BC/NAV TECH DAYS 2022

mibuso.com

10 YEAR ANNIVERSARY

www.bctechdays.com

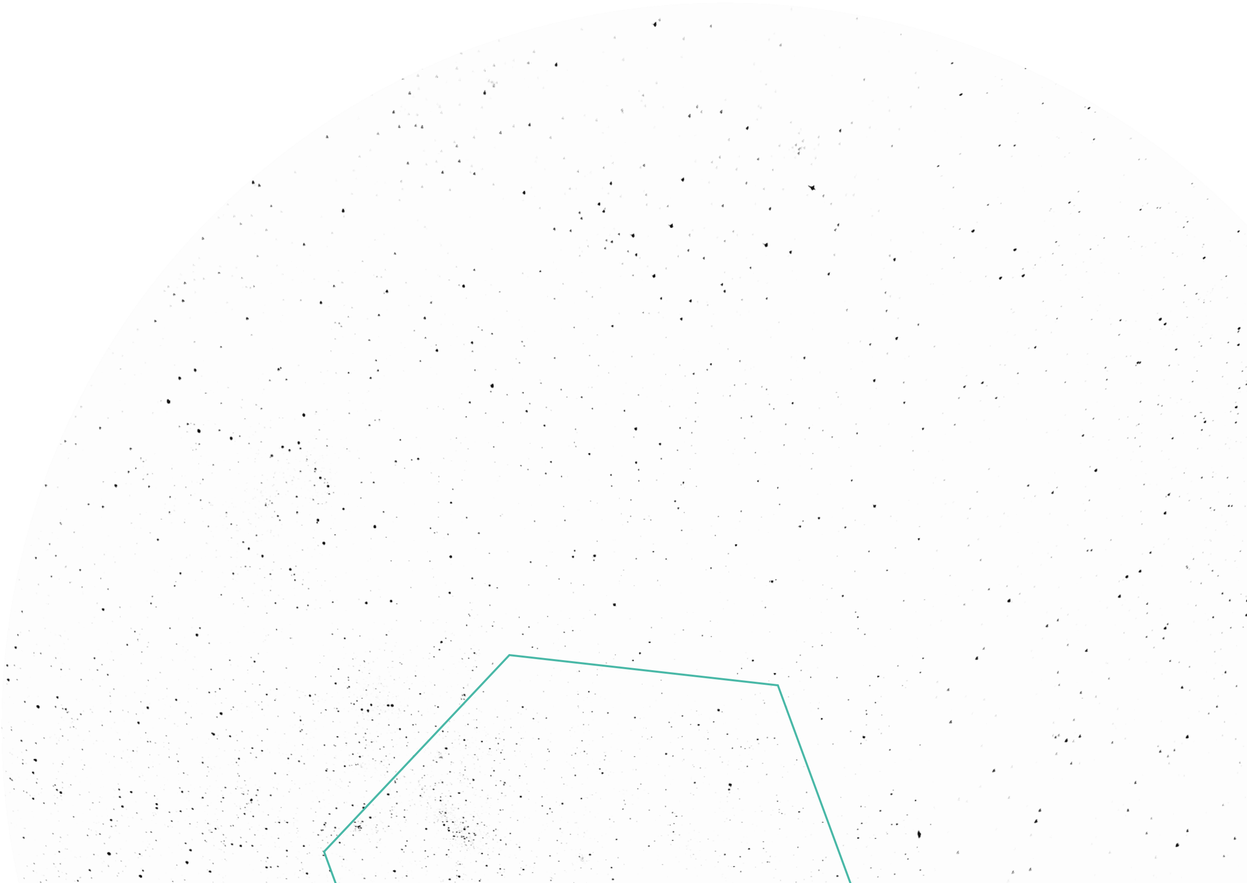
Optimizing SQL queries in Business Central

Milan Milinčević
NAVIPARTNER

mmilince

Agenda

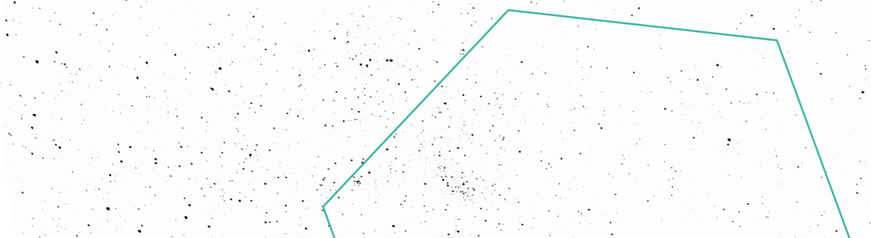
- Index
- Clustered and non-clustered indexes
- Covering index
- SIFT
- NCCI
- … and much more



