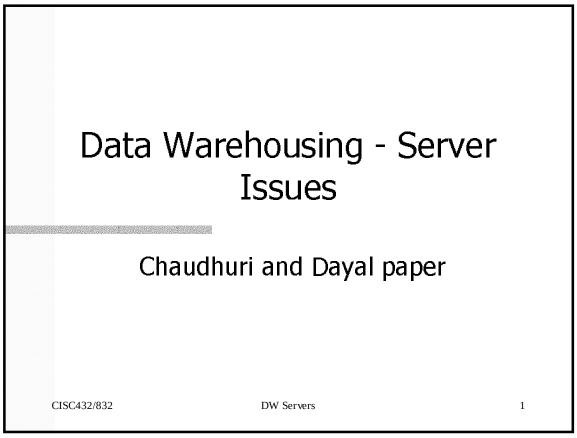
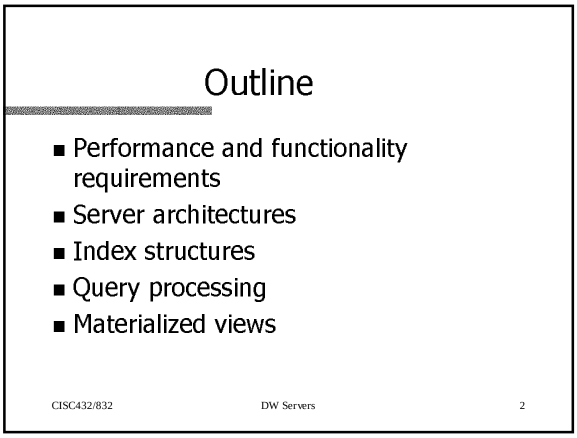


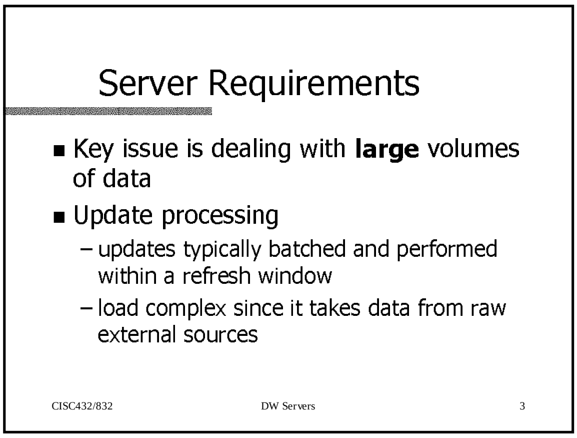
Data Warehousing - Server Issues

Chaudhuri and Dayal paper



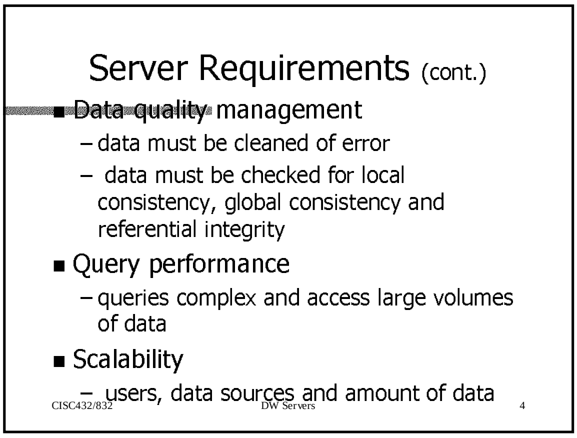
Outline

- Performance and functionality requirements
- Server architectures
- Index structures
- Query processing
- Materialized views



