



*The World's Largest Community
of SQL Server Professionals*

How and When to Implement Indexed Views

Brad M McGehee

Director of DBA Education

[Red Gate Software](#)

www.bradmcgehee.com

My Assumptions About You

- You are probably a DBA (production or developer) who has at least one year's experience using SQL Server.
- You have a basic understanding of how to write T-SQL code.
- You have probably not used indexed views in the past.

What We Are Going to Learn Today

- What is an Indexed View
- Pros and Cons of Using Indexed Views
- How to Create Indexed Views
- How to Use Indexed Views
- Indexed Views Best Practices
- Comprehensive Demo

What We Are Going to Learn Today

- What is an Indexed View
- Pros and Cons of Using Indexed Views
- How to Create and Use Indexed Views
- Indexed Views Best Practices

Types of Views

- SQL Server offers three different types of views:
 - **Standard Views:** Essentially a query that is packaged so that it can easily be reused. When the view is run, the underlying query is expanded and executed. You can also think of a standard view as **virtual table** because the data in the view does not exist until the view is executed.
 - **Partitioned Views:** Used to join horizontally partitioned data from a set of tables across one or more servers, making the data appear as if it comes from a single table.
 - **Indexed Views:** The focus of today's session, and defined on the next slide. (Sometimes known as materialized views.)

What is an Indexed View

- Essentially, an indexed view is a view that has been materialized (made physical) by adding an unique clustered index on it.
- An indexed view is stored in a database similar to a table.
- Whenever the data in the underlying tables is modified, the indexed view is also modified, so it is always current.
- If desired, an indexed view can also have non-clustered indexes.
- *Note: To fully take advantage of indexed views, you need to have the Enterprise Edition of SQL Server. Indexed view work slightly differently in 2000, 2005, and 2008.*

How Does the Query Optimizer Treat Indexed Views

- If an indexed view is **part of the FROM clause**, the Query Optimizer expands the view's definition, just as it does for all views.
- If the indexed view is **not part of the FROM clause**, then it is considered, just as any index is considered by the Query Optimizer. (EE only).
- As the query is optimized, the Query Optimizer decides if using the available indexed view (**named or not named**) is beneficial, and chooses the execution plan that has the least cost.
- This may result in a **query using an indexed view even though it is not named** in the query, or it may mean that a **named indexed query may not be used** (with the base tables accessed directly).
- Once the Query Optimizer has decided to use an indexed view, it then **treats the view the same way it treats a base table** (as an indexed view is essentially a table with a clustered index).

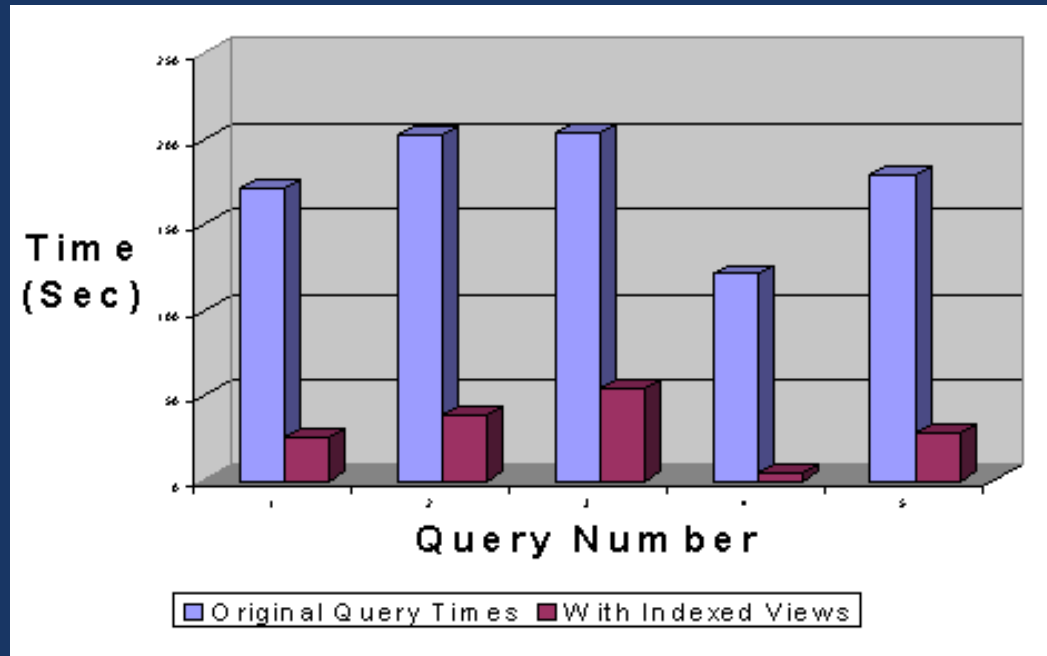
What We Are Going to Learn Today

- What is an Indexed View
- Pros and Cons of Using Indexed Views
- How to Create and Use Indexed Views
- Indexed Views Best Practices

