Tarefas de administração de banco de dados

Prof. Daniel S. Kaster

Departamento de Computação
Universidade Estadual de Londrina
dskaster@uel.br

Outline

- Planejar a infraestrutura
- Instalar o SGBD e ferramentas de suporte
- Criar usuários e definir seus perfis de acesso
- Aspectos de tuning de desempenho

Planejar a infraestrutura

- Definir a estratégia da organização para o armazenamento de dados
  - Evitar instalações de SGBD sob demanda, de acordo com as aplicações que vão surgindo
  - Evitar a heterogeneidade de SGBD
- Estimar custos de migração e treinamento de recursos humanos
- Planejamento a médio prazo e priorização de ações mediante limitações de investimento

Definir a estrutura de armazenamento

- Alternativas
  - Discos isolados
  - Storages

Discos isolados

- Balancear velocidade e capacidade
  - NVM SSDs
  - Discos de alta rotação (10000/15000 rpm)
  - Armazenamento near-line (7200rpm)

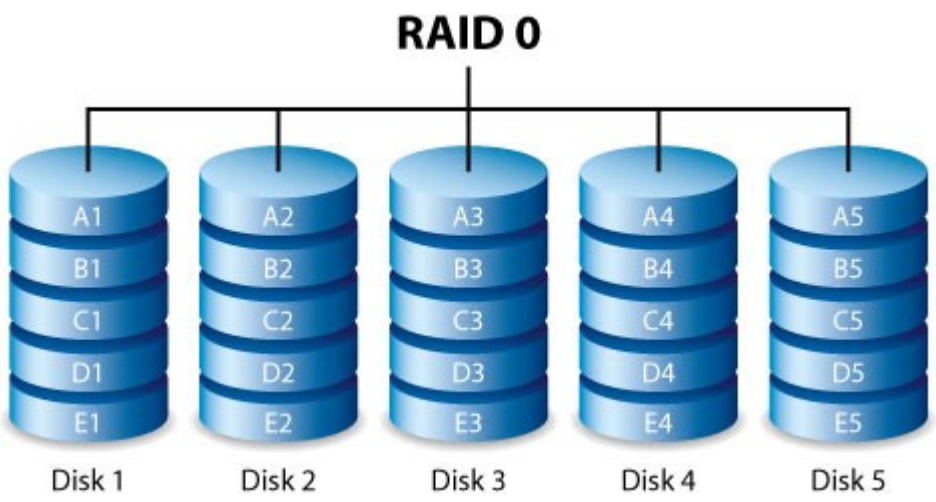
Storages

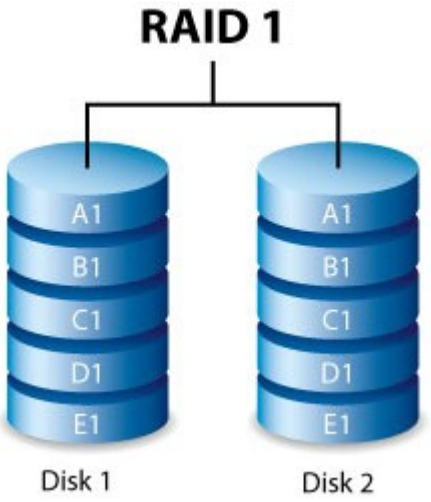
- Ex: Dell EqualLogic PS6210 Series
  - Até 144TB por array e até 2.3PB com 16 arrays
  - Conexões: 2 10GBASE-T com RJ45 e 2 10GbE SFP+ para cabeamento fibre ou twin-ax copper