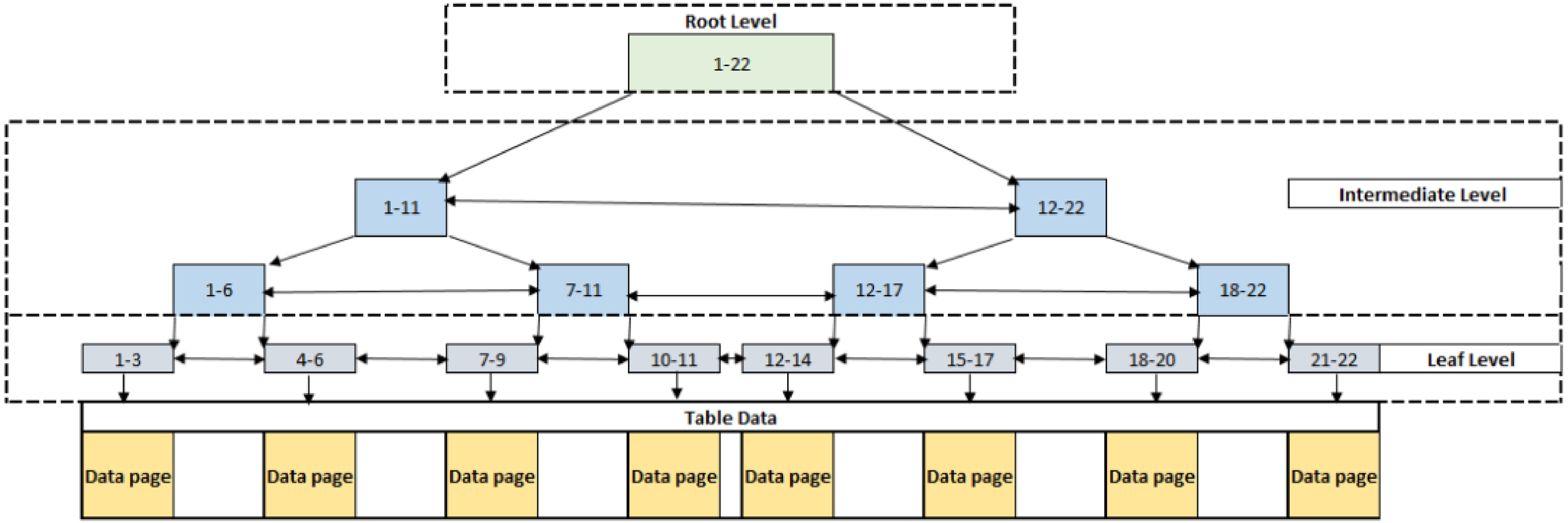

INDEX

„An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.“

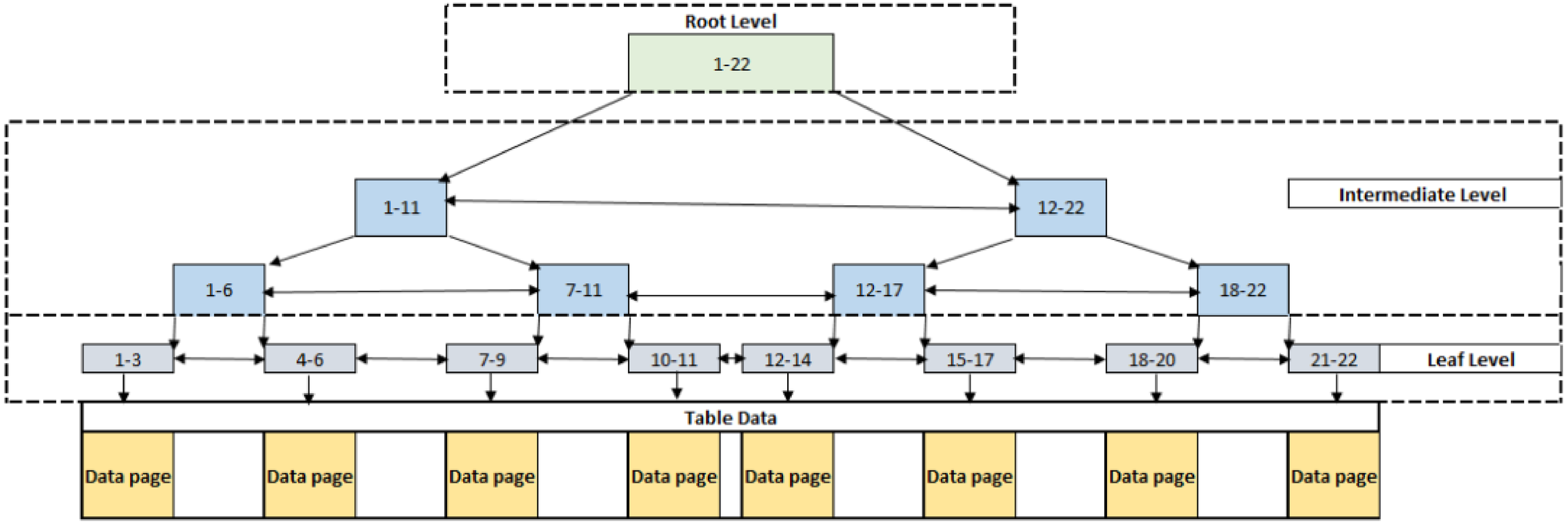
-Microsoft



Clustered index

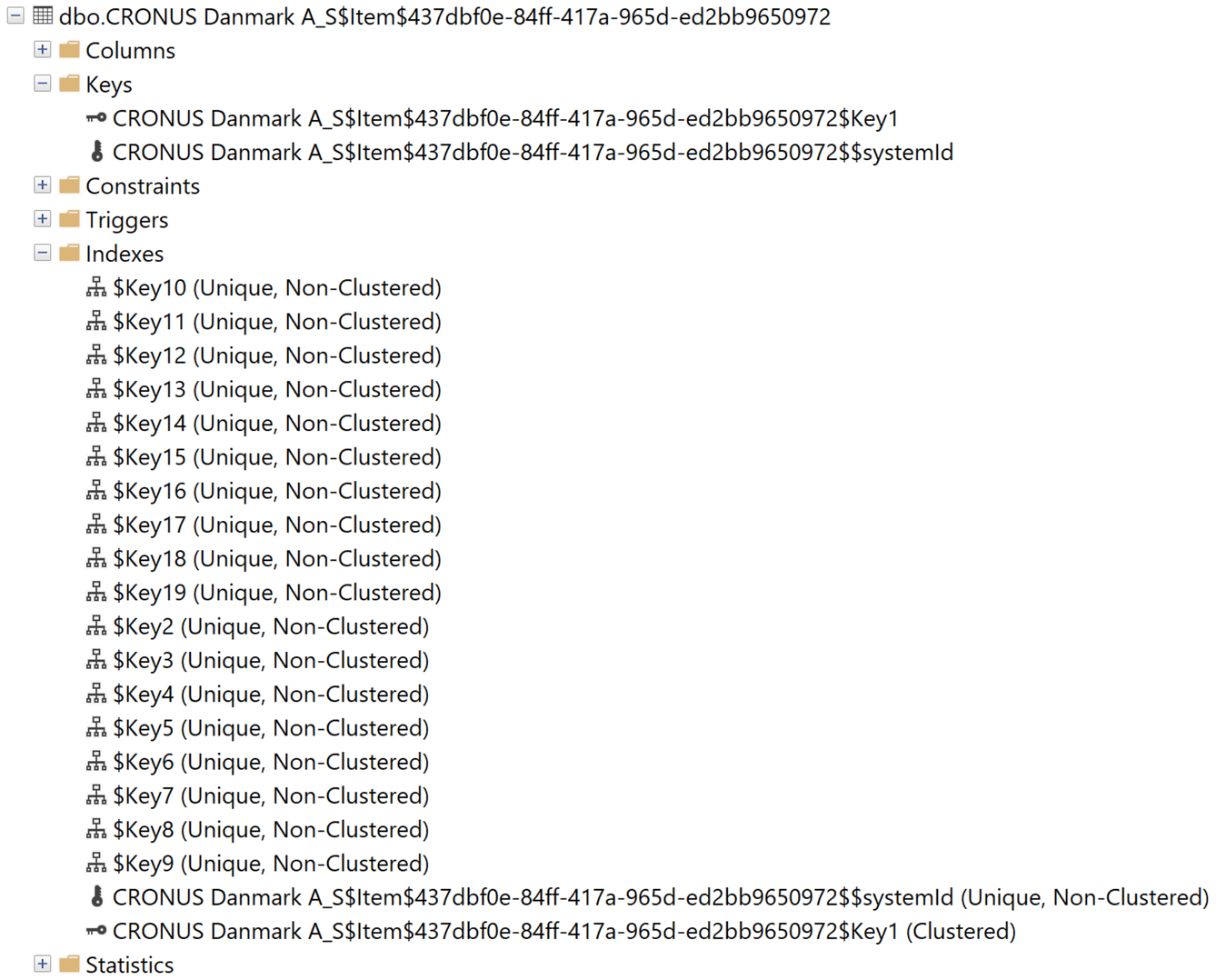


Clustered index




Non-clustered
INDEX

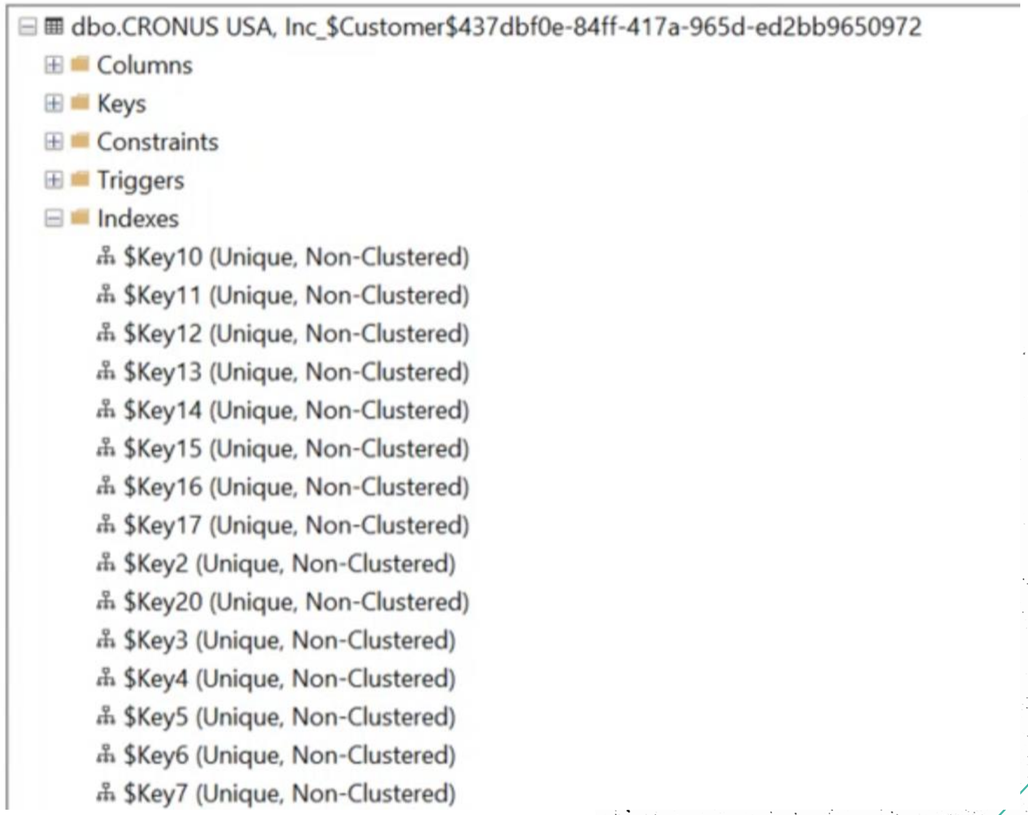
Key vs Index


dbo.CRONUS Danmark A_S$Item$437dbf0e-84ff-417a-965d-ed2bb9650972
Columns
Keys
CRONUS Danmark A_S$Item$437dbf0e-84ff-417a-965d-ed2bb9650972$Key1
CRONUS Danmark A_S$Item$437dbf0e-84ff-417a-965d-ed2bb9650972$$systemId
Constraints
Triggers
Indexes
$Key10 (Unique, Non-Clustered)
$Key11 (Unique, Non-Clustered)
$Key12 (Unique, Non-Clustered)
$Key13 (Unique, Non-Clustered)
$Key14 (Unique, Non-Clustered)
$Key15 (Unique, Non-Clustered)
$Key16 (Unique, Non-Clustered)
$Key17 (Unique, Non-Clustered)
$Key18 (Unique, Non-Clustered)
$Key19 (Unique, Non-Clustered)
$Key2 (Unique, Non-Clustered)
$Key3 (Unique, Non-Clustered)
$Key4 (Unique, Non-Clustered)
$Key5 (Unique, Non-Clustered)
$Key6 (Unique, Non-Clustered)
$Key7 (Unique, Non-Clustered)
$Key8 (Unique, Non-Clustered)
$Key9 (Unique, Non-Clustered)
CRONUS Danmark A_S$Item$437dbf0e-84ff-417a-965d-ed2bb9650972$$systemId (Unique, Non-Clustered)
CRONUS Danmark A_S$Item$437dbf0e-84ff-417a-965d-ed2bb9650972$Key1 (Clustered)
Statistics

```
keys
{
    - reference
    key(Key1; "Code")
    {
    }
    - reference
    key(Key2; "Starting date", "Starting time", "Ending date", "Ending time")
    {
    }
    - reference
    key(Key3; "Ending date", "Ending time")
    {
    }
    - reference
    key(Key4; "Actual Discount Amount", "Actual Item Qty.")
    {
    }

    - reference
    key(Key5; "Replication Counter")
    {
    }
}
```

Keys on base table

- Create new table keys on table extensions
- Not possible to combine standard and custom fields



~~SetCurrentKey()~~ SetOrder()

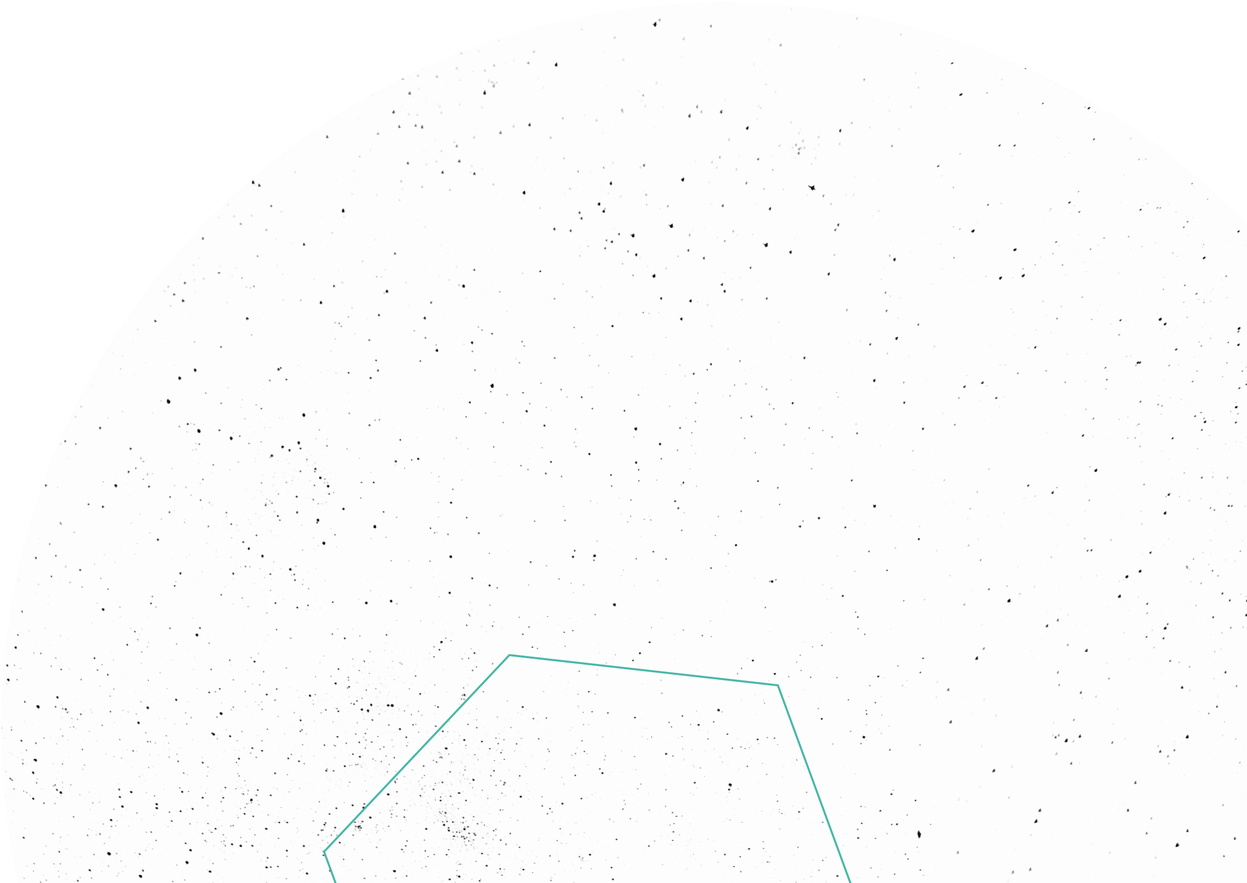
- Used to sort by the specified fields
- SQL Server choose which index to use
- You can use any fields as parameters

```
local procedure TestSetCurrentKey()
var
    Item: Record Item;
begin
    Item.SetCurrentKey("Statistics Group");
    Item.SetFilter("Statistics Group", '>10');
    if Item.FindSet() then begin
        //some business logic
    end;
end;
```

Caching

Two levels of cache:

- BC Server Instance Data Cache
- SQL Server Data Cache



Caching

Two levels of cache:

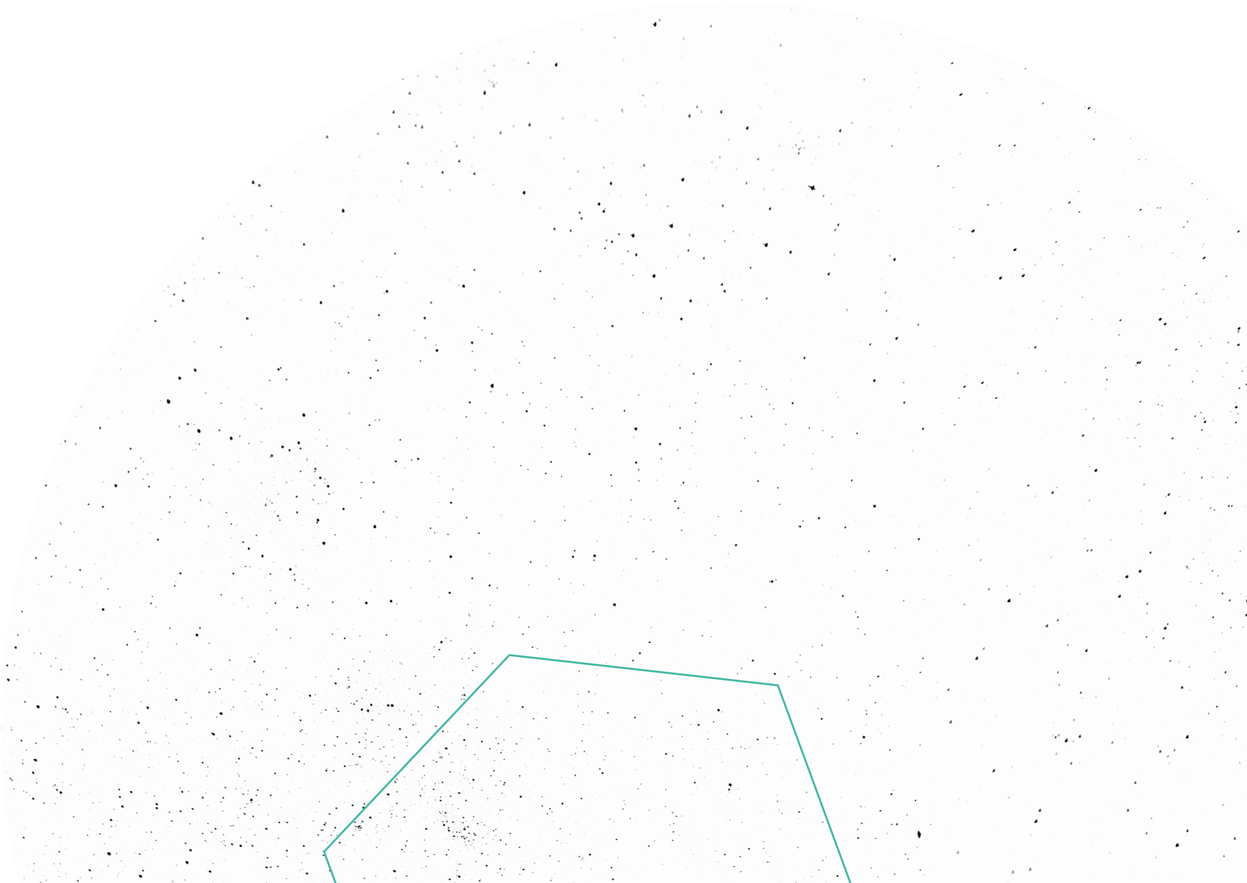
- BC Server Instance Data Cache
- SQL Server Data Cache

BC:

- SelectLatestVersion()

SQL server:

- DBCC DROPCLEANBUFFERS
- DBCC FREEPROCCACHE



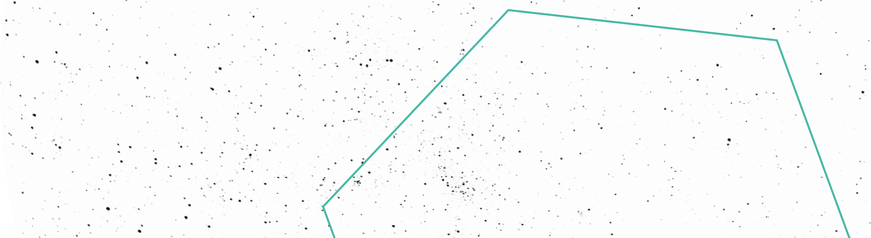
Partial Records

- Loading subset of normal fields
- Avoid potential JOINS

SetLoadFields	AddLoadFields	AreFieldsLoaded	LoadFields
Sets the fields to be initially loaded	Adds fields to the current set of fields to be initially loaded	Checks whether the specified fields are all initially loaded	Accesses the table's corresponding data source to load the specified fields

PROBLEM:

Partial records would require all fields used as filters in calcformulas on flowfield to be always loaded



