SQL Query Optimization for Researchers

- As researcher analysts, we often need costly SQL queries.
  - Wide time ranges
  - Nationwide studies
  - Complicated cohort criteria

- In this presentation, we will talk about how to safely and efficiently approach heavy data needs, and we will troubleshoot some illustrative example queries.

Presented by Andrew Holbrook, VINCI Data Services
Assumptions

I assume:

- You know how to write basic to intermediate SQL queries.
- You know basic CDW architecture, e.g.
  - Dim vs Fact tables
  - Foreign keys
- You can ensure your queries return the correct results.
  - This CyberSeminar is focused on getting the same results faster.
The Plan

When we arrive here, we will have a solid foundation to which to attach these details.

We will start at the bottom with nontechnical explanations.
Meet Your Colleague

Strengths:
• Searching
• Sorting
• Syntax
• Arithmetic

Weaknesses:
• Common sense
• Context
• Matching datatypes
• Algebra
Query Plans

Click! This doesn’t run the query, it just asks your colleague what he’d do IF you ran the query.
How to Read a Query Plan

- Each operation passes rows to the next from right to left.
- Mouse over operations for more info.
- Watch for warnings like 🚨 and 😶

Thicker arrow, more rows

These icons represent operations

Thinner arrow, fewer rows
More Info from Query Plans

This is a running total of the cost

Mouse Over

Relative cost shows which query is more efficient
It was Just an Estimate!

Estimated

Actual
So Where Did Those Estimates Come From?

Table stats, dude
Entries are stored alphabetically, and volumes are labeled with end points.
Entries are stored by date, and partitions are labeled with end points.
Researchers must translate object names when using provisioned views.

- Partitioning, indices, and metadata all apply!
- Your execution plans will have an extra join to your cohort.

```
[CDWWork].[Outpat].[Visit]
```

```
[ORD_Holbrook_202202042D].[Src].[Outpat_Visit]
```
Now, Finally, Best Practices

- Don’t be greedy.
  - Select only the rows and columns you need, especially from fact tables.

Do you really need all these columns?

```
SELECT *
FROM
CDWWork.outpat.Visit
WHERE...
```

Do you really need all these rows?
Partition Elimination

- CDW fact tables are partitioned. In order to use partition elimination:
  - Determine which column is used
  - Check the datatype for that column
  - Use that column and datatype in your WHERE clause

```sql
SELECT DWPartitionKey, DWPartitionKeyDataDataType
FROM cdwwork.meta.DWView
WHERE DWViewName = 'appointment'
```

```sql
SELECT apt.AppointmentSID
FROM CDWWork.Appt.Appointment apt
WHERE apt.AppointmentDateTime > convert(datetime2(0), '2021-01-01')
```
Did It Work?

```
Columnstore Index Scan (Clustered)
Cost: 69%

Columnstore Index Scan (Clustered)
Scan a columnstore index, entirely or only a range.

Physical Operation  | Columnstore Index Scan
Logical Operation   | Clustered Index Scan
Estimated Execution Mode | Batch
Storage             | ColumnStore
Estimated Operator Cost  | 20.4391 (60%)
Estimated I/O Cost       | 15.3258
Estimated Subtree Cost   | 20.4391
Estimated CPU Cost       | 5.11331
Estimated Number of Executions | 1
Estimated Number of Rows to be Read | 46434600
Estimated Number of Rows | 37205500
Estimated Row Size       | 22 B
Partitioned             | True
Ordered                 | True
Node ID                 | 1

Predicate
((CDW12.[Appt].[Appointment_v192].[OpCode] as [a].[OpCode]) <'X'
 AND (CDW12.[Appt].[Appointment_v192].[AppointmentDateTime] as [a].[AppointmentDateTime] > '2021-01-01 00:00:00')

Object
(CDW12.[Appt].[Appointment_v192].[cci_Appointment_v192] as [a])
Output List
(CDW12.[Appt].[Appointment_v192].[AppointmentSID], CDW12.[Appt].[Appointment_v192].[OpCode]

Seek Predicates
Seek Key([1]: Start: PhnId1001 >= Scalar Operator('87'), End: PhnId1001 <= Scalar Operator(122))
```
Use Temp Dims

- Build yourself a temporary dimension with the SIDs you need.
  - Searching with wildcards is fine, because you’re only using the dimension.