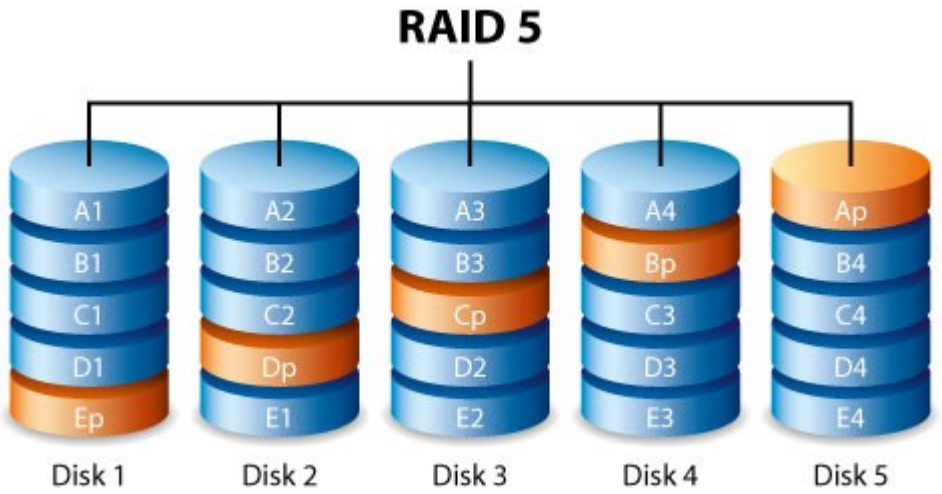
Uso de RAID

- O conceito de RAID (Redundant Array of Inexpensive/Independent Disks) consiste em organizar vários discos para uso em conjunto para aumentar desempenho e disponibilidade
  - Desempenho (stripping)
  - Redundância (espelhamento/paridade)

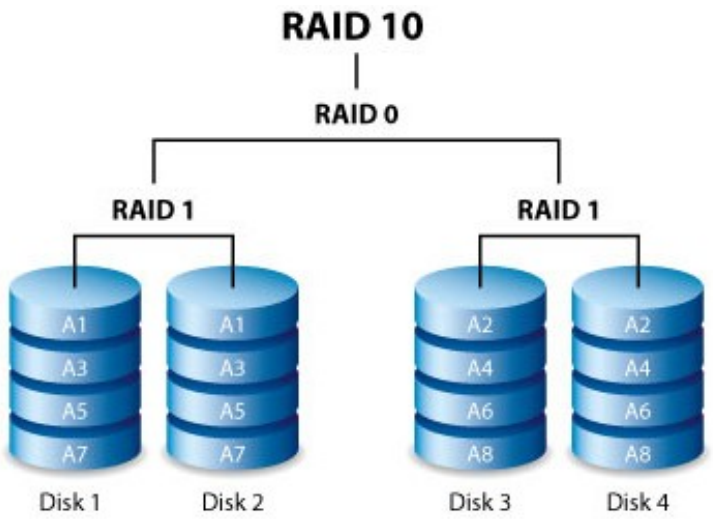
Tipos de RAID

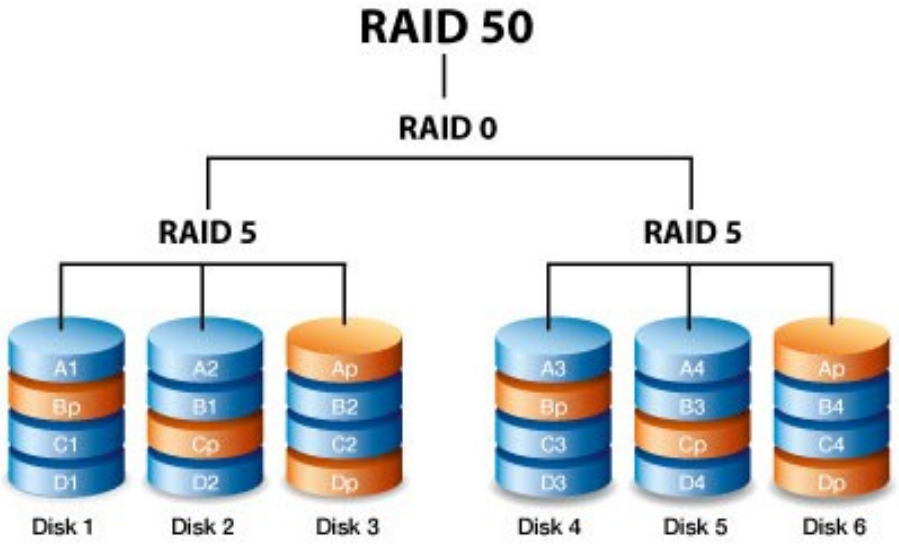






Tipos de RAID(2)





