Single Table, Index (2.1)

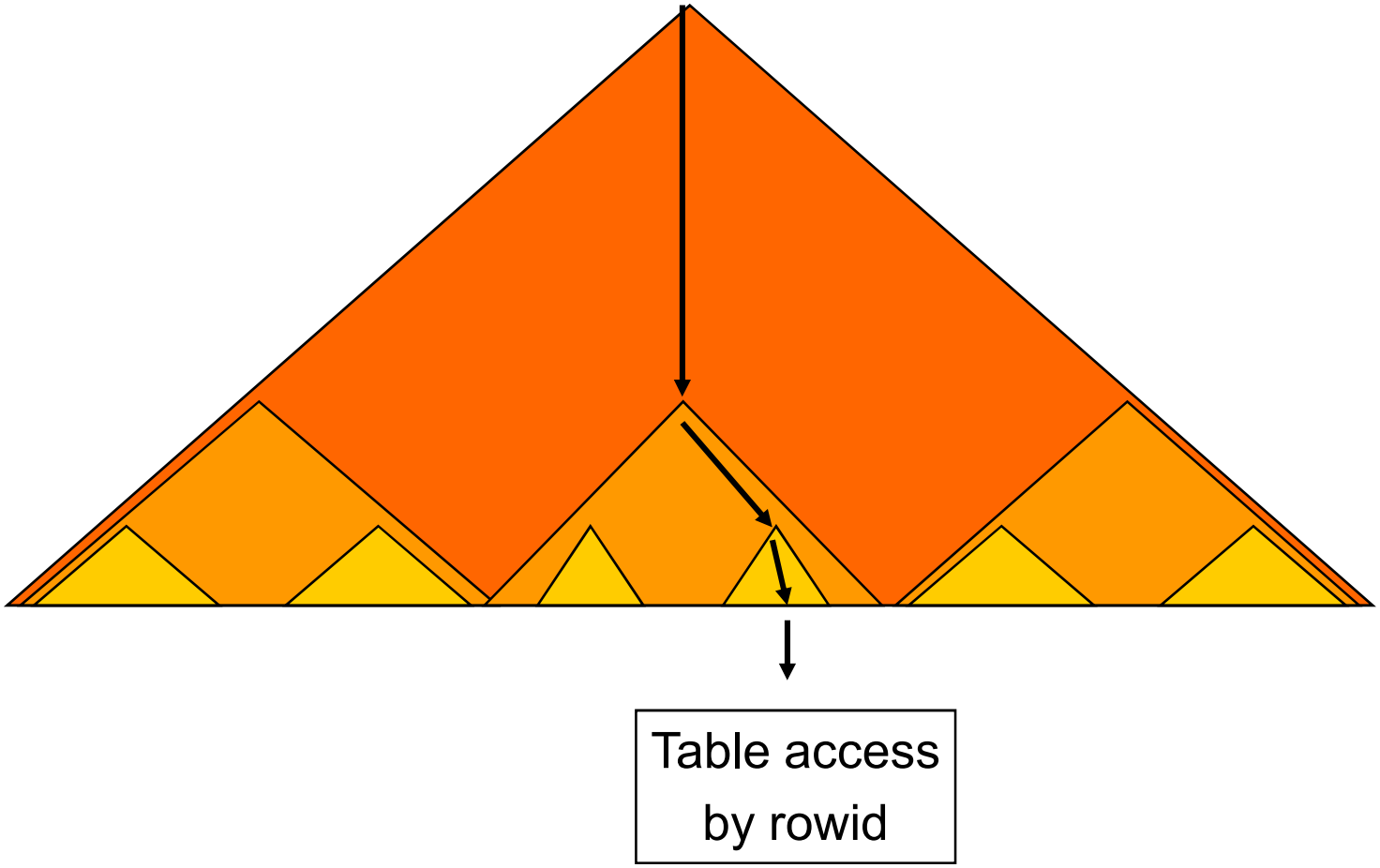
```
SELECT *
FROM emp
WHERE empno=174;

Unique emp(empno)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX unique scan i_emp_pk
```

- Index Unique Scan
  - Traverses the node blocks to locate correct leaf block
  - Searches value in leaf block (if not found => done)
  - Returns rowid to parent row-source
    - Parent: accesses the file+block and returns the row

Index Unique Scan (2.1)



Single Table, Index (2.2)

