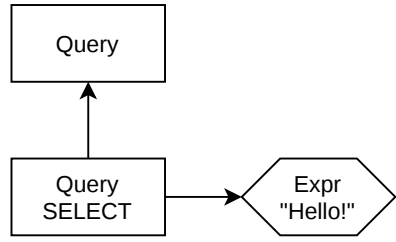


SELECT "Hello World!"

sql_select("Hello World!")

sql_query() |> sql_select("Hello World!")



SELECT p.mrn FROM patient p

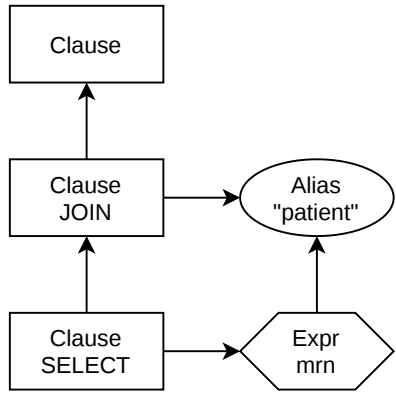
p = sql_alias("patient")

sql_join(p) |> sql_select(p.mrn)

p = sql_alias("patient")

p |> sql_select(p.mrn)

(p = sql_from("patient")) |> sql_select(p.mrn)



SELECT p.mrn, e.date

FROM patient p

JOIN encounter e ON (p.id = e.patient_id)

p = sql_alias("patient")

e = sql_alias("encounter")

sql_from(p) |> sql_join(e, p.id, == e.patient_id) |> sql_select(p.mrn, e.date)

p = sql_alias(catalog["public"]["patient"])

e = sql_alias(catalog["public"]["encounter"])

sql_from(p) |> sql_join(e, autojoin=p) |> sql_select(p.mrn, e.date)

p = sql_alias("patient")

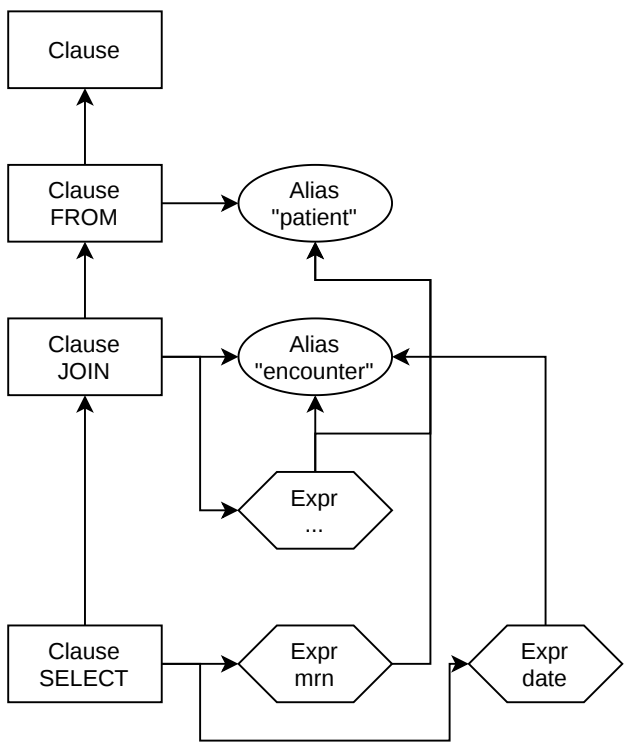
e = sql_alias("encounter")

p |> sql_join(e, p.id, == e.patient_id) |> sql_select(p.mrn, e.date)

p = sql_alias("patient")

e = sql_alias("encounter")

sql_from(p) |> sql_join(e, p.id, == e.patient_id) |> sql_select(p.mrn) |> sql_select(e.date)



SELECT p.mrn, e.date

FROM patient p

JOIN encounter e ON (p.id = e.patient_id)

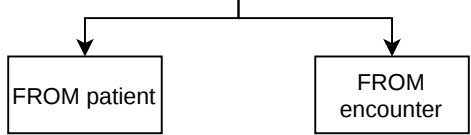
p = From("patient")

e = From("encounter")

j = Join(p, e, p.id, == e.patient_id)

Select(j, p.mrn, e.date)

sql_from((p = sql_from("patient")) |> sql_join((e = sql_alias("encounter")), p.id, == e.patient_id) |> sql_select(p.mrn, e.date))



SELECT p.sex, COUNT(p)

FROM patient p

GROUP BY p.sex

p = sql_alias("patient")

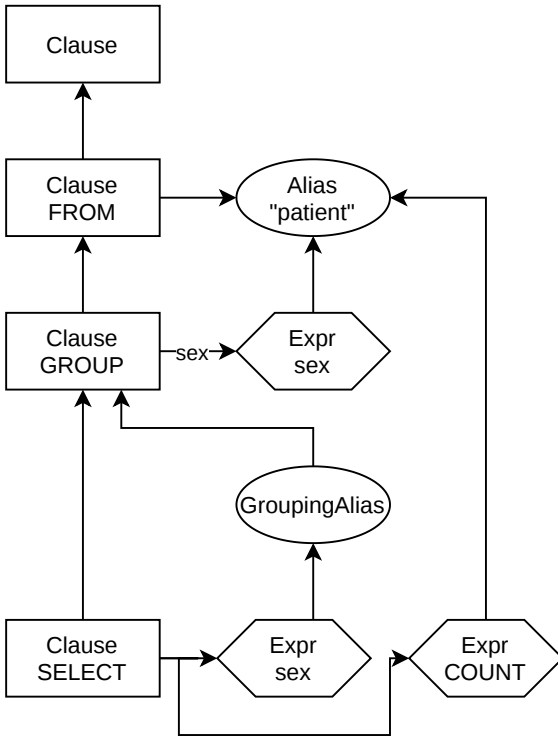
g = sql_from(p) |> sql_group(sex = p.sex)

g |> sql_select(g.sex, sql_count(p))

p = From("patient")

g = Group(p, sex = p.sex)

Select(g, g.sex, Count(p))



SELECT p.mrn, COALESCE(g.n_e, 0)

FROM patient p

LEFT JOIN (

SELECT e.patient_id, COUNT(e) AS n_e

FROM encounter e

GROUP BY e.patient_id) g ON (p.id = g.patient_id)

p = From("patient")

e = From("encounter")

g = Group(p, patient_id = e.patient_id)

j = LeftJoin(p, g, p.id, == g.patient_id, omit_if_unused=true)

Select(j, p.mrn, Coalesce(Count(e), 0))

p = From("patient")

e = From("encounter")

g = Group(e, patient_id = e.patient_id)

gs = Select(g, patient_id = g.patient_id, n = Count(e))

j = LeftJoin(p, gs, p.id, == gs.patient_id)

Select(j, p.mrn, Coalesce(gs.n, 0))

p = From("patient")

e = From("encounter")

g = Group(e, patient_id = e.patient_id, summarize=(, n = Count(e)))

j = LeftJoin(p, g, p.id, == g.patient_id)

Select(j, p.mrn, Coalesce(g.n, 0))

SELECT p.mrn

FROM patient p

WHERE p.sex = 'male'

p = From("patient")

w = Where(p, p.sex, == 'male')

Select(w, p.mrn)

p = From("patient", columns=["mrn", "sex"])

w = Where(p, Ref(1, 2), == "male", select=[Ref(1,1)])

Select(w, select=[Ref(1,1)])

patient_tbl = Table("patient", [{"id", Int}, {"sex", String}, {"mrn", String}])

encounter_tbl = Table("encounter", [{"id", Int}, {"patient_id", Int}, {"date", Date}])

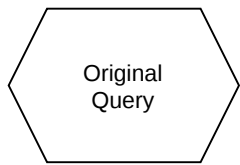
auto_connect(patient_tbl, encounter_tbl, [{"id", "patient_id"}])

p = From(patient_tbl)

e = From(encounter_tbl)

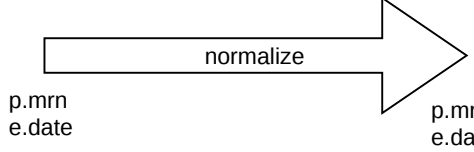
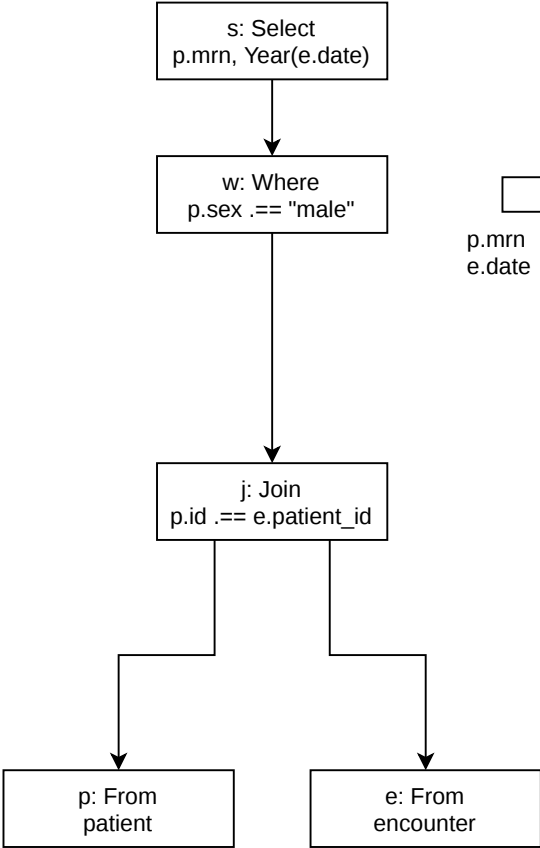
j = LeftJoin(p, e)

Select(j, p.mrn, e.date)

