

# SQL Tuning

A Look Behind The Curtain

Craig Martin

# Craig Martin

- Applications DBA for Nationwide in Columbus, OH
- Worked with SQL and PL/SQL on Oracle since 2002
- Just like all of you!

# Please Participate!

- If you have a question, ask!
- If you have a comment, share!
- If you disagree, let me know!

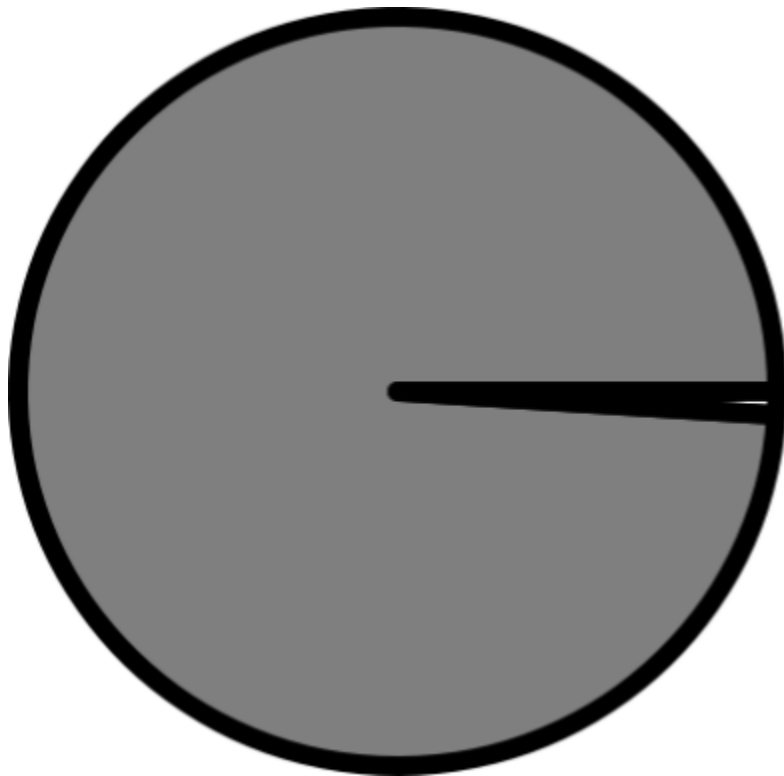
*Whatever you do in life, surround yourself with smart people who'll argue with you.*

*- John Wooden*

# Nationwide Disclaimer

- The opinions I share are mine, and do not necessarily reflect those of Nationwide
- It is up to each individual to evaluate the content in this presentation and do their own testing
- Use this information at your own risk

# Why Learn about Tuning?



- Applications that have needed tuned
- Unused Applications

**Most Effective Tuning**

**Eliminate  
Waste!**

**But I need all this stuff!!!**



# Running Too Often

- Reports running more frequently than useful
  - Just In Time vs. Real Time
- Reports that are no longer useful
  - Even if they are new!
- Scheduled reports for people who are no longer there
- Not using set-based functionality
  - "Row-by-row = slow-by-slow" -- Tom Kyte
  - Watch out for functions in SQL!



# Row-by-Row = Slow-by-Slow!

```
select
  u.user_name,
  users_pkg.last_active_dt(u.user_id) last_active_dt
from
  users u;
```

USERS\_PKG.LAST\_ACTIVE\_DT(:1); -- Takes ~1 sec

select count(\*) from users; -- ~21,000 rows

21,000 rows x 1 second per = 21,000 seconds = ~5.8 hours!

# Too Many Rows

- Not filtered enough
- Trying to answer too many questions at one time
- Not using pagination
  - Allows you to take advantage of first rows optimizations

# Too Many Rows

## Not Using Pagination:

- Entire result set returned, either displayed all at once, or split up only on front end
  - 1st page = 10 seconds
  - 2nd page < 1 second, 3rd page < 1 second, etc.

## Using Pagination:

- Only rows specific to current page retrieved and displayed
  - 1st page < 1 second
  - 2nd page = 1 second, 3rd page = 3 seconds, etc.

# Too Many Rows

Google Result Impressions Percentage

1	2,834,806	34.35%
2	1,399,502	16.96%
3	942,706	11.42%
4	638,106	7.73%
5	471,721	6.19%
6	416,887	5.05%
7	331,500	4.02%
8	286,118	3.47%
9	235,197	2.85%
10	223,320	2.71%
11	91,978	1.11%
12	69,778	0.85%
13	57,053	0.70%
14	46,822	0.57%
15	39,635	0.48%
16	32,168	0.39%
17	26,933	0.33%
18	23,131	0.28%
19	22,027	0.27%
20	23,953	0.29%

# Who Can Tune SQL?



# Diagram-Based Deterministic Method

- Eliminate Guessing
- Don't need years and years of experience
- Simplify the Problem!!!

Train A leaves LA heading east at...

vs.

$$12a + 3x = 60 + 3x$$

# Who Can Tune SQL?

**Oz:** I might not actually be a wizard...

**Glinda:** Yes, but they don't know that.

- Oz the Great and Powerful

# Creating Diagrams

- Nodes = Tables
- Links = Joins
- Inner Numbers = Filter Condition Selectivity
- Outer Numbers = Join Ratios

## **Does not contain:**

- Select Lists
- Ordering / Aggregation
- Table Names
- Detailed Join / Filter conditions



# Why it works

Goal = Eliminate Waste!

- Find the correct starting point
- Find the correct join order
- Touch the least amount of rows possible
- Except for rare occasions, everything else is minor!
- Usually no need to change SQL

# Creating Diagrams

1. Pick any table in query and draw a node to represent it
2. Look for joins to this table's PK / UK
  - a. Draw down arrow into this node and create node for joined table at top of arrow.
3. Look for joins from current table to PK/UK of another table
  - a. Draw down arrow from this node and create node for joined table at bottom of arrow
4. Repeat until all tables on diagram
5. Fill in Filter and Join Ratios
  - a. Asterisk by filter ratio when always will return 1 row

# Creating Diagrams

```
select d.dept_name, e.fname, e.lname  
from employees e  
    inner join departments d  
        on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```

# Creating Diagrams

1. Pick any table in query and draw a node to represent it

```
select d.dept_name, e.fname,  
e.lname  
from employees e  
    inner join departments d  
        on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```



# Creating Diagrams

2. Look for joins to this table's PK / UK
  - a. Draw down arrow into this node and create node for joined table at top of arrow.

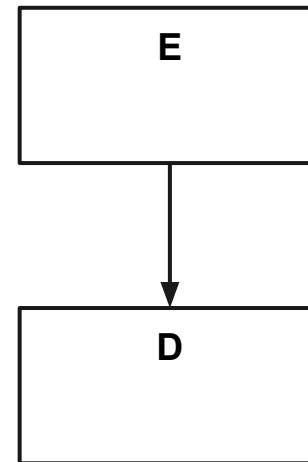
```
select d.dept_name, e.fname,  
e.lname  
from employees e  
  inner join departments d  
    on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```



# Creating Diagrams

3. Look for joins to PK/UK of another table
  - a. Draw down arrow from this node and create node for joined table at bottom of arrow

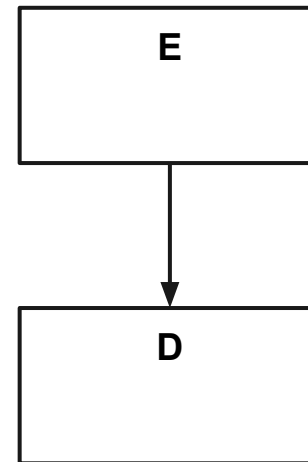
```
select d.dept_name, e.fname,  
e.lname  
from employees e  
inner join departments d  
on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```



# Creating Diagrams

## 4. Repeat until all tables on diagram

```
select d.dept_name, e.fname,  
e.lname  
from employees e  
    inner join departments d  
        on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```



# Creating Diagrams

## 5. Fill in Filter and Join Ratios

Q1: select count(\*) from employees;

**10,000**

Q2: select count(\*) from employees where salary > 75000;

**1,000**

Q3: select count(\*) from departments;

**50**

Q4: select count(\*) from departments where region = 'OHIO';

**45**

Q5: select count(\*) from employees e

inner join departments d

on d.dept\_id = e.dept\_id;

**9,900**



# Creating Diagrams

## 5. Fill in Filter and Join Ratios

Filter Ratio = % of table rows returned after filter applied

$$E = 1,000 / 10,000 = .1 \quad D = 45 / 50 = .9$$

Join Ratio = How many rows returned for every 1 row in joined table

$$\text{Master} = 9,900 / 10,000 = .99 \quad \text{Detail} = 9,900 / 50 = 198$$

# Creating Diagrams

## 5. Fill in Filter and Join Ratios

- Asterisk by filter ratio when always will return 1 row

Filter Ratio

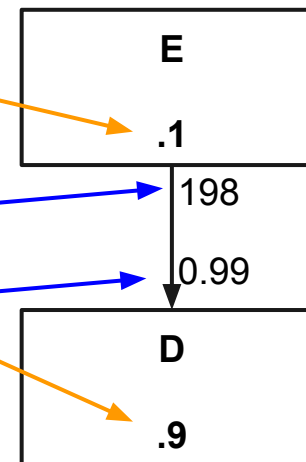
$$E = 1,000 / 10,000 = .1$$

$$D = 45 / 50 = .9$$

Join Ratio

$$\text{Detail} = 9,900 / 50 = 198$$

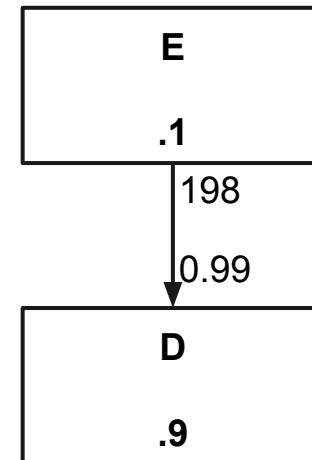
$$\text{Master} = 9,900 / 10,000 = .99$$



# Creating Diagrams

```
select d.dept_name, e.fname,  
e.lname  
from employees e  
    inner join departments d  
        on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```

=



# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
  inner join departments d on d.dept_id = e.dept_id
  inner join employees m on m.emp_id = e.mgr_id
  inner join jobs j on j.job_id = e.job_id
  inner join divisions jd on jd.div_id = j.div_id
  inner join addresses wa on a.addr_id = d.dept_addr_id
  inner join addresses ha on ha.addr_id = e.addr_id
  inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```

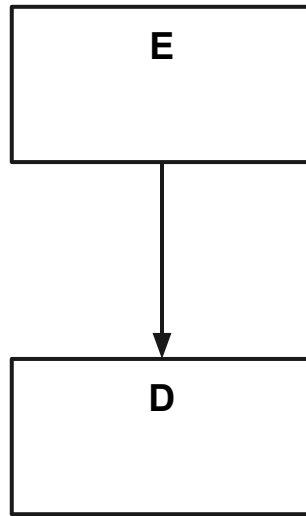
# Creating Diagrams



# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
  inner join departments d on d.dept_id = e.dept_id
  inner join employees m on m.emp_id = e.mgr_id
  inner join jobs j on j.job_id = e.job_id
  inner join divisions jd on jd.div_id = j.div_id
  inner join addresses wa on a.addr_id = d.dept_addr_id
  inner join addresses ha on ha.addr_id = e.addr_id
  inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```

# Creating Diagrams

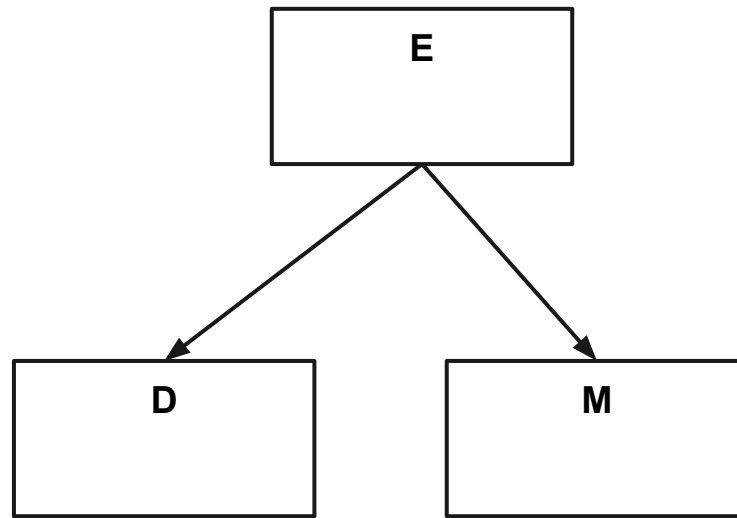


# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
inner join departments d on d.dept_id = e.dept_id
  inner join employees m on m.emp_id = e.mgr_id
  inner join jobs j on j.job_id = e.job_id
  inner join divisions jd on jd.div_id = j.div_id
  inner join addresses wa on a.addr_id = d.dept_addr_id
  inner join addresses ha on ha.addr_id = e.addr_id
  inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```



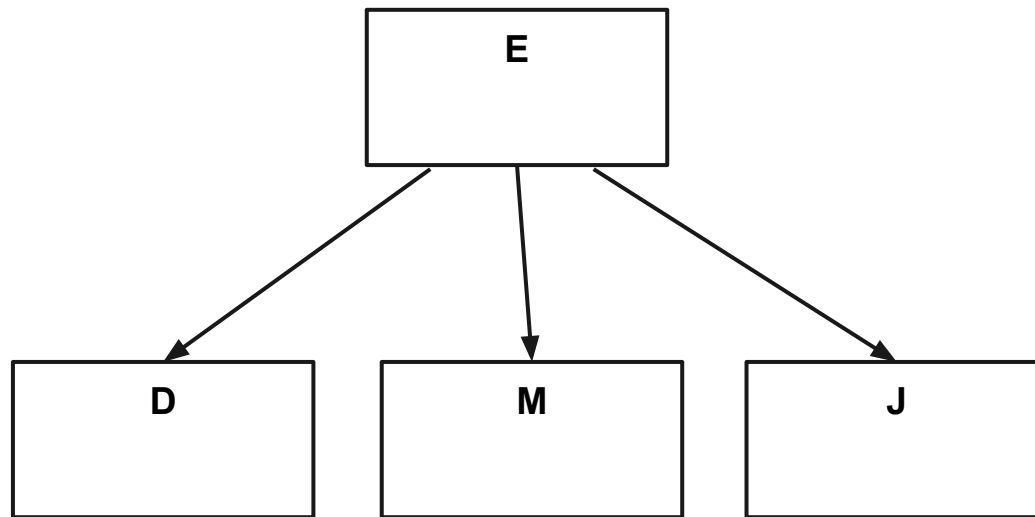
# Creating Diagrams



# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
inner join departments d on d.dept_id = e.dept_id
inner join employees m on m.emp_id = e.mgr_id
inner join jobs j on j.job_id = e.job_id
inner join divisions jd on jd.div_id = j.div_id
inner join addresses wa on a.addr_id = d.dept_addr_id
inner join addresses ha on ha.addr_id = e.addr_id
inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```

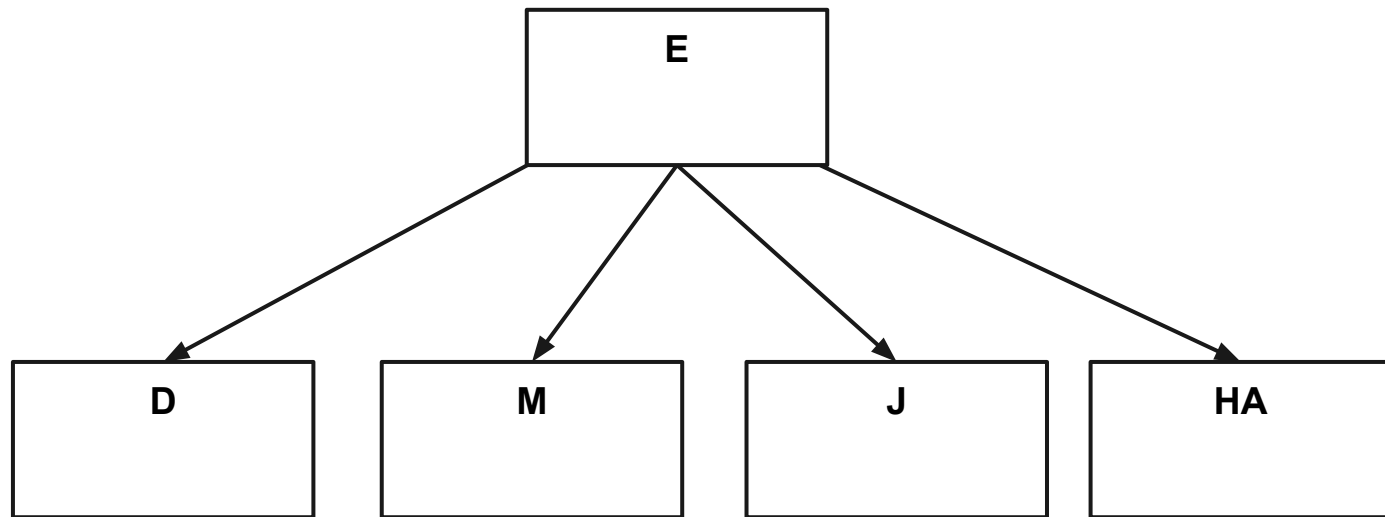
# Creating Diagrams



# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
inner join departments d on d.dept_id = e.dept_id
inner join employees m on m.emp_id = e.mgr_id
inner join jobs j on j.job_id = e.job_id
inner join divisions jd on jd.div_id = j.div_id
inner join addresses wa on a.addr_id = d.dept_addr_id
inner join addresses ha on ha.addr_id = e.addr_id
inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```