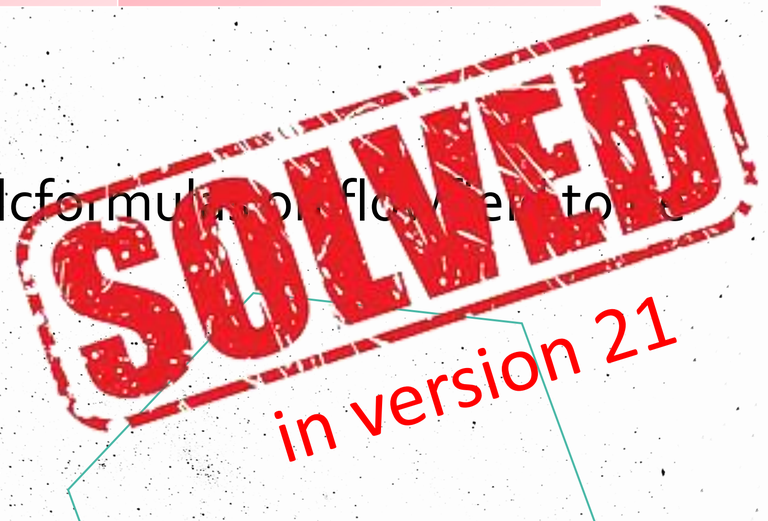
Partial Records

- Loading subset of normal fields
- Avoid potential JOINS

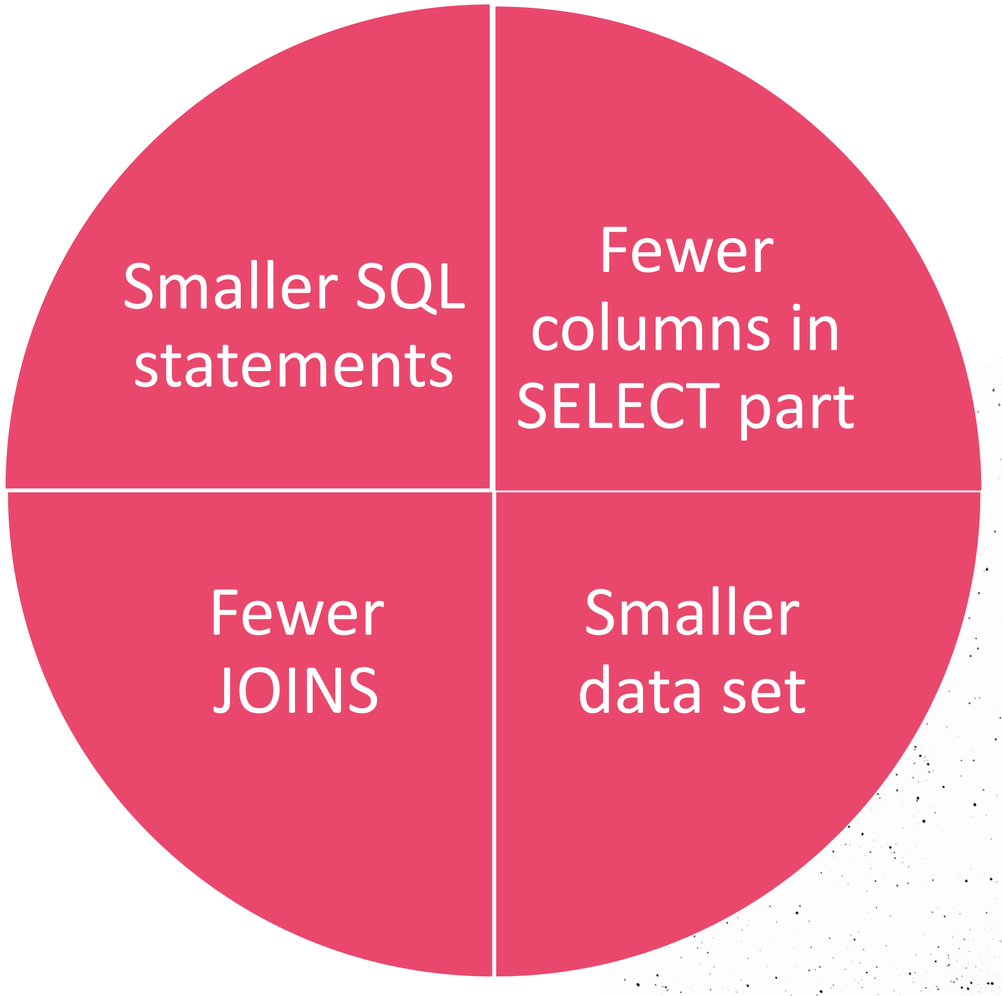
SetLoadFields	AddLoadFields	AreFieldsLoaded	LoadFields
Sets the fields to be initially loaded	Adds fields to the current set of fields to be initially loaded	Checks whether the specified fields are all initially loaded	Accesses the table's corresponding data source to load the specified fields

PROBLEM:

Partial records would require all fields used as filters in calcformulas or flowfields to be always loaded

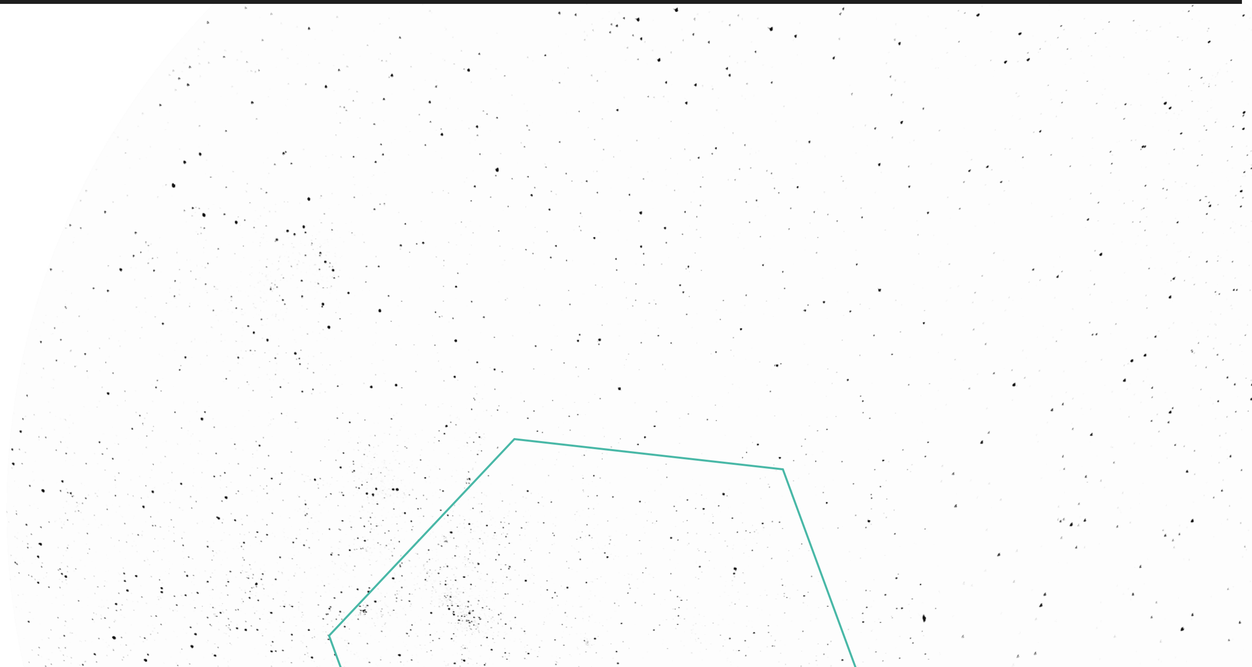


Partial records - benefits



Partial Records - example

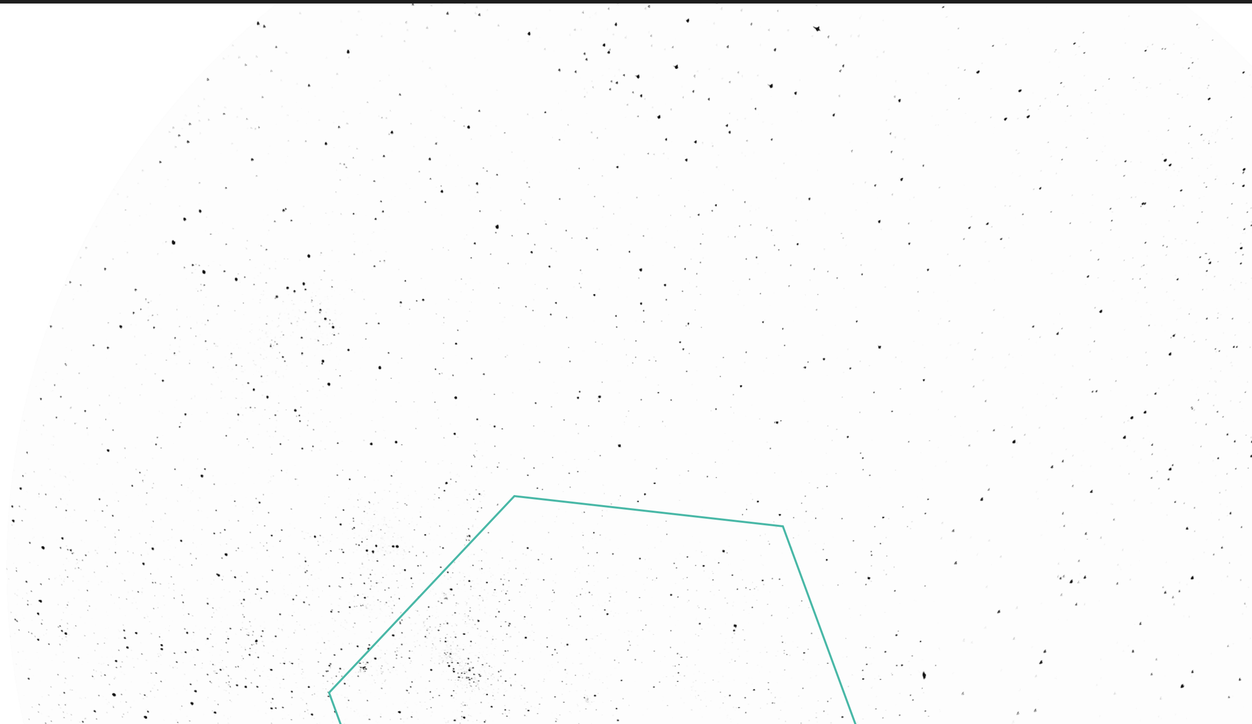

```
SELECT "50100"."timestamp","50100"."Entry No_","50100"."Description","50100"."Description 2","50100"."Amount","50100"."Amount 2","50100"."Amount
3","50100"."Amount 4","50100"."Amount 5","50100"."Start Date","50100"."Color","50100"."Color 2","50100"."Color 3","50100"."Car No_","50100"."Car No
_ 2","50100"."Car No_ 3","50100"."Car No_ 4","50100"."Car No_ 5","50100_e7"."End Date","50100"."$systemId","50100"."$systemCreatedAt","50100"."$sys
temCreatedBy","50100"."$systemModifiedAt","50100"."$systemModifiedBy" FROM "CRONUS".dbo."CRONUS Danmark A_S$Demo$4f09c3af-fe51-4b4d-ade3-f2aee1c28a
06" "50100"  WITH(READUNCOMMITTED)  JOIN "CRONUS".dbo."CRONUS Danmark A_S$Demo$66122ea4-ae42-4836-88d1-aa3a3e123a8e" "50100_e7"  WITH(READUNCOMMITT
ED)  ON ("50100"."Entry No_" = "50100_e7"."Entry No_") WHERE ("50100"."Car No_"=@0) ORDER BY "Entry No_" ASC OPTION(FAST 50)
```



Partial Records - example

BC/NAV TECH DAYS 2022
mibuso.com

```
SELECT "50100"."timestamp","50100"."Entry No_","50100"."Amount","50100"."Start Date","50100"."Car No_","50100"."$systemId","50100"."$systemCrea
tedAt","50100"."$systemCreatedBy","50100"."$systemModifiedAt","50100"."$systemModifiedBy" FROM "CRONUS".dbo."CRONUS Danmark A_S$Demo$4f09c3af-f
e51-4b4d-ade3-f2aee1c28a06" "50100"  WITH(READUNCOMMITTED)  WHERE ("50100"."Car No_"=@0) ORDER BY "Entry No_" ASC OPTION(FAST 50)
```



JUST IN TIME

Covering index (included columns)

- „When an index contains all the columns needed to satisfy a query, it is called a *covering* index."
- Data on leaf nodes of the index

```
SELECT Description, Description2
FROM Item
WHERE ItemNo = '1000'
ORDER BY LastModifiedDate
```

Optimal index:

(ItemNo, LastModifedDate, Description, Description2)