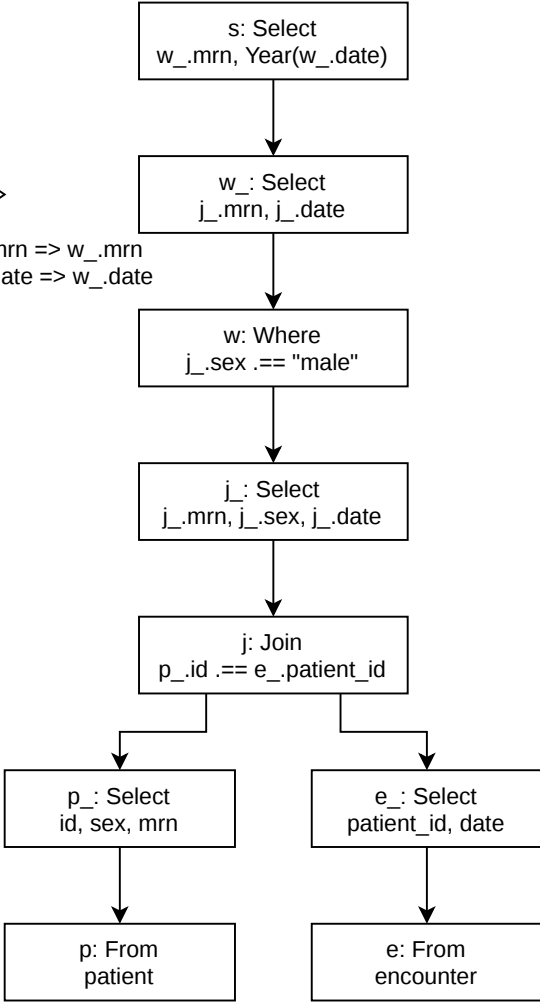


SELECT p.mrn, EXTRACT(YEAR FROM e.date)
FROM patient p
JOIN encounter e
ON (p.id = e.patient_id)
WHERE p.sex = 'male'

p = From(patient)
e = From(encounter)
j = Join(p, e, p.id == e.patient_id)
w = Where(j, p.sex == 'male')
s = Select(w, mrn = p.mrn, year = Year(e.date))

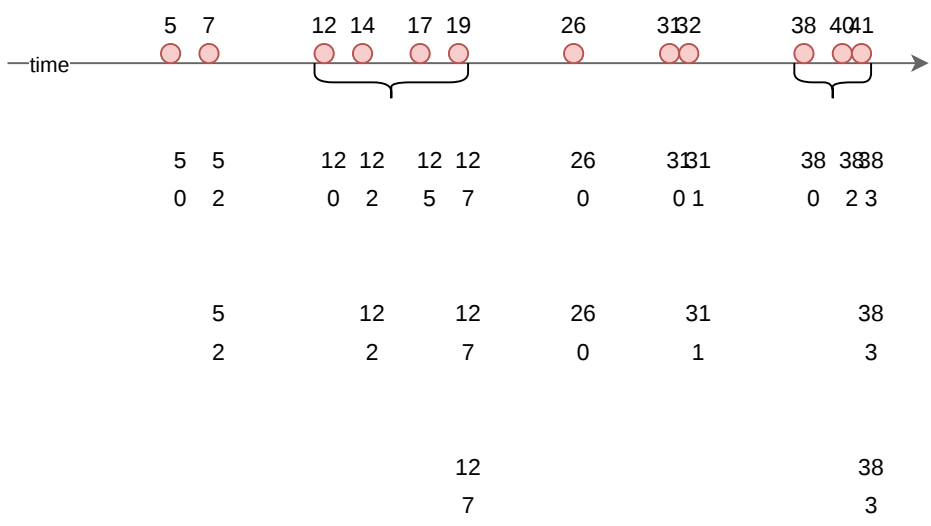


p = From(patient)
p_ = Select(p, id = Const(:id), _sex = Const(:sex), _mrn = Const(mrn))
e = From(encounter)
e_ = Select(e, _patient_id = Const(patient_id), _date = Const(:date))
j = Join(p_, e_, p_.id == e_.patient_id)
j_ = Select(j, _mrn = p_.mrn, _sex = p_.sex, _date = e_.date)
w = Where(j_, j_.sex == 'male')
w_ = Select(w, mrn = j_.mrn, _date = j_.date)
s = Select(w_, mrn = w_.mrn, year = Year(w_.date))

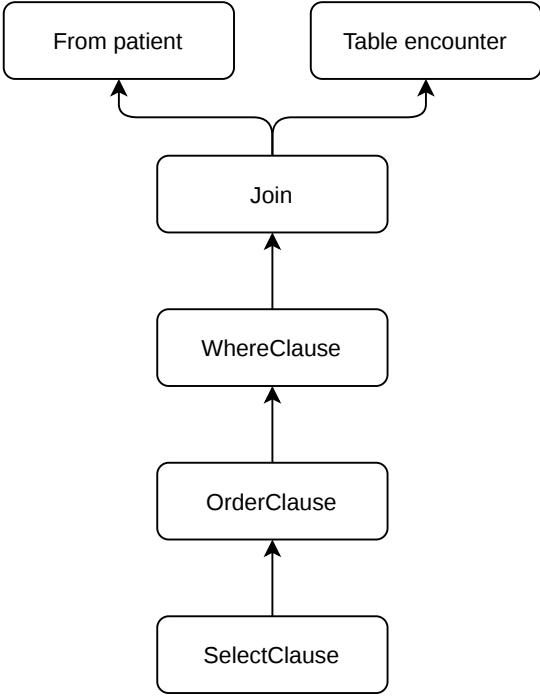
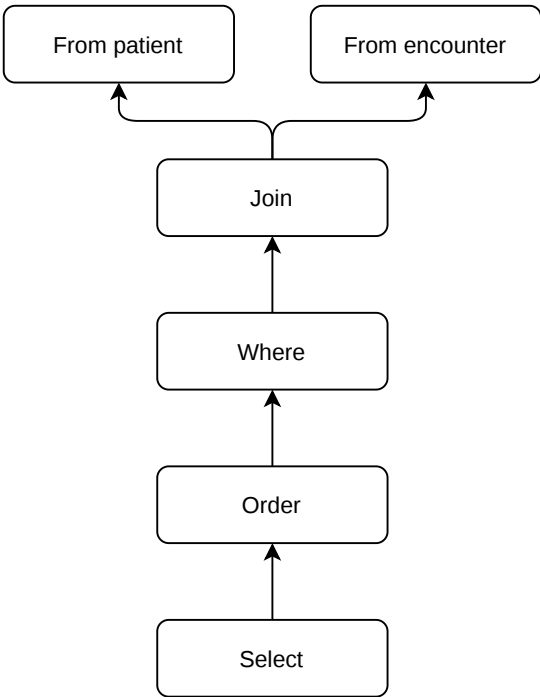


SELECT c.person_id, c.peer_id, c.timestamp, c.distance
FROM contact c

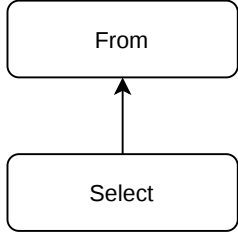
For each pair of persons, find the contact interval when there were detected at least once in a minute in a distance of less than 5 meters.



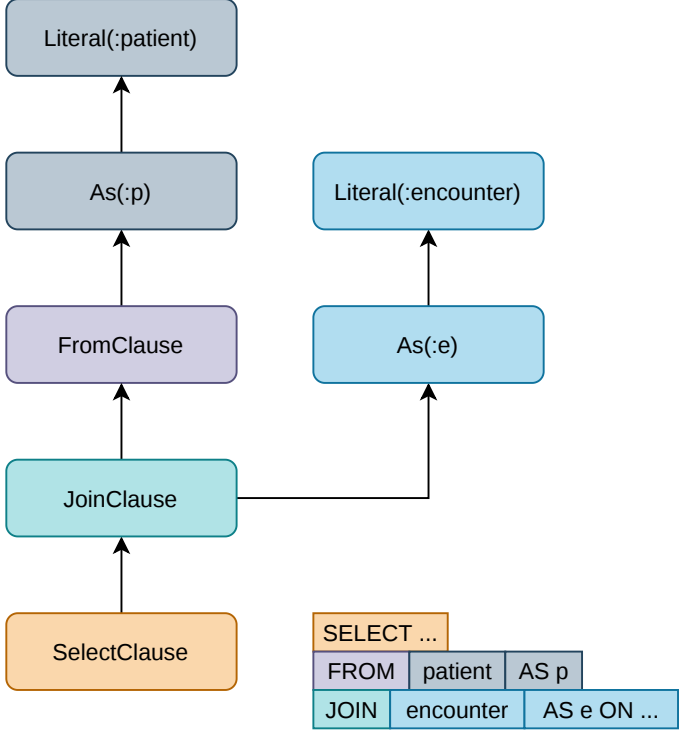
SELECT p.mrn, e.date
FROM patient p
JOIN encounter e ON (p.id = e.patient_id)
WHERE p.sex = 'male'
ORDER BY e.date



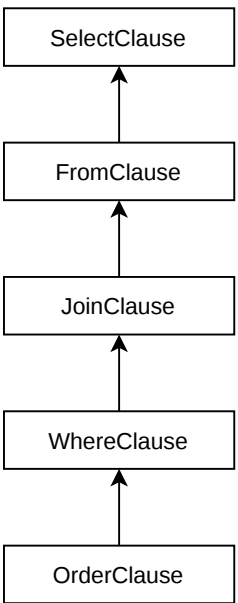
SELECT p.mrn
FROM patient p



SELECT p.mrn, e.date
FROM patient p
JOIN encounter e
ON (p.id = e.patient_id)



