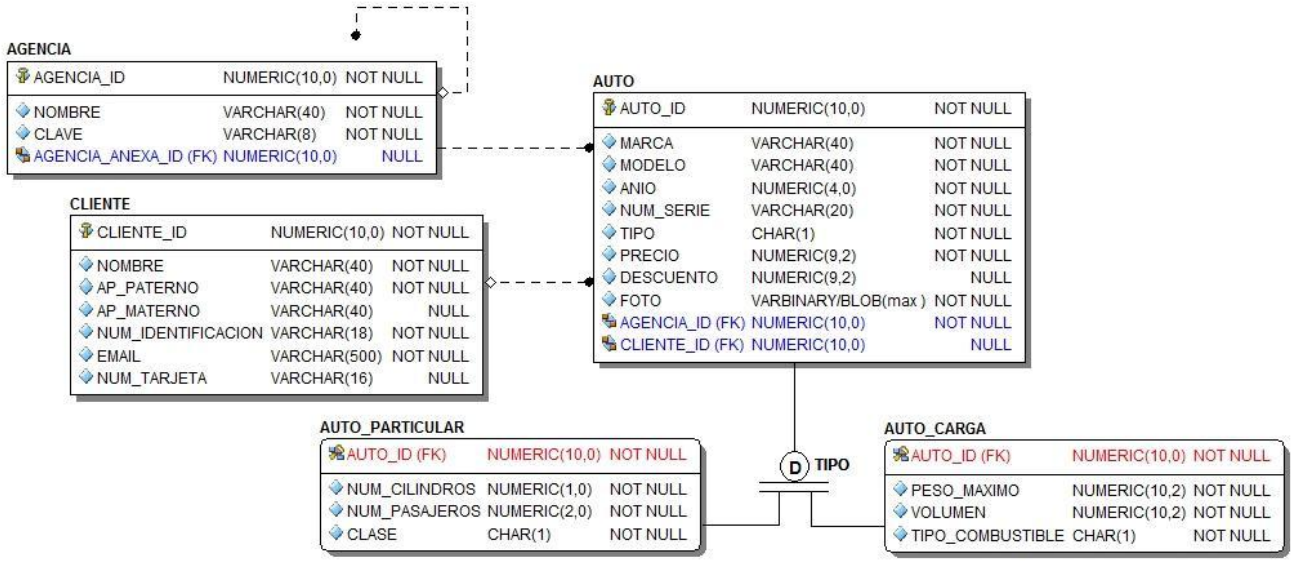
5. OPTIMIZACIÓN DE CONSULTAS DISTRIBUIDAS - PARTE 4

5.5. Optimización de consultas distribuidas.

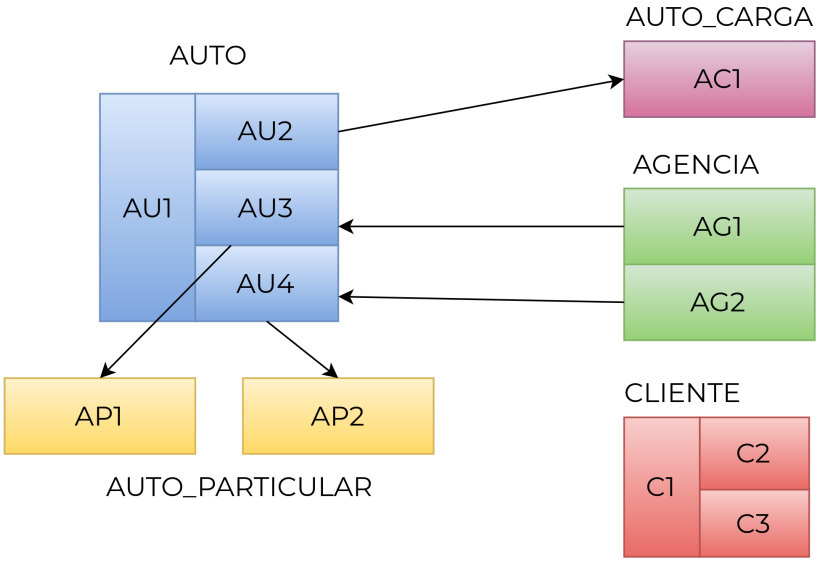
- Características de los DBMS cuyo principal objetivo es la reducción de la cantidad de datos a transferir entre sitios en transacciones que reciben datos de tablas remotas referenciadas en sentencias SQL distribuidas.
- Sentencias distribuidas hacen uso de optimizadores basados en costos que generan sentencias SQL que extraen únicamente los datos necesarios de tablas remotas.
- Los datos que se extraen pueden ser procesados en el sitio remoto o localmente, reduciendo así los tiempos de transmisión.
- El uso de ***hints*** permite influenciar al optimizador para modificar las decisiones anteriores: `driving_site`, `no_merge`, `index`.
- La BD divide una sentencia distribuida en un conjunto de sentencias remotas mismas que son enviadas a otros nodos para ser procesadas.
- El nodo remoto procesa la sentencia y regresa los resultados.
- El nodo local realiza un post – procesamiento y regresa los datos al usuario.
- Las siguientes técnicas se emplean para optimizar el procesamiento de sentencias distribuidas.
    - Collocated Inline views
    - Cost-Based Optimization
    - Uso de Hints
    - Uso de semi – Joins
    - Análisis manual del plan de ejecución.

5.5.1. Caso de estudio.

Para ilustrar las siguientes secciones, considerar el modelo relacional de un sistema de control de agencias de autos y su correspondiente esquema de fragmentación.



5.5.1.1. Esquema de fragmentación



Expresiones del álgebra relacional

Agencia

$$Ag_1 = \sigma_{substr(clave,1,1)\ between\ 'A'\ and\ 'M'}(agencia)$$

$$Ag_2 = \sigma_{substr(clave,1,1)\ between\ 'N'\ and\ 'Z'}(agencia)$$

Auto

$$Au_1 = \pi_{auto-id,foto,precio}(auto)$$

$$Au_1' = \pi_{auto-id,marca,modelo,anio,num-serie,tipo,precio,descuento,agencia-id,cliente-id}(auto)$$

$$Au_2 = \sigma_{tipo='C'}(Au_1')$$

$$Au_3' = \sigma_{tipo='P'}(Au_1')$$

$$Au_3 = Au_3' \ltimes_{agencia-id}(Ag_1')$$

$$Au_4 = Au_3' \ltimes_{agencia-id}(Ag_2')$$

Auto carga

$$AC_1 = auto - carga \ltimes_{auto-id} Au_2$$

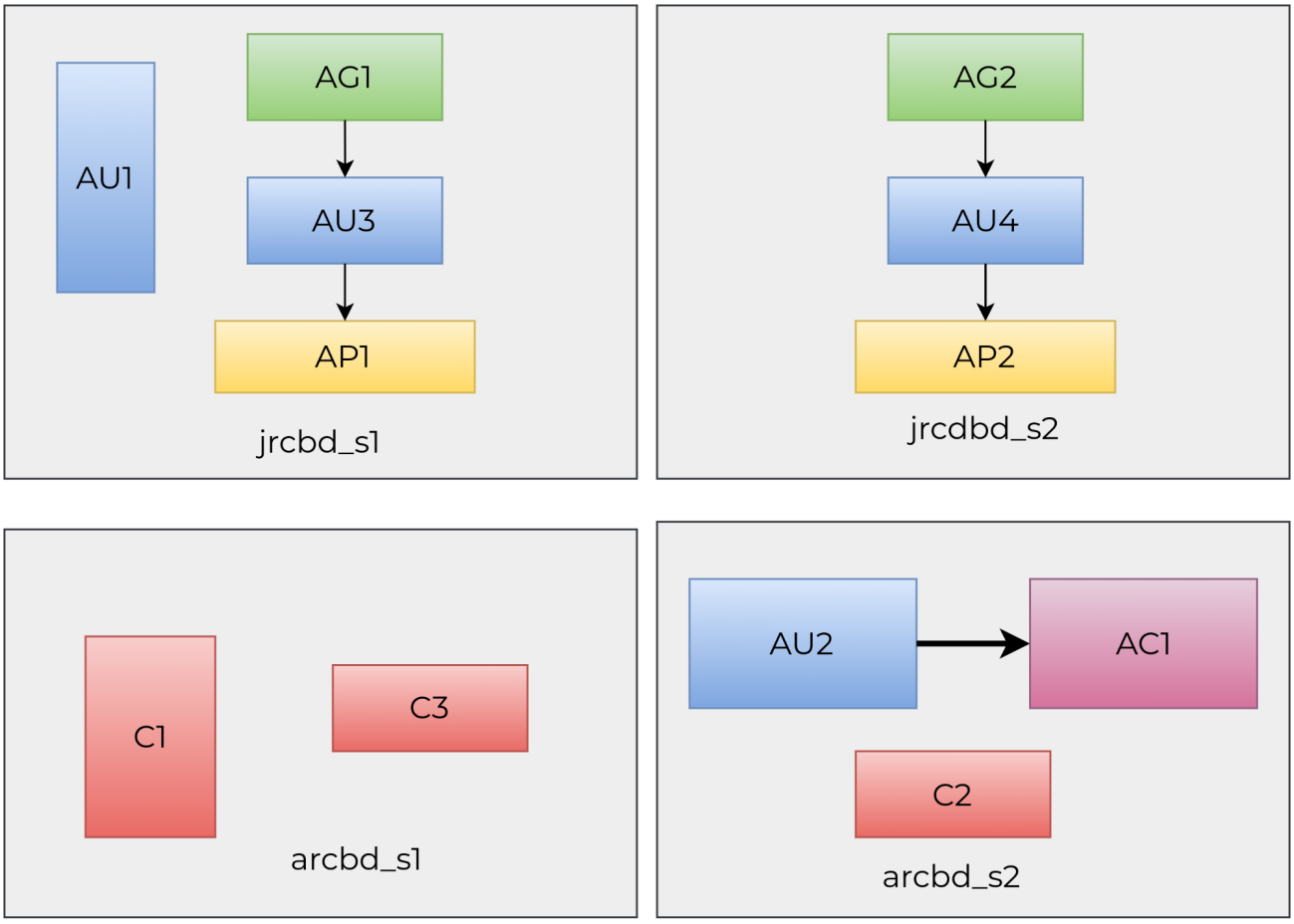
Auto particular

$$AP_1 = auto - particular \ltimes_{auto-id} (Au_3)$$
$$AP_2 = auto - particular \ltimes_{auto-id} (Au_4)$$

