Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea de Calculatoare, Informatică și Microelectronică
Filiera Anglofonă "Computer Science"

Views in SQL

Course Thesis

Baze de Date și Cunoștințe

Student: Vlas Mihai, FAF-141

Conducător: Irina Cojanu

Chișinău 2017

Contents

Introduction

SQL stands for Structured Query Language, is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system. Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

Clauses, which are constituent components of statements and queries. (In some cases, these are optional.) Expressions, which can produce either scalar values, or tables consisting of columns and rows of data Predicates, which specify conditions that can be evaluated to SQL three-valued logic (3VL) (true/false/unknown) or Boolean truth values and are used to limit the effects of statements and queries, or to change program flow. Queries, which retrieve the data based on specific criteria. This is an important element of SQL. Statements, which may have a persistent effect on schemata and data, or may control transactions, program flow, connections, sessions, or diagnostics. SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar. Insignificant whitespace is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

1 Views in SQL

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.
Views, which are a type of virtual tables allow users to do the following:
• Structure data in a way that users or classes of users find natural or intuitive.
• Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
• Summarize data from various tables which can be used to generate reports.

Besides the standard role of basic user-defined views, SQL Server provides the following types of views that serve special purposes in a database:
• Indexed Views: An indexed view is a view that has been materialized. This means the view definition has been computed and the resulting data stored just like a table. You index a view by creating a unique clustered index on it. Indexed views can dramatically improve the performance of some types of queries. Indexed views work best for queries that aggregate many rows. They are not well-suited for underlying data sets that are frequently updated.
• Partitioned Views: A partitioned view joins horizontally partitioned data from a set of member tables across one or more servers. This makes the data appear as if from one table. A view that joins member tables on the same instance of SQL Server is a local partitioned view.

• System Views: System views expose catalog metadata. You can use system views to return information about the instance of SQL Server or the objects defined in the instance. For example, you can query the sys.databases catalog view to return information about the user-defined databases available in the instance.

There are several actions that we can do with views in SQL:

1) **Create View**
2) **Update View**
3) **Delete View**

First of all let's analyze how the views are created and what kind of syntax we need for it. Database views are created using the CREATE VIEW statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Listing 1.1 – Views creating statements

Let's take an example of creating a view from a table:

Now let's create the view that will have only Customer's name and age:

Table 1.1 – Customers

ID	NAME	AGE	ADDRESS	SALARY
1	ION	25	Chisinau	3000
2	Victor	30	Orhei	2000
3	Mihai	20	Balti	4000

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM  CUSTOMERS;
```

Listing 1.2 – Creating view from a table

The result will be the following:

Table 1.2 – CustomersView

NAME	AGE
ION	25
Victor	30
Mihai	20

Next option for Views let's analyze the updating action.
First of all to update a view we need to acomplish several conditions, which are:
• The SELECT clause may not contain the keyword DISTINCT.
• The SELECT clause may not contain summary functions.
• The SELECT clause may not contain set functions.
• The SELECT clause may not contain set operators.
• The SELECT clause may not contain an ORDER BY clause.
• The FROM clause may not contain multiple tables.
• The WHERE clause may not contain subqueries.
• The query may not contain GROUP BY or HAVING.
• Calculated columns may not be updated.
• All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then it can be updated. The following listing has an example to update the age of Victor.

```
SQL > UPDATE CUSTOMERS_VIEW
   SET AGE = 35
   WHERE name = 'Victor';
```

Listing 1.3– Updating a View

Table 1.3– UpdatedView

NAME	AGE
ION	25
Victor	35
Mihai	20

The last operation is Deleting a View, which we can perform by dropping it if it's no longer needed.

```
DROP VIEW view_name;
```

Listing 1.4– Drop View

For our example, the sequence will be:

```
DROP VIEW CUSTOMERS_VIEW;
```

Listing 1.5– Drop CustomerView

Treating Views as simple tables allows us to insert new rows and delete some specific rows or columns from the view. This can be done with INSERT/DELETE command.

