

Databases

Seminar 7

Indexes in SQL Server (II)

T-SQL - Control-of-Flow Language

Indexes in SQL Server (II)

- indexed views

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

- indexed views
 - restrictions
 - the view cannot reference other views
 - the definition of the view must be deterministic
 - AVG, MIN, MAX, STDEV, STDEVP, VAR, VARP - not allowed
 - the index must be clustered and unique
 - the SELECT statement in the definition of the view cannot contain: subqueries, outer joins, EXCEPT, INTERSECT, UNION, TOP, DISTINCT, ORDER BY, etc

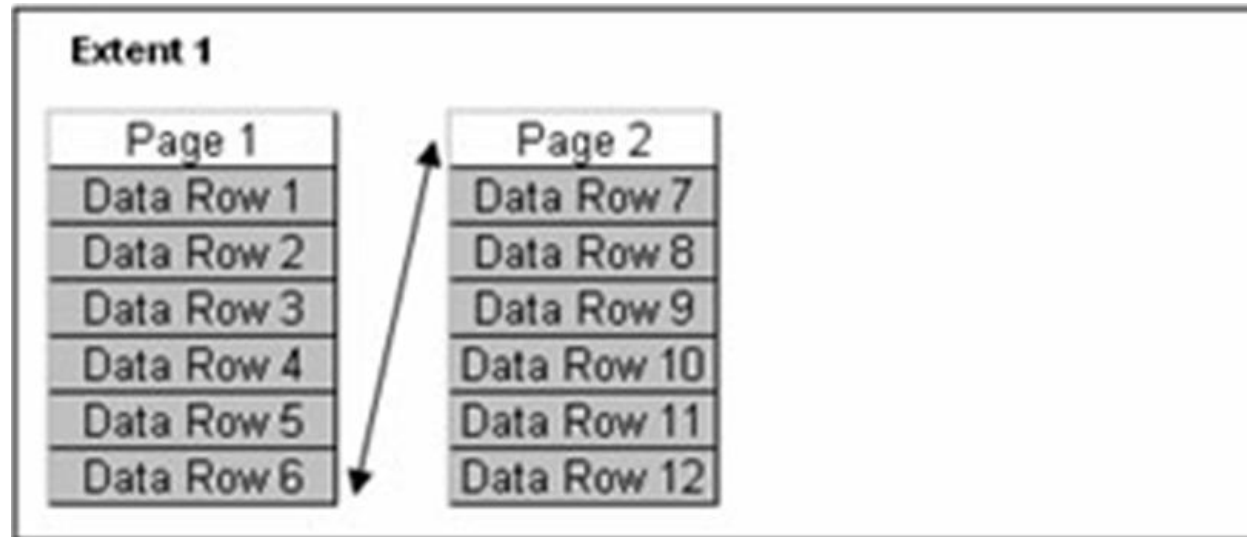
- indexing – rules
 - every table should have a clustered index
 - a clustered index should be small, selective, increasing, static
 - non-clustered indexes should be created on
 - foreign keys
 - columns often used in the WHERE clause
 - covering non-clustered indexes should be created for the most frequently executed queries
 - one shouldn't create single-column indexes on all the columns in a table, as they incur high overhead
 - in multi-column indexes, the most selective columns (nearest to unique) should be placed on the first positions

- fragmentation
 - internal fragmentation
 - records are not stored in a contiguous area on the page, i.e., there is unused space between records in a page
 - the fullness of each page can vary over time; unused space => inefficient use of the cache & more page transfers between disk and main memory => impact on query performance
 - extent fragmentation
 - the extents* of a table are not contiguous