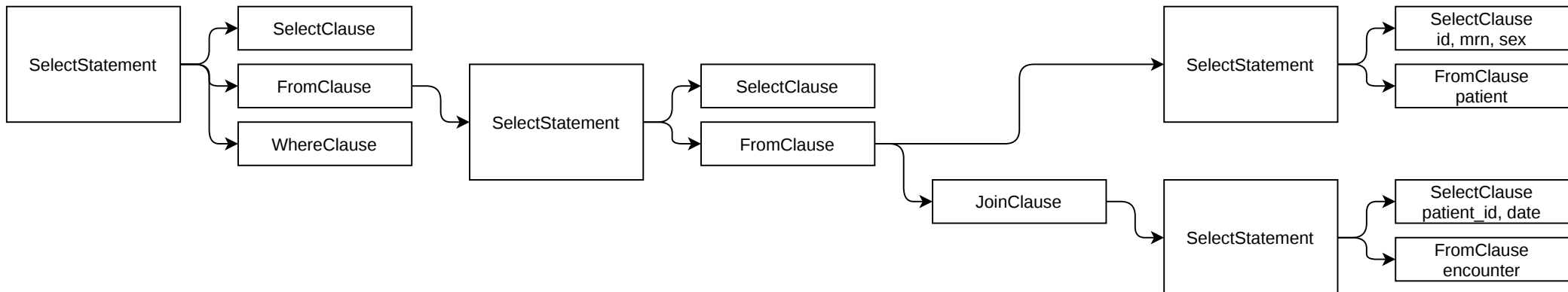
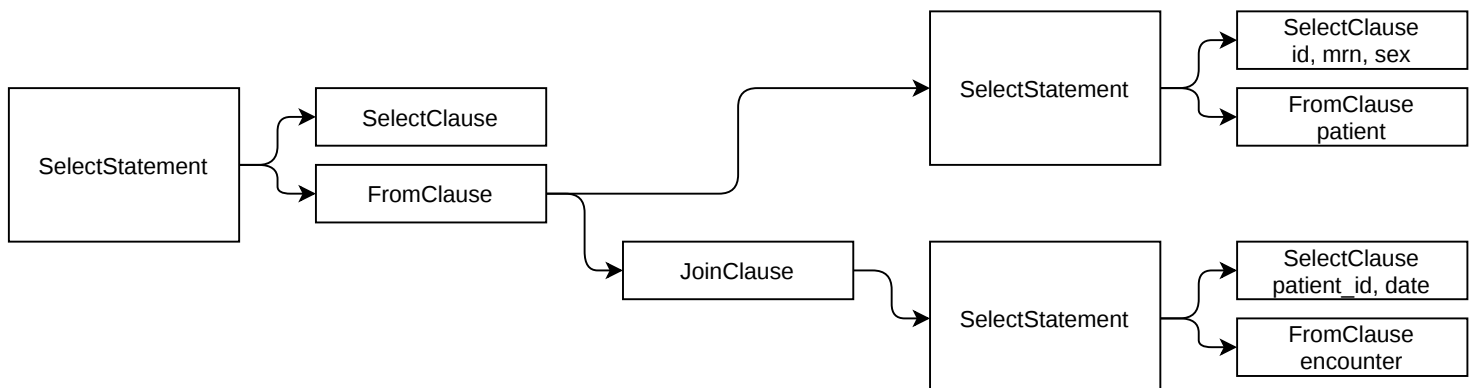
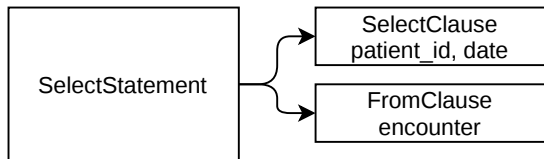
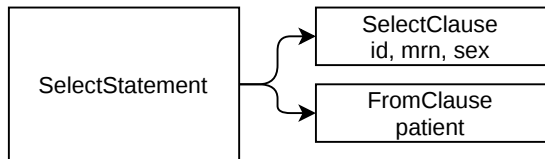
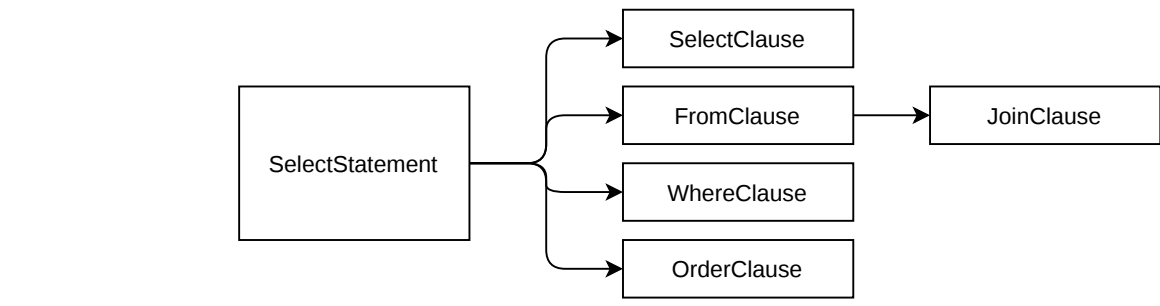
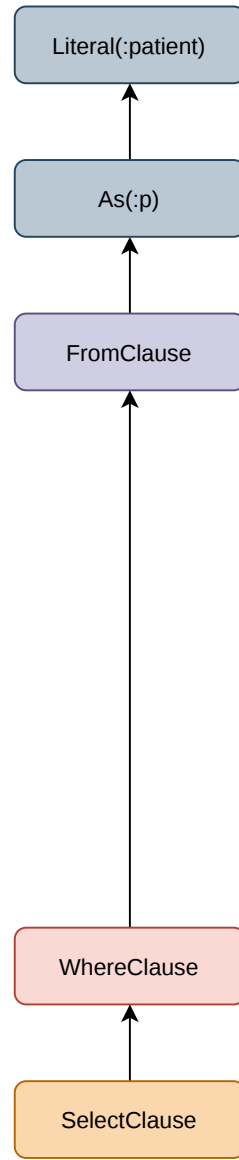
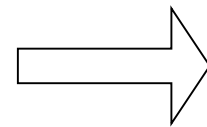
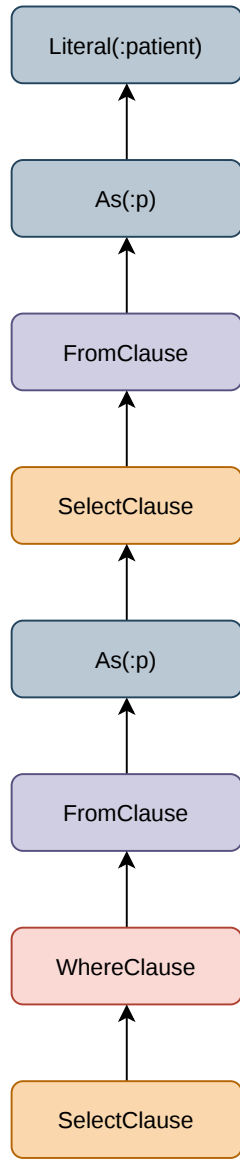
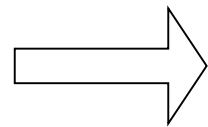
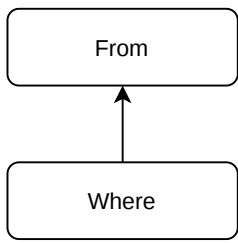
SELECT ...
FROM patient AS p
JOIN encounter AS e ON ...