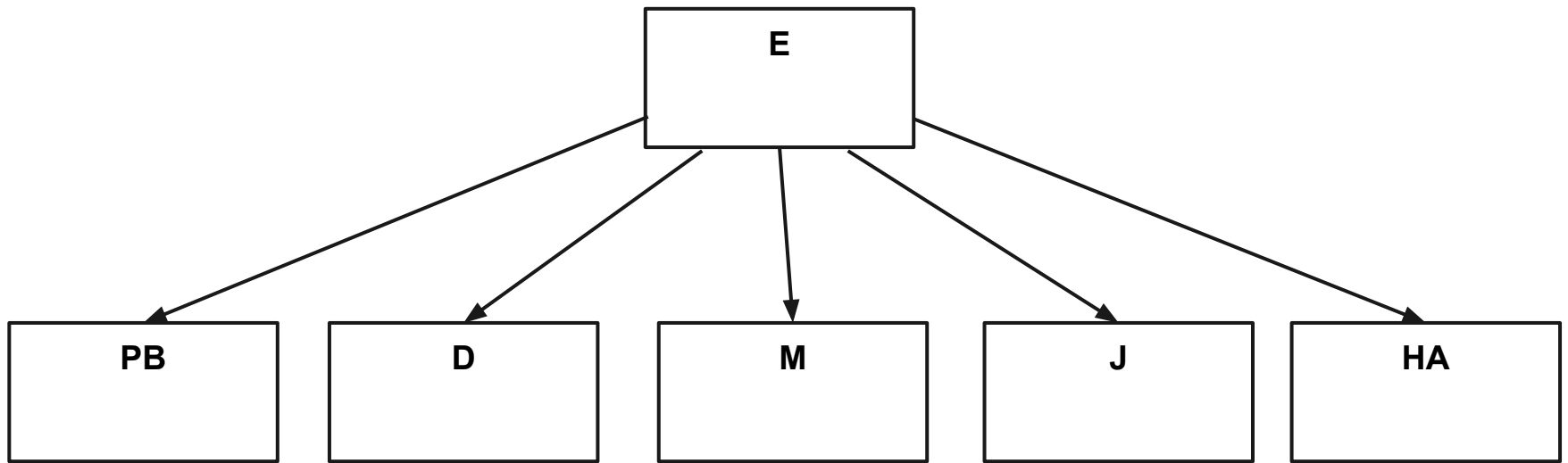
# Creating Diagrams



# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
inner join departments d on d.dept_id = e.dept_id
inner join employees m on m.emp_id = e.mgr_id
inner join jobs j on j.job_id = e.job_id
inner join divisions jd on jd.div_id = j.div_id
inner join addresses wa on a.addr_id = d.dept_addr_id
inner join addresses ha on ha.addr_id = e.addr_id
inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```

# Creating Diagrams

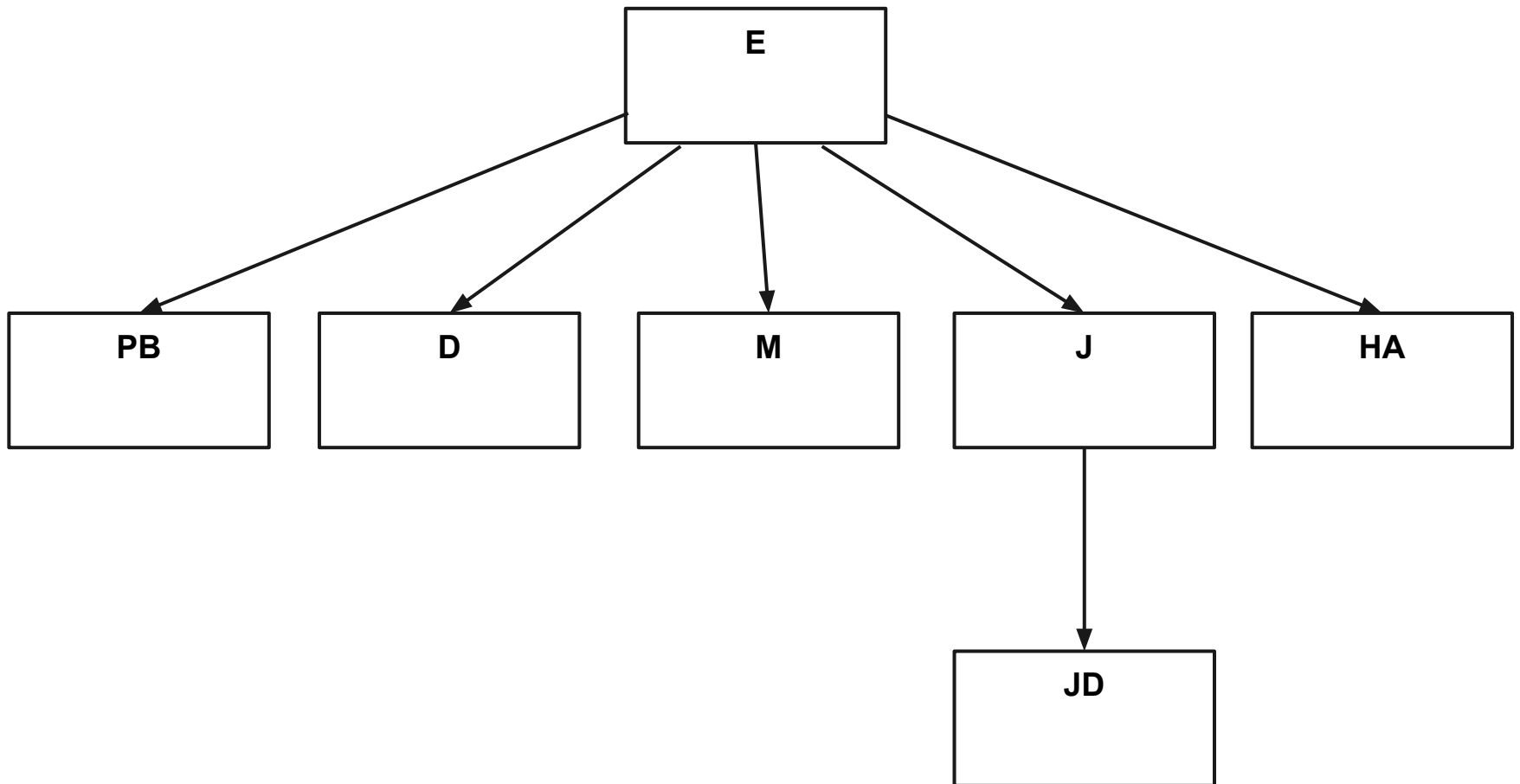


# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
inner join departments d on d.dept_id = e.dept_id
inner join employees m on m.emp_id = e.mgr_id
inner join jobs j on j.job_id = e.job_id
  inner join divisions jd on jd.div_id = j.div_id
  inner join addresses wa on a.addr_id = d.dept_addr_id
inner join addresses ha on ha.addr_id = e.addr_id
inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```



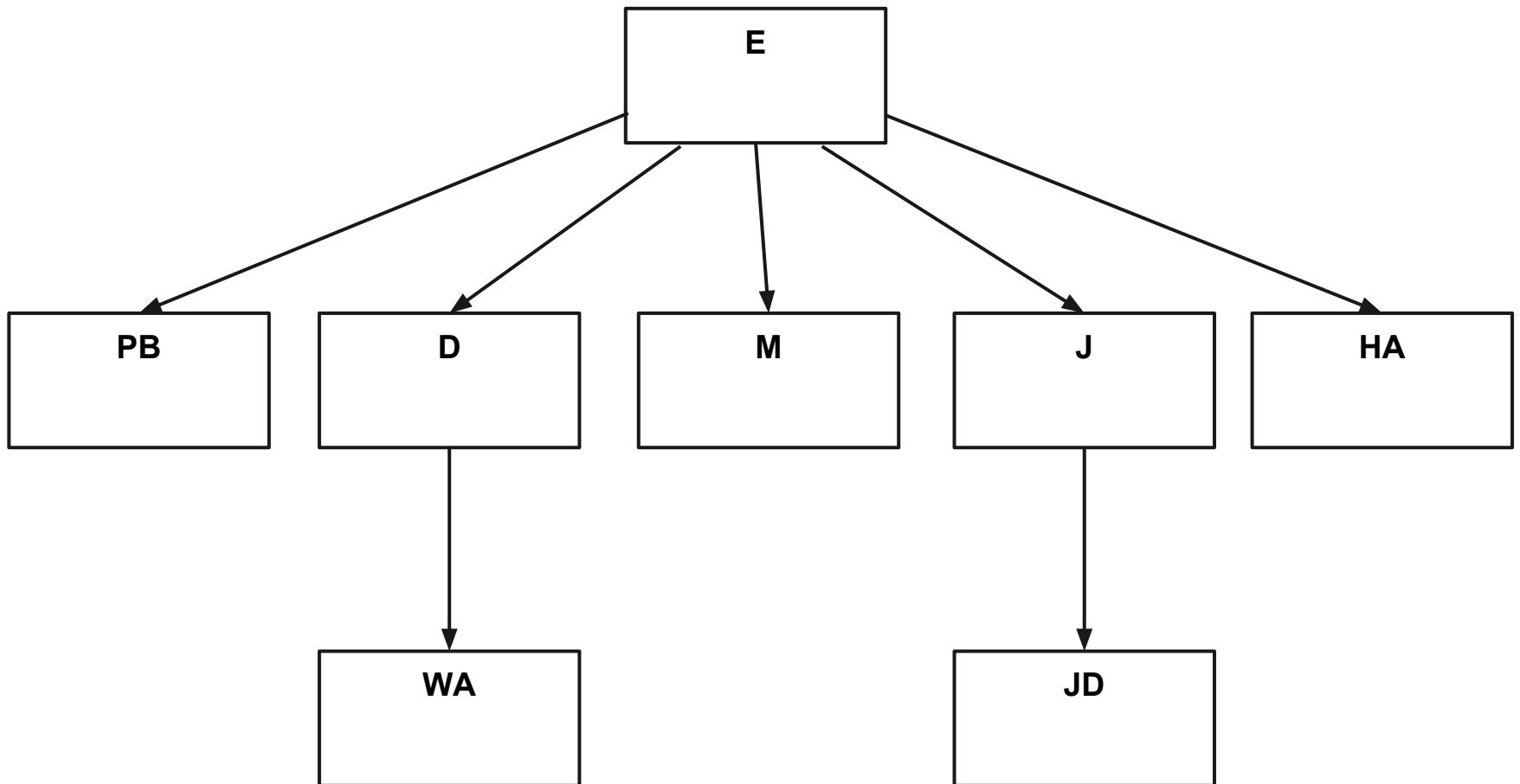
# Creating Diagrams



# Creating Diagrams

```
select
  d.dept_name, e.fname, e.lname, m.lname mgr, j.job_title, jd.div_name, wa.city work_city,
  ha.city home_city, ha.state home_state, pb.name pay_band
from employees e
inner join departments d on d.dept_id = e.dept_id
inner join employees m on m.emp_id = e.mgr_id
inner join jobs j on j.job_id = e.job_id
inner join divisions jd on jd.div_id = j.div_id
inner join addresses wa on a.addr_id = d.dept_addr_id
inner join addresses ha on ha.addr_id = e.addr_id
inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary
where e.status = 'Exempt'
and j.job_class <> 'Executive'
and wa.state = 'OH'
and ha.state <> 'OH'
and jd.area = 'IT'
and d.region = 'OHIO'
order by wa.city, d.dept_name, pb.pay_band desc, e.lname;
```

# Creating Diagrams



# Creating Diagrams

- Q1: select count(\*) from employees; = **10,000**
- Q2: select count(\*) from employees where status = 'Exempt'; = **8,000**
- Q3: select count(\*) from departments; = **50**
- Q4: select count(\*) from departments where region = 'OHIO'; = **45**
- Q5: select count(\*) from jobs; = **200**
- Q6: select count(\*) from jobs where job\_class <> 'Executive'; = **195**
- Q7: select count(\*) from divisions; = **25**
- Q8: select count(\*) from divisions where area = 'IT'; = **3**
- Q9: select count(\*) from addresses; = **15,000**
- Q10: select count(\*) from addresses where state = 'OH'; = **14,750**
- Q11: select count(\*) from paybands; = **8**

# Creating Diagrams

Q12: select count(\*) from employees e  
inner join departments d on d.dept\_id = e.dept\_id; = 9,900

Q13: select count(\*) from employees e  
inner join employees m on m.emp\_id = e.mgr\_id; = 9,999

Q13: select count(\*) from employees e  
inner join jobs j on j.job\_id = e.job\_id; = 10,000

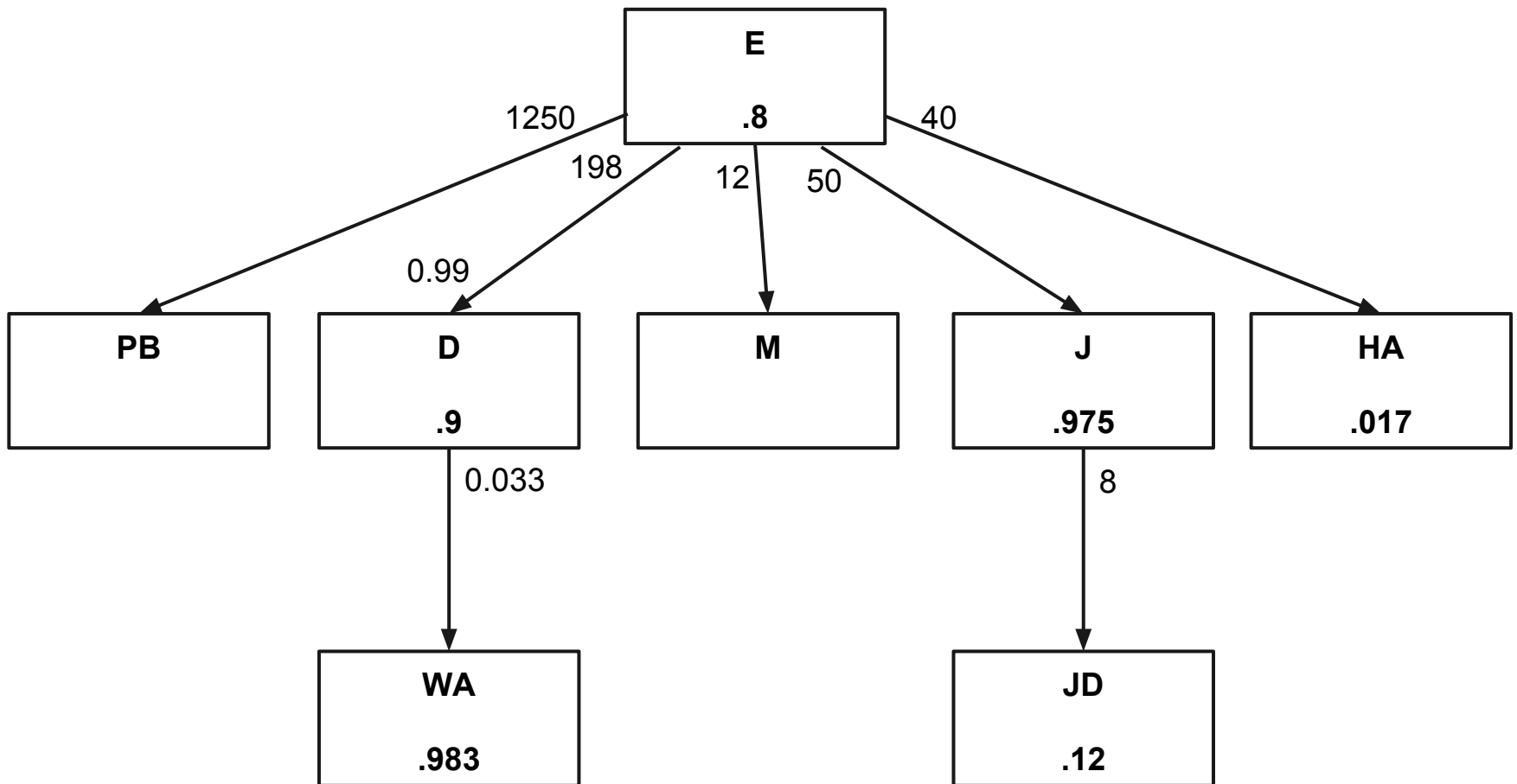
Q13: select count(\*) from employees e  
inner join paybands pb on pb.lowval <= e.salary and pb.highval > e.salary; = 10,000

Q13: select count(\*) from employees e  
inner join addresses ha on ha.addr\_id = e.addr\_id; = 10,000

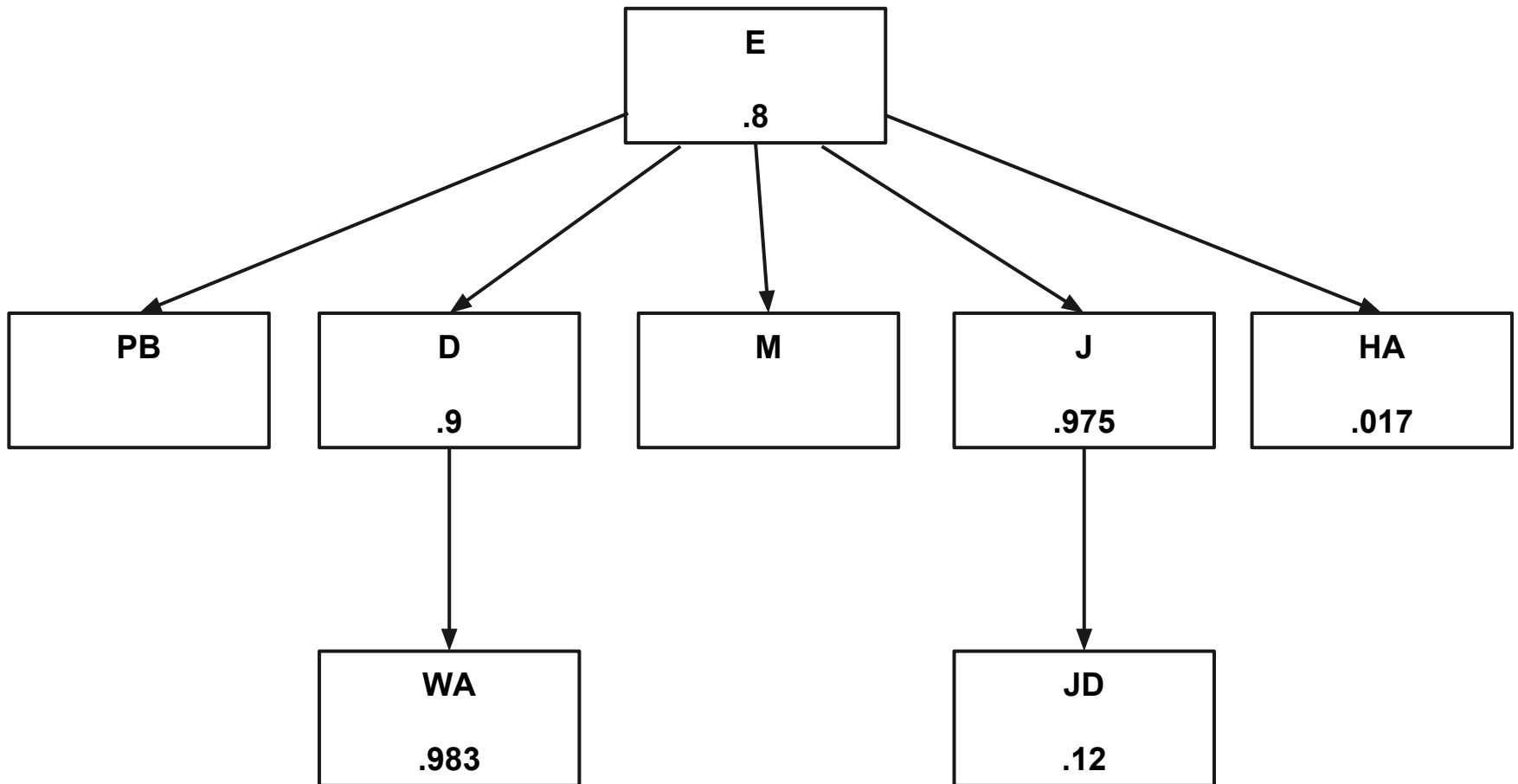
Q13: select count(\*) from departments d  
inner join addresses wa on wa.addr\_id = d.dept\_addr\_id; = 50

