3 Using the SQL Trace Facility and TKPROF

Objectives

After completing this lesson, you should be able to:

- Configure the SQL Trace facility to collect session statistics
- Discuss steps needed before tracing
- Enable application tracing
- Consolidate SQL trace files by using the `trcsess` utility
- Format trace files by using the `tkprof` utility
- Interpret the output of the `tkprof` command
- Verify the SQL problem by using a `TKProf` Report

Using the SQL Trace Facility: Overview

- Identify the most impacted session.
- Enable tracing on the identified session.
- Run the application coordinating with the user.
- Measure the client's response time for the operation.
- Disable tracing.
- Gather the trace file from the "`DIAGNOSTIC_DEST`" location.
- Generate a `TKProf` Report and sort the SQL statements in order of most elapsed time by using the following command:

```
OS> tkprof <trace file name> <output file name>
sort=fchela,exeela,prsela
```

Steps Needed Before Tracing

- Determine the location for diagnostic traces.
- Choose the most important affected sessions:
  - Find sessions with the highest CPU consumption.
  - Find sessions with the highest waits of a certain type.
  - Find sessions with the highest DB time (10*g* or later).
- Choose the most important affected clients, services, modules, actions, users, or sessions through Enterprise Manager (if possible) or through user feedback.

Location for Diagnostic Traces

DIAGNOSTIC_DEST

Diagnostic Data	Previous Location	ADR Location
Foreground process traces	`USER_DUMP_DEST`	`$ADR_HOME/trace`
Background process traces	`BACKGROUND_DUMP_DEST`	`$ADR_HOME/trace`
Alert log data	`BACKGROUND_DUMP_DEST`	`$ADR_HOME/alert` `$ADR_HOME/trace`
Core dumps	`CORE_DUMP_DEST`	`$ADR_HOME/cdump`
Incident dumps	`USER_DUMP_DEST` `BACKGROUND_DUMP_DEST`	`$ADR_HOME/incident/incdir_`*n*

V$DIAG_INFO

$ADR_HOME/trace <= Oracle Database 11*g* trace – critical error trace

Highest CPU Consumption: Example

Find sessions with the highest CPU consumption.

```
SELECT s.sid, s.serial#, p.spid as "OS PID",s.username,
       s.module, st.value/100 as "CPU sec"
FROM v$sesstat st, v$statname sn, v$session s, v$process p
WHERE sn.name = 'CPU used by this session'
AND st.statistic# = sn.statistic#
AND st.sid = s.sid
AND s.paddr = p.addr
AND s.last_call_et < 1800
AND s.logon_time > (SYSDATE - 240/1440)
ORDER BY st.value;
```

Highest Waits of a Certain Type: Example

Find sessions with the highest waits of a certain type.

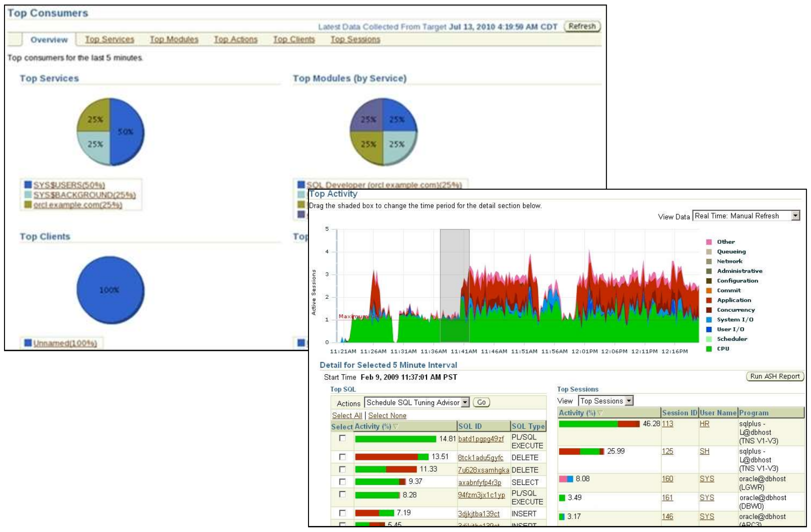
```
SELECT s.sid, s.serial#, p.spid as "OS PID", s.username,
       s.module, se.time_waited
FROM v$session_event se, v$session s, v$process p
WHERE se.event = '&event_name'
AND s.last_call_et < 1800
AND s.logon_time > (SYSDATE - 240/1440)
AND se.sid = s.sid
AND s.paddr = p.addr
ORDER BY se.time_waited;

Enter value for event_name: db file sequential read
```

