Equivalent logical expressions; different costs

$\sigma_{p=knows}(R) \bowtie_{o=s} (\sigma_{p=holdsAccount}(R) \bowtie_{o=s} \sigma_{p=accountHomepage}(R))$

right associative

$(\sigma_{p=knows}(R) \bowtie_{o=s} \sigma_{p=holdsAccount}(R)) \bowtie_{o=s} \sigma_{p=accountHomepage}(R)$

left associative

$\sigma_{p1=knows\ \&\ p2=holdsAccount\ \&\ p3=accountHomepage}(R \times R \times R)$

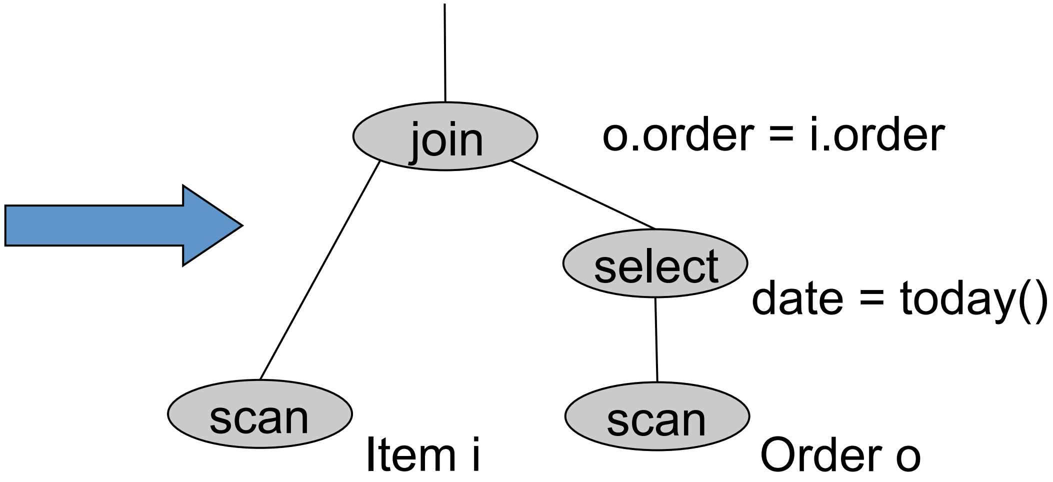
cross product

Key Idea: Declarative Languages

Order(order, date, account)
Item(order, part)

Find all orders from today, along with the items ordered

```
SELECT *
  FROM Order o, Item i
 WHERE o.order = i.order
   AND o.date = today()
```



SQL is the “WHAT” not the “HOW”

