

View-uri indexate. Indecși Columnstore. Fragmentare

SEMINAR 7

View-uri indexate

- Crearea unui **index clustered** pe un **view** îmbunătățește performanța interogărilor deoarece view-ul este stocat în baza de date în aceeași manieră în care este stocat un **tabel pe care este definit un index clustered**
- **Optimizatorul** (*query optimizer*) poate folosi view-uri indexate pentru a **mări** viteza de execuție a **interogărilor**
- View-ul nu trebuie să fie referit în interogare pentru ca optimizatorul să ia în considerare acel view

View-uri indexate

- Pentru a putea crea un **view indexat**, trebuie să creăm în primul rând un view cu opțiunea **WITH SCHEMABINDING**
- După ce am creat view-ul cu opțiunea **WITH SCHEMABINDING**, vom crea un **index clustered unic** pe acest view
- După acest pas, view-ul devine un **view indexat**, adică **un view care conține date**
- Putem crea și **alți indecși nonclustered** pe **view-ul indexat**
- Clauza **SCHEMABINDING** previne orice modificare a tabelelor referite în view care ar afecta definiția view-ului
- Dacă este folosită clauza **SCHEMABINDING**, instrucțiunea SELECT din definiția view-ului va conține în mod obligatoriu numele tabelelor, view-urilor sau funcțiilor definite de către utilizator în format '**nume_schemă.nume_obiect**'

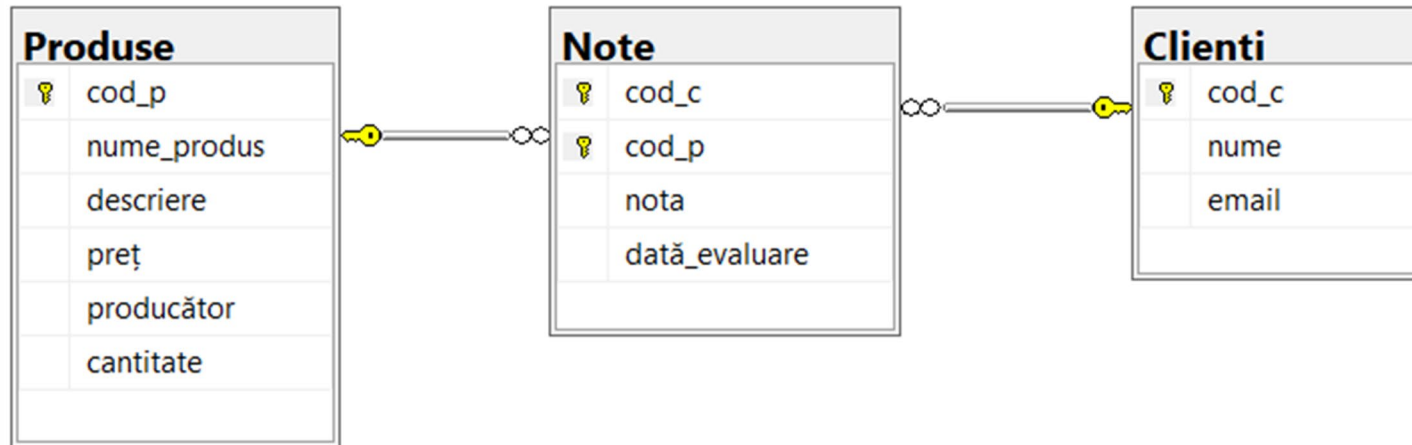
View-uri indexate

- Pentru a ne asigura că un view indexat este întreținut corect și va returna rezultate consistente, sunt necesare anumite valori fixe pentru următoarele opțiuni SET:

SET options	Required value	Default server value
ANSI_NULLS	ON	ON
ANSI_PADDING	ON	ON
ANSI_WARNINGS	ON	ON
ARITHABORT	ON	ON
CONCAT_NULL_YIELDS_NULL	ON	ON
NUMERIC_ROUNDABORT	OFF	OFF
QUOTED_IDENTIFIER	ON	ON

View-uri indexate

- Următoarele trei tabele (*Produse*, *Note*, *Clienți*) vor fi folosite ulterior în definiția unui view indexat:



View-uri indexate

- Exemplu de creare a unui view indexat:

```
-- Definirea view-ului
```

```
CREATE VIEW vw_NoteProduce WITH SCHEMABINDING AS  
  
SELECT C.num, C.email, N.nota, N.data_evaluate, P.num_produ FROM  
  
dbo.Clienți C INNER JOIN dbo.Note N ON C.cod_c=N.cod_c INNER JOIN  
  
dbo.Produse P ON N.cod_p=P.cod_p;  
  
GO
```

```
-- Definirea indexului clustered unic
```

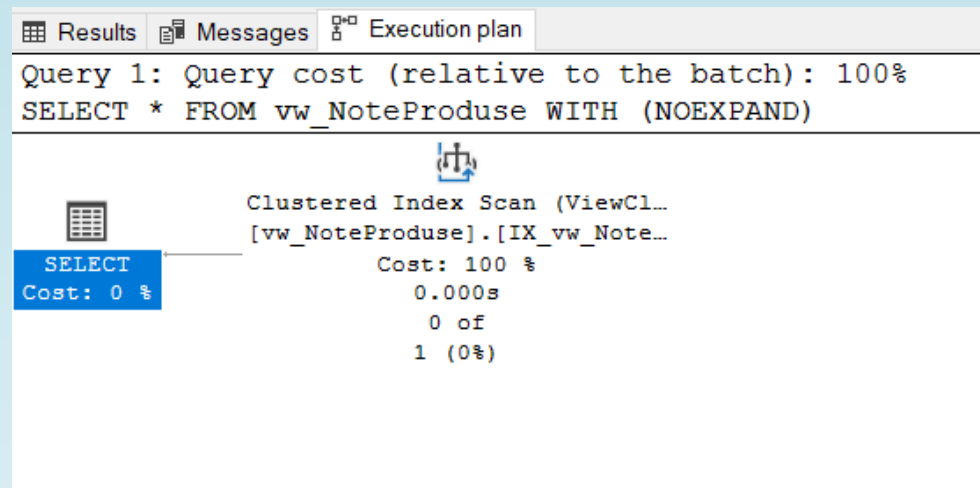
```
CREATE UNIQUE CLUSTERED INDEX IX_vw_NoteProduce  
  
ON vw_NoteProduce (num_produ,num,email);  
  
GO
```

- **!!!Un view indexat poate referi doar tabele care aparțin bazei de date în care se află view-ul**

View-uri indexate

- Exemplu de interogare a view-ului indexat:

```
SELECT * FROM vw_NoteProduse WITH (NOEXPAND);
```



- !!!Dacă **nu** folosim versiunea Microsoft SQL Server Enterprise, poate fi necesar să specificăm hint-ul NOEXPAND în interogare, dacă dorim să fie utilizat view-ul indexat în cadrul execuției interogării

View-uri indexate – Restricții

- Instrucțiunea SELECT din definiția view-ului indexat **nu** poate conține următoarele:
 - OUTER JOINS
 - SELF JOINS
 - Subinterogări
 - EXCEPT, INTERSECT, UNION
 - TOP
 - DISTINCT
 - ORDER BY
 - COUNT, MIN, MAX, STDEV, STDEVP, VAR, VARP, AVG, CHECKSUM_AGG
 - Funcții nedeterministe
 - Tabele derivate (specificate cu ajutorul unei instrucțiuni SELECT în clauza FROM)
 - Variabile tabel
 - View-uri

CASE

- O **expresie CASE** evaluează o listă de condiții și returnează una din multiplele expresii rezultat posibil
- **CASE** poate fi specificat în instrucțiuni SELECT, UPDATE, DELETE, SET dar și în clauzele WHERE, IN, HAVING și ORDER BY
- Expresia CASE are **două** variante:
 - **Simple CASE** (care compară o expresie cu o mulțime de expresii simple pentru a determina rezultatul)
 - **Searched CASE** (care evaluează o mulțime de expresii booleene pentru a determina rezultatul)
- Pentru ambele variante se poate specifica și **ELSE** (care este opțional)