Cliente

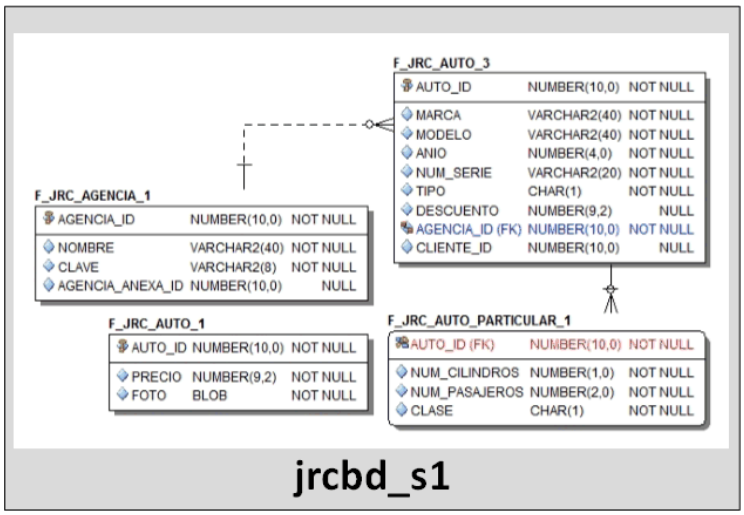
$$C_1 = \pi_{cliente-id,num\_tarjeta}(cliente)$$
$$C_2' = \pi_{cliente-id,nombre,ap-paterno,ap-materno,num-identificacion,email}(cliente)$$
$$C_2 = \sigma_{substr(num-identificacion,1,1)\ between\ 'A'\ and\ 'M'}(C_2')$$
$$C_3 = \sigma_{substr(num-identificacion,1,1)\ between\ 'A'\ and\ 'M'}(C_2')$$

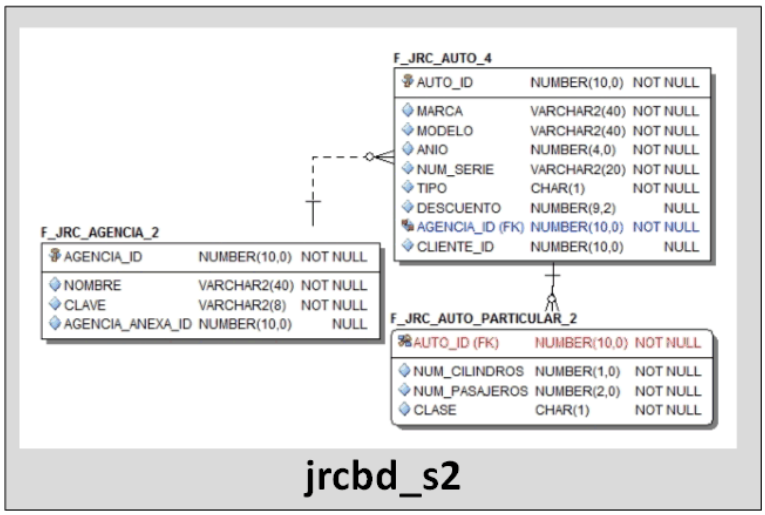
5.5.1.2. Esquema de localización

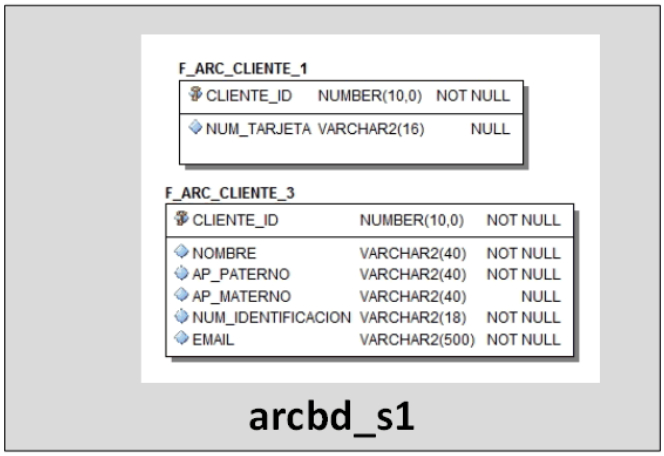


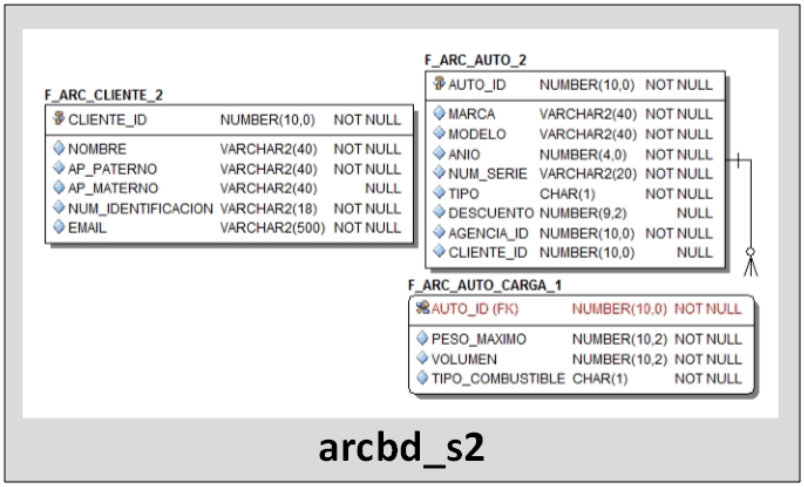
Modelo relacional

- Considerar que las FKs están indexadas para mejorar desempeño.









5.5.2. Uso de Collocated Inline Views

- El término ***collocated*** en BDD se refiere al conjunto de tablas o fragmentos que se encuentran dentro de un mismo nodo.
- El término ***Inline views*** se refiere a una sentencia `select` que toma el lugar de una tabla dentro de otra sentencia sql. Dicho de otra forma, una subconsulta en la cláusula `from`.

Ejemplo

```
select  a3.auto_id,a3.marca,a3.modelo, c1.cliente_id,c1.num_tarjeta
from f_jrc_auto_3 a3
join (select  cliente_id,num_tarjeta
  from f_arc_cliente_1@arcbd_s1
) c1
on a3.cliente_id = c1.cliente_id;
```

- ***Collocated inline view***: Inline view (subconsulta en la cláusula from) que selecciona datos de múltiples tablas, pero de una sola base de datos.
- En una consulta distribuida se recomienda organizar la sentencia en ***collocated inline views***: Esto permite ejecutar una sola vez la sentencia en un solo sitio en lugar de acceder a las tablas varias veces desde otros nodos.

5.5.2.1. Reescritura de sentencias distribuidas empleando collocated inline views.

- Una de las tareas principales de un optimizador distribuido es reescribir una sentencia para hacer uso de ***collocated inline views***.
- No siempre es posible realizar la reescritura, sobre todo en caso donde la sentencia contiene:
  - Funciones de agregación
  - Subconsultas
  - Código SQL complejo.
- Si la consulta contiene alguno de estos 3 elementos, aún es posible optimizarla de forma manual empleando ***hints***.
- Adicionalmente a la transformación de la sentencia distribuida en ***collocated inline views***, el optimizador hace uso de las estadísticas disponibles para reducir los costos de procesamiento.

Ejemplo

Mostrar el identificador del auto, marca, modelo, peso máximo, volumen y precio de los Autos de carga ubicados en `arcbdd_s2`. La consulta se lanza desde `jrcbdd_s1`.

```
select a2.auto_id,a2.marca,a2.modelo,ac1.peso_maximo,ac1.volumen,a1.precio
from f_arc_auto_2@arcbd_s2 a2,
  f_arc_auto_carga_1@arcbd_s2 ac1,
  f_arc_auto_1 a1
where a2.auto_id = ac1.auto_id
and a2.auto_id = a1.auto_id;
```

- Transformar la sentencia anterior empleando ***collocated inline views***:

```
select r.auto_id,r.marca,r.modelo,r.peso_maximo,r.volumen,a1.precio
from (
  select a2.auto_id,a2.marca,a2.modelo,ac1.peso_maximo,ac1.volumen
  from f_arc_auto_2@arcbd_s2 a2,f_arc_auto_carga_1@arcbd_s2 ac1
  where a2.auto_id = ac1.auto_id
) r , f_arc_auto_1 a1
where a2.auto_id = r.auto_id;
```

- Esta estrategia reduce la cantidad de consultas que se lanzarán al sitio remoto, en este caso al sitio `arcbd_s2` ya que se crea un collocated inline view que incluye a 2 tablas que se encuentran en el mismo sitio, se envía una sola en lugar de realizar varios accesos remotos.
- La transformación de esta sentencia puede ser realizada por el optimizador sin intervención del usuario.

