DB Performance and Tuning

Objectives

- Introduction +
- DB Performance +
- Tuning Indexes +
- Tuning the DB Design +
- Tuning the Queries +
- Tuning Views +

- Introduction ...

- Every DB system is developed to fulfill the data requirements of its users and application programs.
- All the workload numbers used during the physical design process are normally good estimates of the expected actual numbers.
- After a DB is deployed and is in operation, actual use of the applications, transactions, queries, and view reveals factors and problem areas that may not have been accounted for during the initial physical DB design.

... Introduction

- The estimated values used in the physical design can be revised by gathering actual statistics about usage patterns.
- Resource utilization as well as internal DBMS processing can be monitored to reveal bottlenecks.
- It is therefore necessary to monitor and revise the physical DB design constantly.

- DB Performance

- Two parameters normally used to measure the performance of a DB system are:
  - Response time
  - Throughput
- The performance of a DBMS on commonly asked queries and typical update operations is the ultimate measure of a DB design.
- A DBA can improve the performance by identifying bottlenecks and adjusting some DBMS parametrs

- DB Tuning ...

- DB tuning implies that the physical DB design be reviewed and modified to accommodate the changes in the requirements and to overcome the DB performance bottleneck.
- The goals of DB tuning can include:
  - To make application run faster
  - To lower the response time of queries
  - To improve the overall throughput of transactions
- During DB tuning, the design decisions made in the physical DB design are revisited and adjusted according to the up-to-date data and performance requirements.

… - DB Tuning

- Specific actions taken to tune a DB are very much DBMS and operating system dependent. A DBA must be aware of these actions.
- Continued DB tuning is important to get the best possible DB performance.
- Three kinds of DB tuning are:
  1. Tuning indexes +
  2. Tuning the DB design +
  3. Tuning the queries and views +
- Other factors for DB performance improvement are:
  - Review of log files
  - Data archiving

-- Tuning Indexes ...

- Existing indexes has to be to be revised for the following reasons.
  - Certain queries may take too long to run for lack of an index
  - Certain indexes may not get utilized at all
  - Certain indexes may be causing excessive overhead
- The activities for tuning indexes can be
  - Re-examining our choices of indexes
  - Periodically reorganizing some indexes
  - Dropping an index and rebuilding it.

… -- Tuning indexes

- Query optimizers generally rely on the statistics maintained in the system catalog. It is the responsibility of the DBA to make sure that the statistics are kept up-to-date.
- How to tune indexes:
  - Most DBMSs have a command or trace facility
  - Analyze the results of trace facility
  - Dropping and building new indexes

-- Tuning the DB Design

- If a given physical DB design does not meet the expected objectives we may revert to the logical design, make adjustments to the logical schema, and remap it to a new set of physical tables and indexes.
- The entire DB design has to be driven by the processing requirements as much as by data requirements.
- The changes to the logical schema may be of the following nature:
  - Existing tables may be joined (denormalized)
  - Vertical partitioning +
  - Horizontal partitioning +
  - Attribute(s) from one table may be repeated in another.

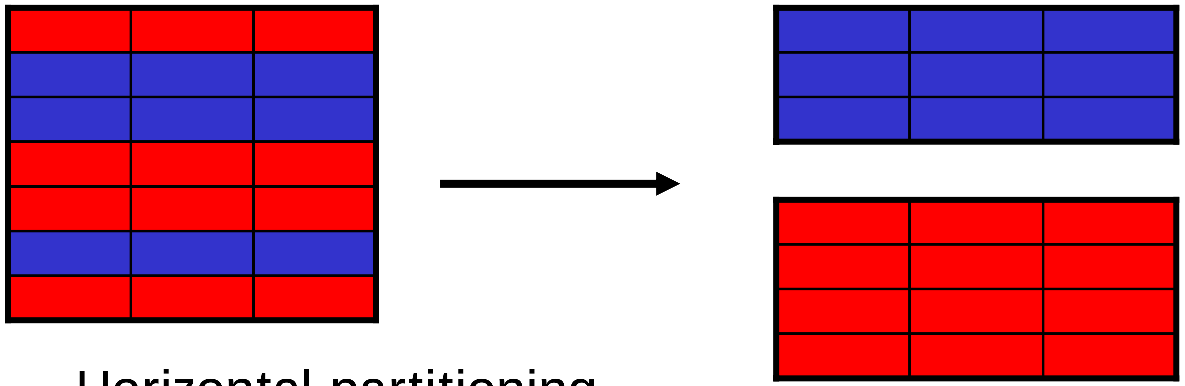
--- Vertical Partitioning

- In vertical partitioning, a relation of the form R(K, A, B, C, D, ...) – with K as a set of key attributes – that is in BCNF can be stored into multiple tables that are also in BCNF.
- For example:
  - The table EMPLOYEE (SSN, Name, Phone, Grade, Salary) may be split in to two tables:
    1. EMP1 (SSN, Name, Phone)
    2. EMP2 (SSN, Grade, Salary)
  - If the original table had a very large number of rows and queries about phone numbers and salary information are totally distinct, this separation of tables may work better.

