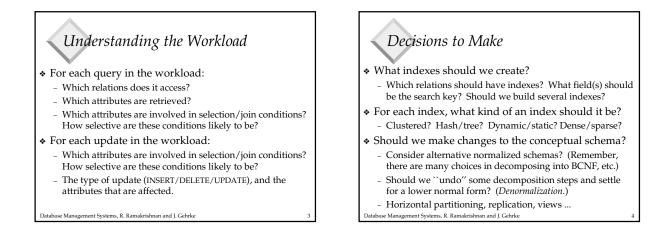




- After ER design, schema refinement, and the definition of views, we have the *conceptual* and *external* schemas for our database.
- The next step is to choose indexes, make clustering decisions, and to refine the conceptual and external schemas (if necessary) to meet performance goals.
- We must begin by understanding the <u>workload</u>:
 The most important queries and how often they arise.
 - The most important queries and how often they arise.
 The most important updates and how often they arise.
 - The desired performance for these queries and updates.
- Database Management Systems, R. Ramakrishnan and J. Gehrke



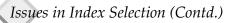
Choice of Indexes

- One approach: consider the most important queries in turn. Consider the best plan using the current indexes, and see if a better plan is possible with an additional index. If so, create it.
- Before creating an index, must also consider the impact on updates in the workload!
 - Trade-off: indexes can make queries go faster, updates slower. Require disk space, too.

Database Management Systems, R. Ramakrishnan and J. Gehrke

Issues to Consider in Index Selection

- Attributes mentioned in a WHERE clause are candidates for index search keys.
- Exact match condition suggests hash index.
- Range query suggests tree index.
 - Clustering is especially useful for range queries, although it can help on equality queries as well in the presence of duplicates.
- Try to choose indexes that benefit as many queries as possible. Since only one index can be clustered per relation, choose it based on important queries that would benefit the most from clustering. Database Management Systems, R. Ramakrishnan and J. Gehrke



- Multi-attribute search keys should be considered
- when a WHERE clause contains several conditions. - If range selections are involved, order of attributes should
 - be carefully chosen to match the range ordering. Such indexes can sometimes enable index-only strategies
 - for important queries.
 - For index-only strategies, clustering is not important!
- When considering a join condition:

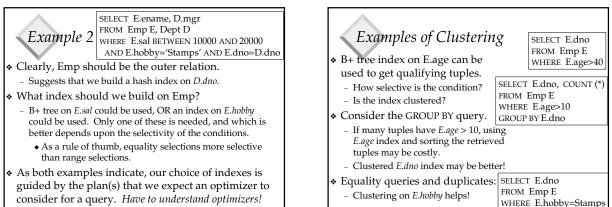
atabase Management Systems, R. Ramakrishnan and J. Gehrke

- Hash index on inner is very good for Index Nested Loops. Should be clustered if join column is not key for inner, and inner tuples need to be retrieved.
- Clustered B+ tree on join column(s) good for Sort-Merge. atabase Management Systems, R. Rama hnan and J. Gehrk

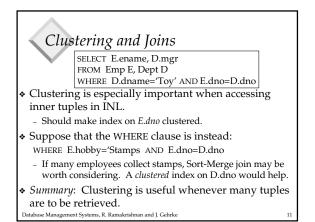
SELECT E.ename, D.mgr Example 1 FROM Emp E, Dept D WHERE D.dname='Toy' AND E.dno=D.dno

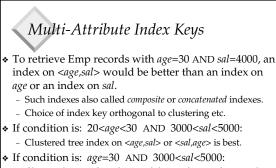
- Hash index on *D.dname* supports 'Toy' selection. - Given this, index on D.dno is not needed.
- Hash index on E.dno allows us to get matching (inner) Emp tuples for each selected (outer) Dept tuple.
- ♦ What if WHERE included: `` ... AND E.age=25"?
 - Could retrieve Emp tuples using index on E.age, then join with Dept tuples satisfying dname selection. Comparable to strategy that used E.dno index.
 - So, if *E.age* index is already created, this query provides much less motivation for adding an *E.dno* index.

se Management Systems, R. Ramakrishnan and J. Gehrke



Database Management Systems, R. Ramakrishnan and J. Gehrl

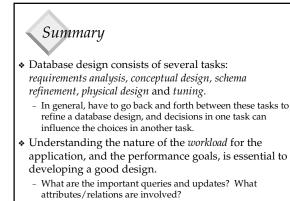




- Clustered <age,sal> index much better than <sal,age> index!
- * Composite indexes are larger, updated more often.

Database Management Systems, R. Ramakrishnan and J. Gehrke

Index-Only	y Plans <e.dn< th=""><th>SELECT D.mgr FROM Dept D, Emp E WHERE D.dno=E.dno</th><th></th></e.dn<>	SELECT D.mgr FROM Dept D, Emp E WHERE D.dno=E.dno	
 A number of queries can be answered 	<e.dno,e.eid: Tree index</e.dno,e.eid: 	EPOM Dopt D Emp E	
without retrieving any	<e.dno></e.dno>	SELECT E.dno, COUNT(*) FROM Emp E GROUP BY E.dno	
tuples from one or more of the <i><e.dno,e.sal></e.dno,e.sal></i> relations <i>Tree index!</i>		SELECT E.dno, MIN(E.sal) FROM Emp E GROUP BY E.dno	
involved if a < <i>E</i> . <i>age,E</i> . <i>sal></i> suitable index or is available. < <i>E</i> . <i>sal</i> , <i>E</i> . <i>age></i> <i>Tree!</i> Database Management Systems, R. Ramakrishnan and L Genteen Suite State			13



14

Database Management Systems, R. Ramakrishnan and J. Gehrke

Summary (Contd.)

 Indexes must be chosen to speed up important queries (and perhaps some updates!).

- Index maintenance overhead on updates to key fields.
- Choose indexes that can help many queries, if possible.
- Build indexes to support index-only strategies.
- Clustering is an important decision; only one index on a given relation can be clustered!
- $\ -$ Order of fields in composite index key can be important.
- Static indexes may have to be periodically re-built.
- Statistics have to be periodically updated.

Database Management Systems, R. Ramakrishnan and J. Gehrke