Instalar o SGBD e ferramentas de suporte

- Instalação inicial do SGBD
- Upgrades e atualizações requrerem uma avaliação de impacto sobre as aplicações em execução
- Muitas ferramentas administrativas não fazem parte do SGBD, são aplicações complementares
  - Dependem do pacote adquirido
- O suporte técnico normalmente entra na negociação
  - Isso vale para SGBD livres também

Definir o layout de armazenamento

- A tarefa de planejamento e gerência de armazenamento é importante na administração de bancos de dados
  - Dimensionar o espaço para armazenar as estruturas do banco de dados
  - Organizar informações de acordo com seu significado
  - Separar dados de sistema de dados de usuários
  - Balancear requisições de I/O entre dispositivos

Oracle Optimal Flexible Arquiteture

- Guidelines de nomenclatura e disposição de arquivos para as soluções da Oracle
  - Manutenção facilitada pelo uso de uma organização de arquivos padronizada
  - Maior confiabilidade pela distribuição dos dados em múltiplos discos
  - Maior desempenho e menor contenção de I/O
- Em um banco de dados
  - Visa agrupar informações relacionadas em diversos tablespaces

Principais tablespaces da OFA

- SYSTEM
  - Objetos do catálogo do sistema
- SYSAUX
  - Catálogo de pacotes complementares
- UNDOTBS
  - Segmentos de rollback mantêm a concorrência de dados no banco de dados através de um protocolo multi-versão
- TEMP
  - Segmentos temporários são objetos dinamicamente criados durante a execução de consultas (operações de ordenação, união, por exmplo)
  - Pela sua natureza dinâmica, não deve ser alocado juntamente com outros tipos de segmentos
- USERS
  - Objetos dos usuários ou aplicações

Limitando o acesso aos tablespaces de sistema

- Algumas ferramentas usam a conta SYSTEM para criação e manipulação de dados
- O tablespace default desta conta é o tablespace SYSTEM, o que “carrega” esse tablespace
- Usuários não devem poder inserir informações neste tablespace
  - Alter user SYSTEM quota 0 on SYSTEM;

Outros tablespaces

- DATA / <app_name>
  - Tabelas associadas a uma aplicação
  - Desta forma, o conteúdo do tablespace contém apenas dados relativos à aplicação
  - Diferentes tablespaces podem ter diferentes tamanhos de bloco, ajustados a cada caso

```
CREATE TABLE <table> (
  ...) TABLESPACE <tbls>...

CREATE INDEX <index>... TABLESPACE <tbls>
```

Carga e movimentação de dados

- Carga
  - Inserções por meio de scripts
  - Mecanismos de carga de dados
    - Oracle SQL*Loader
- Movimentação
  - Mecanismos de importação de dados em massa
  - Oracle DataPump (Import/Export nas versões mais antigas)

Criar usuários e definir seus perfis de acesso

- Tem que estar de acordo com a política de segurança da organização.
- É preciso definir padrões de acesso e de recursos que garantam a integridade dos dados e a segurança de dados confidenciais.

Mecanismos de controle de acesso

- Há dois tipos básicos de mecanismos de controle de acesso:
  - Discricionário: concessão de privilégios a usuários sobre objetos do banco de dados
  - Mandatório: definição de níveis de segurança para usuários e objetos para implementar a política de segurança da organização

Controle de acesso discricionário

- RBAC: Role-Based Access Control
- Concessão de privilégios em nível de:
  - Conta (ou sistema): create, alter, drop
  - Objetos: select, insert, delete, update, execute
- Conjuntos de privilégios são agrupados em atribuições para facilitar a administração
  - DBA, CONNECT, RESOURCE

