Physical Design

Background

• Physical design applied after logical design, mapping conceptual level objects (e.g., schemas) onto physical level objects (files, B-trees, hash tables with clustering choices)
• Inputs: schemas, query load
• Because of data independence, iteration is easy.

Refining Logical Design

• Some attributes need to be stored “together” for the sake of efficient access (denormalization).
• Example:
  ```sql
  SELECT SNAME
  FROM STUDENT S, ENR E
  WHERE S.SID=E.SID
  AND GRADE = $1
  ```

Refining Logical Design, cont’d

• Sometimes relations need to be fragmented either vertically or horizontally.
• Example:
  ```sql
  SELECT *
  FROM ACCOUNTS A
  WHERE A.CITY = $1
  ```
• Normalized views can be defined in terms of fragments or non-normalized relations.
Dimensions of Problem

- Access structures cost, especially during updates.
- Hash indices support equality searches very efficiently, but do not support range searches.
- B-trees support both range and equality searches.

Supporting Joins

```
SELECT SNAME
FROM STUDENT S, ENR E
WHERE S.SID=E.SID
AND GRADE = 'A'
```

- clustered hash or B-tree index on STUDENT.SID
- clustered has or B-tree index on ENR.SID
- STUDENT and ENR clustered together on SID

Multiple-Attribute Indices

- Multiple-attribute indices are often useful for index-only queries.
- Example
  ```
  SELECT SNAME, YOG
  FROM STUDENT S, ENR E
  WHERE S.SID = E.SID
  ```
  can be implemented as a join between STUDENT and an index on ENR with key SID

Multiple-Attribute Indices, cont’d

- Multiple-attribute indices are less useful for non-index-only query.
- An index on student with key <SNAME, YOG, MAJOR> supports searches on prefixes, with range possible on last attribute of prefix.
- Three indices with keys <SNAME>, <YOG>, <MAJOR> support many more kinds of searches.
**Clustering**

- Clustering is important for range queries (simplification; selectivity is what matters).
- Clustering is not useful for index-only strategies.
- Multiple relations can be (co-)clustered together in some DBMSs, including Oracle.

**Tuning**

- Plans can be displayed to determine if access structures are getting used.
- Sometimes it’s useful to rewrite a query to an equivalent one.
- Costs of lock acquisition should be evaluated. Smaller transactions hold onto locks a shorter time. A single bigger transaction may not release and acquire locks as much as a several smaller ones.
- Roots of B-trees can be hotspots for concurrent access.

**Summary**

- Physical design inputs: schemas, query load
- Physical design considerations sometimes affect logical design (de-normalization, fragmentation).
- Lots of heuristics exist, but DBMS optimizer often frustrates attempts at good design. Understand the optimizer!