Oracle Views
What can they do for you?

CDOUG October 1999

Suhas Upadhye
Principal Consultant
Keane Inc.

What we will cover:

Outline:

1.1. Abstract
1.2. What is a VIEW?
1.3. View DDL's
1.4. Steps to create a View
1.5. Obtaining Information on Views
1.6. Practical Considerations
1.7. Categories of Views
1.8. DML operations on Views
1.9. Triggers on Views
1.10. Data Dictionary Views
1.11. View advantages
1.12. Materialized Views
1.13. New for Oracle8i?
1.14. Where do we go from here?

Abstract

- Views are a powerful tool in a relational database such as Oracle.
- This presentation is aimed at novice and experienced users alike of Oracle 7.x/8.x.
- Various usage's will be explained with practical examples .
- New additions for views in Oracle8 will be touched upon.

Introduction to Views:

What is a VIEW?

- A view is a *named query*. It is "a tailored presentation of the data contained in one or more tables (or other views)."

  - For SQL, *view* = a named or derived *virtual* table
  - For users, *view* = a table (which in fact does not exist!)
  - View ≠ a separate copy of an underlying table
  - View = a *window* into the underlying table
  - Changes can be done from *both* directions:

    underlying table ↔ view

Tables v/s Views

- Views are created, dropped or granted access to, identical to a table.
- How do views differ from tables?

TABLE	VIEW
• Contains Data	• Does not contain data
• Has indexes	• Indexes cannot be created on a view
• Is stored in a specific location	• Its *definition* is stored in the data dictionary
• Number of tables could be limited by Oracle Version.	• No limit to number of views you may have – upper limit dictated by SYSTEM tablespace.

View DDL's

- CREATE VIEW <view name>
  AS <sub query>
  [WITH CHECK OPTION];

  (Note: WITH CHECK OPTION means "don't let me update the view if the updated rows would not be retrieved by the query")

- DROP VIEW <view name>;

```
CREATE VIEW reorder
   AS SELECT   id, onhand, reorder
   FROM stock
   WHERE onhand < reorder
   WITH CHECK OPTION;
```

View DDL's

Important features:

- A view can be dependent on multiple tables/views.
- These tables/views must exist in the database, and the columns must be defined.
- The individual creating the view must have 'CREATE VIEW' privileges or a role that includes it.
- The individual creating the view must also have been granted ***direct*** access and necessary privileges on the tables in the view (not via a role).

  (The end user using the view (DML) need not have necessary privileges on the base tables themselves, but privileges on the view itself will suffice.)
- If a table the view depends on is dropped, the view is not dropped, it is merely marked 'invalid' and any access attempts will result in an error.

Steps to create a View

- **Scratch-Pad**: Create the SQL statement in a file first.
- **Correct**: Continue working on it till it yields the desired result(s).
- **Tune**: Tune the SQL using the EXPLAIN plan.
  - Full table scan on driving table OK now (as you may have additional qualifiers at run time)
  - Full table scans on supporting tables should not be necessary.
  - Determine if you are not using an index that you thought you would. Add columns in WHERE clause to facilitate usage of composite indexes.
- **Create**: Add the 'create view' line to the SQL and run it to create the view object.
- **Maintain**: If the underlying tables, security requirements and/or business rules change necessitating a view change, re-tune before deploying it.

Obtaining Information on Views

- **USER_OBJECTS**:

```
SQL> desc user_objects;
 Name                          Null?    Type
 ----------------------------- -------- ----
 OBJECT_NAME                            VARCHAR2(128)
 OBJECT_ID                              NUMBER
 OBJECT_TYPE                            VARCHAR2(13)
 CREATED                                DATE
 LAST_DDL_TIME                          DATE
 TIMESTAMP                              VARCHAR2(75)
 STATUS                                 VARCHAR2(7)
```

- **USER_VIEWS**:

```
SQL> desc user_views;
 Name                          Null?    Type
 ----------------------------- -------- ----
 VIEW_NAME                     NOT NULL VARCHAR2(30)
 TEXT_LENGTH                            NUMBER
 TEXT                                   LONG
```

Practical Considerations

- **Myth** :

  Performance degradation -- at time of execution, view query is usually merged with the statement resulting in single SQL statement => no more additional time to give results.

```
CREATE VIEW emp_view AS
  SELECT empno, ename, sal, loc
    FROM emp, dept
    WHERE emp.deptno = dept.deptno AND dept.deptno = 10;
```

  - Now consider the following user-issued query:

```
SELECT ename
  FROM emp_view
  WHERE empno = 9876;
```

  - The final query constructed by Oracle is

```
SELECT ename
  FROM emp, dept
  WHERE emp.deptno = dept.deptno AND
      dept.deptno = 10 AND  emp.empno = 9876;
```

Practical Considerations

- **Do's/Don'ts** :
  - Adopt consistent naming standards for views. E.g. suffix views with '_v'
  - Expand all columns in a the SQL instead of 'select * from'
  - Views should not be built using views -- though Oracle allows it, it does seem to negetively affect performance.