Controle de acesso mandatório

- Cada usuário ou objeto está classificado em um nível
  - Top Secret (TS), Secret (S), Confidential (C) e Unclassified (U)
- O acesso aos dados é controlado de acordo com o nível de acesso do usuário e do objeto
- Raras implementações em SGBD comerciais

Controle de acesso mandatório(2)

- Fundamento
  - Um usuário U pode ler um objeto O somente se classe(U) >= classe(O)
  - Um usuário U pode escrever um objeto O somente se classe(U) <= classe(O)
    - Evita fluxo de informações de níveis de segurança superiores para níveis inferiores

- Atributos/tuplas sem permissão podem aparecer com null ou “poli-instanciados”

Mecanismos de controle de acesso no Oracle

- Controle de usuários
  - Usuários
  - Perfis
- Controle de acesso
  - RBAC (discricionário)
    - Privilégios
    - Atribuições
  - Oracle Label Security
    - Segue a ideia do controle de acesso mandatório

Perfis de usuário

- Usados para definir políticas, como de uso de memória/CPU/senhas/etc.

```
CREATE PROFILE clerk
  IDLE_TIME 20 FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LIFE_TIME 180
  PASSWORD_GRACE_TIME 5
  PASSWORD_REUSE_TIME 30;
```

Criando usuários

- A criação de usuários permite definir vários parâmetros

```
CREATE USER jward
IDENTIFIED BY aZ7bC2
DEFAULT TABLESPACE USERS
QUOTA 100M ON USERS
QUOTA 500K ON DATA
TEMPORARY TABLESPACE TEMP
PROFILE clerk;
```

Gerenciando usuários

- Habilitar e desabilitar contas

  ALTER USER jward ACCOUNT UNLOCK;

  ALTER USER willy ACCOUNT LOCK;

- Trocar senhas

  ALTER USER jward IDENTIFIED BY sh13jaWKi;

  ALTER USER jward IDENTIFIED BY VALUES ‘1AF2B59CD99AAD3F’

Gerenciando senhas

- O Oracle possui mecanismos para forçar a criação de senhas fortes
- Pode-se, ainda, modificar esses mecanismos
  - $ORACLE_HOME/rdbms/admin/utlpwdmg.sql
  - $ORACLE_HOME/rdbms/admin/catpvf.sql

Concessão de privilégios

- Privilégios de sistema
  - Grant <privilegio_sist>|<role> to <user>|<role> [with admin option];
  - Revoke <privilegio_sist>|<role> from <user>;

```
GRANT CREATE TABLE TO jward;
GRANT CREATE ANY DIRECTORY TO jward WITH ADMIN OPTION;
```

Concessão de privilégios(2)

- Privilégios de objetos
  - Grant <privilegio_obj> (<atributo>) on <esquema>.<objeto> to <user>|<role> [with grant option];
  - Revoke <privilegio_sist> on <esquema>.<objeto> from <user>;

```
GRANT SELECT, INSERT ON jward.project TO public;
GRANT SELECT, UPDATE(nome) ON jward.employee
TO willy WITH GRANT OPTION;
```

Atribuições

- Atribuições agrupam privilégios
- Podem obter privilégios por meio de outras atribuições
- Atribuições frequentemente usadas
  - CONNECT, RESOURCE, DBA, etc.

```
CREATE ROLE account_creator;
GRANT CREATE SESSION, CREATE USER, ALTER
USER TO account_creator;
GRANT CONNECT, account_creator TO jward;
```

Performance tuning

- Indexing
- Materialized views
- Partitioning

Indexing

- Indexes are essential to speed up queries
- However, they need to be carefully created and maintained to obtain maximum performance
  - Create an excessive amount indexes or keep indexes that are no longer useful harms the performance as they are updated with the correponding table
- Use the correct index type for each case
  - B-tree, bitmap, function-based, text search, etc.

