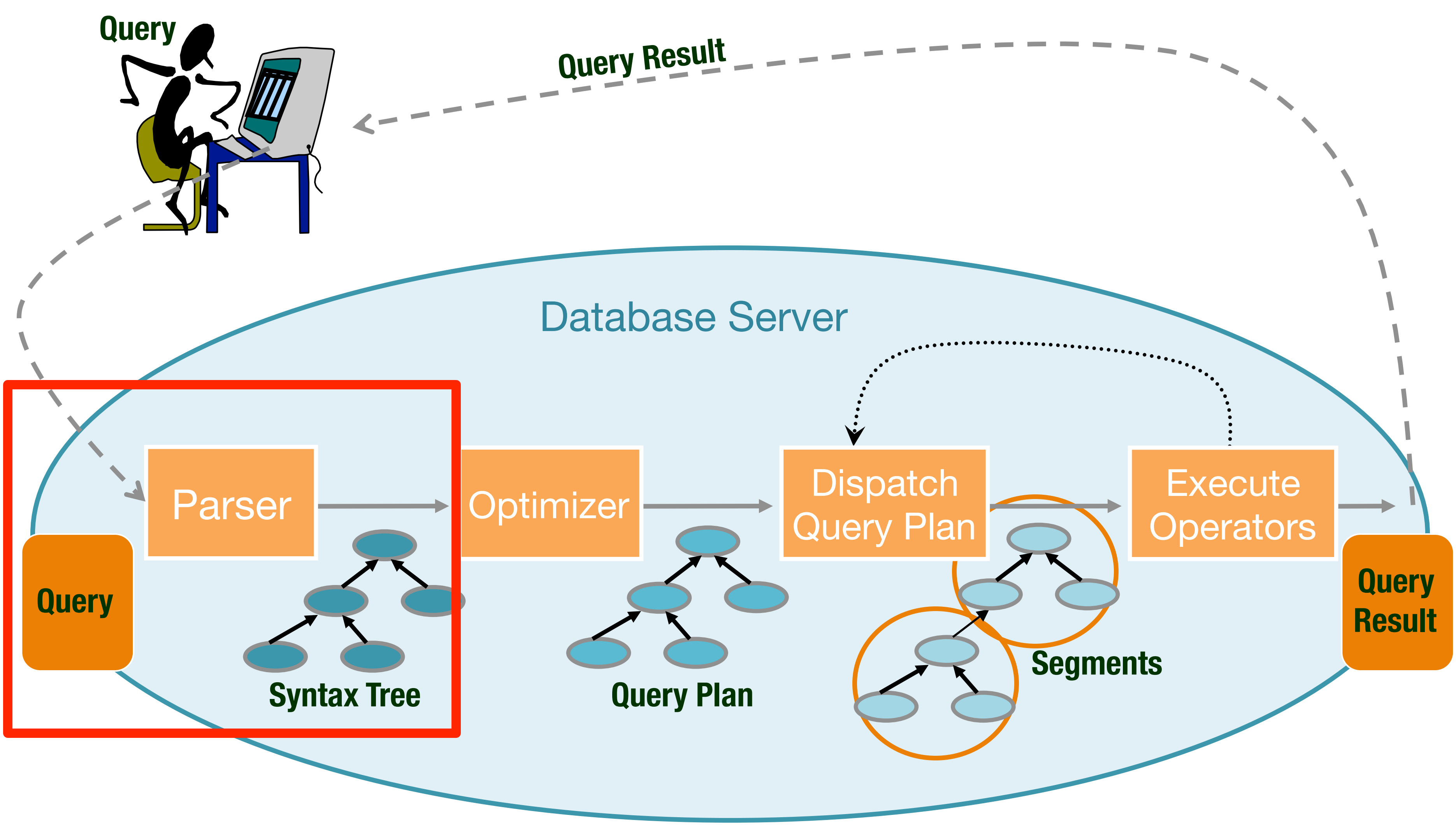
Evaluation of Relational Operations

Chapter 12

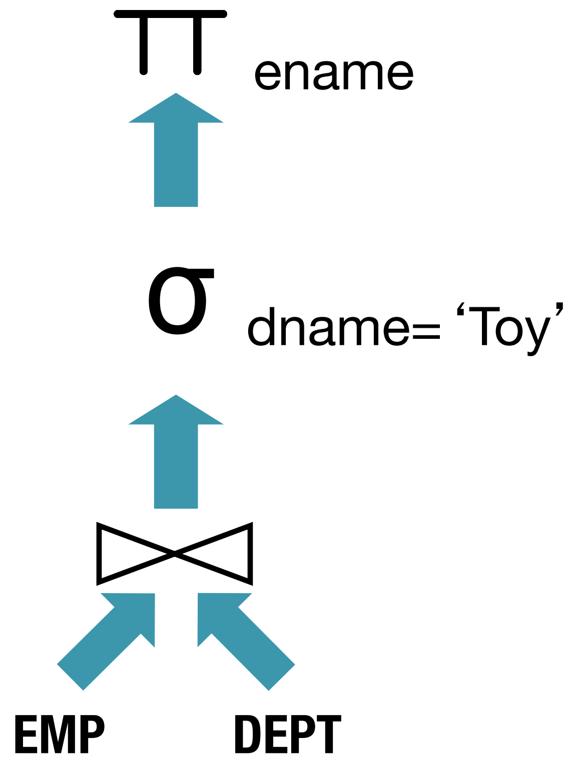
Query Execution Life-Cycle



Query Parsing

- Consult the meta information stored in system catalogs
- Create an initial syntax tree based on relational algebra tree

```
SELECT E.ename
FROM Emp E, Dept D
WHERE D.dname = 'Toy' AND
      D.did = E.did
```



