# Tema 2

## Objetos del esquema

# Introducción.

- Un esquema es:
  - Una colección de estructuras lógicas de datos, u objetos del esquema.
  - Es propiedad de un usuario de base de datos,
  - Tiene el mismo nombre que el usuario y
  - Cada usuario posee su único esquema.

# Algunos objetos del esquema.

| Objeto | Definición |
|---|---|
| Clusters | Un grupo de registros de una o más tablas almacenadas físicamente en un fichero mixto. |
| Enlaces a bases de datos | Son objetos nombrados en Oracle que establecen caminos desde una base de datos hacia otra. Se usan en bases de datos disribuidas. |
| Disparadores (triggers) | Procedimientos que se ejecutan cuando ocurre algún evento (INSERT, UPDATE o DELETE). |
| Indices | Estructura opcional asociada con una tabla o un *cluster*, creadas sobre una o varias columnas, para acelerar la ejecución de sentencias SQL. |
| Vista | Tablas virtuales que se pueden definir sobre tablas de base o sobre otras vistas. |
| Tablas | Unidades básicas de datos. |
| Secuencia | Un tipo de datos especial de Oracle para la generación de valores de los atributos. |
| Sinónimos | Referencia directas a los objetos. |
| Otros | Dimensión.<br>Librerías de procedimientos externos.<br>Clases de Java, recursos de Java, y fuente Java.<br>Operadores<br>Funciones de almacenamiento, procedimientos, y paquete. |

# Introducción.

- Otros tipos de objetos se almacenan en la base de datos y se pueden crear y manipular con SQL, pero no están contenidos en un esquema:
  - Contextos
  - Directorios
  - Perfiles
  - Funciones
  - Tablespaces
  - Usuarios
- Los objetos del esquema son estructuras lógicas de almacenamiento de datos.
- Estos no tienen una correspondencia uno a uno con los archivos físicos en el disco que almacenan su información.

# Introducción.

- Oracle almacena algunos objeto del esquema dentro de un tablespace.
- Los datos de cada objeto se almacenan físicamente en uno o más archivos de datos del tablespace.
- No hay ninguna relación entre los esquemas y el tablespace:
  - Un tablespace puede contener objetos de diferentes esquemas, y
  - Los objetos de un esquema puede estar contenidos en diferentes tablespace.

# Tablas.

- Son la unidad básica de almacenamiento de una BD Oracle.

- Los datos son almacenados en filas y columnas.
  - Se define una tabla con un nombre de tabla y un conjunto de columnas con sus respectivos nombres y un tipo de datos (por ejemplo VARCHAR2, DATE, NUMBER) y una anchura.
  - El ancho puede ser predeterminado por el tipo de datos, como Date.
  - Si las columnas son de tipo numérico, podemos definir su precisión y su escala en vez del ancho.

- Una fila es una colección de información de un único registro.

# Tablas.

- Se puede especificar reglas para cada una de las columnas de la tabla.
  - Esta reglas son llamadas restricción de integridad.

- También puede especificar a las columnas de la tabla que los datos sean encriptados antes de ser almacenados en el archivo de datos.
  - La encriptación evita que los usuarios con acceso puedan mirar dentro de los archivos de datos directamente con las herramientas del sistema operativo.

- Después de crear una tabla, se pueden insertar, actualizar o eliminar filas de datos mediante sentencias DML.

- Los datos de la tabla pueden ser buscados utilizando consultas SQL.

# Cómo se almacena una tabla.

- Cuando se crea una tabla,
  - Oracle automáticamente asigna un segmento de datos de un tablespace para contener los futuros datos de la tabla.

- Podemos controlar la generación de este espacio a partir de los parámetros correspondientes (PCTFREE, PCTUSED...).

# Encadenamiento

- Oracle almacena cada fila de una tabla de menos de 256 columnas, en una o más row pieces.

- Si la fila completa cabe en un único bloque de datos, Oracle la almacena como un único row piece.

- Si la fila no cabe en un bloque de datos, o al modificar una existente, ésta excede el tamaño disponible del bloque, Oracle almacenará la fila empleando múltiples row pieces.

# Encadenamiento

- Normalmente un bloque de datos contendrá un único row piece para cada fila.

- Cuando Oracle debe usar varios row pieces para almacenar una fila que no le cabe en un bloque, éstos se encadenan entre diferentes bloques.

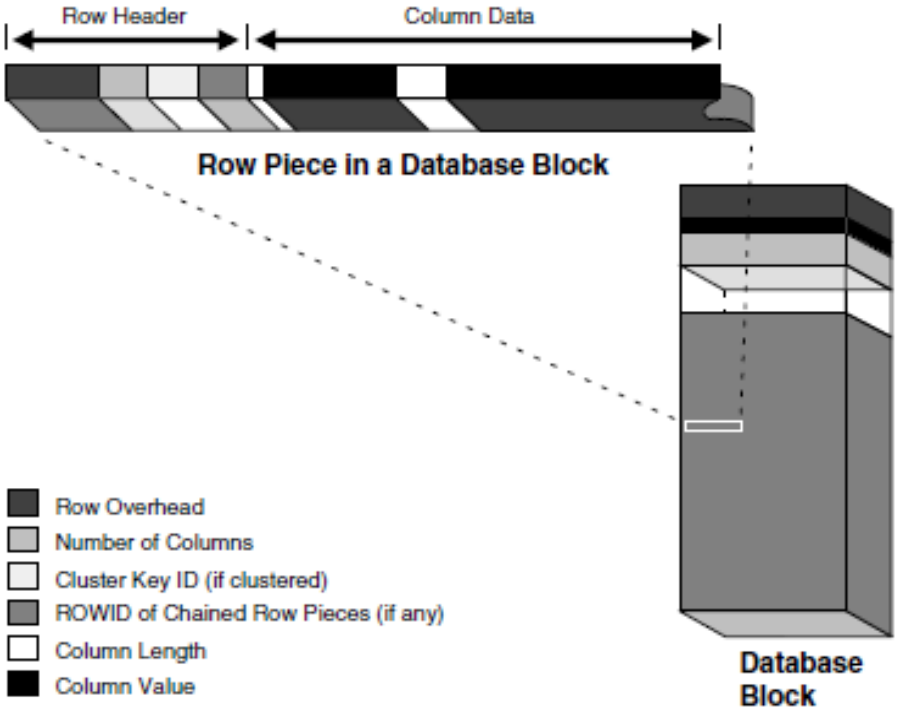
# Encadenamiento intra-bloque

- Cuando una tabla tiene más de 255 columnas, las filas con información más allá de la columna 255 se encadenan en el mismo bloque de datos.
  - Esto se llama encadenamiento intra-bloque.
  - Los row pieces se encadenan juntas usando los rowids, residiendo todos los datos en el mismo bloque.
  - No se produce una perdida en el rendimiento de las E/S ya que no hay operaciones adicionales para recuperar el resto de la fila.