Highest DB Time: Example

Find sessions with the highest DB time (10*g* or later).

```
SELECT s.sid, s.serial#, p.spid as "OS PID", s.username,
       s.module, st.value/100 as "DB Time (sec)",
       stcpu.value/100 as "CPU Time (sec)",
       round(stcpu.value / st.value * 100,2) as "%CPU"
FROM v$sesstat st, v$statname sn, v$session s,
     v$sesstat stcpu, v$statname sncpu, v$process p
WHERE sn.name = 'DB time'
AND st.statistic# = sn.statistic#
AND st.sid = s.sid
AND sncpu.name = 'CPU used by this session'
AND stcpu.statistic# = sncpu.statistic#
AND stcpu.sid = st.sid
AND s.paddr = p.addr
AND s.last_call_et < 1800
AND s.logon_time > (SYSDATE - 240/1440)
AND st.value > 0;
```

Enterprise Manager: Example



Top consumers for the last 5 minutes.

SYS$USERS(50%)
SYS$BACKGROUND(25%)
orcl.example.com(25%)

SQL Developer (orcl.example.com)(25%)

Unnamed(100%)

Drag the shaded box to change the time period for the detail section below.

Detail for Selected 5 Minute Interval

Start Time **Feb 9, 2009 11:37:01 AM PST**

Top SQL

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
☐	14.81	batd1pgpg49zf	PL/SQL EXECUTE
☐	13.51	8tck1adu5gyfc	DELETE
☐	11.33	7u628xsamhgka	DELETE
☐	9.37	axabnfyfp4r3p	SELECT
☐	8.28	94fzm3jx1c1yp	PL/SQL EXECUTE
☐	7.19	3djkjtba139ct	INSERT
☐	5.45	[illegible]	INSERT

Top Sessions

Activity (%)	Session ID	User Name	Program
46.28	113	HR	sqlplus - L@dbhost (TNS V1-V3)
25.99	125	SH	sqlplus - L@dbhost (TNS V1-V3)
8.08	160	SYS	oracle@dbhost (LGWR)
3.49	161	SYS	oracle@dbhost (DBW0)
3.17	146	SYS	oracle@dbhost (ARC3)

Available Tracing Tools: Overview

- SQL Trace at session level
- `ORADEBUG` at session level
- `LOGON` Trigger at a specific user level
- Enterprise Manager
- `DBMS_MONITOR`
- `DBMS_APPLICATION_INFO`
- `DBMS_SERVICE`
- `DBMS_SESSION`
- `SQLTXPLAIN` (MOS: 215187.1)

Trace Your Own Session with SQL

Set an event—10046 optimizer trace.

```
SQL> ALTER SESSION SET EVENTS
  2> '10046 trace name context forever, level 12';
```

Execute the statement of interest.

```
SQL> select *
  2> from hr.employees natural join hr.departments
  3> where department_id = 10;
```

Find and view the trace file.