CASE

- Sintaxa expresiei **simple CASE**:

```
CASE input_expression
```

```
    WHEN when_expression THEN result_expression [ ...n ]
```

```
    [ ELSE else_result_expression ]
```

```
END
```

- Expresia **simple CASE** funcționează astfel:
 - Se compară prima expresie cu expresia din fiecare clauză **WHEN** specificată (se verifică echivalența)
 - Dacă expresiile sunt echivalente, se va returna expresia din clauza **THEN**

CASE

- Este permisă doar o verificare de **egalitate**
- Pentru fiecare clauză **WHEN** specificată, se evaluează **input_expression = when_expression** (în ordinea specificată)
- Se returnează **result_expression** ce corespunde **primei comparații** **input_expression = when_expression** evaluate ca **TRUE**
- Dacă **niciuna** dintre comparații **nu** este evaluată ca **TRUE**:
 - se returnează **else_result_expression** (dacă este specificată **clauza ELSE**)
 - se returnează **NULL** (dacă **nu** este specificată **clauza ELSE**)

CASE

- Exemplu de expresie **simple CASE** folosită într-o instrucțiune **SELECT**:

```
SELECT nume, preț, sezon=CASE sezon  
    WHEN 'I' THEN N'Iarnă'  
    WHEN 'P' THEN N'Primăvară'  
    WHEN 'V' THEN N'Vară'  
    WHEN 'T' THEN N'Toamnă'  
    ELSE N'Nespecificat'  
END  
FROM Haine;
```

CASE

- Sintaxa expresiei **searched CASE**:

CASE

```
WHEN Boolean_expression THEN result_expression [ ...n ]  
[ ELSE else_result_expression ]
```

END

- Expresia **searched CASE** funcționează astfel:
 - Se evaluează, în ordinea specificată, *Boolean_expression* pentru fiecare clauză **WHEN**
 - Se returnează **result_expression** ce corespunde **primei** *Boolean_expression* care este evaluată ca **TRUE**

CASE

- Dacă **nicio** *Boolean_expression* **nu** este evaluată ca **TRUE**:
 - se returnează **else_result_expression** (dacă este specificată **clauza ELSE**)
 - se returnează **NULL** (dacă **nu** este specificată **clauza ELSE**)
- Expresia CASE **nu** poate fi folosită pentru a controla fluxul execuției:
 - Instrucțiunilor Transact-SQL
 - Blocurilor de instrucțiuni
 - Funcțiilor definite de către utilizator
 - Procedurilor stocate

CASE

- Exemplu de expresie **searched CASE** folosită într-o instrucțiune **SELECT**:

```
SELECT nume, preț, [gama de prețuri]=CASE  
    WHEN preț=0 THEN 'gratuit'  
    WHEN preț<50 THEN 'sub 50 lei'  
    WHEN preț>=50 AND preț<=150 THEN N'între 50 și 150 lei'  
    ELSE 'peste 150 lei'  
END  
FROM Haine;
```

CASE

- Exemplu de expresie **searched CASE** folosită într-o instrucțiune **UPDATE**:

```
UPDATE Angajați SET salariu=(CASE  
  
WHEN (DATEDIFF(YEAR,data_angajării,GETDATE())) < 2) THEN salariu  
  
WHEN (DATEDIFF(YEAR,data_angajării,GETDATE())) BETWEEN 2 AND 4) THEN salariu + 400  
  
WHEN (DATEDIFF(YEAR,data_angajării,GETDATE())) BETWEEN 5 AND 7) THEN salariu + 600  
  
ELSE salariu + 800  
  
END)
```

```
OUTPUT deleted.nume, DATEDIFF(YEAR,deleted.data_angajării,GETDATE()) [vechime],  
deleted.salariu [salariu vechi], inserted.salariu [salariu nou];
```


Indecși Columnstore

- **Indexul columnstore** grupează și stochează datele în funcție de **coloane**, nu de înregistrări
- Indecșii columnstore pot fi **clustered** sau **nonclustered**
- Sunt foarte potriviți pentru *data warehouses*
- Interogări de 10 ori mai performante (față de stocarea tradițională în funcție de înregistrări)
- Compresie a datelor de 10 ori mai bună (față de dimensiunea datelor necomprimate)
- La crearea unui index columnstore nu se poate specifica ordine descrescătoare sau crescătoare pentru coloanele din index, deoarece datele sunt stocate într-o manieră care să îmbunătățească compresia și performanța