5.5.3. Configuración del parámetro optimizer_mode.

- Este parámetro permite modificar el comportamiento del optimizador con base a las siguientes opciones:

```
optimizer_mode={ first_rows_[1 | 10 | 100 | 1000] | first_rows | all_rows }
```

- `first_rows_n`: Optimiza con el propósito de obtener el mejor tiempo posible para regresar los primeros n registros, n = 1, 10, 100, 1000.
- `first_rows`: Realiza una mezcla de costos y heurísticas para encontrar el mejor plan de ejecución para recuperar un conjunto pequeño de los primeros registros.
- `all_rows`: El objetivo principal es hacer un uso óptimo de recursos (minimizar uso de recursos) para obtener todos los registros de la consulta solicitada. Representa la opción por default.

5.5.4. Cálculo de estadísticas.

- Recordando los conceptos vistos anteriormente, la recolección de estadísticas debe realizarse de forma local en cada nodo empleando los siguientes paquetes

```
gather_index_stats
gather_table_stats
gather_schema_stats
gather_database_stats
```

5.5.5. Uso de hints

- Si la sentencia no logra ser optimizada lo suficiente, el uso de ***hints*** permite influenciar o guiar al optimizador para generar planes de ejecución alternativos que pudieran conducir a mejores resultados.
- Para casos específicos de consultas distribuidas, existen 2 hints comunes: `no_merge` y `driving_site`.

5.5.5.1. No merge

- Instruir al optimizador para evitar un merge o fusión de ***collocated Inline views***.

- Dicho de otra forma, evita que el optimizador elimine la subconsulta que representa a la collocated inline view y de esta manera reducir la cantidad de sentencias SQL que se envían a un mismo sitio remoto, y por lo tanto, la reducción de tráfico de red.

Ejemplo

```
select /*+no_merge(v)*/ t1.x, v.avg_y
from t1, (
  select x, avg(y) as avg_y
  from t2 group by x
) v,
where t1.x = v.x
and t1.y = 1;
```

- En el ejemplo anterior, el hint se escribe justo después de la cláusula select de la consulta externa.
- Observar que el hint recibe un parámetro cuyo valor corresponde con el alias que se le asigna a la collocated inline view, en este caso, se utiliza la letra v.
- Notar que la collocated inline view tiene en su interior una función de agregación. Esto provocará que la subconsulta sea desintegrada formando una sola instrucción, sin embargo, esto provocaría un mayor número de accesos remotos. El hint entonces, evita que se realice esta desintegración.

Ejemplo

```
select t1.x, v.avg_y
from t1, (
  select /*+no_merge*/ x, avg(y) as avg_y
  from t2 group by x
) v,
where t1.x = v.x and t1.y = 1;
```

- De forma equivalente, el hint se puede ubicar justo después de la cláusula select, pero ahora dentro de la collocated inline view.
- Debido a que se encuentra dentro de la subconsulta, el hint no requiere parámetro.

5.5.5.2. Driving site

- El hint `driving_site` permite elegir a alguno de los nodos como el sitio que se encargará de dirigir u organizar la ejecución de una sentencia distribuida, la consulta inicia su ejecución en el sitio seleccionado, y por lo tanto, los resultados estarán disponibles en dicho sitio.

Ejemplo

```
select /*+driving_site(dept)*/ *
from emp, dept@remote.com
where emp.deptno = dept.deptno;
```

- En este ejemplo, la ejecución de la sentencia será iniciada o administrada por el sitio donde se encuentra la tabla dept.

5.5.6. Análisis del plan de ejecución.

- Como se mencionó en temas anteriores, el análisis del plan de ejecución ofrece retroalimentación muy importante para verificar el costo aproximado del procesamiento de una consulta distribuida.
- Adicional a los conceptos vistos anteriormente, un plan de ejecución de una sentencia distribuida define las operaciones remotas que se aplicarán en otros nodos:

```
Plan Table
-------------------------------------------------------------------------------
| Operation                 |  Name    |  Rows | Bytes|  Cost  | Pstart| Pstop |
-------------------------------------------------------------------------------
| SELECT STATEMENT          |          |       |      |        |       |       |
|  NESTED LOOPS             |          |       |      |        |       |       |
|   VIEW                    |          |       |      |        |       |       |
|    REMOTE                 |          |       |      |        |       |       |
|   TABLE ACCESS BY INDEX RO|DEPT      |       |      |        |       |       |
|    INDEX UNIQUE SCAN      |PK_DEPT   |       |      |        |       |       |
-------------------------------------------------------------------------------
```

- Observar en el plan anterior la sección `REMOTE`.
- En el ejemplo, la tabla outer del nested loop corresponde con el resultado que se obtiene de ejecutar una sentencia SQL en un sitio remoto.
- Para visualizar la consulta que se ejecuta en el sitio remoto se emplea la siguiente instrucción:

```
select other
from plan_table
where operation = 'REMOTE';
```

- Ejecutar la siguiente instrucción si la visualización del plan no es la adecuada: `set long 9999999`

5.5.7. Ejemplos de optimización

Los siguientes ejemplos muestran la aplicación de los conceptos vistos hasta el momento haciendo uso del caso de estudio de control de agencias de autos.

5.5.7.1. Ejemplo visualización de estadísticas.

A. Generar una consulta SQL que muestre el status actual de las estadísticas del esquema control de agencias. Ejecutar la sentencia en los 4 nodos para confirmar resultados.

```
select table_name,stale_stats
from dba_tab_statistics
where owner ='CONTROL_AGENCIA';
```

jrcbd_s1

	TABLE_NAME	STALE_STATS
1	F_JRC_AUTO_1	NO
2	F_JRC_AUTO_3	NO
3	F_JRC_AGENCIA_1	NO
4	F_JRC_AUTO_PARTICULAR_1	NO

jrcbd_s2

	TABLE_NAME	STALE_STATS
1	F_JRC_AGENCIA_2	NO
2	F_JRC_AUTO_4	NO
3	F_JRC_AUTO_PARTICULAR_2	NO

arcbd_s1

	TABLE_NAME	STALE_STATS
1	F_ARC_CLIENTE_1	NO
2	F_ARC_CLIENTE_3	NO

arcbd_s2

	TABLE_NAME	STALE_STATS
1	F_ARC_AUTO_2	NO
2	F_ARC_CLIENTE_2	NO
3	F_ARC_AUTO_CARGA_1	NO

B. Crear un programa PL/SQL que realice la recolección de estadísticas de los 4 nodos para las tablas del control de agencias.

```
prompt Recolectando estadisticas
begin
  dbms_stats.gather_table_stats (
      ownname => 'CONTROL_AGENCIA',
      degree  => 3
  );
end;
/
```

C. Generar una consulta que muestre el nombre de cada una de las tablas y las siguientes estadísticas: número de registros, número de bloques, bloques vacíos, longitud promedio del registro, fecha del último análisis, fecha actual. Ejecutar la consulta en cada uno de los 4 nodos.

```
select table_name,num_rows,blocks,empty_blocks,avg_row_len,
 to_char(last_analyzed,'dd/mm/yyyy hh24:mi:ss') last_analyzed,
 to_char(sysdate,'dd/mm/yyyy hh24:mi:ss')  today
from dba_tab_statistics
where owner ='CONTROL_AGENCIA';
```

	TABLE_NAME	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_ROW_LEN	LAST_ANALYZED	TODAY
1	F_JRC_AUTO_3	1000	13	0	55	23/04/2020 10:27:27	27/04/2020 20:22:41
2	F_JRC_AUTO_PARTICULAR_1	1000	5	0	12	23/04/2020 10:27:27	27/04/2020 20:22:41
3	F_JRC_AUTO_1	3000	20	0	41	23/04/2020 10:27:27	27/04/2020 20:22:41
4	F_JRC_AGENCIA_1	500	5	0	23	23/04/2020 10:27:27	27/04/2020 20:22:41

5.5.7.2. Uso de collocated Inline views, hint NO_MERGE.

A. Para realizar la siguiente consulta SQL, implementar transparencia de localización para las tablas globales `auto` y `auto_carga`. Suponer que la consulta se va a ejecutar en el sitio `jrcbd_s2`.

Para efectos del ejercicio, solo se implementa transparencia de localización para las tablas y el sitio indicado. Esto se realiza a través de la definición de ligas y posteriormente la creación de sinónimos:

```
prompt conectando en jrcbd_s1
connect control_agencia/jorge@jrcbd_s1

--ligas
create database link jrcbd_s2.fi.unam using 'JRCBD_S2';
create database link arcbd_s1.fi.unam using 'ARCBD_S1';
create database link arcbd_s2.fi.unam using 'ARCBD_S2';

--sinonimos para auto
create or replace synonym auto_1 for f_jrc_auto_1;
create or replace synonym auto_3 for f_jrc_auto_3;
create or replace synonym auto_4 for f_jrc_auto_4@jrcbd_s2;
create or replace synonym auto_2 for f_arc_auto_2@arcbd_s2;

--sinonimos para auto_carga  (solo 1)
create or replace synonym auto_carga_1 for f_arc_auto_carga_1@arcbd_s2;
```

- Observar la creación de 3 ligas que permitirán el acceso a los otros 3 nodos. Se asume que el usuario `control_agencia` existe en todos los nodos con el mismo password.