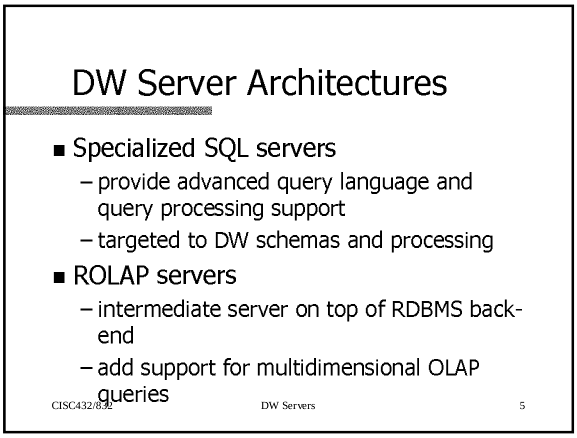
Server Requirements

- Key issue is dealing with **large** volumes of data
- Update processing
  - updates typically batched and performed within a refresh window
  - load complex since it takes data from raw external sources



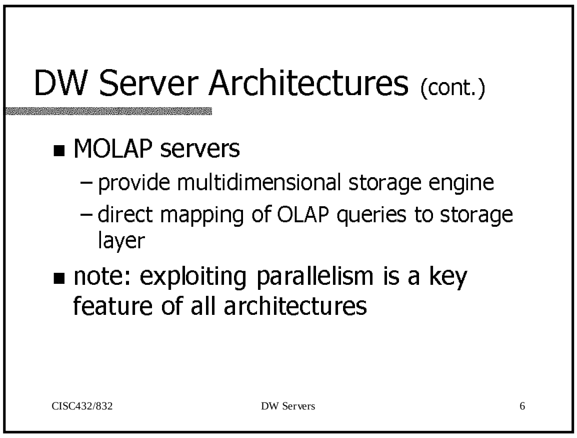
Server Requirements (cont.)

- Data quality management
  - data must be cleaned of error
  - data must be checked for local consistency, global consistency and referential integrity
- Query performance
  - queries complex and access large volumes of data
- Scalability
  - users, data sources and amount of data



DW Server Architectures

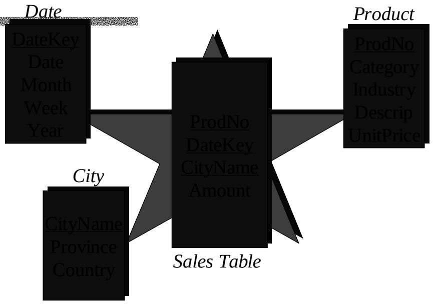
- Specialized SQL servers
  - provide advanced query language and query processing support
  - targeted to DW schemas and processing
- ROLAP servers
  - intermediate server on top of RDBMS back-end
  - add support for multidimensional OLAP queries



DW Server Architectures (cont.)

- MOLAP servers
  - provide multidimensional storage engine
  - direct mapping of OLAP queries to storage layer
- note: exploiting parallelism is a key feature of all architectures

Example Star Schema

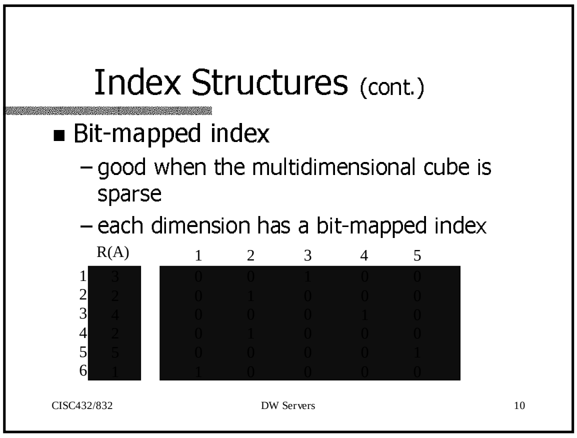


Typical OLAP Queries

- Give total sales for each product in each quarter of 1995.
- In 1996, for each city give the products with the top 5 sales.
- Give the average monthly sales of computer instruction books in each of the last 3 years for each city in Canada.

Index Structures (cont.)

