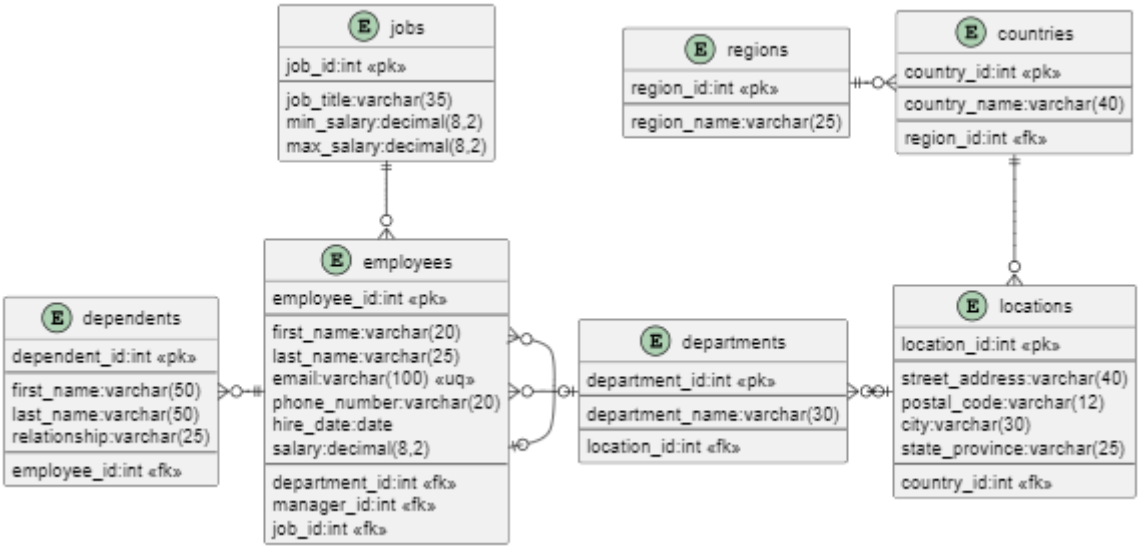
SQL - Data Manipulation Languate (DML)

Schema HR (Human Resources)



Practise

1. Join Table

- Inner Join

```
select
 c.region_id,region_name,country_id,country_name
from regions r join countries c
on r.region_id = c.region_id
order by r.region_name,c.country_id
```

- Inner Join 3 tables

```
select
    c.region_id,region_name,c.country_id,country_name,
    location_id,street_address,city,state_province
from regions r
join countries c on r.region_id = c.region_id
join locations l on c.country_id=l.country_id
```

- Left Join

```
select
    c.country_id,country_name,
```

```
    location_id,street_address,city,state_province
from countries c
left join locations l on c.country_id=l.country_id
```

- Right Join

```
select
    c.country_id,country_name,
    location_id,street_address,city,state_province
from countries c
right join locations l on c.country_id=l.country_id
```

- Full Outer Join

```
select
   c.country_id,country_name,
   location_id,street_address,city,state_province
from countries c
right join locations l on c.country_id=l.country_id
```

** Summary of PostgreSQL JOINs**

JOIN Type	Includes All Countries?	Includes All Locations?
INNER JOIN	✗ No (only with Locations)	✗ No (only with Country)
LEFT JOIN	✓ Yes	✗ No
RIGHT JOIN	✗ No	✓ Yes
FULL OUTER JOIN	✓ Yes	✓ Yes

2. Filtering

- Using In

  Gunakan In untuk filtering data yang kecil/terbatas.

```
select * from employees
where department_id in (9,10)
```

- Using function lower()

```
  select * from employees
  where lower(last_name) like lower('king')
```

- Range Date

```
select          employee_id,first_name||'          '||last_name          as
full_name,hire_date,TO_CHAR(salary, 'FM999999.00') as salary
from HR.employees
where hire_date between '1997-08-17' and '1998-04-23'
```

- Using Extract()

```
select *
from employees where extract(MONTH from hire_date)=08
```

2. SubQuery

SubQuery atau *query dalam query* digunakan :

1. Untuk mendapatkan data dari satu table atau lebih sebelum digunakan oleh query utama.
2. Sebagai alternatif dari join table untuk fetch row data berukuran kecil.