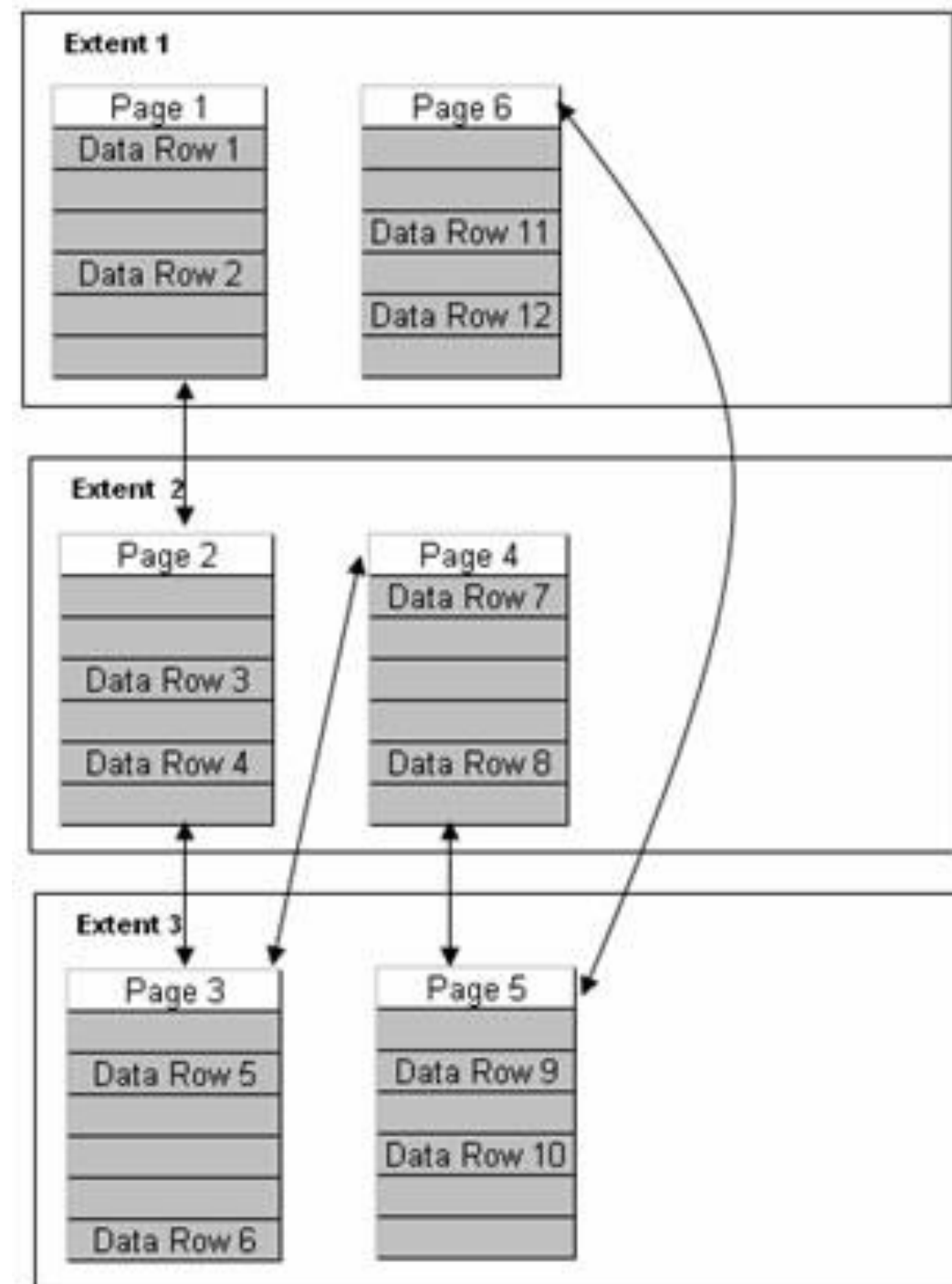
*extent – group of 8 contiguous pages

- fragmentation
 - logical fragmentation
 - every index page is linked with the previous and the next page based on the logical order of the key values
 - when pages are full and there's no room for new data, values are redistributed (*Page Split*) => out-of-order pages
 - out-of-order page
 - a page for which the next physical page in the index is not the next logical page

- fragmentation
 - read operation - page requests: 2
 - extent switches: 0
 - disk space (used by table): 16 KB
 - avg_fragmentation_in_percent: 0
 - avg_page_space_used_in_percent: 100