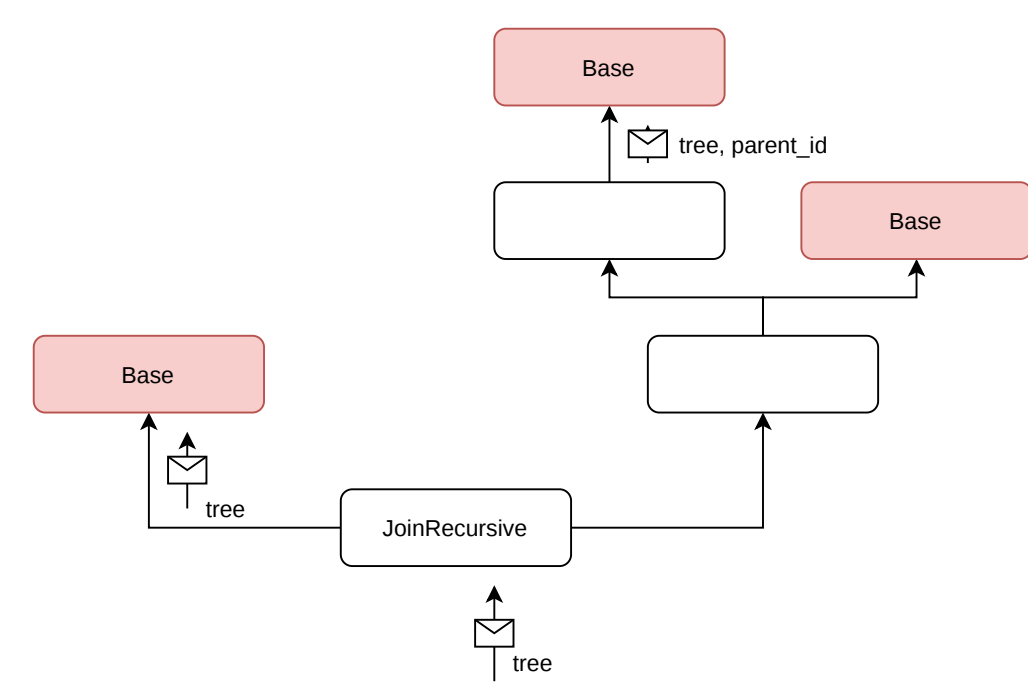


Clause
:SELECT

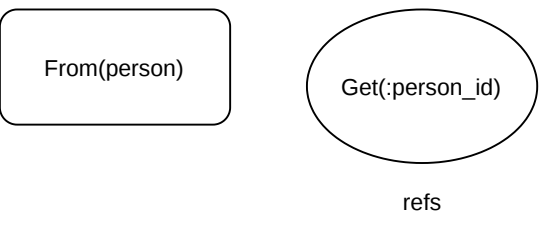
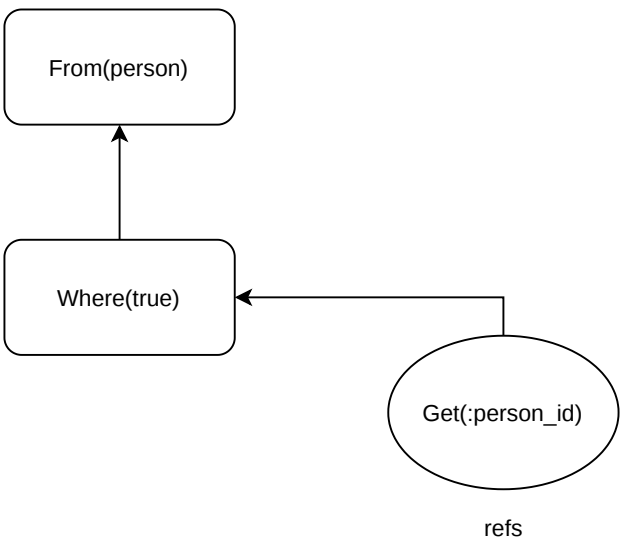
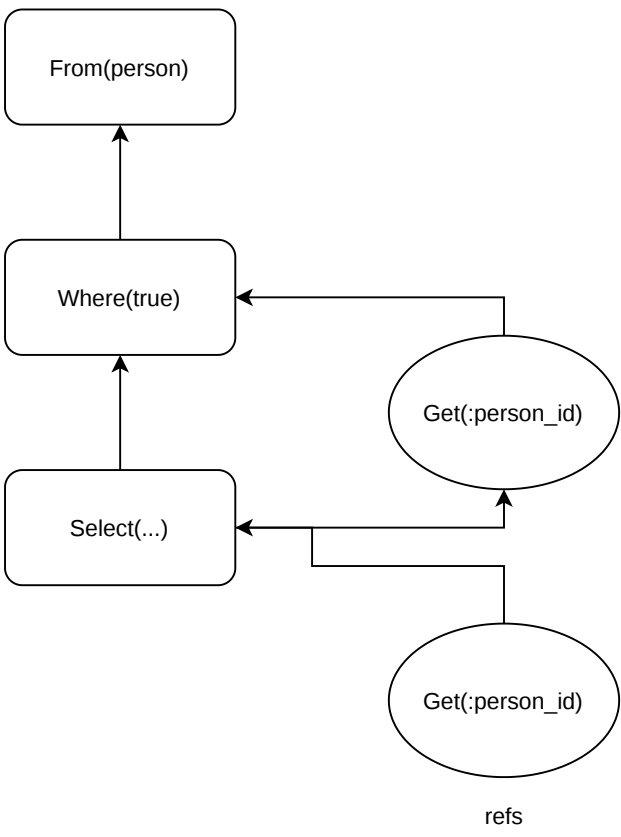
Clause
:FROM

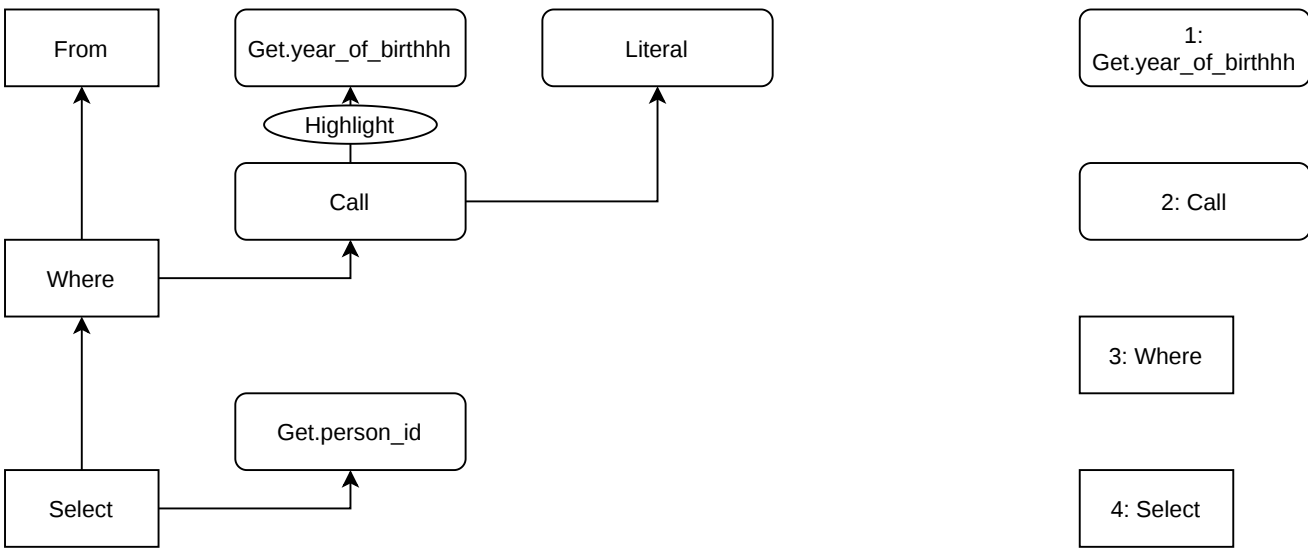
Clause
:JOIN



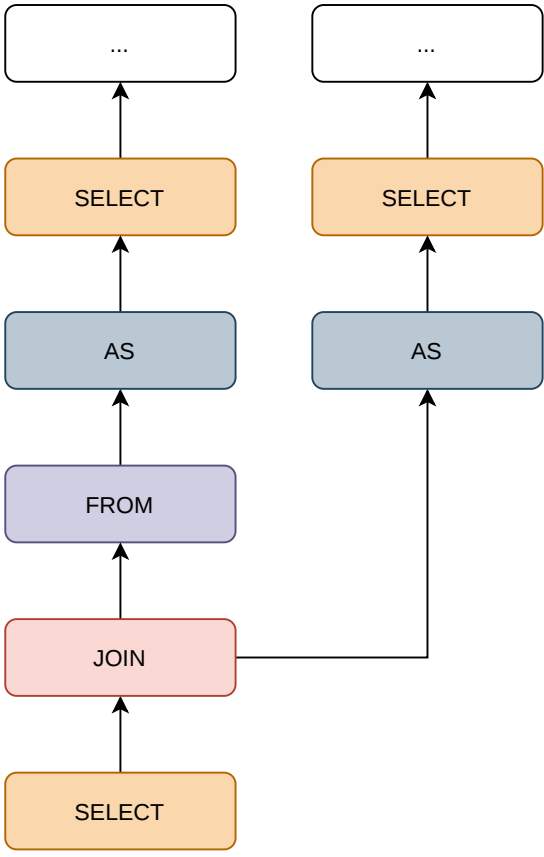
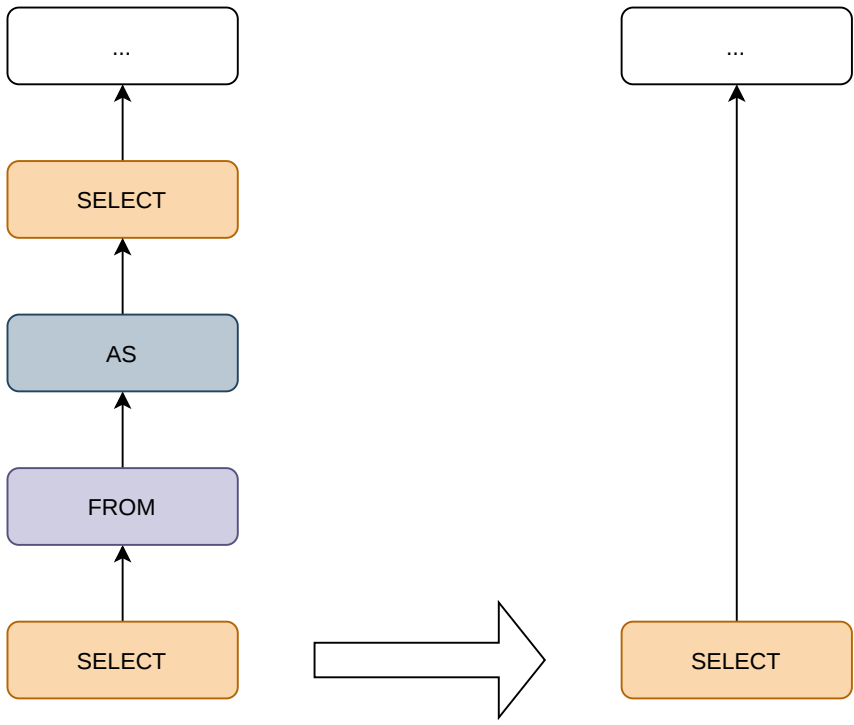


```
WITH RECURSIVE X AS (
  SELECT 1 AS N
  UNION ALL
  SELECT ...
  ...
  FROM X
  ...
  FROM X)
```