When to Use*: It can be used where the session is accessible to the user prior to the start of the statements to be traced.*

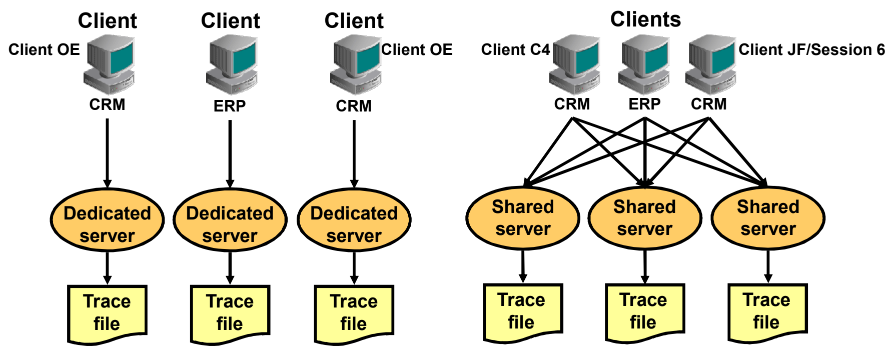
Trace with a Logon Trigger

When to Use*: There may be some situations where it is necessary to trace the activity of a specific user.*

```
CREATE OR REPLACE TRIGGER SYS.set_trace
  AFTER LOGON ON DATABASE
  WHEN (USER like  '&USERNAME')
  DECLARE
      lcommand varchar(200);
 BEGIN
 EXECUTE IMMEDIATE 'alter session set statistics_level=ALL';
 EXECUTE IMMEDIATE 'alter session set max_dump_file_size=UNLIMITED';
 DBMS_MONITOR.SESSION_TRACE_ENABLE(waits=> true, binds=> true);
END set_trace;
 /
```

Find and view the trace file.

Consideration: Tracing Challenge



- I want to retrieve traces from CRM service.
- I want to retrieve traces from client C4.
- I want to retrieve traces from session 6.

What Is a Service?

- Is a means of grouping sessions that perform the same kind of work
- Provides a single-system image instead of a multiple-instances image
- Is a part of the regular administration tasks that provide dynamic service-to-instance allocation
- Is the base for high availability of connections
- Provides a performance-tuning dimension
- Is a handle for capturing trace information

Using Services with Client Applications

```
ERP=(DESCRIPTION=
      (ADDRESS=(PROTOCOL=TCP)(HOST=mynode)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

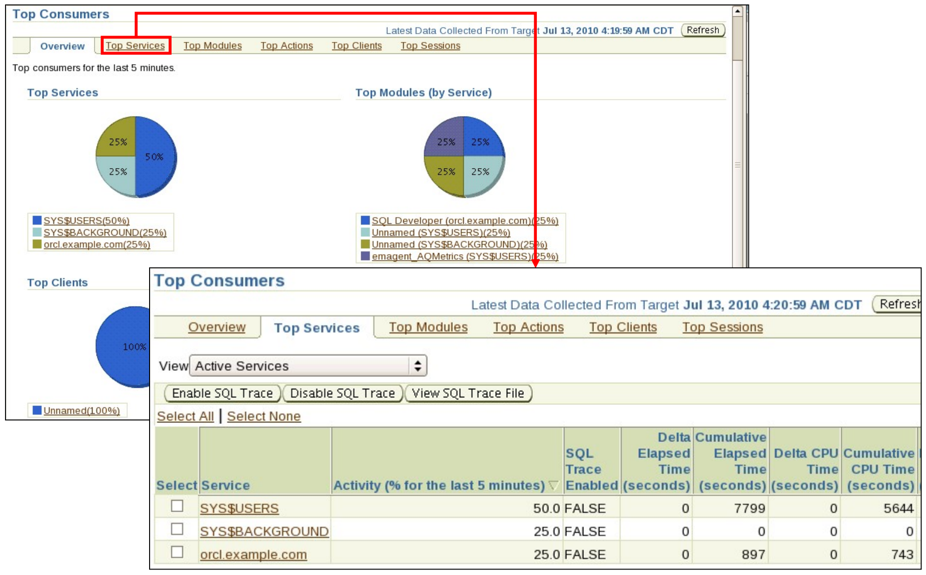
```
url="jdbc:oracle:oci:@ERP"
```