Q13: select count(\*) from jobs j  
inner join divisions jd on jd.div\_id = j.div\_id; = 200

# Creating Diagrams



# Creating Diagrams



# Interpreting Diagrams

## Standard Steps:

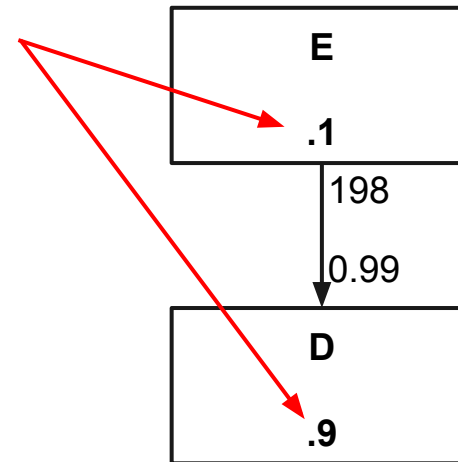
1. Start with table with best (smallest) filter ratio
2. Drive through full, unique indexes downward as far as possible, choosing nodes with best filter ratios first
3. Only when necessary, drive up diagram links through full, non-unique FK indexes
4. All other things being equal, choose path that will get you to lower filter ratios sooner



# Interpreting Diagrams

1. Start with table with best (smallest) filter ratio

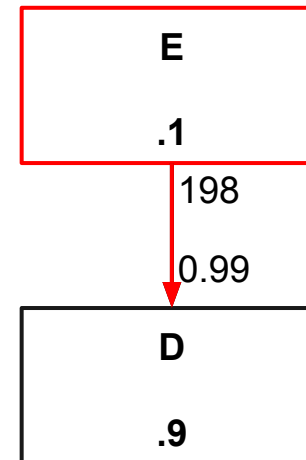
```
select d.dept_name, e.fname,  
e.lname  
from employees e  
    inner join departments d  
        on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```



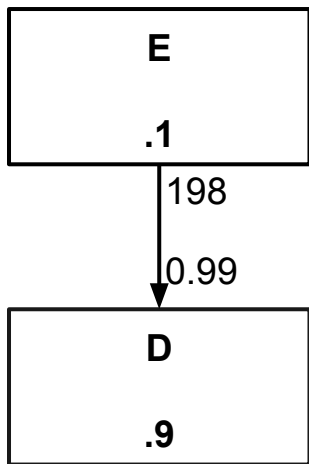
# Interpreting Diagrams

2. Drive through full, unique indexes downward as far as possible, choosing nodes with best filter ratios first

