INFS3200/INFS7907 Advanced Database System - Practical 1

Learning objectives:

- How to horizontally and vertically fragment relations during distributed database design; and compare the impact of different replication strategies (i.e. no, partial, and full replication) to distributed query processing.
- Apply semi-join to reduce data transmission over computer network.

Oracle XE 11g basics

1. From the Windows XP "Start" menu you can find "Oracle Database 11g Express Edition", click it to show Figure 1. Click on "Run SQL Command Line".

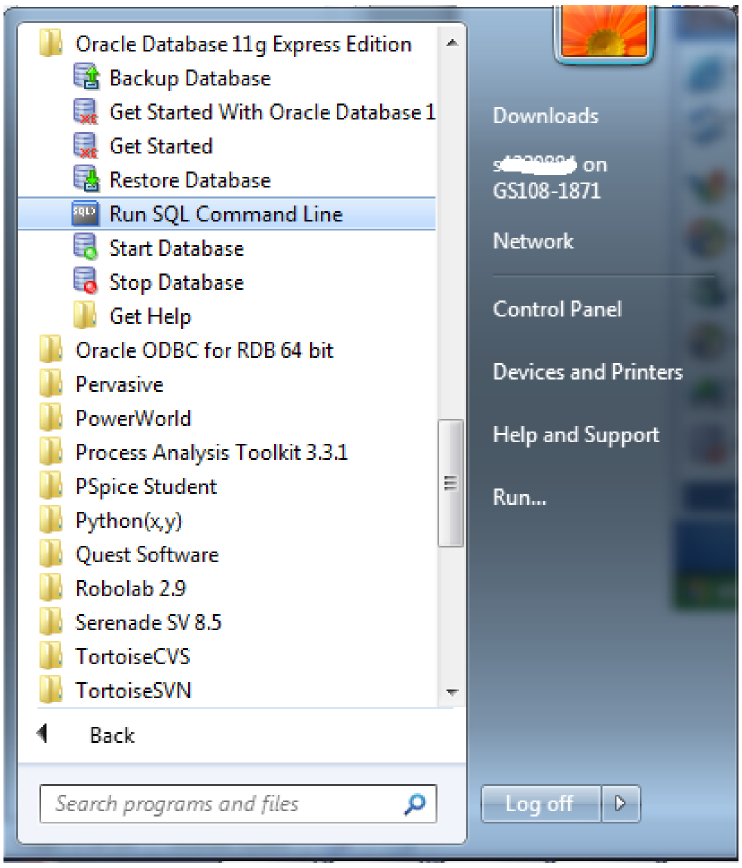


Figure 1

2. In SQL Command Line window, you can create a user "USER_HP_FULL" using SQL*Plus commands:

```
/* log in as "sys" with password "admin" */
conn sys/admin@127.0.0.1 as sysdba;

/* Create a user  named "USER_HP_FULL"  with password "w" */
CREATE USER USER_HP_FULL IDENTIFIED BY w ACCOUNT UNLOCK DEFAULT
TABLESPACE "USERS" TEMPORARY TABLESPACE "TEMP" PROFILE "DEFAULT";
```

```
/* Grant DBA privilege to "USER_HP_FULL" */
GRANT DBA TO USER_HP_FULL;

/* Check if "USER_HP_FULL" has been created */
SELECT * FROM ALL_USERS;
```

3. In the first two practicals, you need to save your data to an external SQL file each time when you logout, so that when you login next time you can import the external SQL file and continue to work on it. This can be done by using "Oracle SQL Developer" which is available from: http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html