Bitmap indexes

- Useful to index low selectivity attributes
  - Stores one bitmap per distinct value
- Multiple bitmap indexes allow checking query conditions using bit-level operations
- Suitable to Data Warehouses
  - More compact than B-trees
  - Data Warehouses are optimized for read, being updated by ETL batches

Syntax

```
CREATE [ UNIQUE | BITMAP ]
INDEX [ schema. ] index
  ON { table_index_clause
     | bitmap_join_index_clause
     }
[ UNUSABLE ] ;

table_index_clause
[ schema. ] table [ t_alias ]
(index_expr [ ASC | DESC ]
  [, index_expr [ ASC | DESC ] ]...)
  [ index_properties ]
```

```
bitmap_join_index_clause
[ schema.]table
   ( [ [ schema. ]table. | t_alias. ]column
     [ ASC | DESC  ]
       [, [ [ schema. ]table. |
t_alias. ]column
          [ ASC | DESC ]
       ]...
   )
   FROM [ schema. ]table [ t_alias ]
         [, [ schema. ]table [ t_alias ]
       ]...
   WHERE condition
      [ local_partitioned_index ]
index_attributes
```

Example of bitmap join index

```
CREATE BITMAP INDEX sales_c_gender_p_cat_bjix
ON sales(customers.cust_gender, products.prod_category)
FROM sales, customers, products
WHERE sales.cust_id = customers.cust_id
AND sales.prod_id = products.prod_id
LOCAL NOLOGGING COMPUTE STATISTICS;
```

```
SELECT …
WHERE customers.cust_gender = 'M' AND
products.prod_category...
```

Function-based indexes

- Used to index the result of functions
  - CREATE INDEX <ix_name> ON <table> (<function>);

```
SELECT *
FROM rmiexams rm
WHERE extractvalue(rm.image.metadata,
  '/DICOM_OBJECT/LONG_STRING[@tag="00081080" and
    @definer="DICOM"]/text()',
  'xmlns=http://xmlns.oracle.com/ord/dicom/metadata_1_0'
 ) = 'EPILEPSIA' ;

CREATE INDEX DICOM_diagnoses_ix ON rmiexams rm (
  extractvalue(rm.image.metadata,
   '/DICOM_OBJECT/LONG_STRING[@tag="00081080" and
     @definer="DICOM"]/text()',
   'xmlns=http://xmlns.oracle.com/ord/dicom/metadata_1_0')
);
```

Temporary tables

- Developers often have create a temporary table to do some work and drop it
- For this, many DBMS has a clause to define a temporary table
- Temporary tables usually have a particular implementation that benefit from their temporary nature
  - In-memory storage is usually welcome
- They are also handy when the user is in a situation where she needs the data to be visible only in the current session

Temporary tables (Oracle)

- Since version 18c, Oracle has two types of temporary tables:
  - Global Temporary Tables (GTT), whose metadata is permanent

```
CREATE TEMPORARY TABLE my_temp_table (
  id NUMBER,
  description  VARCHAR2(20));
```

  - Private Temporary Tables (PTT), which are memory-based temporary tables that are dropped at the end of the session or transaction depending on the setup

```
CREATE PRIVATE TEMPORARY TABLE ora$ptt_my_temp_table (
  id NUMBER,
  description  VARCHAR2(20))
ON COMMIT DROP DEFINITION;
```

Views (Virtual Tables)

- Concept of a view
  - Single table derived from other tables called the defining tables
  - Considered to be a virtual table that is not necessarily populated
  - The difference from a derived table is that a derived table does not maintain association to base tables while a view does
- Views are useful to prepare (intermediate) results, readability, security, and performance when they are materialized

Views in SQL

- CREATE VIEW command
  - Give table name, list of attribute names, and a query to specify the contents of the view
  - DROP VIEW disposes a view
- Once a View is defined, SQL queries can use the View similarly to a table

Example of View Definition

```
CREATE VIEW dept_info(dept_num, dept_name,
num_emp, total_salary) AS
SELECT d.dnumber, d.dname, COUNT(*), SUM(e.salary)
  FROM department d, employee e
  WHERE d.dnumber = e.dno
  GROUP BY dnumber, dname;

SELECT e.fname, e.lname, di.*
FROM employee e JOIN dept_info di
  ON e.dno = di.dept_num;
```