```
SQL > DELETE FROM CUSTOMERS_VIEW
   WHERE age = 20;
```

Listing 1.6– Drop CustomerView

Note!: For INSERT Command, are applied the same rules as for UPDATE command.
Let's delete a row which has recorded the age=20.

Table 1.4– UpdatedView

NAME	AGE
ION	25
Victor	35

2 Views in Oracle 12c

Oracle presents a very similar way to treats views in 12c, we have the same functions for creating, updating and deleting a view, but it comes with an addition of different parameters specification. OR REPLACE, INSTEAD OF, UNUSABLE are just a small part of triggers defined on a conventional view.
To create a View in PL/SQL we use the same implementation as we did with TSQL, using CREATE VIEW command:

```
CREATE OR REPLACE VIEW <view_name> AS
SELECT <column_name>
FROM <table_name>;
```

Listing 2.1– Views creating statements

Different views are created with different purposes, based on what are our needs for the specific task, table or interaction with the database.
We can create a single table view or or multi-table view, with functions in the WHERE or SELECT clause, but much more of that, we can create Force Views and Parameterized Views.

Force View allows us to create a view even if it is invalid.

```
CREATE OR REPLACE VIEW <view_name> AS
SELECT <column_name>
FROM <table_name>;
```

Listing 2.2– Views creating statements

Parameterized View are a type of views with a filter or function in WHERE clause or SELECT clause.

```
CREATE OR REPLACE VIEW <view_name> AS
SELECT <column_name>
FROM <table_name>;
```

Listing 2.3– Views creating statements

Oracle allows us to do such options as altering a view and commenting views or column views. It can be done easily with ALTER VIEW ¡view name¿ COMPILE and COMMENT ON TABLE ¡table name¿ IS '¡comment string¿';
With **View Update** there are also tricks and a set of restrictions that a view can not include in order to be updated. Such constraints are: • Set Operators (INTERSECT, MINUS, UNION, UNION ALL) • DISTINCT • Group Aggregate Functions (AVG, COUNT, MAX, MIN, SUM, etc.) • GROUP BY Clause • ORDER BY Clause • CONNECT BY Clause • START WITH Clause • Collection Expression In A Select List • Subquery In A Select List • Join Query

In order to update a view we have to take into account those constraints at the stage of view

creation and try to complete them in order to update the view later.

Deleting a view in Oracle 12c is made the same as in Microsoft SQL Server, with Drop operation.

