# Database Systems
# CSE 414

Lectures 9: Relational Algebra (part 2) and Query Evaluation (Ch. 5.2 & 16.3 (skim 16.3.2))



# Announcements

- Should have used SQL / Azure now
  - let us know if you had any setup problems
- WQ3 is due on Sunday
- HW3 is due one week from Tuesday
- HW1 grades *should* be posted tonight



# Join Summary

- **Theta-join**: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join of R and S with a join condition $\theta$
  - Cross-product followed by selection $\theta$
- **Equijoin**: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join condition $\theta$ consists only of equalities
- **Natural join**: $R \bowtie S = \pi_A (\sigma_\theta(R \times S))$
  - Equijoin
  - Equality on **all** fields with same name in R and in S
  - Projection $\pi_A$ drops all redundant attributes



# So Which Join Is It ?

When we write $R \bowtie S$ we usually mean an equijoin, but we often omit the equality predicate when it is clear from the context



# More Joins

- **Outer join**
  - Include tuples with no matches in the output
  - Use NULL values for missing attributes
  - Does not eliminate duplicate columns
- Variants
  - Left outer join
  - Right outer join
  - Full outer join



# Outer Join Example

AnonPatient P

| age | zip | disease |
|---|---|---|
| 54 | 98125 | heart |
| 20 | 98120 | flu |
| 33 | 98120 | lung |

AnonJob J

| job | age | zip |
|---|---|---|
| lawyer | 54 | 98125 |
| cashier | 20 | 98120 |

$P ⟕ J$

| P.age | P.zip | disease | job | J.age | J.zip |
|---|---|---|---|---|---|
| 54 | 98125 | heart | lawyer | 54 | 98125 |
| 20 | 98120 | flu | cashier | 20 | 98120 |
| 33 | 98120 | lung | null | null | null |

# More Examples

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)
```

Name of supplier of parts with size greater than 10

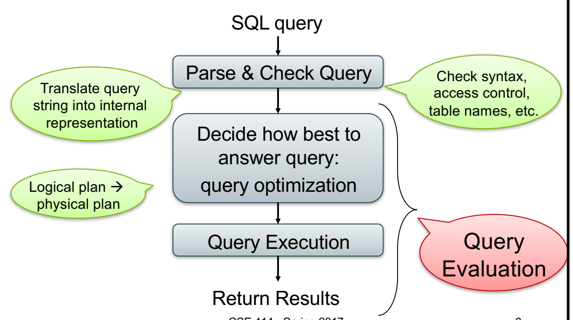
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}\ (\text{Part}))$

Name of supplier of red parts or parts with size greater than 10

$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10}\ (\text{Part}) \cup \sigma_{pcolor='red'}\ (\text{Part})\ )\ )$



# Query Evaluation Steps

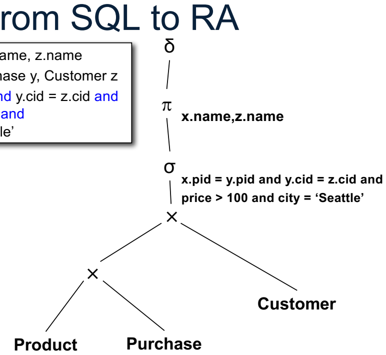





Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

# From SQL to RA

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

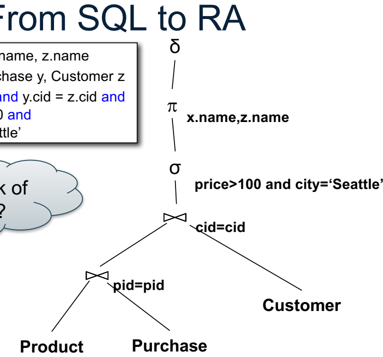



Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

# From SQL to RA

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

Can you think of another plan?

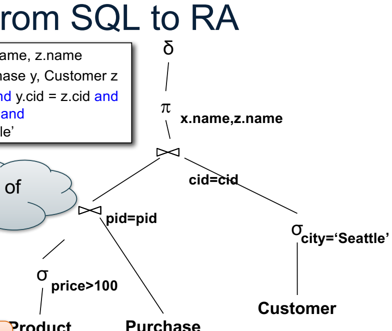



Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

# From SQL to RA

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

Can you think of another plan?




Push selections down the query plan!

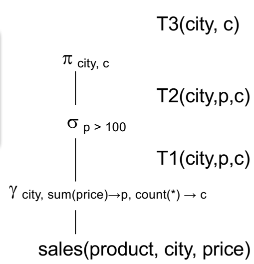
Query optimization: find an equivalent optimal plan

# Extended RA: Operators on Bags

- Duplicate elimination $\delta$
- Grouping & aggregation $\gamma$
- Sorting $\tau$

## Logical Query Plan

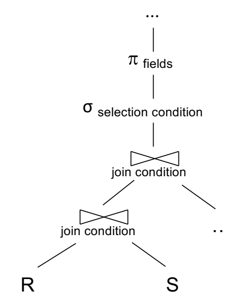
```
SELECT city, count(*)
FROM sales
GROUP BY city
HAVING sum(price) > 100
```




T1, T2, T3 = temporary tables



## Typical Plan for Block (1/2)

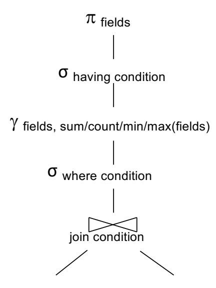



SELECT fields
FROM R, S, …
WHERE condition

SELECT-PROJECT-JOIN Query



## Typical Plan for Block (2/2)




SELECT fields
FROM R, S, …
WHERE condition
GROUP BY fields
HAVING condition



Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)

## How about Subqueries?

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
   and not exists
   (SELECT *
    FROM Supply P
    WHERE P.sno = Q.sno
       and P.price > 100)
```