- Observar en la definición de los sinónimos que los fragmentos `f_jrc_auto_1` y `f_jrc_auto_3` no hacen uso de las ligas ya que se encuentran de forma local en el sitio `jrcbd_s1`.
- Notar que solo crea un sinónimo para la tabla global `auto_carga` debido a que solo existe un único fragmento.
- La misma estrategia puede aplicarse a los demás sitios considerando a los objetos remotos y locales. Por ejemplo, para `jrcbd_s2` se tiene la siguiente definición:

```
Prompt conectando en jrcbd_s2
connect control_agencia/jorge@jrcbd_s2

create database link jrcbd_s1.fi.unam using 'JRCBD_S1';
create database link arcbd_s1.fi.unam using 'ARCBD_S1';
create database link arcbd_s2.fi.unam using 'ARCBD_S2';

--sinonimos
create or replace synonym auto_1 for f_jrc_auto_1@jrcbd_s1;
create or replace synonym auto_3 for f_jrc_auto_3@jrcbd_s1;
create or replace synonym auto_4 for f_jrc_auto_4;
create or replace synonym auto_2 for f_arc_auto_2@arcbd_s2;

create or replace synonym agencia_1 for f_jrc_agencia_1@jrcbd_s1;
create or replace synonym agencia_2 for f_jrc_agencia_2;

create or replace synonym cliente_1 for f_arc_cliente_1@arcbd_s1;
create or replace synonym cliente_2 for f_arc_cliente_2@arcbd_s2;
create or replace synonym cliente_3 for f_arc_cliente_3@arcbd_s1;

create or replace synonym auto_particular_1 for
  f_jrc_auto_particular_1@jrcbd_s1;
create or replace synonym auto_particular_2 for f_jrc_auto_particular_2;

create or replace synonym auto_carga_1 for f_arc_auto_carga_1@arcbd_s2;
```

A partir de este punto se asume que en los 4 nodos se ha implementado transparencia de localización.

B. Considerando la existencia de transparencia de localización a través del uso de sinónimos con el nombre `<nombre_tabla_global>_n`, generar una consulta que muestre el identificador del auto, marca, modelo, peso máximo, volumen y precio de los autos de carga ubicados en `arcbd_s2`. Considerar que la consulta se lanza desde `jrcbd_s1`. Incluir la instrucción para generar el plan de ejecución. No emplear Collocated Inline views.

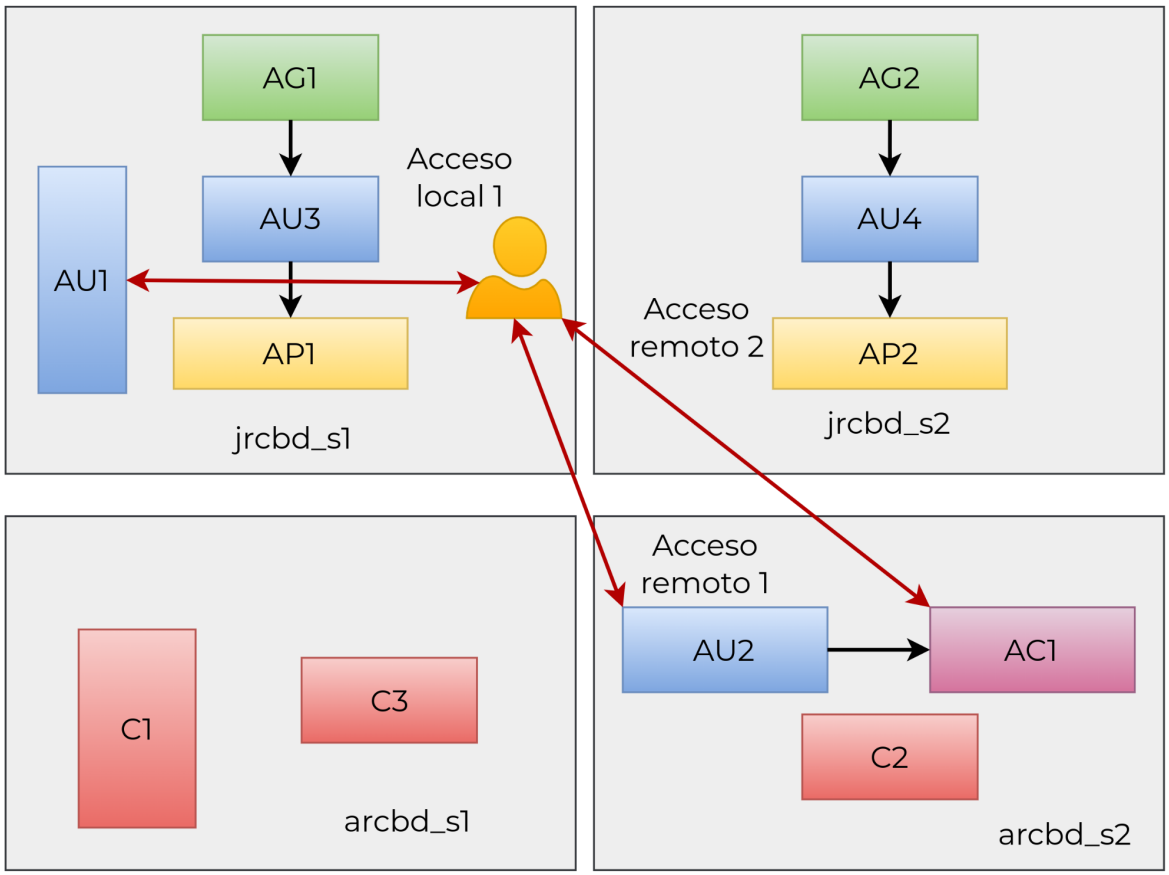
```
explain plan
set statement_id ='s1'
for
select a2.auto_id,a2.marca,a2.modelo,ac1.peso_maximo,ac1.volumen,a1.precio
from auto_2 a2, auto_carga_1 ac1, auto_1 a1
where a2.auto_id = ac1.auto_id
and a2.auto_id = a1.auto_id;
```

C. Mostrar el plan de ejecución.

```
select plan_table_output
from table(dbms_xplan.display('PLAN_TABLE','s1','TYPICAL'));
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 1138025008

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                     |  1000 | 79000 |    14   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN          |                     |  1000 | 79000 |    14   (0)| 00:00:01 |        |      |
|   2 |   REMOTE            | F_ARC_AUTO_CARGA_1  |  1000 | 14000 |     3   (0)| 00:00:01 | ARCBD~ | R->S |
|*  3 |   HASH JOIN         |                     |  1000 | 65000 |    11   (0)| 00:00:01 |        |      |
|   4 |    REMOTE           | F_ARC_AUTO_2        |  1000 | 55000 |     4   (0)| 00:00:01 | ARCBD~ | R->S |
|   5 |    TABLE ACCESS FULL| F_JRC_AUTO_1        |  3000 | 30000 |     7   (0)| 00:00:01 |        |      |
-----------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A2"."AUTO_ID"="AC1"."AUTO_ID")
   3 - access("A2"."AUTO_ID"="A1"."AUTO_ID")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   2 - SELECT "AUTO_ID","PESO_MAXIMO","VOLUMEN" FROM "F_ARC_AUTO_CARGA_1" "AC1" (accessing
       'ARCBD_S2.FI.UNAM' )

   4 - SELECT "AUTO_ID","MARCA","MODELO" FROM "F_ARC_AUTO_2" "A2" (accessing 'ARCBD_S2.FI.UNAM' )
```

- Notar que se realizan 2 accesos remotos. El primero para acceder a `auto_carga_1` y el segundo para acceder a `auto_1`. Ambos fragmentos se encuentran en el mismo nodo remoto `arcbd_s2`. Estas 2 acciones se podrían evitar empleando collocated inline views.
- Notar al final del plan, las actividades 2 y 4 muestran las consultas remotas que serán ejecutadas en `arcbd_s2`.



D. Reescribir la sentencia anterior empleando Collocated Inline views.

```
explain plan
set statement_id ='s2' for
select r.auto_id,r.marca,r.modelo,r.peso_maximo,r.volumen,a1.precio
from (
  select a2.auto_id,a2.marca,a2.modelo,ac1.peso_maximo,ac1.volumen
  from auto_2 a2, auto_carga_1 ac1
  where a2.auto_id = ac1.auto_id
) r , auto_1 a1
where a1.auto_id = r.auto_id;
```

E. Mostrar nuevamente el plan de ejecución.

```
select plan_table_output
from table(dbms_xplan.display('PLAN_TABLE','s2','TYPICAL'));
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 1138025008

-----------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                     |  1000 | 79000 |    14   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN          |                     |  1000 | 79000 |    14   (0)| 00:00:01 |        |      |
|   2 |   REMOTE            | F_ARC_AUTO_CARGA_1  |  1000 | 14000 |     3   (0)| 00:00:01 | ARCBD~ | R->S |
|*  3 |   HASH JOIN         |                     |  1000 | 65000 |    11   (0)| 00:00:01 |        |      |
|   4 |    REMOTE           | F_ARC_AUTO_2        |  1000 | 55000 |     4   (0)| 00:00:01 | ARCBD~ | R->S |
|   5 |    TABLE ACCESS FULL| F_JRC_AUTO_1        |  3000 | 30000 |     7   (0)| 00:00:01 |        |      |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A2"."AUTO_ID"="AC1"."AUTO_ID")
   3 - access("A1"."AUTO_ID"="A2"."AUTO_ID")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   2 - SELECT "AUTO_ID","PESO_MAXIMO","VOLUMEN" FROM "F_ARC_AUTO_CARGA_1" "AC1" (accessing
       'ARCBD_S2.FI.UNAM' )

   4 - SELECT "AUTO_ID","MARCA","MODELO" FROM "F_ARC_AUTO_2" "A2" (accessing 'ARCBD_S2.FI.UNAM' )
```

- Notar que el plan de ejecución no cambió a pesar de la reescritura de la sentencia SQL.
- Esto se debe a que el optimizador aplicó una operación merge para eliminar la subconsulta y así generar un solo plan de ejecución.
- Lo anterior es una operación típica y conveniente que se aplica en una base centralizada, pero no en este escenario en el que se pretende minimizar los accesos remotos.
- Para solucionar lo anterior, se hace uso del hint no_merge.
- El uso de este hint le indica al optimizador que no fusione o elimine la subconsulta ya que esta contiene a los fragmentos que se encuentran en un mismo nodo.