```
url="jdbc:oracle:thin:@(DESCRIPTION=
      (ADDRESS=(PROTOCOL=TCP)(HOST=mynode)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=ERP)))"
```

End-to-End Application Tracing

- Simplifies the process of diagnosing performance problems in multitier environments by allowing application workloads to be seen by:
  - Service
  - Module
  - Action
  - Session
  - Client
- End-to-end application tracing tools:
  - Enterprise Manager
  - DBMS_APPLICATION_INFO, DBMS_SERVICE, DBMS_MONITOR, DBMS_SESSION
  - SQL Trace and trcsess utility
  - tkprof

Trace with Enterprise Manager



Select	Service	Activity (% for the last 5 minutes)	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)	Delta CPU Time (seconds)	Cumulative CPU Time (seconds)
☐	SYS$USERS	50.0	FALSE	0	7799	0	5644
☐	SYS$BACKGROUND	25.0	FALSE	0	0	0	0
☐	orcl.example.com	25.0	FALSE	0	897	0	743

Trace with `DBMS_MONITOR`

- Automatic service aggregation level of statistics
- `DBMS_MONITOR` used for finer granularity of service aggregations:
  - `SERV_MOD_ACT_STAT_ENABLE`
  - `SERV_MOD_ACT_STAT_DISABLE`
- Possible additional aggregation levels:
  - `SERVICE_NAME/MODULE`
  - `SERVICE_NAME/MODULE/ACTION`
- Tracing services, modules, and actions:
  - `SERV_MOD_ACT_TRACE_ENABLE`
  - `SERV_MOD_ACT_TRACE_DISABLE`
- Database settings persist across instance restarts.

Service Tracing: Example

- Trace on service, module, and action:

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE('AP');
```

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE(-
      'AP', 'PAYMENTS', 'QUERY_DELINQUENT');
```

- Trace a particular client identifier:

```
exec DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE
   (client_id=>'C4', waits => TRUE, binds => FALSE);
```

Session Tracing: Example

- For all sessions in the database:

```
EXEC dbms_monitor.DATABASE_TRACE_ENABLE(TRUE,TRUE);
```

```
EXEC dbms_monitor.DATABASE_TRACE_DISABLE();
```

- For a particular session:

```
EXEC dbms_monitor.SESSION_TRACE_ENABLE(session_id=>
27, serial_num=>60, waits=>TRUE, binds=>FALSE);
```

```
EXEC dbms_monitor.SESSION_TRACE_DISABLE(session_id
=>27, serial_num=>60);
```

Trace Your Own Session

- Enabling trace:

```
EXEC DBMS_SESSION.SESSION_TRACE_ENABLE(waits =>
TRUE, binds => FALSE);
```

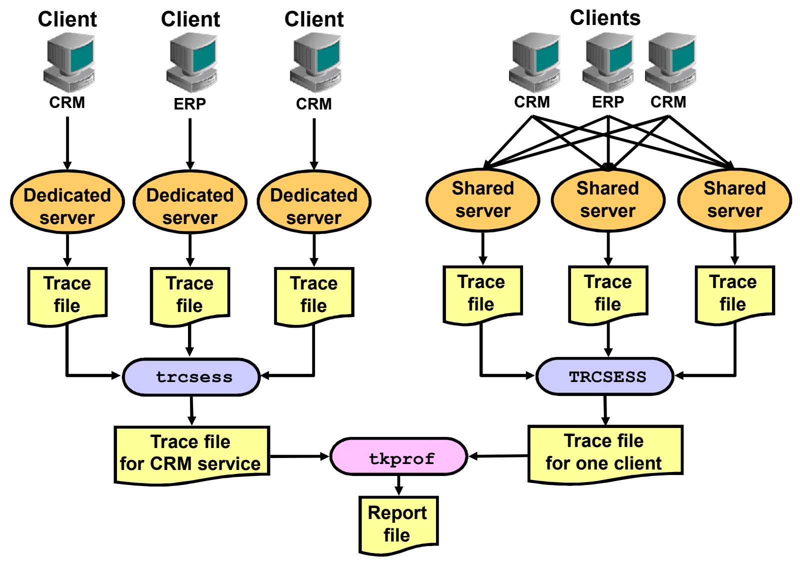
- Disabling trace:

```
EXEC DBMS_SESSION.SESSION_TRACE_DISABLE();
```

- Easily identifying your trace files:

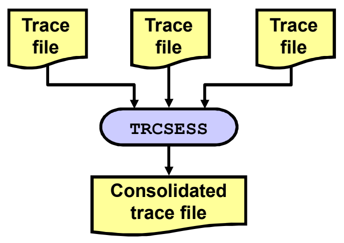
```
alter session set
tracefile_identifier='mytraceid';
```

The `trcsess` Utility



Invoking the `trcsess` Utility

```
trcsess   [output=output_file_name]
          [session=session_id]
          [clientid=client_identifier]
          [service=service_name]
          [action=action_name]
          [module=module_name]
          [<trace file names>]
```



The `trcsess` Utility: Example

```
exec dbms_session.set_identifier('HR session');
```

First session

Second session

```
exec dbms_session.set_identifier('HR session');
```

Third session

```
exec Dclient_id=>'HR session', waits => FALSE, -
binds => FALSE);
DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE( -
```

```
select * from employees;
```

```
select * from departments;
```

...

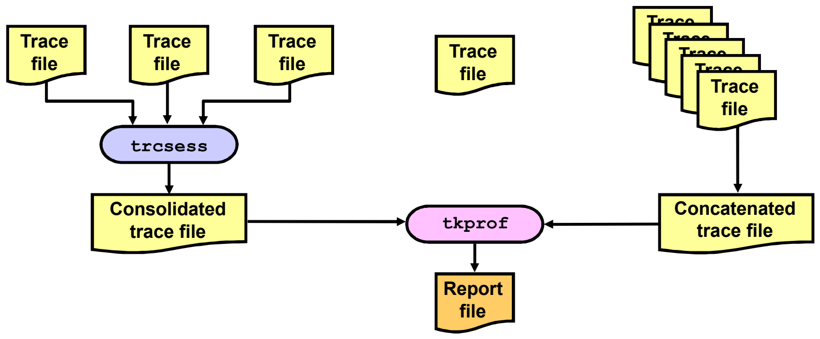
```
exec DBMS_MONITOR.CLIENT_ID_TRACE_DISABLE( -
client_id => 'HR session');
```

```
trcsess output=mytrace.trc clientid='HR session'
$ORACLE_BASE/diag/rdbms/orcl/orcl/trace/*.trc
```

Formatting SQL Trace Files: Overview

Use the `tkprof` utility to format your SQL trace files:

- Sort raw trace file to exhibit top SQL statements.
- Filter dictionary statements.



Invoking the `tkprof` Utility

