SQL Subqueries
Background: execution model

Iterator model:
- open()
- next()
- close()

\[
\begin{align*}
\text{sort} & \quad \bowtie_1 \quad \text{hash} \\
\sigma(R) & \quad \bowtie_2 \quad \text{inl} \\
\sigma(S) & \quad \text{T}
\end{align*}
\]
Background: aggregation

```sql
select max(age)
from students
```
Background: aggregation

```
select max(age)
from students
```

\[
\Gamma_{\text{max(age)}}
\]

\[
\text{students}
\]
Example (1)

```sql
select *
from students
where age = (select max(age)
            from students)
```

How to execute it?
Example (2)

```sql
select *
from R
where R.a in (select S.b
              from S
              where S.c = R.d)
```

How to execute it?
Example (2)

```
select *
from R
where R.a in (select S.b
                from S
                where S.c = R.d)
```

How to execute it?
*Correlated subquery.*

Subquery unnesting – getting rid of subqueries.
Type A subqueries

```sql
select *
from students
where age = (select max(age)
               from students)
```

Evaluate inner block only once – build side of a join.
Outer query becomes the probe side of a join
select *
from students
where student_id in (select student_id
                      from students
                      where semester = 1)

Same technique. Subquery becomes a build side of a *semi-join*
select *
from students s1
where s1.age = (select max(s2.age)
    from students s2
    where s2.dept_id = s1.dept_id)
select *
from students s1
where s1.age = (select max(s2.age)
                from students s2
                where s2.dept_id = s1.dept_id)

Group by s2.dept_id, compute max(s2.age).
Then join with the s1
Type JA subqueries: COUNT bug

```sql
select d.name
from department d
where d.num_prof > (select count(s.id)
    from students s
    where s.dept_id = d.id)
```
Type JA subqueries: COUNT bug

```sql
select d.name
from department d
where d.num_prof > (select count(s.id)
    from students s
    where s.dept_id = d.id)

COUNT(∅) = 0
Use outer-join for unnesting COUNT queries
```
Dealing with quantifiers

- ... < ANY (select ...) ⇒ ... < (select max() ...)
- ... < ALL (select ...)

11 / 16
Dealing with quantifiers

- ... < ANY (select ...) ⇒ ...< (select max() ...)
- ... < ALL (select ...) ⇒ ...< (select min() ...)
- for >: flip the rules
- EXISTS (correlated): semi-join
TPC-H - "22 most well studied SQL queries in history"
select o_orderpriority, count(*) as order_count
from orders
where o_orderdate >= date '1993-07-01'
and o_orderdate < date '1993-10-01'
and exists ( select *
                 from lineitem
                 where l_orderkey = o_orderkey
                      and l_commitdate < l_receiptdate
            )
group by o_orderpriority
order by o_orderpriority
select p_brand, p_type, p_size,
    count(distinct ps_suppkey) as supplier_cnt
from partsupp, part
where  p_partkey = ps_partkey
    and p_brand <> 'Brand#45'
    and p_type not like 'MEDIUM POLISHED%' 
    and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
    and ps_suppkey not in ( 
        select s_suppkey
        from supplier
        where s_comment like '%Customer%Complaints%'
    )
group by  p_brand, p_type, p_size
order by supplier_cnt desc,  p_brand, p_type, p_size
select  sum(l_extendedprice) / 7.0 as avg_yearly
from
  lineitem,
  part
where
  p_partkey = l_partkey
  and p_brand = 'Brand#23'
  and p_container = 'MED BOX'
  and l_quantity < (  
      select  
        0.2 * avg(l_quantity)  
      from  
        lineitem  
      where  
        l_partkey = p_partkey  
    )
Exercises due January 26, 2015.