System Catalogs

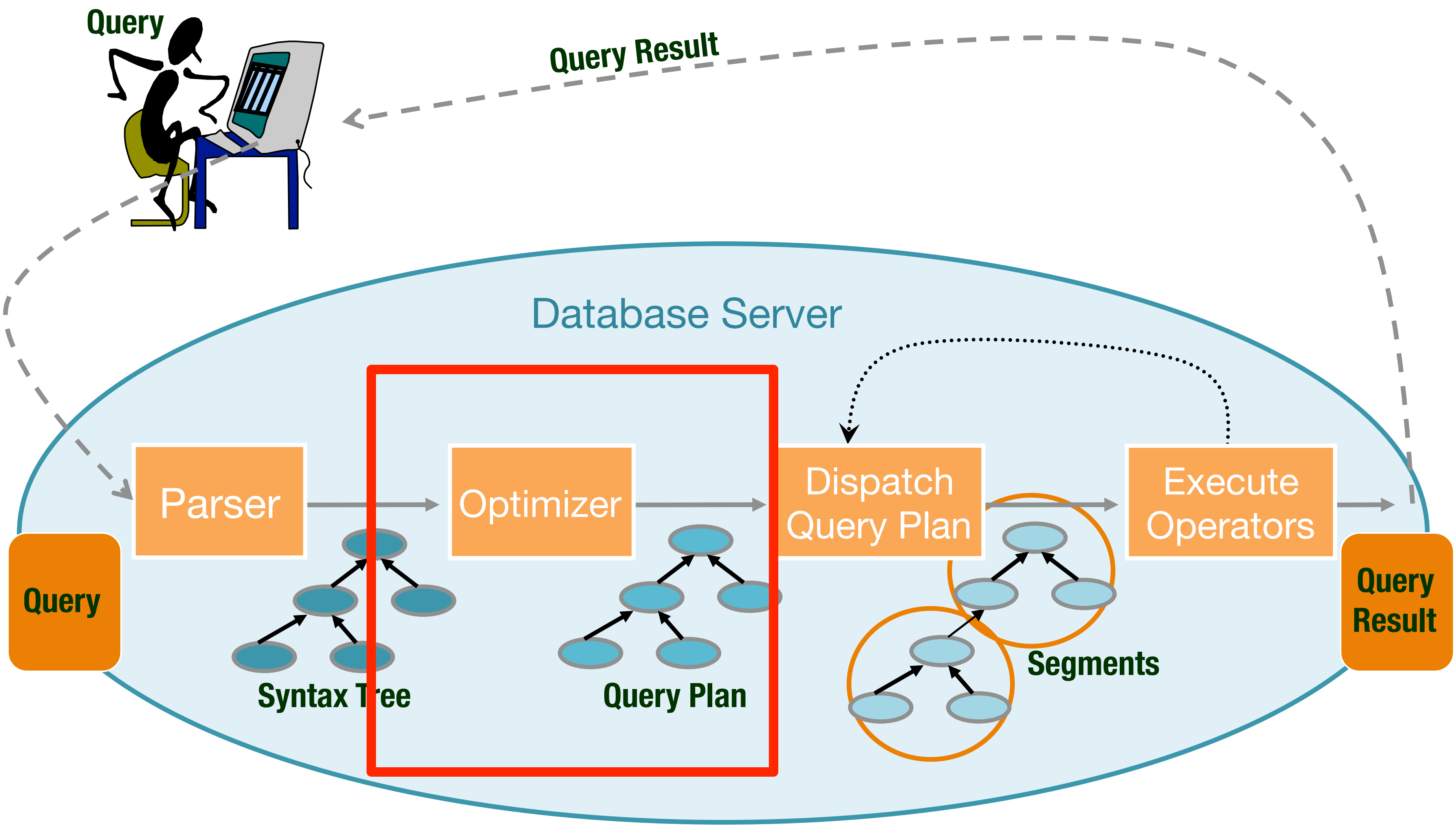
- To help optimize queries, the system keeps information on each relation
  - name, file name, file structure (e.g. heap file)
  - attribute name and type, for each attribute
  - index name, for each index
  - integrity constraints
- For each index:
  - structure (e.g. B+ tree) and search key fields
- For each view:
  - view name and definition
- Plus statistics, authorization, buffer pool size, etc.

Example Catalog: Attribute Catalog

attrName	relName	type	position
attrName	Attribute_cat	string	1
relName	Attribute_cat	string	2
type	Attribute_cat	string	3
position	Attribute_cat	integer	4
sid	Students	string	1
name	Students	string	2
login	Students	string	3
age	Students	integer	4
gpa	Students	real	5
fid	Faculty	string	1
fname	Faculty	string	2
sal	Faculty	real	3

Catalogs are themselves stored as relations

Query Execution Life-Cycle



Query Evaluation Plan

EMP (ssn, ename, addr, sal, did)

DEPT (did, dname, floor, mgr)

```
SELECT E.ename
FROM Emp E, Dept D
WHERE D.dname = 'Toy' AND
      D.did = E.did
```

EMP

ssn	ename	addr	sal	did
13b	Mary Lou Retton	First	9,000	1
29g	Jackie Smith	Main	3,500	2