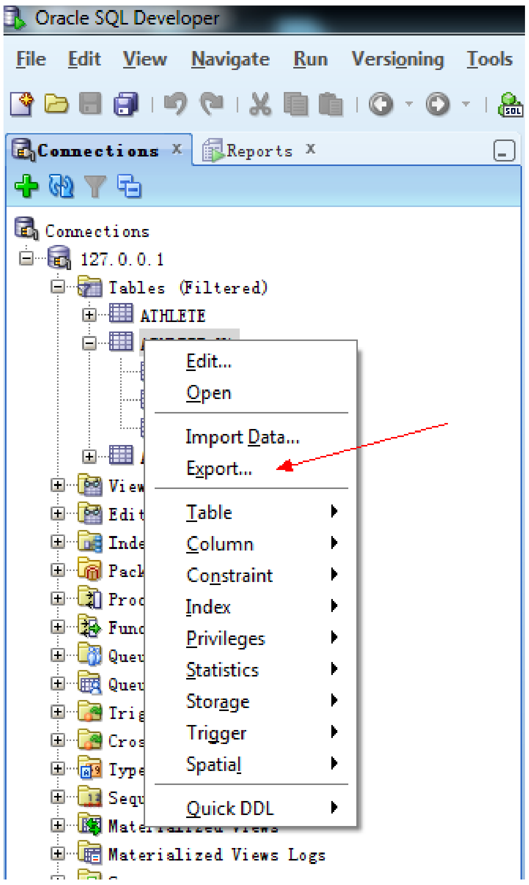
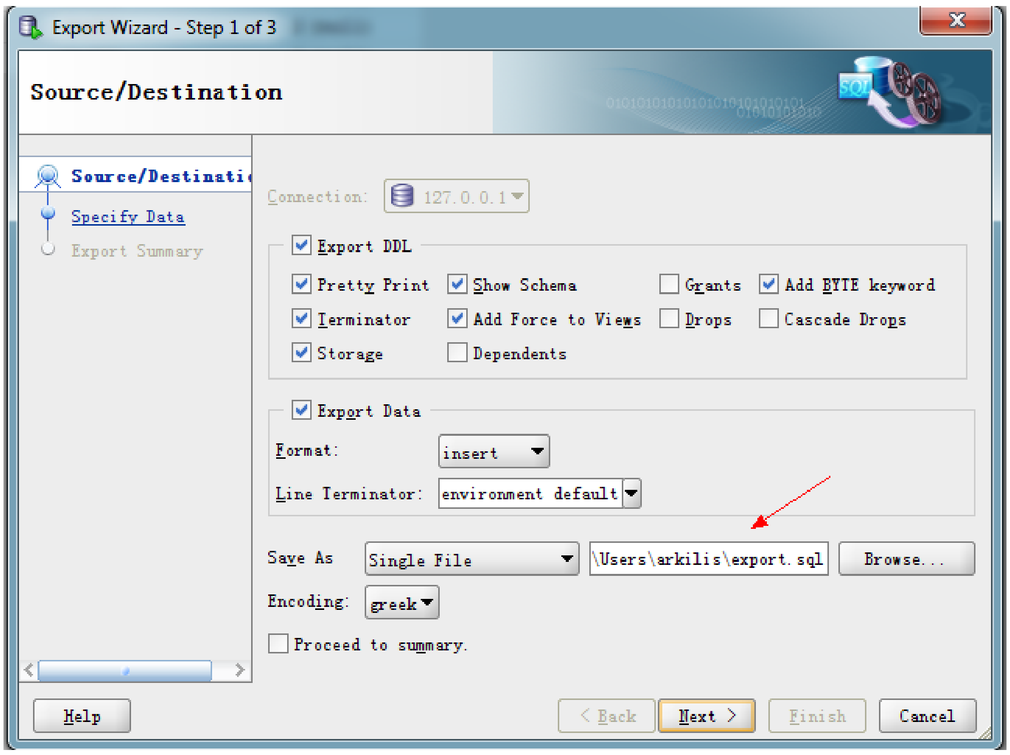


Figure 2

Once Oracle SQL Developer is installed, you need connect to your database in Oracle. Then, Figure 2 is shown. Right clicking a relation and select "Export" to show Figure 3, input the location to export this relation.


Export Wizard - Step 1 of 3

Source/Destination

Source/Destination
Specify Data
Export Summary

Connection: 127.0.0.1

[✓ Export DDL]

[✓ Pretty Print] [✓ Show Schema] [ ] Grants [✓ Add BYTE keyword]
[✓ Terminator] [✓ Add Force to Views] [ ] Drops [ ] Cascade Drops
[✓ Storage] [ ] Dependents

[✓ Export Data]

Format: insert
Line Terminator: environment default

Save As: Single File \Users\arkilis\export.sql Browse...
Encoding: greek
[ ] Proceed to summary.