- Indexes may be very large
  - need both space and time efficient organizations
- Queries may often involve multiple indexed attributes
  - need organizations that support efficient processing

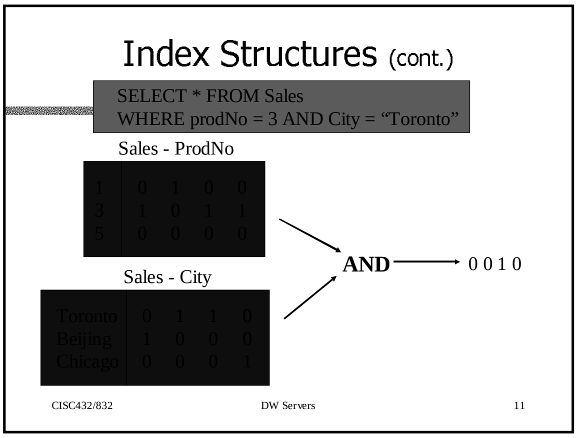


Index Structures (cont.)

- Bit-mapped index
  - good when the multidimensional cube is sparse
  - each dimension has a bit-mapped index

	R(A)	1	2	3	4	5
1	3	0	0	1	0	0
2	2	0	1	0	0	0
3	4	0	0	0	1	0
4	2	0	1	0	0	0
5	5	0	0	0	0	1
6	1	1	0	0	0	0

CISC432/832 DW Servers 10



Index Structures (cont.)

```
SELECT * FROM Sales
WHERE prodNo = 3 AND City = “Toronto”
```

Sales - ProdNo

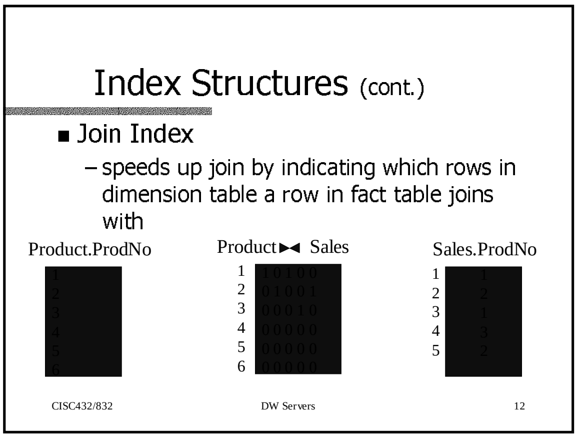
1	0	1	0	0
3	1	0	1	1
5	0	0	0	0

Sales - City

Toronto	0	1	1	0
Beijing	1	0	0	0
Chicago	0	0	0	1

AND → 0 0 1 0

CISC432/832 DW Servers 11



Index Structures (cont.)

- Join Index
  - speeds up join by indicating which rows in dimension table a row in fact table joins with