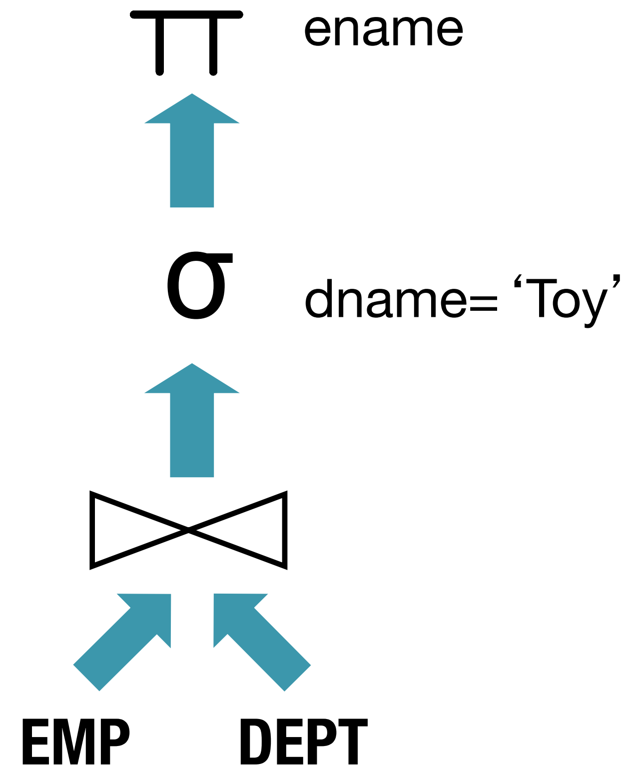
DEPT

did	dname	floor	mgr
1	Adv	5	Depp
2	Toy	10	Brown

Query Evaluation Plan

EMP (ssn, ename, addr, sal, did)
DEPT (did, dname, floor, mgr)

```
SELECT E.ename
FROM Emp E, Dept D
WHERE D.dname = 'Toy' AND
      D.did = E.did
```



EMP

ssn	ename	addr	sal	did
13b	Ma			
29g	Jackie Smith	Main	3,500	2

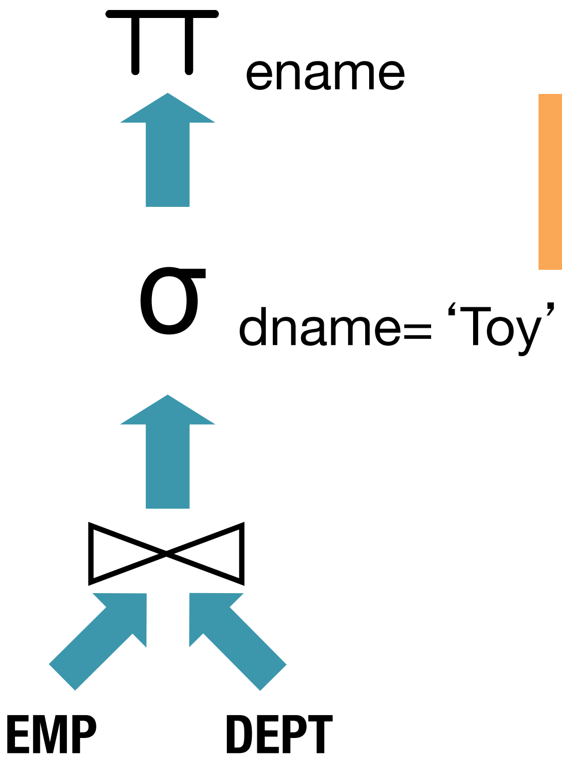
DEPT

did	dname	floor	mgr
			Depp
2	Sales	10	Brown

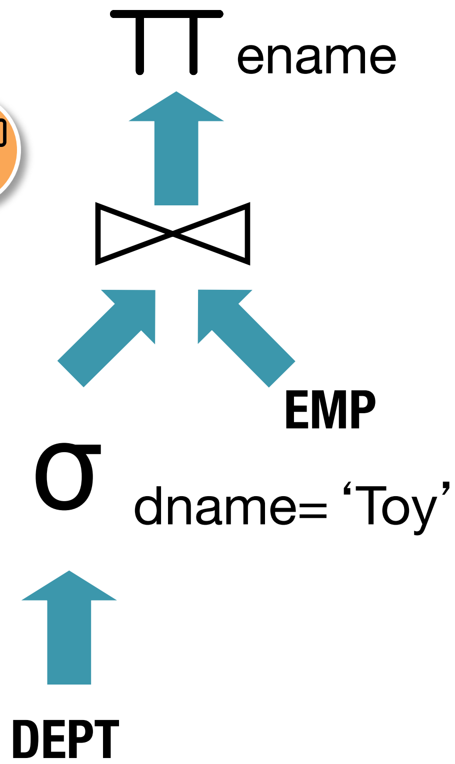
Query Optimizer selects the evaluation plan

Query Optimization Example

1. May modify the query plan



Which plan is cheaper?



