Databases

dr. Jörg Verstraete

Polish Academy of Sciences
Systems Research Institute

jorg.verstraete@ibspan.waw.pl
http://www.mini.pw.edu.pl/~verstraetej/
Today

• Lecture
  – Indexes
    • what?
    • why?
    • how?
• Labs
  – SQL lab
Tables

• Theoretically
  – no order on records
  – results can be requested in any order

• Practically
  – a sorted order on disk may increase performance
  – only one storage order is possible
    • indexes!
Index

• An additional data structure used to speed up information retrieval from a table
  – commonly tree like structures
    • B-tree, B+ tree
  – DBMS specific implementations
    • Oracle bitmap index
  – Refers to the data from a single table
    • single index can be on multiple attributes
    • order can be specified
Index categories

- **SQL Server**
  - **Clustered**
    - physical storage order (usually primary key)
    - one per table
    - typically B-tree, leaves contain data pages
  - **Non-Clustered**
    - Created as heap
    - Created on clustered index
B-tree? B+tree? Heap?

- **B-tree**
  - self-balancing (limited depth)
  - ordered tree structure
    - data both in nodes and leaves
- **B+ tree**
  - data only in leaves
- **Heap**
  - partially ordered tree structure
    - parents greater / smaller than children (max / min heap)
Example

- **Students**
  - primary key
    - Students: { Student id }

- **Clustered index**
  - physical order
    - data can be stored in the index structure
    - extra overhead when data is modified / inserted

<table>
<thead>
<tr>
<th>Student id</th>
<th>Last Name</th>
<th>First Name</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marsh</td>
<td>Stan</td>
<td>F1</td>
</tr>
<tr>
<td>2</td>
<td>Broflovski</td>
<td>Kyle</td>
<td>F1</td>
</tr>
<tr>
<td>3</td>
<td>Cartman</td>
<td>Erik</td>
<td>F2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Heap

- If no clustered index
  - records stored in the form of a heap
  - no imposed ordering
  - faster insert / update
  - no need to preserve order
Non-clustered

- Multiple possible
  - index 1: studentid
  - index 2: last name, first name
- Index tree only contains values for the index
  - the values of the attribute
  - tree pointers point to the records of the table
Index

• Benefits
  – Can improve performance
    • to match a WHERE condition
    • to order records (ORDER BY)

• Drawbacks
  – DBMS has to keep the index updated and consistent with the data
  – Can increase execution time of CRUD
    • Multiple indexes may need to be rebuilt after CRUD
  – Takes up storage space

• DBMS decides whether or not to use an index
  – e.g. if all records are needed, no index has to be used
  – add a referencing step to the reads (read index, find data, read data)
Unique index

• Can be created to preserve uniqueness
  – e.g. primary key

• Reason
  – During updates, DBMS can quickly determine if no duplicates are inserted
    • otherwise, whole table needs to be scanned
Unique index

• Can be created to preserve uniqueness
  – e.g. primary key

• Reason
  – During updates, DBMS can quickly determine if no duplicates are inserted
    • otherwise, whole table needs to be scanned
Guidelines

- Index CAN speed up querying
  - due to decreased number of page reads
  - avoids scanning entire table
- Clustered indexes
  - can be defined to group attribute values that are frequently together
    - e.g. multiple lines of a same order
- Indexes on foreign keys can speed up referential integrity checks
- Unique indexes help preserve uniqueness
Guidelines

• When?
  – Table / attributes where
    • large number of records
    • frequently used columns
      – speeds up globally
    • short index key
    • good selectivity

• When not?
  – Increases CRUD processing time, avoid unnecessary indexes