Product.ProdNo

1
2
3
4
5
6

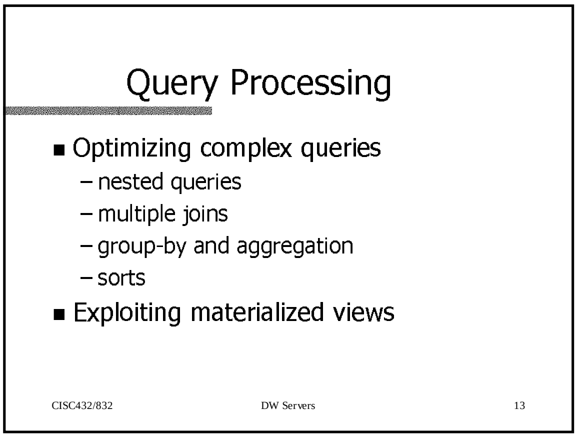
Product ⋈ Sales

1	1 0 1 0 0
2	0 1 0 0 1
3	0 0 0 1 0
4	0 0 0 0 0
5	0 0 0 0 0
6	0 0 0 0 0

Sales.ProdNo

1	1
2	2
3	1
4	3
5	2

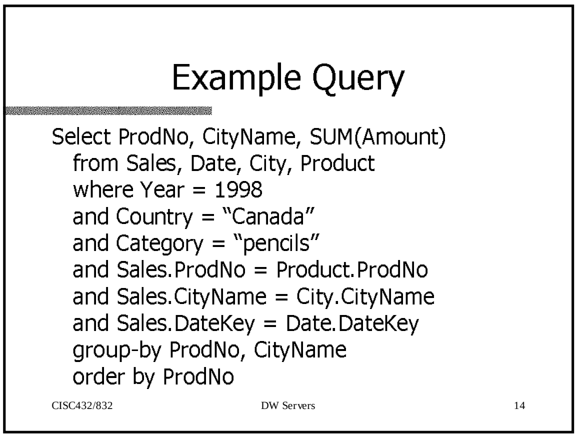
CISC432/832 DW Servers 12



Query Processing

- Optimizing complex queries
  - nested queries
  - multiple joins
  - group-by and aggregation
  - sorts
- Exploiting materialized views

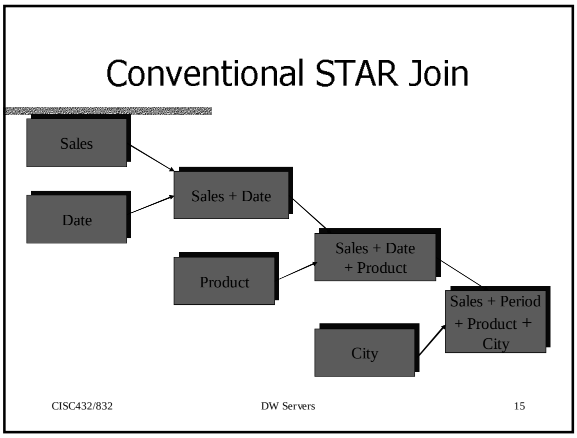
CISC432/832 DW Servers 13



Example Query

```
Select ProdNo, CityName, SUM(Amount)
  from Sales, Date, City, Product
  where Year = 1998
  and Country = "Canada"
  and Category = "pencils"
  and Sales.ProdNo = Product.ProdNo
  and Sales.CityName = City.CityName
  and Sales.DateKey = Date.DateKey
  group-by ProdNo, CityName
  order by ProdNo
```

CISC432/832 DW Servers 14



Conventional STAR Join

Sales

Date

Sales + Date

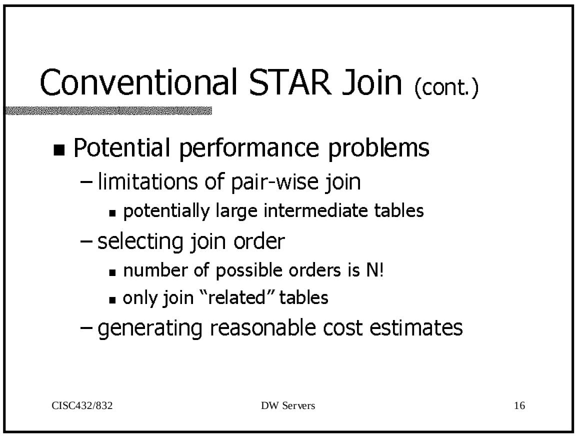
Product

Sales + Date + Product

City

Sales + Period + Product + City

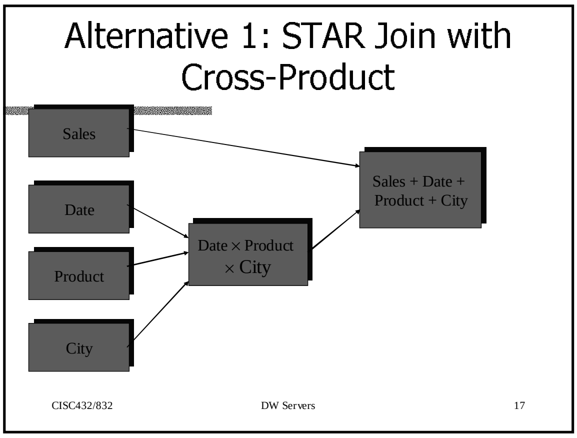
CISC432/832 DW Servers 15



Conventional STAR Join (cont.)

