View-uri indexate. Limbaj de control al fluxului

Seminar 7

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 optiunea **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

- Exemplu de creare a unui view indexat:

```
-- Definirea view-ului

CREATE VIEW vw_Produse_Prețuri WITH SCHEMABINDING AS

SELECT nume_produs, preț, preț/2 AS preț_redus FROM
dbo.Produse;

-- Definirea indexului clustered unic

CREATE UNIQUE CLUSTERED INDEX IX_vw_Produse_Prețuri ON
vw_Produse_Prețuri (nume_produs);
```

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

View-uri indexate – Restricții

- Instrucțiunea SELECT din definiția view-ului indexat nu poate conține următoarele:
  - OUTER 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)
  - View-uri

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 read-only)
- 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

- Exemplu de creare a unui index columnstore clustered:

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

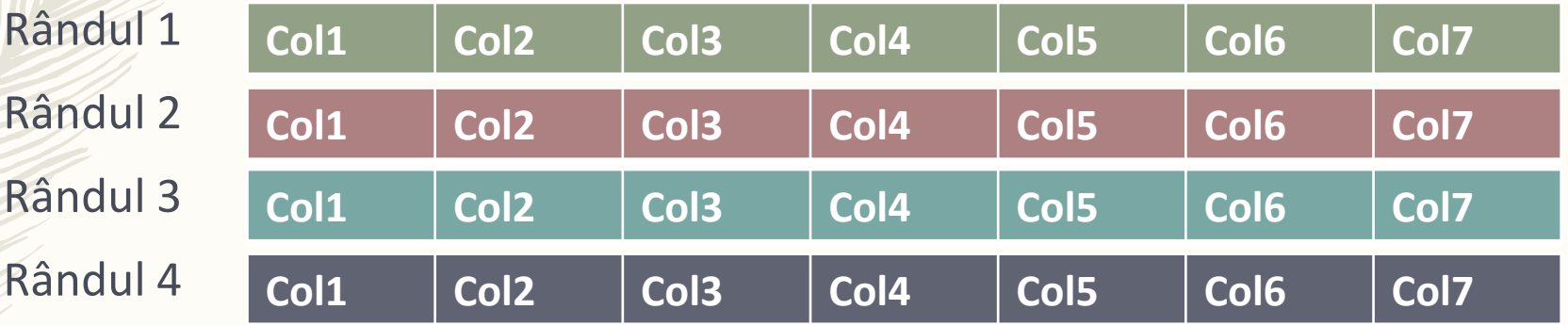
- Exemplu de creare a unui index columnstore nonclustered:

```
CREATE NONCLUSTERED COLUMNSTORE INDEX
NCIX_Produse_nume_produs_preț ON Produse(nume_produs,
preț);
```