Types of Views

- Two types of views
  - Computed (CREATE VIEW my_view ...)
    - The DBMS only stores the view definition, and executes the view query every time the view is used
    - Requires zero maintenance, but does not increase the performance
  - Materialized (CREATE MATERIALIZED VIEW my_view ...)
    - The DBMS stores the view definition, executes the query and store the result as system-controlled table
    - Increases query performance, but requires the system to update the view to reflect updates in base tables

Maintenance of Materialized Views

- View recomputation
  - Drops the previous materialized view, and recompute from scratch
  - Easy to implement but inefficient
- Incremental update
  - Keeps track of updates in base tables and applies only the differences in the materialized view
  - Complex to implement but efficient
  - Many DBMSs rely on a view log to perform incremental maintenance

Maintenance of Materialized Views (2)

- Multiple ways to handle materialization
  - Immediate update strategy: updates a view as soon as the base tables are changed
    - Interleave view update operations with transactions
    - The materialized view is always ready when accessed
  - Lazy update strategy: updates the view when needed by a view query
    - May delay the access to the view, but can be useful in a heavy-loaded transaction system
  - Periodic update strategy: updates the view periodically
    - In this strategy, a view query may get a result that is not up-to-date
    - Periodic update is used to save computing resources when it is not critical to always get up-to-date results

Settings for materialized views (Oracle)

- BUILD IMMEDIATE | DEFERRED
  - IMMEDIATE: populates the view immediately
  - DEFERRED: populates the view by the next refresh
- REFRESH COMPLETE | FAST | FORCE
  - COMPLETE: refreshes by recalculating the materialized view's defining query
  - FAST: applies incremental changes to refresh the materialized view using the information logged in the materialized view logs
  - FORCE: performs a fast refresh if one is possible or a complete refresh if the fast refresh is not possible

Settings for materialized views (2) (Oracle)

- REFRESH ON COMMIT | ON DEMAND | START WITH [NEXT]
  - ON COMMIT: refreshes upon every commit
  - ON DEMAND: refreshes by the user manually launches a refresh through a built-in refresh stored procedures
  - START WITH <datetime> [NEXT <interval>]: specifies a datetime expression for the first automatic refresh time, and the interval between the next ones
- ENABLE QUERY REWRITE
  - Specifies that the materialized view is eligible to be used for query rewrite

Materialized view logs (Oracle)

- A materialized view log is a table associated with the master table of a materialized view
  - Required for incremental (FAST) refreshing
- WITH clause
  - PRIMARY KEY | ROWID | OBJECT ID: indicates that the primary key, row id, or object id of all rows changed should be recorded in the log
  - SEQUENCE: indicates that a sequence value providing additional ordering information should be recorded in the log
  - INCLUDING | EXCLUDING NEW VALUES: including new values is required for a materialized aggregate view be eligible for fast refresh, paying the overhead of recording new values

