Oracle® Reports Services

Release Notes

Release 1.0.2 for Windows or UNIX

October 2000

Part No. A86271-01

The following changes and enhancements have been made for Oracle Reports Services with Oracle9*i* Application Server release 1.0.2:

- Availability of Oracle Portal release 3.0 with Oracle Reports Services 6*i* security
- Changes for NLS
- Changes for server configuration with Oracle Portal release 2.2 and Oracle Portal release 3.0
- Changes in the security API in the RWK layer
- Extended aggregate operations and analytic functions
- Oracle Reports Services server X Windows session requirement

Oracle Portal

This release introduces Oracle Portal release 3.0, which works with Oracle Reports Services 6*i* security. The following changes have been made:

- The images are installed with Oracle Portal release 3.0. You no longer needs to manually copy the image files as mentioned in the previous *Release Notes*. The files are no longer staged in the release 2 patch.
- If the login server is not on the same database as Oracle Portal when you install Oracle Reports Services with Oracle Portal integration scripts, then you are prompted to input the user name, password, and TNSname of the login server. Internally, the script creates a database link from Oracle Portal to the login server and allows you to access some of the functions in the login server.

NLS

Oracle Corporation provides a translated user interface (UI) for all Oracle Reports Services languages. This allows you to use the Oracle Reports Services 6*i* security feature with Oracle Portal in a language other than English, except for Arabic and Hebrew. The UI can be installed by using SQL*Plus and SQL*Loader scripts. These scripts are installed automatically but must be run by you. The scripts are installed at the following location:

```
<ORACLE_HOME>/<REPORTS60>/ADMIN/SECURITY
```

SQL*Plus

To run the SQL*Plus SQL script, you do the following:

1. Set the NLS_LANG=`language_territory.charset` parameter to the O/S character set. For example, to install French in UNIX, you would set the character set to WE8ISO8859P1. The scripts are encoded in the usual character set for the language and platform combination.
2. Run SQL*Plus and log into the Oracle Portal database instance using the administrative user ID and password. Refer to Oracle Portal documentation for more information about logging into the Oracle Portal database.
3. Run the SQL script by typing the following:

   ```
   START rep<lang>30.sql
   ```

4. If there are any errors, then they are displayed in the SQL*Plus session. Errors can occur for the following reasons:
   - An unforeseen database problem
   - An attempt was made to run the script language more than once
   - An attempt was made to process a multibyte language in a single-byte operating system locale
5. You can install as many languages as you want. The strings are stored separately in the database and do not overwrite each other. The language displayed is controlled by the NLS_LANG parameter for Oracle Portal release 3.0.

SQL*Loader

To run the SQL*Loader CTL script, you do the following:

1. Set the NLS_LANG=`language_territory.charset` parameter to the O/S character set. For example, to install French in UNIX, you would set the character set to WE8ISO8859P1. The scripts are encoded in the usual character set for the language and platform combination.
2. Run the SQL*Loader CTL script using the following command line:

   ```
   SQLLDR userid=<admin>/<password>CONTROL=rep<lang>30.ctl
   ```

3. Errors are logged to the `rep<lang>30.log` file. Rejected lines are logged to the `rep<lang>.bad` file. Errors can occur for the following reasons:
   - An unforeseen database problem
   - An attempt was made to run the script for a language more than once
   - An attempt was made to process a multibyte language in a single-byte operating system locale
4. You can install as many languages as you want. The strings are stored separately in the database and do not overwrite each other. The language displayed is controlled by the NLS_LANG parameter for Oracle Portal release 3.0.

Server Configuration

You need to have the following entry in the Oracle Reports Services server configuration file if you are installing Oracle Portal release 2.2, where `tnsname` is the instance where Oracle Portal release 2.2 is installed:

```
SECURITYTNSNAMES=<webdb_2.2_tnsname>
```

You need to have the following entries in the Oracle Reports Services server configuration file if you are installing Oracle Portal release 3.0:

```
SECURITYTNSNAMES=<oracle_portal_tnsname>
PORTALUSERID=<portal_username>/<portal_password>
```

where:

`tnsname`	is the instance where Oracle Portal release 3.0 is installed.
`portal_username`	is the name of the user authorized to log into the Oracle Portal database.
`portal_password`	is the password of the authorized Oracle Portal user.