F. Actualizar la sentencia anterior para hacer uso del hint `no_merge` y mostrar nuevamente el plan

```
explain plan set statement_id ='s3' for

select /*+no_merge(r)*/
r.auto_id,r.marca,r.modelo,r.peso_maximo,r.volumen,a1.precio
from (
  select a2.auto_id,a2.marca,a2.modelo,ac1.peso_maximo,ac1.volumen
  from auto_2 a2, auto_carga_1 ac1
  where a2.auto_id = ac1.auto_id
) r , auto_1 a1
where a1.auto_id = r.auto_id;
```

```
select plan_table_output
from table(dbms_xplan.display('PLAN_TABLE','s3','TYPICAL'));
```

- Notar que el hint recibe como parámetro al alias ‘r’ que representa a la collocated inline view que se desea conservar.
- De forma equivalente, el hint se puede ubicar en la collocated inline view sin el argumento ‘r’.

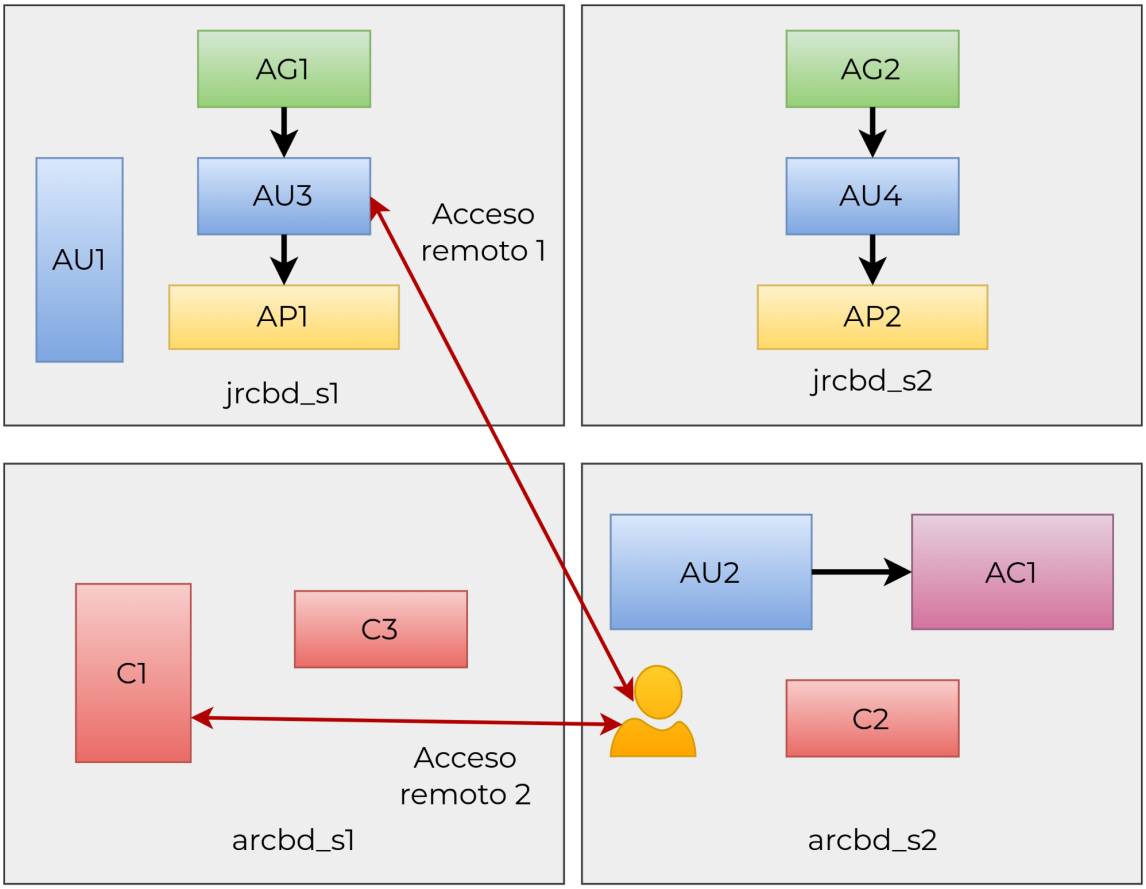
```
select r.auto_id,r.marca,r.modelo,r.peso_maximo,r.volumen,a1.precio
from (
  select /*+no_merge*/
    a2.auto_id,a2.marca,a2.modelo,ac1.peso_maximo,ac1.volumen
  from auto_2 a2, auto_carga_1 ac1
  where a2.auto_id = ac1.auto_id
) r , auto_1 a1
where a1.auto_id = r.auto_id;
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 3872297743

-------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name         | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |              |  1000 | 93000 |    14   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN          |              |  1000 | 93000 |    14   (0)| 00:00:01 |        |      |
|   2 |   TABLE ACCESS FULL| F_JRC_AUTO_1 |  3000 | 30000 |     7   (0)| 00:00:01 |        |      |
|   3 |   VIEW              |              |  1000 | 83000 |     7   (0)| 00:00:01 |        |      |
|   4 |    REMOTE           |              |       |       |            |          | ARCBD~ | R->S |
-------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A1"."AUTO_ID"="R"."AUTO_ID")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   4 - EXPLAIN PLAN SET STATEMENT_ID='s3' INTO PLAN_TABLE@! FOR SELECT
       "A2"."AUTO_ID","A2"."MARCA","A2"."MODELO","A1"."PESO_MAXIMO","A1"."VOLUMEN" FROM
       "F_ARC_AUTO_2" "A2","F_ARC_AUTO_CARGA_1" "A1" WHERE "A2"."AUTO_ID"="A1"."AUTO_ID"
       (accessing 'ARCBD_S2.FI.UNAM' )
```

- Observar que se ha eliminado uno de los accesos remotos al sitio `arcbd_s2`.
- En el detalle de la actividad 4, notar que envía la sentencia SQL conteniendo a todos los fragmentos que se encuentran en el sitio remoto.

5.5.7.3. Uso del hint DRIVING_SITE

A. Generar una consulta SQL que muestre cliente_id, num_tarjeta, auto_id, modelo, marca para todos los autos ubicados en el sitio `jrcbd_s1`. Mostrar el plan de ejecución asumiendo que la consulta se lanza en el sitio `arcbd_s2` como se muestra en la siguiente imagen



Plan de ejecución

```
explain plan
set statement_id 's4' for
select c1.cliente_id,c1.num_tarjeta,a3.auto_id,a3.modelo,a3.marca
from cliente_1 c1, auto_3 a3
where c1.cliente_id = a3.cliente_id;
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 2023380597

-------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name             | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                  |  1000 | 75000 |     8   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN         |                  |  1000 | 75000 |     8   (0)| 00:00:01 |        |      |
|   2 |   REMOTE           | F_JRC_AUTO_3     |  1000 | 55000 |     4   (0)| 00:00:01 | JRCBD~ | R->S |
|   3 |   REMOTE           | F_ARC_CLIENTE_1  |  3000 | 60000 |     4   (0)| 00:00:01 | ARCBD~ | R->S |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("C1"."CLIENTE_ID"="A3"."CLIENTE_ID")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   2 - SELECT "AUTO_ID","MARCA","MODELO","CLIENTE_ID" FROM "F_JRC_AUTO_3" "A3" (accessing
       'JRCBD_S1.FI.UNAM' )

   3 - SELECT "CLIENTE_ID","NUM_TARJETA" FROM "F_ARC_CLIENTE_1" "C1" (accessing
       'ARCBD_S1.FI.UNAM' )
```

- Observar los 2 accesos remotos que se realizan debido a que el Hash Join se realiza en el driving site, representado por `arcbd_s2`.
- Haciendo uso del hint `driving_site` se pueden tener los 2 siguientes escenarios:
  - Seleccionar a `jrcbd_s1` como driving site. Notar que de este sitio se obtienen 1000 registros.
  - Seleccionar a `arcbd_s1` como driving site. Notar que de este sitio se obtienen 3000 registros.

B. Generar el plan de ejecución considerando a `jrcbd_s1` como driving site.

```
explain plan
set statement_id 's5' for
select /*+driving_site(a3)*/ c1.cliente_id,c1.num_tarjeta,a3.auto_id,
  a3.modelo,a3.marca
from cliente_1 c1, auto_3 a3
where c1.cliente_id = a3.cliente_id;
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 1896038782

-------------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name             | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT REMOTE|                  |  1000 | 44000 |     9   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN              |                  |  1000 | 44000 |     9   (0)| 00:00:01 |        |      |
|   2 |   TABLE ACCESS FULL     | F_JRC_AUTO_3     |  1000 | 24000 |     5   (0)| 00:00:01 | JRCBD~ |      |
|   3 |   REMOTE                | F_ARC_CLIENTE_1  |  3000 | 60000 |     4   (0)| 00:00:01 | ARCBD~ | R->S |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A2"."CLIENTE_ID"="A1"."CLIENTE_ID")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   3 - SELECT "CLIENTE_ID","NUM_TARJETA" FROM "F_ARC_CLIENTE_1"@ARCBD_S1.FI.UNAM "A2" (accessing
       'ARCBD_S1.FI.UNAM' )


Note
-----
   - fully remote statement
```

C. Generar el plan de ejecución considerando a `arcbd_s1` como driving site.

```
explain plan
set statement_id 's6' for
select /*+driving_site(c1)*/ c1.cliente_id,c1.num_tarjeta,a3.auto_id,
  a3.modelo,a3.marca
from cliente_1 c1, auto_3 a3
where c1.cliente_id = a3.cliente_id;
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 3525960917

-------------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name             | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT REMOTE|                  |  1000 | 75000 |     9   (0)| 00:00:01 |        |
|*  1 |  HASH JOIN              |                  |  1000 | 75000 |     9   (0)| 00:00:01 |        |
|   2 |   REMOTE                | F_JRC_AUTO_3     |  1000 | 55000 |     4   (0)| 00:00:01 | JRCBD~ | R->S
|   3 |   TABLE ACCESS FULL     | F_ARC_CLIENTE_1  |  3000 | 60000 |     5   (0)| 00:00:01 | ARCBD~ |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A2"."CLIENTE_ID"="A1"."CLIENTE_ID")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   2 - SELECT "AUTO_ID","MARCA","MODELO","CLIENTE_ID" FROM "F_JRC_AUTO_3"@JRCBD_S1.FI.UNAM "A1"
       (accessing 'JRCBD_S1.FI.UNAM' )
```

5.5.7.4. Ejemplo: Consultas con transparencia de distribución.

A. Generar la sentencia SQL correspondiente que defina a la vista auto la cual permitirá implementar transparencia de distribución. Considerar la existencia de los sinónimos creados en ejercicios anteriores. Por simplicidad, para el campo foto regresar un blob vacío.