- **Caveats:**
  - A view's query cannot select the CURRVAL or NEXTVAL pseudocolumns.
  - If a view's query selects the ROWID, ROWNUM, or LEVEL pseudocolumns, they *must have aliases* in the view's query.

Practical Considerations

- **Caveats (cont.):**
  - You can define a view with a query that uses an asterisk (*) to select all the columns of a table:

```
CREATE VIEW emp_vu
AS SELECT * FROM emp;
```

Oracle *translates* the asterisk into a list of all the columns in the table at the time the CREATE VIEW statement is issued. If you subsequently add new columns to the table, the view *will not contain* these columns unless you recreate the view by issuing another CREATE VIEW statement with the OR REPLACE option. Oracle recommends that you explicitly specify all columns in the select list of a view query, rather than use the asterisk.

Categories of Views

- **Simple View :**

  The query for a simple view contains only one table

  - **Column subset**

```
CREATE VIEW s_view
        AS select sno, sname from s where.. ;
```

  - **Row subset**

```
CREATE VIEW s_view
        AS select * from s where.. ;
```

Categories of Views

- **Join View :**

  The query contains the join of one or more tables/views.

```
CREATE VIEW ss_pp
        AS select s.sno, s.sname, p.pno, p.pname
        from s, p
        where s.city = p.city;
```

Categories of Views

- **Partition View :**

  The query combines a large table that has been divided into several physical partitions, into a single view for query purposes. The partition view queries each partition and uses a ‘UNION’ to join the tables (partitions).

- **Object View :**

  An object view synthesizes objects based on queries of relational or object tables.

DML operations on Views

- Assume:
  - sno is the *primary key* on table s
  - CREATE VIEW sno_city_v
    AS select sno, city from s;
  - CREATE VIEW status_city_v
    AS select status, city from s;
- INSERT into “sno_city_v” is o.k.
- INSERT into “status_city_v” cannot be done!!
  - Reason? Because “SNO” cannot be NULL, but we cannot insert SNO via the view “status_city”

DML operations on Views

- **Updateable Join Views (concepts)**
  - A *join view* is a view that contains a join. Join views are updateable under the certain conditions.
  - A *key-preserved table* is a table in a join view, all of whose key columns are present as keys in the join view.

DML operations on Views

- **Updateable Join Views**

  You can execute the DML statements INSERT, UPDATE, and DELETE on a join view only provided that all of the following are true:

  - The DML statement affects only one of the tables underlying the join.
  - If the statement is UPDATE, then all columns updated are extracted from a *key-preserved table*.

DML operations on Views

- **Updateable Join Views(cont.)**
  - If the statement is DELETE, then there is one and only one *key-preserved table* in the join. (This table may be present more than once in the join, unless the view has the CHECK OPTION.)
  - If the statement is INSERT, then all columns into which values are inserted come from a key-preserved table, and the view does not have the CHECK OPTION.

Triggers on Views

- **Triggers**

  Oracle allows you to define procedures that are implicitly executed when an INSERT, UPDATE, or DELETE statement is issued against the associated table. These procedures are called database *triggers*.

  - A “BEFORE” trigger executes *prior* to the triggering statement
  - An “AFTER” trigger executes *subsequent* to the triggering statement, while
  - The “INSTEAD OF” trigger executes *rather than* the triggering statement.

  The INSTEAD OF trigger is available for views only -- they cannot be placed on a table. This is the only type of trigger to place on a view. The trigger will execute separately for each row in the view. These provide a transparent way of updating a normally non-updateable view.

