AND l.region_id = 4 AND t.quarter between 1 and 20
GROUP by t.quarter
```

73

Star join optimization using hashing: Informix (pipelining)

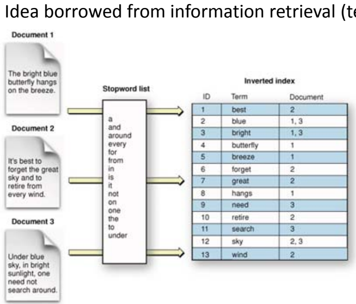



```
SELECT t.quarter, sum(o.quantity)
FROM orders o, location l, customer c, time t
WHERE o-cust_num = c.cust_num  AND o.date_time = t.date_time AND c.type <= 3
     AND l.region_id = 4 AND t.quarter between 1 and 20
GROUP by t.quarter
```

74

Pruning one dimension: Inverted indexes

- Idea borrowed from information retrieval (text mining)



75

Inverted Index: Product Dimension

Index on one or more attribute

Query: Get the products with size = 2 (liters) and name ='Milk'

1. Use size index and retrieve ids for 2l: r4, r18, r32, r34, r35
2. Use name index and retrieve ids for Milk: r18, r32, r52
3. Answer is intersection: r18, r32

76

Star joins: summary

- Goal is to reduce the number of rows from the dimensions prior to joining
- Use idea of SELECTIVITY (from query optimization)
- Use of reducers: semi-joins, bitmaps or inverted indexes
- May use ´traditional` join algorithms on the pruned space

77

Designing the OLAP Database

- Depends on your OLAP technology
- Typical current capacity: Up to 2,100,000,000 dimensions and measures!
- MOLAP - Multidimensional OLAP - Both fact data and aggregations are processed, stored, and indexed using a special format optimized for multidimensional data (some disadvantages).
- ROLAP - Relational OLAP - Both fact data and aggregations remain in the relational data source, eliminating the need for special processing.
- HOLAP - Hybrid OLAP - This mode uses the relational data source to store the fact data, but pre-processes aggregations and indexes, storing these in a special format, optimized for multidimensional data.
- Commercial: https://en.wikipedia.org/wiki/Comparison_of_OLAP_Servers
- Open Source DBs: PostgreSQL and MySQL also offer OLAP databases

78

Next

Data staging: Extracting, Converting and Loading the data

79