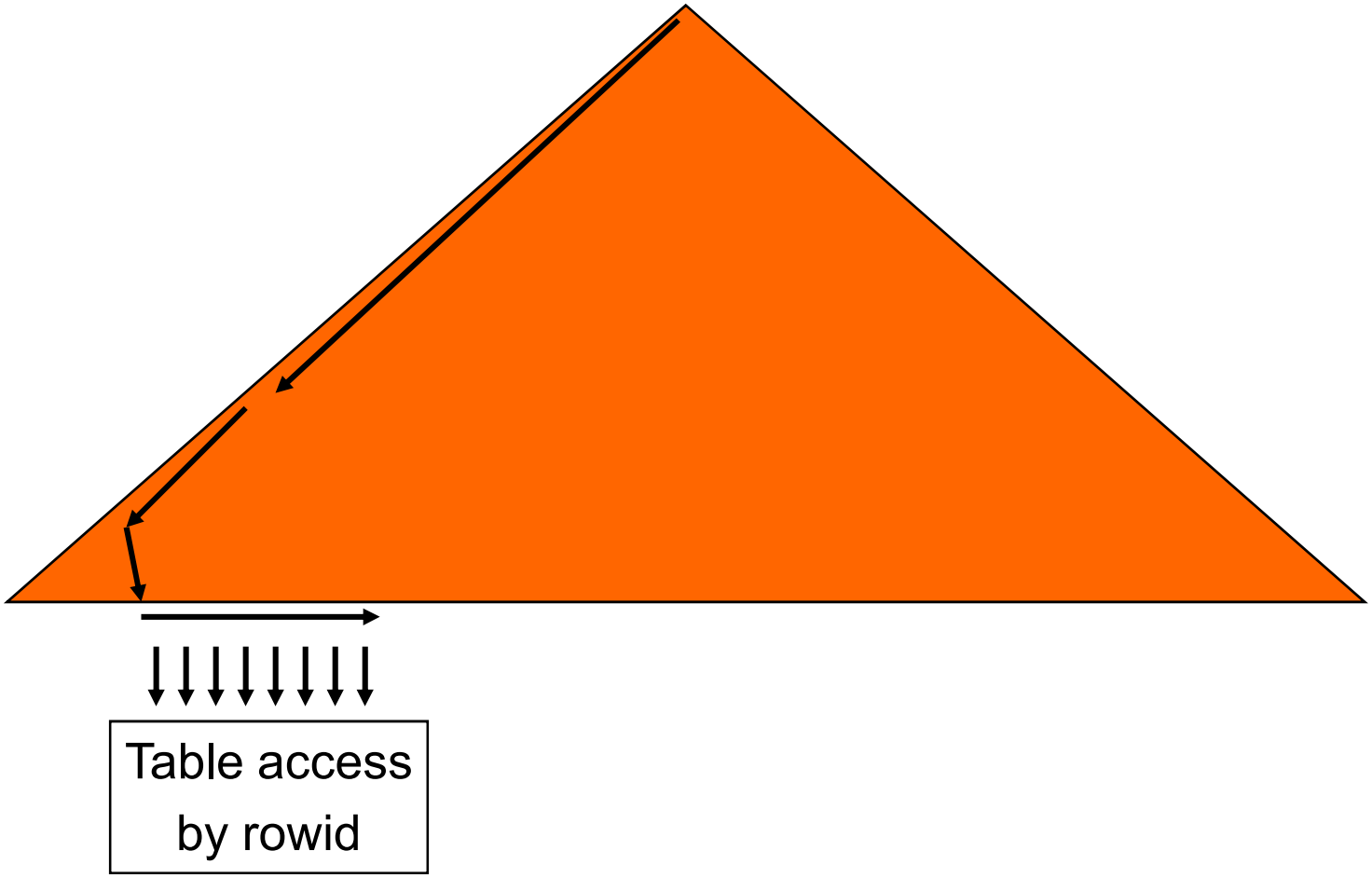
```
SELECT *
FROM emp
WHERE job='manager';

emp(job)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_job
```

- (Non-unique) Index Range Scan
  - Traverses the node blocks to locate most left leaf block
  - Searches 1st occurrence of value in leaf block
  - Returns rowid to parent row-source
    - Parent: accesses the file+block and returns the row
  - Continues on to next occurrence of value in leaf block
    - Until no more occurences

Index Range Scan (2.2)



Single Table, Index (2.3)

