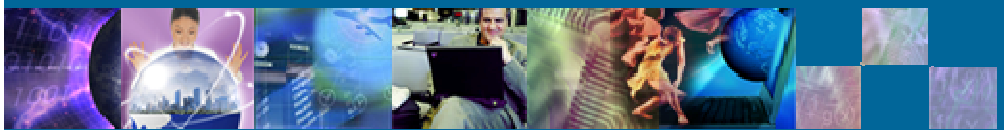


IBM

DB2 V9

Particionamiento con DB2 9

ON DEMAND BUSINESS™

Tipos de Particionamiento

1. PARTICIONAMIENTO DE BASE DE DATOS
   - Previo a V9.
   - Particionamiento mediante hashing.
   - Data Partitioning Feature (DPF).
     - Antiguo DB2 EEE.

2. MULTI DIMENSIONAL CLUSTERING (MDC)
   - Previo a V9.
   - Block Indexes.

3. Particionamiento de Tabla
   - Particionamiento por Rango.
   - Nuevo en V9.

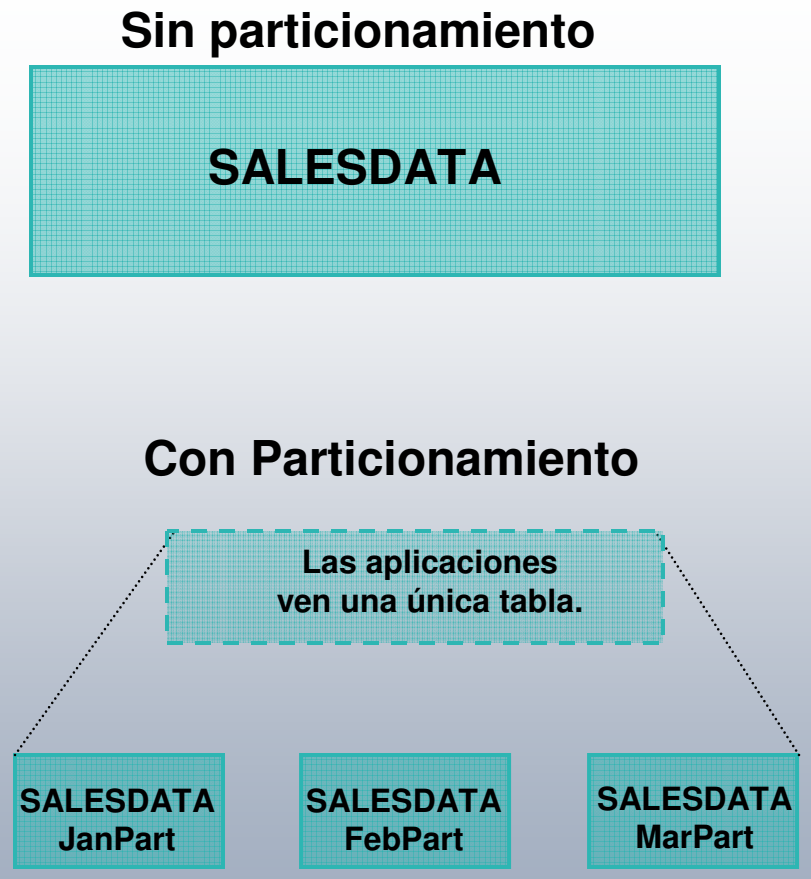
"Se pueden usar los 3 tipos de particionamiento simultáneamente"

Particionamiento de Tabla: Definición

- Particionamiento por Rango
- Permite dividir las filas de una tabla entre distintas particiones, almacenadas físicamente de forma independiente, según el valor de una o más columnas.

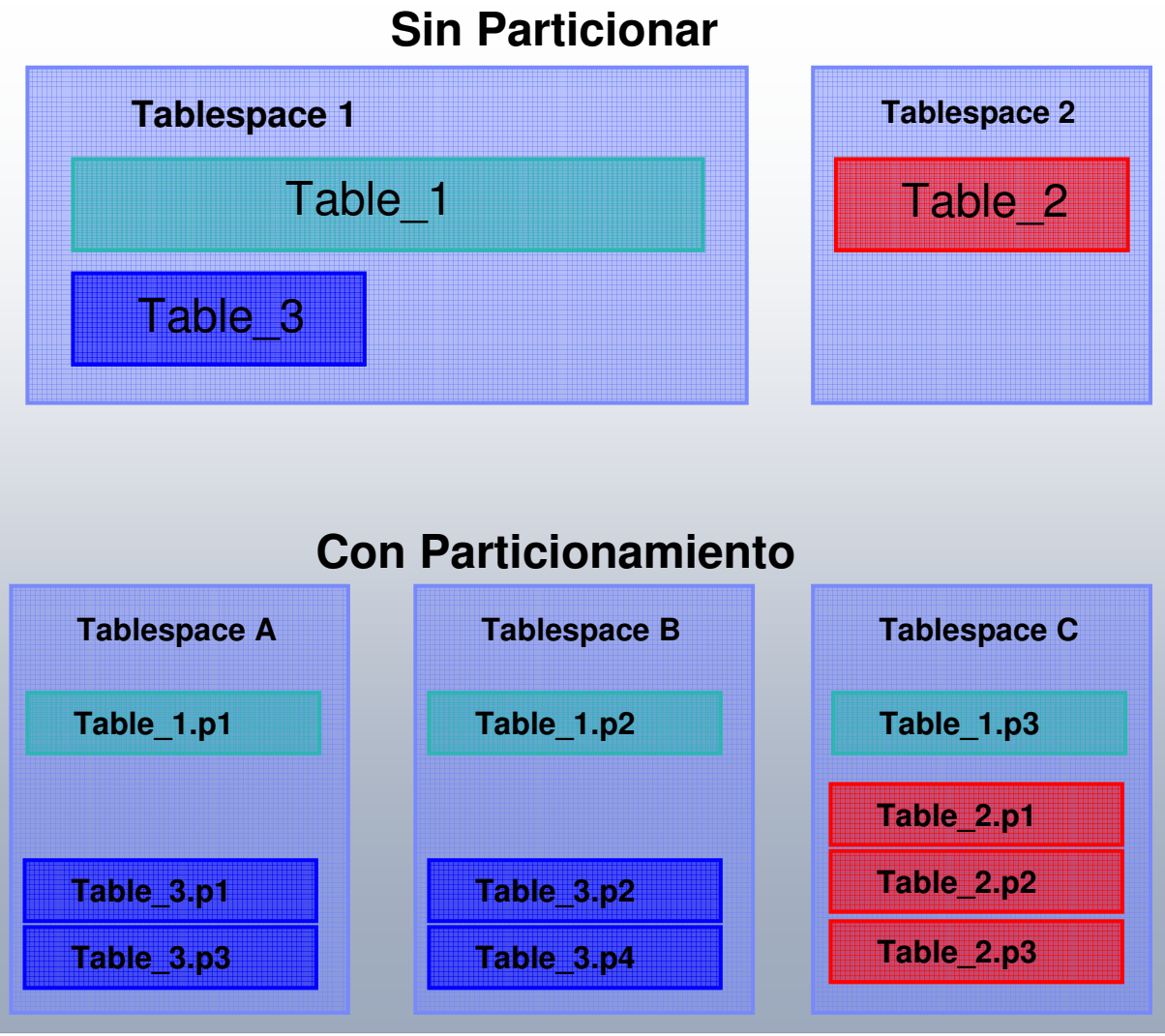
Beneficios:

- Rendimiento de SQL
  - "partitioning elimination"
- Roll-in/roll-out optimizado
- Integración con HSM.



Particionamiento de Tabla: Tablespaces

- Una tabla se puede dividir en múltipes Tablespaces
- Indices de una tabla se pueden poner en distintos Tablespaces

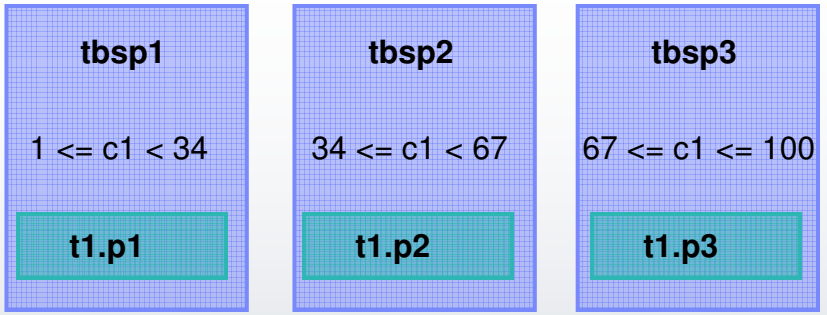


Particionamiento de Tabla: Conceptos

- Sintaxis. Larga y otra Corta
- Columnas de Particionamiento
  - Deben ser tipos de datos básicos (no LOBs o Long Varchars, etc..)
  - Pueden ser múltiples columnas.
  - Pueden ser columnas generadas.
- Valores Especiales
  - `MINVALUE`, `MAXVALUE` pueden usarse para especificar el principio y el fin de los rangos.

```
CREATE TABLE t1 …
     (STARTING(MINVALUE)
      ENDING(MAXVALUE) …
```

- **SQL0327N**: La fila no puede ser insertadas porque cae fuera de los rangos.