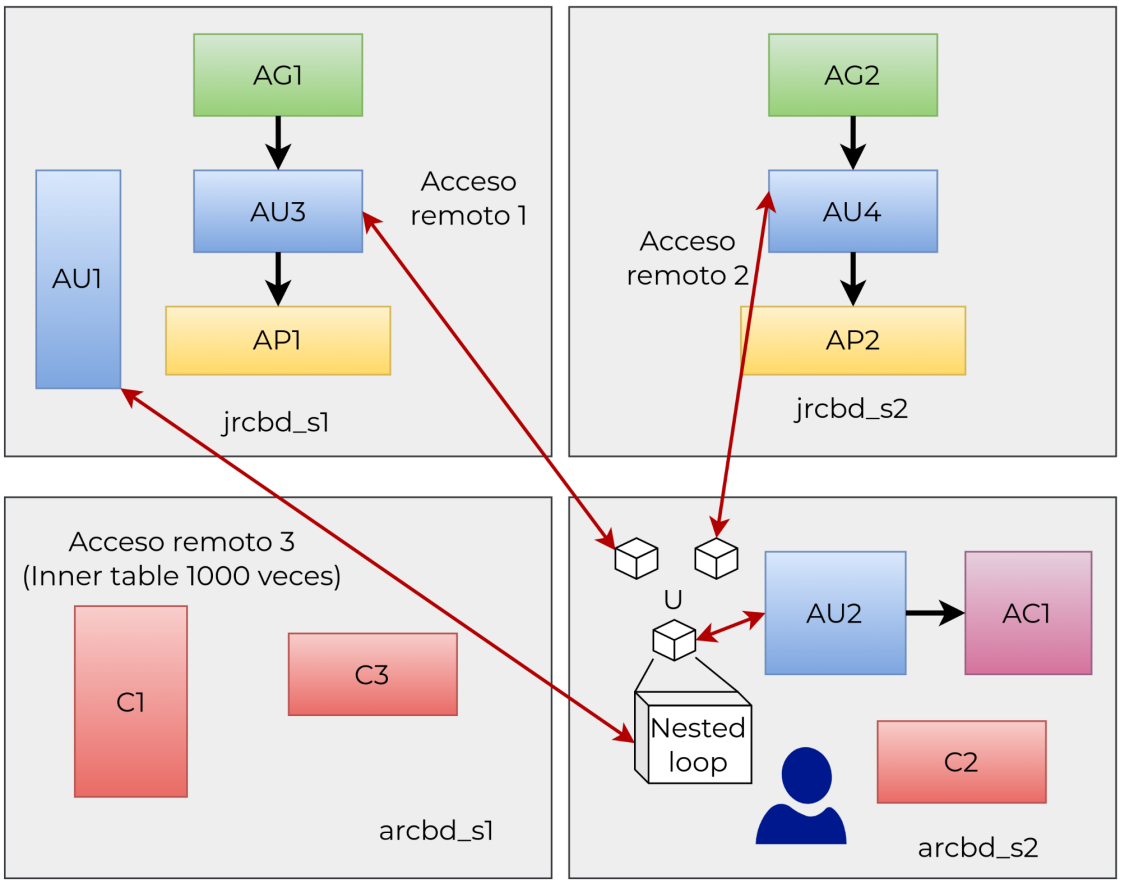
```
create or replace view  auto as
 select c1.auto_id,c1.precio, empty_blob() as foto,
   q.marca,q.modelo,q.anio,q.num_serie,q.tipo,
   q.descuento,q.agencia_id,q.cliente_id
 from auto_1 c1
 join (
   select cliente_id,marca,modelo,anio,num_serie,tipo,
     descuento,agencia_id,cliente_id
   from auto_2
   union all
   select cliente_id,marca,modelo,anio,num_serie,tipo,
     descuento,agencia_id,cliente_id
   from auto_3
   union all
   select cliente_id,marca,modelo,anio,num_serie,tipo,
     descuento,agencia_id,cliente_id
   from auto_4
 ) q on c1.auto_id = q.auto_id;
```

B. Empleando transparencia de distribución, generar una consulta SQL y su correspondiente plan de ejecución para mostrar el id del auto, su marca, modelo y año para todos los autos tipo C. Suponer que la consulta se lanza en el sitio `arcbd_s2`.

```
explain plan
set statement_id 's6' for
select auto_id,marca,modelo,anio
from auto
where tipo ='C';

select plan_table_output
from table(dbms_xplan.display('PLAN_TABLE','s6','TYPICAL'));
```

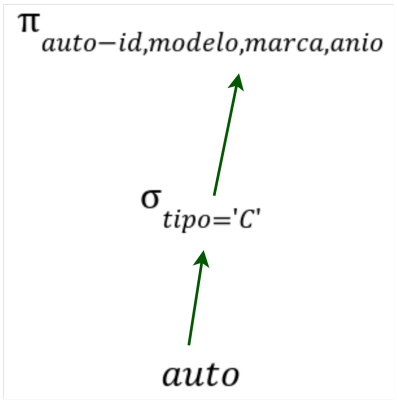
```
PLAN_TABLE_OUTPUT
Plan hash value: 858543363

-------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name          | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |               |  1002 | 86172 |    14   (8)| 00:00:01 |        |      |
|   1 |  NESTED LOOPS        |               |  1002 | 86172 |    14   (8)| 00:00:01 |        |      |
|   2 |   VIEW               |               |  1002 | 73146 |    13   (0)| 00:00:01 |        |      |
|   3 |    UNION-ALL         |               |       |       |            |          |        |      |
|*  4 |     TABLE ACCESS FULL| F_ARC_AUTO_2  |  1000 | 26000 |     5   (0)| 00:00:01 |        |      |
|   5 |     REMOTE           | F_JRC_AUTO_3  |     1 |    55 |     4   (0)| 00:00:01 | JRCBD~ | R->S |
|   6 |     REMOTE           | F_JRC_AUTO_4  |     1 |    55 |     4   (0)| 00:00:01 | JRCBD~ | R->S |
|   7 |   REMOTE             | F_JRC_AUTO_1  |     1 |    13 |     0   (0)| 00:00:01 | JRCBD~ | R->S |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter("AUTO_2"."TIPO"='C')

Remote SQL Information (identified by operation id):
----------------------------------------------------

   5 - SELECT "AUTO_ID","MARCA","MODELO","ANIO","TIPO" FROM "F_JRC_AUTO_3" "AUTO_3" WHERE
       "TIPO"='C' (accessing 'JRCBD_S1.FI.UNAM' )

   6 - SELECT "AUTO_ID","MARCA","MODELO","ANIO","TIPO" FROM "F_JRC_AUTO_4" "AUTO_4" WHERE
       "TIPO"='C' (accessing 'JRCBD_S2.FI.UNAM' )

   7 - SELECT "AUTO_ID" FROM "F_JRC_AUTO_1" "C1" WHERE "AUTO_ID"=:1 (accessing
       'JRCBD_S1.FI.UNAM' )
```

- Notar que el plan generado no es óptimo.
- El plan refleja justamente las operaciones que se deben realizar para reconstruir la tabla global:
    - Se realizan 2 accesos remotos para traer los datos de los fragmentos `auto_3` y `auto_4`. Para ambos casos, observar que se estima obtener un solo registro. Esto se debe a que en realidad, no existen autos tipo C en estos nodos.
    - Posteriormente se realiza la unión con el fragmento local `auto_2`
    - El resultado de esta triple unión es considerada como la tabla outer de un nested loop para realizar el join con `auto_1`, se estiman 1000 registros.
    - Notar que el optimizador ha seleccionado a `auto_1` como tabla inner. Esto significa que se harán 1000 iteraciones y en cada una de ellas se realizará un acceso remoto al sitio `jrcbd_s1`. En cada acceso remoto se estima obtener 1 registro.
- Esto se puede visualizar en el siguiente diagrama.