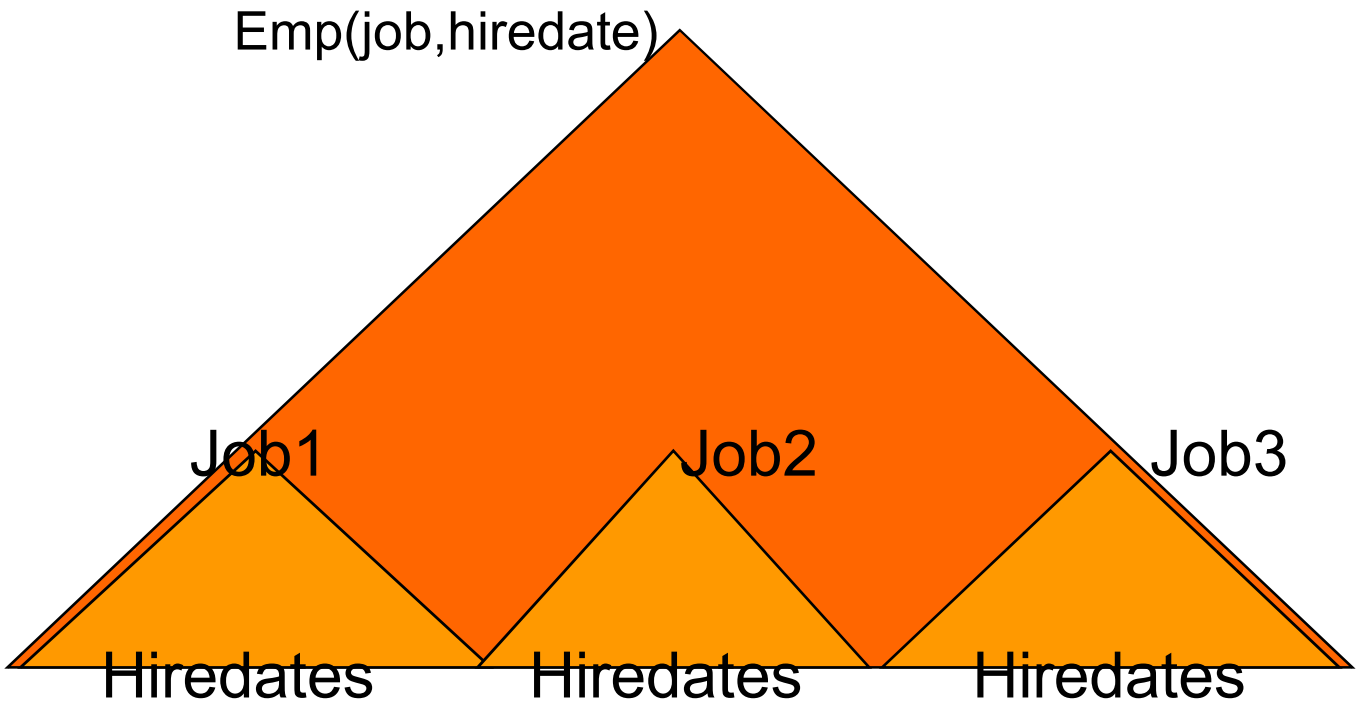
```
SELECT *
FROM emp
WHERE empno>100;

Unique emp(empno)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_pk
```

- Unique Index Range Scan
  - Traverses the node blocks to locate most left leaf block with start value
  - Searches 1st occurrence of value-range in leaf block
  - Returns rowid to parent row-source
    - Parent: accesses the file+block and returns the row
  - Continues on to next valid occurrence in leaf block
    - Until no more occurences / no longer in value-range

Concatenated Indexes



Multiple levels of Btrees, by column order

Single Table, Index (2.4)

```
SELECT *
FROM emp
WHERE job='manager'
AND hiredate='01-01-2001';

Emp(job,hiredate)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_j_h
```

- Full Concatenated Index
  - Use job-value to navigate to sub-Btree
  - Then search all applicable hiredates

Single Table, Index (2.5)