```
select d.dept_name, e.fname,  
e.lname  
from employees e  
  inner join departments d  
    on d.dept_id = e.dept_id  
where e.salary > 75000  
and d.region = 'OHIO';
```



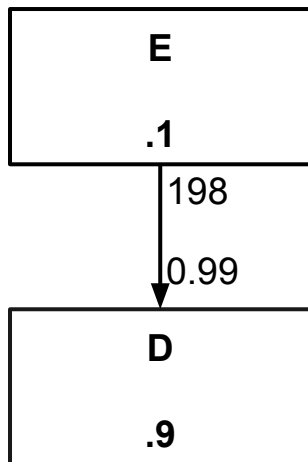
# Interpreting Diagrams



## Desired Execution Plan:

1. Start with Employees, filtering on Salary
2. Join to Departments on DEPT\_ID, where filter on Region is applied

# Does it work?



1. Start with Employees, filtering on Salary  
Read 1,000 rows via index access
2. Join to Departments on DEPT\_ID, where filter on Region is applied  
Nested Loops = 990 reads to Departments, where 99 rows are discarded

**Final Rowcount = 891**

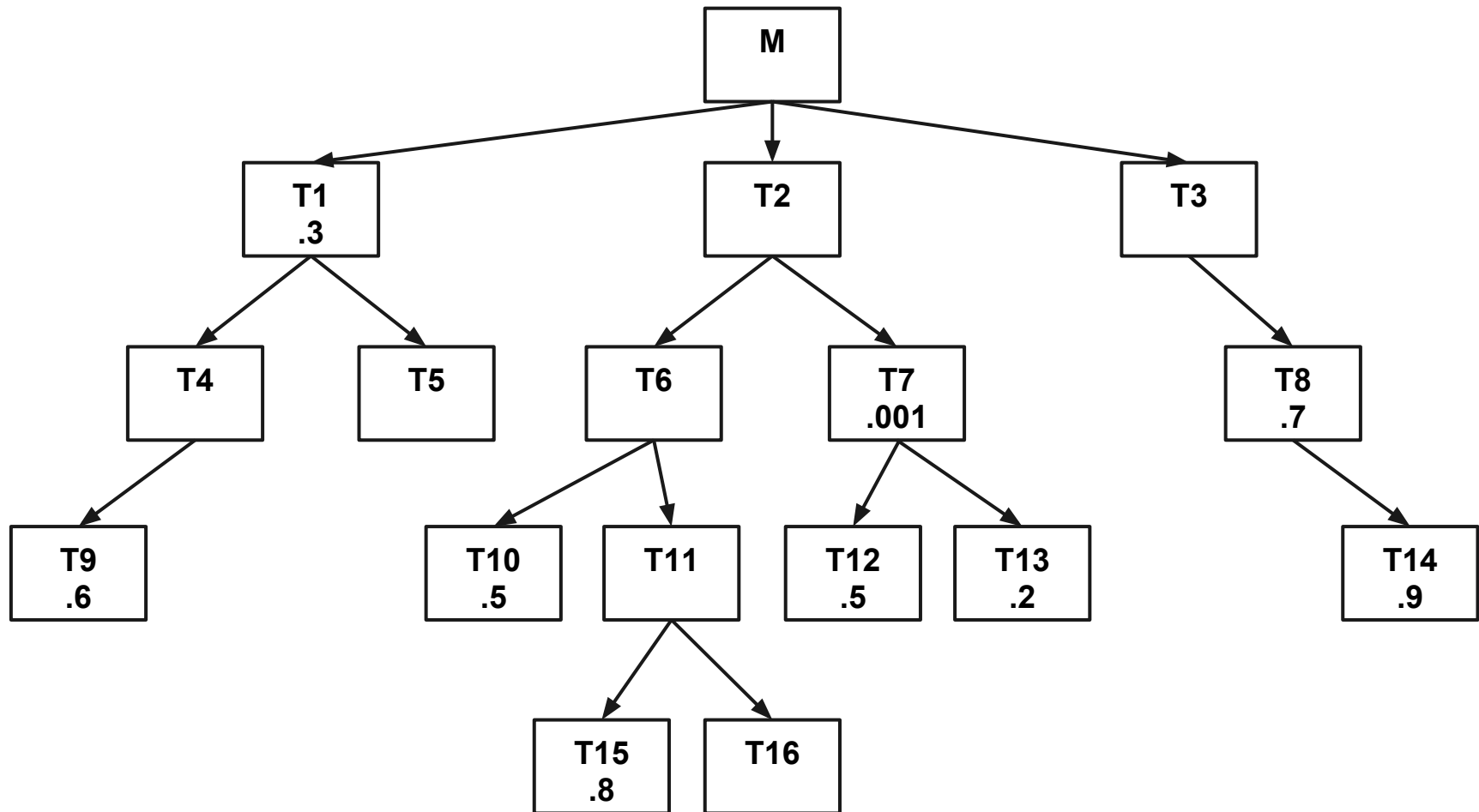
**Total Rows Read = 1,990**

1. Start with Departments, filtering on Region  
Read 45 rows via index access
2. Join to Employees on DEPT\_ID, where filter on Salary is applied  
Nested Loops = 8,910 reads to Employees, where 8,019 rows are discarded

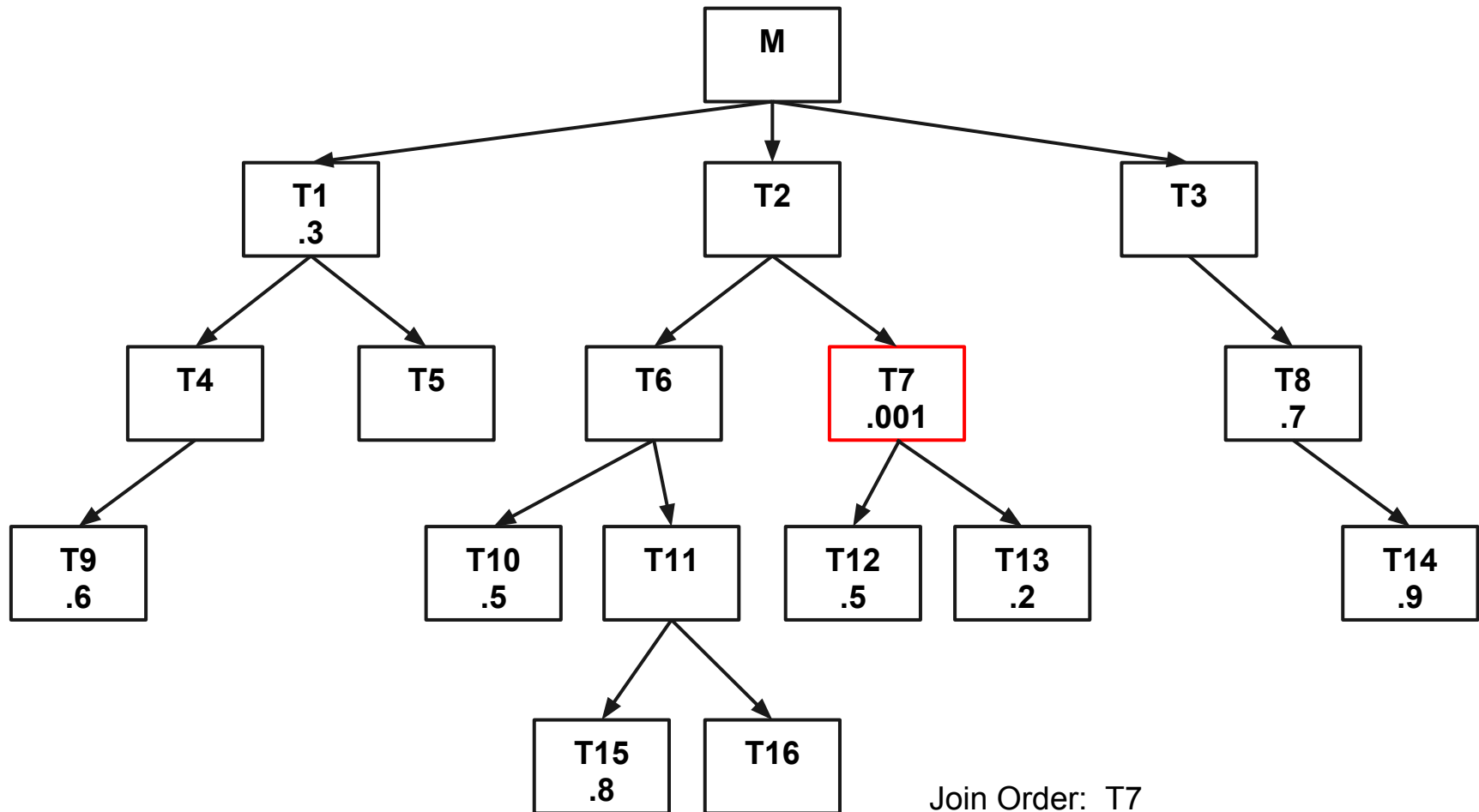
**Final Rowcount = 891**

**Total Rows Read = 8,955**

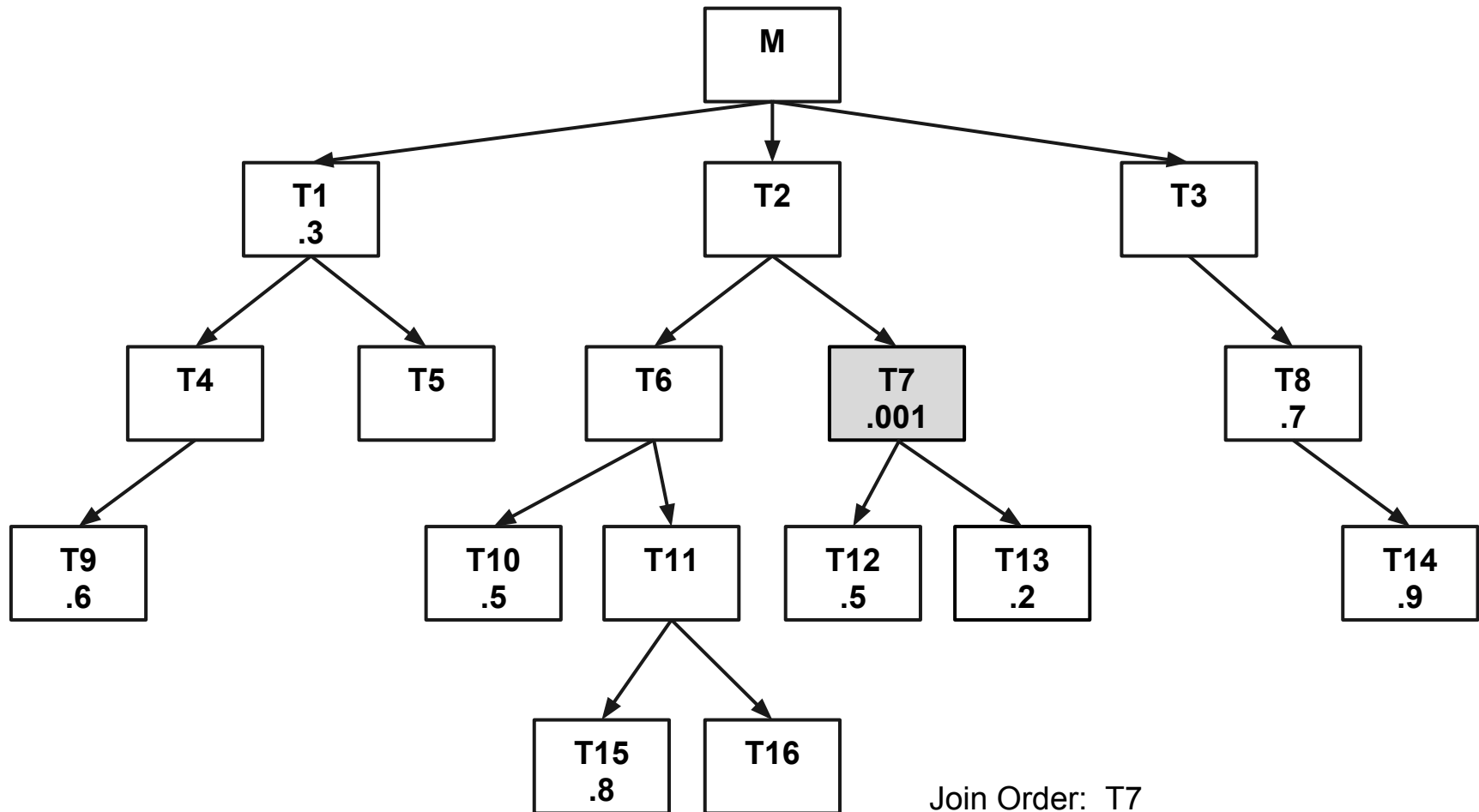
# Interpreting Diagrams



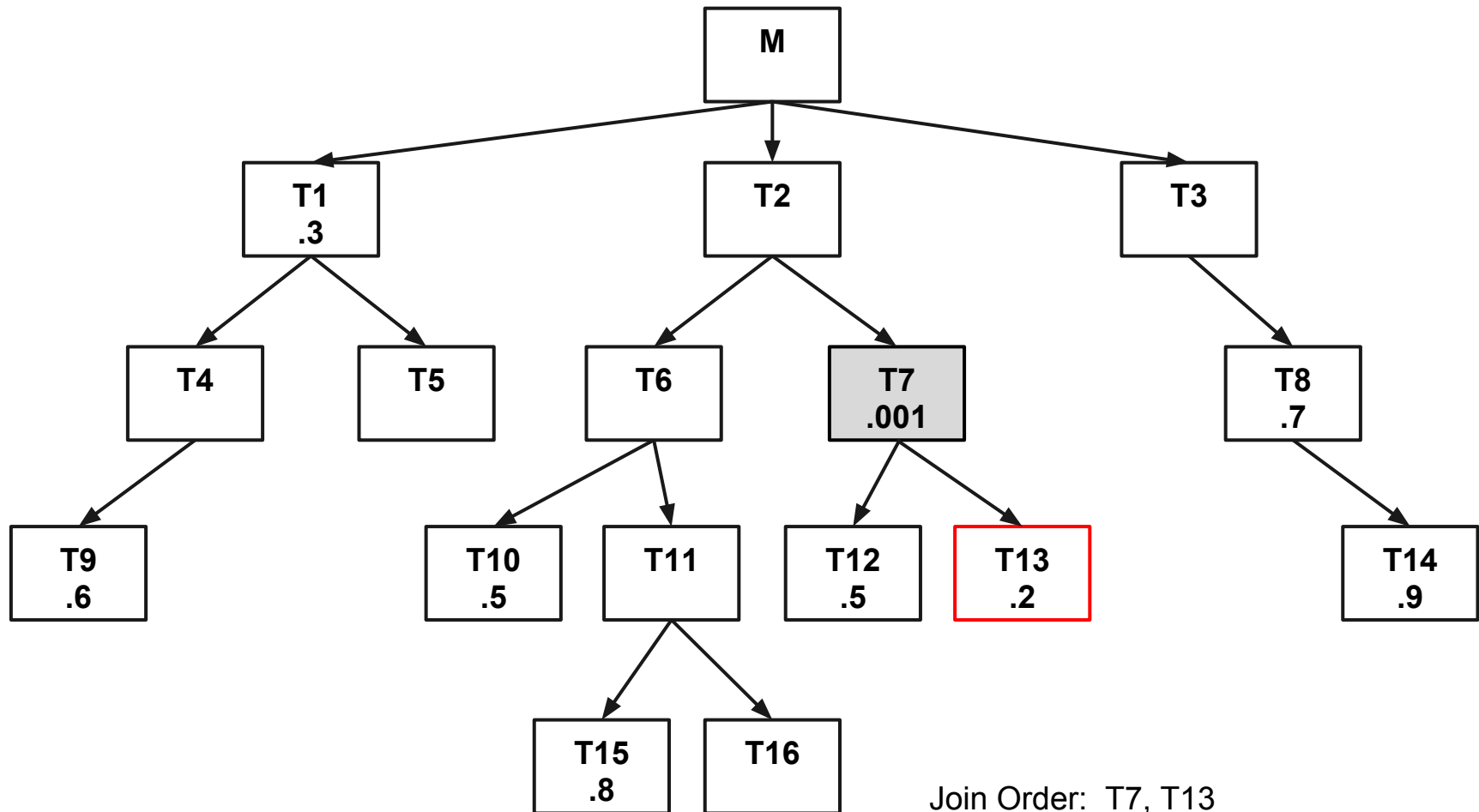
# Interpreting Diagrams



# Interpreting Diagrams

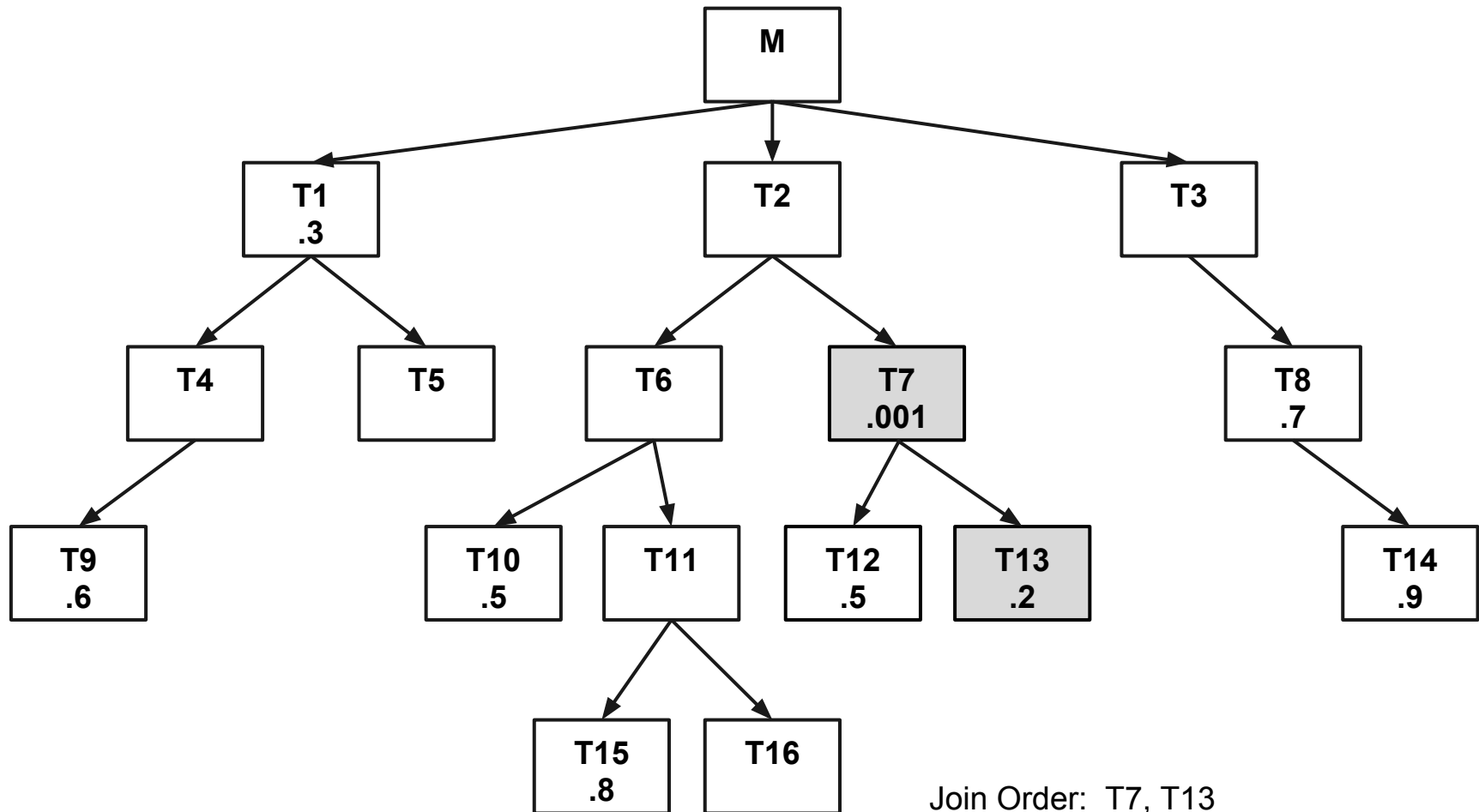