Indecși Columnstore

- Index Rowstore

Rândul 1	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 2	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 3	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 4	Col1	Col2	Col3	Col4	Col5	Col6	Col7

Pagina 1

Rândul 5	Col1	Col2	Col3	Col4	Col5	Col6	Col7
...	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul n	Col1	Col2	Col3	Col4	Col5	Col6	Col7

Pagina 2

Indecși Columnstore

- Index Columnstore

Rândul 1	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 2	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 3	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 4	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul 5	Col1	Col2	Col3	Col4	Col5	Col6	Col7
...	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Rândul n	Col1	Col2	Col3	Col4	Col5	Col6	Col7

Pagina 1 Pagina 2 Pagina 3 Pagina 4 Pagina 5 Pagina 6 Pagina 7

Indecși Columnstore

- Exemplu de creare a unui index columnstore clustered:

```
CREATE CLUSTERED COLUMNSTORE INDEX  
CCIX_Persoane ON Persoane;
```

- Exemplu de creare a unui index columnstore nonclustered:

```
CREATE NONCLUSTERED COLUMNSTORE INDEX  
NCIX_Produse_num_e_produ_s_preț ON Produse  
(num_e_produ_s, preț);
```

Indecși Columnstore

- Un tabel poate avea atât indecși *rowstore*, cât și indecși *columnstore* (maxim un index *columnstore* pe tabel)
- După crearea unui index *columnstore nonclustered*, tabelul devine *read-only* și nu mai poate fi modificat (SQL Server 2014)
- Începând cu SQL Server 2016, crearea unui index *columnstore nonclustered* permite modificarea datelor din tabel (tabelul **nu** devine *read-only*)
- Crearea unui index *columnstore clustered* **nu** împiedică modificarea datelor din tabel, dar **împiedică** crearea **altor indecși** pe tabelul respectiv (SQL Server 2014)
- Începând cu SQL Server 2016, crearea unui index *columnstore clustered* **nu** împiedică crearea altor indecși *rowstore nonclustered* pe tabelul respectiv

Fragmentare

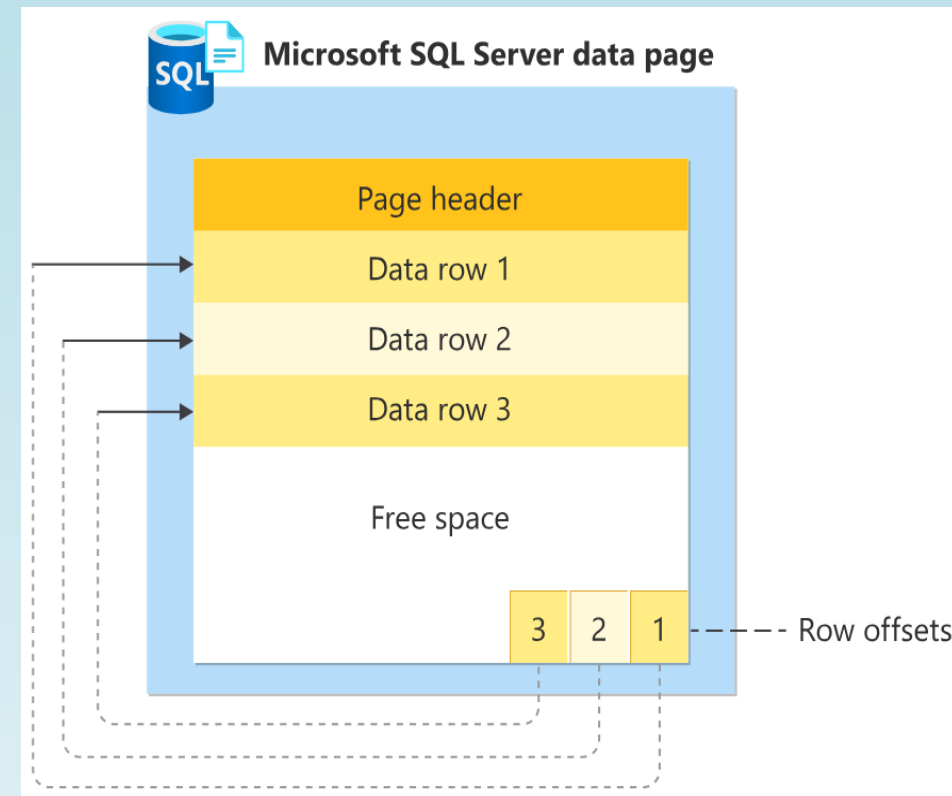
- Unitatea fundamentală de stocare a datelor din SQL Server este pagina
- Spațiul alocat pe disc unui fișier de date (.mdf sau .ndf) într-o bază de date este divizat logic în pagini numerotate contiguu de la 0 la n
- Operațiile I/O au loc la nivel de pagină (SQL Server scrie sau citește pagini întregi de date)
- O zonă (extent) conține 8 pagini de date fizic contigue, sau 64 KB
- Fragmentarea **internă** are loc atunci când există spațiu nefolosit între înregistrările din aceeași pagină
- Spațiul nefolosit dintre înregistrările aflate pe aceeași pagină cauzează utilizarea deficitară a *cache*-ului și mai multe operații I/O, iar acestea duc la performanța slabă a interogărilor