Figure 3

Distributed Databases

In this practical, the computer network consists of four sites, each of which is simulated by a user account in Oracle XE. The data transferred among relations belonging to different user accounts correspond to the data transferred over computer network among different sites. You are asked to create three user accounts to simulate three sites in the computer network (refer to Oracle XE 11g basics – 2 above for how to create a new user account in Oracle XE). Given a global conceptual schema and do the following tasks:

Athlete[AthleteID, FName, LName, Gender, DOB, Sports, CountryCode]
Event[EventID, Name, Type, Year, SportsID]
Venue [VenueID, Location, Name, Year]
Country[CountryCode, CountryName]
Sports [SportsID, Name]
Result [EventID, Medal, Score]

Task One.
Distribution Design - Relation Fragmentation, Replication and Allocation

Horizontal Fragment	**Minterm and/or predicates**
Athlete1	1<= AthleteID < 7656
Athlete2	7657<= AthleteID < 17318
Athlete3	17319 <= AthleteID <= 24345

Strategy 1 – full replication Each fragment will be a relation located on every site in the computer network (i.e. each site has a copy of this fragment). You can create three sites (USER1_HP_FULL_StudentNumber, USER2_HP_FULL_ StudentNumber, USER3_HP_FULL_ StudentNumber). To load fragments into site USER1_HP_FULL_ StudentNumber, login Oracle XE as user “USER1_HP_FULL_ StudentNumber” and run all script files in folder “Prac_1/HP-FULL”. For other sites, repeat the same operations.

Strategy 2 – partial replication (reasonable) Each fragment will be a relation located on selected sites in the computer network (i.e. more than one site may have a copy of this fragment). You can create three sites (USER1_HP_RA_ StudentNumber, USER2_HP_RA_ StudentNumber, USER3_HP_RA_ StudentNumber). To load fragments into site USER1_HP_RA_ StudentNumber, login Oracle XE as user “USER1_HP_RA_ StudentNumber” and run all script files in folder “Prac_1/HP-RA”. For other sites, repeat the same operations.

Strategy 3 – no replication Each fragment will be a relation located on only one site in the computer network. You can create four sites (USER1_HP_NO_ StudentNumber, USER2_HP_NO_ StudentNumber, USER3_HP_NO_ StudentNumber). To load fragments into site USER1_HP_NO_ StudentNumber, login Oracle XE as user “USER1_HP_NO_ StudentNumber” and run all script files in folder “Prac_1/HP-NO”. For other sites, repeat the same operations.

(Note: you created different sites in each strategy. The purpose is to make the different replication strategies clear from each other. Here, we simulate horizontal fragmentation only, but the vertical/hybrid fragments can be processed in the exactly same way).

Question: at different replication strategies, what will happen when you update a single tuple at: site USER1_HP_FULL_999999 for Strategy 1; site USER1_HP_RA_99999999 for Strategy 2; and USER1_HP_NO_99999999 for Strategy 3

Task Two.
Distributed Query Processing – Semi-join

Useful Tools

1) Check query execution information - open SQL*Plus and login as SYSTEM.

   SQL> SET TIMING ON; /* set timing */

   SQL> SET AUTOTRACE ON STATISTICS; /* set autotrace */

2) Query Execution Statistics – go to Oracle Home Page by login as SYSTEM and go to SQL -> SQL Commands. Copy and paste your SQL command. After running the commands, click "Explain" as shown in figure 4, and you will see the query execution plan, indexing and other table information.