- Don’t join to the fact table until you know what you’re looking for.
  - In general, wildcard searches on fact tables should be avoided.

- Similarly, you can use temp tables for patient cohorts.
SELECT diag.PatientSID
FROM CDWWork.outpat.VDiagnosis diag
JOIN CDWWork.dim.ICD10 icd
    ON diag.ICD10SID = icd.ICD10SID
WHERE diag.Sta3n = 523
    AND diag.VisitDateTime > convert(datetime2(0), '2021-01-01')
    AND icd.ICD10Code LIKE 'B18%'

SELECT icd10SID
INTO #ICD_Hep
FROM CDWWork.dim.ICD10 icd
WHERE icd.ICD10Code LIKE 'B18%'

SELECT diag.PatientSID
FROM CDWWork.outpat.VDiagnosis diag
JOIN #ICD_Hep icd
    ON diag.ICD10SID = icd.ICD10SID
WHERE diag.Sta3n = 523
    AND diag.VisitDateTime > convert(datetime2(0), '2021-01-01')
Multiple Fact Tables In One Query

- Joining multiple fact tables is not usually recommended.
  - Typically, it is better to refine a cohort over multiple queries.
- If you do use multiple fact tables in one query, remember to use the partition key for each one.
Use the Partition Key For Each Fact Table

select diag.PatientSID, prov.ProviderSID from CDWWork.outpat.VDiagnosis diag
join #ICD_Hep icd
  on diag.ICD10SID = icd.ICD10SID
join CDWWork.Outpat.VProvider prov
  on diag.VisitSID = prov.VisitSID
where diag.Sta3n = 523
  and diag.VisitDateTime > convert(datetime2(0), '2021-01-01')

select diag.PatientSID, prov.ProviderSID from CDWWork.outpat.VDiagnosis diag
join CDWWork.Outpat.VProvider prov
  on diag.VisitSID = prov.VisitSID
join #ICD_Hep icd
  on diag.ICD10SID = icd.ICD10SID
where diag.Sta3n = 523
  and diag.VisitDateTime > convert(datetime2(0), '2021-01-01')
  and prov.VisitDateTime > convert(datetime2(0), '2021-01-01')

Does this change the results? Nope! But it sure changes the performance.
Functions

- Avoid using functions on columns in WHERE or JOIN clauses.
  - In the SELECT clause is fine.
- Comparing a column to a function output is the correct way.
- Treat math like a function.
- Bottom line: Columns should be by themselves on one side of the operator in the WHERE clause if at all possible.
Functions on Columns

- When comparing columns, you will sometimes need some logic applied to one. In this case:
  - Leave fact table columns as-is, especially the partition column.
  - Pre-filter and pre-calculate in a CTE or temp table.
Column Function Example

Note the column inside a function in the SELECT clause. This is fine! Just don’t do it in a JOIN or WHERE.

```
select PatientSID,
FROM CDWWork.Outpat.Visit vis
WHERE year(vis.VisitDateTime) = '2020'
and Sta3n = 523
```

```
select PatientSID, year(vis.visitdatetime) as VisitYear
into #temp
From CDWWork.Outpat.Visit vis
WHERE vis.VisitDateTime >= convert(datetime2(0), '2020-01-01')
and vis.VisitDateTime < convert(datetime2(0), '2021-01-01')
and Sta3n = 523
```
Functions

- Especially avoid functions like `DATEDIFF()` with multiple columns used as inputs. You should split the columns so they are on opposite sides of the operator.
Date Function Example

SELECT PatientSID
FROM CDWWork.Appt.Appointment apt
WHERE datediff(month, apt.AppointmentDateTime, getdate()) between 0 and 1

Also, this yields the wrong result

SELECT PatientSID
FROM CDWWork.Appt.Appointment apt
WHERE apt.AppointmentDateTime >= convert(datetime2(0), dateadd(month, -1, getdate())
AND apt.AppointmentDateTime < convert(datetime2(0), getdate())
Leave That Column Alone

\[
\text{select } p.\text{PatientICN}, \ p.\text{PatientSID} \\
\text{from cdwwork.Patient.Patient } p \\
\text{where } \text{Sta3n + 1 = 661}
\]

I just really want the column by itself, ok?

\[
\text{select } p.\text{PatientICN}, \ p.\text{PatientSID} \\
\text{from cdwwork.Patient.Patient } p \\
\text{where } \text{Sta3n = 660}
\]
Functions

Don’t worry about functions and math getting complicated. Just make sure it’s isolated from the column.

