

- ✤ For each query in the workload:
 - Which relations does it access?
 - Which attributes are retrieved?
 - Which attributes are involved in selection/join conditions? How selective are these conditions likely to be?
- For each update in the workload:
 - Which attributes are involved in selection/join conditions? How selective are these conditions likely to be?
 - The type of update (INSERT/DELETE/UPDATE), and the attributes that are affected
- How important is a query/update? Frequent, long-running queries are usually the most important to optimize
 - se Ma ement Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Indices and Clustering: Decisions to Make

- What indexes should we create?
 - Which relations should have indexes?
 - What field(s) should be the search key?
 - Should we build several indexes?
- ✤ For each index, what kind of an index should it be? – Clustered?
 - Hash/tree?
- Need to apply your knowledge of indexing Also need to make sure that optimizer uses the indices! (including index-only plans)
 - Need to apply your knowledge of optimizers!

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 the additional index
 - "Greedy"
- ✤ 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 (secondary issue)
- Have been attempts to automate this
- Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Tuning the Conceptual Schema

- Should be guided by the workload, in addition to redundancy issues:
 - We may settle for a 3NF schema rather than BCNF.
 - We may further decompose a BCNF schema!
 - We might denormalize (i.e., undo a decomposition
 - step), or we might add fields to a relation.
 - We might consider horizontal decompositions.

Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Example Schemas

Contracts (<u>Cid</u>, Sid, Jid, Did, Pid, Qty, Val) Depts (<u>Did</u>, Budget, Report) Suppliers (<u>Sid</u>, Address) Parts (<u>Pid</u>, Cost) Projects (<u>Jid</u>, Mgr)

- ◆ We will concentrate on Contracts, denoted as CSJDPQV. The following ICs are given to hold: JP→ C, SD → P, C is the primary key.
 - What are the keys for CSJDPQV?
 - What are the Keys for CoJDI QV:
 What normal form is this relation schema in?
 - base Management Systems. 2nd Edition. R. Ramakrishnan and I. Gehrke

Settling for 3NF vs BCNF

 CSJDPQV can be decomposed into SDP and CSJDQV, and both relations are in BCNF.

- Lossless decomposition, but not dependency-preserving.Adding CJP makes it dependency-preserving as well.
- Suppose that this query is very important:
- Find the number of copies Q of part P ordered in contract C.
- Requires a join on the decomposed schema, but can be answered by a scan of the original relation CSJDPQV.
- Could lead us to settle for the 3NF schema CSJDPQV.

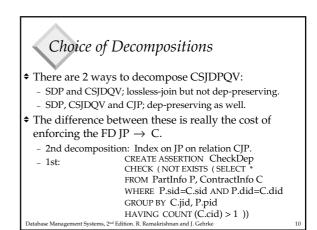
Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Denormalization

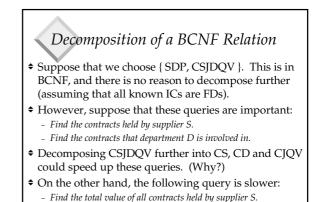
Suppose that the following query is important:*Is the value of a contract less than the budget of the department?*

- To speed up this query, we might add a field *budget* B to Contracts.
 - This introduces the FD $D \rightarrow B$ in Contracts
 - Thus, Contracts is no longer in 3NF.
- We might choose to modify Contracts thus if the query is sufficiently important
 - Note: we cannot improve performance otherwise (i.e., by adding indexes or by choosing an alternative 3NF schema.)

Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke



Choice of Decompositions (Contd.)
The following ICs were given to hold: JP→ C, SD → P, C is the primary key.
Suppose that, in addition, a given supplier always charges the same price for a given part: SPQ → V.
If we decide that we want to decompose CSJDPQV into BCNF, we now have a third choice:
Begin by decomposing it into SPQV and CSJDPQ.
Then, decompose CSJDPQ (not in 3NF) into SDP, CSJDQ.
This gives us the lossless-join decomp: SPQV, SDP, CSJDQ.
To preserve JP → C, we can add CJP, as before.
Choice: {SPQV, SDP, CSJDQ } or {SDP, CSJDQV } ?



tabase Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Horizontal Decompositions

- Our definition of decomposition: Relation is replaced by a collection of relations that are *projections*. Most important case.
- Sometimes, might want to replace relation by a collection of relations that are *selections*.
 - Each new relation has same schema as the original, but a subset of the rows.
 - Collectively, new relations contain all rows of the original. Typically, the new relations are disjoint.

Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

Horizontal Decompositions (Contd.)

- Suppose that contracts with value > 10000 are subject to different rules. This means that queries on Contracts will often contain the condition val>10000.
- One way to deal with this is to build a clustered B+ tree index on the *val* field of Contracts.
- A second approach is to replace contracts by two new relations: LargeContracts and SmallContracts, with the same attributes (CSJDPQV).
 - Performs like index on such queries, but no index overhead.
 - Can build clustered indexes on other attributes, in addition!

abase Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

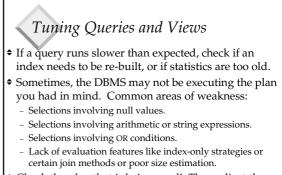
Logical Data Independence CREATE VIEW Contracts(cid, sid, jid, did, pid, qty, val)

AS SELECT * FROM LargeContracts UNION

SELECT *

FROM SmallContracts

- The replacement of Contracts by LargeContracts and SmallContracts can be masked by the view.
- However, queries with the condition val>10000 must be asked wrt LargeContracts for efficient execution: so users concerned with performance have to be aware of the change.



Check the plan that is being used! Then adjust the choice of indexes or rewrite the query/view. Database Management Systems, 2nd Edition. R. Ramakrishnan and J. Gehrke