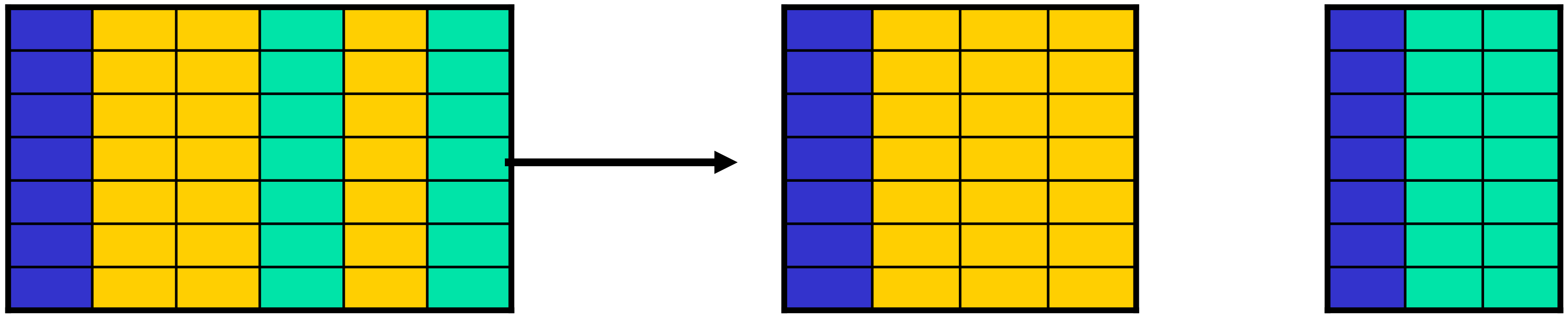
--- Horizontal Partitioning

- Horizontal partitioning takes horizontal slices of a table and stores them as distinct tables.
- For example:
  - Product sales data may be separated into ten tables based on ten products lines. Each table has the same set of columns but contain a distinct set of products (tuples)

--- Examples of H and V partitioning



Horizontal partitioning



Vertical partitioning

-- Tuning Queries ...

- There are mainly two indications that suggest that query tuning may be needed:
  1. A query issues too many disk accesses
  2. The query plan shows that relevant indexes are not being used
- A query plan is a sequence of function calls where each function call implements an operator like SELECT, PROJECT, JOIN, sort, scan, etc. It can be represnted as annotated tree.
- As an example, plan for computing R * P using nested loops approach may look like:

Natural-Join (nested-loop(sort(scan(R)), sort(scan(P)))

… -- Tuning Queries …

- Some typical instances of situations prompting query tuning include the following:
  - Many query optimizers do not use indexes in the presence of:
    - arithmetic operations, such as Salary/365 > 10.5
    - Comparing attributes of different sizes and precision
    - NULL Comparisons
    - Substring comparisons
  - Indexes are often not used for nested queries
  - Some sorting can be avoided
  - Views can have negative affect on performance hence replace them by base tables in the query.

... -- Tuning Queries ...

- A query with multiple selection conditions that are connected via OR may not prompt the query optimizer to use any index. Such a query may be split up and expressed as a UNION of queries, each with a condition on an attribute that causes an index to be used.
- For example:

```
SELECT Fname, Lname, Salary, Age
FROM Employee
WHERE Age > 45 OR Salary > 50000;
```

May be executed using sequential scan giving poor performance. Splitting it up as:

```
SELECT Fname, Lname, Salary, Age
FROM Employee
WHERE Age > 45
UNION
SELECT Fname, Lname, Salary, Age
FROM Employee
WHERE Salary < 50000
```

May utilize index on Age as well as on Salary

… -- Tuning Queries

- Where conditions may be rewritten to utilize the indexes on multiple.
- For example:

```
SELECT Region#, Prod_type, Month, Sales
FROM Sales_Statistics
WHERE Region# = 3
AND ((Product_type BETWEEN 1 AND 3)  OR
        (Product_type BETWEEN 8 AND 10))
```

May use index only on Region#. Instead, using:

```
SELECT Region#, Prod_type, Month, Sales
FROM Sales_Statistics
WHERE (Region# = 3 AND (Product_type BETWEEN 1 AND 3)) OR
        (Region# = 3 AND (Product_type BETWEEN 8 AND 10))
```

May use a composite index on(Region#, Product_type)

-- Tuning View

- Because the views are the named queries therefore the rules for tuning queries are also applicable to views.
- The first step in tuning a view is to understand the plan used by the DBMS to materialize the view.
- One we understand the plan selected by the system, we can consider how to improve it.