```
tkprof inputfile outputfile [waits=yes|no]
                            [sort=option]
                            [print=n]
                            [aggregate=yes|no]
                            [insert=sqlscriptfile]
                            [sys=yes|no]
                            [table=schema.table]
                            [explain=user/password]
                            [record=statementfile]
                            [width=n]
```

tkprof Sorting Options

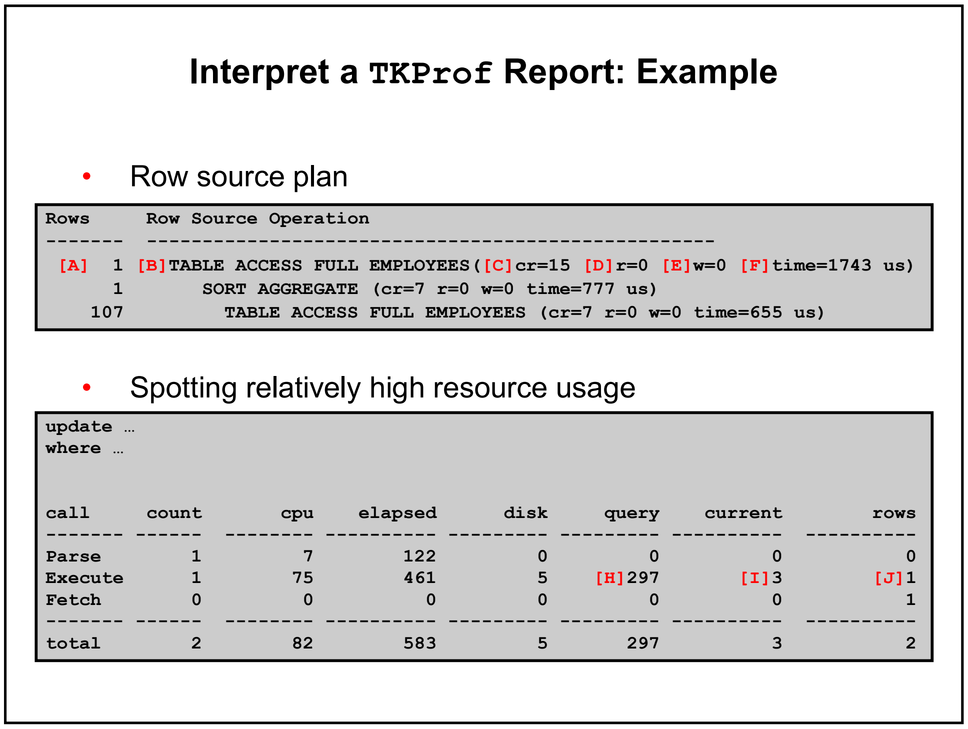
Sort Option	Description
prscnt	Number of times parse was called
prscpu	CPU time parsing
prsela	Elapsed time parsing
prsdsk	Number of disk reads during parse
prsqry	Number of buffers for consistent read during parse
prscu	Number of buffers for current read during parse
prsmis	Number of misses in the library cache during parse
execnt	Number of executes that were called
execpu	CPU time spent executing
exeela	Elapsed time executing
exedsk	Number of disk reads during execute
exeqry	Number of buffers for consistent read during execute
execu	Number of buffers for current read during execute

`tkprof` Sorting Options

Sort Option	Description
`exerow`	Number of rows processed during execute
`exemis`	Number of library cache misses during execute
`fchcnt`	Number of times fetch was called
`fchcpu`	CPU time spent fetching
`fchela`	Elapsed time fetching
`fchdsk`	Number of disk reads during fetch
`fchqry`	Number of buffers for consistent read during fetch
`fchcu`	Number of buffers for current read during fetch
`fchrow`	Number of rows fetched
`userid`	User ID of user that parsed the cursor

`TKProf` Report Structure

```
...
select max(cust_credit_limit) from customers where cust_city ='Paris'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.02       0.02          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0         15          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.02       0.02          0         15          0           1

Misses in library cache during parse: 1
Optimizer mode: FIRST_ROWS
Parsing user id: 88

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  TABLE ACCESS FULL EMPLOYEES (cr=15 r=0 w=0 time=1743 us)
      1    SORT AGGREGATE (cr=7 r=0 w=0 time=777 us)
    107      TABLE ACCESS FULL EMPLOYEES (cr=7 r=0 w=0 time=655 us)