Example

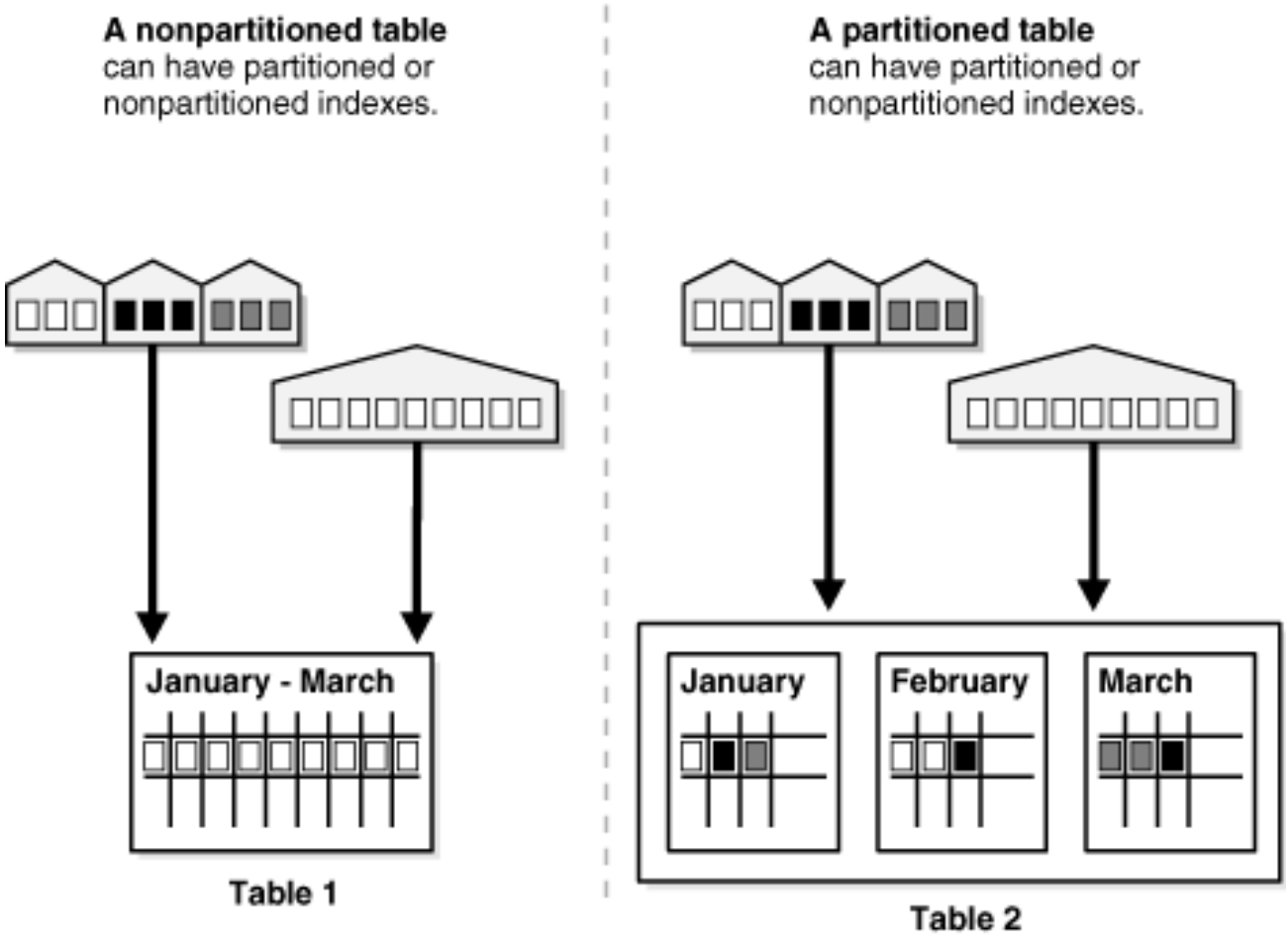
```
CREATE MATERIALIZED VIEW LOG ON sales
  WITH ROWID, SEQUENCE (amount_sold, time_id, prod_id) INCLUDING NEW VALUES;
CREATE MATERIALIZED VIEW LOG ON times
  WITH ROWID, SEQUENCE (time_id, calendar_year) INCLUDING NEW VALUES;
CREATE MATERIALIZED VIEW LOG ON products
  WITH ROWID, SEQUENCE (prod_id) INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW sales_mv
  BUILD IMMEDIATE
  REFRESH FAST ON COMMIT
  ENABLE QUERY REWRITE
  AS SELECT t.calendar_year, p.prod_id,
    SUM(s.amount_sold) AS sum_sales
    FROM times t, products p, sales s
    WHERE t.time_id = s.time_id AND p.prod_id = s.prod_id
    GROUP BY t.calendar_year, p.prod_id;
```

