

Seminar 4

Multiversionarea

Monitorizarea blocărilor

- *SQL Server Profiler*

- Interfață grafică pentru SQL Trace pentru monitorizarea unei instanțe Database Engine sau Analysis Services

- Evenimentele sunt salvate într-un fișier *trace* care poate fi mai apoi analizat sau folosit pentru a relua o serie de pași specifici în încercarea de a diagnostica o problemă

- *sp_lock* - oferă informații despre blocări

- *sys.dm_tran_locks* - returnează informații despre resursele lock manager active în prezent

Monitorizarea blocărilor

- *SQL Server Profiler*
- *sp_lock, sys.dm_tran_locks*
- *sys.dm_tran_active_transactions*
 - Tipuri de resurse:
 - RID – identificator de înregistrare
 - Key – interval de chei într-un index (blocări *key-range*)
 - Pagină – pagină de 8 KB din tabele/indecși
 - HoBT – *Heap or balanced tree*
 - Tabel, fișier, bază de date
 - Metadata
 - Aplicație

DBCC LOG

- DBCC LOG – returnează informații despre logul de tranzacții

DBCC LOG (<dbname>,<output id>)

- Output id: 0-4 precizează gradul de detaliere

Niveluri de izolare în SQL Server

- **READ UNCOMMITTED**: fără blocări la citire
- **READ COMMITTED**: menține blocările pe durata execuției instrucțiunii (default) (*Dirty reads*)
- **REPEATABLE READ**: menține blocările pe durata tranzacției (*Unrepeatable reads*)
- **SERIALIZABLE**: menține blocările și blocările *key range* pe durata întregii tranzacții (*Phantom reads*)
- **SNAPSHOT**: lucrează pe un *snapshot* al datelor
- sintaxă SQL:

```
SET TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED|READ  
COMMITTED|REPEATABLE READ|SNAPSHOT|SERIALIZABLE}
```

Multiversionarea

- Într-un SGBD cu multiversionare, fiecare scriere a unui item x produce o nouă copie (sau versiune) a lui x .
- La fiecare citire a lui x , SGBD selectează pentru citire una dintre versiunile lui x .
- Întrucât nu există suprascrieri (între operațiile de scriere), iar operațiile de citire pot citi orice versiune, SGBD are mai multă flexibilitate în controlul ordinii citirilor și scrierilor.

Versionarea la nivel de rând (RLV)

- Introdusă în SQL Server 2005
- Utilă când este nevoie de date *comise*, dar nu neapărat de *cea mai recentă versiune*
 - Read Committed Snapshot Isolation & Full Snapshot Isolation – cititorul nu se blochează niciodată. Accesează valoarea comisă anterior.
- Toate versiunile mai vechi sunt stocate în baza de date tempdb.
 - Pe baza versiunilor vechi se poate construi un *snapshot* al bazei de date.

Read Committed Snapshot Isolation

```
ALTER DATABASE MyDatabase  
SET READ_COMMITTED_SNAPSHOT ON
```

- Operațiunile văd înregistrări comise la începerea execuției lor =>
 - *snapshot* al datelor la nivel de comandă
 - citire consistentă la nivel de comandă
 - disponibil la utilizarea nivelului de izolare READ COMMITTED (default)