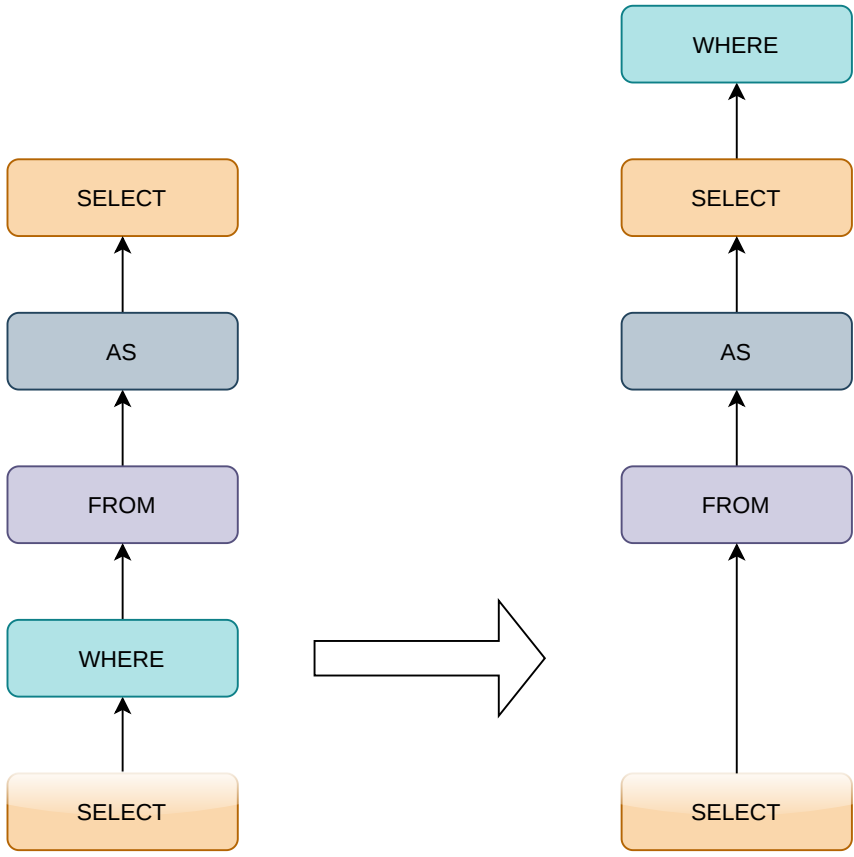




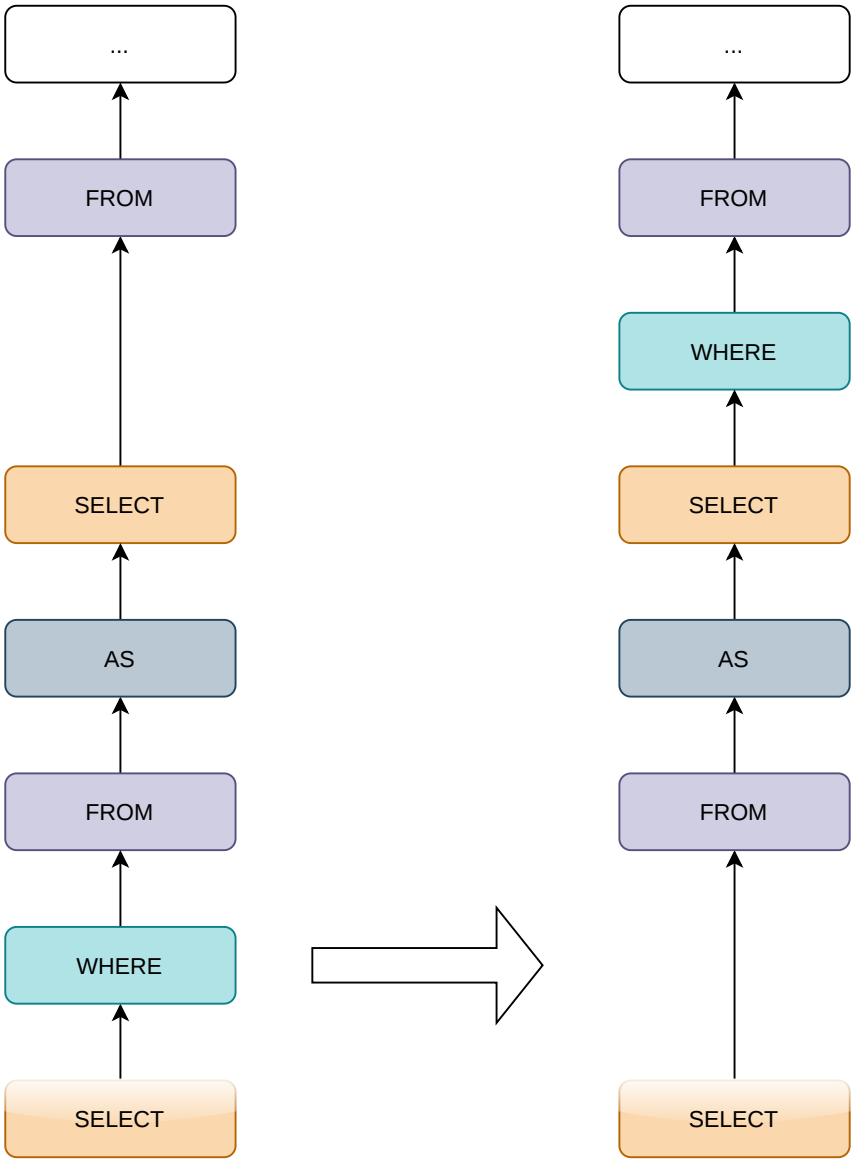
SELECT ... FROM (SELECT) AS ...



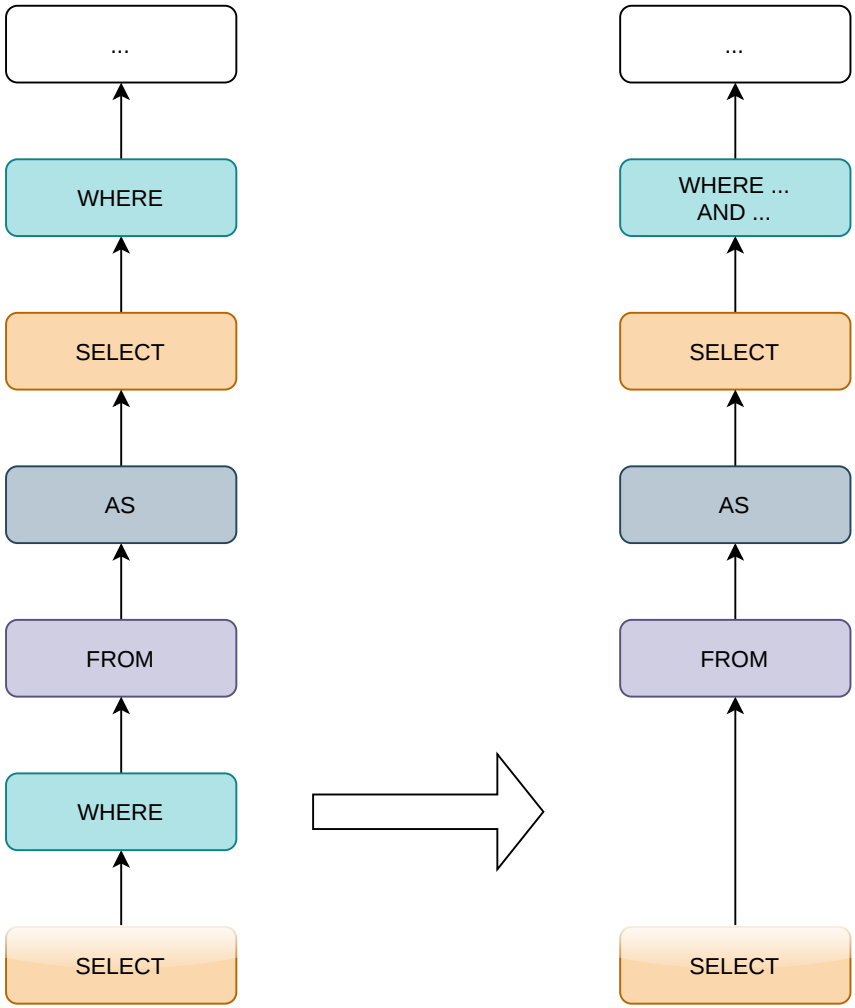
SELECT ... FROM (SELECT ...) AS ... WHERE ...



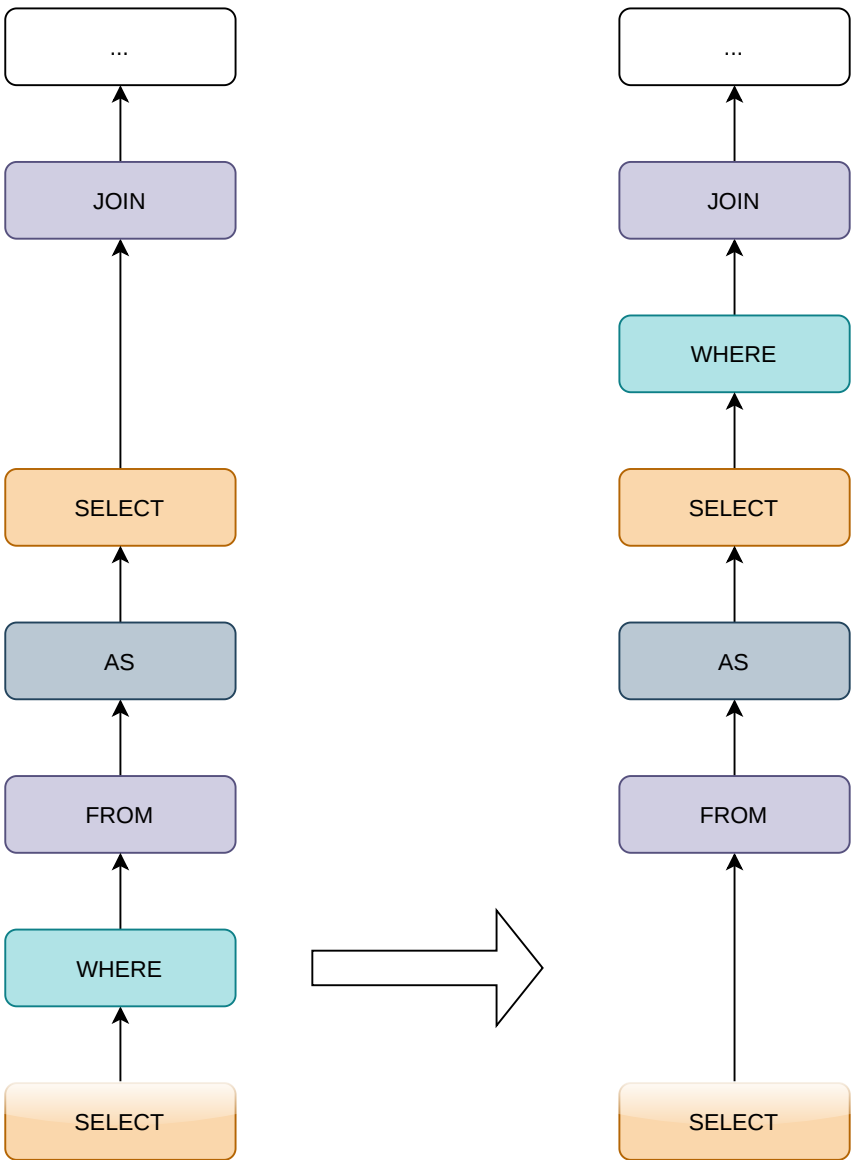
SELECT ... FROM (SELECT ... FROM ...) AS ... WHERE ...

