Indecși

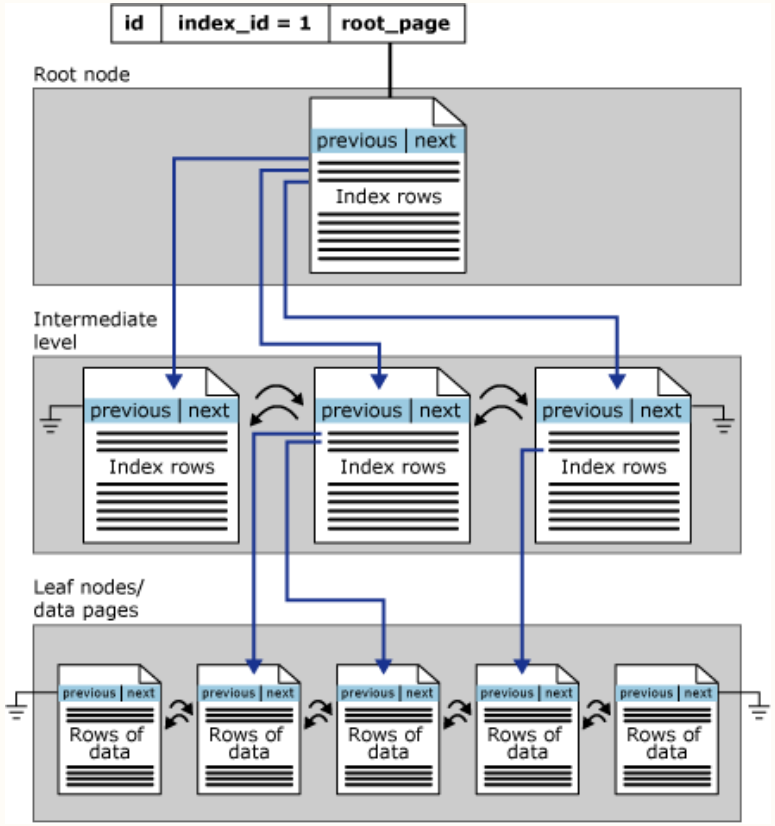
Seminar 5

Indecși

- Un index este o structură on-disk asociată unui tabel sau unui view care crește viteza de returnare a înregistrărilor
- Alegerea corectă a indecșilor îmbunătățește performanța
- Alegerea incorectă a indecșilor generează probleme de performanță
- Indecșii sunt definiți pentru a localiza mai rapid înregistrările care urmează să fie returnate
- Dacă nu este definit un index, SQL Server verifică fiecare înregistrare din tabel pentru a determina dacă ea conține sau nu informația necesară interogării (table scan)

Indecși

- Indecșii sunt organizați ca B-trees



Indecși - Caracteristici

- Clustered versus nonclustered
- Unique versus non unique
- Single column versus multi-column
- Ordine crescătoare sau descrescătoare pe coloanele din index
- Full-table versus filtered pentru indecșii nonclustered

Indecși – clustered vs nonclustered

- Index **clustered** – definește ordinea fizică în care sunt stocate datele în tabel (dacă indexul **clustered** conține mai multe coloane, datele vor fi stocate în ordine secvențială în funcție de coloane: prima coloană, a doua coloană și așa mai departe)
- Se poate defini doar un singur index **clustered** pe fiecare tabel, deoarece nu putem stoca fizic datele decât într-o singură ordine
- Paginile de date ale unui index **clustered** vor conține întotdeauna toate coloanele din tabel
- Sintaxa:

```
CREATE CLUSTERED INDEX index_name ON table_name
(column_name(s) [ASC|DESC]);
```

Indecși – clustered vs nonclustered

- Index **nonclustered** – spre deosebire de un index **clustered**, stochează pointeri la date din **heap/indexul clustered** ca parte din index key
- Putem avea mai mulți indecși **nonclustered** pe același tabel
- Sintaxa:

```
CREATE INDEX index_name ON table_name (column_name(s)
[ASC|DESC]);
```

- SQL Server suportă până la 999 indecși **nonclustered** pe tabel
- Un index key poate fi format din maxim 16 coloane și 900 bytes

Indecși – clustered vs nonclustered

- Când este creată o **cheie primară** pe un tabel, dacă nu este definit deja un index clustered și un index nonclustered nu este specificat, se creează un **index clustered unique**
- Dacă atunci când este creată o **cheie primară** pe un tabel este deja definit un index clustered, se va crea un **index nonclustered unique**
- Dacă toate coloanele returnate de către o interogare se află în index, indexul se numește **covering index**, iar interogarea se numește **covered query**

Indecși – clustered

- Poate fi folosit pentru interogările care se execută în mod frecvent
- Poate fi folosit în **range queries**
- Nu este bine să definim un **index clustered** pe coloane care sunt actualizate des, deoarece SQL Server va trebui să modifice constant ordinea fizică a datelor
- Exemplu de definire a unui index clustered unique:

```
CREATE UNIQUE CLUSTERED INDEX
IX_Produse_cod_produs_asc ON Produse (cod_produs
ASC);
```