Fragmentare

- Fragmentarea **externă** are loc atunci când stocarea fizică a paginilor și a zonelor pe disc **nu** este contiguă
- Când zonele (extents) unui tabel **nu** sunt stocate contiguu pe disc, trecerea de la o zonă la alta generează rotații mai mari ale discurilor
- Fragmentarea logică
 - Fiecare pagină din index este legată de pagina anterioară și de cea următoare în ordinea logică a datelor din coloane
 - Când are loc o diviziune a unei pagini (*page split*), paginile se transformă în pagini **out-of-order**
 - O pagină este *out-of-order* atunci când pagina următoare la nivel fizic **nu** este pagina următoare la nivel logic

Fragmentare

- Fiecare pagină de date conține un **header**, înregistrări, spațiu liber și un tabel **row offset** care conține câte o intrare pentru fiecare înregistrare de pe pagină
- Fiecare intrare din tabelul **row offset** înregistrează cât de departe este primul byte al înregistrării față de începutul paginii
- Intrările din tabelul **row offset** se află în ordine inversă față de înregistrările de pe pagină
- O pagină are dimensiunea de 8 KB



Fragmentare

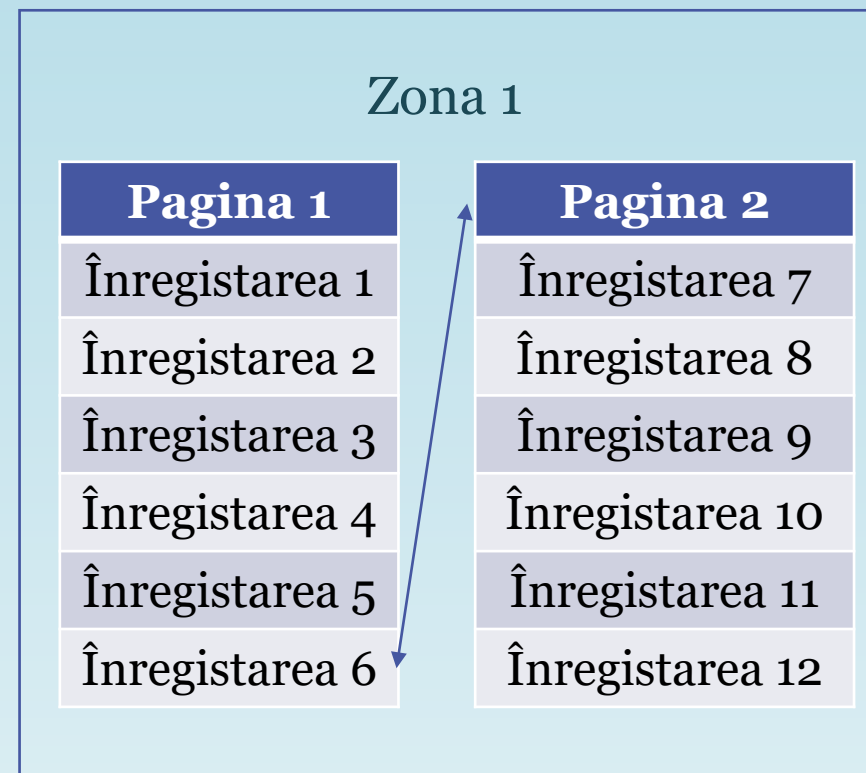
- Funcția sistem **sys.dm_db_index_physical_stats** oferă informații despre dimensiunea și fragmentarea datelor și indecșilor pentru tabelul sau view-ul specificat
- Dacă dorim să aflăm aceste informații pentru tabelul *Produse*, vom executa următoarea interogare:

```
SELECT * FROM sys.dm_db_index_physical_stats  
  
(DB_ID(N'MagazinOnline'), OBJECT_ID(N'MagazinOnline.dbo.Produse'),  
  
NULL, NULL, 'DETAILED');
```