Optimal seek

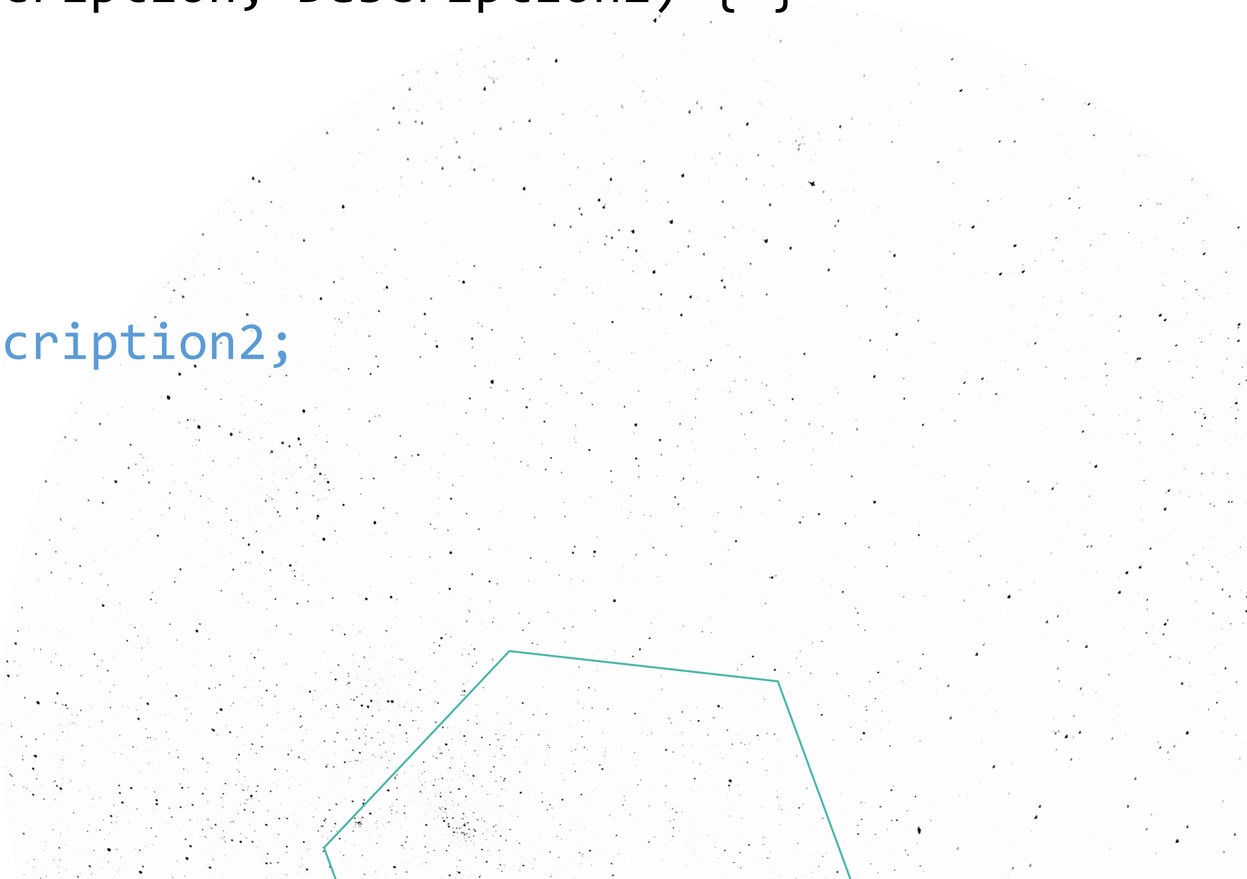
No need to sort

Covering – no need to lookup

Covering index (included columns)

BC/NAV TECH DAYS 2022
mibuso.com

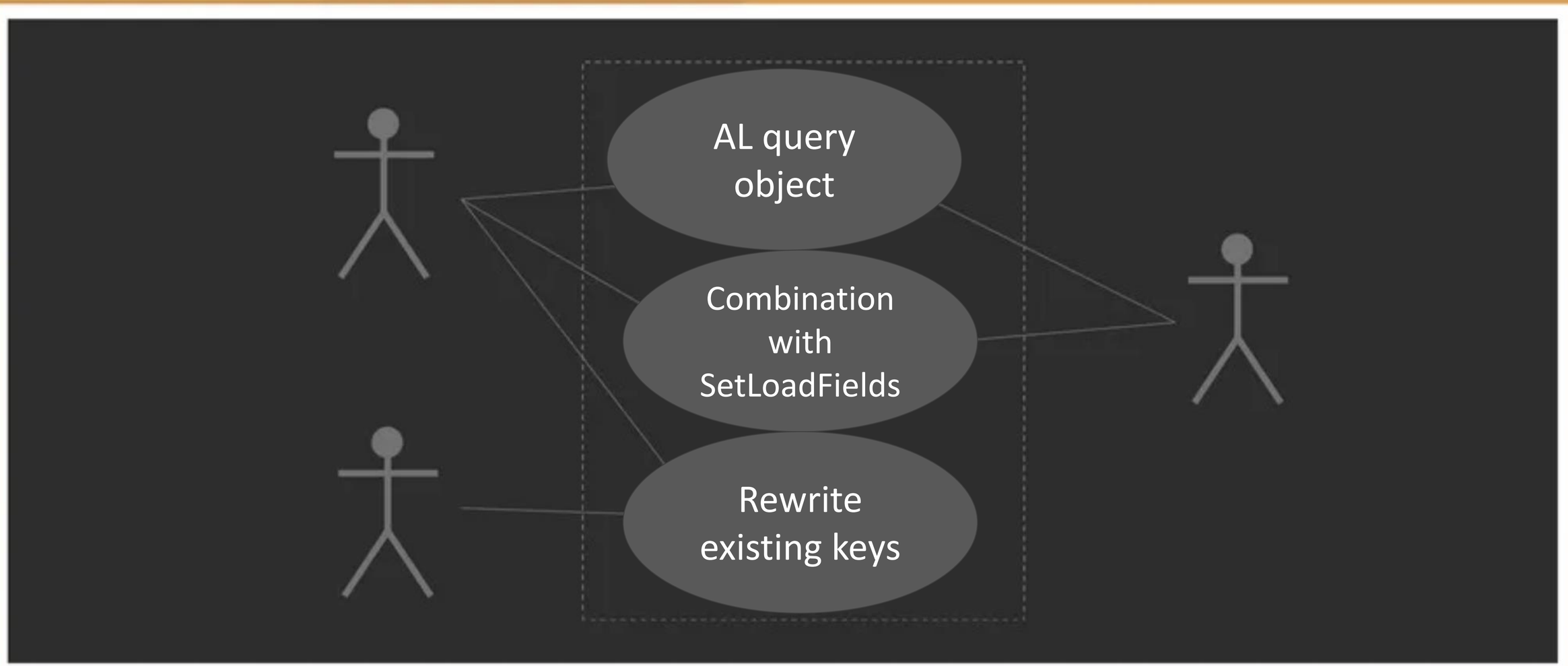
```
keys
{
    // Before
    key(K1; ItemNo, LastModifiedDate, Description, Description2) { }

    // After
    key(K2; ItemNo, LastModifiedDate)
    {
        IncludedFields = Description, Description2;
    }
}
```



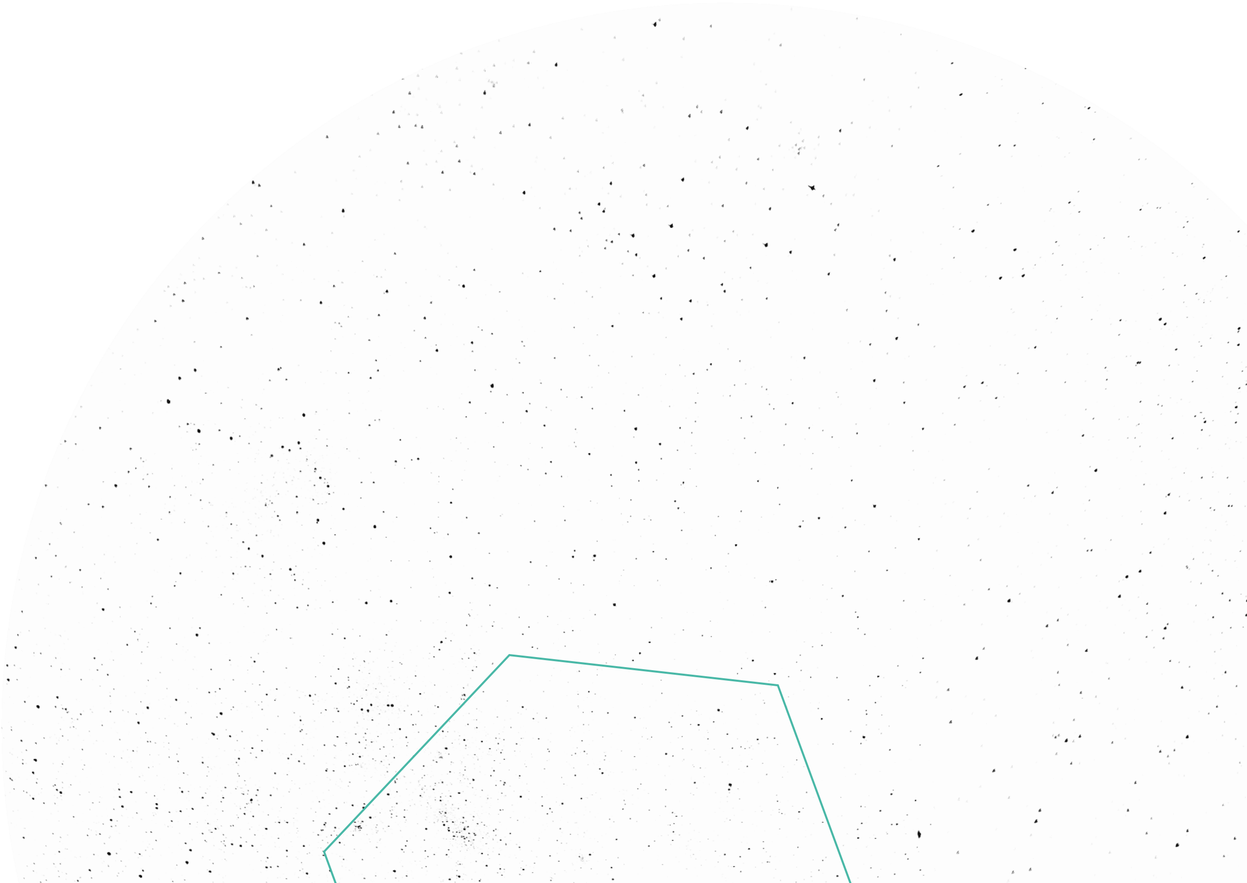
Why not put the included columns in the index keys?

- Storage cost
- Data type limits
- SQL Server limits

Use Case

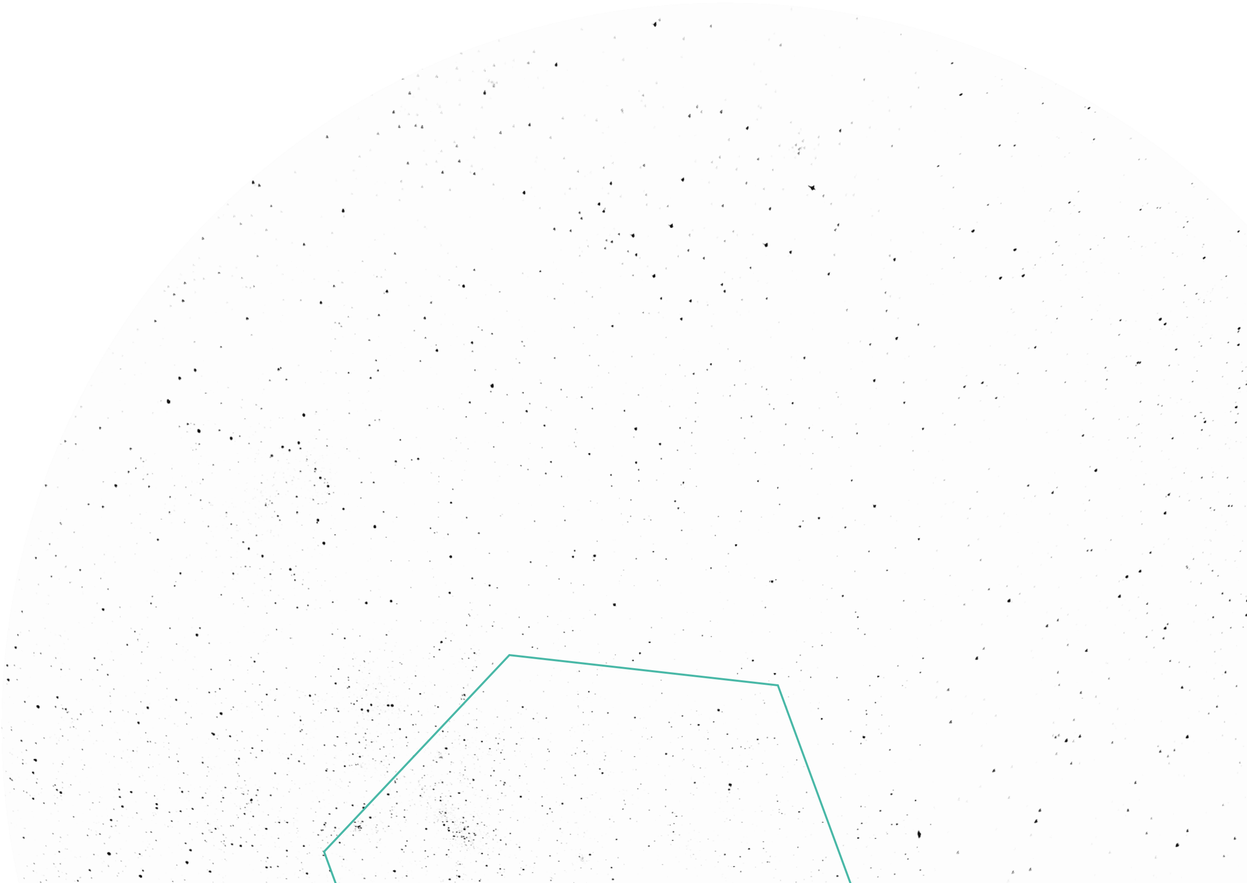

AL query object
Combination with SetLoadFields
Rewrite existing keys

Index + View = ?