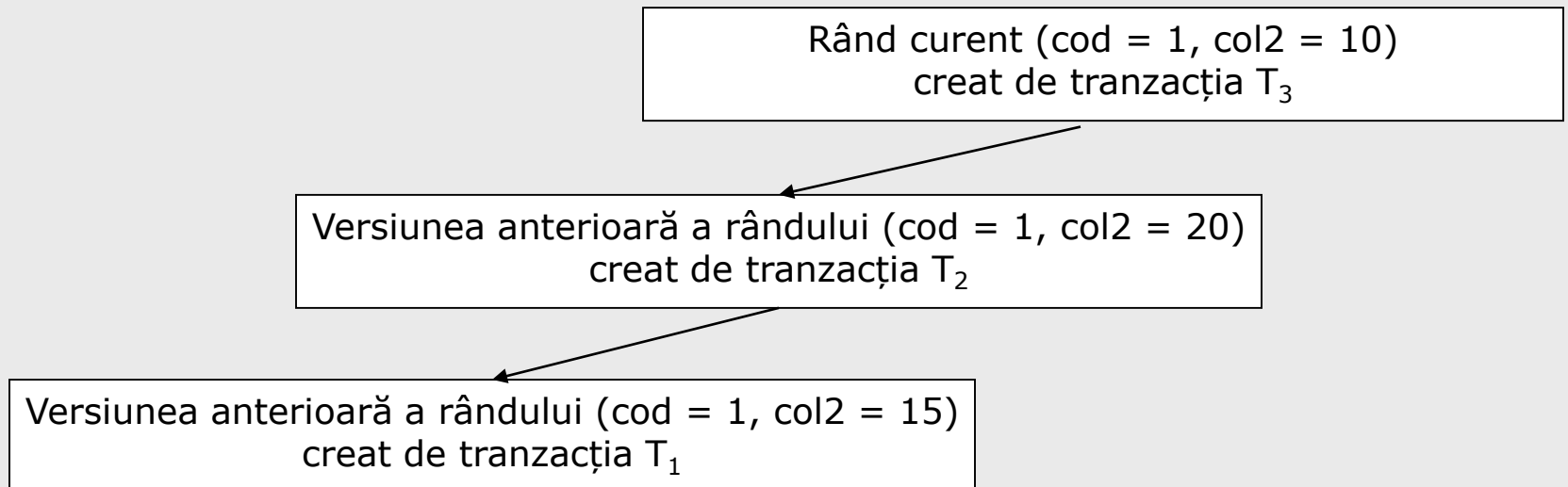
Full Snapshot Isolation

```
ALTER DATABASE MyDatabase  
SET ALLOW_SNAPSHOT_ISOLATION ON
```

- Operațiile văd înregistrări comise la începerea execuției tranzacției =>
 - *snapshot* al datelor la nivel de tranzacție
 - citire consistentă la nivel de tranzacție
 - nivelul de izolare SNAPSHOT

Versionare la nivel de rând

- O înregistrare conține TSN (*transaction sequence number*).
- Toate versiunile sunt stocate într-o listă înlănțuită.



Versionare la nivel de înregistrare

■ Avantaje

- Crește nivelul de concurență.
- Sporește performanța triggerelor / creării indecșilor.

■ Dezavantaje

- Cerințe de gestiune suplimentare pentru monitorizarea utilizării *tempdb*
- Performanță mai scăzută a operațiilor de modificare
- Viteza cititorilor este afectată de costul traversării listelor înlănțuite.
- Rezolvă conflictul între scriitor și cititori, dar scriitorii simultani tot nu sunt permisi.

Triggere & RLV

- Triggerele au acces la 2 *pseudo-tabele*:
 - tabelul '*deleted*' – conține rânduri șterse sau versiuni vechi ale rândurilor modificate;
 - tabelul '*inserted*' – conține rânduri inserate sau versiuni noi ale rândurilor modificate.
- Înainte de SQL Server 2005:
 - tabelul '*deleted*' creat pe baza logului de tranzații – afectează performanța
- Utilizând RLV:
 - Pentru tabele cu trigger relevante schimbările sunt versionate.

Crearea indecșilor & RLV

- În versiunile anterioare de SQL Server crearea sau reconstrucția indecșilor însemna:
 - tabel blocat exclusiv, date complet inaccesibile (index clustered);
 - tabel disponibil doar pentru citire; indexul nu este disponibil (index non-clustered).
- Cu versionare la nivel de rând:
 - Indecșii sunt creați și reconstruiți online.
 - Toate cererile sunt procesate pe date versionate.