- fragmentation
 - page requests: 6
 - extent switches: 5
 - disk space: 48 KB
 - avg_fragmentation_in_percent > 80
 - avg_page_space_used_in_percent: 33



- fragmentation
 - sys.dm_db_index_physical_stats
 - avg_fragmentation_in_percent
 - percentage value
 - heaps – extent fragmentation
 - indexes – logical fragmentation
 - avg_page_space_used_in_percent
 - average percentage of available space in all pages

- reduce fragmentation
 - in a heap
 - create a clustered index on the table, then drop it
 - creating the clustered index => records are redistributed

- reduce fragmentation
 - in an index
 - if avg_fragmentation_in_percent > 5% and <= 30%
 - reorder the leaf pages of the index based on the key: ALTER INDEX REORGANIZE
 - if avg_fragmentation_in_percent > 30%
 - use ALTER INDEX REBUILD (replacement for DBCC DBREINDEX)
 - similar effect: drop and recreate the index
 - drop and recreate a clustered index
 - when a clustered index is recreated, the data is redistributed => full data pages
 - the level of fullness can be configured with the FILLFACTOR option in CREATE INDEX

T-SQL - Control-of-Flow Language

- BEGIN...END
- BREAK
- CONTINUE
- GOTO label
- IF...ELSE
- RETURN
- THROW
- TRY...CATCH
- WAITFOR
- WHILE

- RETURN
 - RETURN [integer_expression]
 - exits from a procedure / batch / statement block
 - returning status codes
 - unless specified otherwise, system stored procedures return:
 - 0 - success
 - a nonzero value – failure

->

- **RETURN**

```
CREATE PROCEDURE uspCheckCountry @country varchar(50)
AS
    IF @country = 'Romania'
        RETURN 1
    ELSE
        RETURN 2;
GO

-----

DECLARE @ret_status_code int;
EXEC @ret_status_code = uspCheckCountry 'Romania';
SELECT @ret_status_code
GO
```


- WHILE
 - WHILE boolean_expression
 { sql_statement | statement_block | BREAK | CONTINUE }
 - repeated execution of a SQL statement or statement block while the specified condition is true
- BREAK
 - exits current WHILE loop (if the latter is nested inside another WHILE loop, BREAK exits only the current loop)
 - can appear in an IF statement
- CONTINUE
 - restarts a WHILE loop
 - any statements after CONTINUE are ignored

- GOTO
 - execution continues at the label

Label:

GOTO Label

- **WAITFOR**
 - `WAITFOR { DELAY 'time_to_pass' |
TIME 'time_to_execute' }`
 - blocks the execution of a batch / stored procedure / transaction
 - execution continues at 07:15
 - `WAITFOR TIME '07:15';`
 - execution continues after 3 hours
 - `WAITFOR DELAY '03:00';`
 - if the server is busy, the counter may not start immediately => the delay may be longer than the specified one

- THROW

- THROW [

- { error_number | @local_variable},

- { message | @local_variable},

- { state | @local_variable }] [;]

- error_number: >= 50000

- message: nvarchar(2048)

- state: 0 - 255

- raises an exception, transfers execution to the CATCH block of a TRY...CATCH construct

- THROW 51000, '50 rows have been modified.', 1;

- THROW

- THROW [

- { error_number | @local_variable},

- { message | @local_variable},

- { state | @local_variable }] [;]

- severity - always 16

- RAISERROR

- can specify one of 26 severity levels

- severity level between 20 and 25 => fatal error, the connection is terminated

- TRY ... CATCH

- BEGIN TRY

- { sql_statement | statement_block }

- END TRY

- BEGIN CATCH

- [{ sql_statement | statement_block }]

- END CATCH [;]

- implements error handling in Transact-SQL

- catches execution errors with severity >10 and that do not close the database connection

- TRY ... CATCH
 - ERROR_NUMBER() - returns the error number
 - ERROR_SEVERITY() - returns the error severity
 - ERROR_STATE() - returns the error state number
 - ERROR_PROCEDURE() - returns the name of the stored procedure / trigger where the error occurred
 - ERROR_LINE() - returns the number of the line where the error occurred
 - ERROR_MESSAGE() - returns the error message