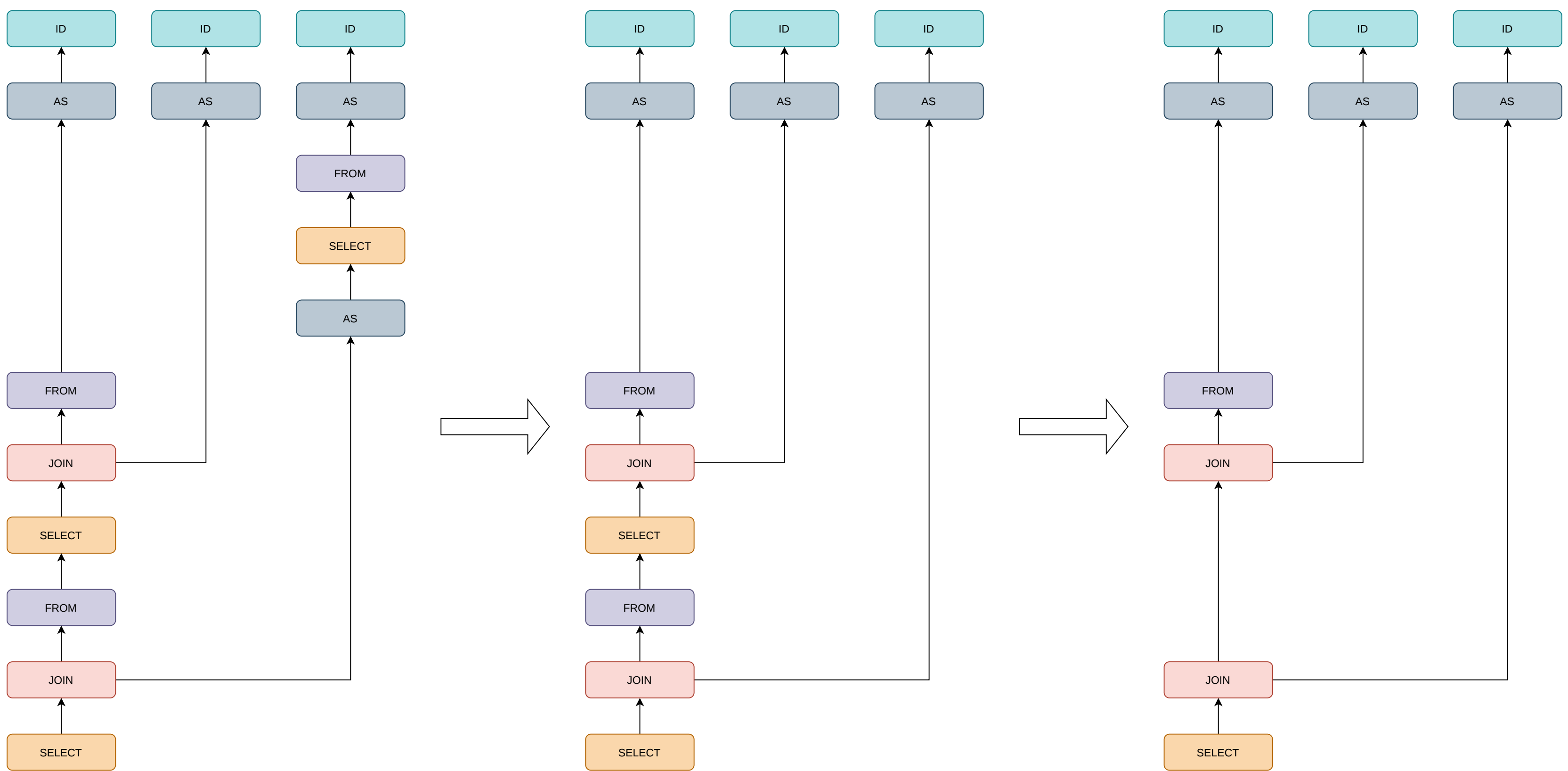
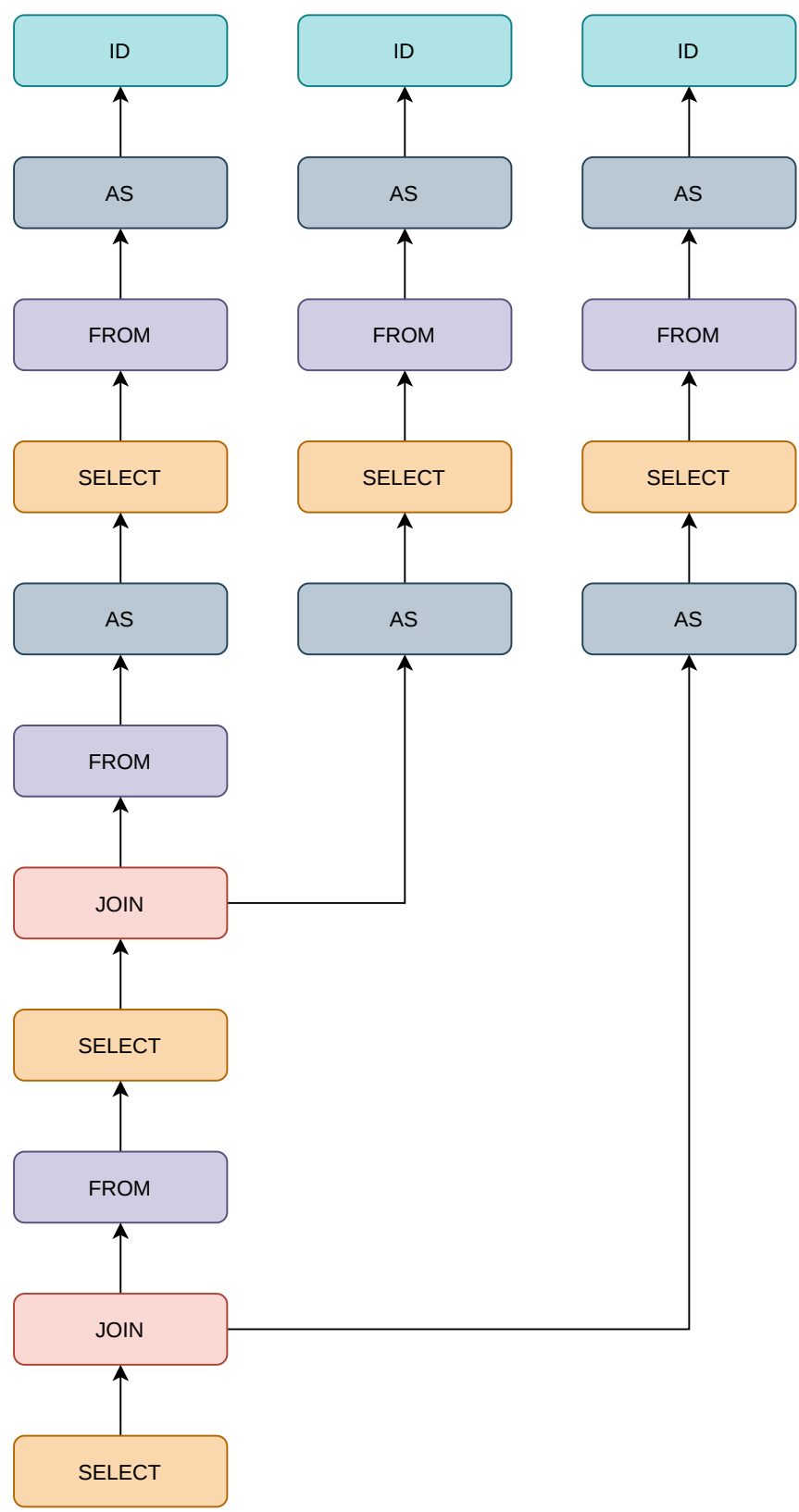
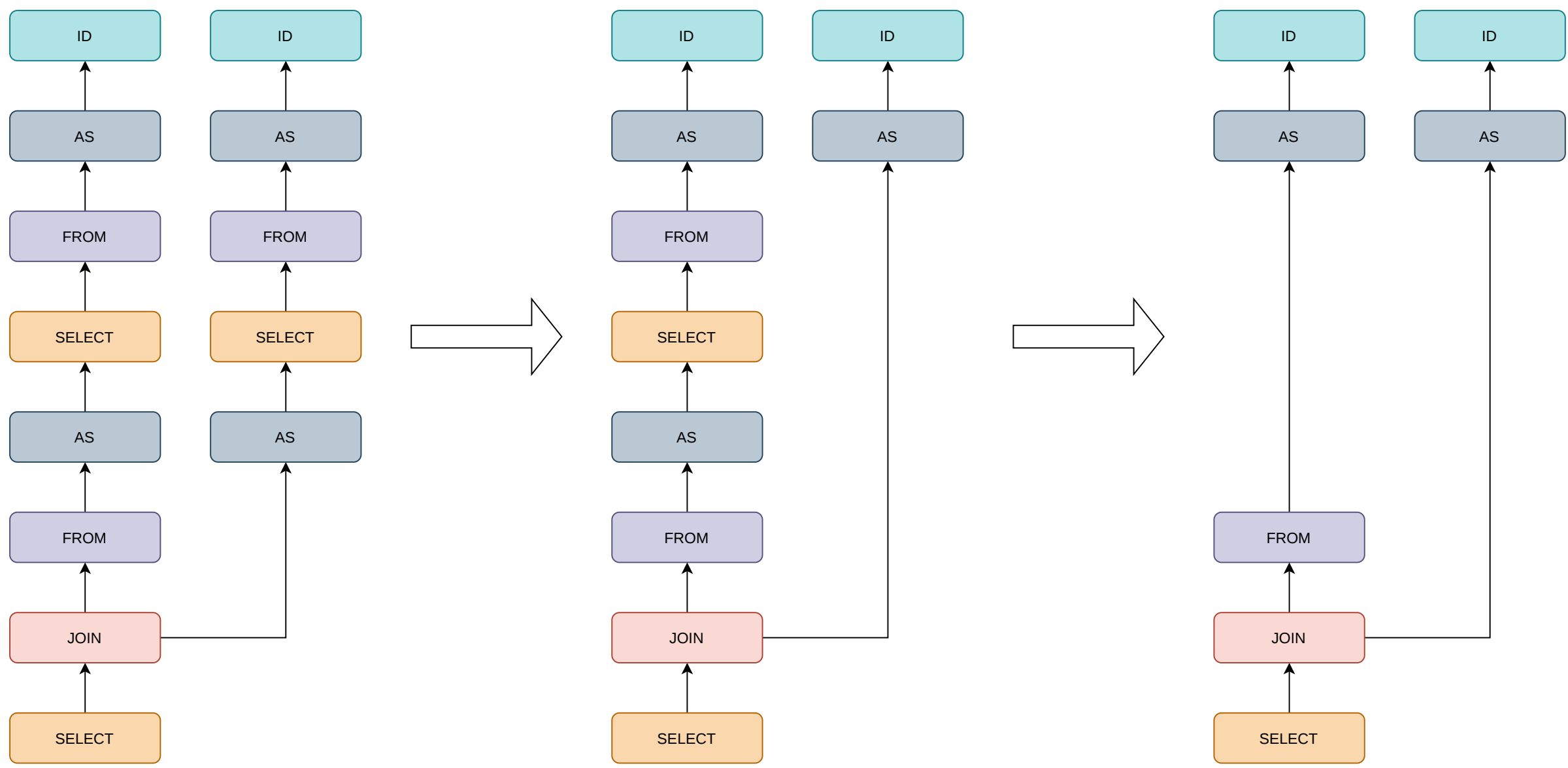