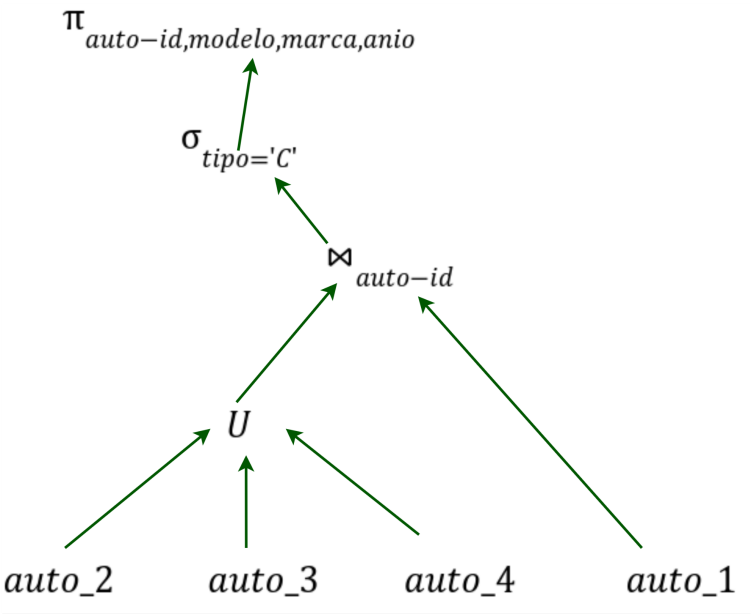


C. Analizar el plan anterior y proponer una solución que represente la mejor opción en cuanto a desempeño sacrificando transparencia. Realizar para ello una optimización manual a través del cálculo de la generación de la consulta algebraica, consulta localizada y finalmente, la consulta reducida.

- Consulta algebraica:



- Consulta localizada



- Consulta reducida.

$$R = \pi_{auto-id,modelo,marca,anio}\left(\sigma_{tipo='C'}\left(\left(a_2\ U\ a_3\ U\ a_4\right)\bowtie a_1\right)\right)$$

- Observar que al distribuir la selección con los fragmentos, es posible eliminar a a3 y a4 ya que en ellos no existen autos tipo C debido al esquema de fragmentación. Por lo tanto:

$$R = \pi_{auto-id,modelo,marca,anio}\left(\sigma_{tipo='C'}\left(a_2\right)\bowtie a_1\right)$$

- Nuevamente, al distribuir la selección, es factible eliminar ahora al fragmento a1 ya que al ser vertical puro, y no contener a las columnas de interés, este se puede suprimir. Por lo tanto:

$$R = \pi_{auto-id,modelo,marca,anio}\left(\sigma_{tipo='C'}\left(a_2\right)\right)$$

Notar que este fragmento se encuentra en el mismo sitio donde se lanzó la consulta. En SQL generará el siguiente plan de ejecución:

```
explain plan
set statement_id 's7' for
select auto_id,marca,modelo,anio
from auto_2
where tipo ='C';
```

```
select plan_table_output
from table(dbms_xplan.display('PLAN_TABLE','s7','TYPICAL'));
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 3599990263

-------------------------------------------------------------------------------------
| Id  | Operation          | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |               |  1000 | 26000 |     5   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| F_ARC_AUTO_2 |  1000 | 26000 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("TIPO"='C')
```

- En conclusión:
  - La optimización estática realizada permitió reducir los costos de la consulta de forma significativa.
  - En este ejemplo se sacrificó transparencia de distribución por transparencia de fragmentación, pero como beneficio se mejoró el desempeño ya que esta consulta en particular pudo ser resuelta totalmente de forma local, sin requerir accesos remotos.

5.5.8. Uso de semijoin

- Empleados para reducir el tiempo total de ejecución de un join en el que las fuentes de datos se ubican en nodos distintos.
- Esta técnica evita la transmisión de relaciones completas.
- En esta técnica el semi-join actúa como un reductor aplicado a una relación, de forma similar a una operación de selección.
- Un join entre 2 relaciones R y S puede ser calculado en términos de Semi-Joins empleando las siguientes reglas:

$R \bowtie_A S \Leftrightarrow \left(R \ltimes_A S\right) \bowtie_A S$ o de forma equivalente:

$\Leftrightarrow R \bowtie_A \left(S \ltimes_A R\right)$ o de forma equivalente:

$\Leftrightarrow \left(R \ltimes_A S\right) \bowtie_A \left(S \ltimes_A R\right)$

El semijoin es adecuado si el costo de producir y enviar datos para su cálculo a un sitio es menor al costo de enviar la relación completa.

5.5.8.1. Análisis de costos: Join vs Semi-join

- Se asume que `size(R) < size(S)`
- Considerar que la consulta se lanza en S2.

Sin semi join

1. En s2 se realiza un acceso remoto hacia s1 para obtener R y enviarla a s2.

$$R \rightarrow Sitio\ 2$$

2. En s2 se realiza el Join con S (sitio origen)

$$R \ltimes_A S$$

Con semi join

1. En el sitio 2 se realiza una primera consulta para enviar los datos mínimos de S para hacer un semi –join en Sitio 1. Típicamente se envía la PK de S. El semijoin siempre se realiza en el sitio remoto.

$$sitio\ 2:\ S' = \pi_A(S)$$

$$S' \rightarrow Sitio\ 1$$

2. En el sitio 1 se realiza el semi-join para detectar a los registros de R que harán correspondencia con S y se envían a S2.

$$sitio\ 1: R' = R \ltimes S'$$

$$R' \rightarrow Sitio\ 2$$

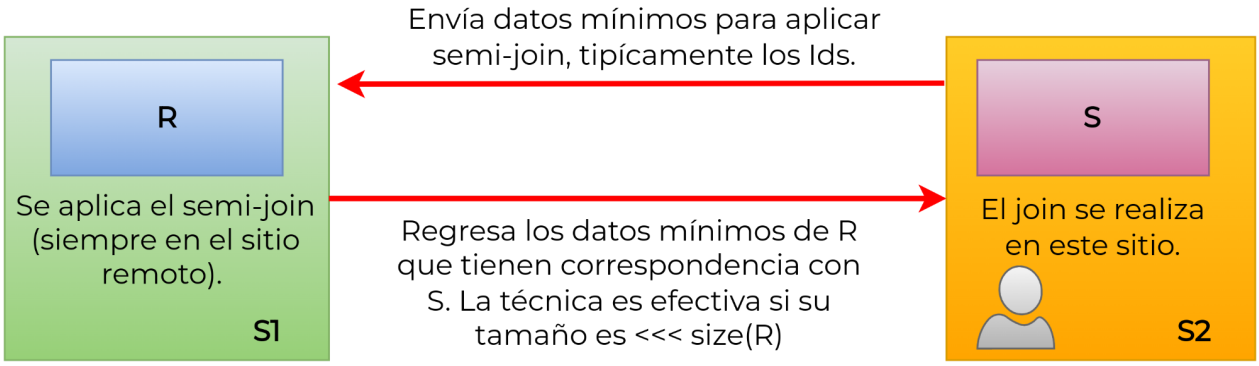
3. En el sitio 2 se realiza el join únicamente con los registros que tienen correspondencia, evitando así el descarte de registros.

$$sitio\ 2: R' \ltimes_A S$$

- Lo anterior implica que un semi-join es mejor si:

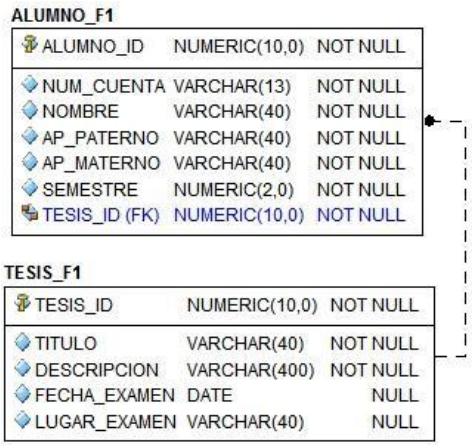
$$\left(size\left(\pi_A(S)\right) + size(R \ltimes S)\right) < size(R)$$

- El semi-join proporciona una mejor opción cuando pocos registros de R participan en el Join.
- Ejecutar el JOIN sin semijoin es mejor si casi todos los registros de R participan en el Join.



5.5.8.2. Ejemplo: Uso de Semi-join.

Considere los 2 siguientes fragmentos. `alumno_f1` se encuentra en el sitio s1 y `tesis_f1` se encuentra en s2. El manejador ha decidido emplear la técnica de semi-joins para obtener el número de cuenta y el título de la tesis para los alumnos del noveno semestre y cuyas tesis aún no tienen fecha de examen. Suponer que la consulta se lanza en S1.



A. Generar la sentencia SQL que obtendrá los datos mínimos a enviar a S2.
B. Suponer que S2 almacena los datos recibidos en una tabla temporal `Temp`. Genere la sentencia SQL que obtendría los datos mínimos a enviar de regreso a S1
C. Suponer que S1 almacena los datos recibidos en una tabla temporal `Temp`, Genere la sentencia SQL que S1 debe realizar para mostrar el resultado solicitado.