Indexed View

- First index must be unique clustered index
- Stores the data
- Data is updated automatically
- Can supply data in a query



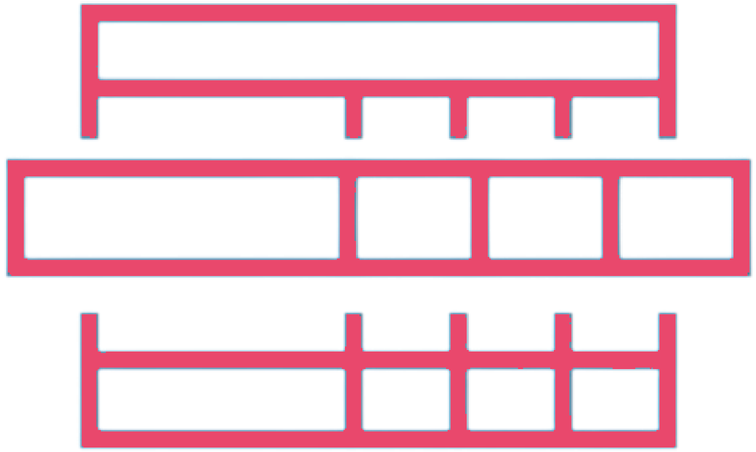
- Inserts, modifies and deletes query performance can degrade a lot
- Frequently updated data
- Performance gains vc costs

SIFT

- Quickly calculate sums of numeric data
- Optimize performance of FlowFields
- Indexed Views are used to maintan SIFT

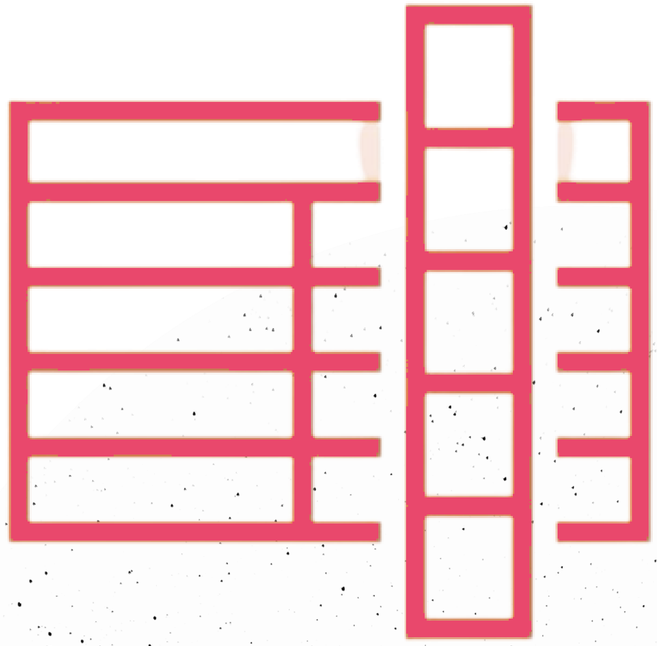
```
keys
{
    key(Key1; "Transaction No.")
    {
    }
    key(Key2; "Card Code", "Posting Date")
    {
        SumIndexFields = Points, "Remaining Points";
        MaintainSiftIndex = true;
    }
}
```

- Designing SIFT indexes optimally
- Only for large tables
- Supporting too many SIFT indexes will affect performance
- The fields that are used most regularly in queries must be positioned to the left in the index.



RowStore

All rows stored on pages

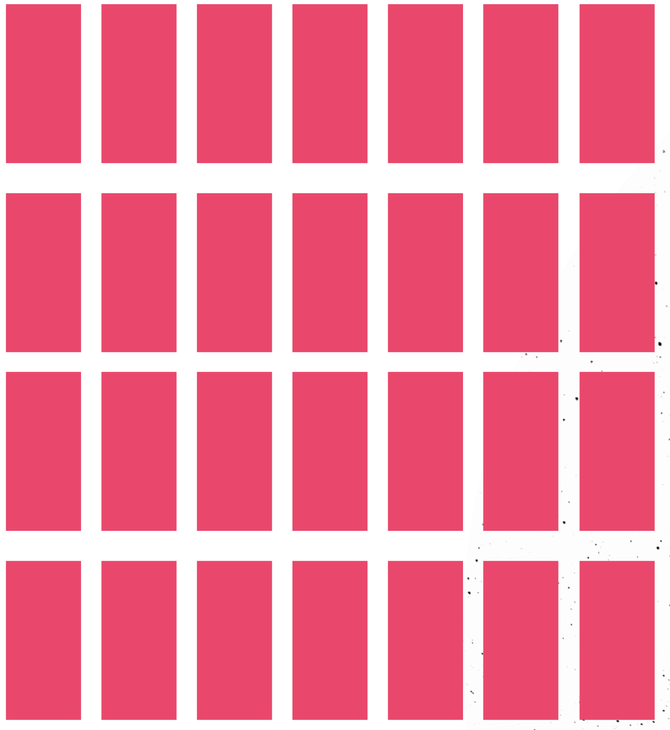


Columnstore

Specific columns stored in segments

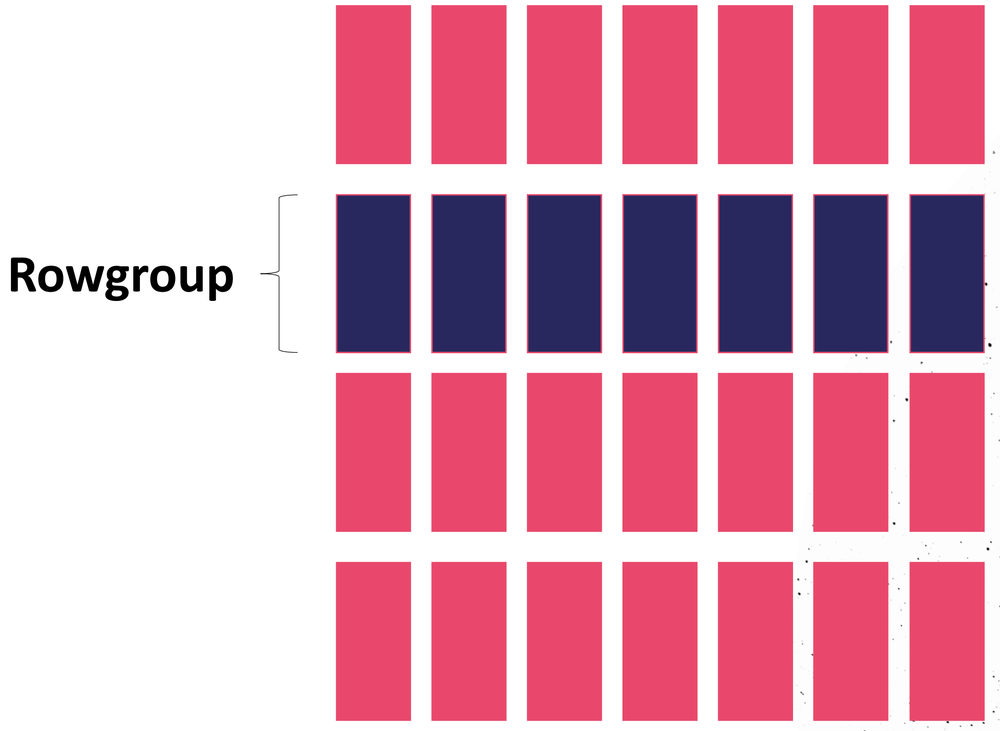
Columnstore indexes

- It is fast
- Allows advanced compression
- Provides segment elimination
- Batch mode processing



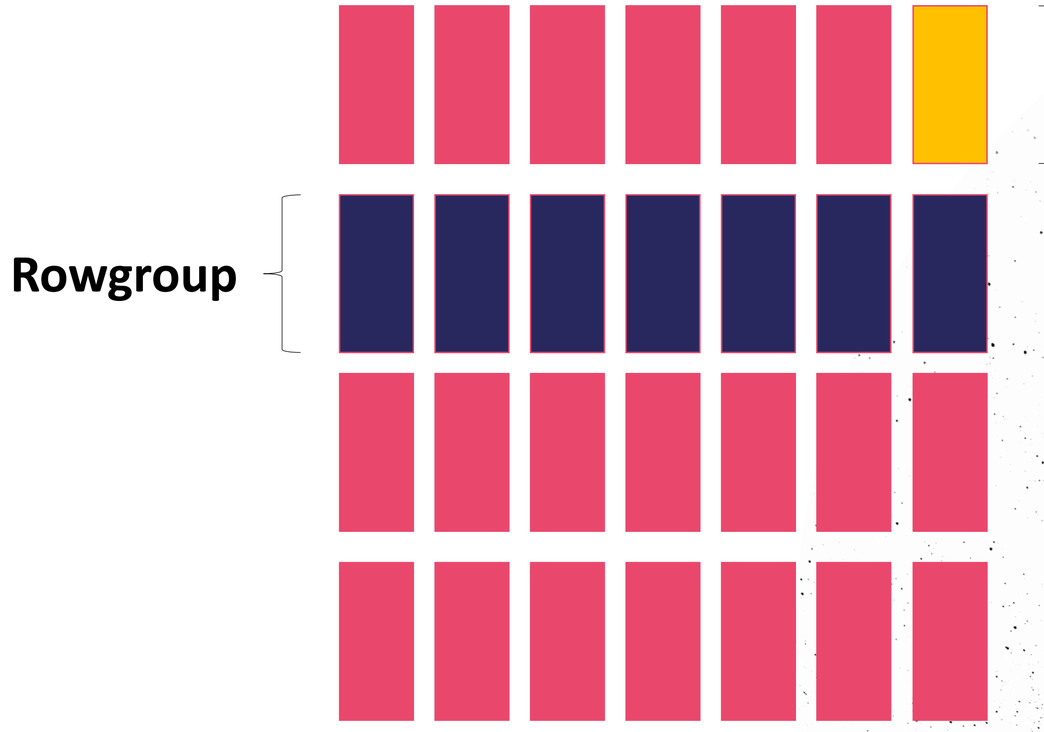
Columnstore indexes

- It is fast
- Allows advanced compression
- Provides segment elimination
- Batch mode processing



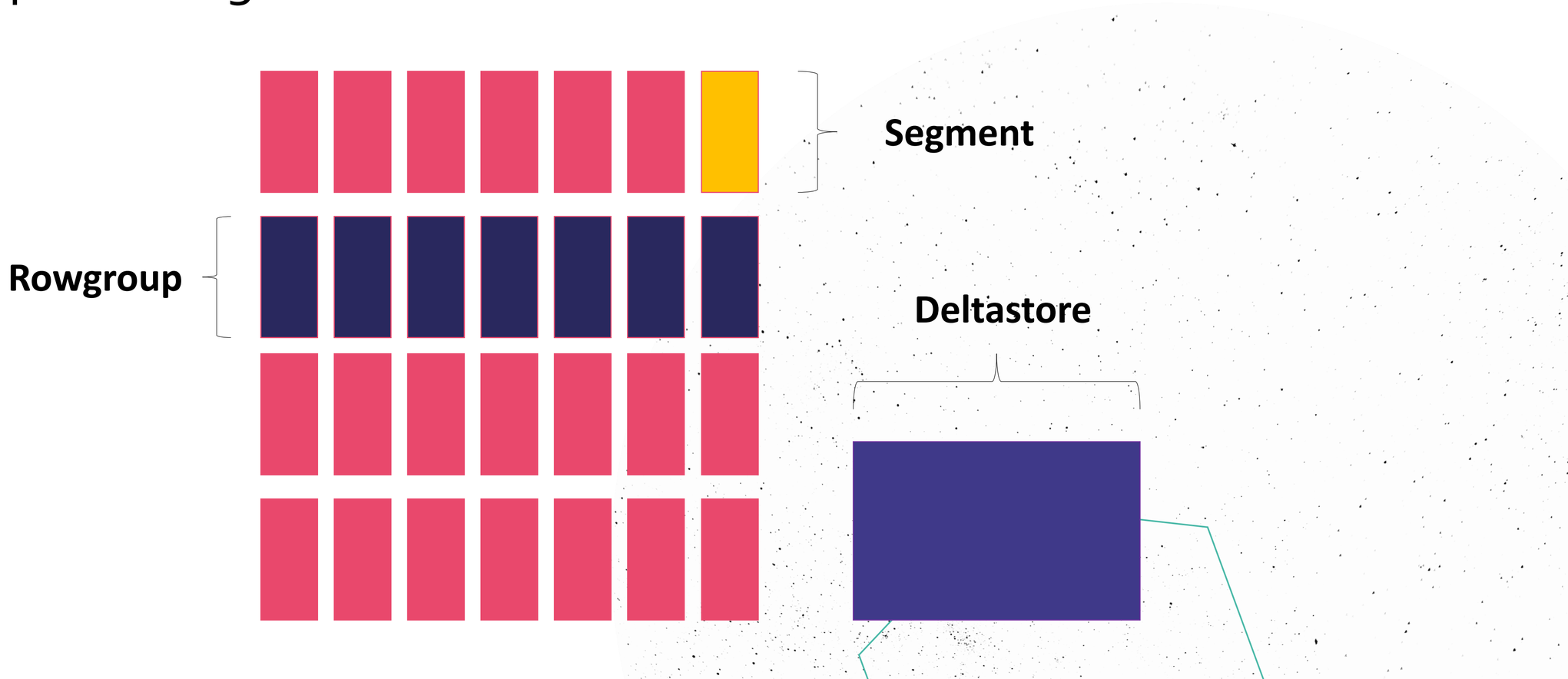
Columnstore indexes

- It is fast
- Allows advanced compression
- Provides segment elimination
- Batch mode processing