Correlation !



Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)

## How about Subqueries?

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
   and not exists
   (SELECT *
    FROM Supply P
    WHERE P.sno = Q.sno
       and P.price > 100)
```

De-Correlation

```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
   and Q.sno not in
   (SELECT P.sno
    FROM Supply P
    WHERE P.price > 100)
```



Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)

## How about Subqueries?

Un-nesting

```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = 'WA')
  EXCEPT
(SELECT P.sno
  FROM Supply P
  WHERE P.price > 100)
```

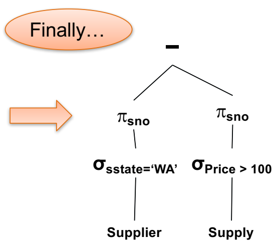
```
SELECT  Q.sno
FROM Supplier Q
WHERE  Q.sstate = 'WA'
   and Q.sno not in
   (SELECT P.sno
    FROM Supply P
    WHERE P.price > 100)
```

EXCEPT = set difference

Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,price)

## How about Subqueries?

```
(SELECT  Q.sno
 FROM Supplier Q
 WHERE  Q.sstate = 'WA')
  EXCEPT
(SELECT P.sno
  FROM Supply P
  WHERE P.price > 100)
```






## From Logical Plans to Physical Plans



## Physical Operators

Each of the logical operators may have one or more implementations = physical operators

Will discuss several basic physical operators, with a focus on join



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

## Main Memory Algorithms

Logical operator:

Product(pid, name, price) $\bowtie_{pid=pid}$ Purchase(pid, cid, store)

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join O( ?? )
2. Merge join O( ?? )
3. Hash join O( ?? )

(note that pid is a key)



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

## Main Memory Algorithms

Logical operator:

Product(pid, name, price) $\bowtie_{pid=pid}$ Purchase(pid, cid, store)

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join $O(n^2)$ two nested loops
2. Merge join O( ?? )
3. Hash join O( ?? )



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

## Main Memory Algorithms

Logical operator:

Product(pid, name, price) $\bowtie_{pid=pid}$ Purchase(pid, cid, store)

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join $O(n^2)$
2. Merge join O(n log n)
3. Hash join O( ?? )

sort both – O(n log n)
merge – O(n)

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

## Main Memory Algorithms

Logical operator:

Product(pid, name, price) $\bowtie_{pid=pid}$ Purchase(pid, cid, store)

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join — $O(n^2)$
2. Merge join — $O(n \log n)$
3. Hash join — $O(n) \dots O(n^2)$

add n to hash – O(n)?
lookup n in hash – O(n)?



## BRIEF Review of Hash Tables

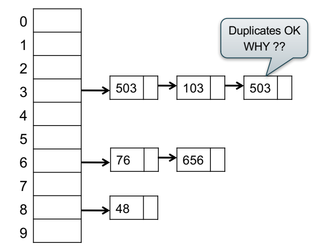
A (naïve) hash function:

h(x) = x mod 10

Operations:

find(103) = ??
insert(488) = ??

Separate chaining:

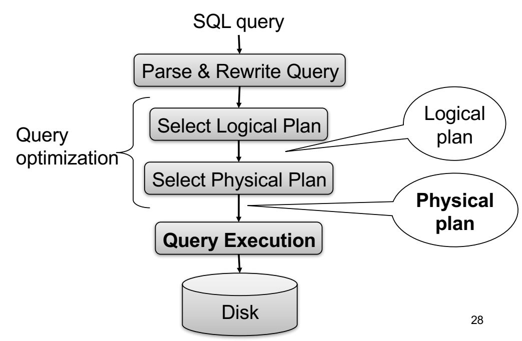



## BRIEF Review of Hash Tables

- insert(k, v) = inserts a key k with value v
- Many values for one key
  - Hence, duplicate k's are OK
- find(k) = returns the ***list*** of all values v associated to the key k



## Query Evaluation Steps Review






Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

## Relational Algebra