- Cada row piece, encadenado o no, contiene tanto un encabezado de fila (row header) como los datos para todas o algunas de las columnas de la fila (column data).

# Formato de un row piece.

- El encabezado de la fila contiene información sobre:
  - Un row piece
  - El encadenamiento (*para row pieces encadenadas).
  - Columnas en el row piece.
  - Claves de clústeres (*para los datos agrupados)



# Formato de un row piece.

•Row header*:* En una fila de un único bloque ocupa 3 bytes como mínimo.

•Column data**:** Almacena la longitud de cada columna y sus datos.

- –1 bytes por columnas que almacenan 250 bytes o menos.
- –3 bytes por columna que almacenan mas de 250 bytes

•Si el tipo de datos que almacena una columna es variable, entonces el espacio requerido puede crecer o disminuir con actualizaciones de datos.

•El valor NULL se representa almacenando cero en la longitud y nada en los datos.

•Cada fila usa además 2 bytes en la cabecera de página del directorio de fila.

•Las filas agrupadas contienen:

- –La misma información que las filas no agrupadas.
- –Información que hace referencia a la clave del cluster al que pertenecen.

# ROWIDs y orden de columnas.

- ROWIDs (Identificadores de filas) de página:
  - Identifican cada página por su ubicación o dirección.
  - Después de que se les asigna, un row piece dado conserva su ROWID **hasta que la fila correspondiente se elimina o se exporta e importa usando los servicios de Oracle.**
  - Para las tablas de clúster, si los valores clave de un cluster cambia de row piece, ésta mantiene el mismo ROWID, pero también recibe un puntero adicional ROWID para los nuevos valores.
  - Eso ocurre porque los **ROWIDs son constantes durante la vida útil de un row piece**
    - Es útil hacer referencia a ROWIDs en sentencias SQL, como SELECT, UPDATE, y DELETE.

# ROWIDs y orden de columnas.

- El orden de columna:
  - El orden de las columnas es el mismo para todas las filas de una tabla dada.
  - Las columnas se suelen almacenar en el orden en el que se enumeran en la instrucción CREATE TABLE, pero esto no está garantizado.
  - Si una tabla tiene alguna columna de tipo de datos LONG, Oracle siempre las almacena en las últimas posiciones.
  - Si una tabla se altera añadiendo una nueva columna, la nueva columna se convierte en la última columna almacenada.
  - En general, se debe tratar de colocar las columnas que suelen contener valores nulos al final, así las filas ocupan menos espacio.
  - Sin embargo, si la tabla creada incluye una columna LONG, entonces se pierden los beneficios de la colocar columnas con valores nulos al final.

# Tabla comprimida.

•Para reducir el uso del disco y la memoria, se pueden almacenar las tablas y las tablas con particiones en un formato comprimido dentro de la base de datos.

•Esto conlleva, a menudo, un mayor rendimiento en operaciones de sólo lectura.

•La compresión de la tabla también puede acelerar la ejecución de la consulta.

•Sin embargo hay un ligero costo de sobrecarga de la CPU.

# Tabla comprimida.

- La característica de compresión de tablas en Oracle comprime los datos al eliminar los valores duplicados en un bloque de la base de datos.
- Los datos comprimidos almacenados en un bloque son autónomos.
  - Toda la información necesaria para volver a crear los datos sin comprimir en un bloque está disponible dentro de este.
- Los valores duplicados en todas las filas y columnas en un bloque se almacenan una vez al principio del mismo, en lo que se llama una tabla de símbolos.
- Todas las apariciones de estos valores se sustituyen por una breve referencia a la tabla de símbolos.
- Con la excepción de la tabla de símbolos al comienzo, los bloques comprimidos se parecen mucho a los bloques normales de la base de datos.
- Todas las características y funciones que trabajan en una base de datos normal también funcionan en bloques comprimidos.

# Tabla comprimida.

- Los objetos de base de datos que pueden ser comprimidos incluyen tablas y vistas materializadas.
- Para las tablas con particiones, se puede elegir comprimir algunas o todas las particiones.
- Los atributos de compresión pueden ser declarados para un tablespace, una tabla o una partición de ella.
- Si se declara a nivel del tablespace, a continuación, todas las tablas creadas en ese tablespace se comprimen por defecto.
- Se puede modificar el atributo de compresión de una tabla (o una partición o tablespace), y el cambio sólo se aplica a los nuevos datos que van a esa tabla.
- Como resultado, una sola tabla o partición puede contener algunos bloques comprimidos y algunos bloques normales.
- Eso garantiza que el tamaño de los datos no se incrementará como resultado de la compresión, en los casos en que la compresión podría aumentar el tamaño de un bloque, no se aplica.

# Uso de tablas comprimidas.

- La compresión se produce mientras se insertan o cargan los datos.

- Los datos existentes en la base de datos también se pueden comprimir empleando ALTER TABLE y MOVE.

- Se puede utilizar la utilidad de redefinición en línea de Oracle (DBMS_REDEFINITION PL/SQL).

- La compresión de datos para todos los tipos de datos funciona con excepción de todas las variantes de LOB y tipos de datos derivados de estos, como VARRAYs almacenados fuera de la línea o el tipo de datos XML almacenados en un CLOB.

# Uso de tablas comprimidas.

- Los costes generales asociados a la compresión son más apreciables en el momento de la carga de datos de forma masiva en la base de datos.

- Este es el principal cambio que hay que tener en cuenta al considerar la compresión.

- Las tablas comprimidas o particiones comprimidas pueden ser modificadas igual que otras tablas o particiones de Oracle.

# Uso de tablas comprimidas.

- La eliminación de los datos comprimidos es tan rápida como sin comprimir.
- La inserción de nuevos datos es también más rápida, porque los datos no se comprime en el caso de INSERT convencionales, sino que se comprime sólo al hacer la carga masiva.
- La actualización de datos comprimidos puede ser más lenta en algunos casos.
- Por estas razones, la compresión es más adecuado para aplicaciones de almacenamiento de datos que las aplicaciones OLTP.
- Los datos deben organizarse de tal manera que sean de sólo lectura o con poca frecuencia de cambio (por ejemplo, datos históricos), estos se mantiene comprimidos.