I’m good at arithmetic
Your Colleague Is Helpful

SELECT PatientSID
FROM CDWWork.Chem.LabChem lc
WHERE lc.LabChemCompleteDateTime >
    convert(datetime2(0), dateadd(day, (3+2)*-1, getdate()))

DECLARE @result datetime2(0) = (SELECT dateadd(day, (3+2)*-1, getdate()))

SELECT PatientSID
FROM CDWWork.Chem.LabChem lc
WHERE lc.LabChemCompleteDateTime > @result

SELECT PatientSID
FROM CDWWork.Chem.LabChem lc
WHERE lc.LabChemCompleteDateTime >
    convert(datetime2(0), '2021-03-12 10:54:14.253')
Repetition Legitimizes

- Just make sure functions and math are isolated from the columns!
What About NULLs in the Partition Field?

--fine query, but I want to include patients that have not been discharged

```sql
select i.InpatientSID, i.PatientSID, i.AdmitDateTime, i.DischargeDateTime
from cdwwork.Inpat.Inpatient i
where i.DischargeDateTime > convert(datetime2(0), '2022-05-01')
order by i.DischargeDateTime desc
```
What About NULLs in the Partition Field?

--no good, column inside function
--plus there are some weird old records where discharge never got filled in
select i.InpatientSID, i.PatientSID, i.AdmitDateTime, i.DischargeDateTime
from cdwork.Inpat.Inpatient i
where isnull(i.DischargeDateTime, getdate()) > convert(datetime2(0), '2022-05-01')
order by i.DischargeDateTime desc

--no good, this use of OR prevents SQL from using the partition key
--humans know discharges should be after admissions, but SQL doesn't know that
select i.InpatientSID, i.PatientSID, i.AdmitDateTime, i.DischargeDateTime
from cdwork.Inpat.Inpatient i
where (i.DischargeDateTime > convert(datetime2(0), '2022-05-01')
    or i.AdmitDateTime > convert(datetime2(0), '2022-05-01'))
order by i.DischargeDateTime desc

--this is the way
--explicitly tell SQL which partitions to look in for each set of conditions, either with a date range or NULL
select i.InpatientSID, i.PatientSID, i.AdmitDateTime, i.DischargeDateTime
from cdwork.Inpat.Inpatient i
where i.DischargeDateTime > convert(datetime2(0), '2022-05-01')
    or (i.DischargeDateTime is null and i.AdmitDateTime > convert(datetime2(0), '2022-05-01'))
order by i.DischargeDateTime desc
Some Other Recommendations

- If you use many similar subqueries, consider CTEs or temp tables.
- If you iteratively create many temp tables while refining a dataset, consider using UPDATE.
More Complicated Example

```sql
select temp.PatientSID, temp.IndexDate, vis.VisitDateTime
FROM CDWWork.Outpat.Visit vis
JOIN #visits temp
    on vis.PatientSID = temp.PatientSID
WHERE vis.Sta3n = 523
    and abs(datediff(month, temp.IndexDate, vis.VisitDateTime)) <= 1

select temp.PatientSID, temp.IndexDate, vis.VisitDateTime
FROM CDWWork.Outpat.Visit vis
JOIN #visits temp
    on vis.PatientSID = temp.PatientSID
WHERE vis.Sta3n = 523
    and vis.VisitDateTime >= dateadd(month, -1, temp.IndexDate)
    and vis.VisitDateTime < dateadd(month, 1, temp.IndexDate)
```
Did It Work?

This Photo by Unknown Author is licensed under CC BY-SA
Why Not?