Forma Corta

```
CREATE TABLE t1(c1 INT)
  IN tbsp1, tbsp2, tbsp3
  PARTITION BY RANGE(c1)
  (STARTING FROM (1) ENDING (100) EVERY (33))
```

Forma Larga

```
CREATE TABLE t1(c1 INT)
  PARTITION BY RANGE(a)
  (STARTING FROM (1) ENDING(34)  IN tbsp1,
                     ENDING(67)  IN tbsp2,
                     ENDING(100) IN tbsp3)
```

Particionamiento de Tabla: Definición de Rangos

- Sintaxis Larga
- Uso de `STARTING … ENDING …` para especificar rangos

```
CREATE TABLE sales (sale_date DATE, customer INT, …)

  PARTITION BY RANGE(sale_date)
    (
    STARTING '1/1/2000'  ENDING '3/31/2000',
    STARTING '4/1/2000'  ENDING '6/30/2000',
    STARTING '7/1/2000'  ENDING '9/30/2000',
    STARTING '10/1/2000' ENDING '12/31/2004'
    );
```

Particionamiento de Tabla: Minvalue y Maxvalue

- Uso de `MINVALUE` y `MAXVALUE` para especificar los rangos de la primera y última partición.

- Ejemplo: Primera Partición incluye cualquier fecha más antigua que el año 2000.

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
  PARTITION BY RANGE(sale_date)
    (
    STARTING MINVALUE    ENDING '12/31/1999',
    STARTING '1/1/2000'  ENDING '3/31/2000',
    STARTING '4/1/2000'  ENDING '6/30/2000',
    STARTING '7/1/2000'  ENDING '9/30/2000',
    STARTING '10/1/2000' ENDING '12/31/2004'
    );
```

Particionamiento de Tabla: Excluyendo los Límites

- Uso de la cláusula EXCLUSIVE para indicar si los límites de la partición forman parte de de ella o no.
  - Por defecto se incluyen.

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
    PARTITION BY RANGE(sale_date)
      (
      STARTING MINVALUE    ENDING '1/1/2000'  EXCLUSIVE,
      STARTING '1/1/2000'  ENDING '4/1/2000'  EXCLUSIVE,
      STARTING '4/1/2000'  ENDING '7/1/2000'  EXCLUSIVE,
      STARTING '7/1/2000'  ENDING '10/1/2000' EXCLUSIVE,
      STARTING '10/1/2000' ENDING '12/31/2004'
      );
```

Particionamiento de Tabla: Límites implícitos

- Establecimiento de los límites de forma implícita.
  - En el ejemplo el primer rango termina el ’1/1/2000’ EXCLUSIVE
  - Se pueden quitar los límites de comienzo y fin siempre que no haya rangos ambiguos.

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
 PARTITION BY RANGE(sale_date)
   (
   STARTING MINVALUE,
   STARTING '1/1/2000',
   STARTING '4/1/2000',
   STARTING '7/1/2000',
   STARTING '10/1/2000' ENDING '12/31/2004'
   );
```

Nomenclatura de las Particiones

- Se puede poner nombre a las particiones mediante la cláusula PART o PARTITION
  - Si no se especifican se generan nombres automáticamente.
  - Los nombres de la partición se usan en operaciones como DETACH.

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
  PARTITION BY RANGE(sale_date)
    (
    PART rest     STARTING MINVALUE,
    PARTITION q1 STARTING '1/1/2000',
    PARTITION q2 STARTING '4/1/2000',
    PARTITION q3 STARTING '7/1/2000',
    PARTITION q4 STARTING '10/1/2000' ENDING '12/31/2004'
    );
```

Manejo de los Nulos

- Uso the NULLS FIRST/LAST para especificar en que partición se guardan los valores nulos.
  - Por defecto van en partición que termina en MAXVALUE
  - Uso de NULLS FIRSTs para que vayan en la partición que comienza con MINVALUE.

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
    PARTITION BY RANGE(sale_date NULLS FIRST)
      (
      STARTING MINVALUE,
      STARTING '1/1/2000',
      STARTING '4/1/2000',
      STARTING '7/1/2000',
      STARTING '10/1/2000' ENDING '12/31/2004'
      );
```

Sintaxis Corta

- Uso de STARTING..ENDING ….EVERY
  - Para crear particiones homogéneas basadas en fechas o números.
  - Ejemplo que crea 20 particiones, una por trimestre:

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
  PARTITION BY RANGE(sale_date)
     (
     STARTING '1/1/2000' ENDING '12/31/2004'
       EVERY 3 MONTHS
     );
```

Sintaxis Corta y Larga

- Se puede mezclar las sintaxis corta y larga.

```
CREATE TABLE sales(sale_date DATE, customer INT, …)
  PARTITION BY RANGE(sale_date)
    (
    STARTING MINVALUE ENDING '1/1/2000' EXCLUSIVE,
    STARTING '1/1/2000' ENDING '12/31/2004'
      EVERY 3 MONTHS
    );
```

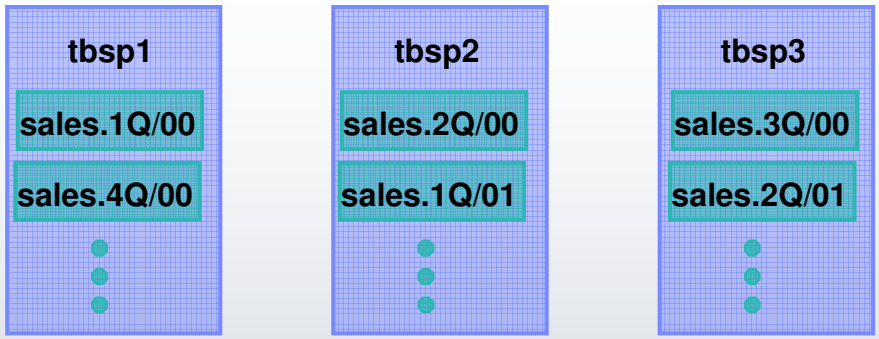
Particionamiento según Múltiples Columnas

- Se pueden especificar múltiples columnas para realizar el particionamiento.

```
CREATE TABLE sales(year INT, month INT, …)
  PARTITION BY RANGE(year, month)
    (
    STARTING (2000,1)  ENDING (2000,3),
    STARTING (2000,4)  ENDING (2000,6),
    STARTING (2000,7)  ENDING (2000,9),
    STARTING (2000,10) ENDING (2000,12),
    STARTING (2001,1)  ENDING (2001,3)
    );
```

Mapeo de Rangos en Tablespaces I

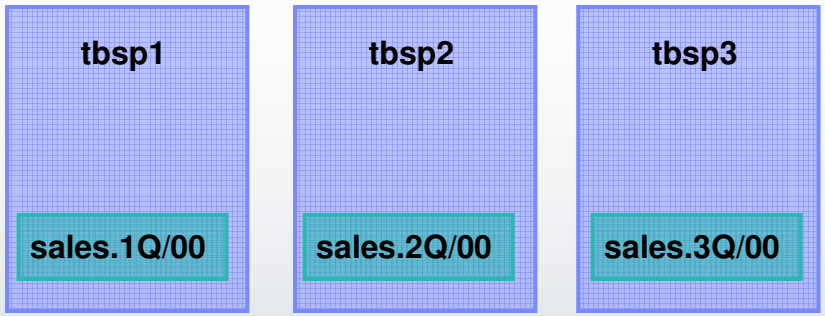
- Sintaxis corta:
  - La clausula IN acepta una lista de *Tablespaces*.
  - Modo “Round Robin”.




```
CREATE TABLE sales(sale_date DATE, customer INT, …)
  IN TBSP1, TBSP2, TBSP3
  PARTITION BY RANGE(sale_date)
    (
    STARTING '1/1/2000' ENDING '12/31/2004'
      EVERY 3 MONTHS
    );
```