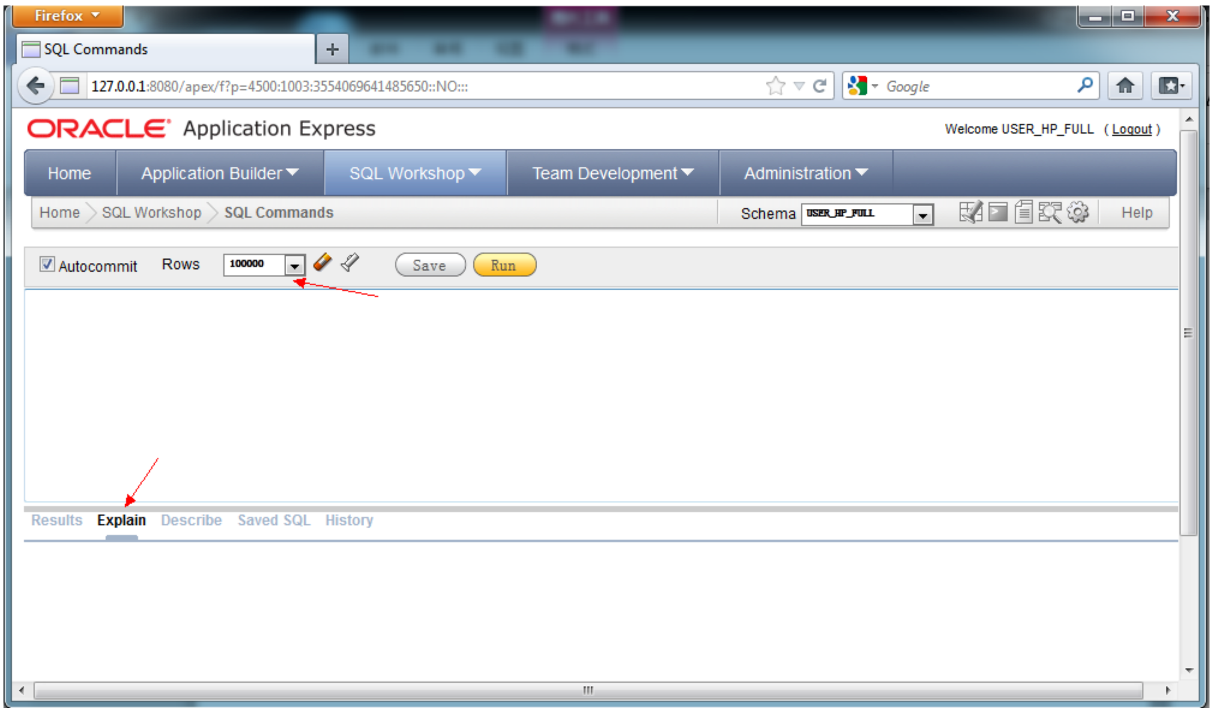


Figure 4

Semi-join vs. inner-join queries

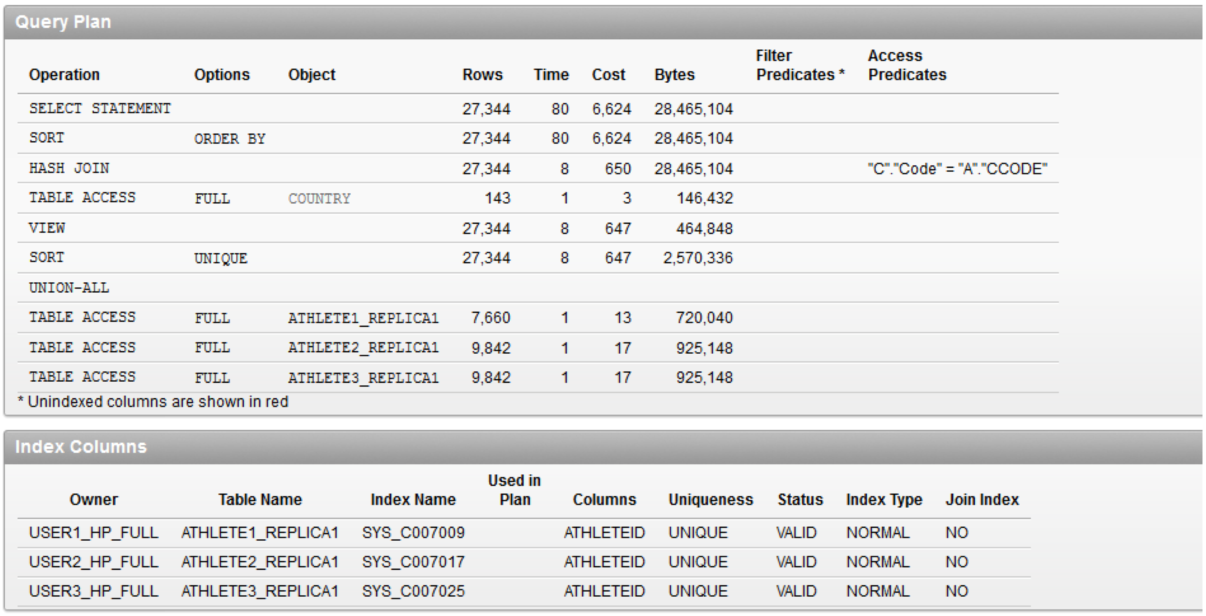
Given a user query defined over global conceptual schema

```
SELECT Countryname, Countrycode
From Athlete, Country
where athlete.ccode= Country.countrycode;
```