2. Also, compares different evaluarion algorithms for SELECT, JOIN, and PROJECT operator based on whether an index is available, table sizes, etc.

Query Optimization

Query optimizer selects an evauation plan with the lease cost in two steps:

1. **Plan Enumeration**
   - Different query plans
   - Different implementations (i.e., evaluation algorithms) for each operator
2. **Cost Estimation**
   - Cost of each operator
   - Overall cost of the plan

Next Week

Today

Operator Evaluation

- How to implement common operators?
  - Selection
    - Matching Indexes
  - Join
  - Projection (optional DISTINCT)
  - Set Difference
  - Union
  - Aggregate operators (SUM, MIN, MAX, AVG)
  - GROUP BY

Selection (on one table)

- Access path defines a strategy to do a *selection* on a table, possibly utilizing an index
- Example of a selection condition
  - a predicate: gpa > 3.0 and age = 21
- Examples of access paths
  - File scan
  - Index that *matches* a selection in the query. Examples:
    - B+tree index on the <gpa, age> attributes
    - B+tree index on gpa
    - B+tree index on age
    - Hash index on age

Selection Cost

- `WHERE R.a` *`op`* `value`
- Options:
  - Heap file — Cost: $O(N)$
  - Sorted File — Cost: $O(\log_2 N) + \ldots$
  - Index
    - Hash — Cost: $O(1) + \ldots$
    - B+Tree: Clustered/Unclustered — Cost: $O(\log_F N) + \ldots$

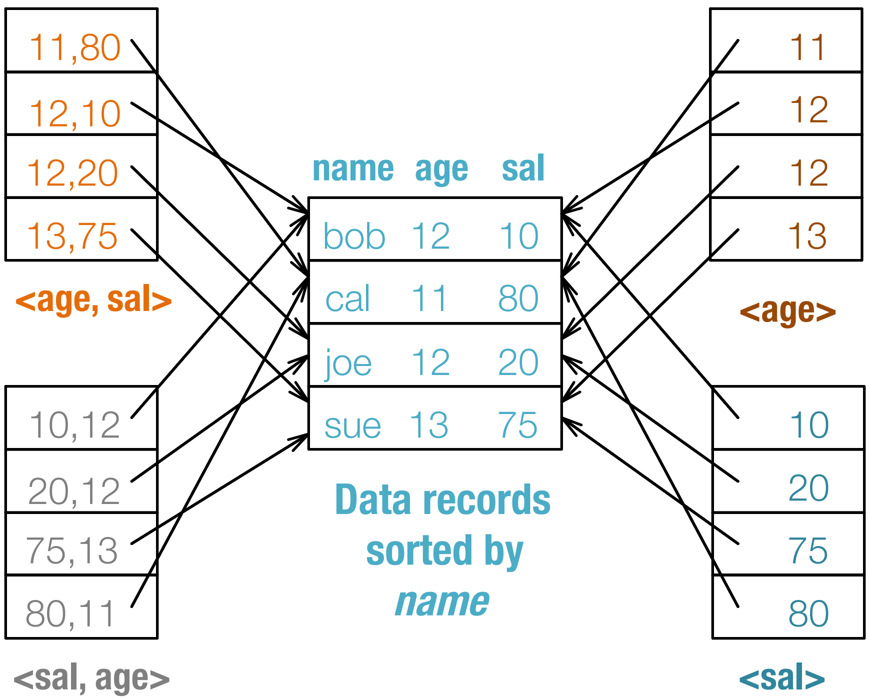
Composite Search Keys

- Index on <age>: search key is a single-attribute age
- Index on <name, age>: search key is composite (name, age) pair
  - For B+-tree, name is the primary comparison attribute and age matters only when names are equal
  - For hash-index, h((name, age)) used – both name and age are needed to hash