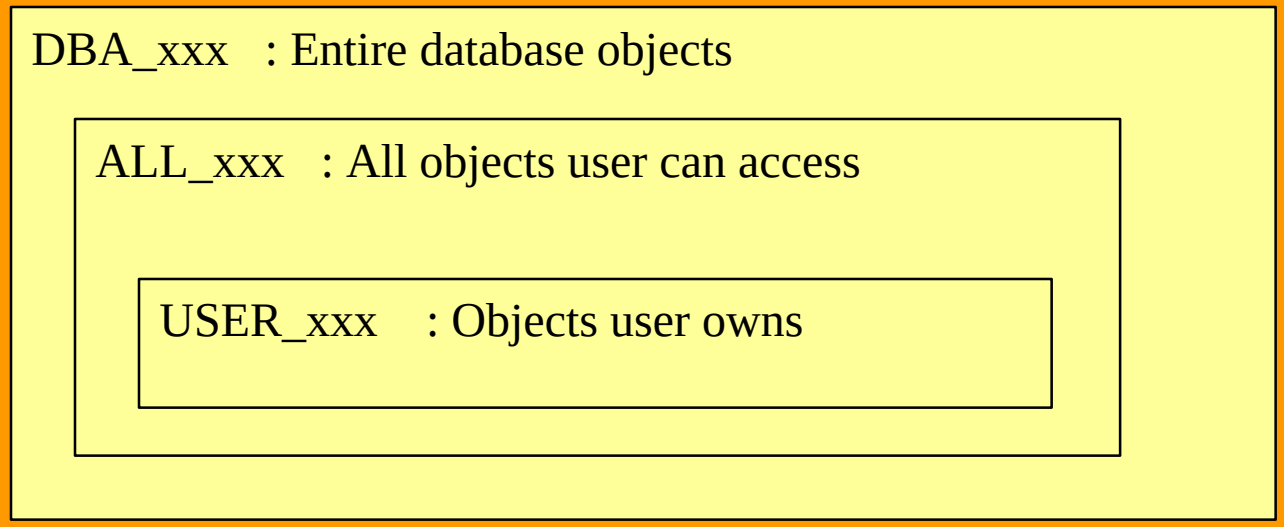
Data Dictionary Views

- **Static Data Dictionary Views**
- **Dynamic Performance views.**

Data Dictionary Views

- **Static Data Dictionary Views**
  - Base Tables
    - normalized
    - created with the ***sql.bsq*** script
  - Data Dictionary views
    - Views simplify the base table information
    - created with the ***catalog.sql*** script
  - Data-dictionary tables consume space in physical database files