Mapeo de Rangos en Tablespaces II

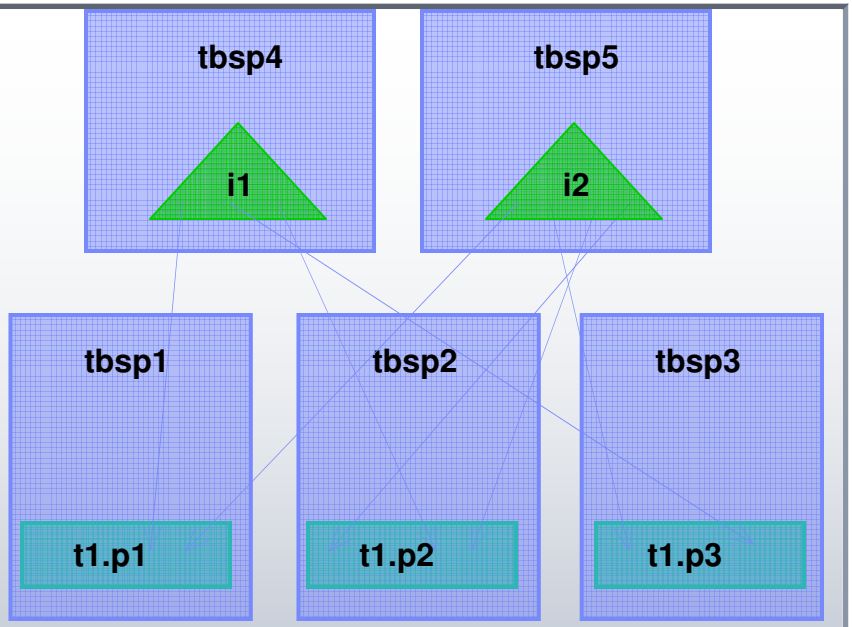
- Sintaxis Larga
  - Se especifica una *Tablespace* por cada partición.




```
CREATE TABLE sales(sale_date DATE, customer INT, …)
  PARTITION BY RANGE(sale_date)
    (
    STARTING MINVALUES  IN TBSP1,
    STARTING '3/1/2000' IN TBSP2,
    STARTING '6/1/2000' ENDING '9/30/2000' IN TBSP3
    );
```

Indices

- Indices son globales.
  - No se particionan.
  - RIDs del índice contienen ID de 2 bytes para identificar la partición.
- Cada índice se puede almacenar por separado
  - Por defecto en el mismo *Tablespace* que la primera partición.
  - Se puede especificar distintos Tablespaces vía:
    - INDEX IN en el CREATE TABLE
    - Clausula IN en el CREATE INDEX (nuevo).

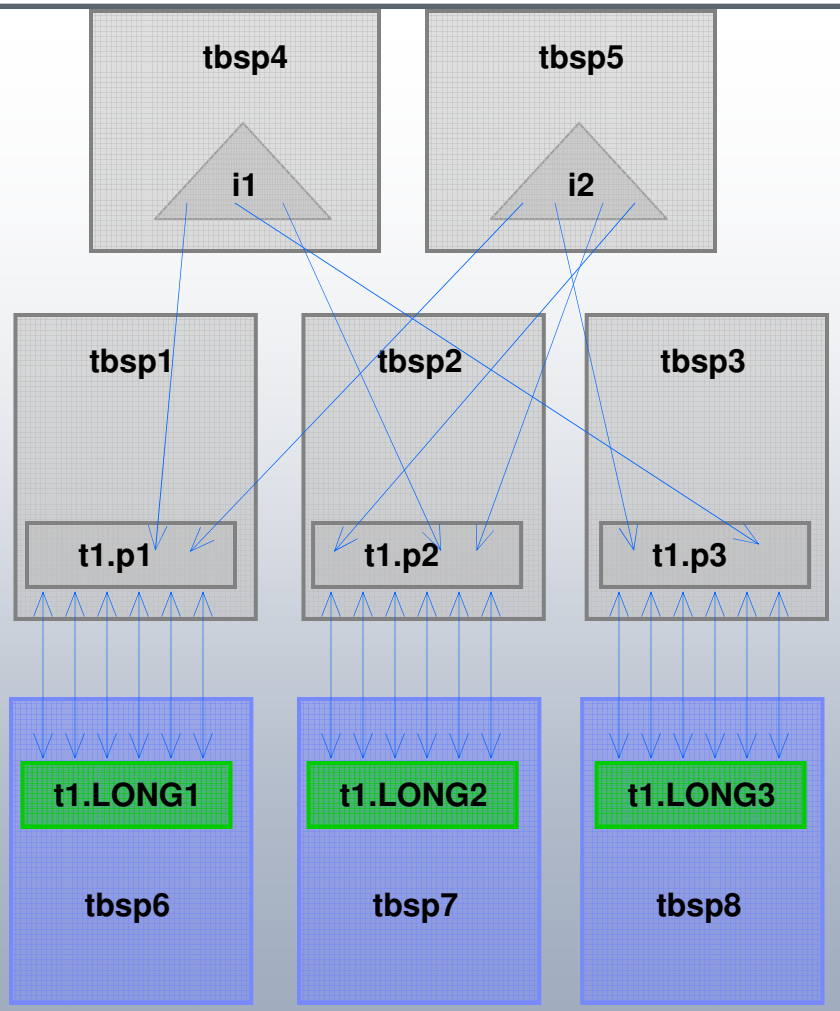



```
CREATE TABLE t1(c1 INT, c2 INT, …)
 IN tbsp1, tbsp2, tbsp3
 INDEX IN tbsp4
 PARTITION BY RANGE(c1)
 (STARTING FROM (1) ENDING (100)
   EVERY (33));
CREATE INDEX i1(c1);
CREATE INDEX i2(c2) IN tbsp5
```

LOBs

- LOBs son locales a la partición.
  - Almacenamiento distinto para cada partición.
  - Se puede especificar para cada partición vía la cláusula LONG IN

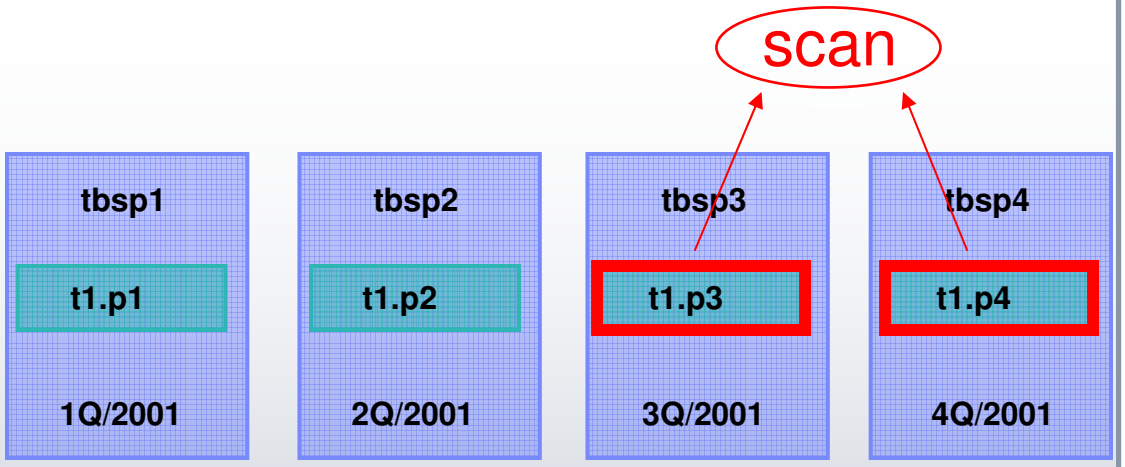
```
CREATE TABLE t1(c1 INT, c2 INT, c3 BLOB)
   IN tbsp1, tbsp2, tbsp3
   INDEX IN tbsp4
   LONG IN tbsp6, tbsp7, tbsp8
   PARTITION BY RANGE(a)
   (STARTING FROM (1)
            ENDING (100)
            EVERY (33));
CREATE INDEX i1 on t1(c1);
CREATE INDEX i2 on t1(c2) IN tbsp5;
```



“Partition Elimination” en *Table Scans*

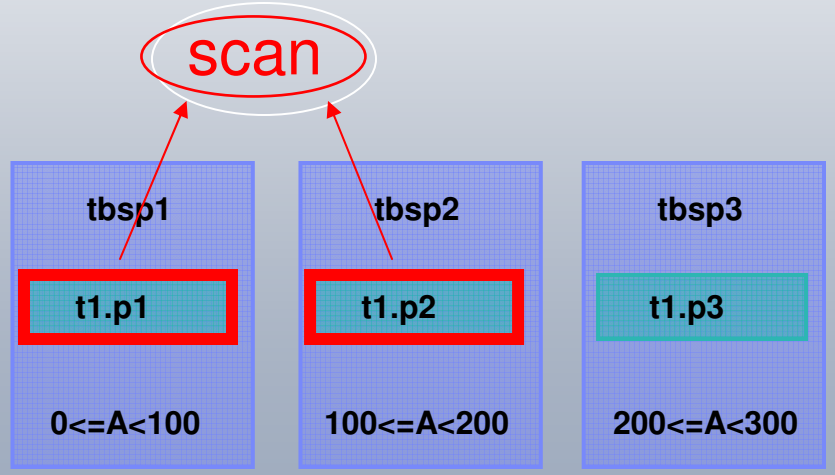
```
SELECT * FROM t1
  WHERE
  year = 2001 AND
  month > 7
```

- Sólo accederá a los *Tablespaces* tbsp3 y tbsp4.




```
SELECT * FROM t1
  WHERE
  A > 50 AND A < 150
```

- Sólo accederá a los *Tablespaces* tbsp1 y tbsp2.