Solución

A. SQL que obtiene los datos mínimos a enviar.

```
select tesis_id
from alumno_f1
where semestre = 9;
```

B. SQL que se ejecuta en S2 al llegar los datos de S1 en Temp

Semi-join entre `temp` y `tesis_f1`. Esta operación permite filtrar registros de `tesis_f1`. Solo se envían los registros que se requieren para ejecutar la operación join en el sitio 1. Se envía el título de la tesis ya que requiere ser mostrado en la consulta final y se agrega la condición de la fecha de examen no asignada. Notar el uso de `exists` para hacer énfasis en el uso de un semi-join.

```
select te.tesis_id, te.titulo
from tesis_f1 te
where exists(
  select tesis_id
  from temp
  where tesis_id = te.tesis_id
)
and te.fecha_examen is null;
```

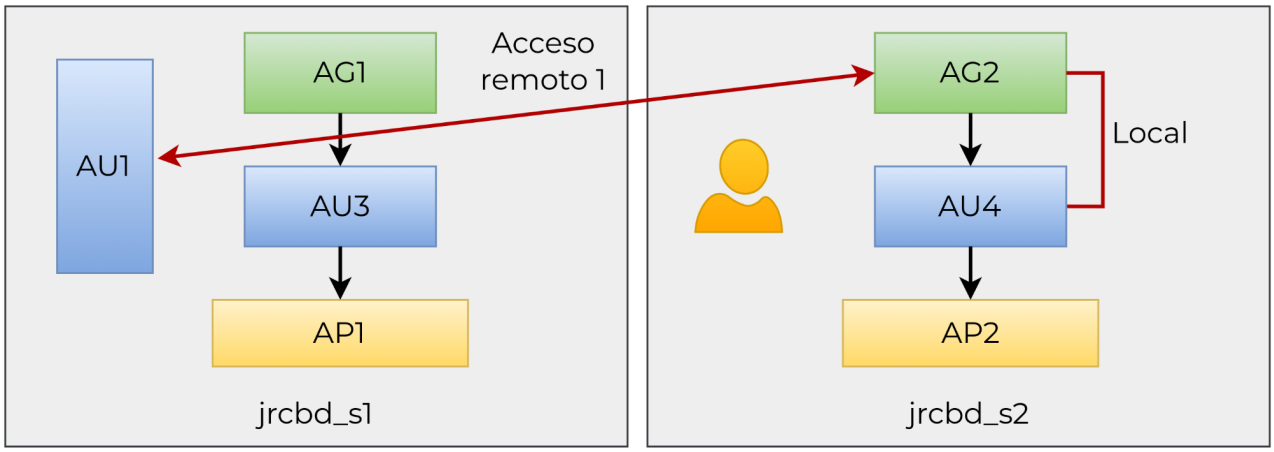
C. SQL que se ejecuta en S1 al recibir el resultado de T2

Join entre `temp` y `alumno_f1` para mostrar el resultado final:

```
select t.titulo, a.num_cuenta
from alumno a, temp t
where a.tesis_id = t.tesis_id;
```

5.5.8.3. Ejemplo - programación de una consulta con semijoin

Considerar los fragmentos `auto_4` y `agencia_2` ubicados en `jrcbd_s2` y el fragmento `auto_1` ubicado en `jrcbd_s1`. Se desea generar una sentencia SQL que obtenga auto_id, precio, marca, modelo de todos los autos cuyas claves de agencia inicie con 'W'. Considerar que la consulta se lanza en `jrcbd_s2`.



A. Generar una sentencia SQL, y plan de ejecución sin considerar el uso de semi –joins.

```
explain plan
set statement_id 's12' for
select a1.auto_id, a1.precio, a4.marca,a4.modelo
from auto_4 a4, agencia_2 a2, auto_1 a1
where  a4.agencia_id = a2.agencia_id
and   a4.auto_id = a1.auto_id
and a2.clave  like 'W%';

select plan_table_output
from table(dbms_xplan.display('PLAN_TABLE','s12','TYPICAL'));
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 1767603162

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation           | Name             | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                  |    51 |  3213 |    13   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN          |                  |    51 |  3213 |    13   (0)| 00:00:01 |        |      |
|*  2 |   HASH JOIN         |                  |    51 |  1887 |     8   (0)| 00:00:01 |        |      |
|*  3 |    TABLE ACCESS FULL| F_JRC_AGENCIA_2  |    22 |   286 |     3   (0)| 00:00:01 |        |      |
|   4 |    TABLE ACCESS FULL| F_JRC_AUTO_4     |  1000 | 24000 |     5   (0)| 00:00:01 |        |      |
|   5 |   REMOTE            | F_JRC_AUTO_1     |  3000 | 78000 |     5   (0)| 00:00:01 | JRCBD~ | R->S |
-----------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A4"."AUTO_ID"="A1"."AUTO_ID")
   2 - access("A4"."AGENCIA_ID"="A2"."AGENCIA_ID")
   3 - filter("A2"."CLAVE" LIKE 'W%')

Remote SQL Information (identified by operation id):
----------------------------------------------------

   5 - SELECT "AUTO_ID","PRECIO" FROM "F_JRC_AUTO_1" "A1" (accessing 'JRCBD_S1.FI.UNAM' )
```

- Notar que en este plan se realiza un join local para determinar los registros que cumplen con el predicado `clave like 'W%'` haciendo uso de un hash join.
- Posteriormente se realiza un acceso remoto que hace una consulta del 100% de los registros de `auto_1`. En este caso, 3000 registros.
- Una vez que los registros son transferidos al sitio origen `jrcbd_s2`, se realiza un segundo join del cual se puede observar que aproximadamente 51 registros tienen correspondencia con los 3000 registros de `auto_1`. ¿Qué pasa con los demás?, se descartan.
- Lo anterior provoca un uso inadecuado de los recursos de red. Se transmiten 3000 registros y solo 47 de ellos se muestran en el resultado.

B. Proponer una mejora al plan anterior haciendo uso del algoritmo de semi –joins.

Con base al algoritmo del uso de semi –joins, la estrategia estará formada de los siguientes pasos:

1. En `jrcbd_s2` generar una consulta que obtenga los datos mínimos a enviar a `jrcbd_s1`:

```
select a4.auto_id
from auto_4 a4, agencia_2 a2
where  a4.agencia_id = a2.agencia_id
and a2.clave  like 'W%'
```

2. Estos datos deben ser transmitidos hacia `jrcbd_s1` y ahí realizar el semi –join. Una forma de implementar este paso es generar la consulta desde el sitio origen `jrcbd_s2` pero indicarle al optimizador que la ejecución inicie en `jrcbd_s1`. Esto permitirá que en dicho sitio se soliciten los datos obtenidos en la sentencia del punto 1, y localmente se aplique el semi–join en `jrcbd_s1` tal como lo indica el algoritmo.

```
explain plan
set statement_id 's98' for

  select /*+driving_site(a1)*/ a1.auto_id,a1.precio
  from auto_1 a1
  where exists(
    select a4.auto_id
    from auto_4 a4, agencia_2 a2
    where  a4.agencia_id = a2.agencia_id
    and a2.clave  like 'W%'
    and a1.auto_id = a4.auto_id
  );
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 3839853780

-----------------------------------------------------------------------------------------------------------
| Id  | Operation               | Name         | Rows  | Bytes | Cost (%CPU)| Time     | Inst   |IN-OUT|
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT REMOTE|              |   147 |  3381 |    14   (0)| 00:00:01 |        |      |
|*  1 |  HASH JOIN RIGHT SEMI   |              |   147 |  3381 |    14   (0)| 00:00:01 |        |      |
|   2 |   VIEW                  | VW_SQ_1      |   147 |  1911 |     7   (0)| 00:00:01 | JRCBD~ |      |
|   3 |    REMOTE               |              |       |       |            |          |      ! | R->S |
|   4 |   TABLE ACCESS FULL     | F_JRC_AUTO_1 |  3000 | 30000 |     7   (0)| 00:00:01 | JRCBD~ |      |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A1"."AUTO_ID"="ITEM_0")

Remote SQL Information (identified by operation id):
----------------------------------------------------

   3 - EXPLAIN PLAN SET STATEMENT_ID='s98' INTO "PLAN_TABLE" FOR SELECT "A2"."AUTO_ID" FROM
       "CONTROL_AGENCIA"."F_JRC_AUTO_4" "A2","CONTROL_AGENCIA"."F_JRC_AGENCIA_2" "A1" WHERE
       "A2"."AGENCIA_ID"="A1"."AGENCIA_ID" AND "A1"."CLAVE" LIKE 'W%' (accessing '!' )
```

- Observar en este plan de ejecución que efectivamente la consulta se inicia en `jrcbd_s1`. Esto se logra con la ayuda del hint `/*+driving_site(a1)*/`
- En este sitio se hace una llamada remota a `jrcbd_s2` y justamente se obtienen los datos mínimos que se deben transmitir para poder ejecutar el semi –join. Con base al plan anterior, se transmiten aproximadamente 51 registros.
- Observar la operación 1 hash join right semi que hace referencia a la ejecución del semi –join empleando los datos que se transmitieron con la tabla local `auto_1`.
- Observar en la actividad 0, el resultado se transmite de regreso al sitio `jrcbd_s1`. Aquí se puede apreciar que se transmite una cantidad mucho menor respecto a los 3000 registros, reduciendo así considerablemente la cantidad de datos que viajan por la red.

3. Una vez que los datos llegan a `jrcbd_s2`, se almacenan en una tabla temporal.

```
create global temporary table t_semi_auto_1(
  auto_id number(10,0) constraint t_semi_auto_1_pk primary key,
  precio number(9,2)
) on commit delete rows;

insert into t_semi_auto_1
  select /*+driving_site(a1)*/ a1.auto_id,a1.precio
  from auto_1 a1
  where exists(
    select a4.auto_id
    from auto_4 a4, agencia_2 a2
    where  a4.agencia_id = a2.agencia_id
```

```
    and a2.clave  like 'W%'
    and a1.auto_id = a4.auto_id
  );
```

4. Finalmente, se realiza el join local entre `auto_4` y la tabla temporal que contiene los datos transmitidos de `jrcbd_s1`:

```
explain plan
set statement_id 's97' for
select a4.auto_id, t.precio, a4.marca,a4.modelo
from auto_4 a4, t_semi_auto_1 t
where a4.auto_id = t.auto_id;
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 3525493444

-------------------------------------------------------------------------------------------
| Id  | Operation          | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                |    40 |  1840 |     7   (0)| 00:00:01 |
|*  1 |  HASH JOIN         |                |    40 |  1840 |     7   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| T_SEMI_AUTO_1  |    40 |  1040 |     2   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| F_JRC_AUTO_4   |  1000 | 20000 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A4"."AUTO_ID"="T"."AUTO_ID")

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan
```

- De este plan se observa que solo viajaron 40 registros.
- En general, la cantidad de datos que viajan por la red se disminuyeron, pero con las siguientes implicaciones:
    - Menos datos, pero aumentó el número de accesos a la red de 1 a 2.
    - De la creación de la tabla temporal y la programación de estos pasos. La consulta no es posible resolverla en una sola instrucción.