# Null indica la ausencia de valor.

- Un Null es la ausencia de un valor en una columna de una fila.
- Null indican datos faltantes, desconocidos o inaplicable.
- Estos podría no ser usado para implicar algún otro valor, como el cero.
- Una columna admite null a menos que tenga una restricción de integridad de NOTNULL o PRIMARY KEY definida para la columna y en tal caso ninguna fila puede ser insertada sin un valor para esa columna.
- Además se almacenan si estos caen entre columnas con valores de datos.
- En estos casos requieren que se almacenen 1 byte en la longitud de la columna (el cero).

# Null indica la ausencia de valor.

- Rastrearlos en fila no requieren almacenamiento porque una nueva cabecera de fila indica que las columnas siguen en las filas previas que son null.
  - Por ejemplo, si las últimas tres columnas de una tabla son null, no se almacena ninguna información para esas columnas.
- En tablas con muchas columnas, estas deberían estar definidas al final para conservar el espacio en disco como se ha mencionado anteriormente.

- La mayoría de las comparaciones entre null y otros valores son por definición ni verdadera ni falso, sino desconocido.
- Para identificar null en el SQL, hay que usar el predicado de IS NULL.
- Usar la función de SQL NVL para convertir valores nulos en no nulos.

- Los valores null no se almacenan en el índice, excepto cuando el valor de la columna clave del clúster es null ó es un índice de mapa de bits

# Valores predeterminados para las columnas.

- Se puede asignar un valor predeterminado a una columna de una tabla de modo que cuando se inserta una nueva fila y se omite un valor para la columna o se pone la palabra clave DEFAULT, se suministra un valor por defecto de forma automática.
- Los valores por defecto de columna funcionan como si una sentencia INSERT especificara el valor predeterminado.
- El tipo de datos por defecto, literal o expresión, debe coincidir o ser convertibles al tipo de datos de la columna.
- Si el valor predeterminado no está explícitamente definido para una columna, el valor por defecto para la columna implícitamente será NULL.

# Valores predeterminados, restricciones de integridad.

- La comprobación de las restricciones de integridad tienen lugar después de que una fila con un valor predeterminado se Inserta.

- Para ejemplo:
  - Se inserta una fila en la tabla *emp* que no incluye un valor para el número de departamento del empleado.
  - Como no hay ningún valor suministrado por el número de departamento, Oracle introduce el valor de la columna deptno por defecto a 20.
  - Después de insertar el valor predeterminado, Oracle comprueba la restricción FOREIGN KEY definida en deptno.

# Tablas particionadas.

- Permiten que sus datos sean desglosados en trozos más pequeños y manejables llamadas particiones, o incluso subparticiones.

- Los índices pueden dividirse de manera similar.

- Cada partición puede considerarse de forma individual, y puede operar independientemente de las otras particiones, lo que mejora la disponibilidad y el rendimiento.

# Tablas anidadas.

- Las tablas se pueden anidar dentro de otras tablas como valores de una columna.

- El servidor de base de datos Oracle almacena los datos de tablas anidadas fuera de las filas de la tabla principal, utilizando una tabla almacén que está asociada con la columna de tabla anidada.

- La fila principal contiene un conjunto de valores, identificador único, asociado a una instancia de la tabla anidada.

# Tablas temporales.

- Son tablas que duran mientras dura la transacción o sesión actuales.
- Se crean con la sentencia CREATE GLOBAL TEMPORARY TABLE.
- Debe especificarse si son datos específicos de la transacción o de la sesión.
- Son datos privados de cada sesión: No pueden compartirse.
- TRUNCATE en una sesión específica de la tabla temporal trunca los datos de su propia sesión.
- No trunca los datos de otras sesiones que están utilizando la misma tabla.

# Tablas temporales.

- DML no genera datos de redo para una tabla temporal, ya que al cerrar la sesión o al terminar de forma anómala, la tabla se eliminará automáticamente.

- Puede contener índices, los cuales son también temporales con CREATE INDEX.

- De igual forma pueden crearse vistas o disparadores sobre una tabla temporal.

- Las herramientas de exportación de Oracle permiten exportar la definición de una tabla temporal pero nunca sus datos.

# Tablas temporales. Asignación de segmentos.

- Las tablas temporales utilizan segmentos temporales.
- A diferencia de las tablas permanentes, a las tablas temporales y sus índices no se les asigna automáticamente un segmento cuando se crean.
- En cambio, los segmentos se asignan cuando ocurre el primer INSERT (o CREATE TABLE AS SELECT).
- Esto significa que si un SELECT, UPDATE, o DELETE se realiza antes de la primera inserción, la tabla aparece vacía.
- Pueden llevarse a cabo las instrucciones de DDL (ALTER TABLE, DROP TABLE, de CREATE INDEX, etc) en una tabla temporal sólo cuando no hay ninguna sesión actualmente ligada a ella por realizar un INSERT en ella.
- La sesión se libera por un TRUNCATE, o ROOLBACK, COMMIT, sobre una transacción especifica sobre la tabla o al terminar el periodo de sesión.

# Tablas externas.

- Las tablas externas permiten acceder a datos de fuentes externas como si se tratara de una tabla en la base de datos.
- Puede conectarse a y crear los metadatos de la tabla externa utilizando DDL.
- El DDL consta de dos partes:
  - Una que describe los tipos de columna, y
  - Otra que describe la asignación de los datos externos a las columnas de datos de Oracle (los parámetros de acceso).

# Tablas externas.

- Una tabla externa no describe todos los datos que se almacenan en la base de datos.
- Tampoco describe cómo se almacenan los datos en la fuente externa.
- En su lugar, se describe cómo la capa de la tabla externa tiene que presentar los datos al servidor.
- Es responsabilidad del controlador de acceso y de la capa de las tabla externa hacer las transformaciones necesarias requeridas en los datos en el archivo de datos para que coincida con la definición de la tabla externa.
- Son de sólo lectura, por lo tanto, no hay operaciones de DML, y no se puede crear índices en ellas.

# Carga de datos con tablas externas.

•El principal uso de una tabla externa es emplearla como origen de información para cargar datos en una tabla real.