To process the same query in distributed database, semi-join can be applied to reduce data transmission over computer network. We suppose fragments are of full replication. The following SQL is an implementation of semi-join. Copy and paste in Oracle Home Page and run it.

```
SELECT C.COUNTRYNAME, C.CODE
FROM "USER_HP_FULL"."COUNTRY" C
WHERE EXISTS(
     SELECT 1 FROM
            (SELECT * FROM "USER1_HP_FULL"."ATHLETE1_REPLICA1"
             UNION SELECT * FROM "USER2_HP_FULL"."ATHLETE2_REPLICA1"
     UNION
     SELECT * FROM "USER3_HP_FULL"."ATHLETE3_REPLICA1") A
     WHERE A.CCODE= C.CODE)
ORDER BY C.COUNTRYNAME;
```

Click on "explain" and you will see the query plan (as shown in figure 4):



Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			27,344	80	6,624	28,465,104		
SORT	ORDER BY		27,344	80	6,624	28,465,104		
HASH JOIN			27,344	8	650	28,465,104		"C"."Code" = "A"."CCODE"
TABLE ACCESS	FULL	COUNTRY	143	1	3	146,432		
VIEW			27,344	8	647	464,848		
SORT	UNIQUE		27,344	8	647	2,570,336		
UNION-ALL								
TABLE ACCESS	FULL	ATHLETE1_REPLICA1	7,660	1	13	720,040		
TABLE ACCESS	FULL	ATHLETE2_REPLICA1	9,842	1	17	925,148		
TABLE ACCESS	FULL	ATHLETE3_REPLICA1	9,842	1	17	925,148		

* Unindexed columns are shown in red

Index Columns

Owner	Table Name	Index Name	Used in Plan	Columns	Uniqueness	Status	Index Type	Join Index
USER1_HP_FULL	ATHLETE1_REPLICA1	SYS_C007009		ATHLETEID	UNIQUE	VALID	NORMAL	NO
USER2_HP_FULL	ATHLETE2_REPLICA1	SYS_C007017		ATHLETEID	UNIQUE	VALID	NORMAL	NO
USER3_HP_FULL	ATHLETE3_REPLICA1	SYS_C007025		ATHLETEID	UNIQUE	VALID	NORMAL	NO

To compare with the execution cost without semi-join strategy, you can use a traditional inner-join query to process the same query:

```
SELECT C.Countryname, C.Code
FROM "USER_HP_FULL"."COUNTRY" C,
(SELECT * FROM "USER1_HP_FULL"."ATHLETE1_REPLICA1"
UNION SELECT * FROM "USER2_HP_FULL"."ATHLETE2_REPLICA1"
UNION SELECT * FROM "USER3_HP_FULL"."ATHLETE3_REPLICA1") A
WHERE C.CODE= A.CCODE ORDER BY C.Countryname;
```

Semi-join step-by-step

To show step-by-step semi-join procedure, a case of vertical fragmentation plan (shown in the following table) is used as an example where we have two sites USER1_VP and USER2_VP.