```
1 DROP VIEW view_name;
```

Listing 2.4– Views creating statements

Interesting thing about views is that even if the base table was deleted the view will continue to exist however if we will try to query the Oracle VIEW we will receive a message indicating that the Oracle VIEW has errors.

Materialized Views:
Originally called snapshots, materialized views were introduced in Oracle8i and are only available in the Enterprise Edition. Like a regular view, the data in a materialized view results from a query. However, the results of a regular view are transitory—they are lost once the query is complete and, if needed again, the query must be re-executed. In contrast, the results from a materialized view are kept and physically stored in a database object that resembles a table. This feature means that the underlying query only needs to be executed once and then the results are available to all who need them.

Oracle Database 12c allows for synchronous refreshes of the materialized views when configured to use a refresh method besides manual or on-demand. It utilizes partitioning and dependencies between the objects to minimize the time it takes to refresh and maintain the data as close to the underlying tables as possible. Materialized views are used as a performance-enhancing technique. In this section, you learn about the following uses of these views, as they are applicable to the topic of large databases.

Performing data summarization (for example, sums and averages) Prejoining tables Performing CPU-intensive calculations Replicating and distributing data In large databases, particularly data warehousing environments, there is always a need to summarize, join, perform calculations, or do all three operations at once on large numbers of records for the purposes of reporting and analysis. To improve performance in the past, a combination of views and physical tables were usually implemented that contained the results of these operations. The summary tables would require some type of extraction, transformation, and load (ETL) process to populate and refresh them. In addition to the base tables containing the detailed data, the users would need to know which combinations of the views and/or summary tables to use.

Using materialized views has several advantages over more traditional methods. These include the following:

• Materialized views have a built-in data refresh process, which can provide an automatic update or repopulation of a materialized view without any programming on the part of the DBA.
• As mentioned earlier, the data in materialized views can be partitioned, using the same techniques that apply to tables.

• Materialized views are transparent to the users. This is probably the most attractive feature of using materialized views. We expand more on this in the next section when we discuss automatic query rewriting.

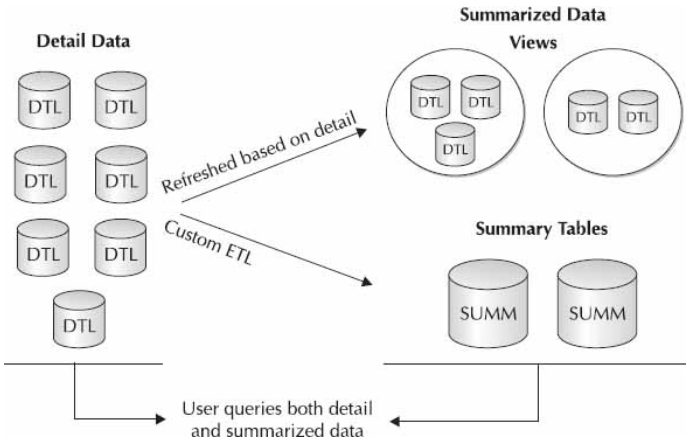

Figure 2.1– Summarization using views and summary tables

At this point, you may be asking yourself: "How do I determine what materialized views to create

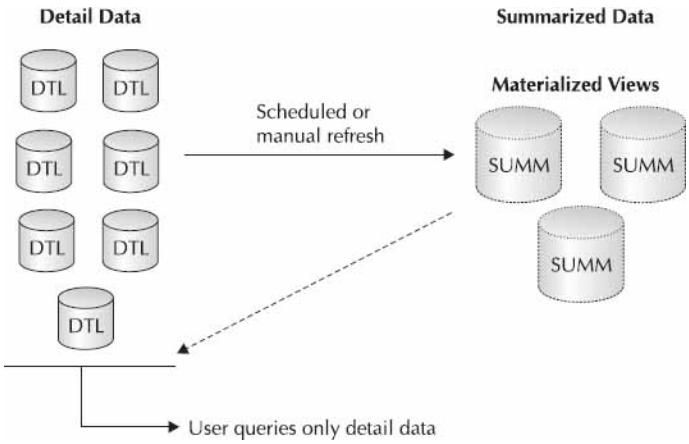

Figure 2.2– Summarization using materialized views

and at what level of summarization?" Oracle Database 12c has some utilities to help. These utilities are collectively called the SQL Tuning Advisor and will recommend materialized views based on historical queries, or based on theoretical scenarios. They can be run from the Oracle Enterprise Manager (OEM) Grid Control, or by calling the dbms advisor package.

3 Practical part