The SECURITYTNSNAME and PORTALUSERID parameters can be added to the Oracle Reports Services server configuration file. For example, `rep60_machinename.ora` found in the following Windows NT directory:

```
%ORACLE_HOME%\REPORT60\SERVER
```

Or, rep60_machinename.ora found in the following UNIX directory:

```
$ORACLE_HOME/reports60/server
```

Security

Oracle Corporation has modified the security API in the RWK layer. If you have created your own version of RWK60.DLL for an earlier version, then you need to make some changes in your code and recompile with the new RWKSS.H. The RWKSS_ValidateSession() function has been added for validating an Oracle Portal user session, and RWKSS_AuthenticationSystemUser() has been added to distinguish the database authentication RWKSS_AuthenticateUser().

Extended Aggregate Operations and Analytic Functions

Oracle Reports Services now supports extended aggregate operations and analytic functions.

Aggregate operations return a single result row based on the groups of rows, rather than on single rows. Aggregate functions can appear in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement, where Oracle8*i* divides the rows of a queried table or view into groups. For example:

```
SELECT dname, sum(sal)
FROM dept, emp
WHERE dept.deptno =emp.deptno
GROUP BY dname
```

The database applies the aggregate functions to each group of rows and returns a single result row for each group. So the above example shows the total sum of all salaries on an individual department basis.

Introduced in Oracle8*i* release 8.1.5, extended aggregate operations extend this functionality by adding CUBE and ROLLUP extension, where super aggregate groups are produced in addition to the regular groupings (as extra rows). ROLLUP creates subtotals at increasing levels of aggregation, from the most detailed up to a grand total. CUBE is an extension similar to ROLLUP, enabling a single statement to calculate all possible combinations of subtotals. CUBE can generate the information needed in cross-tabulation reports with a single query. You can distinguish between the real data rows being returned by the query and the extra rows added by the CUBE and ROLLUP extensions by using the GROUPING function in the select list. For real rows the GROUPING function returns a 0; otherwise, it returns a 1.

Oracle8*i* release 2 (8.1.6) introduced a powerful new family of SQL functions for business intelligence and data warehousing applications. These functions are collectively called analytic functions and they provide significantly improved performance and simplified coding for many business analysis queries. These new SQL functions are also being reviewed by ANSI for addition to the new SQL standard. Oracle Corporation has created four families of analytic functions, each of which contains several functions:

- Ranking family
- Window aggregate family
- Reporting aggregate family
- LAG and LEAD family

Ranking Family

This family supports business questions such as "show the top 10 and bottom 10 salespersons for each region" or "show, for each region, the salespersons that make up 25% of the sales." The functions examine the entire output before producing an answer. Oracle Corporation provides RANK, DENSE_RANK, PERCENT_RANK, CUME_DIST, and NTILE functions.

Window Aggregate Family

This family addresses questions such as "show the 13-week moving average of a stock price" or "show the cumulative sum of sales per each region." The new features provide moving and cumulative processing for all the SQL aggregate functions, including AVG, SUM, MIN, MAX, COUNT, VARIANCE, and STDDEV.

Reporting Aggregate Family

One of the most common types of calculations is the comparison of a non-aggregate value to an aggregate value. All percent-of-total and market share calculations require this processing. The reporting aggregate family makes these sorts of calculations simple. It lets you place values calculated at different aggregation levels on the same row. Without needing a join operation, you can now compare aggregate values to the detail rows. The new family provides reporting aggregate processing for all SQL functions, including AVG, SUM, MAX, COUNT, VARIANCE, and STDDEV.

LAG and LEAD Family

Studying change and variation is at the heart of analysis. Necessarily, this involves comparing the values of different rows in a table. While this has been possible in SQL, usually through self-joins, it has not been efficient or easy to formulate. The LAG and LEAD family enables queries to compare different rows of a table simply by specifying an offset from the current row.