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

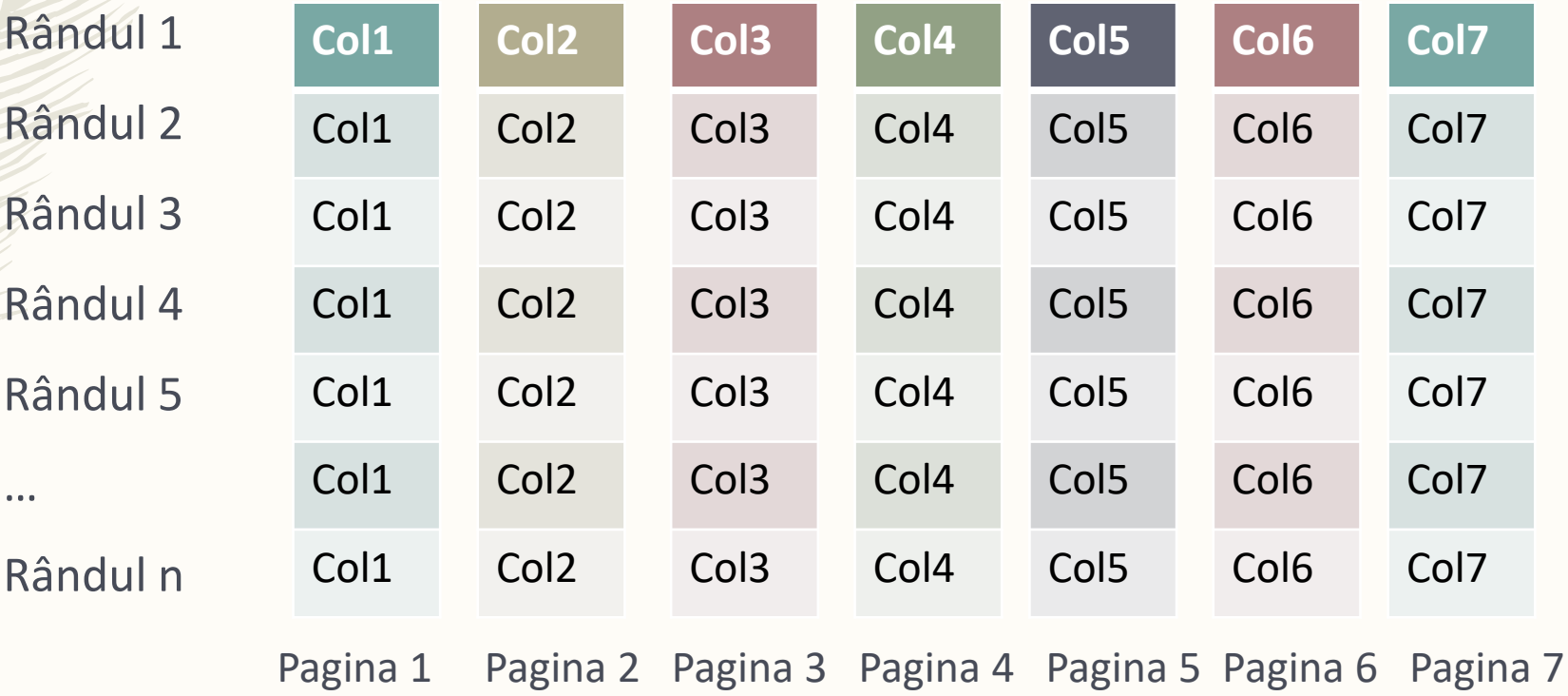
Indecși Columnstore

- Index Rowstore



Indecși Columnstore

- Index Columnstore



Fragmentare

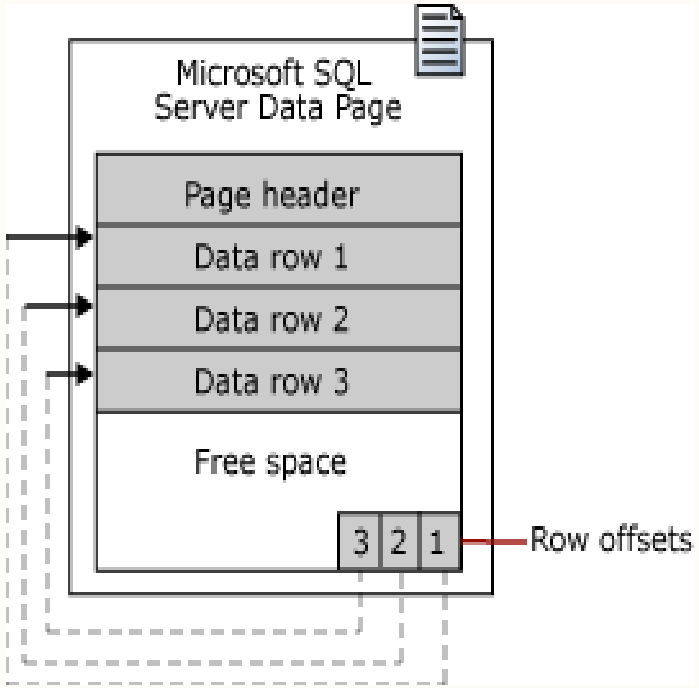
- Unitatea fundamentală de stocare a datelor din SQL Server este pagina
- Spațiul alocat pe disc unui fișier de date (.mdf sau .ndf) într-o bază de date este divizat logic în pagini numerotate contiguu de la 0 la n
- Operațiile I/O au loc la nivel de pagină (SQL Server scrie sau citește pagini întregi 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