Data Dictionary Views



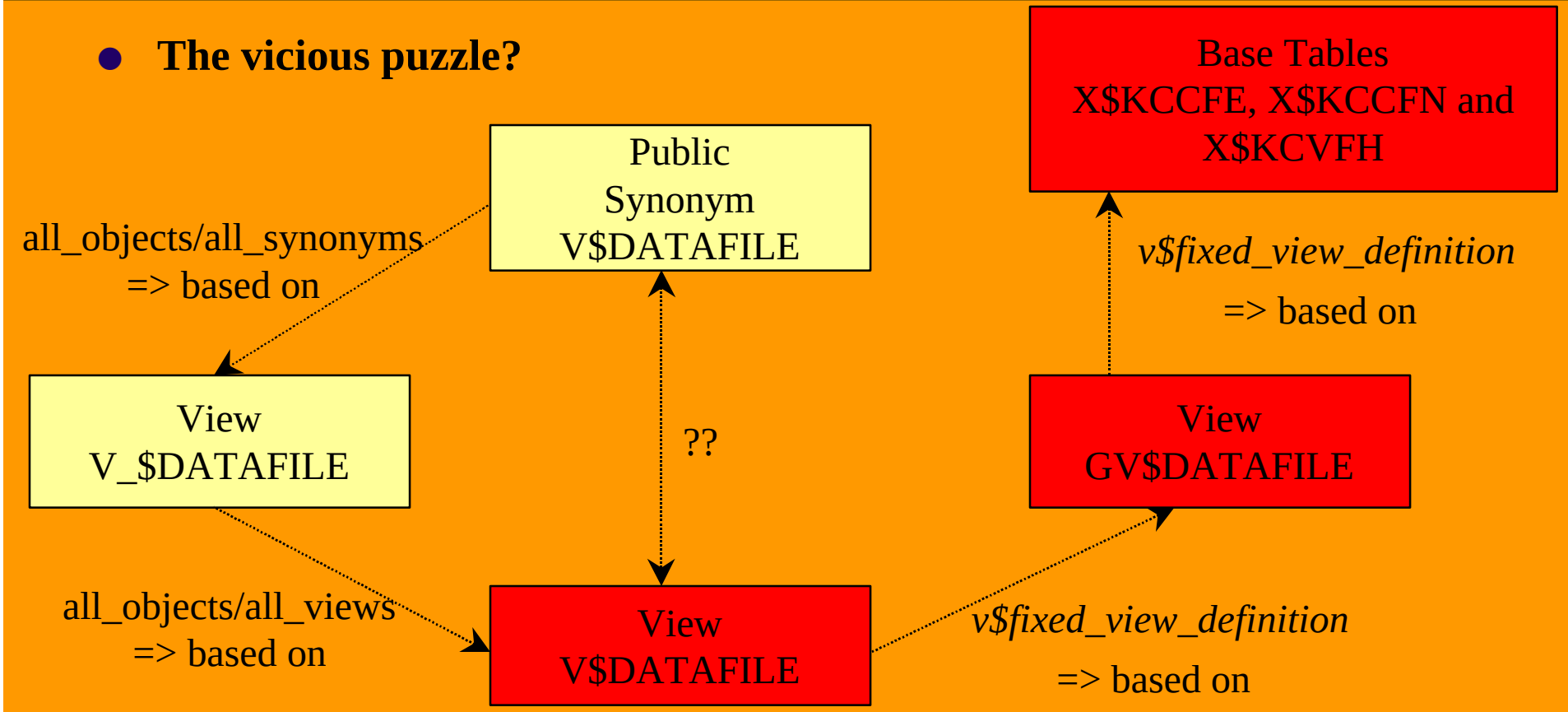
Data Dictionary Views

- **Dynamic Performance views.**
  - Dynamic performance tables (X$ tables) consume space in the System Global Area (SGA)
  - The X$ tables and their related V$ views are created in memory each time a database instance is started.
  - Oracle hides the complex X$ table queries (script from catalog.sql)
    1. Creates a view say v$datafile from a complex query.
    2. *create or replace view v_$datafile as select * from v$datafile;*
    3. *drop public synonym v$datafile;*
    4. *create public synonym v$datafile for v_$datafile;*
  - V$-view-creation statement is stored in the v$fixed_view_definition view.

```
select view_name, view_definition
from v$fixed_view_definition
order by view_name;
```

Data Dictionary Views

- **The vicious puzzle?**



Data Dictionary Views

- **And the complex SQL that was hidden is…**

```
[create view gv$datafile as]
select fe.inst_id, fe.fenum, to_number(fe.fecrc_scn),
    to_date(fe.fecrc_tim,'MM/DD/RR HH24:MI:SS'), fe.fetsn, fe.ferfn,
    decode(fe.fetsn,0,decode(bitand(fe.festa,2),0,'SYSOFF','SYSTEM'),
    decode(bitand(fe.festa,18),0,'OFFLINE',2,'ONLINE','RECOVER')),
    decode(bitand(fe.festa,12),0,'DISABLED',4,'READONLY',12,'READ WRITE','UNKNOWN'),
    to_number(fe.fecps),to_date(fe.fecpt,'MM/DD/RR HH24:MI:SS'),
    to_number(fe.feurs),to_date(fe.feurt,'MM/DD/RR HH24:MI:SS'), to_number(fe.fests),
    decode(fe.fests,NULL,to_date(NULL),to_date(fe.festt,'MM/DD/RR HH24:MI:SS')),
    to_number(fe.feofs),to_number(fe.feonc_scn),
    to_date(fe.feonc_tim,'MM/DD/RR HH24:MI:SS'), fh.fhfsz*fe.febsz, fh.fhfsz,
    fe.fecsz*fe.febsz, fe.febsz, fn.fnnam
from x$kccfe fe, x$kccfn fn, x$kcvfh fh
where fn.fnfno=fe.fenum and fn.fnfno=fh.hxfil and fe.fefnh=fn.fnnum and fe.fedup!=0 and
    fn.fntyp=4 and fn.fnnam is not null;
```

Data Dictionary Views

- **Changes**
  - The number of V$ views increased from 97 to 140 from Release 7.3.2 to Release 8.0.4
  - The number of X$ tables increased from 158 to 208.
  - Oracle added a new set of V$ views in Oracle8, the GV$ views -- these contain an additional column for the Instance ID (inst_id)