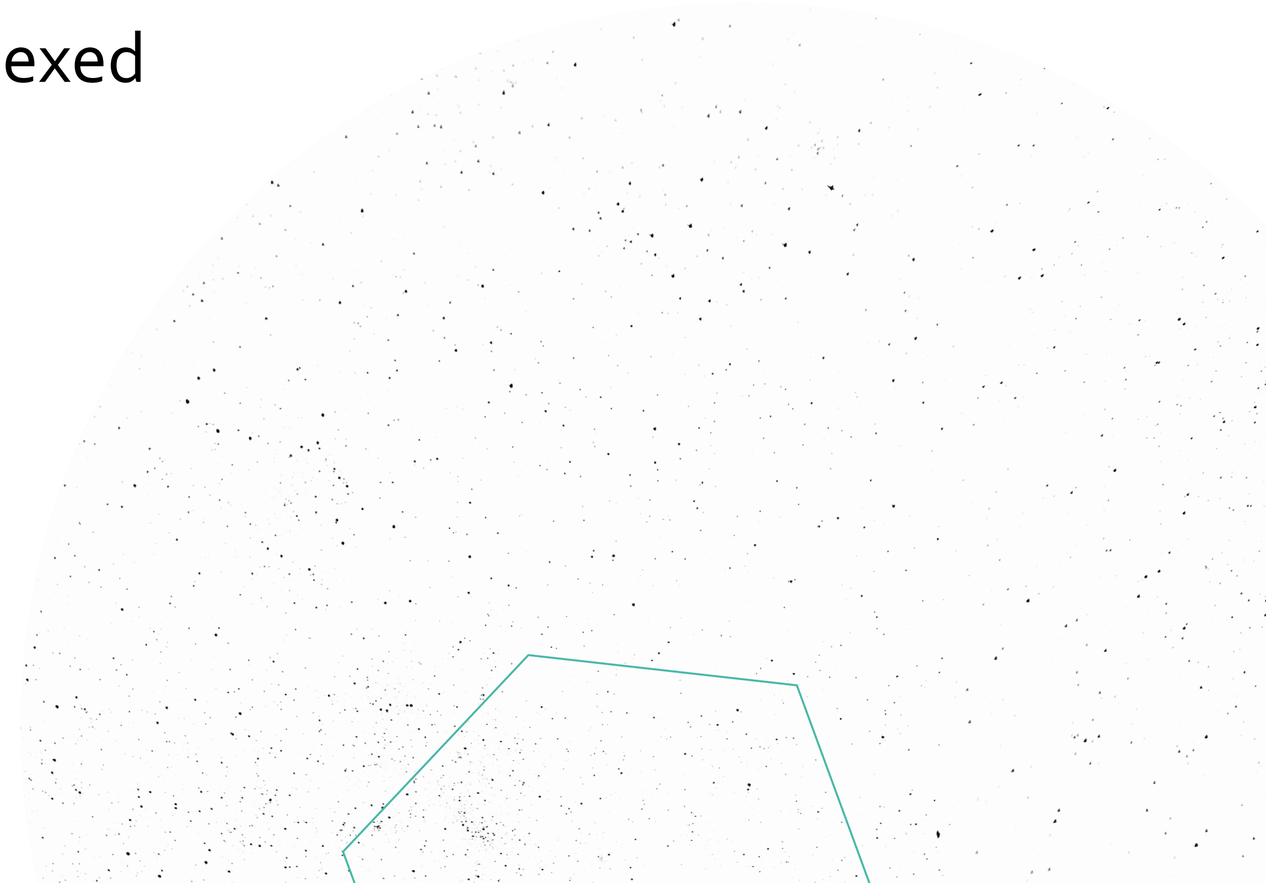
Columnstore indexes

- It is fast
- Allows advanced compression
- Provides segment elimination
- Batch mode processing



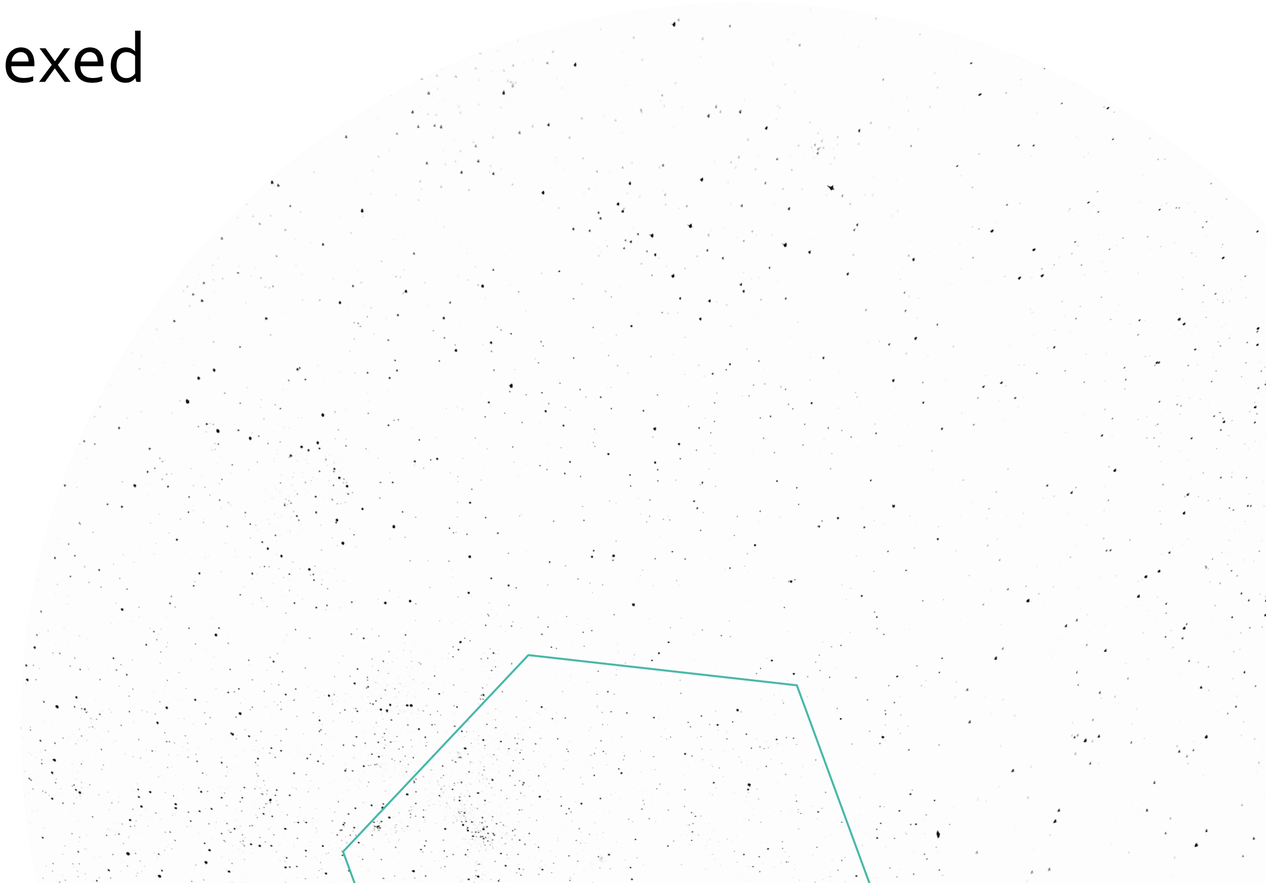
Non-clustered columnstore indexes

- In-memory
- Optimized for analytics
- Highly compressed
- Reduced locking compared to SIFT indexed



Non-clustered columnstore indexes

- In-memory
- Optimized for analytics
- Highly compressed
- Reduced locking compared to SIFT indexed
- For large tables



- Do you need NCCI at all?
- Which fields do you need to index?
- Can you disable some or all SIFT indexes and just use an NCCI?

Migrate SIFT to NCCI

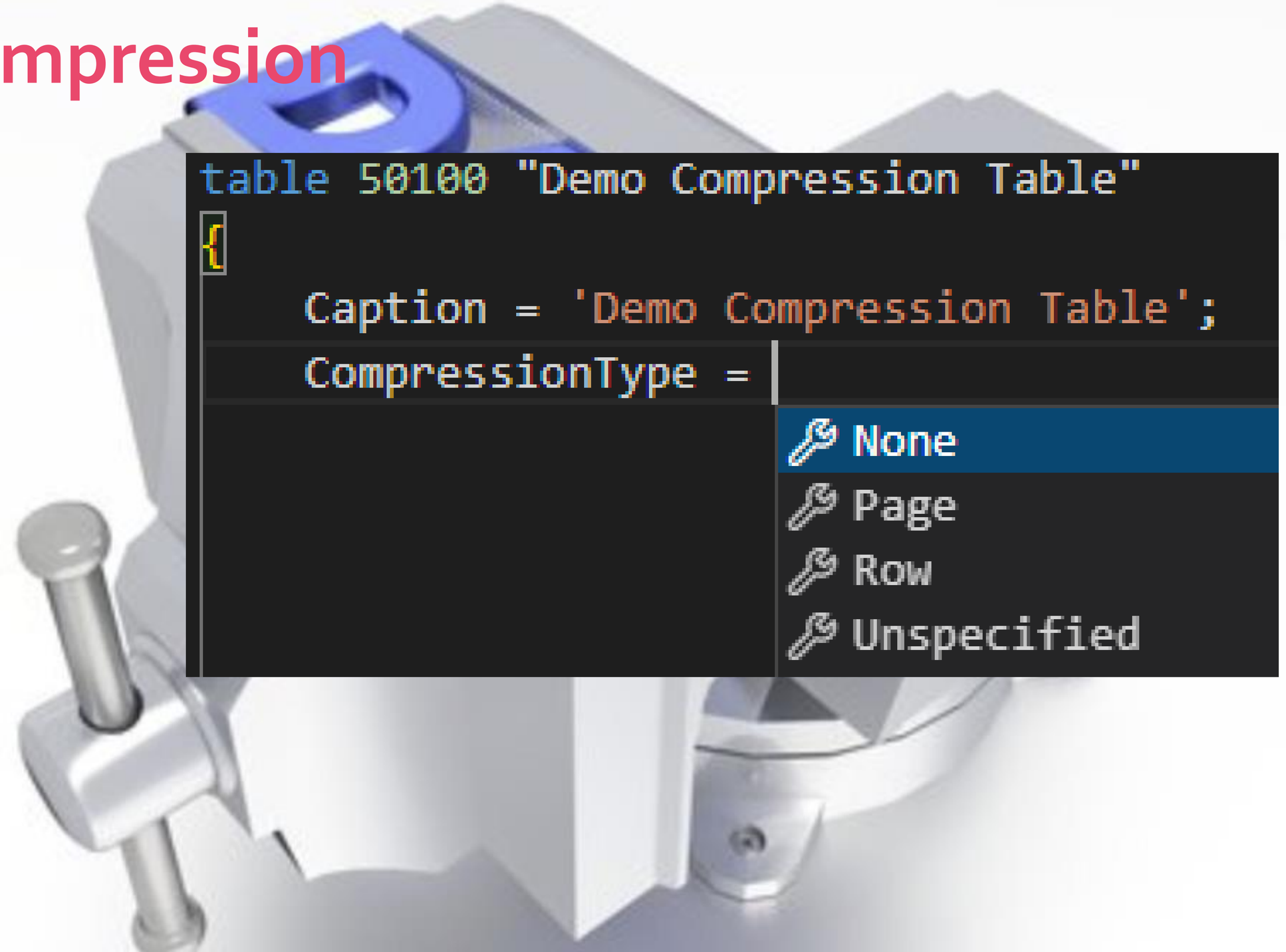
- Why?
- How?

```
table 50100 Student
{
    DataClassification = CustomerContent;
    // Defines NCCI to replace the SIFT keys
    ColumnStoreIndex = Code,FirstNames,ECTSPoints,NumberOfCourses;
```

Data compression

Two types:

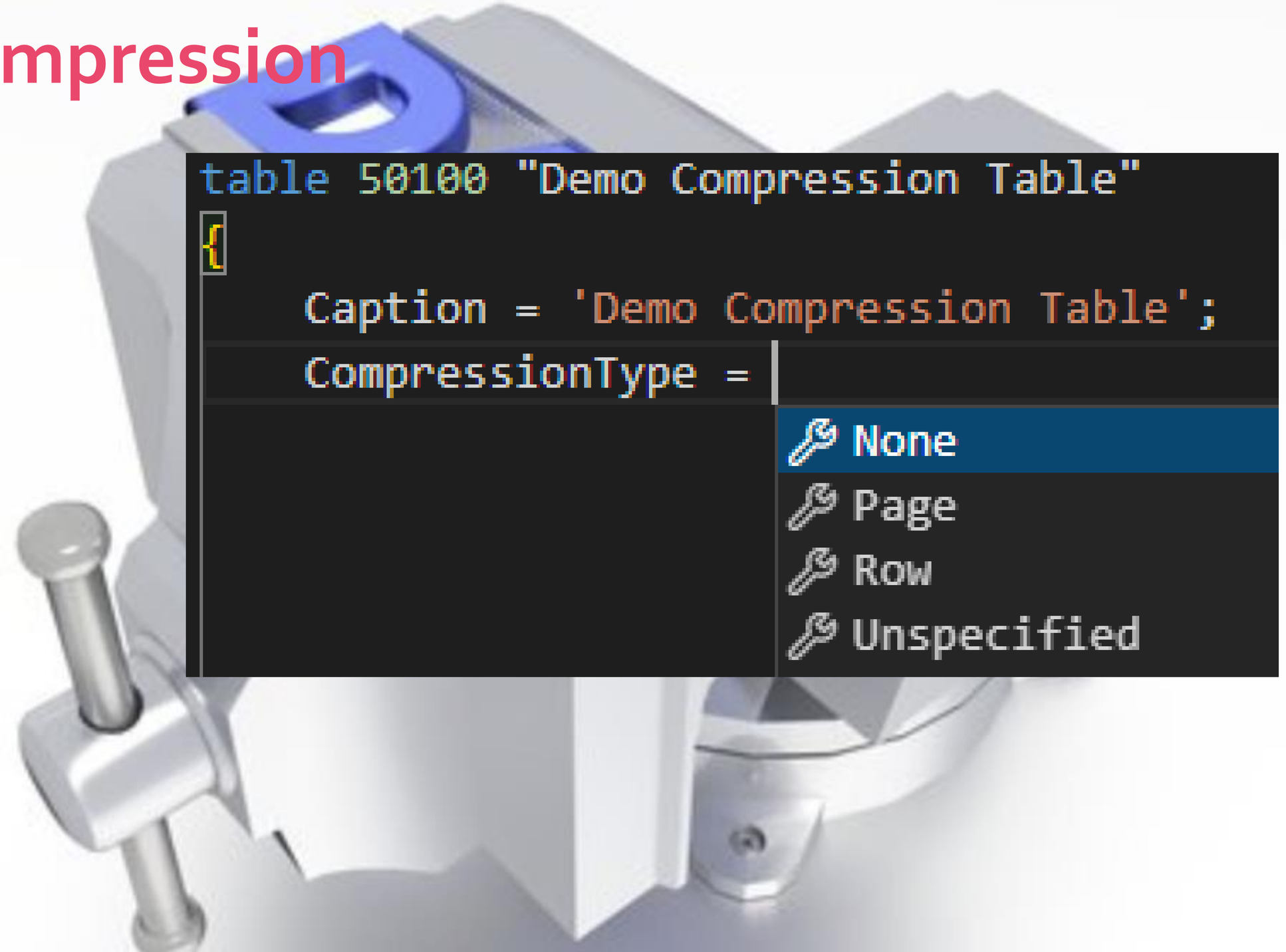
- Row-level
- Page-level



Data compression

Two types:

- Row-level
- Page-level

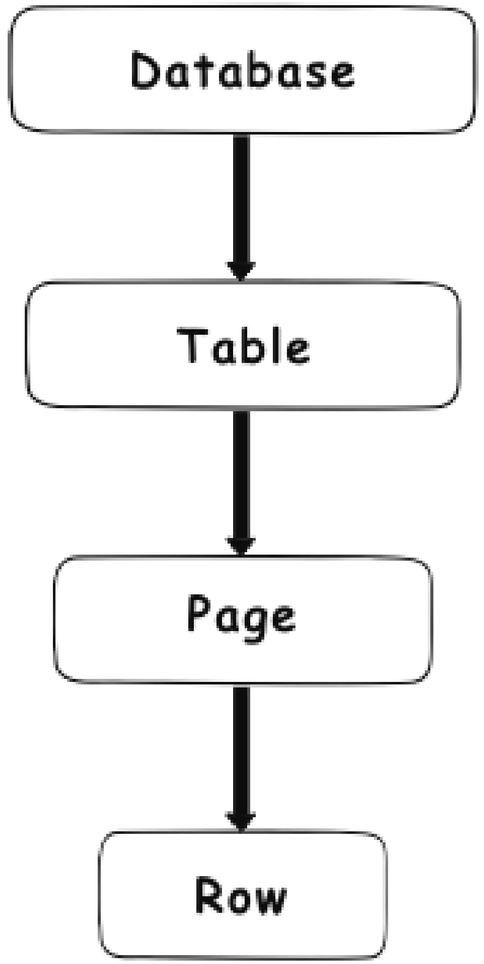


Locks

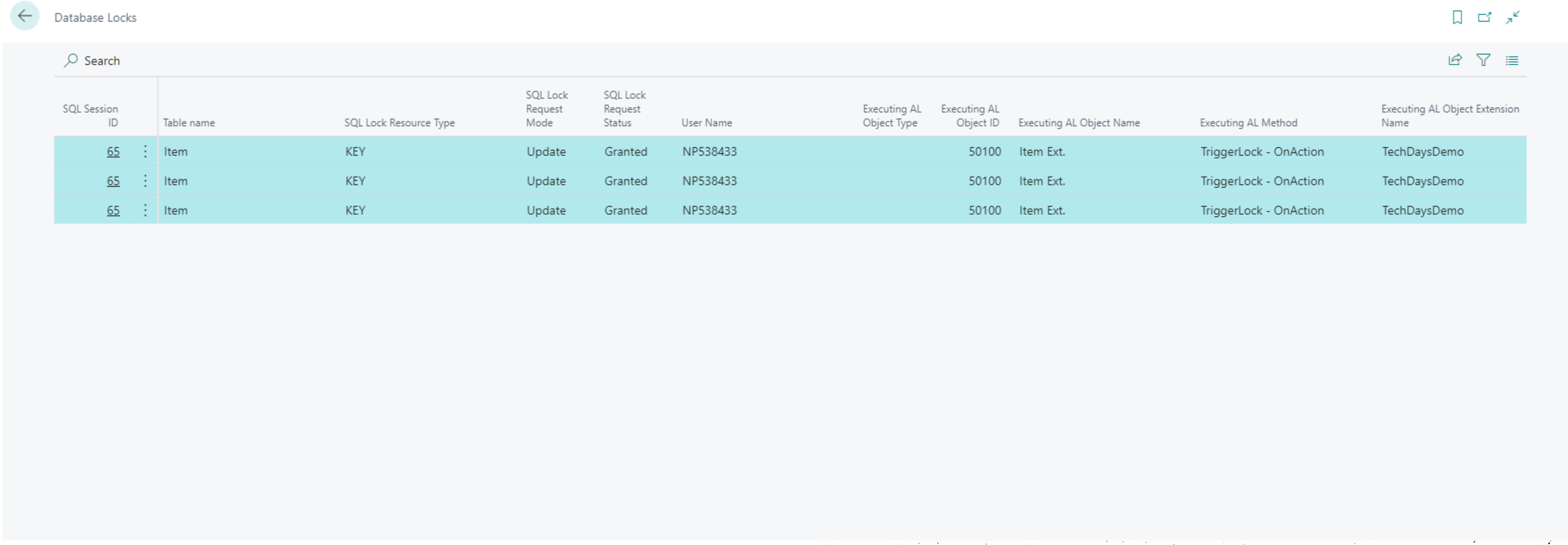
- Keep transactions as short as possible
- More keys we have, more time is needed on write operations

Locks

- Keep transactions as short as possible
- More keys we have, more time is needed on write operations
- LockTable() in BC

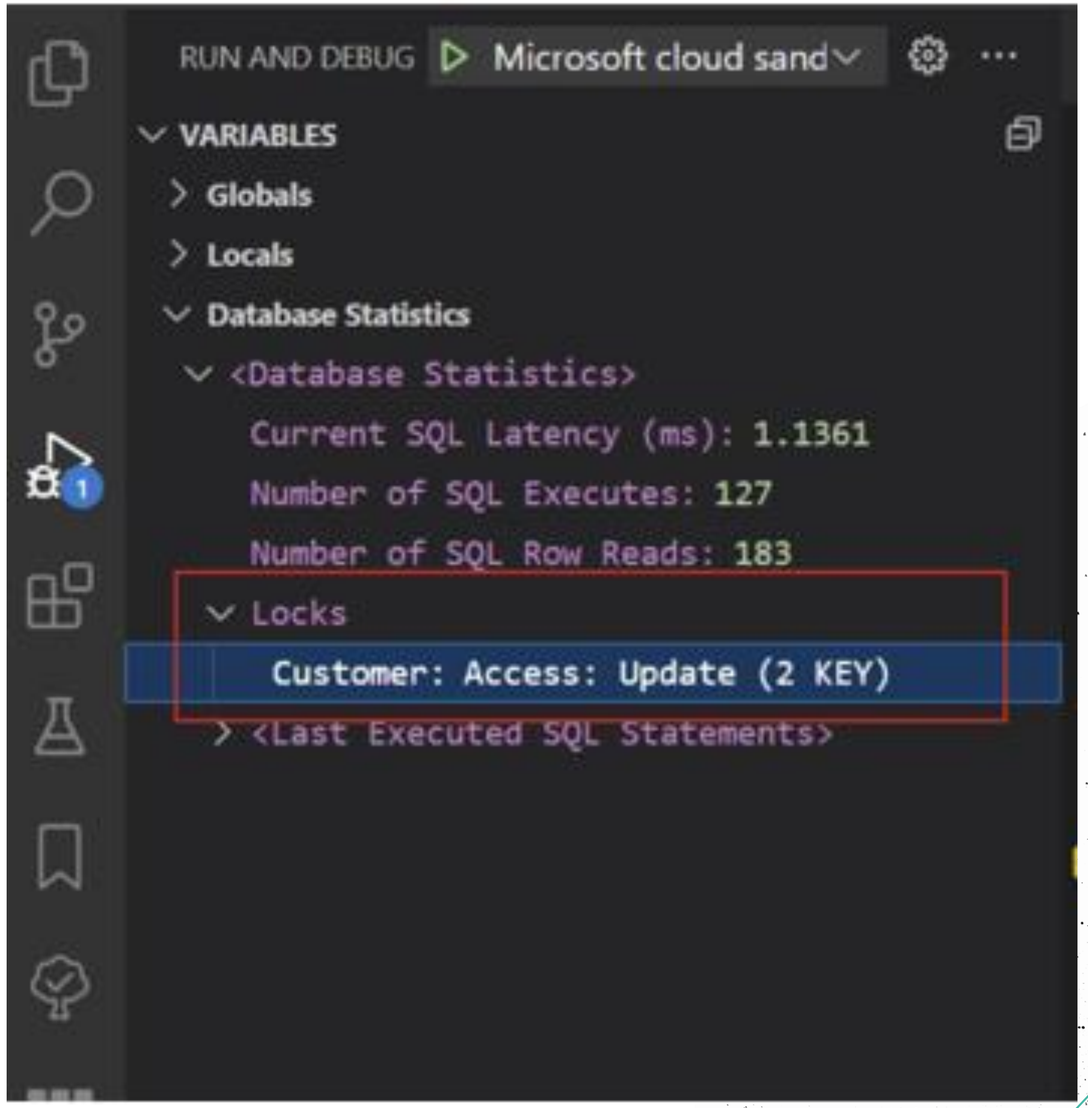


Database Locks



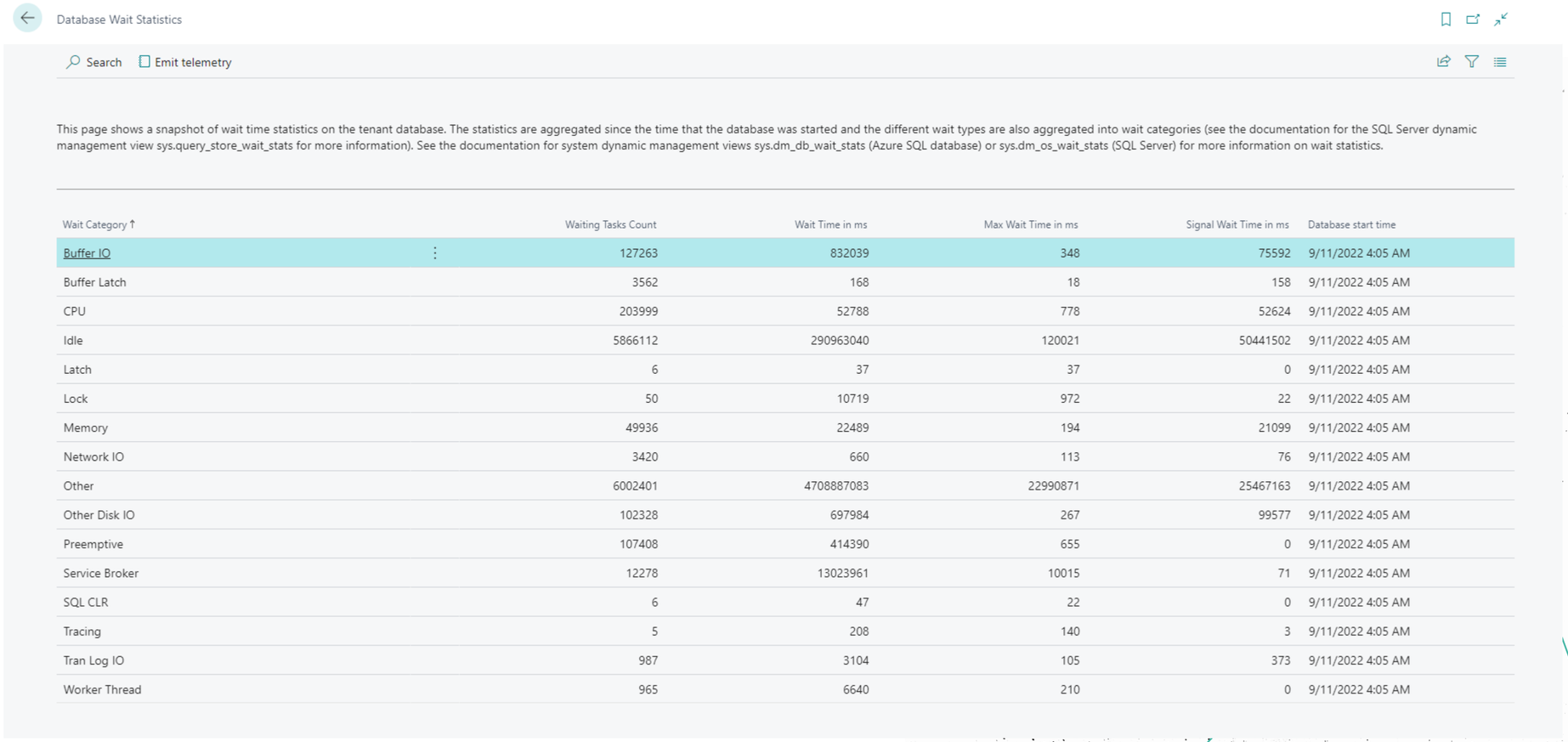
SQL Session ID	Table name	SQL Lock Resource Type	SQL Lock Request Mode	SQL Lock Request Status	User Name	Executing AL Object Type	Executing AL Object ID	Executing AL Object Name	Executing AL Method	Executing AL Object Extension Name
65	Item	KEY	Update	Granted	NP538433		50100	Item Ext.	TriggerLock - OnAction	TechDaysDemo
65	Item	KEY	Update	Granted	NP538433		50100	Item Ext.	TriggerLock - OnAction	TechDaysDemo
65	Item	KEY	Update	Granted	NP538433		50100	Item Ext.	TriggerLock - OnAction	TechDaysDemo

New feature


RUN AND DEBUG
Microsoft cloud sand
VARIABLES
Globals
Locals
Database Statistics
<Database Statistics>
Current SQL Latency (ms): 1.1361
Number of SQL Executes: 127
Number of SQL Row Reads: 183
Locks
Customer: Access: Update (2 KEY)
<Last Executed SQL Statements>