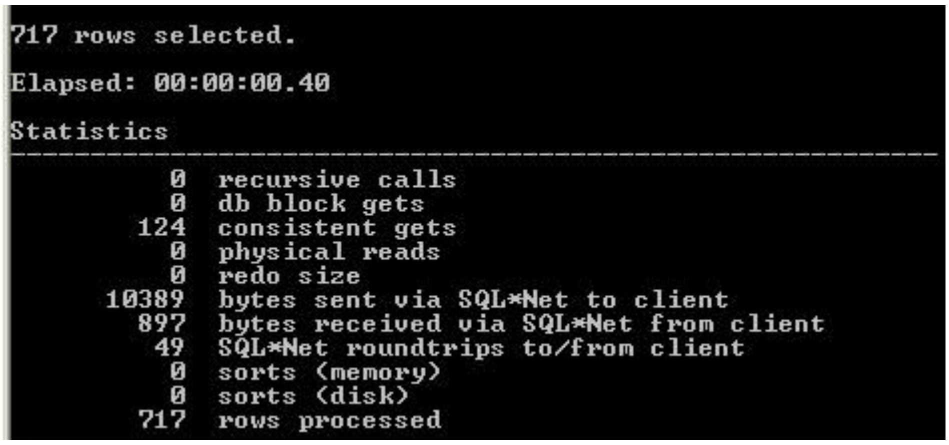
Distributed Site	Table	Column
USER1_VP	Athlete_V1	Athlete_ID, Fname, Sname

USER2_VP	Athlete_V2	Athlete_ID, dob, ccode, sportID

You are asked to go through "Distribution Design - Relation Fragmentation, Replication and Allocation", "SET TIMING" and "SET AUTOTRACE ON STATISTICS" as above. (Note: to load fragments into site USER1_VP, login Oracle XE as user "USER_VP" and run all script files in folder "Prac_1/VP-FULL". For USER2_VP, repeat the same operations.)

Step One - Get all the Athletes whose nationality is Australia from Athlete_V2

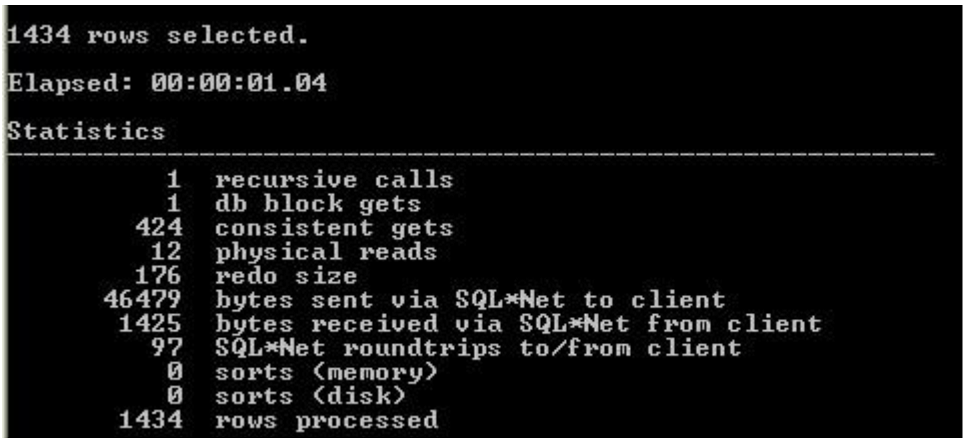
```
select AthleteID from "USER2_VP"."ATHLETE_V2" where CCODE='"AUS"';
```



```
717 rows selected.

Elapsed: 00:00:00.40

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        124  consistent gets
          0  physical reads
          0  redo size
      10389  bytes sent via SQL*Net to client
        897  bytes received via SQL*Net from client
         49  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        717  rows processed
```

Step Two - Use the results above to make a query to select all the related data from Athlete_V1