- If partition elimination is used, it will be the first step.
- Therefore, it can’t be used with something that needs to be evaluated for every row.

```sql
select temp.PatientSID, temp.IndexDate, vis.VisitDateTime
FROM CDWWork.Outpat.Visit vis
JOIN #visits temp
on vis.PatientSID = temp.PatientSID
WHERE vis.Sta3n = 523
and vis.VisitDateTime >= dateadd(month, -1, temp.IndexDate)
and vis.VisitDateTime < dateadd(month, 1, temp.IndexDate)
```
Should We Use the Partition?

select min(dateadd(month, -1, IndexDate)) FROM #visits
--2017-12-01 00:00:01
select max(dateadd(month, 1, IndexDate)) FROM #visits
--2018-02-01 23:59:54

Look at that narrow range! We SHOULD use the partition…but how?

A good plan would use the max and min dates to grab the right partition.

This is the sort of thing I can’t do. Sorry.
Let’s Help Out

```
create clustered index idx_date on #visits (PatientSID,indexdate) --didn't help

create statistics dates on #visits (indexdate) --didn't help

select temp.PatientSID, temp.IndexDate, vis.VisitDateTime
FROM CDWWork.Outpat.Visit vis
JOIN #visits temp
ON vis.PatientSID = temp.PatientSID
WHERE vis.Sta3n = 523
and vis.VisitDateTime >= dateadd(month, -1, temp.IndexDate)
and vis.VisitDateTime < dateadd(month, 1, temp.IndexDate)
and vis.VisitDateTime >= (select min(dateadd(month, -1, IndexDate)) FROM #visits) --didn't help
```
vis.VisitDateTime >= (select min(dateadd(month, -1, IndexDate)) FROM #visits) --didn't help

If IndexDate was replaced with getdate(), then SQL would be able to use partition elimination. Even though select min()… will obviously only return one value, SQL doesn’t know to do that part first.
Do That Part First, Please

- How about if we move that line up to be the first thing in the WHERE clause?
- How about if we change the JOIN order to look at the temp table first?
- How about if we move that WHERE criterion into the inner JOIN clause?

Sorry Dave, I can’t let you do that.
So Order Doesn’t Matter?

- Yes it does! Remember that your script should be readable for you and other humans.

Doesn’t matter to me.
Back to the Problem

declare @mindate datetime2(0) =
    (select dateadd(month, -1, min(IndexDate))
    FROM #visits)

declare @maxdate datetime2(0) =
    (select dateadd(month, 1, max(IndexDate))
    FROM #visits)

select temp.PatientSID, temp.IndexDate, vis.VisitDateTime
FROM CDWWork.Outpat.Visit vis
JOIN #visits temp
    on vis.PatientSID = temp.PatientSID
WHERE vis.Sta3n = 523
and vis.VisitDateTime >= dateadd(month, -1, temp.IndexDate)
and vis.VisitDateTime < dateadd(month, 1, temp.IndexDate)
and vis.VisitDateTime >= @mindate --eureka
and vis.VisitDateTime < @maxdate --I have found it
Whoa, Bad Estimates!

Whoops.
Remember Table Statistics?

If you see these warnings in your project or temp tables, consider (re)indexing them.
Bad Estimates

- Try adding a predicate on an indexed column like Sta3n.
  - Remember that sometimes you want to add criteria that don’t change your results, just for performance.
Bonus Tip: Live Query Statistics

If you click this, you will see a live, animated execution plan.

Still working on it

Done!
Key Takeaways

- Don’t be greedy!
- Use partitions!
- Separate functions from columns.
- Work with your colleague.
VINCI Resources

- [VINCI University (va.gov)](va.gov)
- [VINCI Training & Office Hour (va.gov)](va.gov)
  - Especially “Managing Research Data in SQL Server” VINCI Training Hour.
  - VINCI Office Hours every Wednesday at 3 PM ET.
BISL Resources

- **CDW Guide Query Best Practices.docx (va.gov)**
- **Six Steps to Query Improvement**
- **BISL Training (sharepoint.com)**
  - Especially SQL Office Hours on Tuesdays and Fridays.
One More Resource

Questions?

- VINCI@va.gov
- Andrew.Holbrook@va.gov