Database wait statistics

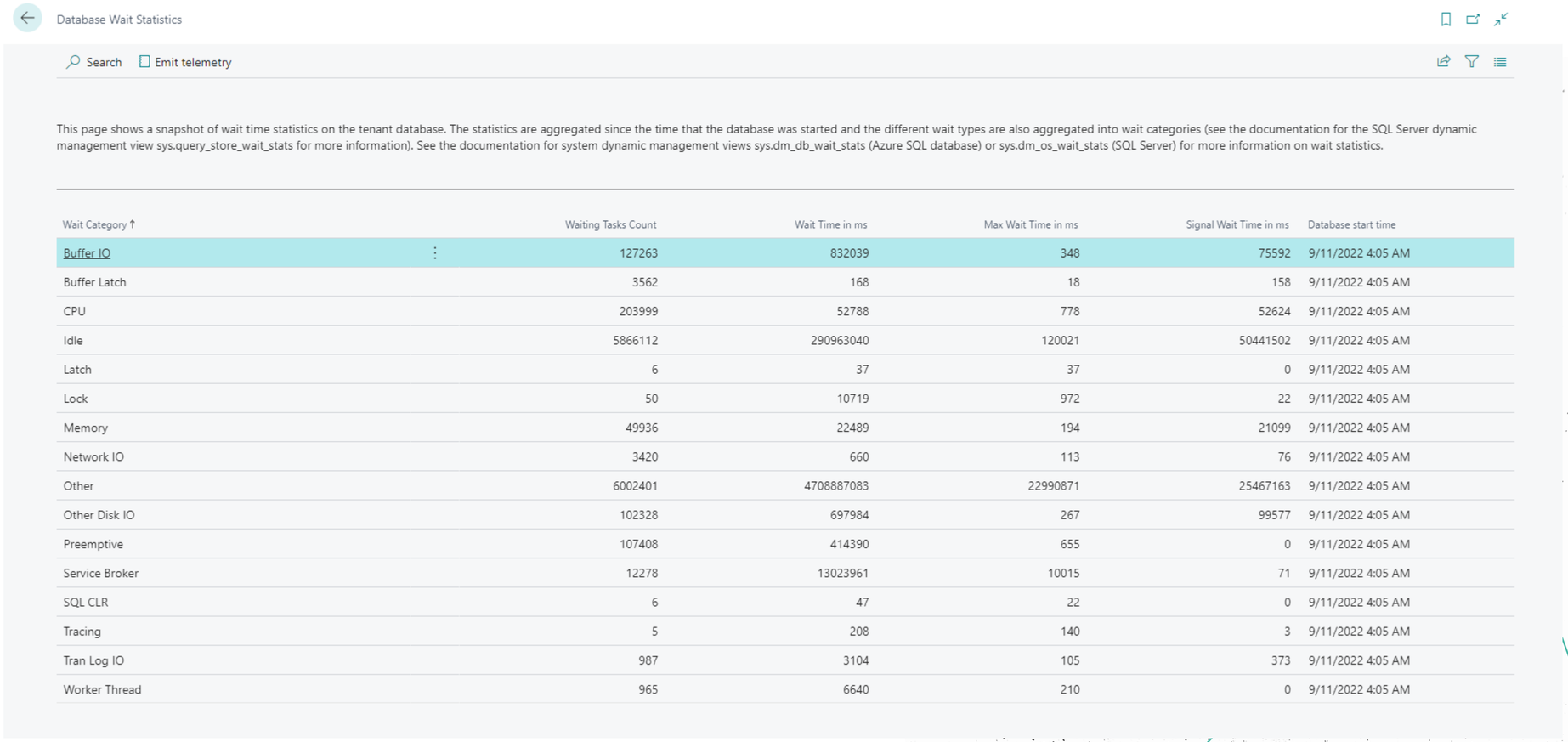
- sys.dm_os_wait_stats
- Resource waits, Queue waits, External waits



Wait Category	Waiting Tasks Count	Wait Time in ms	Max Wait Time in ms	Signal Wait Time in ms	Database start time
Buffer IO	127263	832039	348	75592	9/11/2022 4:05 AM
Buffer Latch	3562	168	18	158	9/11/2022 4:05 AM
CPU	203999	52788	778	52624	9/11/2022 4:05 AM
Idle	5866112	290963040	120021	50441502	9/11/2022 4:05 AM
Latch	6	37	37	0	9/11/2022 4:05 AM
Lock	50	10719	972	22	9/11/2022 4:05 AM
Memory	49936	22489	194	21099	9/11/2022 4:05 AM
Network IO	3420	660	113	76	9/11/2022 4:05 AM
Other	6002401	4708887083	22990871	25467163	9/11/2022 4:05 AM
Other Disk IO	102328	697984	267	99577	9/11/2022 4:05 AM
Preemptive	107408	414390	655	0	9/11/2022 4:05 AM
Service Broker	12278	13023961	10015	71	9/11/2022 4:05 AM
SQL CLR	6	47	22	0	9/11/2022 4:05 AM
Tracing	5	208	140	3	9/11/2022 4:05 AM
Tran Log IO	987	3104	105	373	9/11/2022 4:05 AM
Worker Thread	965	6640	210	0	9/11/2022 4:05 AM

Database wait statistics

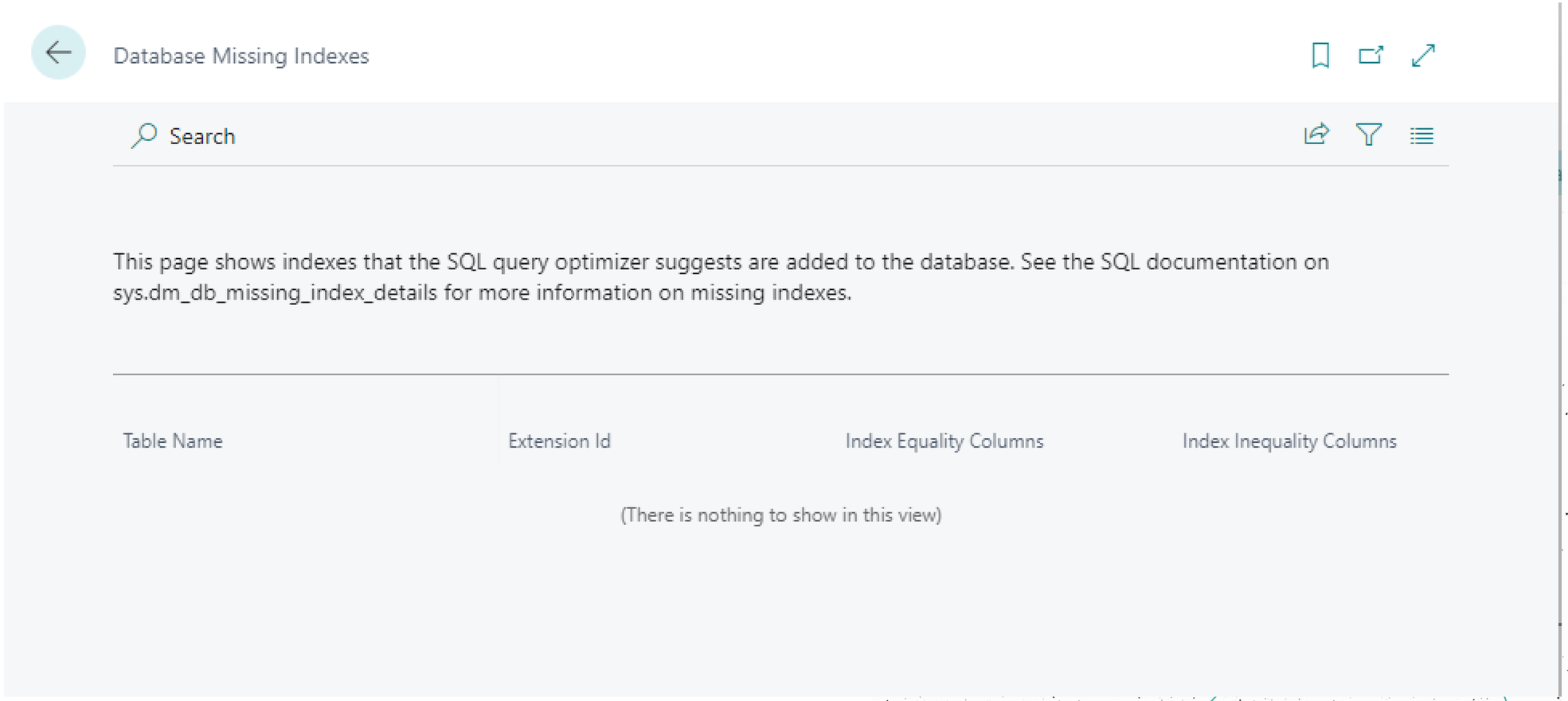
- sys.dm_os_wait_stats
- Resource waits, Queue waits, External waits



Wait Category	Waiting Tasks Count	Wait Time in ms	Max Wait Time in ms	Signal Wait Time in ms	Database start time
Buffer IO	127263	832039	348	75592	9/11/2022 4:05 AM
Buffer Latch	3562	168	18	158	9/11/2022 4:05 AM
CPU	203999	52788	778	52624	9/11/2022 4:05 AM
Idle	5866112	290963040	120021	50441502	9/11/2022 4:05 AM
Latch	6	37	37	0	9/11/2022 4:05 AM
Lock	50	10719	972	22	9/11/2022 4:05 AM
Memory	49936	22489	194	21099	9/11/2022 4:05 AM
Network IO	3420	660	113	76	9/11/2022 4:05 AM
Other	6002401	4708887083	22990871	25467163	9/11/2022 4:05 AM
Other Disk IO	102328	697984	267	99577	9/11/2022 4:05 AM
Preemptive	107408	414390	655	0	9/11/2022 4:05 AM
Service Broker	12278	13023961	10015	71	9/11/2022 4:05 AM
SQL CLR	6	47	22	0	9/11/2022 4:05 AM
Tracing	5	208	140	3	9/11/2022 4:05 AM
Tran Log IO	987	3104	105	373	9/11/2022 4:05 AM
Worker Thread	965	6640	210	0	9/11/2022 4:05 AM

Missing indexes

- sys.dm_db_missing_index_details



Consideration

- Based on estimates during optimization of a single query
- Suggestion doesn't specify the order for key columns
- No cost-benefit analysis regarding the size on included columns
- Not working for trivial query plans

Consideration

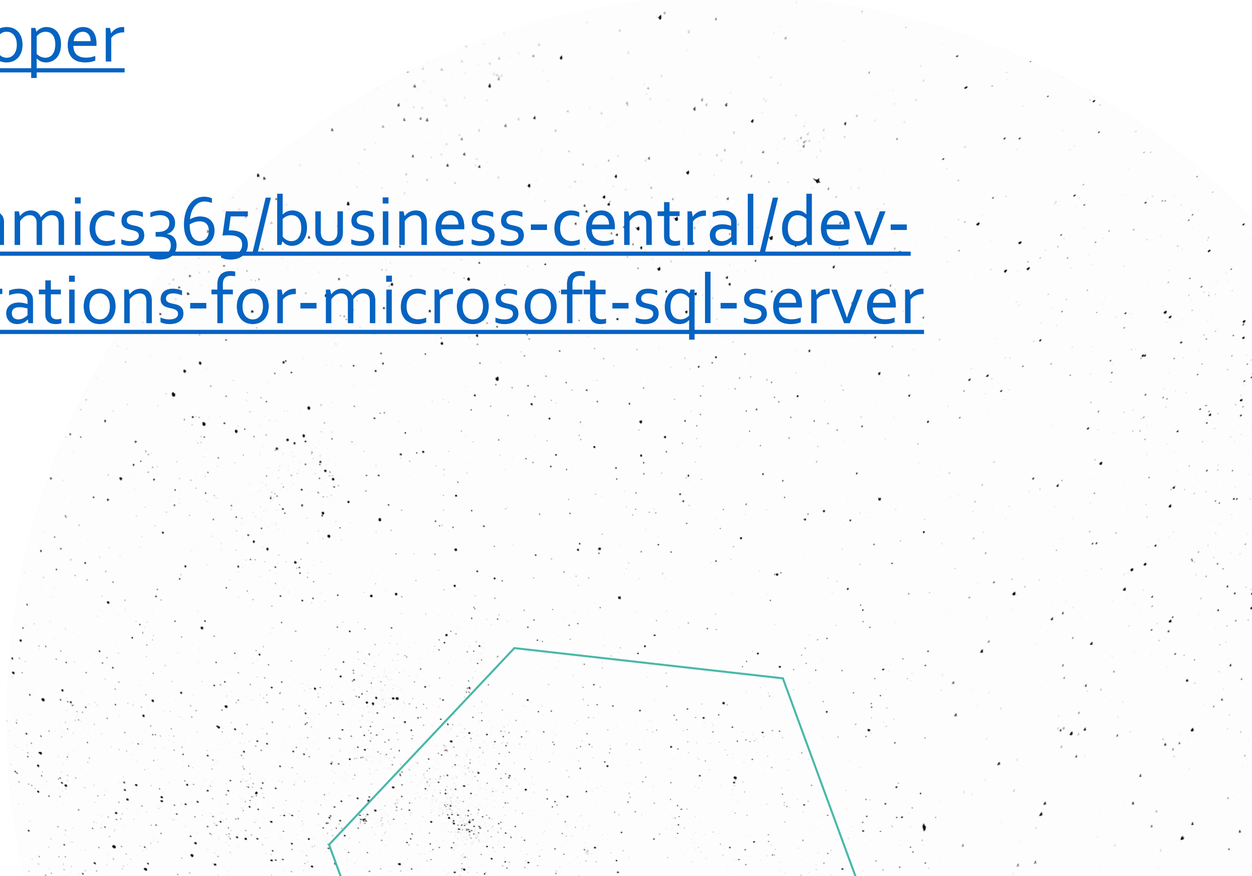
- Based on estimates during optimization of a single query
- Suggestion doesn't specify the order for key columns
- No cost-benefit analysis regarding the size on included columns
- Not working for trivial query plans

Tune nonclustered indexes

- Review the table structure
- Review existing indexes for overlap
- Verify if index change is successful

Useful links

- https://aka.ms/bcperformance
- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/performance/performance-developer
- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/deployment/installation-considerations-for-microsoft-sql-server



Any Questions?

BC/NAV TECH DAYS 2022
mibuso.com

Thank You!