Elapsed times include waiting on following events:
Event waited on                             Times  Max. Wait  Total Waited
---------------------------------------- Waited ----------  ------------
SQL*Net message to client                      2       0.00          0.00
SQL*Net message from client                    2       9.62          9.62
```



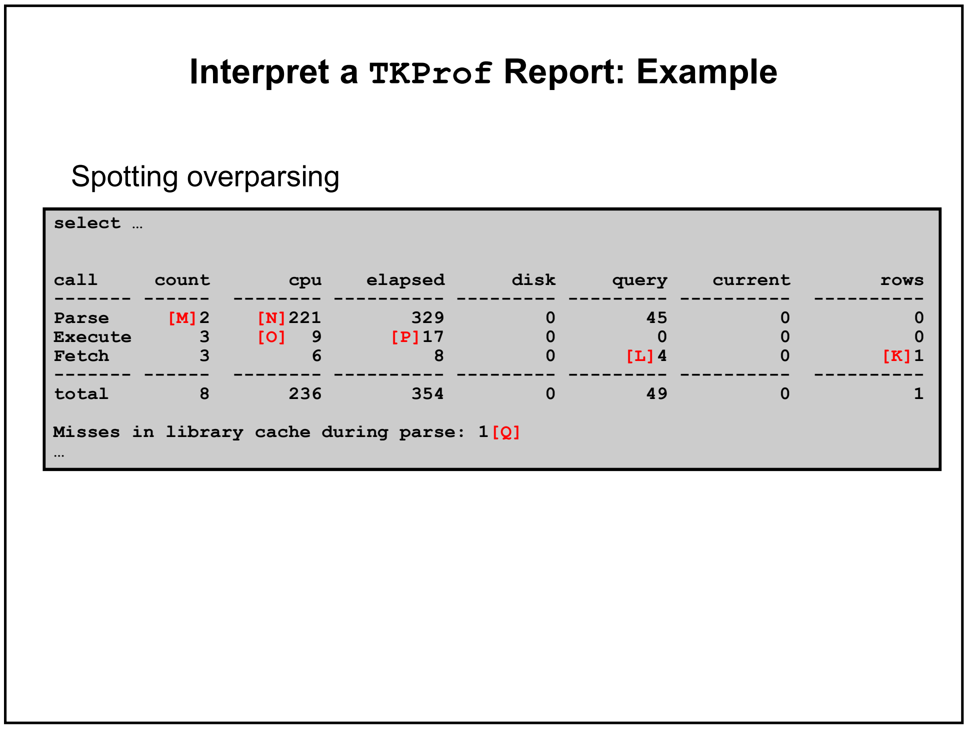
Interpret a `TKProf` Report: Example

- Row source plan

```
Rows     Row Source Operation
-------  ---------------------------------------------------
 [A]   1 [B]TABLE ACCESS FULL EMPLOYEES([C]cr=15 [D]r=0 [E]w=0 [F]time=1743 us)
       1      SORT AGGREGATE (cr=7 r=0 w=0 time=777 us)
     107        TABLE ACCESS FULL EMPLOYEES (cr=7 r=0 w=0 time=655 us)
```

- Spotting relatively high resource usage

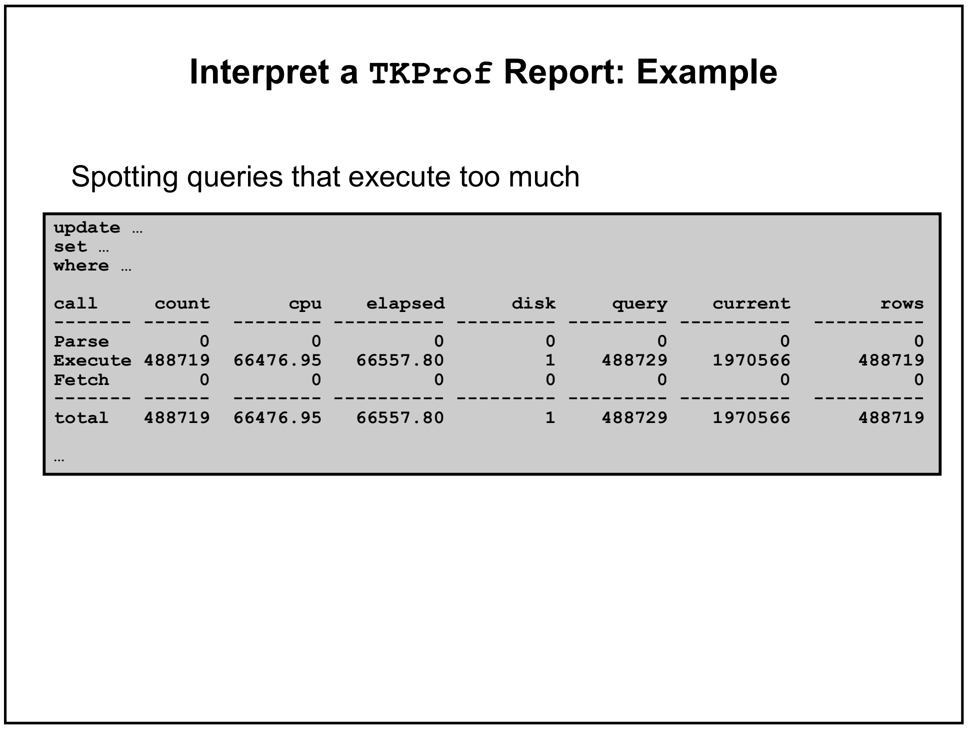
```
update …
where …