Contoh :

- Using IN operator

```
select * from departments where location_id in (
    select
    l.location_id
    from regions r
    join countries c on r.region_id = c.region_id
    join locations l on c.country_id=l.country_id
    order by l.location_id
)
```

```
select department_id,department_name,
(select  street_address||'  '||postal_code||'  '||city||'  '||state_province
from locations l where l.location_id=d.location_id)
as address
from departments d
```

Kedua query diatas memilkiki performance yang lambat karena me-disable-kan index, jadi prioritaskan menggunakan join seperti sql dibawah :

```

```

```
select
    department_id,department_name,l.location_id
from regions r
join countries c on r.region_id = c.region_id
join locations l on c.country_id=l.country_id
join departments d on d.location_id=l.location_id
order by l.location_id
```

- Using EXISTS Operator

  Kita bisa gunakan operator *Exists* yang memiliki performance lebih cepat dibanding *IN* Operator.

```
select * from departments where exists (
    select
    1
    from regions r
    join countries c on r.region_id = c.region_id
    join locations l on c.country_id=l.country_id
    order by l.location_id
)
```

3. Query Function Aggregate

- Function Sum

```
select
    d.department_id,d.department_name,
    sum(salary) as total_salary
from departments d
join employees e on d.department_id=e.department_id
group by d.department_id,d.department_name
order by department_name
```

- Function Average (AVG)

```
select
    d.department_id,d.department_name,
    avg(salary) as total_salary
from departments d
join employees e on d.department_id=e.department_id
group by d.department_id,d.department_name
order by department_name
```

- Count Operator

```
select
    d.department_id,d.department_name,
    count(1) as total_employee
from departments d
join employees e on d.department_id=e.department_id
group by d.department_id,d.department_name
order by department_name
```

  - Symbol 1, menghitung seluruh baris tapi value null akan di-skip.
  - Symbol *, menghitung seleuruh baris termasuk nilai null.

- Function Minimun (Min)

```
select
  d.department_id,d.department_name,
  min(salary) as min_salary
from departments d
join employees e on d.department_id=e.department_id
group by d.department_id,d.department_name
order by department_name
```

- Function Maximum (Max)

```
select
  d.department_id,d.department_name,
  max(salary) as max_salary
from departments d
join employees e
on d.department_id=e.department_id
group by d.department_id,d.department_name
order by department_name
```

- Function Having

  Having digunakan untuk filtering data setelah aggregation.

```
select
    d.department_id,d.department_name,
    max(salary) as max_salary
from departments d
join employees e
on d.department_id=e.department_id
group by d.department_id,d.department_name
having max(salary) >= 12000
order by department_name
```

4. Command Table Expression (CTE)

Prioritaskan gunakan CTE dibanding SubQuery.

- Simple CTE

```
with emps as(
  select *
  from employees where department_id in (9,10))
select * from emps where salary >= 8000
```

- Multiple CTE

```
with cte1 as(
    select
  l.location_id
  from regions r
  join countries c on r.region_id = c.region_id
  join locations l on c.country_id=l.country_id
),
cte2 as (
    select * from departments where department_id in (9,10)
)
select * from cte1 join cte2 on cte1.location_id = cte2.location_id
```

- CTE Recursive

```
  with recursive hirarki as (
      select
            employee_id,first_name||' '||last_name as full_name,manager_id,1
as level,
          cast(first_name||' '||last_name as text) as path
      from employees where manager_id is null
      union all
      select
                              k.employee_id,first_name||'   '||last_name   as
full_name,k.manager_id,  h.level + 1,
          h.path || ' > ' || first_name||' '||last_name
      from employees k
      join hirarki h on k.manager_id=h.employee_id
  )
  select * from hirarki
  order by path
```

Result :

employee_id integer	full_name	manager_id integer	level integer	path
100	Steven King	(NULL)	1	Steven King
121	Adam Fripp	100	2	Steven King > Adam Fripp
179	Charles Johnson	100	2	Steven King > Charles Johnson
114	Den Raphaely	100	2	Steven King > Den Raphaely
115	Alexander Khoo	114	3	Steven King > Den Raphaely > Alexander Khoo
118	Guy Himuro	114	3	Steven King > Den Raphaely > Guy Himuro
119	Karen Colmenares	114	3	Steven King > Den Raphaely > Karen Colmenares
116	Shelli Baida	114	3	Steven King > Den Raphaely > Shelli Baida
117	Sigal Tobias	114	3	Steven King > Den Raphaely > Sigal Tobias
177	Jack Livingston	100	2	Steven King > Jack Livingston
145	John Russell	100	2	Steven King > John Russell
176	Jonathon Taylor	100	2	Steven King > Jonathon Taylor
146	Karen Partners	100	2	Steven King > Karen Partners
178	Kimberely Grant	100	2	Steven King > Kimberely Grant
102	Lex De Haan	100	2	Steven King > Lex De Haan
103	Alexander Hunold	102	3	Steven King > Lex De Haan > Alexander Hunold