•Después de crearla se puede utilizar un CREATE TABLE AS SELECT or INSERT INTO ... AS SELECT, utilizando la tabla externa como la fuente del SELECT.

•Al acceder a la tabla externa a través de una sentencia SQL, sus campos se usan como en una tabla normal.

•Se pueden manipular los datos de la fuente externa, utilizando los campos como argumentos para cualquier SQL incorporando, función de PL/SQL o función de Java.

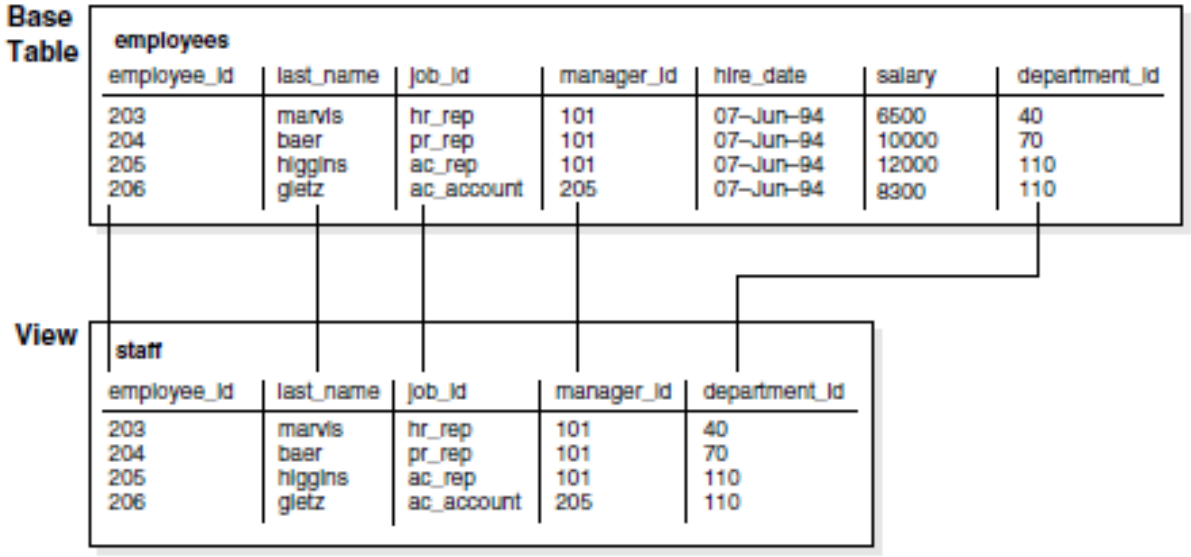
•Para data warehousing se pueden hacer transformaciones más complejas.

•También se puede utilizar este mecanismo de almacenamiento de datos para hacer limpieza en data warehousing .

•Mientras que las tablas externas no puede contener un objeto columna, se puede construir un objeto de columna utilizando atributos de la tabla externa.

# Vistas.

- Una vista es una representación particular de los datos contenidos en una o más tablas (tablas bases) u otras vistas.
- Toma el salida de una consulta y la trata como una tabla.
- Podemos pensar que una vista es una consulta guardada o una tabla virtual.
- Pueden usarse en la mayoría de las situaciones en las que usaríamos una tabla.



# Cómo se almacena una vista.

- A diferencia de las tablas, las vistas no ocupan ningún espacio de almacenamiento.

- Oracle almacena la definición de las vistas en el diccionario de datos de donde la toma cuando se ejecuta una instrucción que referencia a la vista.

# Mecánica de una vista.

- Al hacer referencia a una vista en una sentencia SQL, Oracle:

  1. Combina la declaración que hace referencia a la vista con la consulta que define la vista.
  2. Analiza la instrucción fusionada en un área compartida de SQL
  3. Ejecuta la sentencia

- Oracle analiza sintácticamente la declaración de una vista en una área compartida de SQL, siempre y cuando no exista en ese área una declaración similar, con lo cual obtiene una reducción en memoria.

# Uso de índices en comparación a las vistas.

- Oracle determina si se utiliza los índices de una consulta frente a una vista mediante la transformación de la consulta original, cuando la fusiona con la definición de la vista.

Considerando la siguiente vista:

```
CREATE VIEW employees_view AS
  SELECT employee_id, last_name, salary, location_id
  FROM employees JOIN departments USING (department_id)
  WHERE departments.department_id = 10;
```

El usuario realiza una consulta en ella:

```
SELECT last_name
  FROM employees_view
  WHERE employee_id = 9876;
```

La consulta final construida por oracle.

```
SELECT last_name
  FROM employees, departments
  WHERE employees.department_id =
        departments.department_id AND
        departments.department_id = 10 AND
        employees.employee_id = 9876;
```

- Combinando la consulta y las vista, Oracle puede acceder a los índice de las tabla base que hace referencia en la vista.
- En el caso de que no pueda combinarlas, no puede usar esos índices.

# Vistas Materializadas.

•Una vista materializada es una aproximación diferente:

1. El resultado de la consulta se almacena en una tabla caché real, que será actualizada de forma periódica a partir de las tablas originales.
2. Esto proporciona:
   + Acceso mucho más eficiente.
   - Incremento en el tamaño de la base de datos.
   - Posible falta de sincronía.
3. Muy utilizadas en entornos de data warehousing, donde el acceso frecuente a las tablas básicas resulta demasiado costoso.

# Restricción de clave primaria.

- Asegura que los valores sean únicos para cada registro y constituyen el identificador único de la fila.

- Es un campo o la combinación de campos que definen de manera única un registro.

- Ninguno de los campos que forman parte de la clave principal puede contener un valor nulo.

- Una tabla sólo puede tener una clave principal.

# Restricción de unicidad e integridad referencial.

- Una restricción de unicidad es un campo o combinación de campos que definen de manera única un registro.

- Algunos de los campos pueden contener valores nulos, siempre y cuando la combinación de valores sea única.

- Integridad referencial.

# Refresco de las vistas materializadas.

- Para mantener actualizadas las vistas, Oracle las refresca después de cualquier cambio en sus tablas base.
- El método de refresco puede ser incremental ó completo
- Las vistas materializadas que emplean el refresco incremental disponen de un controlador de la vista materializada, o controlador de carga directa, que guarda memoria de los cambios efectuados en las tablas base.
- El refresco de las vistas puede hacerse también bajo petición o a intervalos de tiempo regulares.
- Las vistas materializadas que se encuentran en la misma base de datos que sus tablas base puede refrescarse cuando se confirma una transacción que ha modificado alguna de las tablas base.