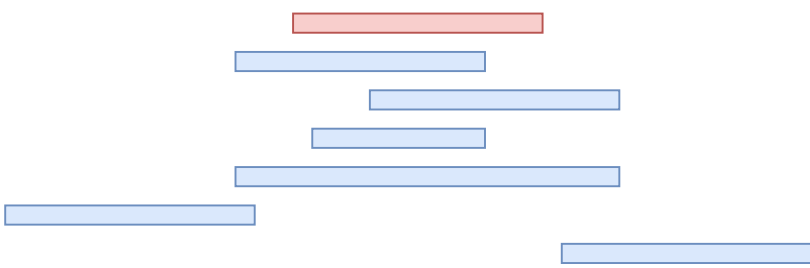
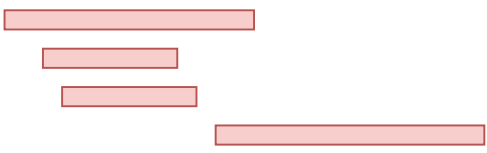
SELECT ... FROM (SELECT WHERE ...) AS ... WHERE ...



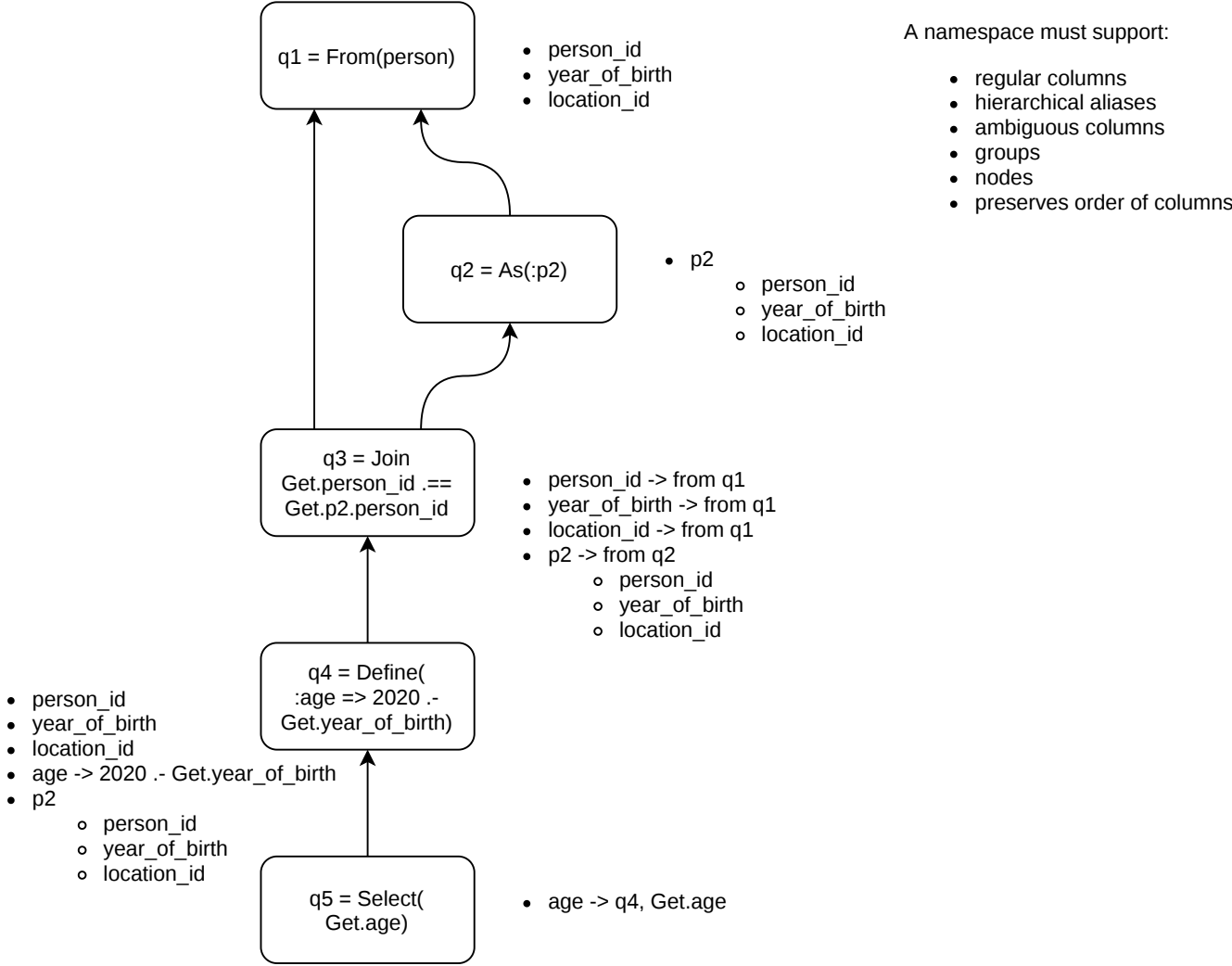
SELECT ... FROM (SELECT JOIN ...) AS ... WHERE ...