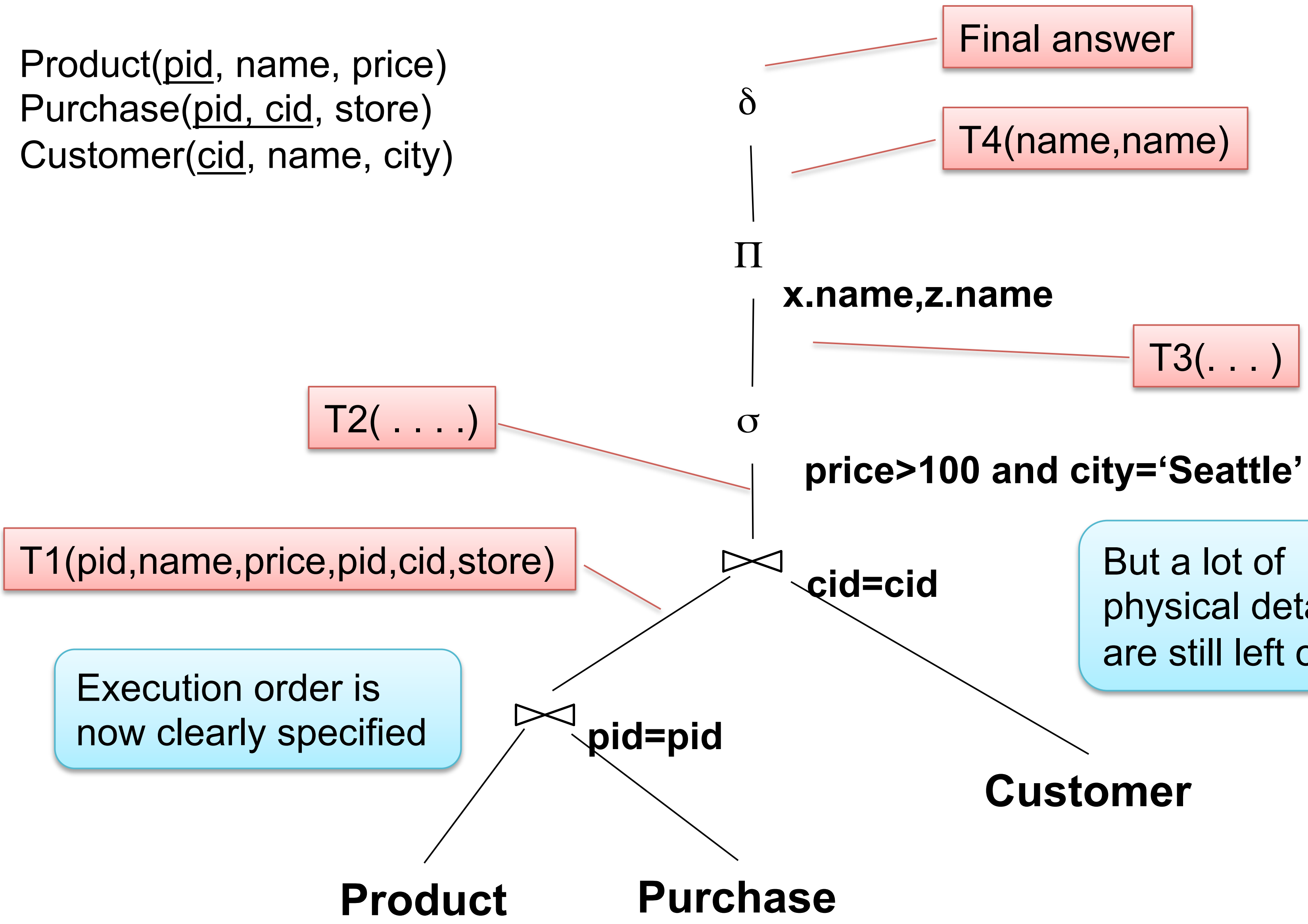
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = y.cid and
               x.price > 100 and z.city = ‘Seattle’
```

It’s clear WHAT we want, unclear HOW to get it

Relational Algebra

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)



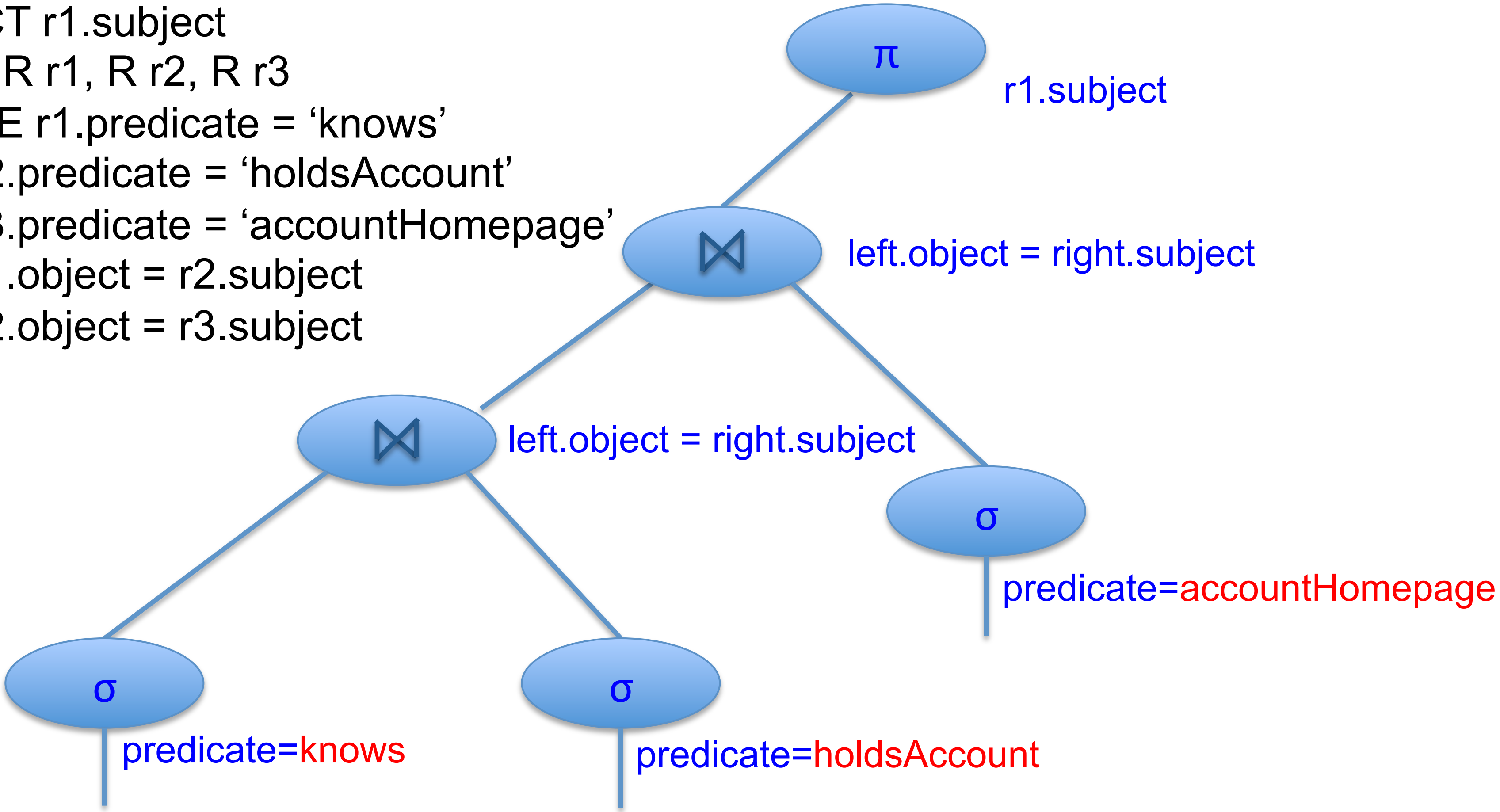
Execution order is now clearly specified

But a lot of physical details are still left open!

Another Example

R(subject, predicate, object)

```
SELECT r1.subject
FROM R r1, R r2, R r3
WHERE r1.predicate = ‘knows’
AND r2.predicate = ‘holdsAccount’
AND r3.predicate = ‘accountHomepage’
AND r1.object = r2.subject
AND r2.object = r3.subject
```


π
r1.subject
⋈
left.object = right.subject
⋈
left.object = right.subject
σ
predicate=accountHomepage
σ
predicate=knows
σ
predicate=holdsAccount