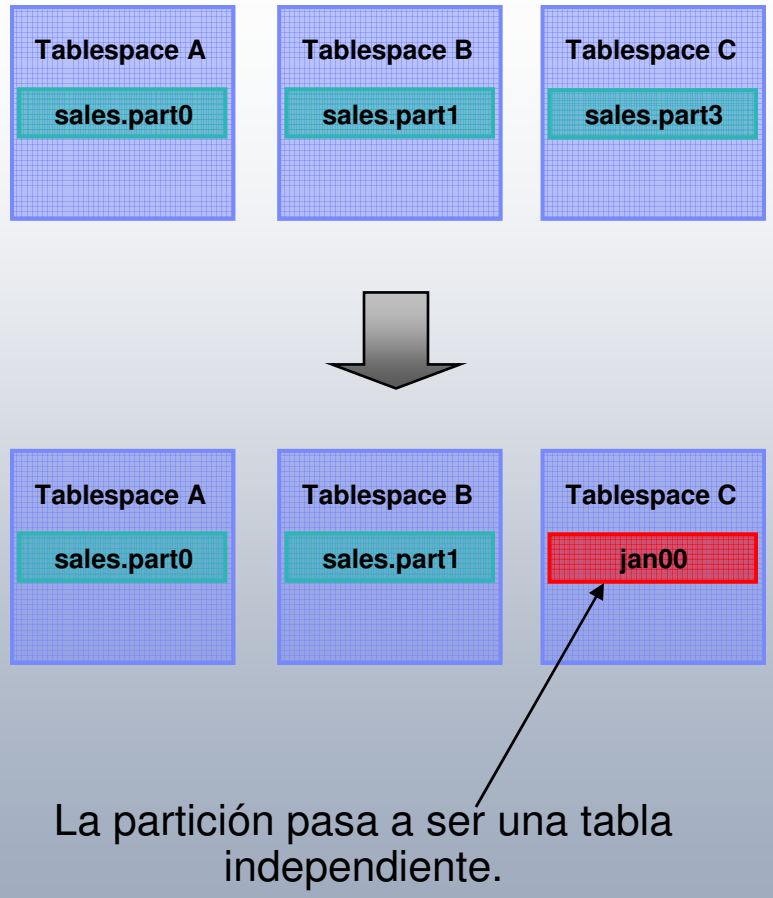
Operaciones: Roll-Out y Roll-In

- ALTER TABLE … DETACH
  - Separa una partición existente.
  - Datos se vuelven invisibles de forma instantánea
- ALTER TABLE … ATTACH
  - Incorpora una tabla como un nuevo rango
  - SET INTEGRITY para validar los datos y mantener los índices.
  - Datos son “visibles” después del commit.
- No hay movimiento de datos
- ATTACH y DETACH casi instantáneos
- SET INTEGRITY ahora ONLINE.

ROLL-OUT

- ALTER TABLE DETACH -> bloquea hasta el commit.
- COMMIT -> Datos desaparecen de la tabla inmediatamente -> Pasan a formar una nueva tabla independiente.
- Mantenimiento de índices se hace asíncronamente en *background*.
- Triggers de DELETE no se disparan con DETACH
- MQTs dependientes se quedan offline y deben ser refrescadas vía SET INTEGRITY.
- Partición separada queda accesible como tabla independiente.
- Se puede hacer un DROP o moverlo a HSM.
- Ejemplo:

```
ALTER TABLE sales DETACH
  PARTITION part3 INTO jan00
```



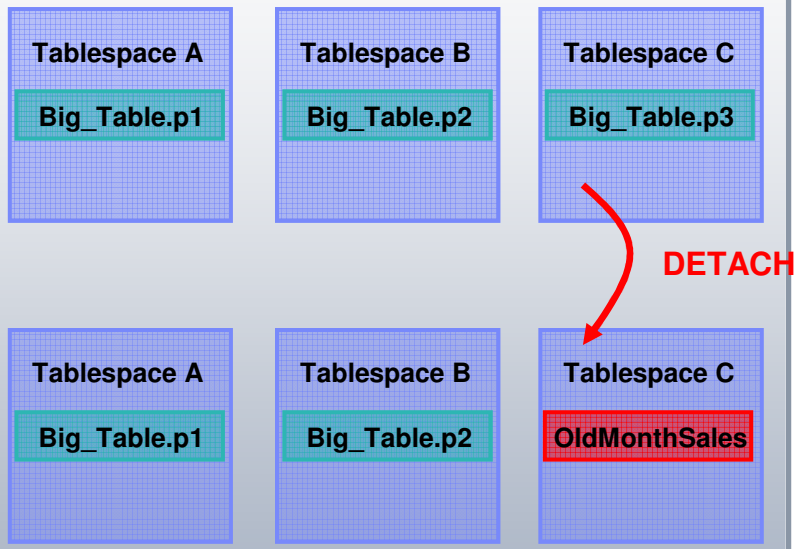
DETACH: Mantenimiento Asíncrono de los índices

- Baja prioridad y en background
- Reclama el espacio del índice correspondiente a la partición separada.
- Comienza tan pronto el DETACH se confirma (commit).
- Puede parar y continuar donde se quedó.

```
$ DB2 LIST UTILITIES SHOW DETAIL

ID                       = 3
Type                     = ASYNCH INDEX CLEANUP
Database Name            = WSDB
Partition Number         = 0
Description              = Table: T1, Index: I1
Start Time               = 12/15/2005 11:15:01.978513
State                    = Executing
Invocation Type          = Automatic
Throttling:
   Priority              = 50
Progress Monitoring:
      Total Work         = 5 pages
      Completed Work     = 0 pages
      Start Time         = 12/15/2005 11:15:01.980518

$ DB2 UTIL_IMPACT_PRIORITY FOR 3 TO 90
```

Resumen del ROLL-OUT

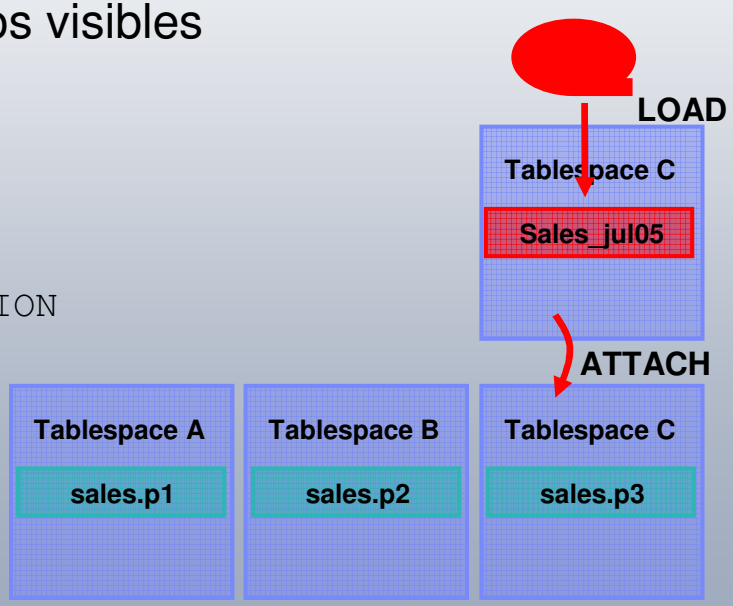
1. `ALTER TABLE Big_Table`
   `DETACH PARTITION p3`
   `INTO TABLE OldMonthSales`
   - Operación muy rápida
   - No hay movimiento de datos
   - Mantenimiento de indices en *background* asíncronamente.
2. `COMMIT`
   - Partición desaparece.
   - Resto de la tabla disponible.
3. `SET INTEGRITY FOR Mqt1, Mqt2`
   - (Opcional) MQTs
4. `EXPORT OldMonthSales; DROP OldMonthSales`
   - (Opcional): La tabla pasa a ser una tabla independiente para hacer lo que se quiera.