# Interpreting Diagrams

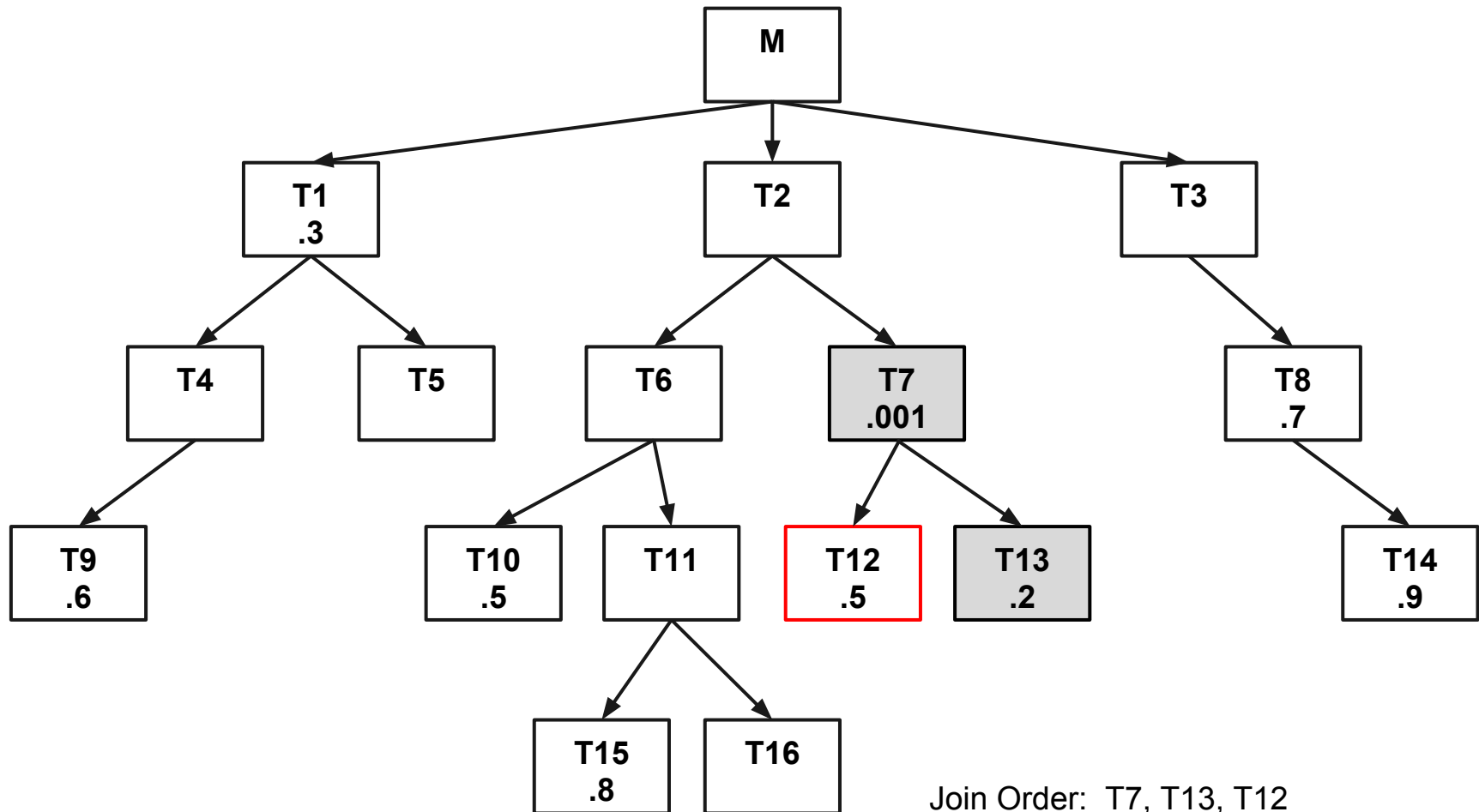




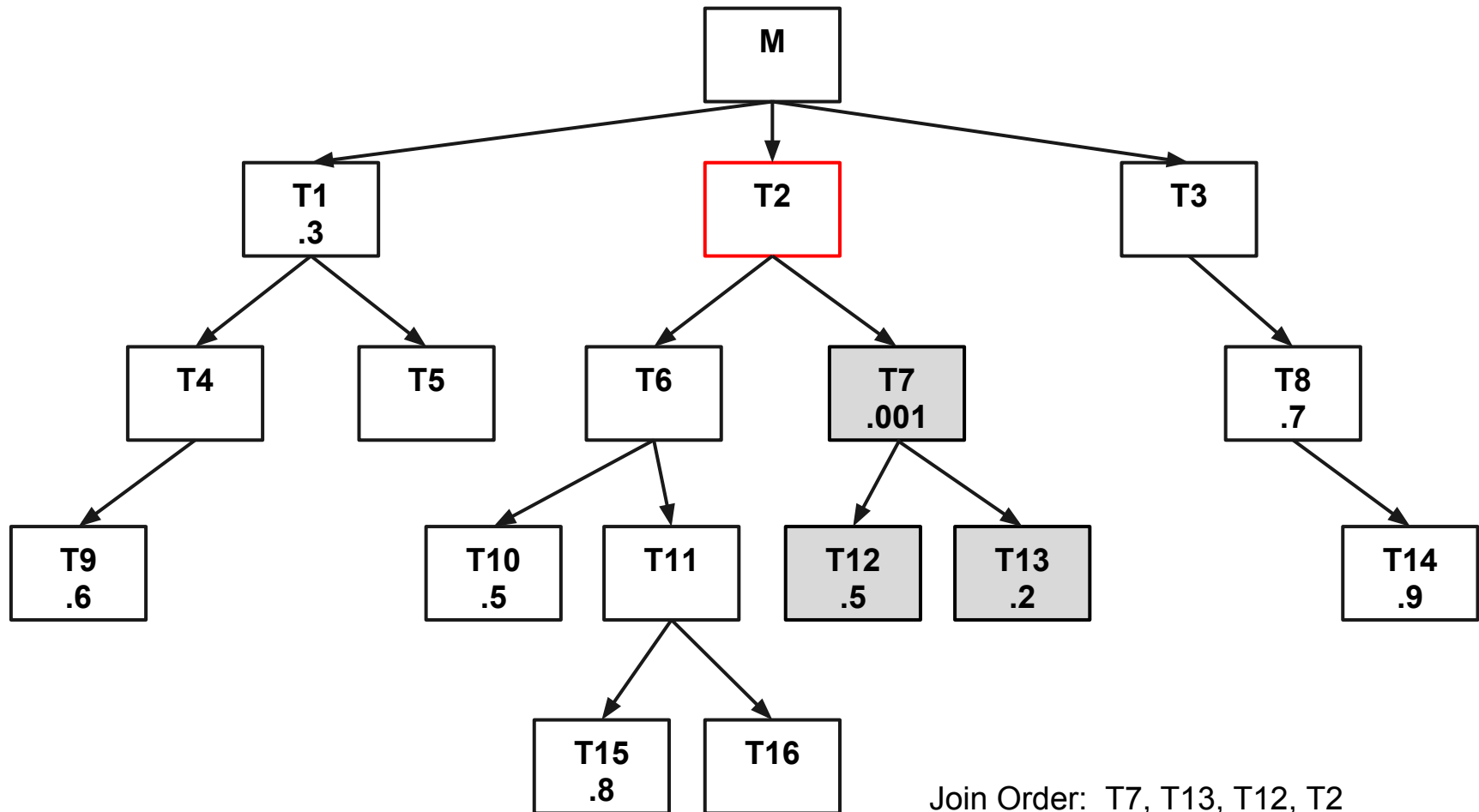
# Interpreting Diagrams



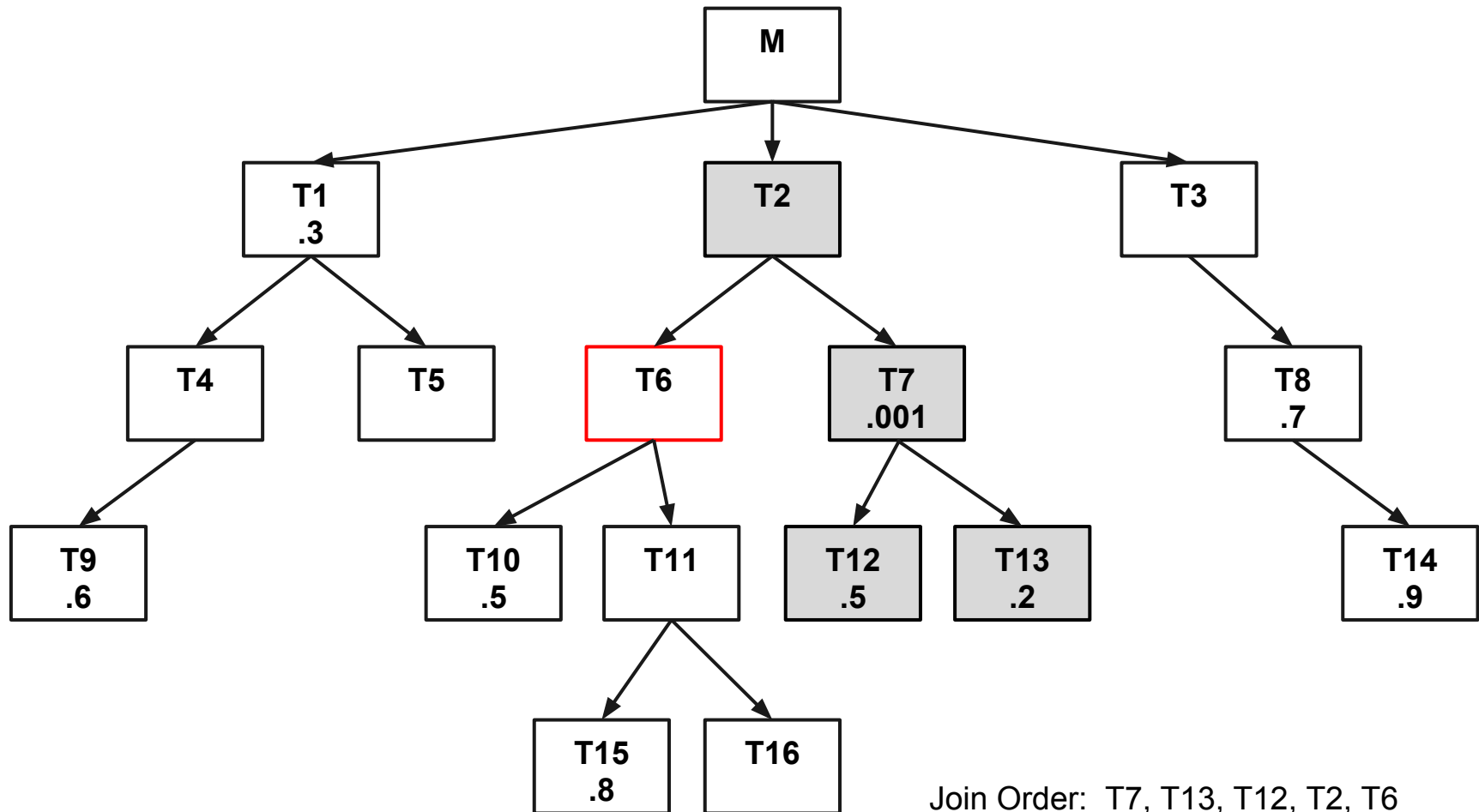
# Interpreting Diagrams



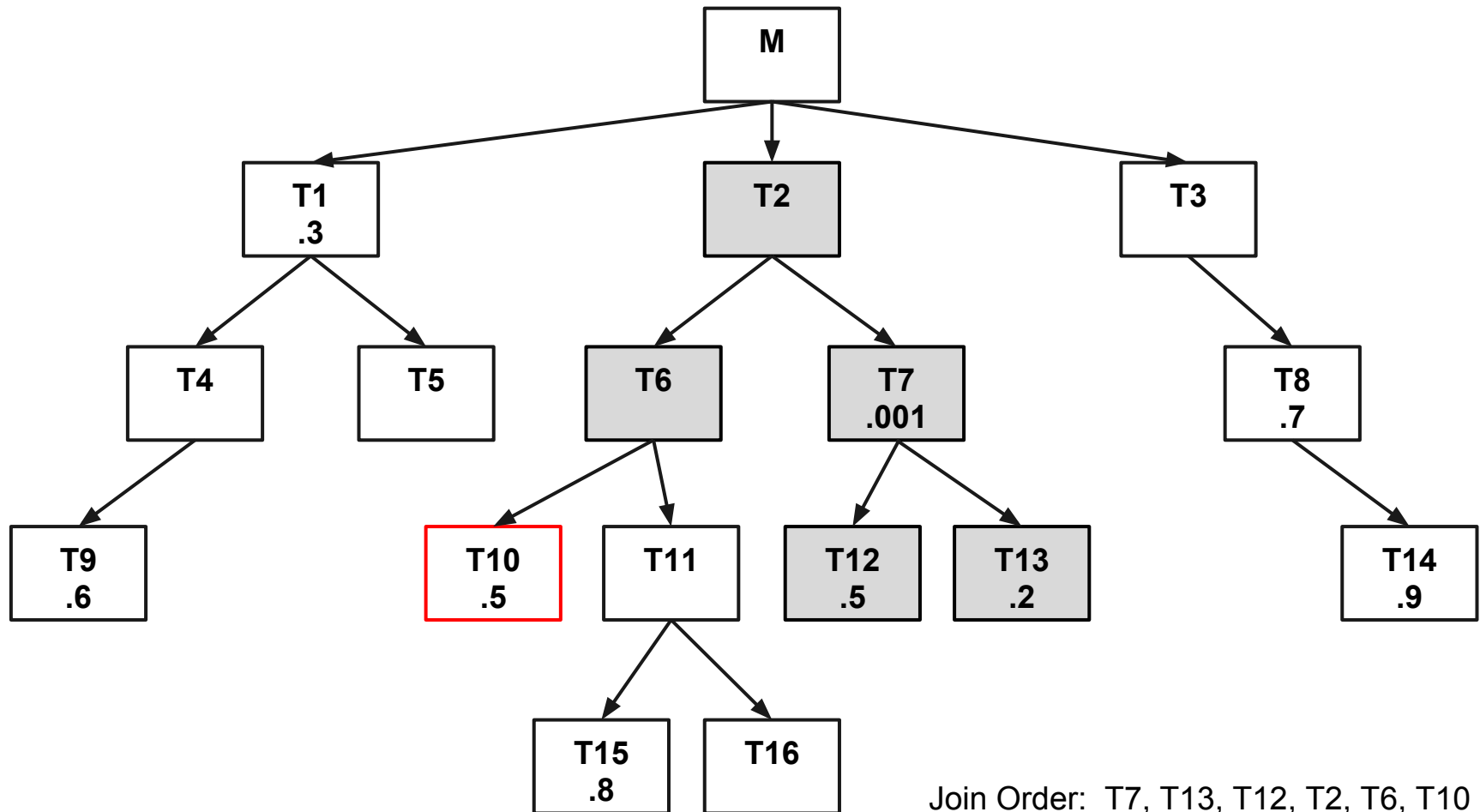
# Interpreting Diagrams



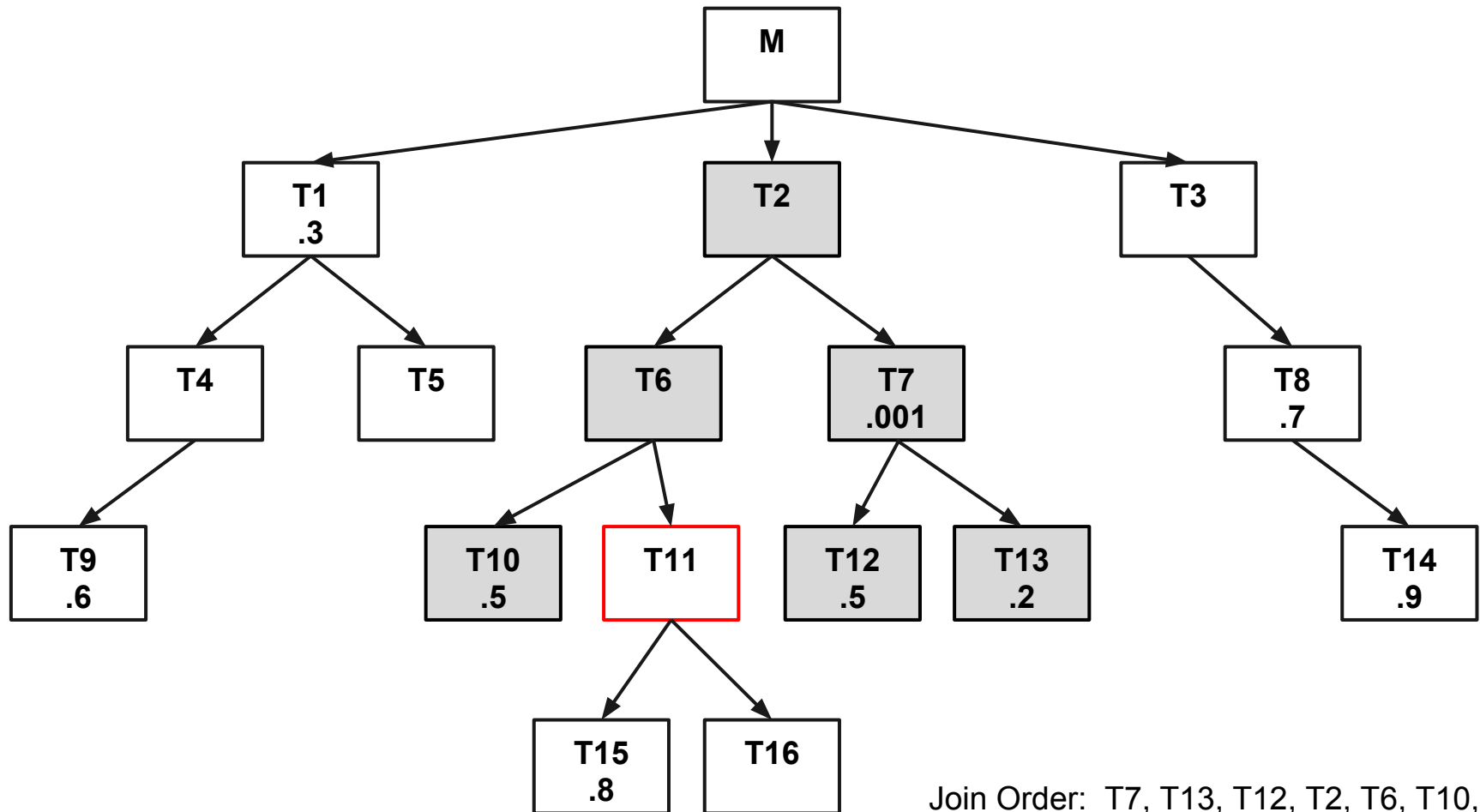
# Interpreting Diagrams



# Interpreting Diagrams

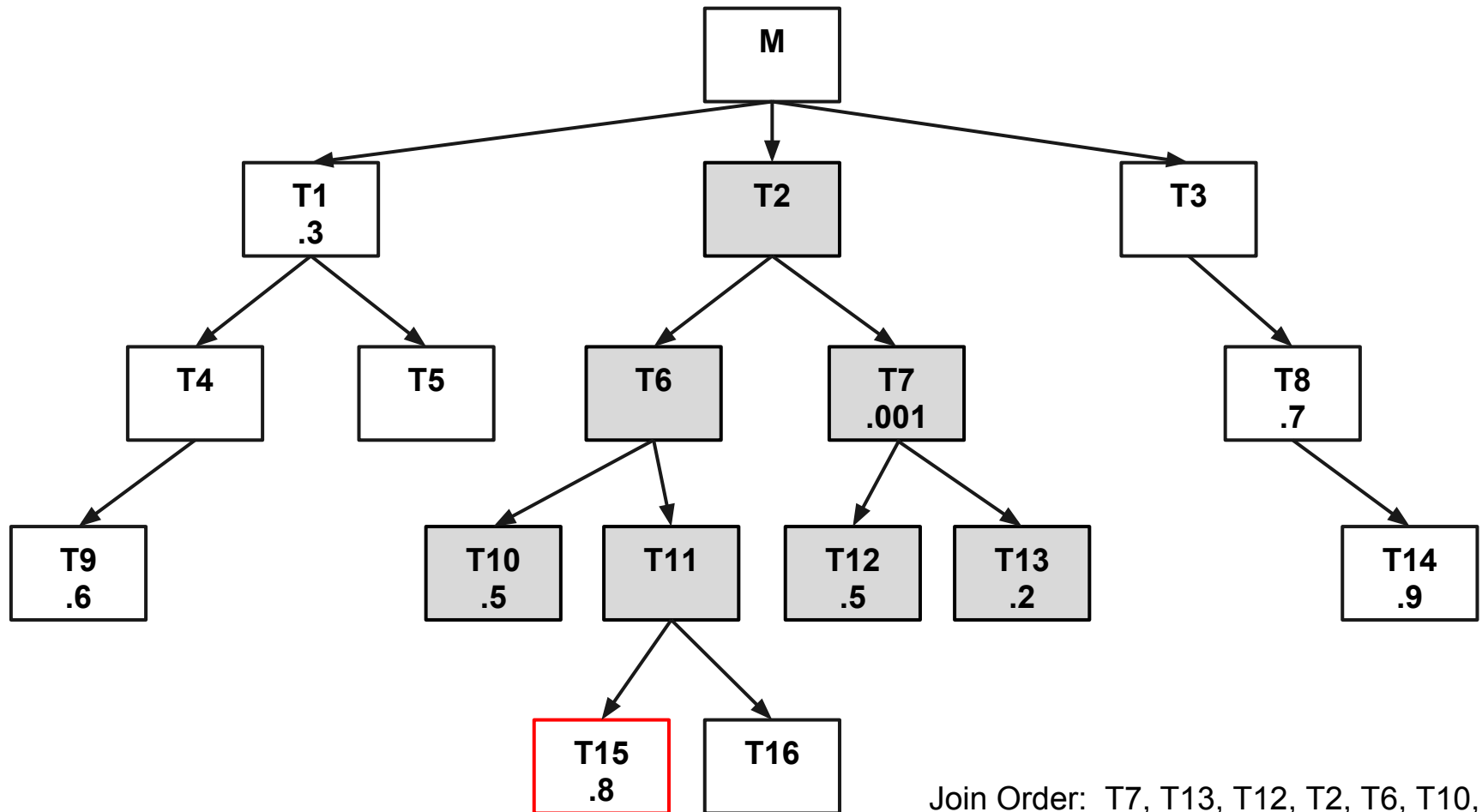


# Interpreting Diagrams



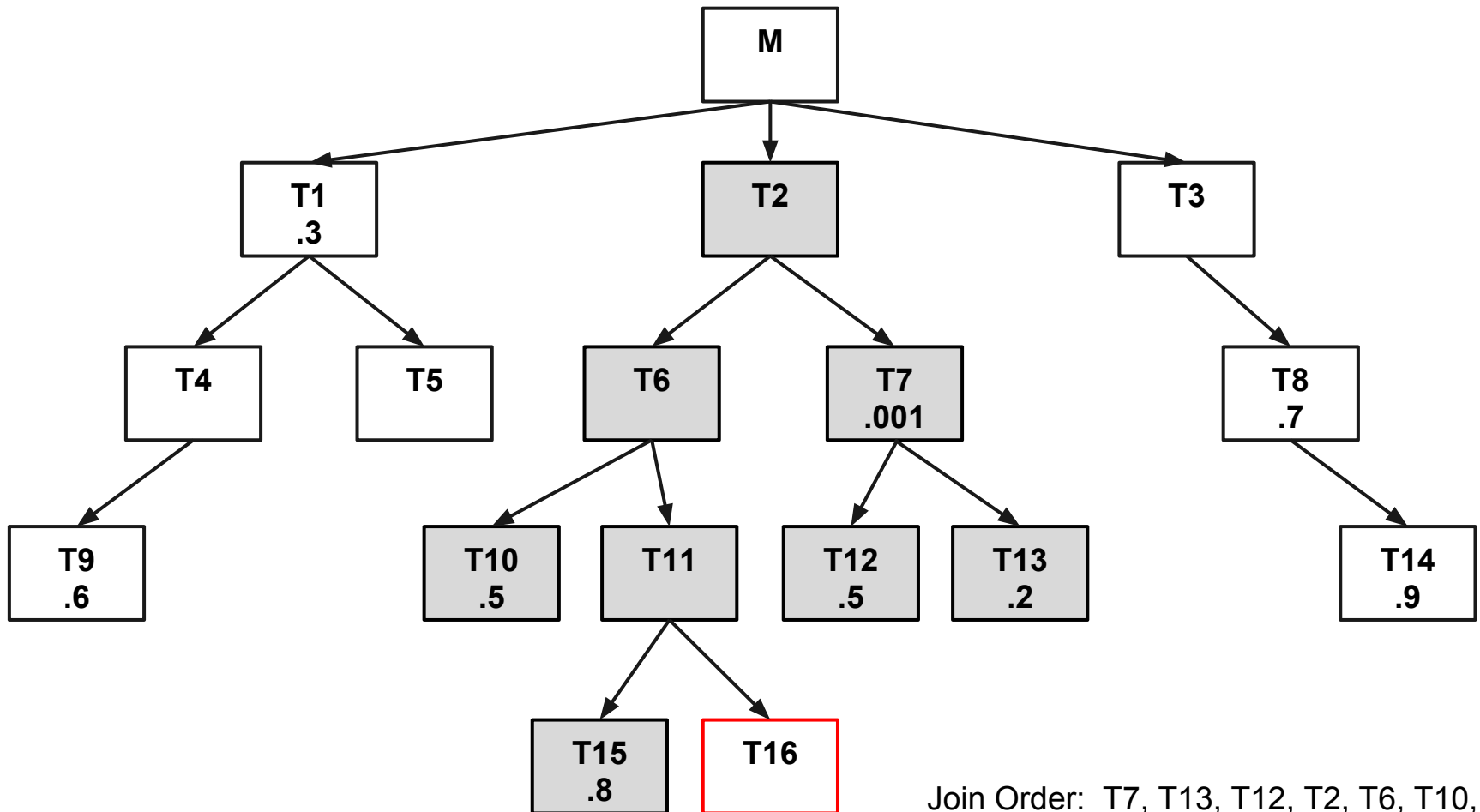
Join Order: T7, T13, T12, T2, T6, T10, T11

# Interpreting Diagrams



Join Order: T7, T13, T12, T2, T6, T10, T11, T15

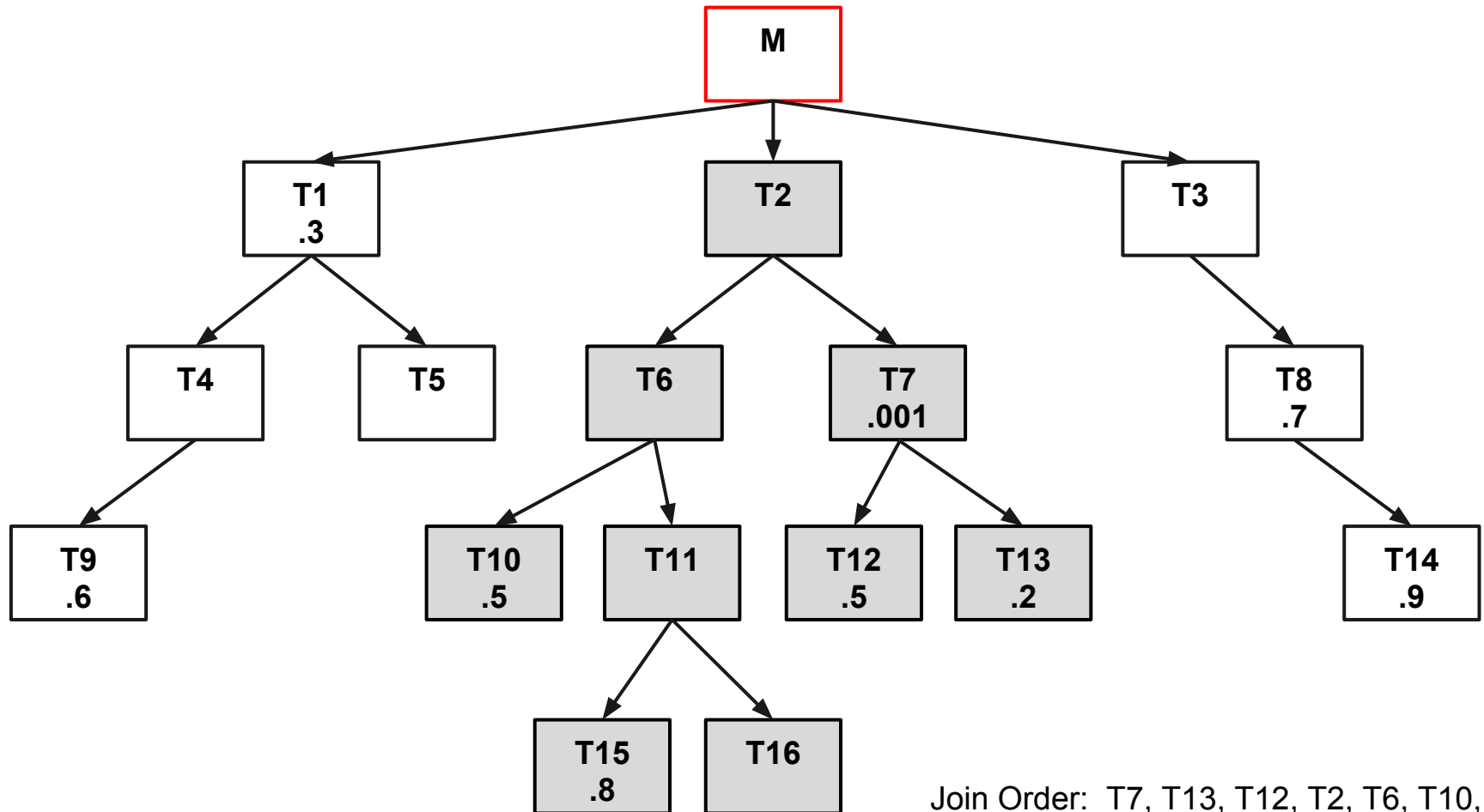
# Interpreting Diagrams



Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16

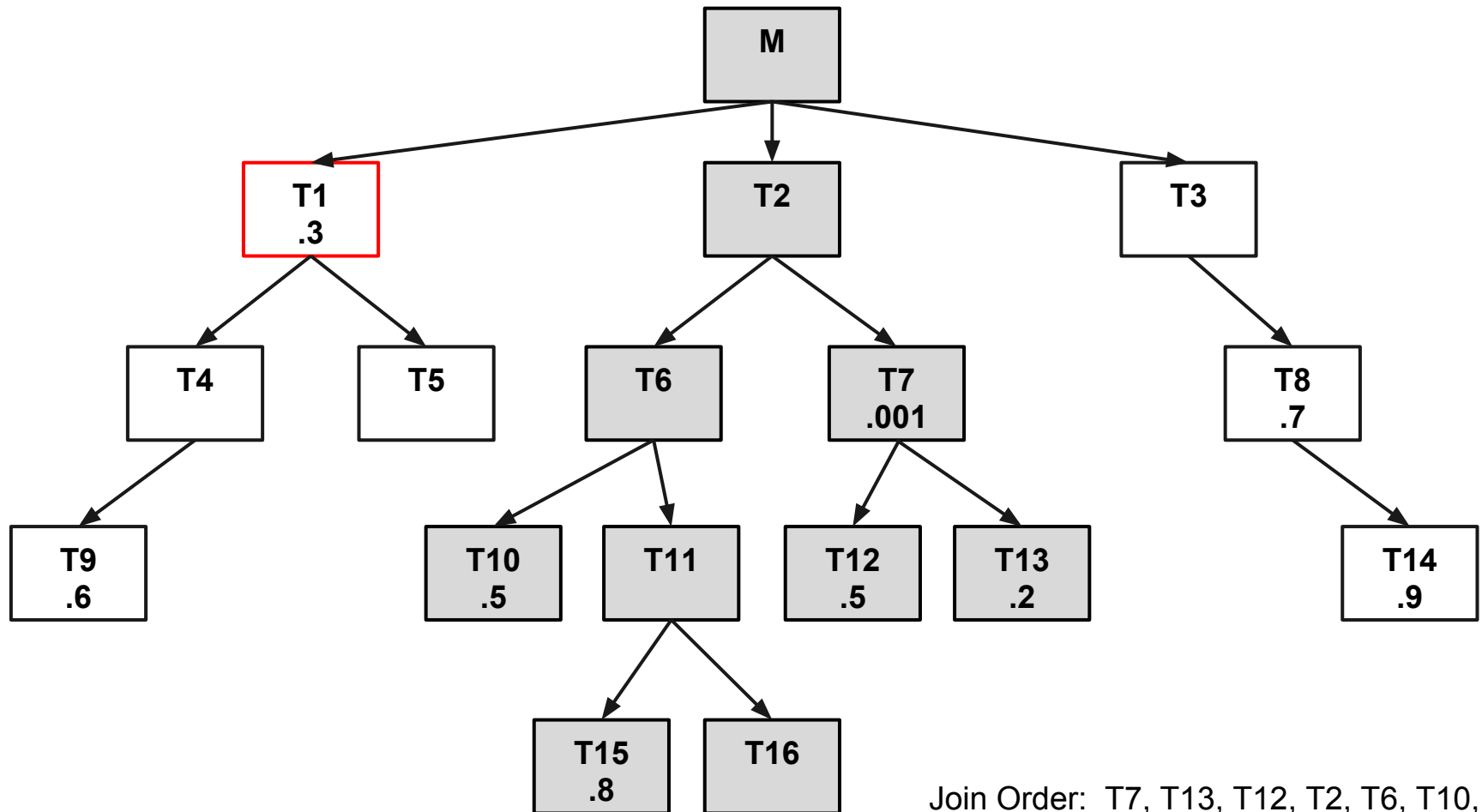


# Interpreting Diagrams



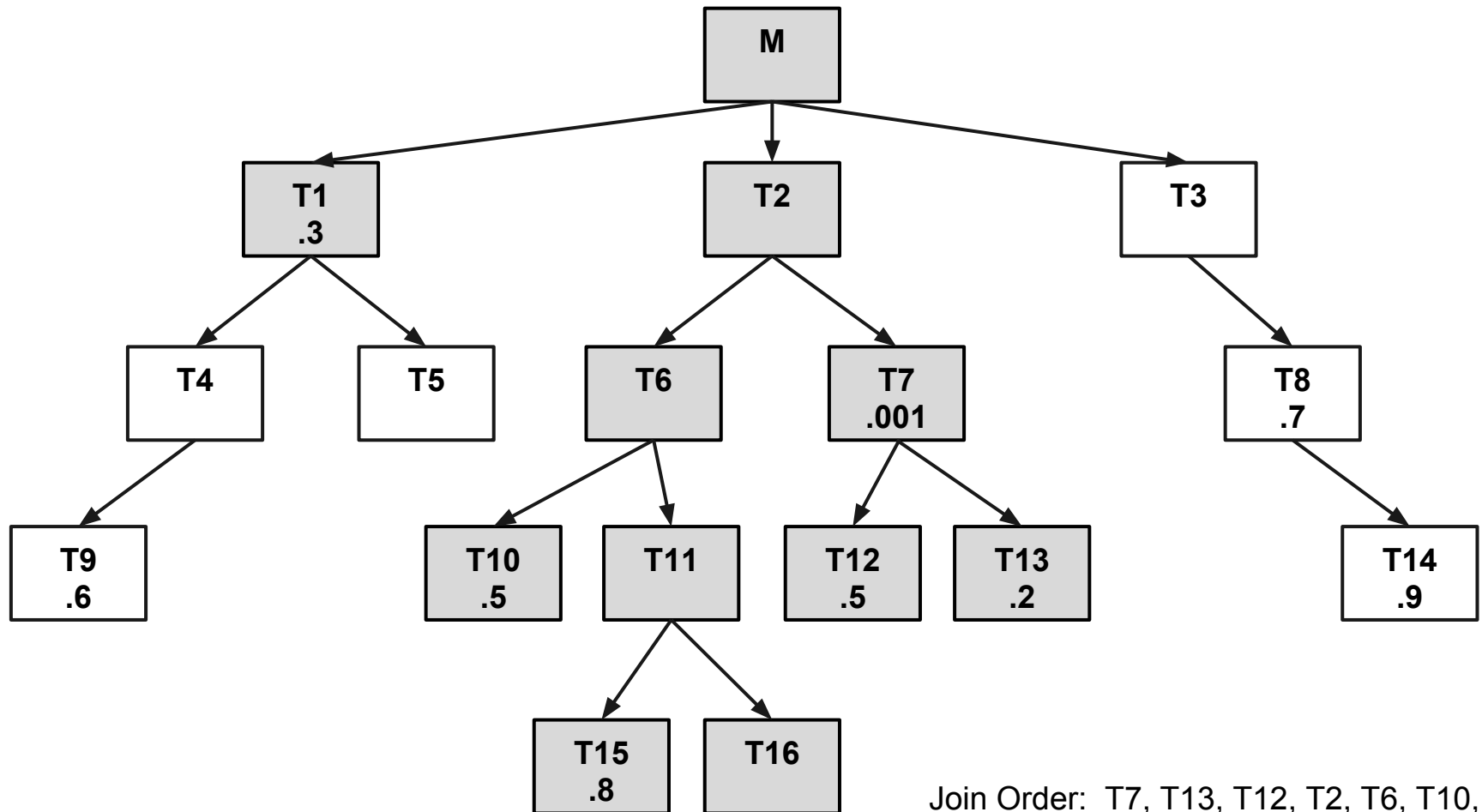
Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M

# Interpreting Diagrams



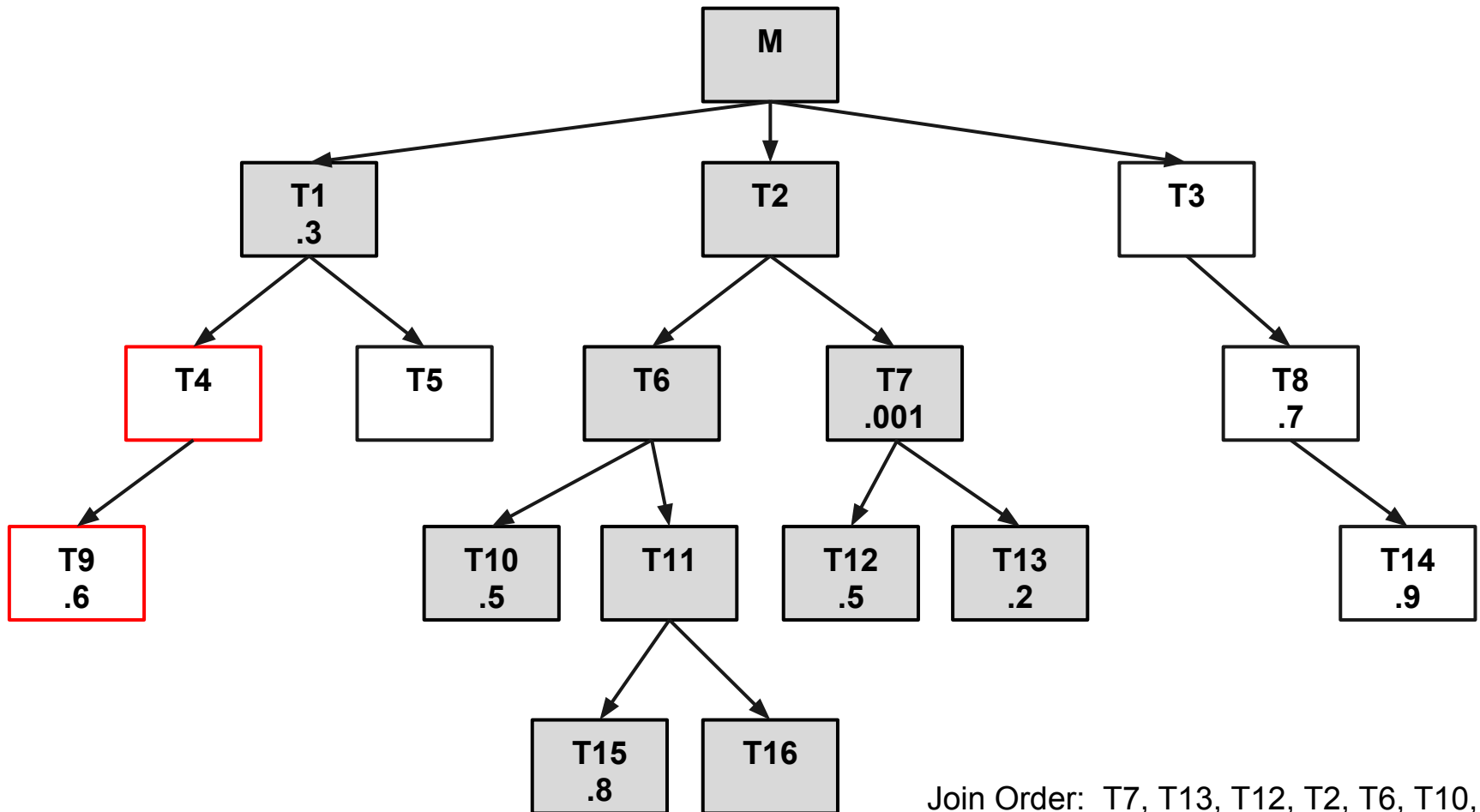
Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M, T1

# Interpreting Diagrams



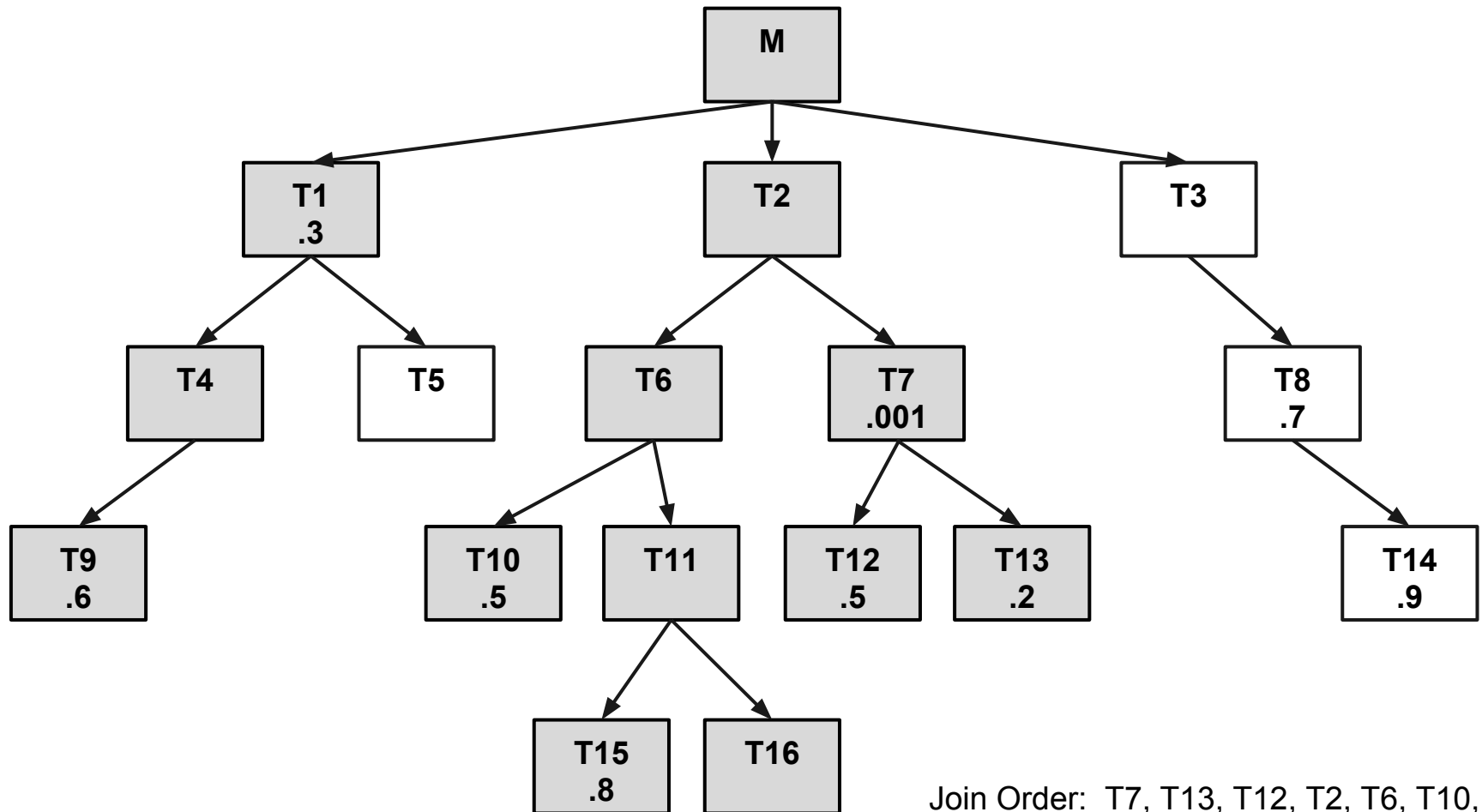
Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M, T1

# Interpreting Diagrams



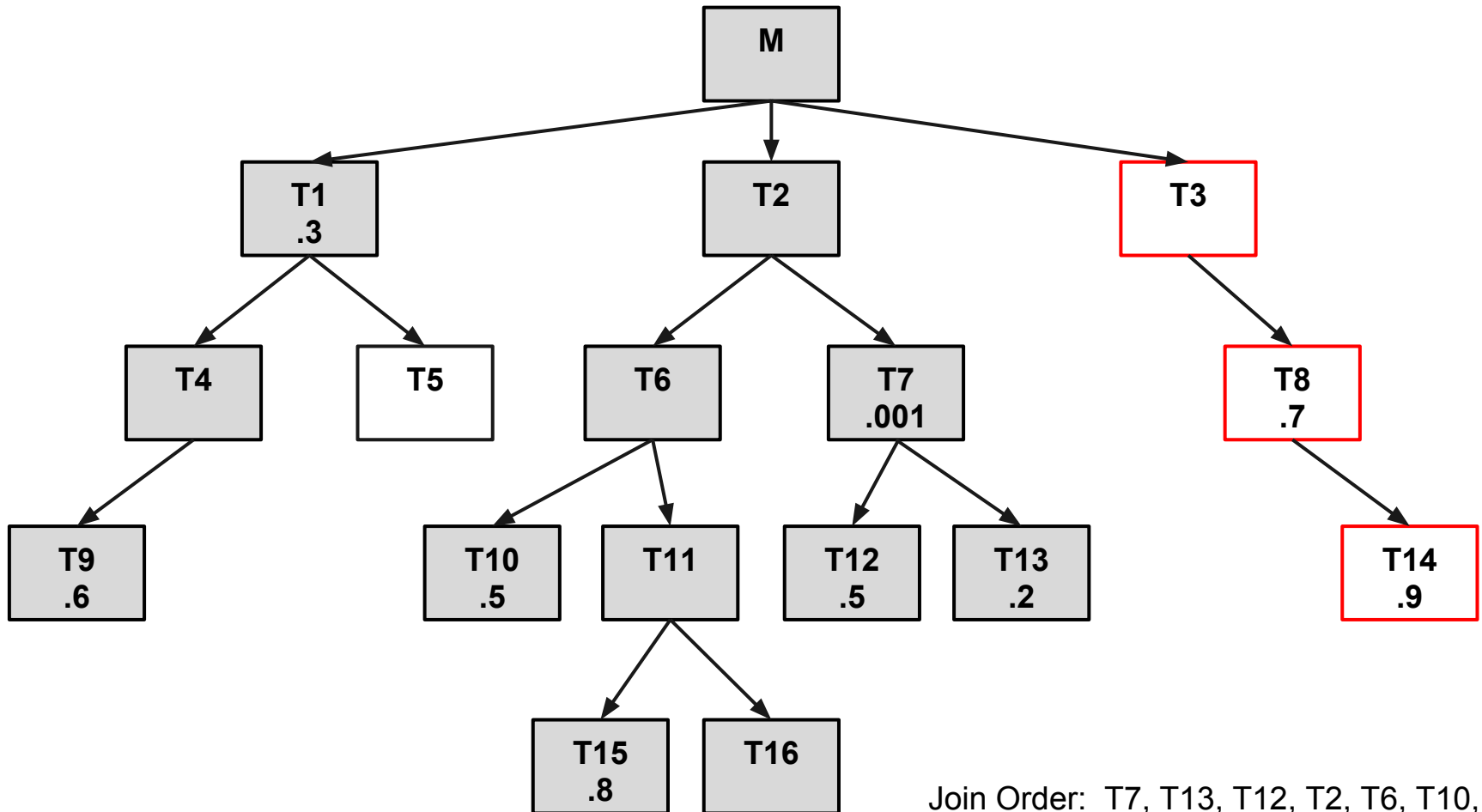
Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M, T1, T4, T9

# Interpreting Diagrams



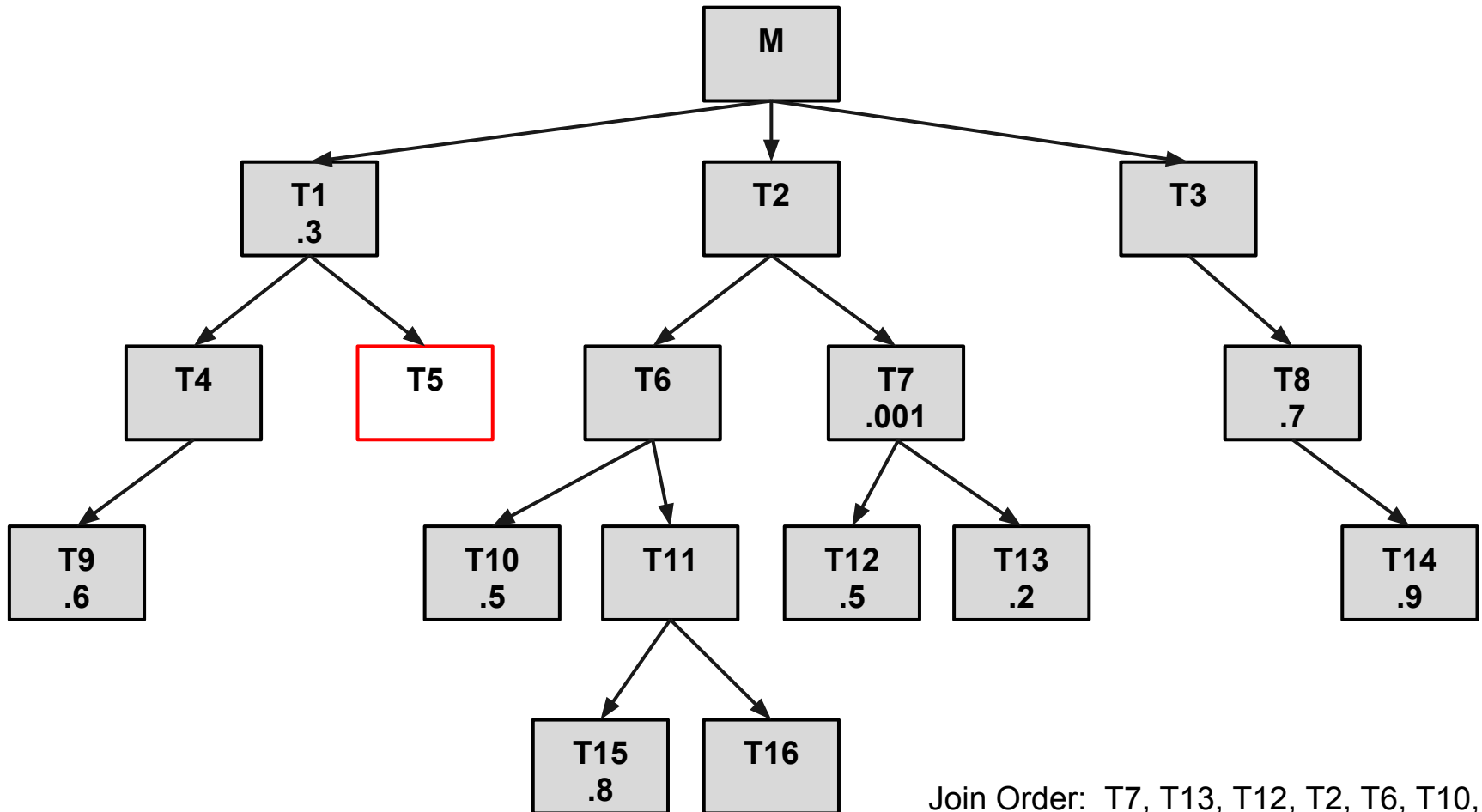
Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M, T1, T4, T9

# Interpreting Diagrams



Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M, T1, T4, T9, T3, T8, T14

# Interpreting Diagrams



Join Order: T7, T13, T12, T2, T6, T10, T11, T15, T16, M, T1, T4, T9, T3, T8, T14, T5

# Why Not Just Guess?

Are these diagrams really necessary?

Can't we just try different options until we get it right?

For a 17 table join, there are theoretically 355,687,428,096,000 (yes, trillion!) possible join paths!



# Still not fast enough?

After driving table and join order have been identified, next places to look are:

- Filter method
- Join method

# Filter Method

Do you need an index that doesn't exist?

Are you using an index where you shouldn't be?

# Index Usage

- Percentage of rows should not factor into whether you should use an index
- Based on data distribution within tables
- All about percentage of blocks that will need to be touched
- Applies to both:
  - Rows being accessed by filter
  - Rows being accessed by join

# Indexes are not free

- Slow down Inserts, Updates, Deletes