# Generador de secuencia.

- El generador de secuencia proporciona una serie secuencial de números.
- El generador de secuencia es especialmente útil en entornos multiusuario para generar un único número secuencial sin el añadido de la sobrecarga de E/S o el bloqueo de la transacción.
- El generador de secuencia reduce la serialización en la ejecución de dos transacciones que deben generar números secuenciales al mismo tiempo.
- Al evitar el serialización que se produce cuando varios usuarios esperan entre sí para generar y utilizar un número de secuencia, el generador de secuencia mejora el rendimiento de las transacciones, y la espera del usuario es mucho más corta.

# Crear una secuencia.

- Con el siguiente código se crean las secuencias:

```
CREATE SEQUENCE nombre_secuencia
INCREMENT BY numero_incremento
START WITH numero_por_el_que_empezara
MAXVALUE valor_maximo | NOMAXVALUE
MINVALUE valor_minimo | NOMINVALUE
CYCLE | NOCYCLE
ORDER | NOORDER
```

- CYCLE | NOCYCLE:

  La funcionalidad de esta directiva es la de crear una secuencia cíclica, cuando su valor halla pasado el MAXVALUE vuelve al valor inicial.

# Sinónimos.

- Un sinónimo es un alias para cualquier tabla, vista, vista materializada, secuencia, procedimiento, función, paquete, etc. definidas por el usuario.
- Debido a que un sinónimo es simplemente un alias, no requiere de almacenamiento que no sean los de definición en el diccionario de datos.
- Los sinónimos son de uso frecuente para la seguridad y comodidad.
- Por ejemplo:
  - Enmascarar el nombre y propietario de un objeto.
  - Ofrecer transparencia de ubicación de los objetos a distancia de una base de datos distribuida.
  - Simplificar las instrucciones SQL para los usuarios de la base de datos.

# Índices.

- Estructura opcional asociada con una tabla o un *cluster*.

- Se crean sobre una o varias columnas, para acelerar la ejecución de sentencias SQL.

- Una tabla puede tener cualquier cantidad de índices, siempre que la combinación de columnas en cada uno sea diferente.
  - Incluso, cambiando el orden.

  - Ejemplo:

```
CREATE INDEX Pieza_idx1 ON Pieza (Nombre, Cantidad);
CREATE INDEX Pieza_idx2 ON Pieza (Cantidad, Nombre);
```

# Índices.

- Son lógica y físicamente Independientes de los datos de las tablas asociadas y se mantienen dinámica y automáticamente.

- Se puede crear o borrar un índice sin ningún efecto sobre los datos de la tabla u otros índices.

- Operaciones cuando existen índices:
  - Consultar:
    - El tiempo de acceso es casi constante, aunque se inserten más filas en la tabla.
  - Borrar, Insertar o Actualizar:
    - El tiempo aumenta si existen muchos índices sobre la tabla (Oracle debe también actualizar los índices).

# Índices.

- Los indices pueden ser únicos (*unique*) o no únicos (*nonunique*), según exijan o no que las columnas del índice, si admiten o no valores duplicados en distintas filas.

CREATE [UNIQUE] INDEX…

- Es recomendable que sean únicos y que, al menos, se cree uno por cada clave primaria o externa de cada tabla, así como por cada columna que contenga valores de búsqueda usuales, sin excedernos.

# Índices basados en funciones.

- Se puede construir un índice sobre una función definida por el usuario.

- Ejemplo: Para facilitar búsquedas insensibles a mayúsculas/minúsculas:

```
CREATE INDEX uppercase_idx ON Pieza (UPPER(Nombre));

SELECT * FROM Pieza WHERE UPPER(Nombre) ='TORNILLO‘;
```

# Cómo se almacena un índice.

•Cuando se crea un índice, Oracle le asigna un segmento de índice para contener sus valores en el *tablespace* correspondiente.

•Es preferible que este *tablespace* no sea el mismo en el que está contenida la tabla asociada y que ambos *tablespaces* estén almacenados en discos diferentes, para que Oracle pueda leerlos en paralelo (si el hardware lo permite).

•Al crear un índice, Oracle ordena las columnas del índice y almacena el valor de los índices junto con el ROWID de las filas.

•Los índices pueden crearse en orden ascendente (ASC), descendente (DESC), comprimidos (COMPRESS) o no comprimidos (NOCOMPRESS).

# Cómo se almacena un índice.

- Oracle ofrece varios esquemas de indexación:
  - Árbol B.
    - Clave inversa.
  - Bitmap.
    - Unión de bitmap.

# Árbol B.

- Esta estructura desempeña una excelente funcionalidad para una amplia variedad de consultas, desde búsqueda exacta a búsquedas por rango de valores.

- Son eficientes las inserciones, actualizaciones, y borrados, manteniéndose el orden de clave.

- Ventajas de la estructura de árbol-B:
  - Todos los nodos hoja están a la misma profundidad.
  - La recuperación de cualquier registro lleva aproximadamente el mismo tiempo.
  - Permanecen automáticamente balanceados.
  - Todos los bloques del árbol están llenos de media en ¾.

# Índice de clave inversa.

- La estructura subyacente sigue siendo el árbol B,

- Puede ocurrir que las inserciones o modificaciones a un índice se concentren en un conjunto pequeño de bloques, esto puede disminuir considerablemente el rendimiento.

  - Se puede generar un índice inverso, que consiste en que las claves (valores de las columnas) se insertan invirtiendo el orden de los bytes.

    Por ejemplo: ABEL se almacenaría como LEBA.

# Índice de clave inversa.

- Esta posibilidad es recomendable tan solo para operaciones de acceso a un único valor

- Las recuperaciones por rango de índice no se beneficiarán del índice inverso al no estar ahora las entradas ordenadas por el valor de la columna.

Ejemplo: CREATE INDEX  ON Nombre_tabla(a, b, c) REVERSE

# Índice de bitmap.

- En un índice de mapa de bits, para cada valor de clave se utiliza un mapa de bits en vez de el ROWID.

- Cada bit del mapa de bits corresponde a un ROWID, y vale 1 si la fila pertenece a ese valor clave, y 0 si no pertenece.

- Las ventajas de utilizar índices de mapa de bits son mayores para las columnas en las que el número de valores distintos es pequeño en comparación con el número de filas de la tabla.