View Advantages

- **Security**
- **Query Simplification**
- **Allows Different Perspective**
- **Schema Transparency / Location Transparency**
- **Schema Consistency**
- **Allows work-around for SQL limitations**

View Advantages

- **Security**
  - to provide an additional level of table security by restricting access to a predetermined set of rows and/or columns of a table

```
CREATE VIEW emp_sal_hist_v
AS
SELECT  ratehist.employee, ratehist.beg_date, ratehist.pay_rate
FROM    ratehist, employee
WHERE   ratehist.company = employee.company
AND     ratehist.employee = employee.employee
AND     USER = employee.user_id;
```

View Advantages

- **Query Simplification**

For example, a single view might be defined with a join, which is a collection of related columns or rows in multiple tables. However, the view *hides* the fact that this information actually originates from several tables.

Saving of complex queries also permits *simplified commands* for an end-user who does not know how to make joins and/or cryptic business rules governing a join.

View Advantages

- **Allows Different Perspective**
  - For example, the columns of a view can be renamed without affecting the tables on which the view is based.
  - Columns cannot be dropped from tables in version 7.x, but you could recreate views without the unnecessary column.

View Advantages

- **Schema Transparency / Location Transparency**
  - Ability to hide the schema of data from the application, and therefore the user.
  - For example, if a view's defining query references three columns of a four column table and a fifth column is added to the table, the view's definition is not affected and all applications using the view are not affected.
  - Views can also be used to join tables across database schemas OR across databases (using remote links), thereby encapsulating schema names from the end user.

View Advantages

- **Schema Consistency**
  - If a web application is accessing legacy data and then we migrate over to a new system.
  - Identify legacy tables accessed through the web.
  - Create a view look-alike for each legacy table and have it return the same data.
  - Though not a long-term solution, will allow intermediate means of allowing the web application to run while the interface is rebuilt to the new system.

View Advantages

- **Allows work-around for SQL limitations**
  - A view can be defined that joins a GROUP BY view with a table.
  - A view can be defined that joins a UNION view with a table.

Materialized Views

- **Materialized Views**
  - **Much like a snapshot (in fact, snapshots are now called Materialized Views)**
  - **Refresh is either time/demand based or Real Time**
  - **Supports**
    - **Materialized Join Views (MJV)**
      - equi or outer join of many tables without aggregates
    - **Materialized Aggregate Views (MAV)**
      - may include joins
    - **Materialized Subquery Views (MSV)**
      - Tables ‘joined’ by “EXISTS” and primary keys
    - **Materialized views of Materialized Views**

Materialized Views

```
create materialized view SALES_MONTH_MV
    tablespace AGG_DATA
    refresh complete
    start with sysdate
    next sysdate+1
    enable query rewrite
    as
    select Sales_Month, SUM(Amount)
      from SALES
     group by Sales_Month;
```

New for Oracle8i

- create view as select … order by

```
create view emp_view
as  select ename, sal
from emp
order by sal;

SQL> select * from emp_view;
ENAME      SAL
----------      ---------
KING         5000
SCOTT       3000
FORD         3000
JONES        2975
BLAKE       2850…..

14 rows selected.
```

Where do we go from here?

- Oracle Manuals
- Oracle Underground Frequently Asked Questions
  (www.orafaq.org)
- Oracle Technology Network *(technet.oracle.com)*
- OraPub of Earth *(www.orapub.com/)*
- Oracle Source Code
  (www.osborne.com/oracle/source.htm)
- Oracle NewsGroups
  (comp.databases.oracle.[server/tools/misc])
- Capital District Oracle User Group *(www.cdoug.org)*

Thank you !!!!!

References :

- *Oracle 8 Server Administrator's Guide, Release 8.0*. Primary Author: Joyce Fee. Oracle Corporation, 1997.
- *Oracle 8 Server SQL Reference, Release 8.0*. Primary Author: Denise Oertel. Oracle Corporation, 1997.
- *Oracle 8 Server Concepts, Release 8.0*. Primary Authors: Lefty Leverenz, Richard Mateosian, Steve Bobrowski. Oracle Corporation, 1997
- "Securing Web Access to Database Applications with Views", Beth S. Wolfset, *ECO '99 Conference Proceedings*, March 1999.
- "The V$ Views: Critical Knowledgebase for DBA's", Joe Trezzo, Oracle Magazine, March 1999.
- "Oracle 8i New Features", John Dybas, *CDOUG*, June 1999