Advantages of Using Indexed Views

- Because the data is physical, some queries can be greatly sped up by using indexed views because **much or all of the work has already been done** (pre-joined, pre-calculated, pre-aggregated), as the base tables don't need to be accessed and processed to return the requested data.
- Indexed views may be able to **replace the need** to use triggers or other techniques to create summarized/aggregated data.
- Indexed views can be added to virtually any database **without requiring the front-end application to be modified**.
- **Even if an indexed view is not explicitly named in a query**, it will be considered by the query optimizer, just like any. Think of it as a form of a covering index, or another clustered index. (EE only.)

Indexed Views Can Increase Performance



Microsoft test results. <http://msdn.microsoft.com/en-us/library/dd171921.aspx>

Disadvantages of Using Indexed Views #1

- Because indexed views are physical, they **occupy space**.
- Because indexed views are maintained when the data in their underlying tables are changed, some **overhead is required to maintain an indexed view**.
- Following the above logic, the use of indexed views can contribute to **blocking issues** if the multiple base tables used in an indexed view are being updated at the same time.
- While **indexed views can be created in any SQL Server edition**, it is only the Enterprise Edition where the Query Optimizer automatically **considers an indexed view** as part of the execution plan when it is not named in the query.
- In other editions, to use an indexed view, you must use the **NOEXPAND** table hint, along with naming it in the query.

Disadvantages of Using Indexed Views #2

- Before the Query Optimizer can use an indexed view, it must meet a lot of **stringent requirements**. These will be discussed later when we learn how to create an indexed view.
- In many cases, because of these requirements, **you won't be able to create** or use an indexed view you would like to use.

When Should Indexed Views be Considered

- For queries with **joins** that process **huge** numbers of rows.
- For queries with expensive **joins** that are used **frequently**.
- For queries with **aggregations** that process **huge** numbers of rows.
- For queries with expensive **aggregations** that are used **frequently**.
- If you have a **very wide table**, consider using an indexed view to vertically partition the data to speed up some queries, assuming that there are a lot of queries that need to only access this subset of columns.
- For **decision support-based** systems with mostly read-only data.

When Should Indexed Views Not be Considered

- Don't use for OLTP systems where underlying tables are frequently modified.
- Don't use for queries that don't include joins or aggregations. The overhead of maintaining indexed views in these cases will probably outweigh their benefits. An exception is when you use an indexed view on wide rows for vertical partitioning.
- Don't replace a regular index with an indexed view if the regular index works fine.

What We Are Going to Learn Today

- What is an Indexed View
- Pros and Cons of Using Indexed Views
- How to Create and Use Indexed Views
- Indexed Views Best Practices

How to Create and Use Indexed Views

- **Creating Indexed Views**
 - Creating an Indexed View Is a Two-Step Process
 - Prerequisites for Creating Indexed Views
 - How to Create an Indexed View
- **Executing Indexed View**
 - Prerequisites for Executing an Indexed View
 - How to Execute an Indexed View
 - Using the NOEXPAND Hint with Indexed Views

Creating an Indexed View Is a Two-Step Process

- First, you must create a standard view using:
 - `CREATE VIEW`
- Second, you must add a clustered index to a view using:
 - `CREATE UNIQUE CLUSTERED INDEX`
- Before you can do either of the above steps, you first must ensure that the prerequisites for both the `CREATE VIEW` and `CREATE UNIQUE CLUSTERED INDEX` are in place, which are discussed next.

Prerequisites for CREATE VIEW #1

- ANSI_NULLS and QUOTED_IDENTIFIER options must be set to ON.
- All tables referenced by the indexed view must have been created with the ANSI_NULLS option ON.
- An indexed view can only reference tables, not other views.
- Indexed views must be created with the SCHEMABINDING option.
- Any user-defined functions referenced by the view must be created with the SCHEMABINDING option ON.
- Objects referenced in the view must use the two-part name; not one or three or four.
- Any functions referenced in the view must be deterministic.
- Tables included in an indexed view must come from the same database.
- If the indexed view includes an aggregate function, the SELECT list must include COUNT_BIG(*).
- The data access property of a user-defined function must be set to NO SQL and the external access property set to NO.
- CLR functions can be included in the SELECT list, but they cannot be a part of the definition of the clustered index key, nor can they be included in the WHERE clause, or the ON clause of a JOIN.
- CLR functions also must have these properties set: DETERMINISTIC = TRUE, PRECISE = TRUE, DATA ACCESS = NO SQL, EXTERNAL ACCESS = NO.