# Índice de bitmap.

- Si el número de valores distintos de una columna es menos del 1% del número de filas en la tabla, o si los valores de una columna se repiten más de 100 veces, la columna es una candidato para un índice de mapa de bits.

| CUSTOMER # | MARITAL_ STATUS | REGION | GENDER | INCOME_ LEVEL |
|---|---|---|---|---|
| 101 | single | east | male | bracket_1 |
| 102 | married | central | female | bracket_4 |
| 103 | married | west | female | bracket_2 |
| 104 | divorced | west | male | bracket_4 |
| 105 | single | central | female | bracket_2 |
| 106 | married | central | female | bracket_3 |

| REGION='east' | REGION='central' | REGION='west' |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |

# Índice de bitmap.

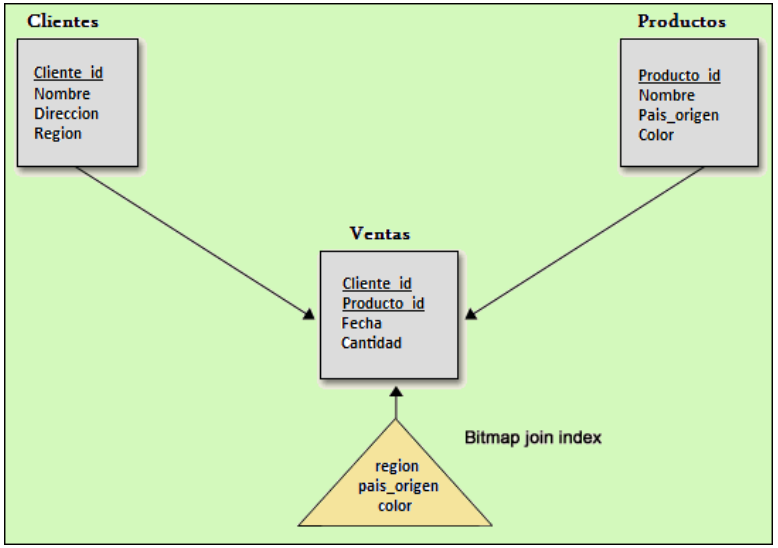
SELECT COUNT(*) FROM CUSTOMER
WHERE MARITAL_STATUS = ’married’ AND REGION IN (’central’,’west’);

| status = 'married' | | region = 'central' | | region = 'west' | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | | 0 | | 0 | | 0 | | 0 |
| 1 | | 1 | | 0 | | 1 | | 1 | | 1 |
| 1 | AND | 0 | OR | 1 | = | 1 | AND | 1 | = | 1 |
| 0 | | 0 | | 1 | | 0 | | 1 | | 0 |
| 0 | | 1 | | 0 | | 0 | | 1 | | 0 |
| 1 | | 1 | | 0 | | 1 | | 1 | | 1 |

# Índice de unión de bitmaps.

- Es un índice de mapa de bits que se describe a través de una consulta de unión.

- Son estructuras poderosas para acelerar el proceso de consulta sobre tablas que se consultan juntas (join) y cuyos atributos son de baja cardinalidad.



# Índice de unión de bitmaps.

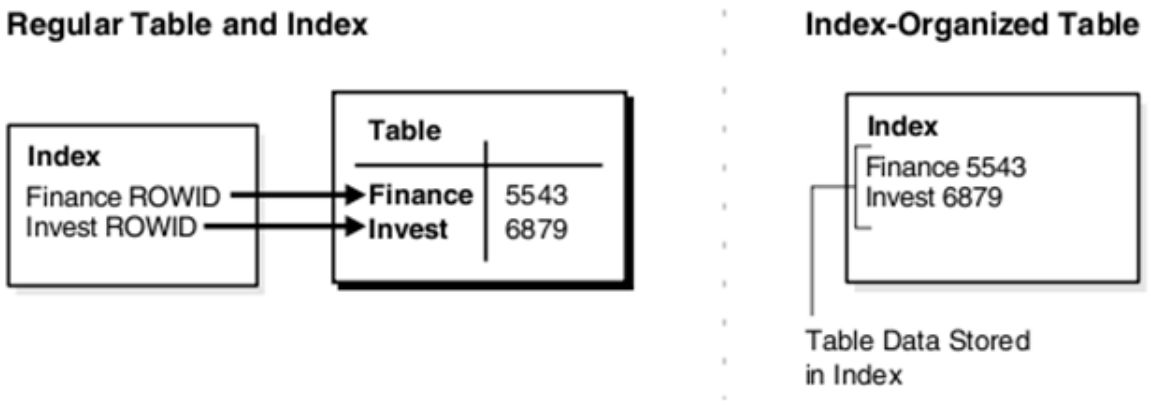
- Ejemplo:

```
CREATE BITMAP INDEX REGION_PAIS_COLOR
ON VENTAS( C.REGION, P.PAIS_ORIGEN, P.COLOR)
FROM VENTAS V,
CLIENTES C,
PRODUCTOS P
WHERE V.CLIENTE_ID=C.CLIENTE_ID
AND V.PRODUCTO_ID=P.PRODUCTO_ID;
```

# Tablas organizadas en índice (IOT).

•Una tabla organizada en índice tiene una organización de almacenamiento que es una variante de un árbol-B.

•A diferencia de una tabla común, cuyos datos se almacenan como una colección desordenada (heap), **los datos de una tabla organizada en índice se almacena en una estructura de índice árbol B**.




•En fin**, una tabla organizada en índice, es básicamente una tabla almacenada en un índice**.

# Ventajas de las IOT.

| VENTAJAS |
| --- |
| Colocación física de los datos |
| Aumento en la eficacia del buffer cache. |
| Reducidas operaciones de E/S lógicas. |
| Disminución de la necesidad de índice. |
| Los datos son almacenados de forma ordenada por la clave primaria, lo cual puede ser beneficioso. |
| Los índices de las tablas IOT pueden ser reconstruidos (reorganizados) en línea en el raro caso de que sea necesario. |
| Las tablas IOT pueden ser particionadas por rango o por hash . |
| Soportan características como constraints(restricciones), triggers, LOB y columnas de objetos, particionamiento, operaciones paralelas, reorganización en línea, desbordamiento en el área de almacenamiento ...etc |

# Desventajas de las IOT.

- Como pasa con las tablas agrupadas, las tablas IOT son más lentas en las operaciones de inserción que las tablas convencionales debido a que los datos tienen una localización a donde ir.