Look at the following example of the application of some of these functions and operations. Imagine that the HR manager in your organization has requested the following information about the employees in your company:

> "I'm trying to make sure our compensation policy and job bands are fair and in line with corporate directives. To determine this, I need to see the following from our HR database:
>
> 1. A breakdown of the number of employees and the average salary for each job we have in the company. I'd like to see the first for the entire company, and then on a per department basis.

2. For each employee, on a per department basis, I would like to see their total compensation and how this compensation ranks within the company as a whole, within their department (and it's proportion), and how much above or below the company compensation average this is.

On the same report, I'd like to see their date of employ, their seniority in the company (that is, the order in which they were employed), who was the next person employed after them, and how long afterwards."

With the new Oracle8*i* analytic functions, you can now achieve all of this for your HR manager with just two SQL statements and no processing of the data in PL/SQL. Take a look at the first request. This can be written as follows:

```
SELECT GROUPING(dname) dept_grouping_code,
 DECODE(GROUPING(dname), 1, 'All Departments',initcap(dname)) AS dname,
 GROUPING(job) job_grouping_code,
 DECODE(GROUPING(job), 1, 'All Jobs', job) AS job, COUNT(*) "Total Empl",
 AVG(sal+nvl)(comm,0)) "Average Comp"
FROM emp, dept
WHERE dept.deptno = emp.deptno
GROUP BY CUBE (dname, job)
```

For the second request, this can be expressed in SQL as follows:

```
SELECT emp.deptno,
       dept.dname,
      avg(sal+nvl(comm,9)) over (partition by dept.deptno) avg_dept_sal,
       ename,
       job,
       sal,
       nvl(comm,0),
       (sal+nvl(comm,0)) Compensation,
       hiredate
RANK () OVER (PARTITION BY emp.deptno ORDER BY (sal+nvl(comm,0)) DESC) as rk,
RANK () OVER (ORDER BY (sal+nvl(comm,0)) DESC) "Rank in Company",
RANK () OVER (ORDER BY hiredate) "rank in employ",
((LEAD(hiredate, 1) OVER (ORDER BY hiredate))-hiredate) "Days over emp",
(LEAD(ename,1) OVER (ORDER BY hiredate)) next_emp
FROM emp, dept
WHERE dept.deptno = emp.deptno
ORDER BY rk
```

Naturally, you want to produce boardroom quality output for your HR manager, so instead of SQL*Plus, you use Oracle Reports Services to produce these reports.

Now, the question is, how do you put these queries with this special syntax in your enterprise report? The answer is simple, just as you would do it with any other regular SQL statement.

Oracle Reports Services passes the SQL straight through to the database so developers can utilize all these functions and the extended aggregate operations within individual (stand-alone or not linked) queries in all versions of Oracle Reports Services. In addition, Oracle Reports Services 6*i* release 2 has been further enhanced to understand these extensions to the SQL syntax, so that queries using these functions can be linked to any other query or the break order of fields changed in the data model. Oracle Reports Services 6*i* release 2 is required since these features actually rewrite the SQL statement behind the scenes in Oracle Reports Services.

You can create your reports either with the wizard or by manually creating the queries with break groups defined for a group-above report. You do not link the queries since the HR manager wanted two summaries. You can use the multisectioning and bursting feature of Oracle Reports Services to display the results of the first query on the first section, and then the results of the second query in another. Of course, with a single run of the report, you can run each section to a number of different formats (for example, PDF, PostScript, HTML, or RTF) and to a number of different destinations (for example, a printer, e-mail, or a portal).

You now run the report wizard again and create your layout for the second analysis. You can also invoke the chart wizard to view some of the data pictorially.

As you can see, the new extended aggregate operations and analytic functions are very powerful and can easily be used to enhance the analysis of your data from within Oracle Reports Services.

Oracle Reports Services Server X Windows Session Requirement

In order to run bitmapped reports, the engines spawned by the Oracle Reports Services server need to have access to an appropriate windowing system. On Windows platforms this is a non-issue, but on non-Windows platforms this means that a valid X Windows session must be available. To ensure that this is the case, be sure that the Oracle Reports Services server is started from the session that has a valid DISPLAY environment variable. If this is not the case, then you see REP-3000 and REP-1800 errors.