- Once on production, are EXTREMELY difficult to get rid of
- Make sure to ask:
  - Is this performance improvement worth the cost?
  - What else will be helped?
  - What else will be harmed?
- Test thoroughly!

# Join Method

## Nested Loops vs. Hash Join

- Nested Loops can use data from driving row source when looking up data in child row source
  - Hash Joins can not
- Hash Joins will only read the child row source once, putting results into hash table in memory
  - Nested Loops will read once for every row in driving row source
  - Watch out for memory issues!

# Joins - Further Reading

## Jonathan Lewis:

- <http://jonathanlewis.wordpress.com/2010/08/09/joins-nlj/>
- <http://jonathanlewis.wordpress.com/2010/08/10/joins-hj/>
- <http://jonathanlewis.wordpress.com/2010/08/15/joins-mj/>

## Tanel Poder:

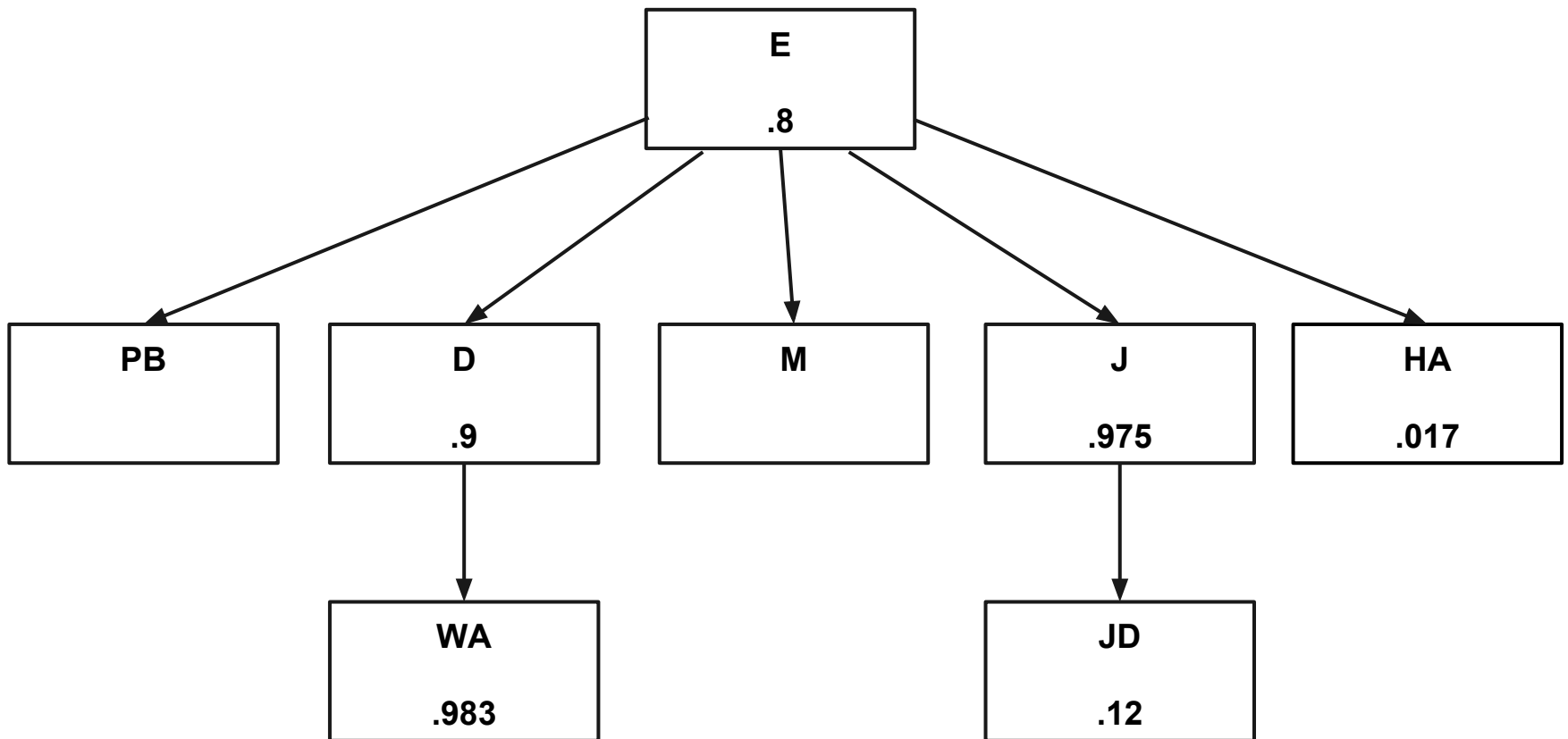
- <http://tech.e2sn.com/oracle/sql/the-fundamental-difference-between-nested-loops-and-hash-joins>

# Interpreting Diagrams

## Somewhat Common Exceptions:

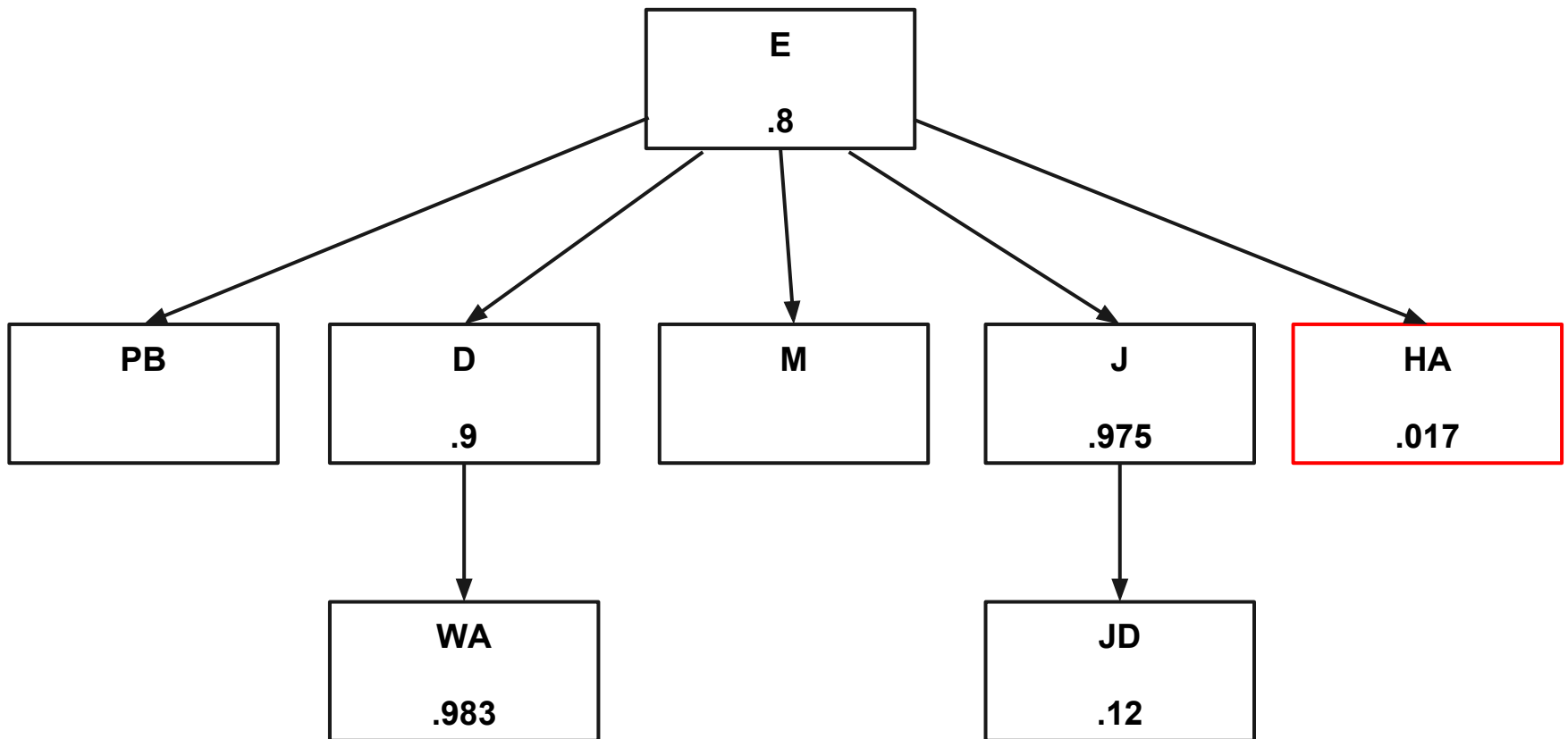
- **Filter to Exactly 1 Row**
  - These can be done out of order, cross joining the results
- **Detail Join Ratios Near 1**