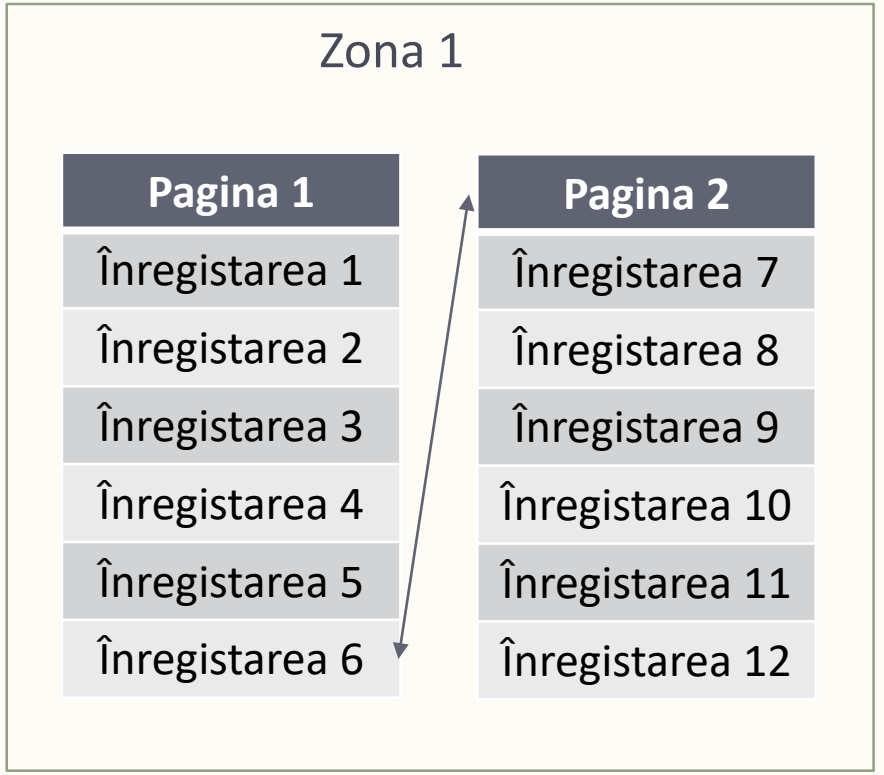
- Funcția sistem **sys.dm_db_index_physical_stats** oferă informații despre dimensiunea și fragmentarea datelor și indecșilor pentru tabelul sau view-ul specificat
- Dacă dorim să aflăm aceste informații pentru tabelul *Produse*, vom executa următoarea interogare:

```
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

Limbaj de control al fluxului

- Transact-SQL oferă un set de cuvinte speciale, numit limbaj de control al fluxului care pot fi folosite pentru a controla fluxul execuției instrucțiunilor Transact-SQL, al blocurilor de instrucțiuni, al funcțiilor definite de utilizator și al procedurilor stocate
- În lipsa limbajului de control al fluxului, instrucțiunile Transact-SQL se execută în ordine secvențială
- BEGIN … END – delimitează un grup de instrucțiuni SQL care se execută împreună
- Blocurile BEGIN … END pot fi încorporate

Limbaj de control al fluxului

- Sintaxa:

```
BEGIN
        { sql_statement | sql_block}
END
```

- RETURN – iese necondiționat dintr-o interogare sau dintr-o procedură stocată
- Poate fi folosit în orice punct pentru a ieși dintr-o procedură, batch sau bloc de instrucțiuni
- Sintaxa:

```
RETURN [integer_expression]
```

- Se poate folosi pentru a returna status codes - procedurile stocate returnează zero (success) sau o valoare întreagă diferită de zero (failure)

Limbaj de control al fluxului

- Exemplu de procedură stocată care returnează status codes:

```
CREATE PROCEDURE usp_verifica_status @cod_student INT
AS
BEGIN
IF (SELECT judet FROM Studenti WHERE cod_student=@cod_student)='Cluj'
        RETURN 1;
ELSE
        RETURN 2;
END;
---------------
DECLARE @status INT;
EXEC @status=usp_verifica_status 1;
SELECT 'Status'=@status;
```

Limbaj de control al fluxului

- IF … ELSE – condiționează execuția unei instrucțiuni SQL sau a unui bloc de instrucțiuni SQL
- Sintaxa:

```
IF Boolean_expression
 { sql_statement | statement_block }
[ ELSE
 { sql_statement | statement_block } ]
```

Limbaj de control al fluxului

- WHILE – setează o condiție pentru execuția repetată a unei instrucțiuni SQL sau a unui bloc de instrucțiuni SQL
- Sintaxa:

```
WHILE Boolean_expression
      { sql_statement | statement_block | BREAK | CONTINUE }
```

Limbaj de control al fluxului

- BREAK – iese din cea mai interioară buclă WHILE sau dintr-o instrucțiune IF… ELSE din interiorul unei bucle WHILE
- CONTINUE – cauzează reînceperea buclei WHILE, ignorând toate instrucțiunile care apar după CONTINUE
- GOTO – execută un salt în execuție la o porțiune din cod marcată printr-un label

```
Label: -- some SQL statements
GOTO Label;
```

Limbaj de control al fluxului

- WAITFOR – blochează execuția unui batch, tranzacție sau procedură stocată până când un interval de timp sau timp specificat este atins sau o instrucțiune specificată modifică sau returnează cel puțin o înregistrare
- Sintaxa:

```
WAITFOR
{
DELAY 'time_to_pass' | TIME 'time_to_execute' |
[ ( receive_statement ) | (
get_conversation_group_statement ) ]
[ , TIMEOUT timeout ]}
```

Limbaj de control al fluxului

- În funcție de nivelul activității pe server, timpul de așteptare poate varia, deci poate fi mai mare decât timpul specificat în **WAITFOR**
- Exemple:

```
-- execuția continuă la 22:00
WAITFOR TIME '22:00';
-- execuția continuă peste 3 ore
WAITFOR DELAY '03:00';
```

Limbaj de control al fluxului

- THROW – aruncă o excepție și transferă execuția unui bloc CATCH dintr-o construcție TRY … CATCH
- Severitatea excepției este mereu setată pe valoarea 16
- Sintaxa:

```
THROW [ { error_number | @local_variable },
        { message | @local_variable },
        { state | @local_variable } ] [ ; ]
```

- Exemplu:

```
THROW 50002, 'Înregistrarea nu există! ', 1;
```

Limbaj de control al fluxului

- TRY … CATCH – implementează tratarea erorilor pentru Transact-SQL
- Captează toate erorile de execuție care au o severitate mai mare decât 10 și care nu închid conexiunea la baza de date
- Sintaxa:

```
BEGIN TRY
{ sql_statement | statement_block }
END TRY
BEGIN CATCH
[ { sql_statement | statement_block } ]
END CATCH
[ ; ]
```

Limbaj de control al fluxului

- În interiorul unui bloc **CATCH** pot fi folosite următoarele funcții sistem pentru a obține informații despre eroarea care a cauzat execuția blocului **CATCH**:
- **ERROR_NUMBER()** – returnează numărul erorii
- **ERROR_SEVERITY()** – returnează severitatea erorii
- **ERROR_STATE()** – returnează error state number
- **ERROR_PROCEDURE()** – returnează numele procedurii stocate sau al trigger-ului în care a avut loc eroarea
- **ERROR_LINE()** – returnează numărul liniei care a cauzat eroarea
- **ERROR_MESSAGE()** – returnează mesajul erorii

Limbaj de control al fluxului

Mesaje de eroare

- Error number (numărul erorii)
  - o valoare întreagă cuprinsă între 1 și 49999
  - Pentru erorile custom (definite de către utilizator) valoarea este cuprinsă între 50000 și 2147483647
- Error severity (severitatea erorii)
  - 26 de nivele de severitate
  - Erorile care au nivelul de severitate ≥ 16 sunt înregistrate în error log în mod automat
  - Erorile care au nivelul de severitate cuprins între 20 și 25 sunt fatale și închid conexiunea
- Error message (mesajul erorii) – NVARCHAR(2048)