ROLL-IN: Pasos

1. Crear una tabla de *staging* vacia no particionada.
2. Cargar datos en la tabla de *staging.*
3. Realizar transformaciones de datos, limpieza, etc.
4. ATTACH: Añade la tabla de *staging* como una nueva partición.
5. SET INTEGRITY -> Datos Nuevos visibles

```
CREATE TABLE sales_jul05…;
 LOAD FROM file OF DEL
   REPLACE INTO sales_jul05;
 ALTER TABLE sales ATTACH PARTITION
   STARTING ('2005-07-01')
   ENDING ('2005-07-31')
   FROM sales_jul05;
```



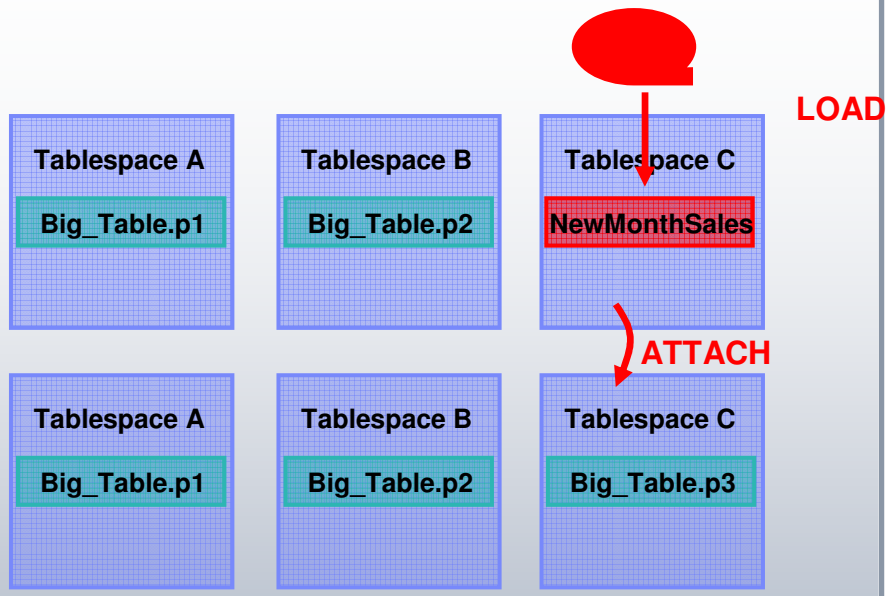
ROLL-IN: Set Integrity

- SET INTEGRITY
  - Comprueba los valores del rango.
  - Mantenimiento de índices.
  - Mantenimiento de MQT
- La tabla está *online* en todo el proceso excepto en el ATTACH.
- La nueva partición es visible al final del SET INTEGRITY.
- Recomendación: usar tabla de excepción para que deje allí las filas que no son válidas. Sin tabla de excepción cualquier violación hará fallar toda la operación.

```
SET INTEGRITY FOR
  sales ALLOW WRITE ACCESS,
  sales_by_region ALLOW WRITE ACCESS
  IMMEDIATE CHECKED INCREMENTAL
    FOR EXCEPTION IN sales USE sales_ex;
```

Resumen del ROLL-IN

- LOAD / Insert into NewMonthSales
- (Limpieza y Procesamiento)
- `ALTER TABLE Big_Table …`
  `ATTACH PARTITION …`
  `STARTING '03/01/2005'`
  `ENDING '03/31/2005'`
  `FROM TABLE NewMonthSales`
  - Operación muy rápida.
  - No hay movimiento de datos.
  - Mantenimiento de índices después.
- COMMIT
  - Nueva partición todavía no visible.
- SET INTEGRITY FOR Big_Table ……
  - Operación que puede ser larga.
    - Valida los datos.
    - Mantiene Indices y MQTs.
- COMMIT
  - Nueva partición visible.



Ejemplo de ROTATE= ROLL OUT + ROLL IN

```
ALTER TABLE Big_Table
    DETACH PARTITION Dec01
    INTO TABLE Reuse
  COMMIT

  SET INTEGRITY FOR Mqt1, Mqt2
  COMMIT (if necessary)

  LOAD FROM file OF DEL
    REPLACE INTO Reuse

  ALTER TABLE Big_Table …
    ATTACH PARTITION Mar04
    STARTING '03/01/2004'
    ENDING '03/31/2004'
    FROM TABLE Reuse
  COMMIT

  SET INTEGRITY FOR Big_Table …
  COMMIT
```

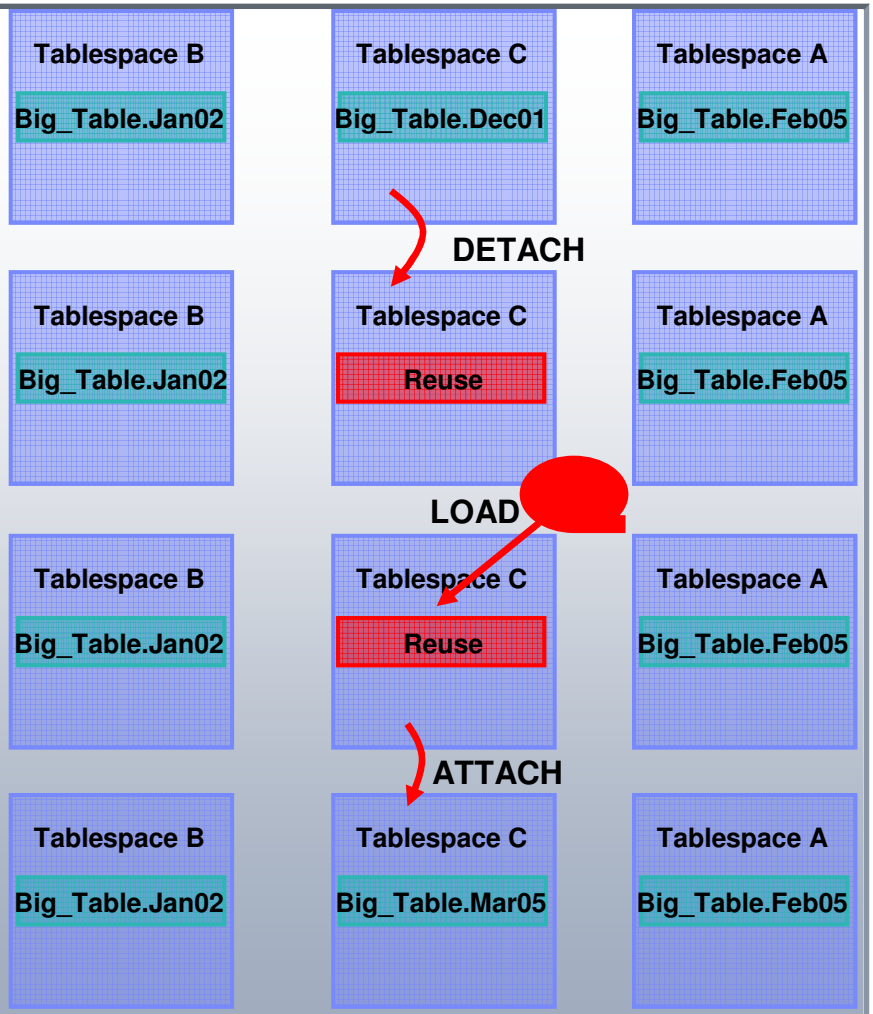
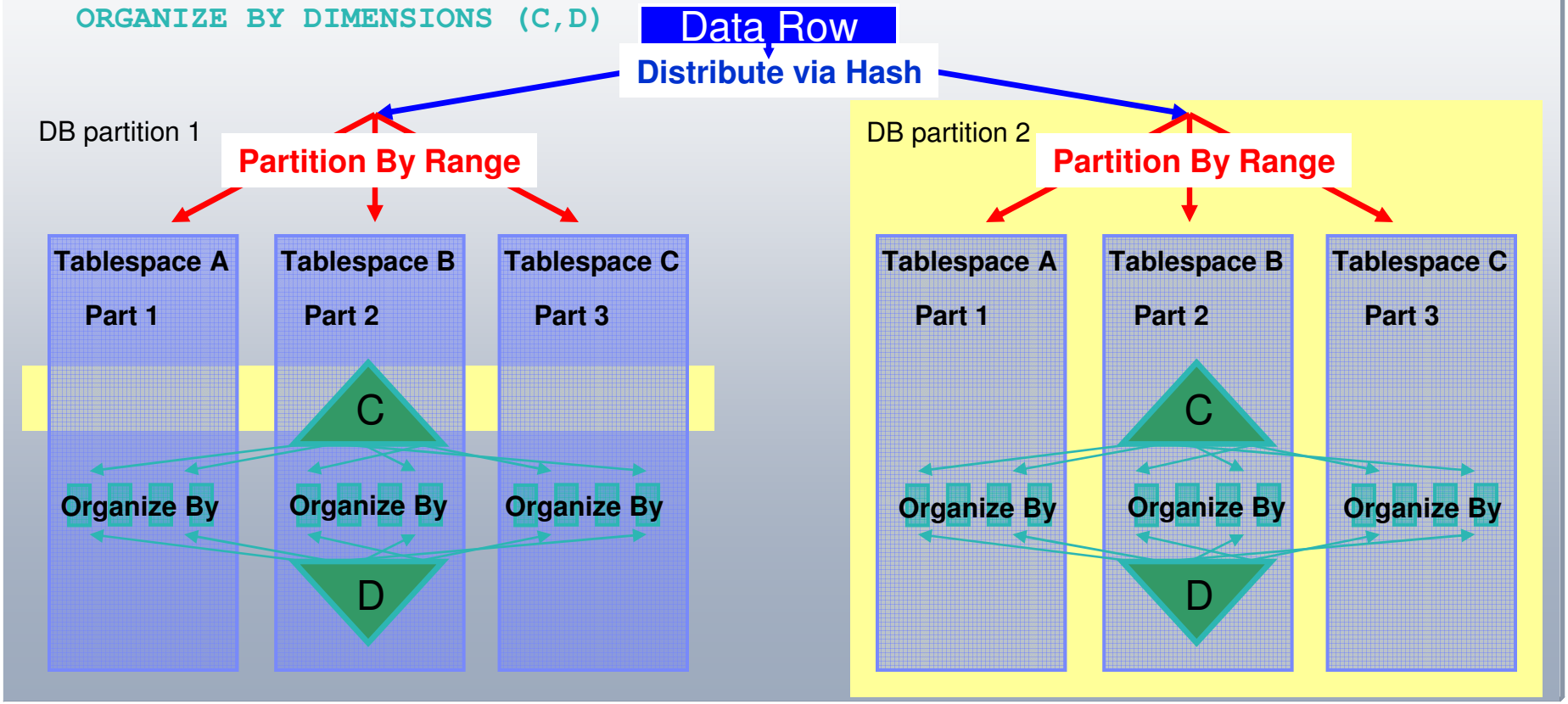