  - Don't have to follow down-first guideline. Go in whatever direction to get to next best filter ratio
- **Filter Ratios Almost Equal**
  - Drive to a slightly higher filter ratio first if it will lead to a much lower filter ratio

# Interpreting Diagrams



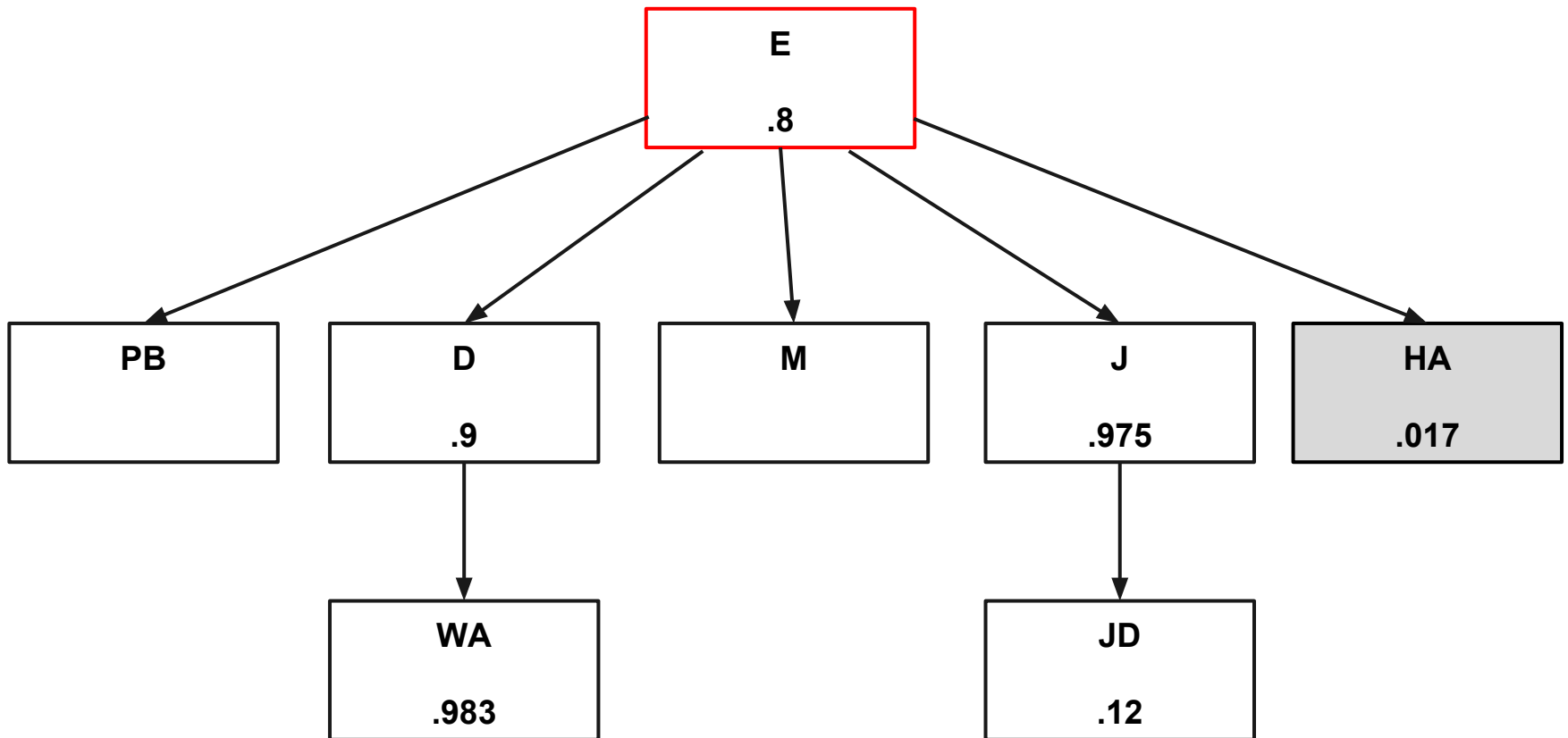


# Interpreting Diagrams



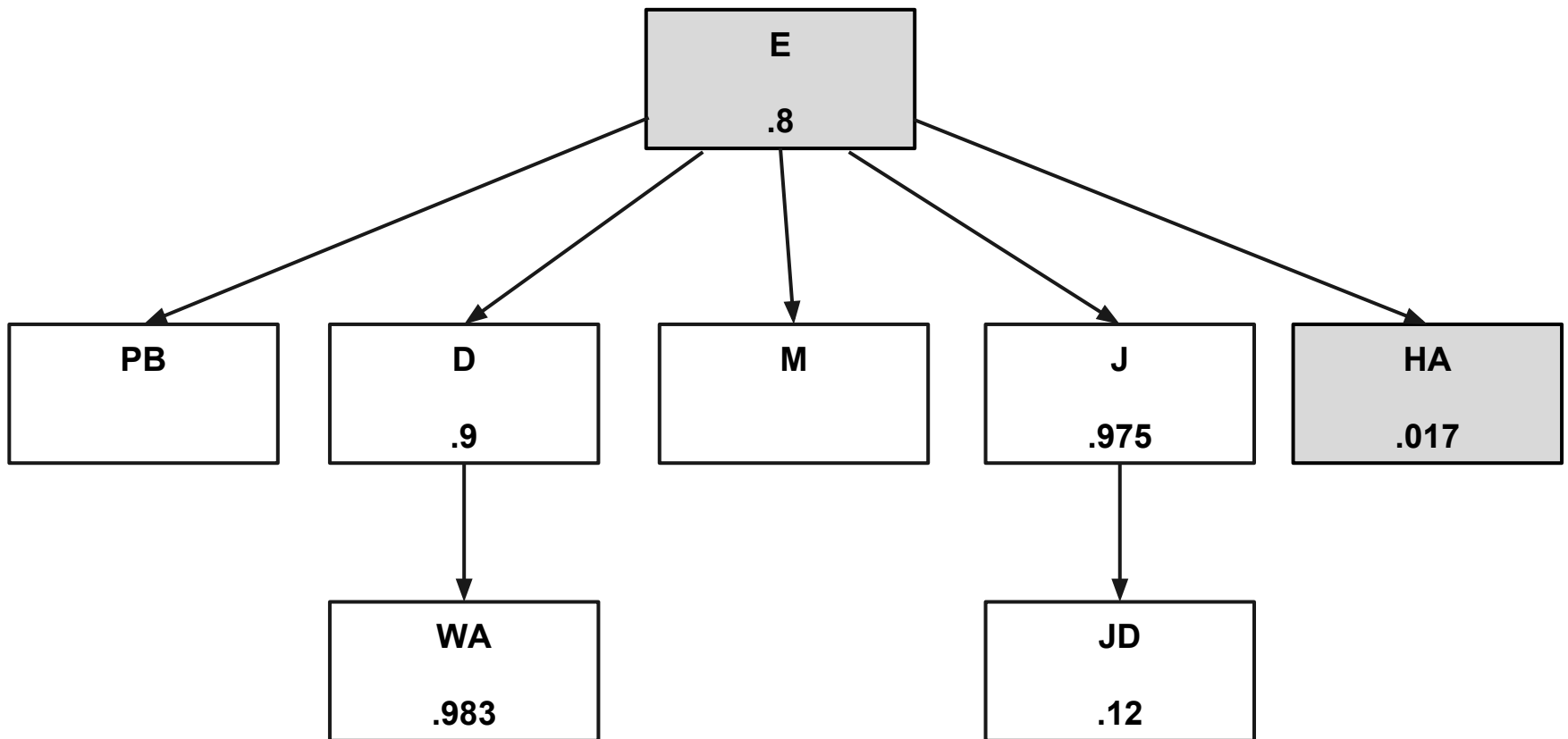
Join Order: HA

# Interpreting Diagrams



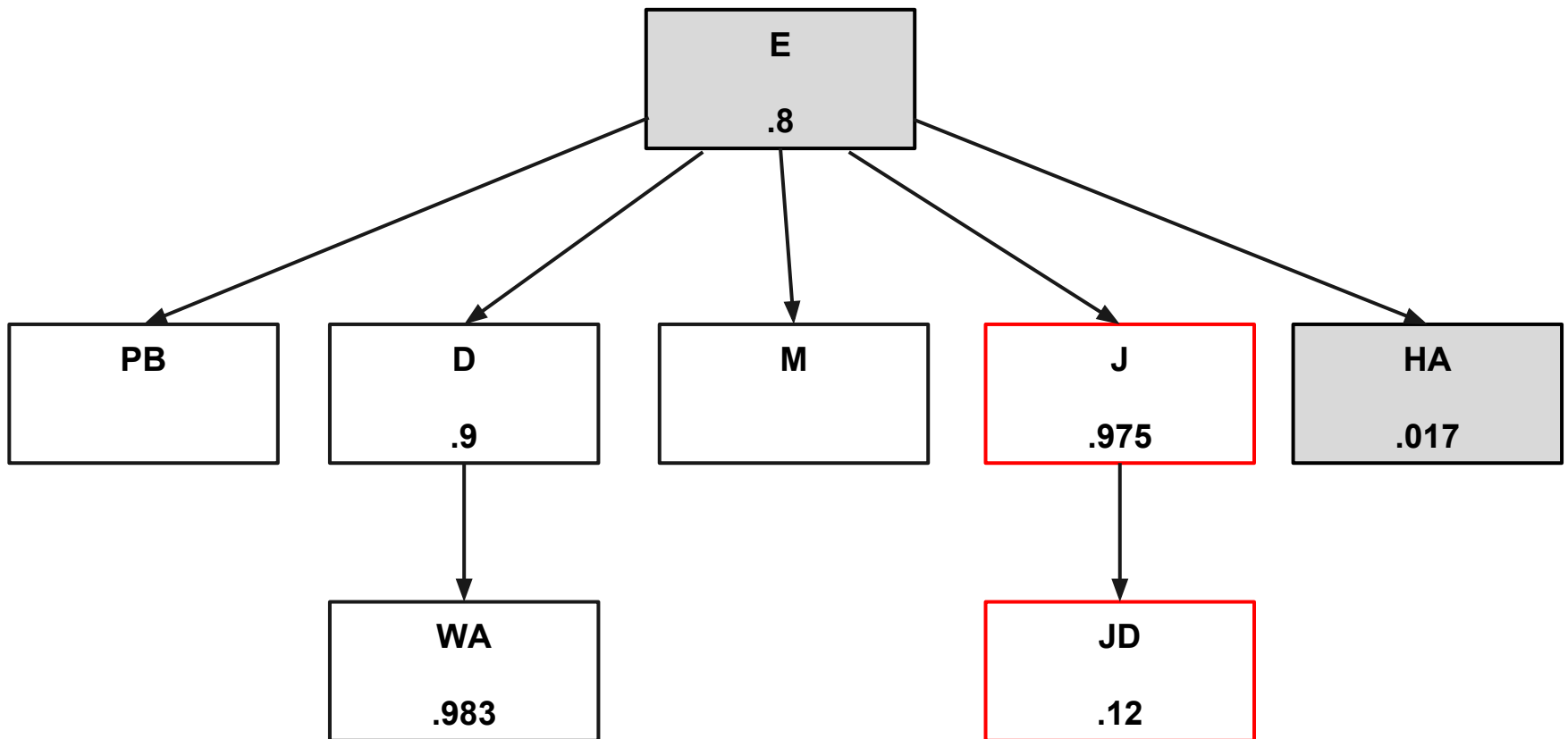
Join Order: HA, E

# Interpreting Diagrams



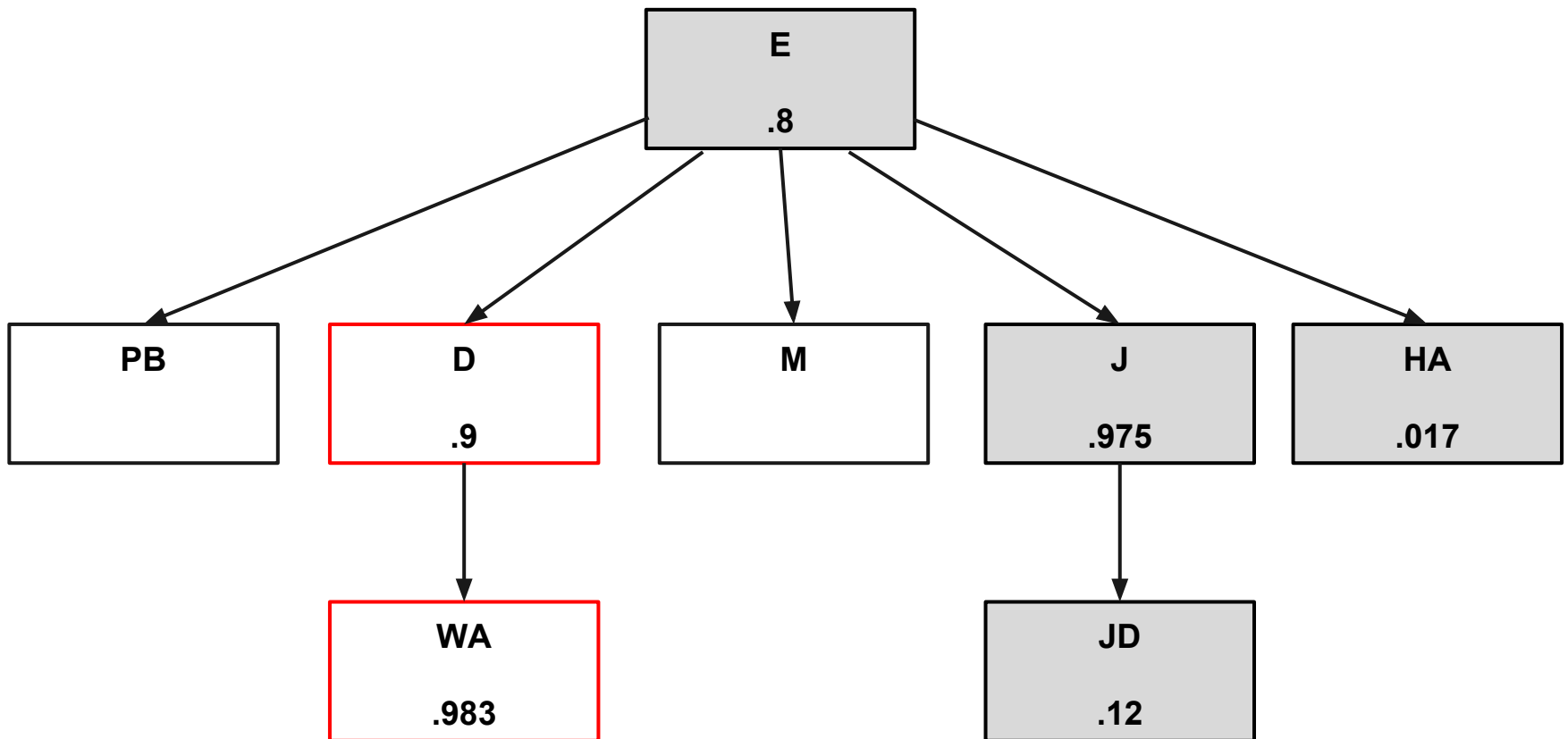
Join Order: HA, E

# Interpreting Diagrams



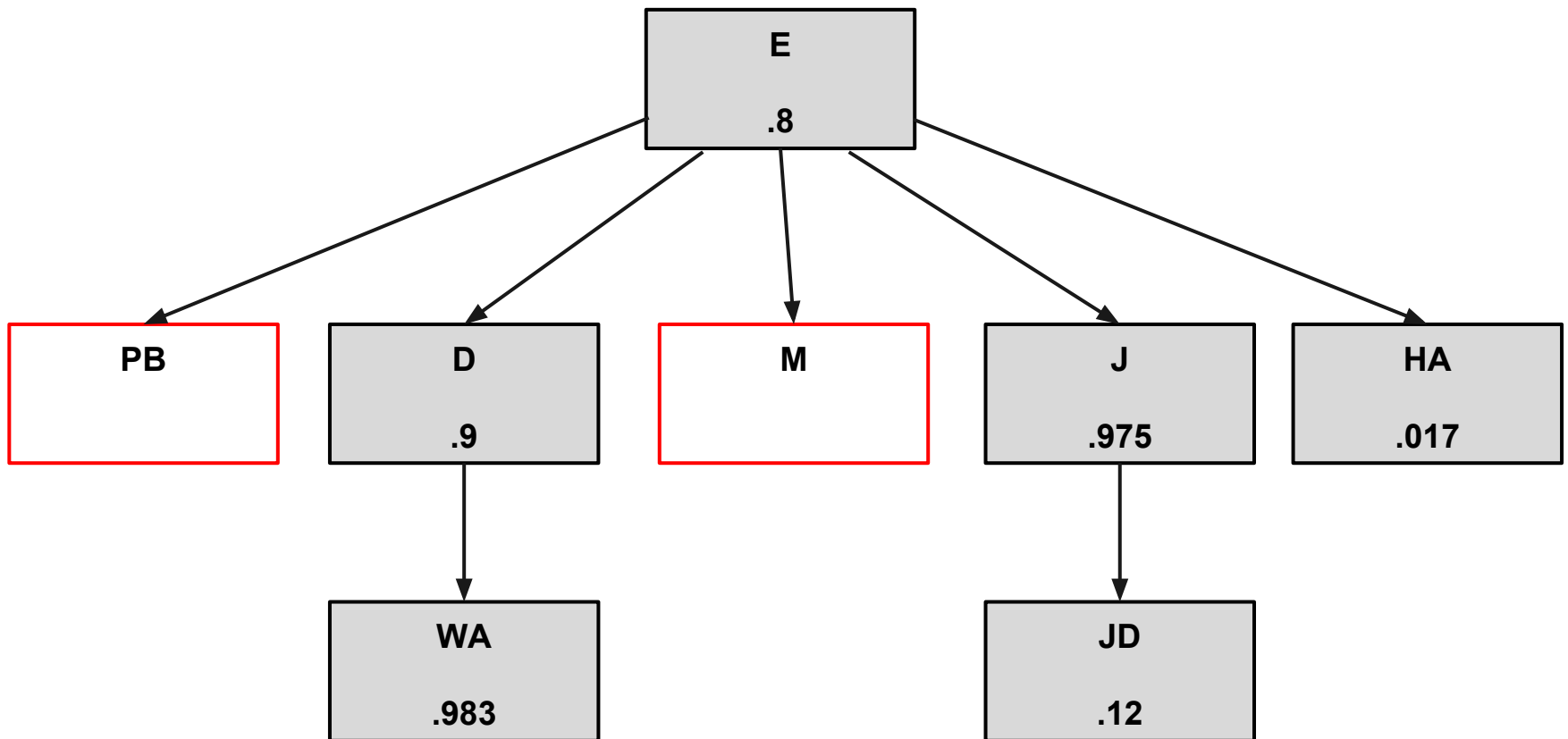
Join Order: HA, E, J, JD

# Interpreting Diagrams



Join Order: HA, E, J, JD, D, WA

# Interpreting Diagrams



Join Order: HA, E, J, JD, D, WA, (PB, M)

# Now What?

- Update statistics to get Oracle to pick correct path
  - Extended Statistics
- Update query to allow index usage
- SQL Plan Management
- SQL Profiles
- Hints

# Additional Information

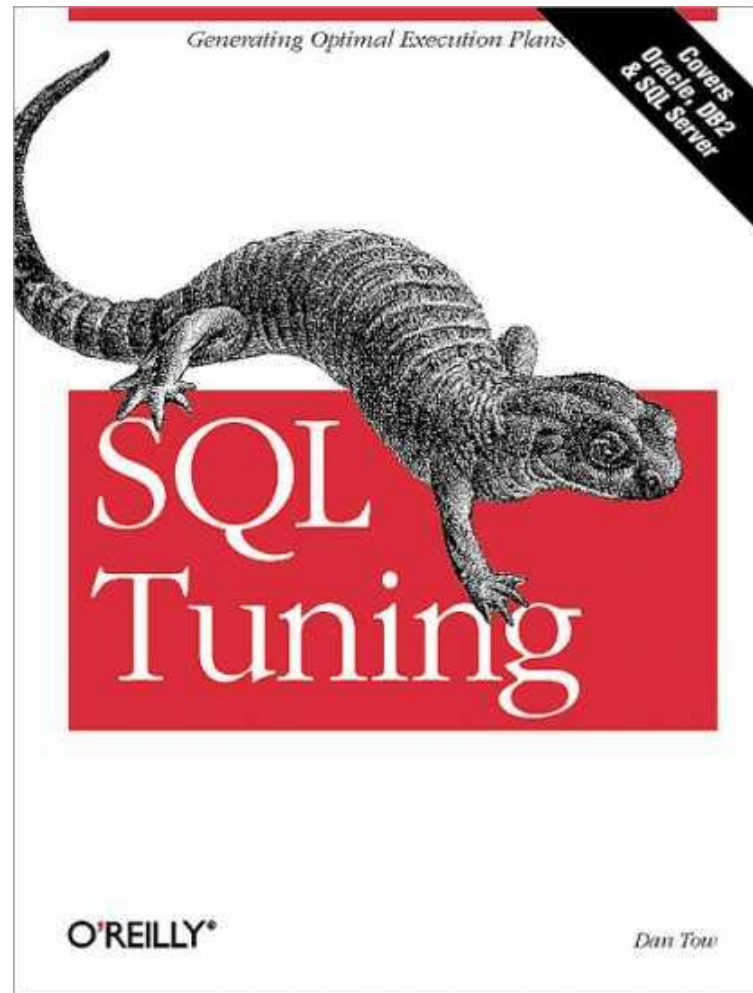
The rules covered today will cover 90% of queries

Not Covered:

- Outer Joins
- IN / EXISTS
- VIEWS
- Special Exceptions



# Additional Information



# Questions?

<http://www.cmartin2.com>

[cmartin2@cmartin2.com](mailto:cmartin2@cmartin2.com)

@c\_martin2