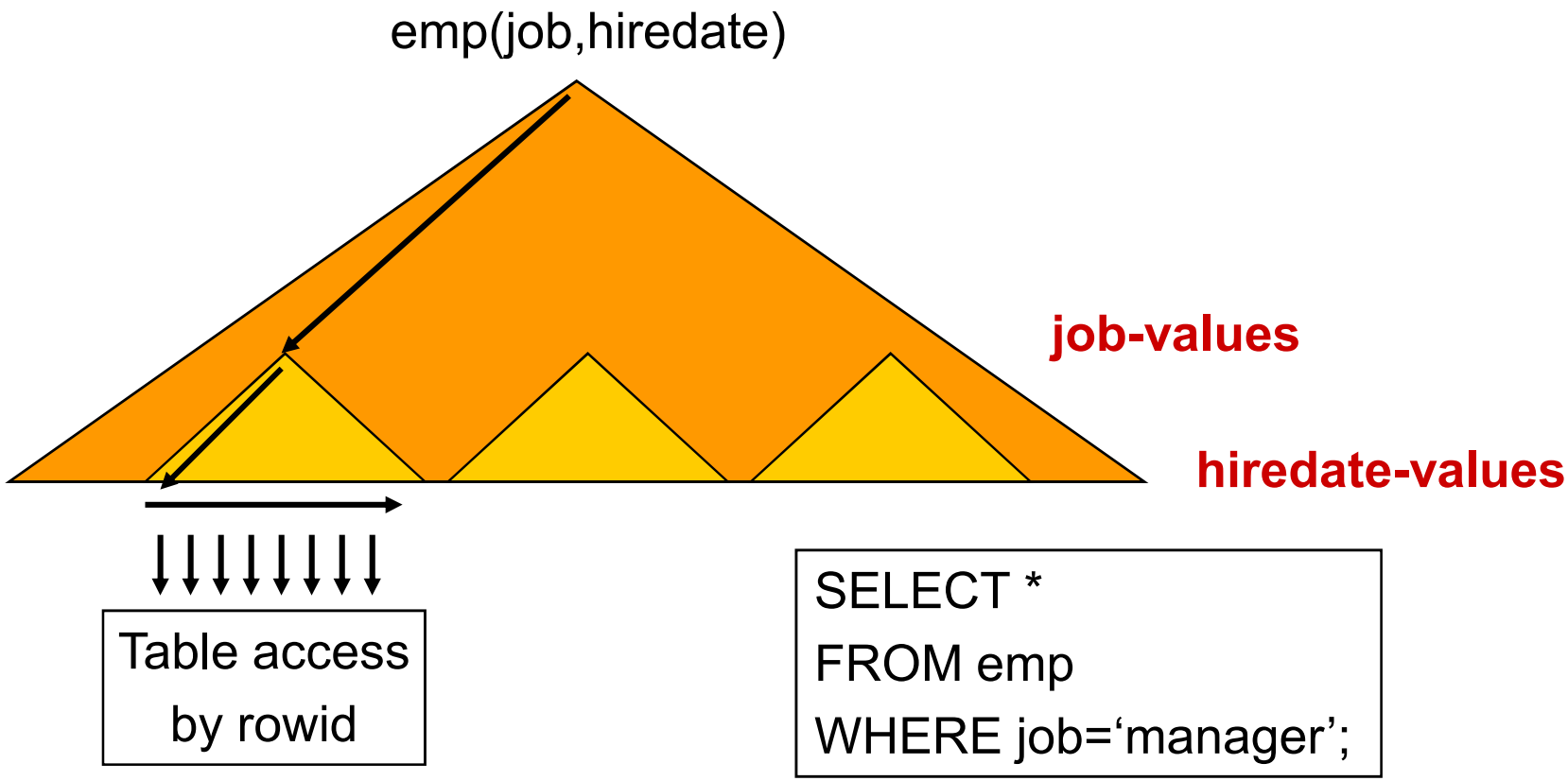
SELECT *
FROM emp
WHERE job='manager';

Emp(job,hiredate)

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_j_h
```

- (Leading) Prefix of Concatenated Index
  - Scans full sub-Btree inside larger Btree

Index Range Scan (2.5)