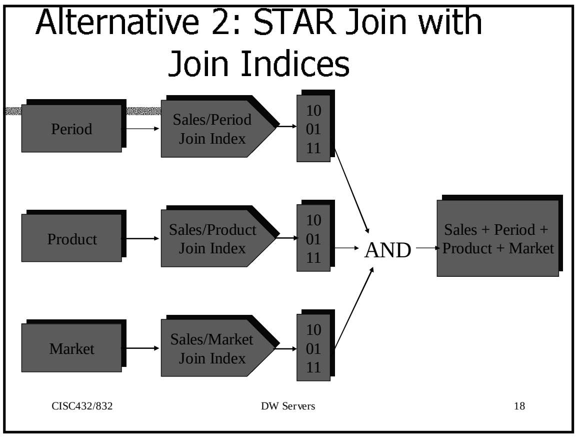
- Potential performance problems
  - limitations of pair-wise join
    - potentially large intermediate tables
  - selecting join order
    - number of possible orders is N!
    - only join "related" tables
  - generating reasonable cost estimates

CISC432/832 DW Servers 16



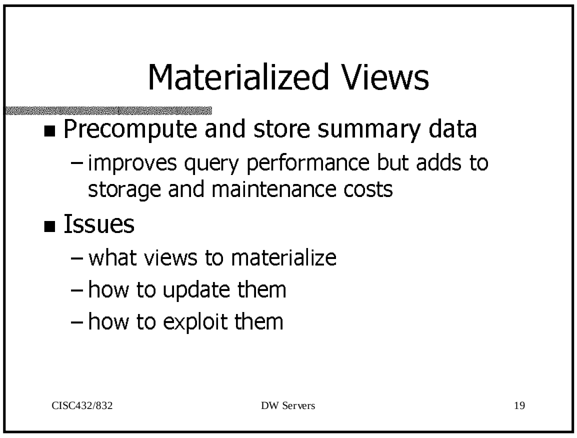
Alternative 1: STAR Join with Cross-Product

CISC432/832 DW Servers 17



Alternative 2: STAR Join with Join Indices

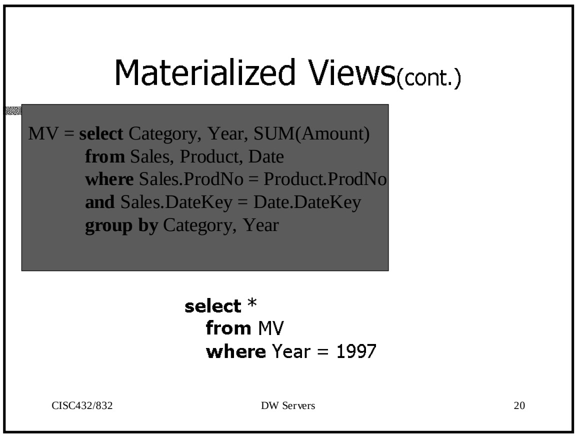
CISC432/832 DW Servers 18



Materialized Views

- Precompute and store summary data
  - improves query performance but adds to storage and maintenance costs
- Issues
  - what views to materialize
  - how to update them
  - how to exploit them

CISC432/832 DW Servers 19

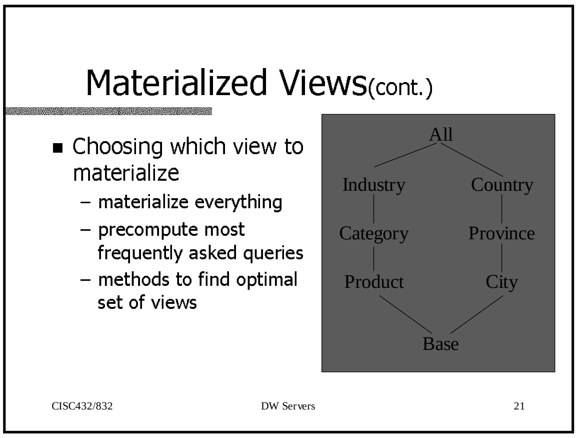


Materialized Views(cont.)

```
MV = select Category, Year, SUM(Amount)
        from Sales, Product, Date
        where Sales.ProdNo = Product.ProdNo
        and Sales.DateKey = Date.DateKey
        group by Category, Year
```

```
select *
   from MV
   where Year = 1997
```

CISC432/832 DW Servers 20



Materialized Views(cont.)