- Puede ser necesario considerar el segmento de overflow.

- Hay cuestiones clave con tablas muy amplias en una estructura IOT.

- Las IOT son más adecuadas para tablas "altas" y "flacas", pero con un mínimo de previsión también pueden trabajar con tablas "anchas".

# Crear una IOT.

```
CREATE TABLE <table_name>
( <field_definitions>,
CONSTRAINT <index_name> PRIMARY KEY
(<key_columns>)
)
ORGANIZATION INDEX ...;
```

Cualificador que indica que es una tabla organizada por índices.

---

```
CREATE TABLE admin_docindex(
        token char(20),
        doc_id NUMBER,
        token_frequency NUMBER,
        token_offsets VARCHAR2(2000),
        CONSTRAINT pk_admin_docindex
PRIMARY KEY (token, doc_id))
    ORGANIZATION INDEX
    TABLESPACE admin_tbs
    PCTTHRESHOLD 20
    OVERFLOW TABLESPACE admin_tbs2;
```

Nos indica el porcentaje del bloque reservado del índice. Cualquier área que exceda este porcentaje se almacenara en la **OVERFLOW TABLESPACE** que nosotros definamos

Cualificador usado para definir el área de desbordamiento

# Limitaciones de las IOT.

- El numero máximo de columnas es 1000.

- El número máximo de columnas en la parte de índice de una fila es de 255, incluyendo la clave y columnas sin clave. Si hay más de 255 columnas en el índice, se debe utilizar un segmento de desbordamiento.

- El numero máximo de columnas que puedes incluir en la clave primaria es 32.

- PCTTHRESHOLD debe estar en el rango de 1-50. El valor predeterminado es 50.

- Todas las columnas de clave debe encajar en el umbral especificado

- Si el tamaño máximo de una fila supera el 50% del tamaño de bloque de índice y no se especifica un segmento de desbordamiento, la sentencia CREATE TABLE falla.

- El índice no puede contener columnas virtuales.

# Índices secundarios en las IOT.

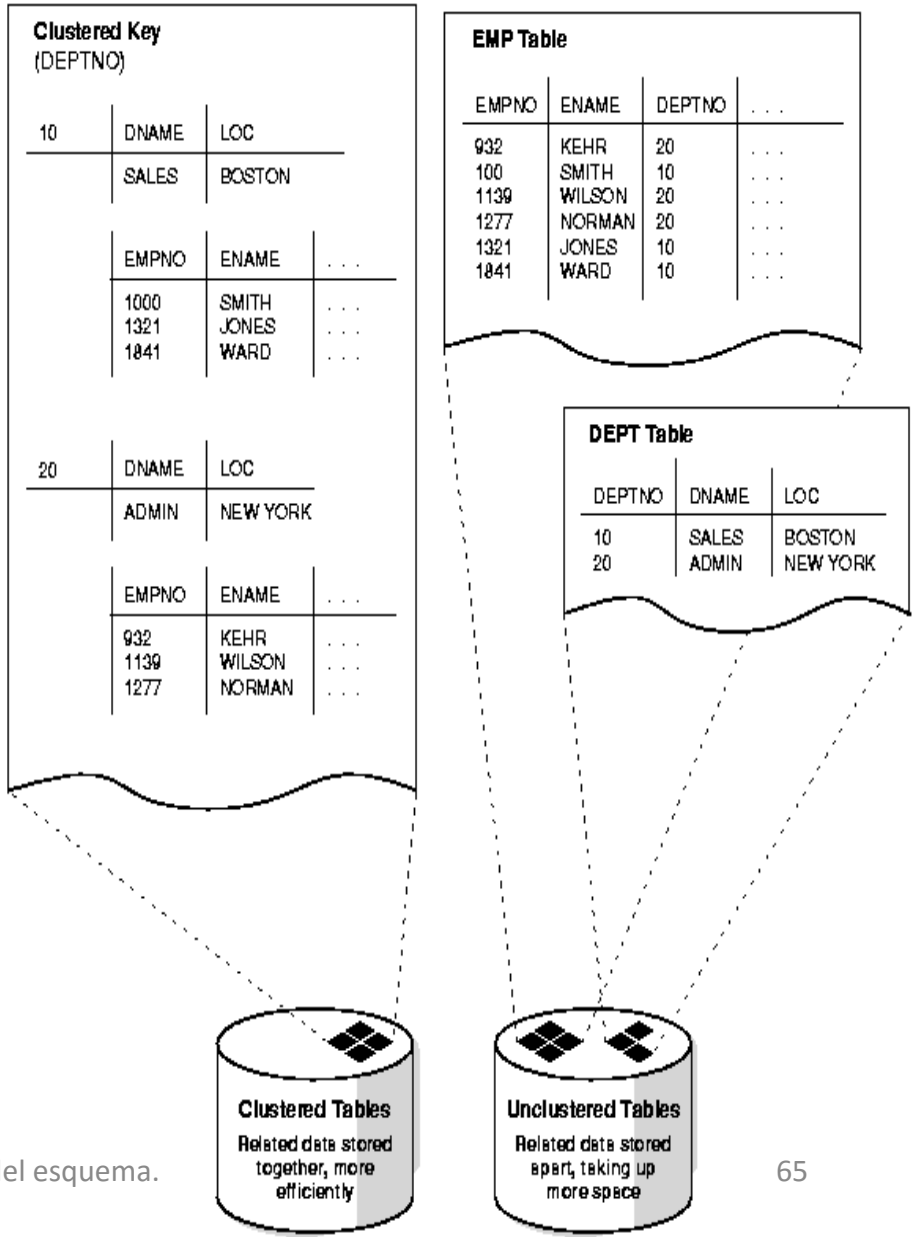
• Se pueden crear índices secundarios en una IOT para proporcionar rutas de acceso múltiple.

•Difieren de los índices secundarios de las tablas ordinarias de dos maneras:

1. Guardan los identificadores lógicos de registro en lugar de los identificadores de fila física.
   - Si la ubicación física de una fila cambia, su identificador de fila lógica sigue siendo válida.
   - Un efecto de esto es que una operación de mantenimiento de la tabla, como ALTER TABLE ... MOVE, no convierte el índice secundario en inservible.
2. El identificador de fila lógica también incluye una conjetura física que identifica la dirección del bloque de base de datos en la que la fila es probable que se encuentre.
   - Si la conjetura física es correcto, el recorrido del índice secundario incurriría en una sola E/S adicional una vez que se encuentra la clave secundaria.