```
SELECT *
FROM emp
WHERE job='manager';
```

Single Table, Index (2.6)

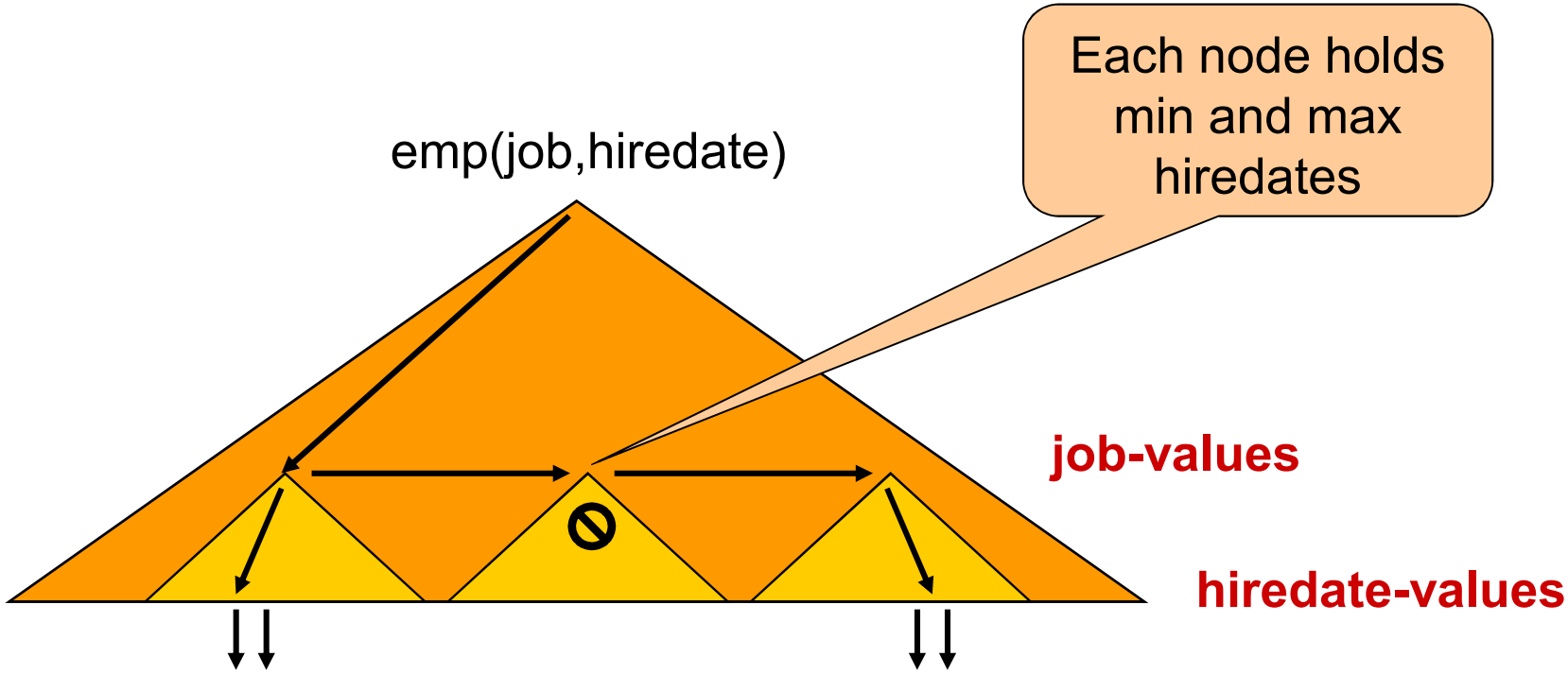
```
SELECT *
FROM emp
WHERE hiredate='01-01-2001';

Emp(job,hiredate)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_j_h
```

- Index Skip Scan (prior versions did FTS)
  - "To use indexes where they've never been used before"
  - Predicate on leading column(s) no longer needed
  - Views Btree as collection of smaller sub-Btrees
  - Works best with low-cardinality leading column(s)

Index Skip Scan (2.6)




```
SELECT *
FROM emp
WHERE hiredate='01-01-2001';
```

Single Table, Index (2.7)

```
SELECT *
FROM emp
WHERE empno>100
AND job='manager';

Unique Emp(empno)
Emp(job)
```

```
>.SELECT STATEMENT
>...TABLE ACCESS by rowid emp
>.....INDEX range scan i_emp_job
```

- Multiple Indexes
  - Rule: uses heuristic decision list to choose which one
    - Avaliable indexes are 'ranked'
  - Cost: computes most selective one (ie. least costing)
    - Uses statistics

RBO Heuristics

- Ranking multiple available indexes
  1. Equality on single column unique index
  2. Equality on concatenated unique index
  3. Equality on concatenated index
  4. Equality on single column index
  5. Bounded range search in index
     - Like, Between, Leading-part, …
  6. Unbounded range search in index
     - Greater, Smaller (on leading part)

  Normally you hint which one to use

CBO Cost Computation

- Statistics at various levels
  - Table:
    - Num_rows, Blocks, Empty_blocks, Avg_space
  - Column:
    - Num_values, Low_value, High_value, Num_nulls
  - Index:
    - Distinct_keys, Blevel, Avg_leaf_blocks_per_key, Avg_data_blocks_per_key, Leaf_blocks
- Used to compute selectivity of each index
  - Selectivity = percentage of rows returned
    - Number of I/O's plays big role
  - FTS is also considered at this time!