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1         7        122          0          0          0           0
Execute      1        75        461          5     [H]297       [I]3        [J]1
Fetch        0         0          0          0          0          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2        82        583          5        297          3           2
```



Interpret a `TKProf` Report: Example

Spotting overparsing

```
select …


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse     [M]2    [N]221        329          0         45          0           0
Execute      3    [O]  9     [P]17           0          0          0           0
Fetch        3         6          8          0      [L]4           0        [K]1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        8       236        354          0         49          0           1

Misses in library cache during parse: 1[Q]
…
```



Interpret a `TKProf` Report: Example

Spotting queries that execute too much

```
update …
set …
where …

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0         0          0          0          0          0           0
Execute 488719  66476.95   66557.80          1     488729    1970566      488719
Fetch        0         0          0          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total   488719  66476.95   66557.80          1     488729    1970566      488719

…
```

What to Verify: Example

Was total elapsed time in `TKProf` account for the application response time measured when the application was executed?

```
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse     1165      0.66       2.15          0         45          0           0
Execute   2926      1.23       2.92          0          0          0           0
Fetch     2945    117.03     398.23       5548    1699259         16       39654
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     7036    118.92     403.31       5548    1699304         16       39654

OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0      0.00       0.00          0          0          0           0
Execute      0      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        0         0       0.00          0          0          0           0
……
```

What to Verify: Example

Has the time spent parsing, executing, and fetching account for most of the elapsed time recorded in the trace?

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.00       0.00          0         14          0          14
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10         0       0.00          0         14          0          14

…

Rows     Row Source Operation
-------  ---------------------------------------------------
     14  TABLE ACCESS FULL EMPLOYEES (cr=14 r=0 w=0 time=377 us)

Elapsed times include waiting on following events:
Event waited on                             Times Max. Wait  Total Waited
---------------------------------------- Waited ----------  ------------
SQL*Net message to client                     8       0.00          0.00
SQL*Net message from client                   8      16.36         78.39
```

What to Verify: Example

- Is the query you expect to tune shown at the top of the `TKProf` Report?
- Does the query spend most of its time in the Execute and Fetch phases (not Parse phase)? Make sure the trace file contains data only from the recent test.

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse      555    100.09     300.83          0          0          0           0
Execute    555      0.42       0.78          0          0          0           0
Fetch      555     14.04      85.03        513    1448514          0       11724
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     1665    114.55     368.65        513    1448514          0       11724
```

Quiz

In an environment with an applications server that uses a connection pool, you use _________ to identify which trace files need to be combined to get an overall trace of the application.

a. `trcsess`
b. `tkprof`
c. Oracle SQL Developer
d. `DBMS_APPLICATION_INFO`

Summary

In this lesson, you should have learned how to:

- Configure the SQL Trace facility to collect session statistics
- Discuss steps needed before tracing
- Enable application tracing
- Consolidate SQL trace files by using the `trcsess` utility
- Format trace files by using the `tkprof` utility
- Interpret the output of the `tkprof` command
- Verify the SQL problem by using a `TKProf` Report

Practice 3: Overview

This practice covers the following topics:

- Creating a service
- Tracing your application by using services
- Interpreting trace information by using `trcsess` and `tkprof`