- Coloana **avg_fragmentation_in_percent** conține valoarea procentuală a fragmentării externe
- Coloana **avg_page_space_used_in_percent** conține valoarea procentuală medie a spațiului de stocare folosit în toate paginile

Fragmentare

- Cereri de citire a paginilor – 2
- Schimbări de zonă – 0
- Spațiu utilizat de către tabel – 16 KB
- avg_fragmentation_in_percent - 0
- avg_page_space_used_in_percent - 100



Fragmentare

- Pentru a reduce fragmentarea în *heap* se va crea un index *clustered* pe tabel
- La crearea indexului *clustered*, înregistrările sunt rearanjate într-o anumită ordine, iar paginile sunt așezate contiguu pe disc
- Pentru a reduce fragmentarea unui index, se pot efectua următoarele acțiuni:
 - Dacă **avg_fragmentation_in_percent** este cuprins între 5% și 30%, se va folosi comanda **ALTER INDEX REORGANIZE** (reordonează paginile de la nivelul nodurilor terminale în ordine logică)
 - Dacă **avg_fragmentation_in_percent** este mai mare decât 30%, se va folosi comanda **ALTER INDEX REBUILD** (indexul va fi șters și recreat) sau pur și simplu se va șterge și crea din nou indexul
 - Se poate elimina și crea din nou indexul clustered (în cazul în care indexul clustered este creat din nou, datele vor fi redistribuite în pagini pline)

Fragmentare

- Nivelul de umplere a unei pagini poate fi specificat la crearea indexului folosind opțiunea **FILLFACTOR**
- Când un index este creat sau recreat, valoarea specificată cu ajutorul opțiunii **FILLFACTOR** determină procentul de spațiu care trebuie umplut cu date pentru fiecare pagină de date din index de la nivelul nodurilor terminale, rezervând restul spațiului de pe pagină pentru adăugări ulterioare
- De exemplu, dacă se specifică o valoare de 80 pentru opțiunea **FILLFACTOR**, 20% din spațiul de pe fiecare pagină din index de la nivelul nodurilor terminale va fi lăsat liber, oferind spațiu pentru extinderea indexului pe măsură ce date noi sunt adăugate în tabel

Fragmentare

- Exemplu de creare a unui index cu opțiunea **FILLFACTOR** specificată:

```
CREATE INDEX IX_Haine_numa_ASC_preț_DESC  
ON Haine (numa ASC, preț DESC)  
WITH (FILLFACTOR=80);
```

- Exemplu de reactivare a unui index dezactivat cu opțiunea **FILLFACTOR** specificată:

```
ALTER INDEX IX_Haine_preț_ASC ON Haine  
REBUILD WITH (FILLFACTOR=70);
```

Bibliografie

- <https://learn.microsoft.com/en-us/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/t-sql/language-elements/case-transact-sql?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/relational-databases/views/create-indexed-views?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16>
- <https://learn.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-design-guidance?view=sql-server-ver16>