```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
   and  y.pno = 2
   and x.scity = 'Seattle'
   and x.sstate = 'WA'
```

Give a relational algebra expression for this query



Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

## Relational Algebra

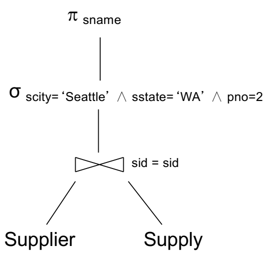
```
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
   and  y.pno = 2
   and x.scity = 'Seattle'
   and x.sstate = 'WA'
```

$\pi_{sname}(\sigma_{scity='Seattle' \wedge sstate='WA' \wedge pno=2}(Supplier \bowtie_{sid=sid} Supply))$

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

## Relational Algebra

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and y.pno = 2
and x.scity = 'Seattle'
and x.sstate = 'WA'

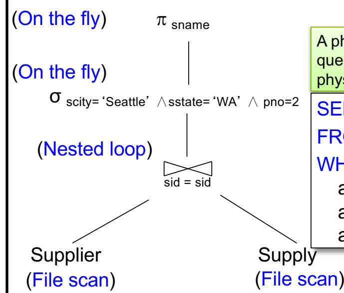



Relational algebra expression is also called the "logical query plan"



Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

## Physical Query Plan 1



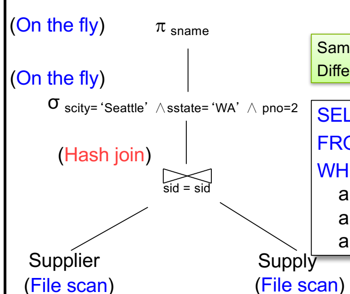

A physical query plan is a logical query plan annotated with physical implementation details

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and y.pno = 2
and x.scity = 'Seattle'
and x.sstate = 'WA'



Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

## Physical Query Plan 2




Same logical query plan
Different physical plan

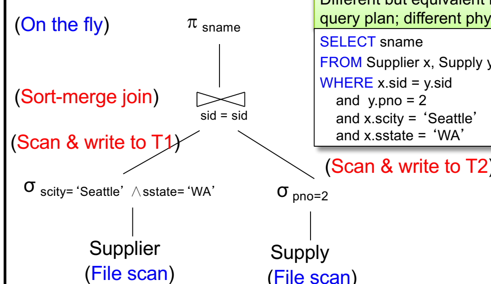
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and y.pno = 2
and x.scity = 'Seattle'
and x.sstate = 'WA'



Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

## Physical Query Plan 3

Different but equivalent logical query plan; different physical plan




SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and y.pno = 2
and x.scity = 'Seattle'
and x.sstate = 'WA'



## Query Optimization Problem

- For each SQL query… many logical plans
- For each logical plan… many physical plans
- How do find a fast physical plan?
  - Will discuss in a few lectures