- Choosing which view to materialize
  - materialize everything
  - precompute most frequently asked queries
  - methods to find optimal set of views

CISC432/832 DW Servers 21

Materialized Views(cont.)

- How to update them?
  - Periodically recompute (snapshot view)
  - incrementally update views
- How to exploit them?
  - Optimizer must recognize where a materialized view can be substituted in a query and then query rewritten

MV Selection Problem

Given a set of queries

$Q = \{Q_1, Q_2, \ldots, Q_n\}$

choose a set of MVs

$V = \{V_1, V_2, \ldots , V_k\}$

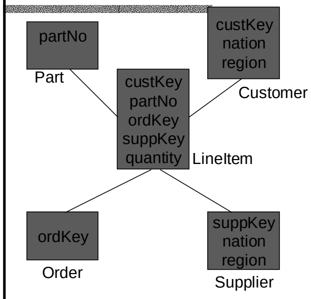
that minimize one or more of

storage costs

query processing costs

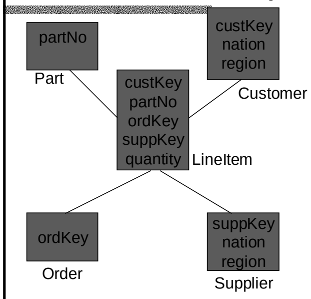
maintenance costs

An Example: TPC-D



```
Select partNo, custKey,
        SUM (quantity)
from LineItem
group by partNo, custKey
```

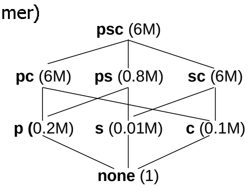
An Example (cont.)



Possible Views (group-bys)
(part, supplier, customer) (6M)
(part, customer) (6M)
(part, supplier) (0.8M)
(supplier, customer) (6M)
(part) (0.2M)
(supplier) (0.01M)
(customer) (0.1M)
(none) (1)

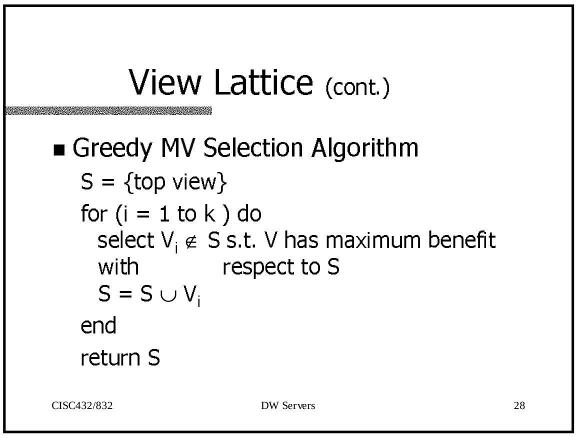
View Lattice Approach

- Dependence relation on views
  - $V_1 \leq V_2$ iff V_1 can be answered using only results of V_2
  - e.g. (part) ≤ (part, customer) but (part) ≤/ (customer)



View Lattice (cont.)

- To answer a query Q_i
  - choose ancestor of Q_i in lattice, Q_A , that is materialized
  - cost of processing Q is number of rows in Q_A
- benefit of materializing view V is amount by which it improves costs of queries in Q



View Lattice (cont.)

- Greedy MV Selection Algorithm

```
S = {top view}
for (i = 1 to k ) do
  select $V_i \notin$ S s.t. V has maximum benefit
  with            respect to S
  S = S $\cup$ $V_i$
end
return S
```

CISC432/832 DW Servers 28