Table Partitioning + DPF + MDC

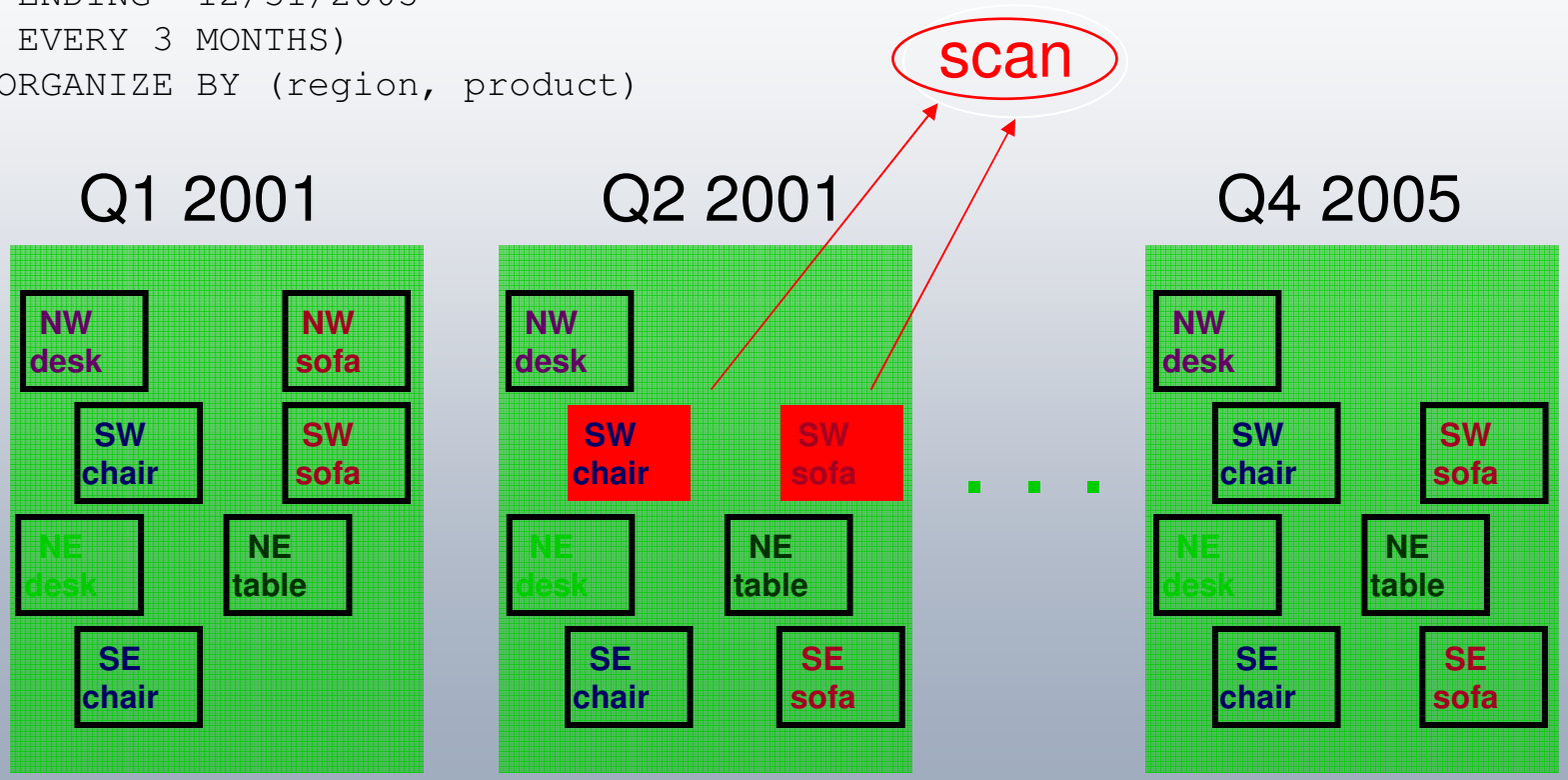
```
CREATE TABLE my_hybrid
   (A INT, B INT, C INT, D INT …)
   IN Tablespace A, Tablespace B, Tablespace C
   INDEX IN Tablespace B
   DISTRIBUTE BY HASH (A)
   PARTITION BY RANGE (B) (STARTING FROM (100) ENDING (300) EVERY (100))
   ORGANIZE BY DIMENSIONS (C,D)
```

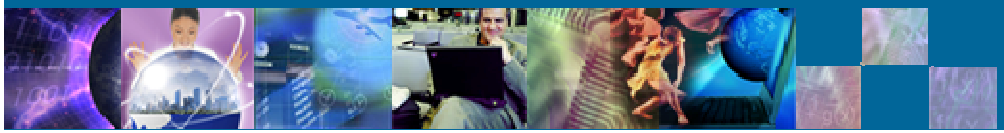


“Partition Elimination “ + “Block Elimination”

```
CREATE TABLE sales(…)
  PARTITION BY (sale_date)
  (STARTING '01/01/2001'
   ENDING '12/31/2005'
   EVERY 3 MONTHS)
  ORGANIZE BY (region, product)
```

```
SELECT * FROM sales
  WHERE sale_date > '04/01/2001' AND
  sale_date < '06/01/2001' AND region =
  'SW'
```





IBM

DB2 V9

Gestión Automática del Almacenamiento

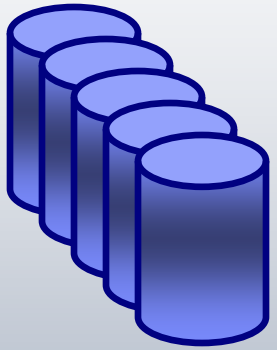
ON DEMAND BUSINESS™

Almacenamiento Automático: Conceptos

- Existe desde la versión 8.2.2 (Saturno).
- Punto único de gestión del almacenamiento para Tablespaces.
- Cuando se crea la base de datos el usuario especifica los dispositivos de almacenamiento donde DB2 creará los Tablespaces.
- Una vez habilitado se puede crear un Tablespace sin especificar *containers*.
- Los *containers* se crean automáticamente en los dispositivos especificados al crear la base de datos.
- El crecimiento y la adición nuevos *containers* lo maneja DB2 transparente al usuario.
- Se pueden añadir nuevos dispositivos una vez creada la base de datos.

Almacenamiento Automático: Sintaxis

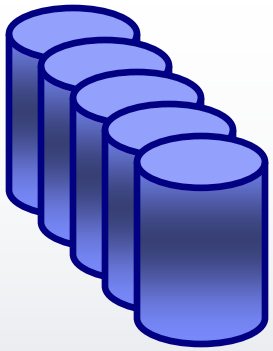
```
CREATE DATABASE DB1
    AUTOMATIC STORAGE YES (por defecto)
    ON /data/path1, /data/path2

CREATE TABLESPACE TS1

CREATE TABLESPACE TS2
        INITIALSIZE 500 K
        INCREASESIZE 100 K
        MAXSIZE 100 M

CREATE TABLE … .IN TS1
```



Almacenamiento Automático: Restore

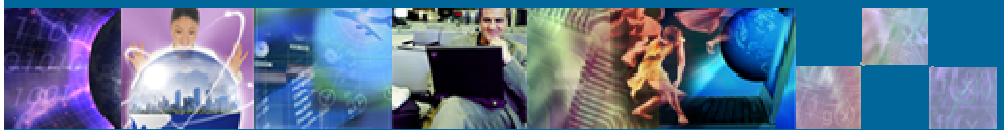
- RESTORE DATABASE TEST1

- RESTORE DATABASE TEST3
  ON /path1, /path2, /path3



- Si la cláusula ON se especifica se usan estos *paths* en vez de los almacenados en la imagen del *backup*.

- Si la cláusula ON no se especifica se matienen los *paths* especificados en el *backup*.