Indecși – nonclustered

- Atunci când se definește o **constrângere UNIQUE** pe un tabel, se va crea un **index nonclustered unique** pe coloana sau coloanele pe care este definită **constrângerea UNIQUE**
- Exemplu de definire a unui index nonclustered non unique:

```
CREATE NONCLUSTERED INDEX IX_Produse_nume_asc ON
Produse (nume ASC);
```

sau

```
CREATE INDEX IX_Produse_nume_asc ON Produse (nume);
```

Indecși – coloane key vs coloane nonkey

- **Coloane key** – coloanele specificate la crearea unui index
- **Coloane nonkey** – coloanele specificate în **clauza INCLUDE** a unui **index nonclustered**
- Sintaxa:

```
CREATE INDEX index_name ON table_name
(key_column_name(s) [ASC|DESC])
INCLUDE (nonkey_column_name(s));
```

- Exemplu:

```
CREATE INDEX IX_Persoane_nume_asc_prenume_asc ON
Persoane (nume ASC, prenume ASC)
INCLUDE (pseudonim, localitate);
```

Indecși – coloane key vs coloane nonkey

- Beneficii ale utilizării **coloanelor nonkey**:
  - Coloanele pot fi accesate cu un **index scan**
  - Tipurile de date care nu sunt permise în coloanele care fac parte din index (index key columns) sunt permise în coloanele nonkey (exceptând tipurile de date text, ntext și image)
  - Pot fi incluse coloane calculate, dar valorile trebuie să fie deterministe
  - Coloanele care apar în **clauza INCLUDE** nu se iau în calcul în cazul limitei de 900 bytes a unui index impusă de SQL Server

Indecși unique versus indecși non unique

- Un **index unique** (unic) definit pe una sau mai multe coloane asigură unicitatea valorilor la nivelul coloanei sau combinației de coloane pe care este definit
- De exemplu, dacă vom crea un index nonclustered unic în tabelul *Categorii* pe coloana *nume*, nu vom putea avea două înregistrări cu aceeași valoare pentru coloana *nume* în tabel
- Dacă vom crea un index nonclustered unic în tabelul *Persoane* pe coloanele *nume* și *prenume*, nu vom putea avea două înregistrări cu aceleași valori pentru coloanele *nume* și *prenume* în tabel
- Dacă vom crea un index unic după ce am introdus date în tabel și avem valori duplicate în coloana sau coloanele pe care am definit indexul unic, operațiunea de creare a indexului va eșua

Indecși unique versus indecși non unique

- Pentru a putea crea un index unic pe un tabel, va trebui să eliminăm înainte toate valorile duplicate din coloana sau coloanele pe care definim indexul unic
- Un index unic garantează că fiecare valoare (inclusiv NULL) pentru coloana pe care a fost definit apare o singură dată în tabel
- Exemplu de index nonclustered unique definit pe o singură coloană:

```
CREATE UNIQUE INDEX IX_Categorii_nume_desc_uq ON
Categorii (nume DESC);
```

- Exemplu de index nonclustered unique definit pe mai multe coloane:

```
CREATE UNIQUE INDEX
IX_Persoane_nume_asc_prenume_asc_uq ON Persoane
(nume ASC, prenume ASC);
```

Indecși unique versus indecși non unique

- În cazul indecșilor unici se poate seta opțiunea **IGNORE_DUP_KEY**
- Dacă se setează **IGNORE_DUP_KEY=ON**, în cazul unui batch INSERT, se vor insera toate înregistrările care nu conțin valori duplicate, iar înregistrările care conțin valori duplicate vor fi ignorate și nu vor fi inserate în tabel
- Exemplu de definire a unui index nonclustered unique pe o singură coloană cu opțiunea **IGNORE_DUP_KEY=ON**:

```
CREATE UNIQUE INDEX IX_Produse_nume_asc_uq ON
Produse (nume ASC) WITH (IGNORE_DUP_KEY=ON);
```

- Indecșii non unique (care nu sunt unici) permit inserarea de valori duplicate în tabel

Indecși single column versus indecși multi-column

- Un **index single column** este un index definit pe o singură coloană (care conține o singură coloană key în index key)
- Un **index multi-column** este un index definit pe mai multe coloane (care conține mai multe coloane key în index key)
- Dacă dorim să folosim indexul și pentru sortarea înregistrărilor care sunt returnate, va trebui să ținem cont de ordinea crescătoare sau descrescătoare a coloanelor care fac parte din index key
- În cazul unui index **single column**, ordinea specificată pentru coloana key nu este atât de importantă deoarece se poate folosi indexul pentru a sorta după coloana respectivă atât în ordine crescătoare cât și în ordine descrescătoare

Indecși single column versus indecși multi-column

- Exemplu de definire a unui index nonclustered non unique single column:

```
CREATE INDEX IX_Produse_nume_desc ON Produse
(nume  DESC);
```