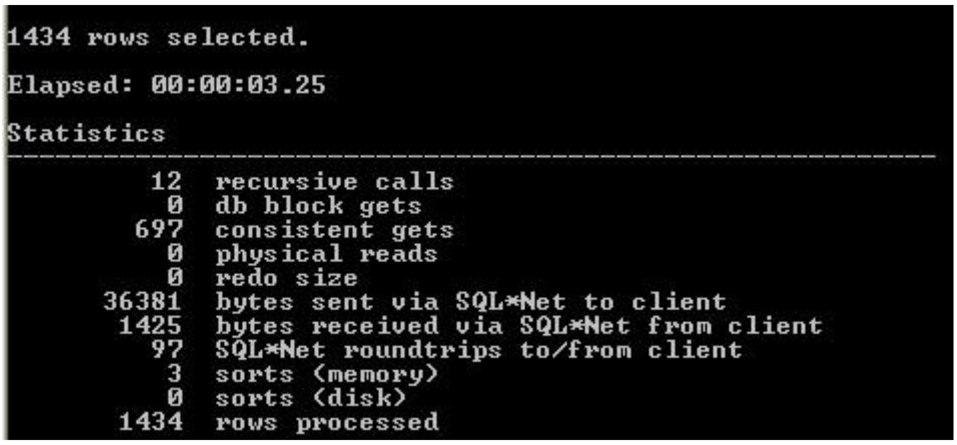
```
select * from "USER1_VP"."ATHLETE_V1" a where a.AthleteID in
(select AthleteID from "USER2_VP"."ATHLETE_V2" where CCODE='"AUS"');
```



```
1434 rows selected.

Elapsed: 00:00:01.04

Statistics
----------------------------------------------------------
          1  recursive calls
          1  db block gets
        424  consistent gets
         12  physical reads
        176  redo size
      46479  bytes sent via SQL*Net to client
       1425  bytes received via SQL*Net from client
         97  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       1434  rows processed
```

Step Three - Get the rest of the athlete details

```
select b.AthleteID, b.FName, b.SName, c.BDate, c.CCode, c.SportID
from "USER2_VP"."ATHLETE_V2" c,
(select * from "USER1_VP"."ATHLETE_V1" a where a.AthleteID in
(select AthleteID from "USER2_VP"."ATHLETE_V2" where CCODE='"AUS"'))
b where b.AthleteID= c.AthleteID;
```

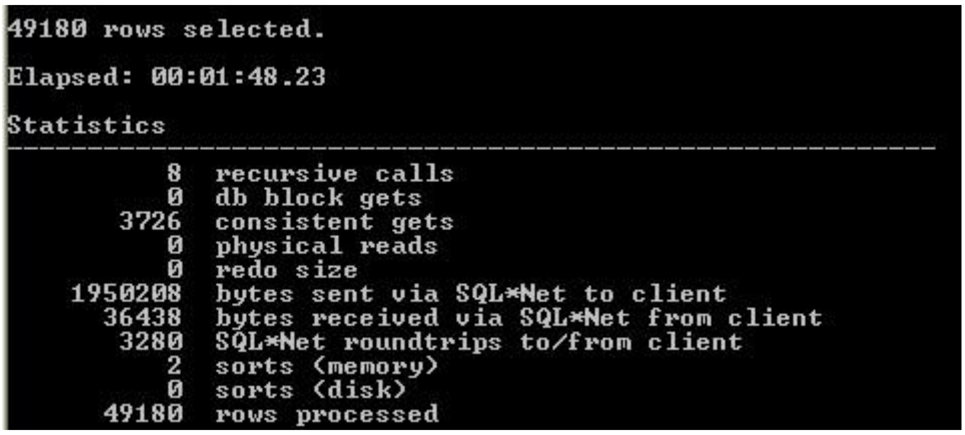


```
1434 rows selected.

Elapsed: 00:00:03.25

Statistics
----------------------------------------------------------
         12  recursive calls
          0  db block gets
        697  consistent gets
          0  physical reads
          0  redo size
      36381  bytes sent via SQL*Net to client
       1425  bytes received via SQL*Net from client
         97  SQL*Net roundtrips to/from client
          3  sorts (memory)
          0  sorts (disk)
       1434  rows processed
```

Note that in practice we only need the query at Step 3. The purpose of the previous two steps is to show you the intermediate results.

Step Four - Compare with a traditional inner-join query

```
select b.AthleteID, b.FName, b.SName, c.BDate, c.CCode, c.SportID
from "USER1_VP"."ATHLETE_V1" b, "USER2_VP"."ATHLETE_V2" c
where b.AthleteID= c.AthleteID;
```



```
49180 rows selected.

Elapsed: 00:01:48.23

Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       3726  consistent gets
          0  physical reads
          0  redo size
    1950208  bytes sent via SQL*Net to client
      36438  bytes received via SQL*Net from client
       3280  SQL*Net roundtrips to/from client
          2  sorts (memory)
          0  sorts (disk)
      49180  rows processed
```

Question: what is the total number of data items transmitted over computer network with and without semi-join?