```
1 CREATE OR REPLACE VIEW <view_name> AS
2 SELECT <column_name>
3 FROM <table_name>;
```

Listing 3.1– Views creating statements

For practical part, I will go step by step on this topic, will create the view, update it and then drop it in the end. Views can be used to reduce the complexity of the database schema, used as a mechanism to implement row and column level security and can be used to present aggregated data and hide detailed data.

To begin, let's define 2 tables with data, from whom we will create our views.

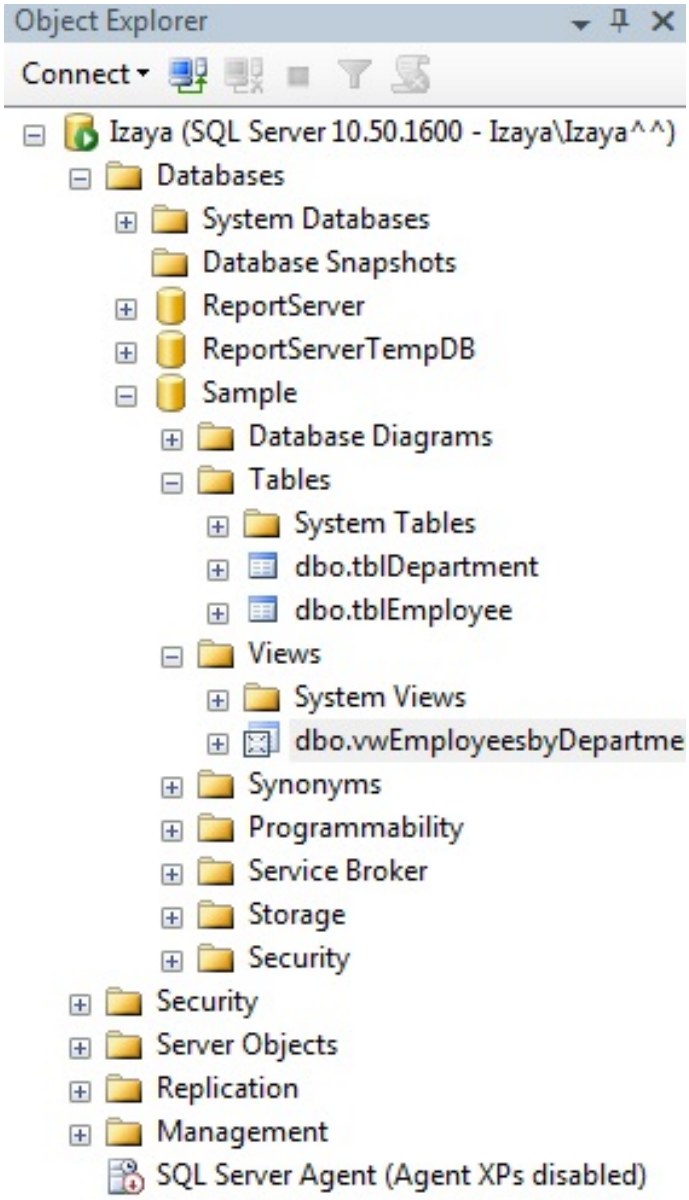

Figure 3.1– Project's Objects Explorer

As we can observe, we have 2 tables: - tblDepartment (containing names of departments and their IDs) - tblEmployee (containing all the information about the employees)

Employee table will have 5 columns (ID, Name, Salary, Gender, DepartmentName) and we will operate first of all by merging two tables, after this we will create a view. (note: by convention it is a good experience to start view names with vW)

Figure 3.2– Creating View Query

Views simplify our work presenting the database for non-IT users, making it a lot more easier to generate, to manage accesses or to hide confidential data. For our two tables, I will try to make different views for different kind of requests we'll get.
Let's give a specific table for someone who wants to join IT department and want to know his future co-workers:

```
1  Create view vwITEmployees
2  as
3  SELECT Id, Name, Salary, Gender, DeptName
4  from tblEmployee
5  join tblDepartment
6  on tblEmployee.DepartmentId=tblDepartment.DeptId
7  where tblDepartment.DeptName='IT'
8
9  GO
10 Select * from vwITEmployees
```

Listing 3.2– Views creating statements

So that's how the query will look like and the result will be this:

	Id	Name	Salary	Gender	DeptName
1	2	Mihai	4000	Male	IT
2	4	Grigore	4500	Male	IT

Figure 3.3– ITDepartment Query Result

In the Object Explorer we can see how we slowly add views in the Views Folder to the Sample Database that we are working in.

Let's simulate the situation when we are publishing the database to all the employees or to third party. This is the case when some information from the table must be hide in order to protect status and behavior of the person. For such a cases Views are a powerful tool that allows us to create a view from a table which will not contain the disturbing or personal information. This is shown

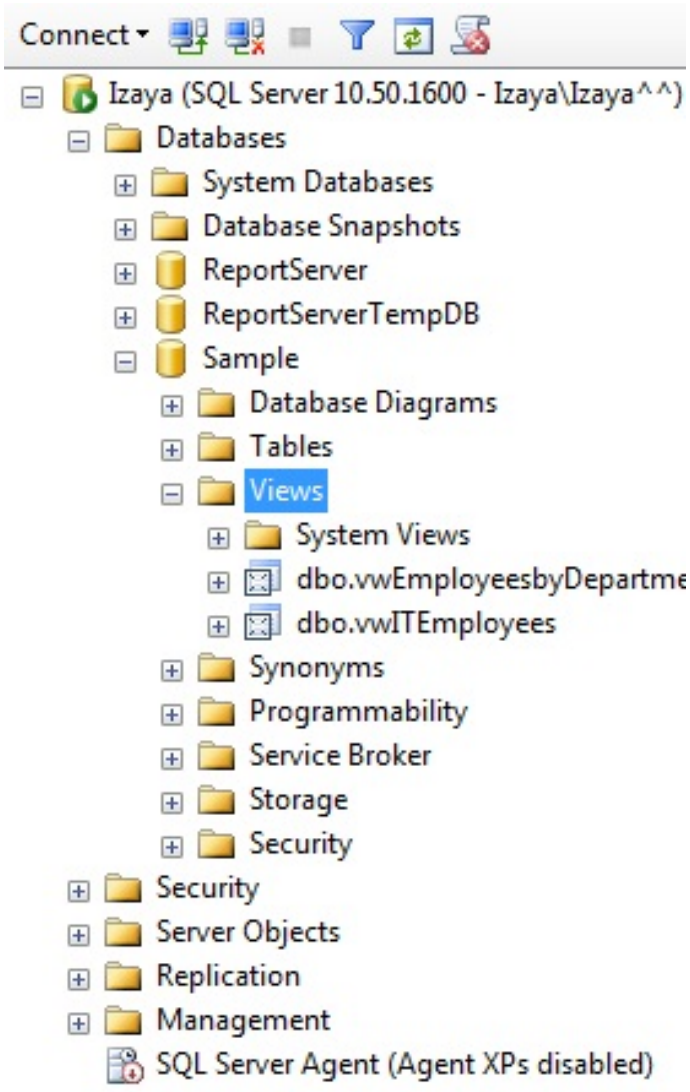

Figure 3.4– ITDepartment Query Result

below:

As we can see, with a simple query I eliminated the column in which where the Salary of the

	Id	Name	Gender	DeptName
1	1	Ion	Male	Marketing
2	2	Mihai	Male	IT
3	3	Maria	Female	HR
4	4	Grigore	Male	IT
5	5	Victor	Male	Management

Figure 3.5– ITDepartment Query Result

employees by creating a view without this column, so the main table still holds this info, but for public eyes it is hiden.

```
1 Create view vwNonConfidentialEmployees
```

```
as
SELECT Id, Name, Gender, DeptName
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId=tblDepartment.DeptId

GO
Select * from vwNonConfidentialEmployees
```

Listing 3.3– Views creating statements

Update for a view is done simply with clause Update and Set. In the following example I will change the name of a specific employee with a specific ID using Update clause.

	Id	Name	Gender	DeptName
1	1	Ion	Male	Marketing
2	2	Michael	Male	IT
3	3	Maria	Female	HR
4	4	Grigore	Male	IT
5	5	Victor	Male	Management

Figure 3.6– Updated View

Let's not forget that Views are also tables, and we can do things like deleting from a row or specific element, or to insert in a specific place our new data if it's free to insert.

Deleting a view is done by the command Drop, which performs in the following way:

```
Drop view Employees
```

Listing 3.4– Views creating statements

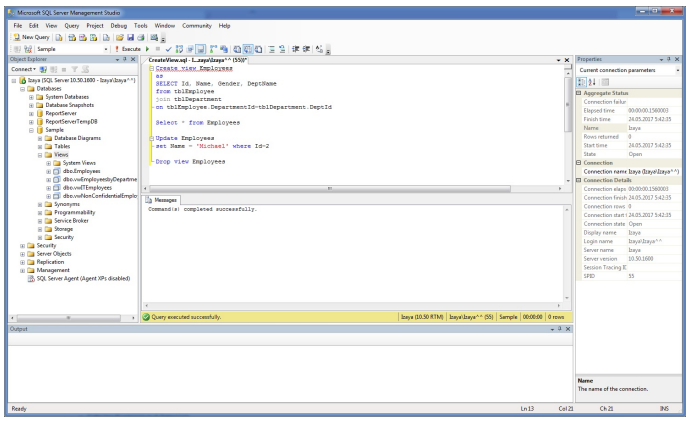

Figure 3.7– Updated View

After we execute the DROP command we should see the elimination of the view in the Views folder of the project.

In such a simple way Views can facilitate work using simple query requests.

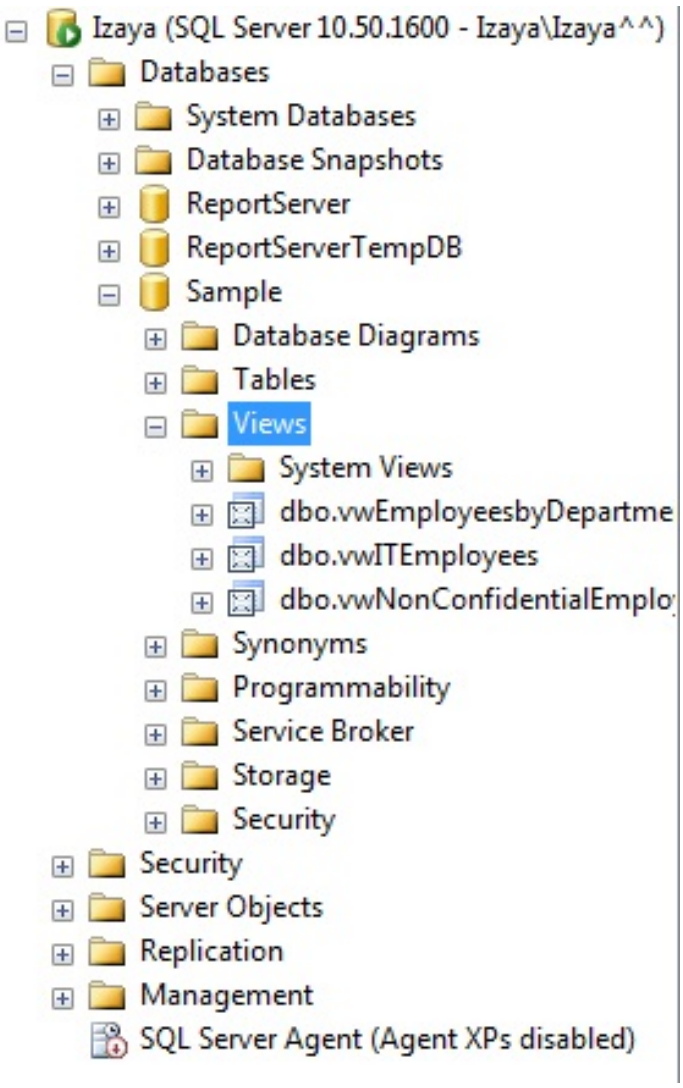

Figure 3.8– Updated View

Conclusions

Views in SQL are a powerful tool that can simplify the work of the database architect or developer. By combining multiple tables in one view, easily updating and deleting some rows is facilitating the work and makes it a lot easier than using typical update and insert commands directly in the table of work. Views are statements with a name associated to it and stored in the database, this allows an easy creation and drop of the view in any moment of work. Depending on the type of view we need to use, the specific attributes should be modified in views. Views are virtual tables that facilitate the work for an IT person using a simpler way to give information and it can be used in different ways. From my opinion, views has 3 main functions, to reduce the complexity of the database schema, to implement higher row and column levels of security and to present aggregated data and hide detailed data. The way how easy it is to create, update or delete a view we can exclaim that it is a tool that should be used more often and implemented on different databases.

References

1 LogicalRead *open source*, https://logicalread.com/materialized-views-improve-oracle-dat

2 Oracle-Base , https://oracle-base.com/articles/12c/control-invoker-rights-privileges-

3 Official Oracle Documentation, https://docs.oracle.com/database/121/SQLRF/statements_8004.htm#SQLRF01504

4 Kudvenkat, *youtube channel*, https://www.youtube.com/watch?v=VQpmOmZO2mo

5 Kudvenkat, *Open Source*, http://www.morganslibrary.org/reference/views.html