- Indexul definit mai sus va putea fi folosit pentru ambele interogări de mai jos:

```
SELECT nume FROM Produse ORDER BY nume DESC;

SELECT nume FROM Produse ORDER BY nume ASC;
```

- În cazul indecșilor **multi-column**, ordinea coloanelor key este importantă dacă dorim să se utilizeze indexul pentru sortarea înregistrărilor după coloanele care fac parte din index key

Indecși single column versus indecși multi-column

- Exemplu de definire a unui index nonclustered non unique multi-column:

```
CREATE INDEX IX_Produse_nume_asc_cod_cat_asc ON
Produse (nume ASC, cod_cat ASC);
```

- Indexul definit mai sus va putea fi folosit pentru sortare în cazul interogărilor de mai jos:

```
SELECT nume, cod_cat FROM Produse ORDER BY nume ASC,
cod_cat ASC;

SELECT nume, cod_cat FROM Produse ORDER BY nume DESC,
cod_cat DESC;

SELECT nume, cod_cat FROM Produse ORDER BY nume ASC;

SELECT nume, cod_cat FROM Produse ORDER BY nume DESC;
```

Indecși single column versus indecși multi-column

- Indexul IX_Produse_nume_asc_cod_cat_asc definit anterior nu va putea fi folosit pentru sortare în cazul interogărilor de mai jos:

```
SELECT nume, cod_cat FROM Produse ORDER BY nume ASC, cod_cat DESC;

SELECT nume, cod_cat FROM Produse ORDER BY nume DESC, cod_cat ASC;

SELECT nume, cod_cat FROM Produse ORDER BY cod_cat ASC;

SELECT nume, cod_cat FROM Produse ORDER BY cod_cat DESC;
```

Full-table versus filtered pentru indecșii nonclustered

- Indecșii **nonclustered full-table** conțin toate valorile coloanei sau coloanelor pe care au fost definiți
- Indecșii **nonclustered filtered** conțin doar acele valori pentru care evaluarea condiției specificate la crearea indexului returnează true
- Exemplu de definire a unui index nonclustered non unique single column filtered:

```
CREATE INDEX IX_Produse_nume_asc_filtered ON Produse
(nume ASC) WHERE nume>'C';
```

Full-table versus filtered pentru indecșii nonclustered

- Indexul IX_Produse_nume_asc_filtered va putea fi folosit pentru următoarele interogări:

```
SELECT nume FROM Produse WHERE nume>'C';
```

```
SELECT nume FROM Produse WHERE nume>'E';
```

- Indexul IX_Produse_nume_asc_filtered nu va putea fi folosit pentru următoarele interogări:

```
SELECT nume FROM Produse WHERE nume<'C';
```

```
SELECT nume FROM Produse;
```

Indecși pentru DELETE

- Indecșii pot fi utili și în momentul în care trebuie să ștergem date din baza de date, nu doar atunci când returnăm date
- Atunci când se efectuează o operațiune de ștergere, SQL Server verifică toate tabelele dependente de tabelul din care se șterg date pentru a determina dacă există înregistrări dependente de cele pe care dorim să le ștergem
- În acest caz, un index definit pe un **foreign key** va putea fi folosit pentru a găsi înregistrările dependente mult mai rapid
- În caz contrar, SQL Server va verifica toate înregistrările din tabelul respectiv, prin urmare operațiunea de ștergere va fi lentă

Indecși – Modificarea unui index

- Dacă dorim să ștergem sau să adăugăm coloane într-un index va trebui să ștergem și să creăm din nou indexul
- Dacă dorim să dezactivăm un index sau să setăm anumite opțiuni, putem folosi instrucțiunea **ALTER INDEX**
- Exemplu de dezactivare a unui index:

```
ALTER INDEX IX_Produse_nume_desc ON Produse DISABLE;
```

- Exemplu de reactivare a unui index dezactivat:

```
ALTER INDEX IX_Produse_nume_desc ON Produse REBUILD;
```

Indecși - ștergerea unui index

- În anumite situații, un index devine inutil și trebuie eliminat
- Sintaxa pentru ștergerea unui index:

```
DROP INDEX index_name ON table_name;
```

- Exemplu de ștergere a unui index:

```
DROP INDEX IX_Studenți_nume_desc ON Studenți;
```

Indecși - Recomandări

- Indecșii sunt utili pentru mărirea performanței operațiunilor de citire, dar scad performanța operațiunilor de scriere
- Tipuri de coloane recomandate ca index key columns:
  - coloane care au definită o constrângere foreign key
  - coloane care apar în clauza WHERE a interogărilor
  - coloane care apar în clauza ORDER BY a interogărilor
  - coloane pe baza cărora se fac JOIN-uri
  - coloane care apar în clauza GROUP BY a interogărilor
  - coloane cu grad mare de varietate a valorilor

Recomandări de proiectare a indecșilor

- Înțelegerea caracteristicilor bazei de date (OLTP versus OLAP)
- Înțelegerea caracteristicilor celor mai frecvente interogări
- Înțelegerea caracteristicilor coloanelor care sunt folosite în interogări
- Determinarea locației de stocare optimă pentru index