# CLUSTERS.

- Un cluster es un grupo de tablas almacenadas como una sola tabla que comparten una columna en común.
- Solamente afecta a la organización física de los datos y no a la estructura lógica.
- Si a menudo se necesita recuperar datos de dos o más tablas basado en un valor de la columna que tienen en común, entonces es más eficiente organizarlas como un cluster, ya que la información podrá ser recuperada en una menor cantidad de operaciones de lectura realizadas sobre el disco.
- Las columnas que relacionan las tablas de un *cluster* se llaman clave del *cluster*.



# CLUSTER HASH.

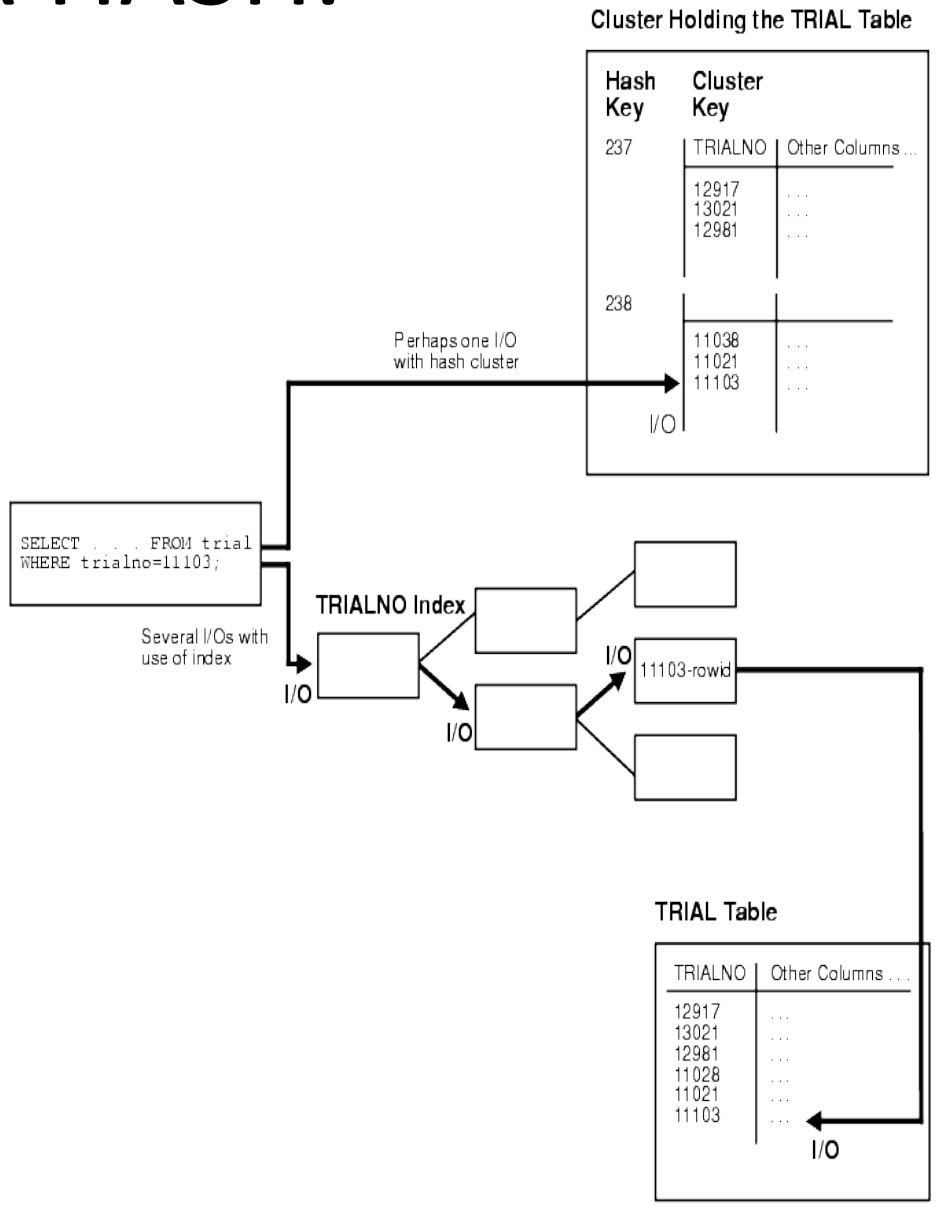
- Los clusters hash agrupan los datos de las tablas de una manera similar a los clusters regulares (la clave es un índice).
  - Se usa una función hash que se aplica a la columna o columnas de la clave del cluster.
  - La función hash genera una distribución de valores numéricos, llamados valores hash, basados en claves del cluster.

- Al igual que en los clusters indexados, la clave de un cluster hash puede ser una columna o varias columnas clave.

- El resultado de la función corresponde a un bloque de datos en el cluster, el cual Oracle lee o escribe según la declaración emitida.

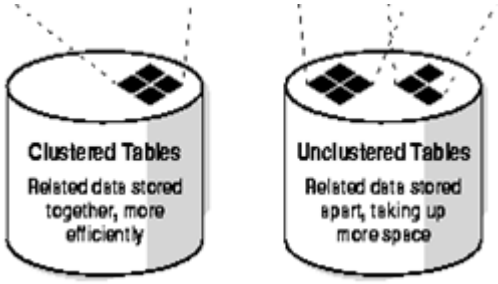
# CLUSTER HASH.

- Son una mejor opción que una tabla indexada o un cluster indexado cuando la tabla se consulta con frecuencia con consultas de igualdad (todas las filas de DPTO 10).
- Ahorra tener que leer el índice para localizar o insertar un dato, ya que la función hash NO requiere lecturas de disco.



# IOT VS CLUSTER.

En su naturaleza es semejante a un cluster en el que los datos son almacenados físicamente según un valor de clave, pero difieren en las maneras siguientes:




| IOT | Cluster |
|---|---|
| Una sola estructura de datos | Tiene segmentos de índice y datos |
| Datos almacenados por la clave | Idem, pero las claves no son almacenadas por orden. |
| No es necesario saber el nº máximo de claves | Es necesario |

UTILIDAD DE LAS IOT

- Tablas de asociación, tablas que se usan en relaciones muchos-a-muchos.
- Tablas donde la colocación física de los datos es importante pero el orden de inserción es impredecible.
- Cuando los datos llegan en un orden que hace imposible para un cluster con el tiempo para mantener datos colocados.