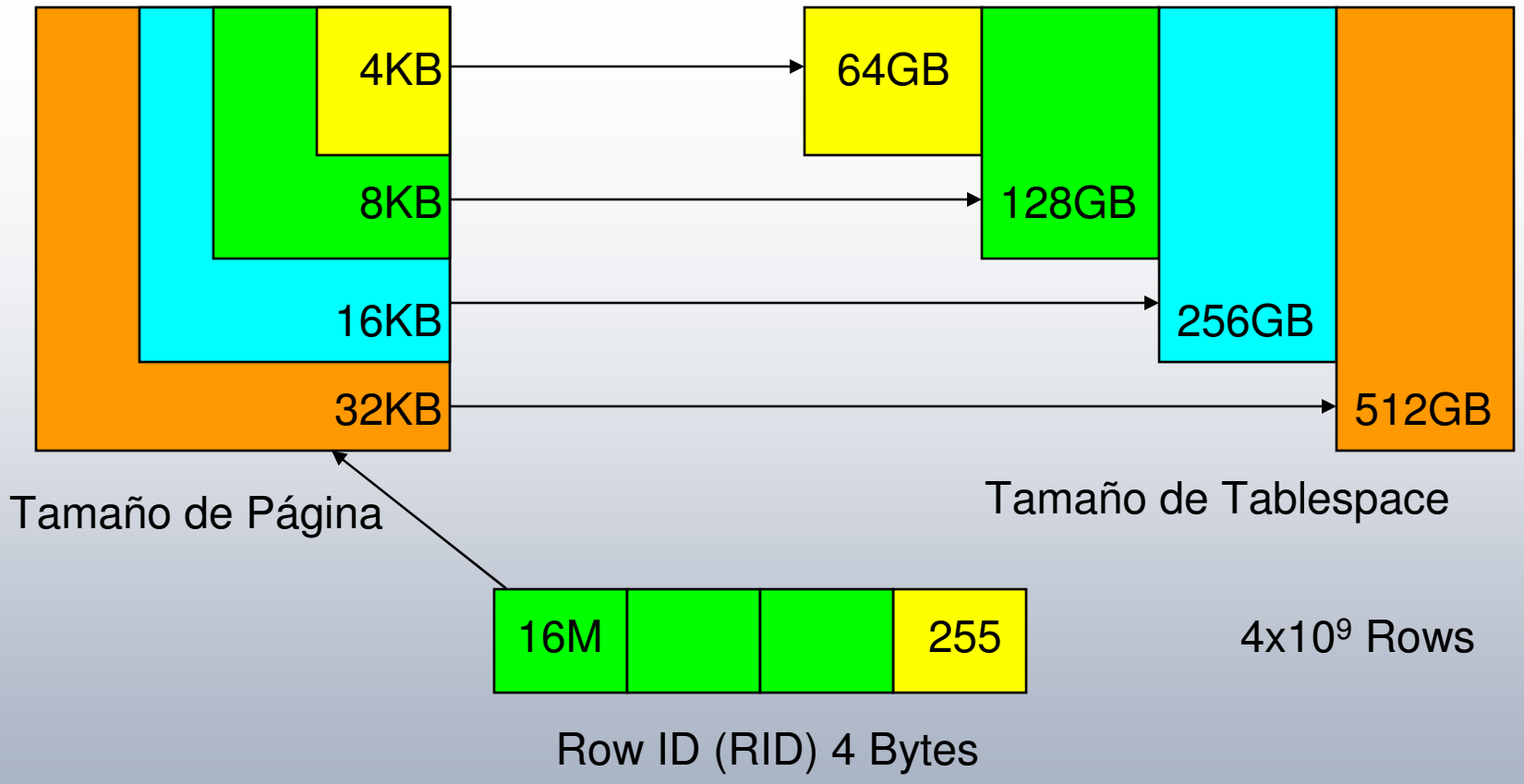


DB2 V9

Incremento de Límites

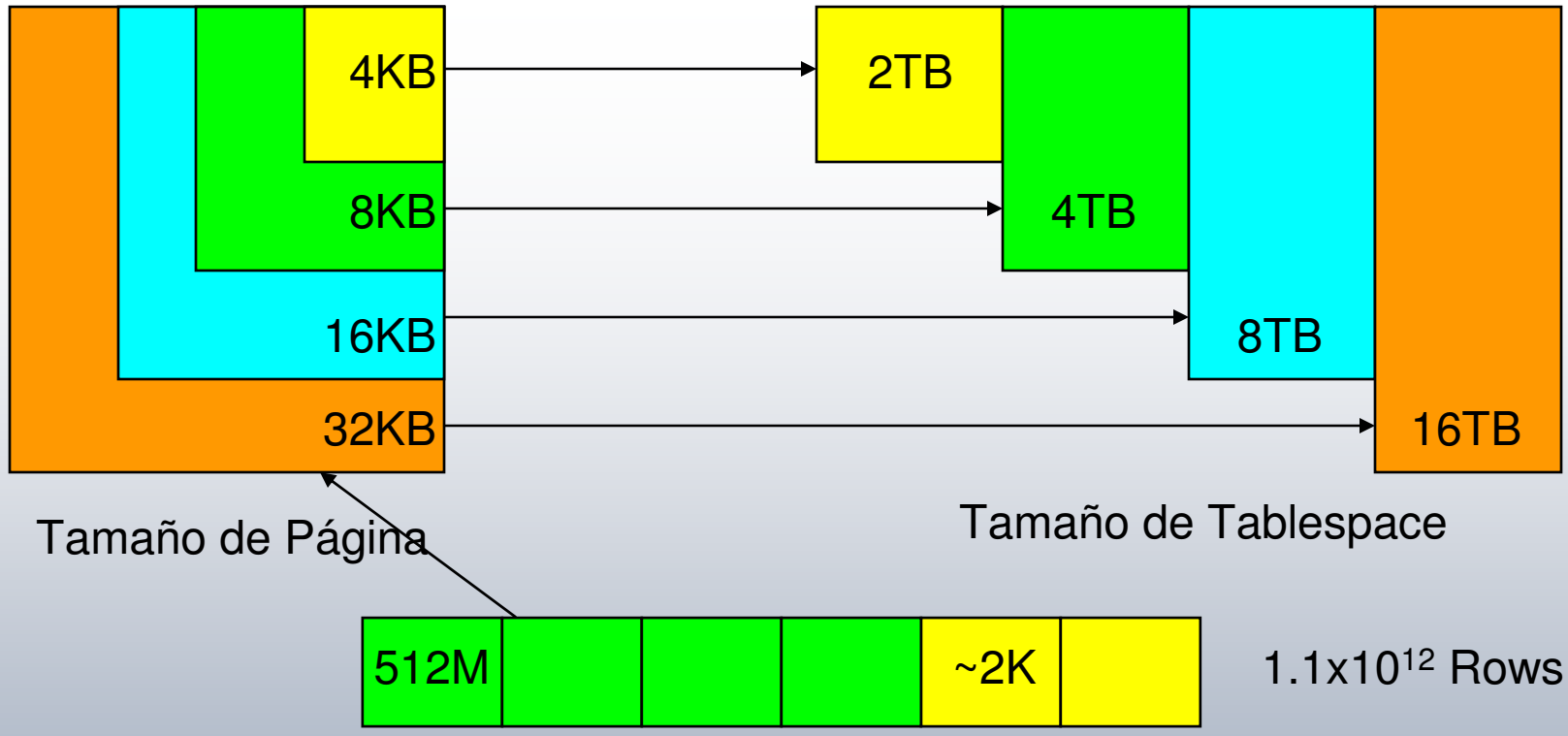
ON DEMAND BUSINESS™

Diseño de Tablespace Anterior a DB2 V9



Para las tablas en cualquier TS (regular, temporary, DMS, SMS)

Nuevo Diseño: Tablespace LARGE



Para tablas en Tablespaces LARGE (DMS sólo)

Conversión de Tablespace (1/2)

- Nuevo tipo de Tablespace: LARGE.
- Conversión de REGULAR a LARGE.
- ALTER TABLESPACE <name> CONVERT TO LARGE
  - Opción CONVERT TO LARGE no puede ser combinada con más opciones.
  - Operación “logeada” soporta ROLLBACK y RESTORE/ROLLFORWARD.
  - Devuelve Warning: SQL1237W.
- Si la tabla esta particionada deben estar todas las particiones en el mismo tablespace.
  - Uso de LARGE tablespace para tablas particionadas (recomendación).
- Rápido pero produce ciertos bloqueos en:
  - SYSCAT.TABLESPACES
  - SYSCAT.DATAPARTITIONS
  - SYSCAT.INDEXES
  - SYSCAT.TABLES
  - El tablespace está OFFLINE hasta que termina el ALTER.

Conversión de Tablespace (2/2)

- Recomendaciones:
  - Realizar ALTER TABLESPACE durante la migración a V9.
  - Reorganizar las tablas después.
    - >255 filas por página
  - Crear Tablas Particionadas en LARGE Tablespace.