Niveluri de izolare și anomalii de concurență

<i>Nivel de izolare</i>	Dirty Read	Non-repeat. Read	Phantom Read	Update Conflict	Model conc.
<i>Read Un-committed</i>	Da	Da	Da	Nu	Pesimist
<i>Read committed Locking</i>	Nu	Da	Da	Nu	Pesimist
<i>Read committed Snapshot</i>	Nu	Da	Da	Nu	Optimist
<i>Repeatable read</i>	Nu	Nu	Da	Nu	Pesimist
<i>Row-Level Versioning</i>	Nu	Nu	Nu	Da	Optimist
<i>Serializable</i>	Nu	Nu	Nu	Nu	Pesimist

PIVOT / UNPIVOT

- Schimbă o expresie *table-valued* într-un alt tabel.
- PIVOT rotește o expresie *table-valued* transformând valorile unice dintr-o coloană din expresie în mai multe coloane în output și calculează valori agregate acolo unde este necesar, pe valorile oricăror coloane rămase care sunt dorite în rezultatul final.
- UNPIVOT realizează operația opusă, rotind coloanele dintr-o expresie *table-valued* în valori de coloană.

PIVOT

```
SELECT <coloana nepivotata>,  
    [prima coloana pivotata] AS <nume coloana>,  
    [a doua coloana pivotata] AS <nume coloana>,  
    ...  
    [ultima coloana pivotata] AS <nume coloana>  
FROM  
    (<interogare SELECT care produce datele>) AS <interogare  
sursa>  
PIVOT  
(  
    <functie agregare>(<coloana agregata>)  
FOR  
    [<coloana care contine valorile care devin capete de  
coloane>]  
    IN ( [prima coloana pivotata], [a doua coloana pivotata],  
        ... [ultima coloana pivotata])  
) AS <alias pentru tabelul pivot>  
<clauza optionala ORDER BY>;
```


Clauza OUTPUT

- Oferă acces la înregistrările *inserate*, *modificate* sau *șterse*.
- Poate implementa anumite funcționalități care se pot realiza doar prin trigger.

```
UPDATE Cursuri
SET numec = 'Sisteme de gestiune a bazelor de date'
OUTPUT inserted.codc, deleted.numec, inserted.numec,
      GETDATE(), SUSER_SNAME()
INTO ModificariCursuri
WHERE codc = 3
```

Instrucțiunea MERGE

- MERGE – oferă posibilitatea de a compara rânduri dintr-un tabel sursă și un tabel destinație; pe baza rezultatului acestei comparații se pot executa comenzi INSERT, UPDATE sau DELETE.

Sintaxa generală MERGE

```
MERGE Definitie_Tabel AS Destinatie
USING (Tabel_sursa) AS Sursa
ON (Termeni de cautare)
WHEN MATCHED THEN
    UPDATE SET
        sau
    DELETE
WHEN NOT MATCHED [BY TARGET] THEN
    INSERT
WHEN NOT MATCHED BY SOURCE THEN
    UPDATE SET
        sau
    DELETE
```

Exemplu MERGE

Tabelul Carti

	CodCarte	Titlu	Autor	ISBN	Pagini
1	1	In cautarea timpului pierdut	Marcel Proust	NULL	NULL
2	2	In cautarea timpului pierdut	NULL	NULL	350
3	3	In cautarea timpului pierdut	NULL	9789731246420	NULL

Exemplu MERGE

MERGE Carti

USING

(SELECT MAX(CodCarte) CodCarte, Titlu, MAX(Autor) Autor,
MAX(ISBN) ISBN, MAX(Pagini) Pagini

FROM Carti

GROUP BY Titlu

) MergeData ON Carti.CodCarte = MergeData.CodCarte

WHEN MATCHED THEN

UPDATE SET Carti.Titlu = MergeData.Titlu,



Carti.Autor = MergeData.Autor,

Carti.ISBN = MergeData.ISBN,

Carti.Pagini = MergeData.Pagini

WHEN NOT MATCHED BY SOURCE THEN DELETE;

Exemplu MERGE

 Results  Messages					
	CodCarte	Titlu	Autor	ISBN	Pagini
1	3	In cautarea timpului pierdut	Marcel Proust	9789731246420	350