Indexes with Composite Search Keys

- Composite Search Keys: Search on a combination of fields
  - Equality query: Every field value is equal to a constant value. e.g. wrt <sal,age> index:
    - age =12 and sal = 75
  - Range query: Some field value is not a constant
    - e.g. age = 12 and sal > 10
- Data entries in index sorted by search key to support range queries

Examples of composite key indexes using lexicographic order



Data entries in index sorted by *<sal,age>*

Data records sorted by *name*

Data entries sorted by *<sal>*

Index Matching

- When can we use an index to evaluate a selection predicate?
- An index matches a predicate if the index can be used to evaluate (at least part of) the predicate

Quiz: Index Matching

- Index on <a, b, c>

	Tree Idx	Hash Idx
a=5 and b= 3	yes	no!
a > 5 and b < 3	yes	no!
b=3	no!	no!
a=7 and b=5 and c=4 and **d>4**	yes	yes
a=7 and c=5	yes	no!

Index matches (part of) a predicate if:

- Conjunction of terms involving only attributes (no disjunctions)
- **Hash:** only equality operation, predicate has all index attributes
- **Tree:** Attributes are a prefix of the search key, any ops

Index Selectivity

- To retrieve Emp records with age=30 AND sal=4000, an index on <age, sal> would be better than an index on age or an index on sal
  - <age, sal> is more selective than just <age> or just <sal>
  - It identifies fewer spurious records that will later be rejected
- If condition is: age=80 AND 3000 < sal <20,000:
  - <age> index much better than <sal> index!
- Ideally, the more selective index preferred

Example: Index Matching

- Predicate could match more than 1 index
- Hash index on <a, b> and B+tree index on <a, c>
- Predicate: a=7 and b=5 and c=4. Which index?
  - Option 0: Neither. Simply use file scan
  - Option 1: More selective one. Then, scan among the selected records
  - Option2: Use both!
  - Algorithm: Intersect rid sets
    - Sort rids, retrieve rids in both sets

Time for review!

quiz: Selection

- Hash index on <a> and Hash index on <b>
  - a=7 **or** b>5 — Which index?
  - Neither! File scan required for b>5
- Hash index on <a> and B+tree on <b>
  - a=7 **or** b>5 — Which index?
  - Option 1: Neither
  - Option 2: Use both! Fetch rids and union
    - Note: Option 1 could be better sometimes. (When?)
- Hash index on <a> and B+tree on <b>
  - (a=7 **or** c>5) **and** b > 5 — Which index?
  - B+-tree (high selectivity) or File Scan (poor selectivity)

When to Use a B+tree Index?

- Consider
  - A relation with 1M tuples
  - 100 tuples on a page
  - 500 (key, rid) pairs on a page

data pages
= 1M/100 = 10K pages
leaf idx pgs
= 1M / (500 * 0.67)
~ 3K pages

	1% SELECTION	10% SELECTION
CLUSTERED	~ 30 + 100	~ 300 + 1000
NON-CLUSTERED	~ 30 + 10,000	~ 300 + 100,000
NC + SORT RIDS	~ 30 + (~10,000)	~ 300 + (~10,000)

⇨ Choice of Index access plan, consider:
1. Index Selectivity 2. Clustering

⇨ Similar consideration for hash-based indices

Summary

- DBMS use catalogs, which are relations themselves.
- Query Optimizer selects the evaluation plan after evaluating the cost of *many* plans. It evaluates the potential algorithms and chooses the best one per operator.
- Single vs. composite search keys
- Each relation may have multiple indexes
  - Selectivity
  - Clustering

Optional Exercises

12.1 (1-4), 12.3, 12.5