Número de Registros por Página

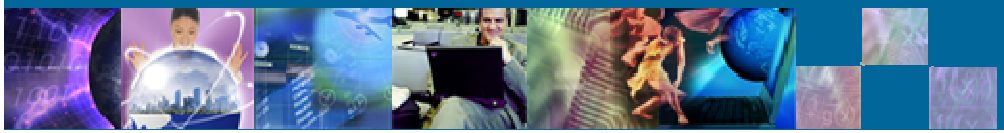
Tamaño de Página	*TABLESPACE REGULAR* *Máximo Número de Registros por Pág.*	*TABLESPACE LARGE* *Máximo Número de Registros por Pág.*
4 KB	*251*	*287*
8 KB	*253*	*580*
16 KB	*254*	*1165*
32 KB	*253*	*2335*

Tamaño del Indice

Versión	Número de columnas en la clave del índice.
Pre-Viper	16
Post-Viper	64

Resumen

- Nuevo 6 bytes RID para Tablespaces de tipo LARGE
  - Sólo para DMS
  - Se pueden crear tablas de tamaño de Teras.
  - Más filas en cada página.
- Los Tablespaces existentes se pueden migrar al nuevo tipo LARGE.
- Los Indices soportan hasta 64 columnas.

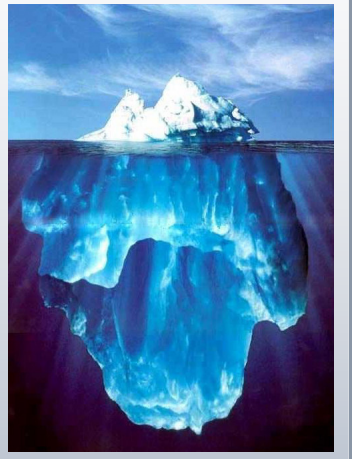


DB2 V9

Otras Características

Otras Carácterísticas

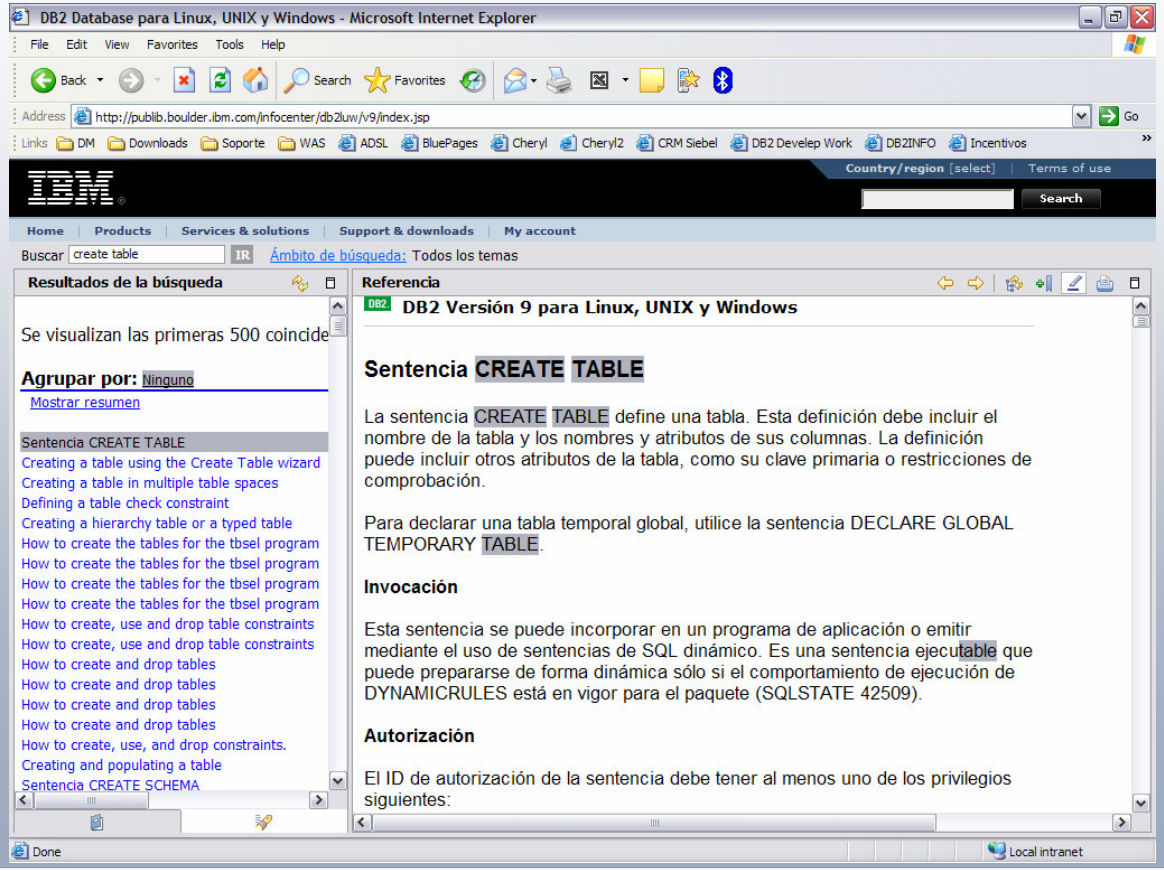
- Multiples instalaciones de DB2.
- Cliente ligero de verdad! (fichero zip).
- Mejoras en los drivers JDBC y SQLj.
- Restore a nivel de Tablespace sin full backup.
- Soporte de IPV6.
- Soporte automático para IO_SERVER y AVG_APPLS.
- Nuevas operaciónes a nivel de SCHEMA (COPY, DROP…).
- Nuevo Developer Center -> Workbench (Eclipse).
- Inplace ALTER.
- Reconstrucción de Indices asíncronamente después de Restart.
- Generación de *scripts* para Redirect Restore.
- Nuevas funciones de seguridad (RESTRICT Create DB..)
- ……



Más información

- INFORMATION CENTER

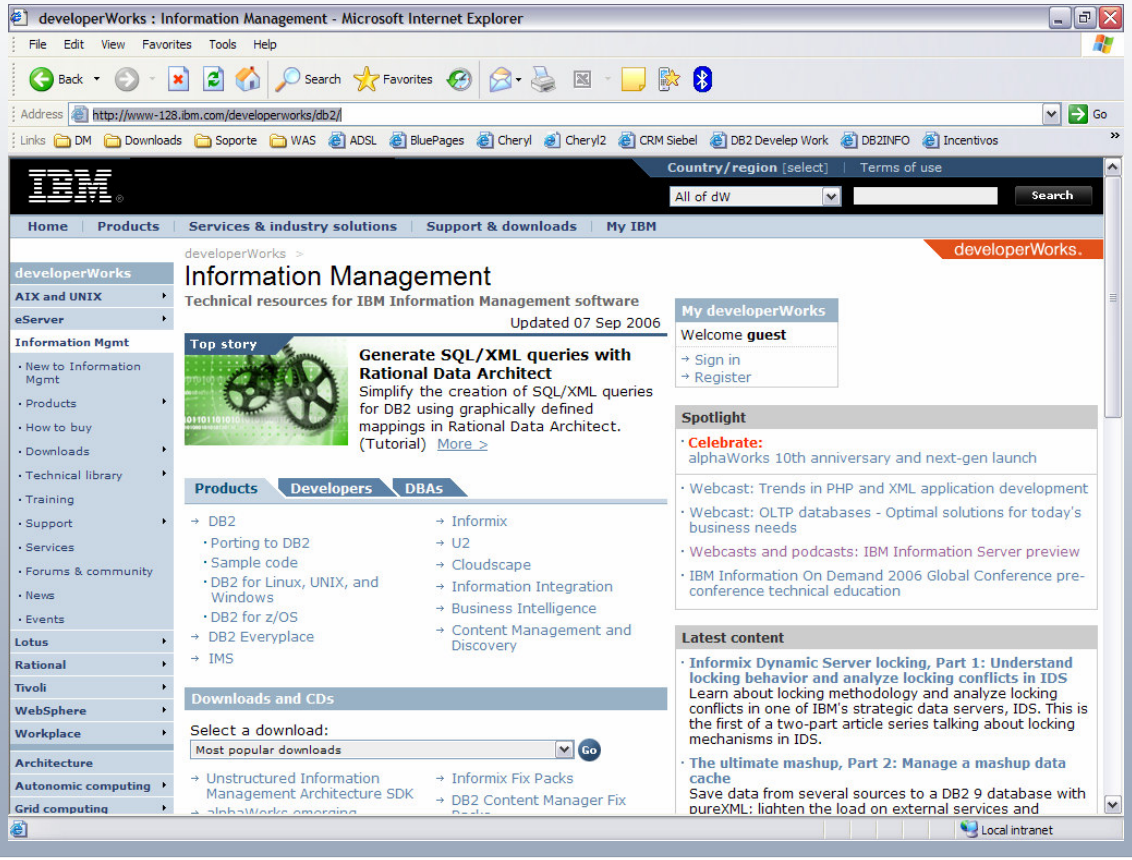
http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp



Más información

- DEVELOPERS WORKS

http://www.ibm.com/developerworks/db2/



Más información

- MANUALES EN PDF
  - Se bajan desde la página de los fixpacks.
  - Vienen con un índice.