Prerequisites for CREATE VIEW #2

- SELECT statements cannot include the following:
 - SELECT *
 - Column names can't be repeated.
 - A derived table.
 - A common table expression (CTE).
 - Rowset functions
 - UNION, EXCEPT and INTERSECT operators.
 - Subqueries
 - Outer or self-JOINs
 - TOP clause
 - ORDER BY or OVER clause
 - DISTINCT keyword
 - COUNT keyword
 - AVG, MAX, MIN, STDEV, STDEVP, VAR, or VARP
 - A SUM function that references a nullable expression.
 - CLR UDF aggregate function
 - COMPUTE or COMPUTE BY
 - CROSS APPLY or OUTER APPLY
 - PIVOT or UNPIVOT
 - JOIN hints
 - Direct references to Xquery
- Note: Not every prerequisite is listed on these slides.

Prerequisites for CREATE UNIQUE CLUSTERED INDEX

- When creating an indexed view, you must start by adding the unique clustered index first. Once that is done, you can add non-clustered indexes to an indexed view (if needed).
- The user creating the clustered index must own the view.
- These SET options must be ON when the CREATE INDEX statement is run:
 - ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS,
CONCAT_NULL_YIELDS_NULL, QUOTED_IDENTIFIER
- This SET option must be OFF: NUMERIC_ROUNDABORT
- The view cannot include text, ntext, or image columns.
- If the SELECT statement includes a GROUP BY clause, the key of the clustered index can only reference columns specified in the GROUP BY clause.

Creating the View

```
CREATE VIEW view_name  
WITH SCHEMABINDING  
AS  
  
    SELECT ...
```

Note: The SCHEMABINDING option ensures that the tables and objects included in the view definition won't change underneath the view.

Creating the UNIQUE CLUSTERED INDEX

```
CREATE UNIQUE CLUSTERED INDEX  
    index_name  
        ON view_name (column_name ...)
```

Prerequisites for Executing an Indexed View

- In order for an indexed view to be used by the Query Optimizer, **all of the following have to be meet:**
 - These SET options must be ON: ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL, QUOTED_IDENTIFIER.
 - NUMERIC_ROUNDABORT must be OFF.
- Every table referenced in the query (directly or via the indexed view) must have the same types of table hints (assuming hints are used).

Using the NOEXPAND Hint with Indexed Views

- As previously mentioned, unless you have the EE edition of SQL Server, indexed views won't be automatically used or considered for use by the Query Optimizer unless it includes the NOEXPAND hint.
- The NOEXPAND hint can only be used if the indexed view is explicitly named in the query. This means that indexed views not named in the query won't automatically be considered by the Query Optimizer if you don't use EE.
- If you use this hint, then you force the Query Optimizer to use it, whether it is a good choice or not.

```
SELECT * FROM indexed_view_name WITH (NOEXPAND)
```

Demo

- Let's see all of this in action.

What We Are Going to Learn Today

- What is an Indexed View
- Pros and Cons of Using Indexed Views
- How to Create and Use Indexed Views
- Indexed Views Best Practices

Indexed Views Best Practices #1

- To take full advantage of indexed views, you really need the **Enterprise Edition** of SQL Server.
- The underlying tables of indexed views should **not be frequently modified**, otherwise the overhead could be detrimental to performance. OLTP applications aren't indexed view friendly.
- If underlying tables of an indexed view are updated by a **batch** process, consider dropping the indexed view first, run the batch process, then re-add the indexed view. You will have to test to see if this benefits you (if it saves you time or not).
- If your query needs are complex, and can't be handled by a single indexed view, **you can create multiple indexed views**, and then query them within a single query.

Indexed Views Best Practices #2

- Be sure to test to see if an indexed view you have created really helps query performance. Test, don't assume.
- Keep the keys of indexed views as narrow as possible.
- Indexed views need maintenance just like any other index.
- Make sure your regular indexes and indexed views don't unnecessary overlap each other, making the standard indexes redundant.
- The DTA can be used to identify opportunities for identifying potential indexed views. Otherwise, choosing indexed views is a manual process.

Take Aways From This Session

- If you have not explored indexed views before, then take the time to learn more about them.
- Once you have mastered the basics, then experiment with some of your queries (in a test environment) to see if employing indexed views can help out your database applications.
- Indexed views are an often forgotten feature that could significantly speed up the performance of many queries.

Q & A

- Please ask your questions clearly and loudly.
- If you don't get your questions answered now, see me after the session, or e-mail me.

Find Out More

Free E-Books:

- www.sqlservercentral.com/Books

Check these out:

- www.SQLServerCentral.com
- www.simple-talk.com

Contact me at:

bradmcgehee@hotmail.com

Blogs:

www.bradmcgehee.com

www.twitter.com/bradmcgehee

[Click Here for a free 14-day trial of the Red Gate SQL Server Toolbelt](#)