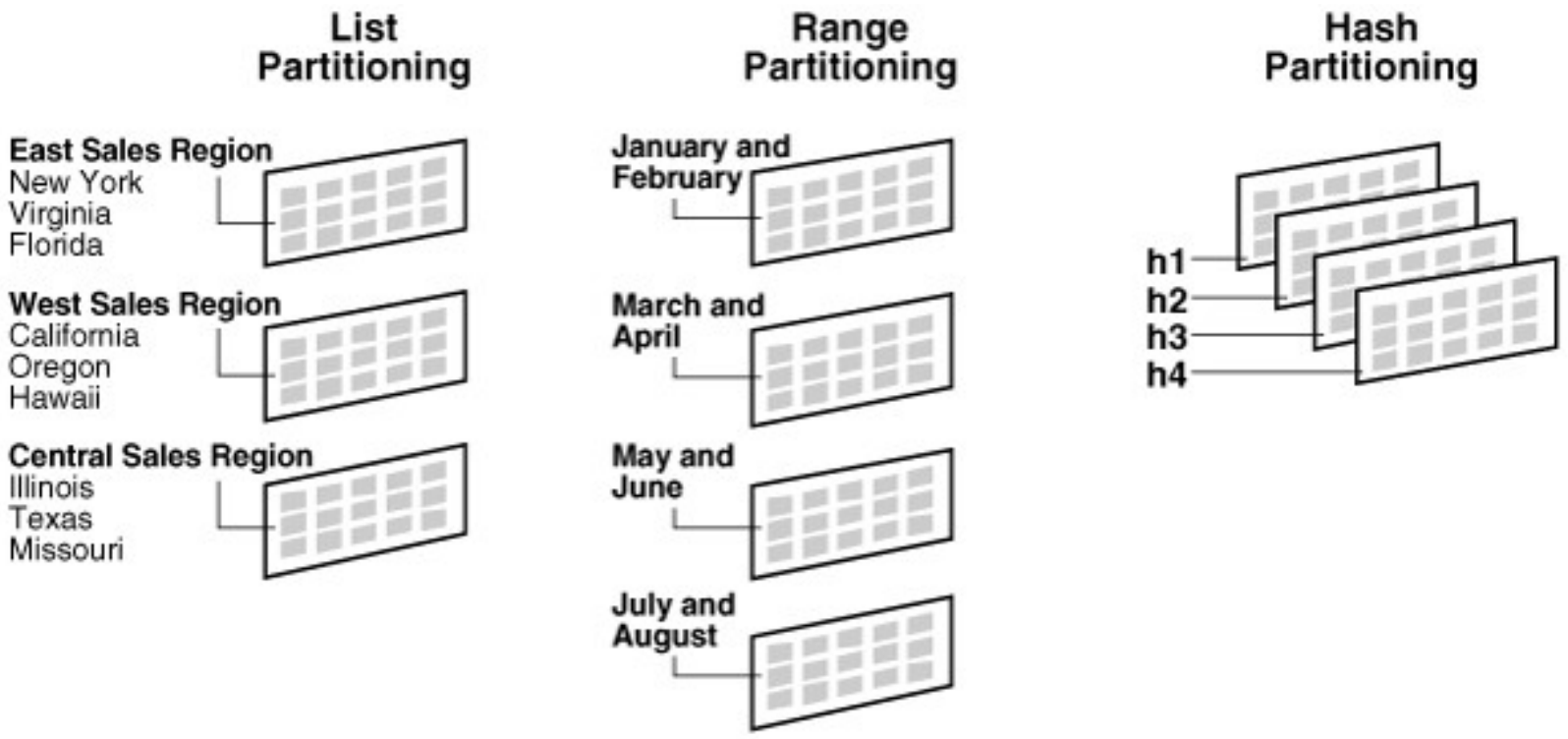
Partitioning

- Partitioning enhances the performance, manageability, and availability of a wide variety of applications
- Partitioning allows tables, indexes, and index-organized tables to be subdivided into smaller pieces
  - Enabling these database objects to be managed and accessed at a finer level of granularity

Partitioned tables and indexes



Data partitioning methods



Example

```
CREATE TABLE departments_hash (
  department_id NUMBER(4) NOT NULL,
  department_name VARCHAR2(30)
) PARTITION BY HASH(department_id)
  PARTITIONS 16;
```