Pseudo-códigos para funciones de comunicación de tablas SQL

La función **distribuye_tabla**(*Tab_in*, *K*) se ejecuta en paralelo en cada procesador y toma como entrada una tabla *Tab_in* y la identificación de un campo llave *K* de la misma tabla. Sobre el campo *K* se aplica una función de hashing para determinar a cual procesador se envía cada tupla de la tabla *Tab_in*. Las tuplas enviadas en la forma de mensajes a los distintos procesadores son posteriormente recolectadas en cada procesador y son almacenadas en una tabla de salida *Tab_out* creada en cada procesador. Es decir, tanto la tabla *Tab_in* como la tabla *Tab_out* contienen sus tuplas distribuidas en cada procesador.

```
tabla distribuye_tabla( tabla Tab_in, llave K ) {
      // Una instancia se ejecuta en paralelo en cada procesador.
      // Cada instancia actúa sobre las tuplas almacenadas en su
      // respectivo procesador.

      // envía las tuplas del procesador actual al procesador destino
      foreach tupla t in Tab_in do {
         bsp_send( hash( t.K ), message( t ) );
      }

      bsp_sync(); // sincronización de procesadores (recepción mensajes)

      // recibe las tuplas y las almacena en la tabla de salida
      Tab_out= crea_tabla();
      while( bsp_next_message() != void() ) {
         tupla= bsp_get_message(); // recibe mensaje conteniendo tupla
         Tab_out.insert( tupla );
      }
      return Tab_out;
}
```

Ejemplo de funcionamiento. Suponemos 2 procesadores.

EMPLEADOS (procesador 0)

RUT	NOMBRE	DIR
1111-1	N1	D1
4444-4	N4	D4

EMPLEADOS (procesador 1)

RUT	NOMBRE	DIR
2222-2	N2	D2
3333-3	N3	D3

Resultados que entrega la función *hash*(rut)

Hash(1111-1) --> procesador 1

Hash(2222-2) --> procesador 0

```
Hash(3333-3) --> procesador 1
Hash(4444-4) --> procesador 0
```

Si cada procesador ejecuta en paralelo

```
DEMP = distribuye_tabla( EMPLEADOS, RUT );
```

Entonces el resultado es

DEMP (procesador 0)

RUT	NOMBRE	DIR
2222-2	N2	D2
4444-4	N4	D4

DEMP (procesador 1)

RUT	NOMBRE	DIR
1111-1	N1	D1
3333-3	N3	D3

La función **envia_a_todos**(*T_in*) se ejecuta en paralelo en cada procesador. Cada instancia de la función toma como entrada una tabla *Tab_in* y la envía a todos los otros procesadores. En cada procesador, cada instancia de la función entrega como resultado una tabla *Tab_out* que es la unión de todas las tablas recibidas desde los otros procesadores (incluido a sí mismo). El supuesto es que la *Tab_in* es una tabla que contiene sus tuplas distribuidas en los procesadores. Es decir, en cada procesador se tiene el mismo esquema de tablas y las tuplas de cada tabla están distribuidas en los procesadores.

```
tabla envia_a_todos( tabla T_in ) {
      // Una instancia se ejecuta en paralelo en cada procesador.
      // Cada instancia actúa sobre la tabla almacenada en su
      // respectivo procesador.

      // Envía la tabla del procesador actual a todos los procesadores.
      // Es un patrón de comunicación all-to-all.

      for( int i= 0; i<P; i++ )
         bsp_send( i, new message( T_in ) );

      bsp_sync(); // sincronización de procesadores (recepción mensajes)

      Tab_out= crea_tabla();
      while( bsp_next_message() != void() ) {
         tabla= bsp_get_message(); // recibe cada tabla
         Tab_out.union( tabla ); // forma la union de tablas
      }
      return Tab_out;
}
```

Ejemplo de funcionamiento. Suponemos 2 procesadores.

EMPLEADOS (procesador 0)

RUT	NOMBRE	DIR
1111-1	N1	D1
4444-4	N4	D4

EMPLEADOS (procesador 1)

RUT	NOMBRE	DIR
2222-2	N2	D2
3333-3	N3	D3

Si cada procesador ejecuta en paralelo

```
DEMP = envia_a_todos( EMPLEADOS );
```

Entonces el resultado es

DEMP (procesador 0)

RUT	NOMBRE	DIR
1111-1	N1	D1
2222-2	N2	D2
3333-3	N3	D3
4444-4	N4	D4

DEMP (procesador 1)

RUT	NOMBRE	DIR
1111-1	N1	D1
2222-2	N2	D2
3333-3	N3	D3
4444-4	N4	D4

La función **recibe_resultados**(P) es ejecutada por un procesador maestro o broker para esperar por los resultados provenientes desde un conjunto de *P* procesadores esclavos que contienen la base de datos.

```
tabla recibe_resultados( P ) {
      Tab_out= crea_tabla();
      for( int i= 0; i<P; i++ ) {
         tabla= bsp_get_async_message(); // espera por siguiente mensaje
         Tab_out.union( tabla );
      }
      return Tab_out;
}
```

El patrón de ejecución del procesador maestro para solicitar la ejecución de la solución de una consulta a *P* procesadores que mantienen la base de datos distribuida entre ellos (mismos esquemas de tablas en cada procesador con tuplas distribuidas uniformemente en los procesadores, es el siguiente.

Procesador maestro ::

```
  for( int i= 0; i<P; i++ )
      bsp_send( i, new message( "ID consulta" ) );

  resultado = recibe_resultados(); // recibe tabla total de resultados
```

fin procesador maestro