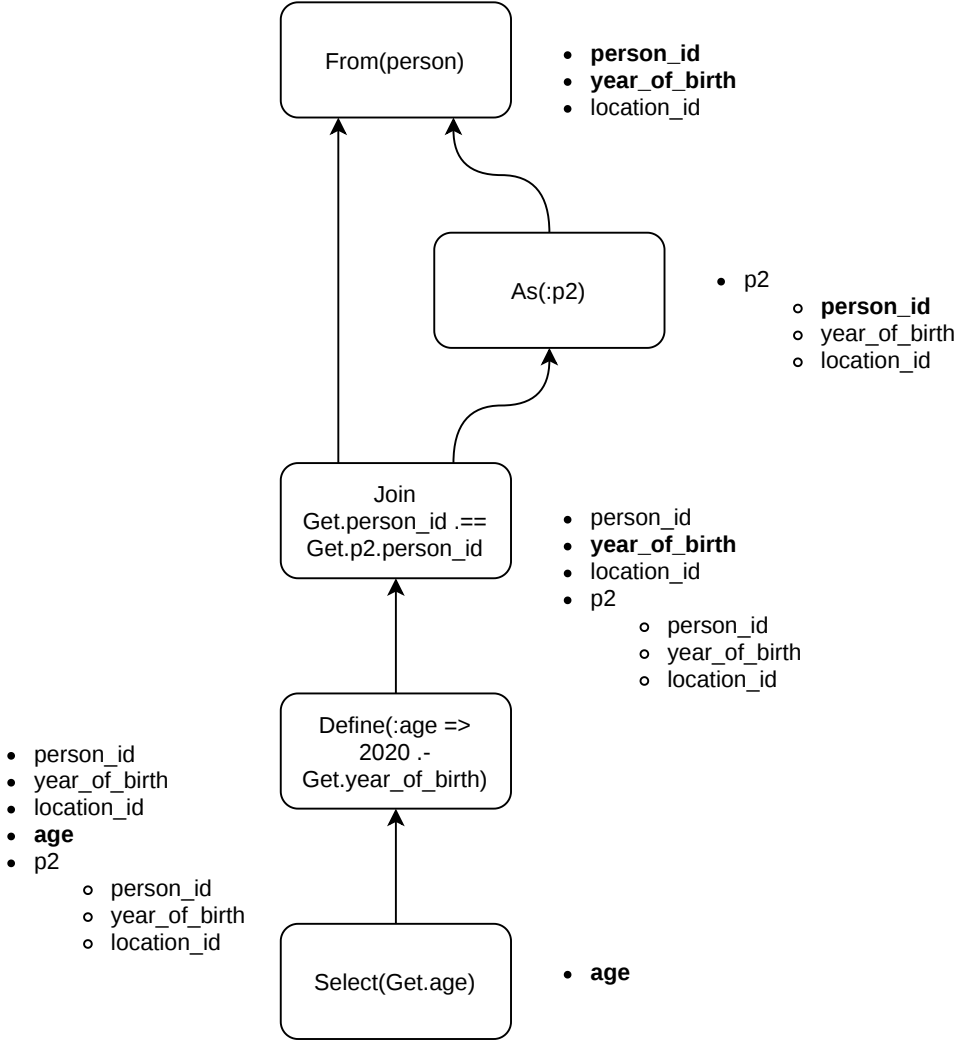




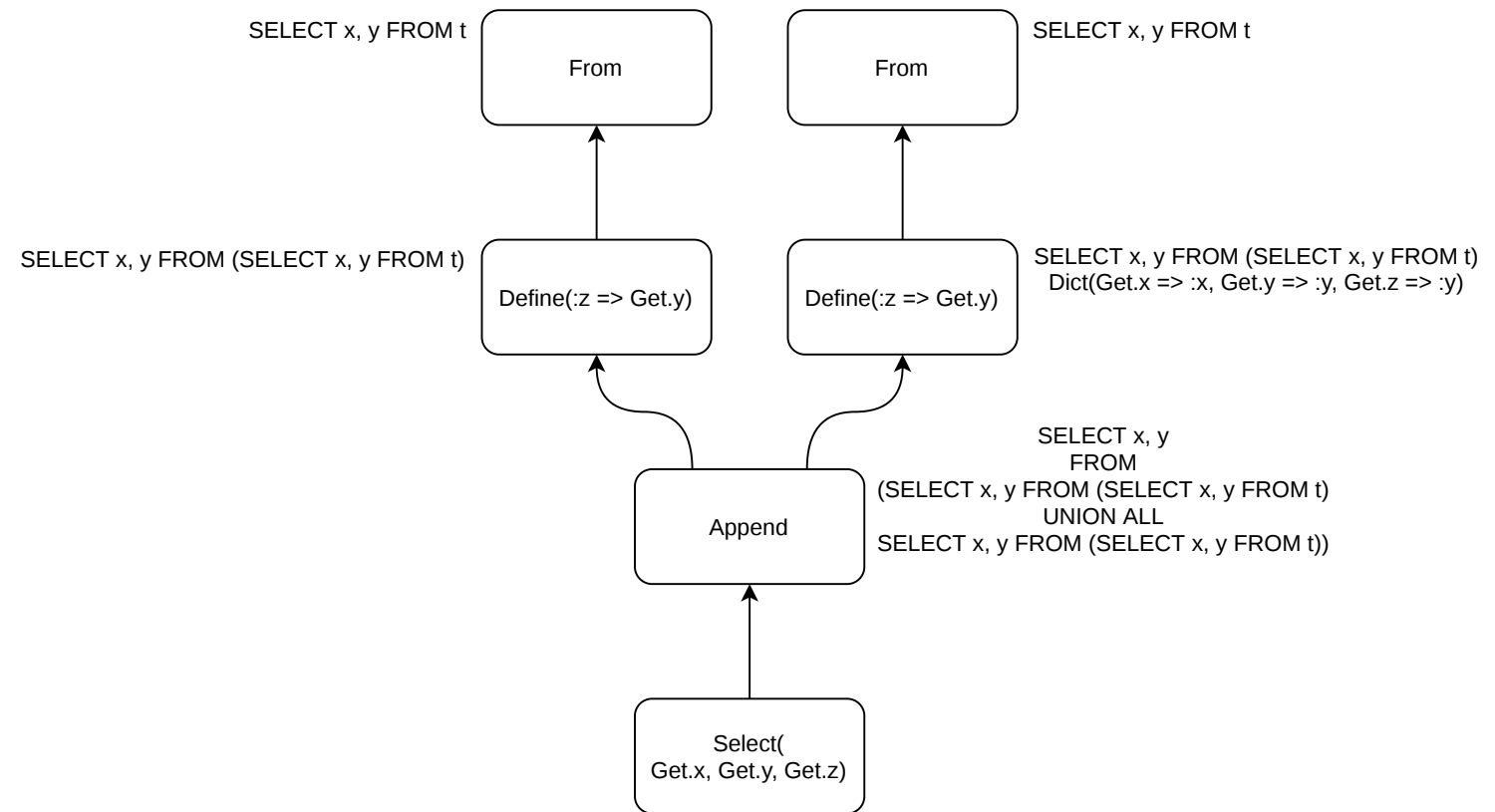
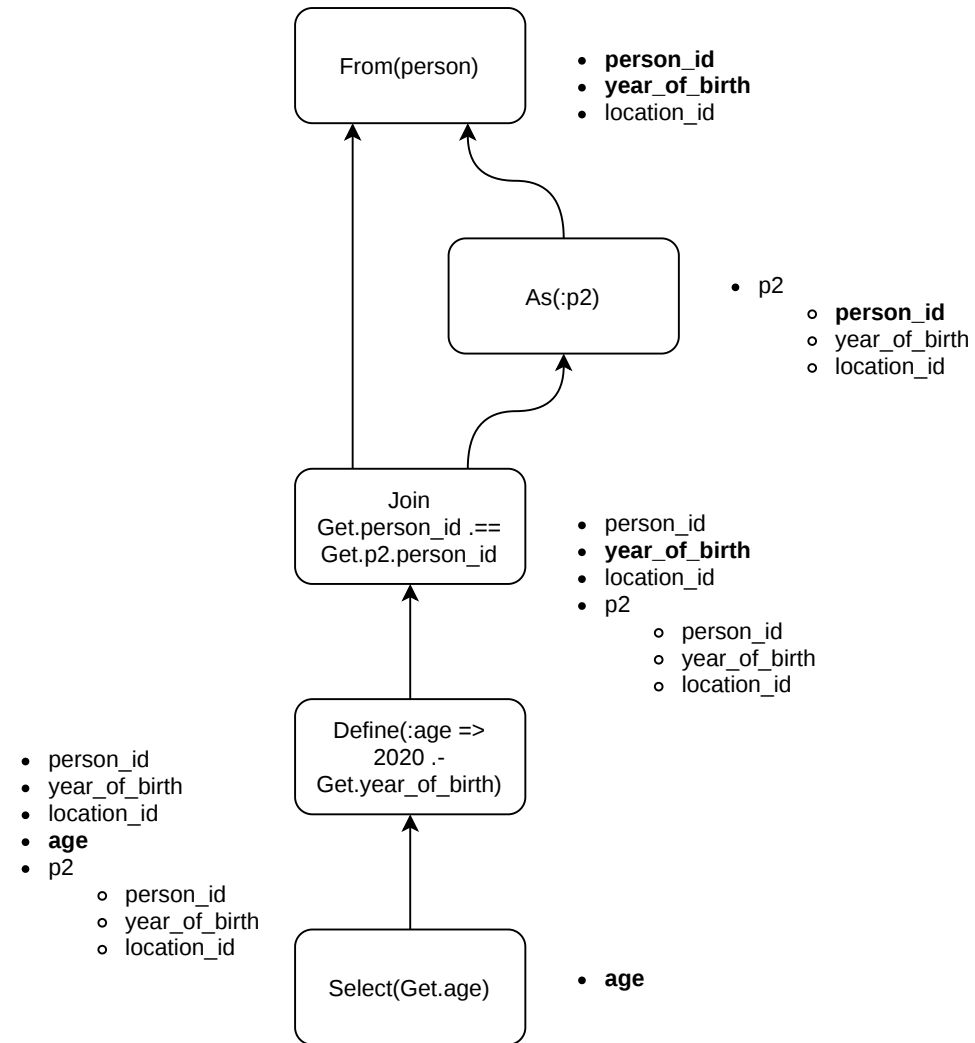
Generate a namespace for each node

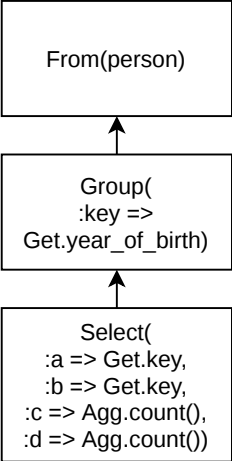


Generate an order for each node



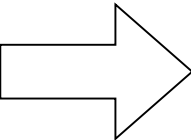
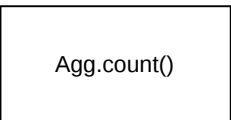
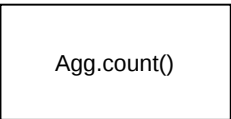
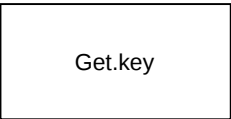
Generate an order for each node





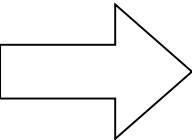
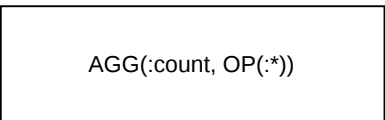
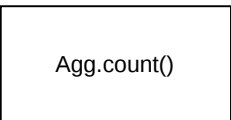
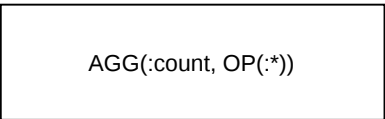
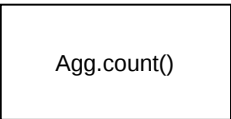
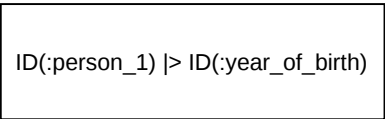
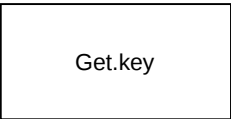
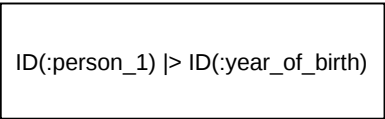
Generate clauses

req.refs::Vector{SQLNode}



req.refs::Vector{SQLNode}

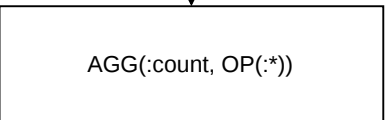
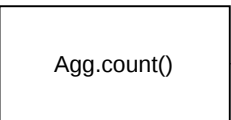
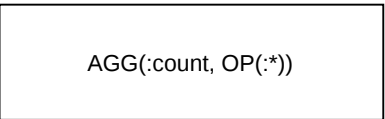
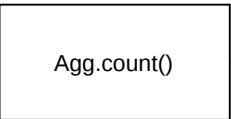
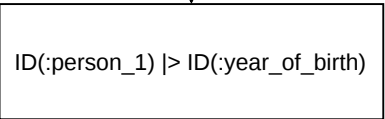
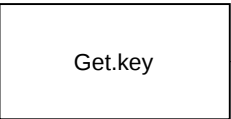
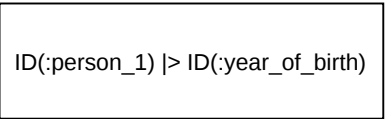
Vector{SQLClause}



Find duplicate clauses

req.refs::Vector{SQLNode}

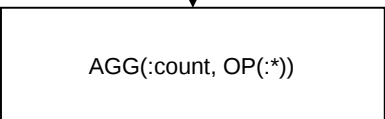
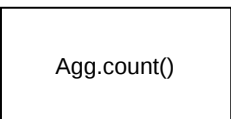
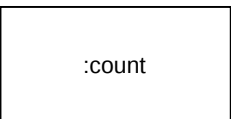
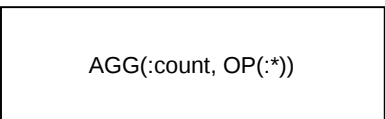
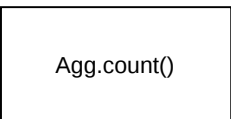
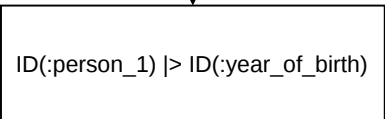
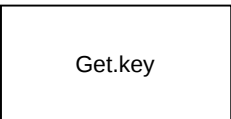
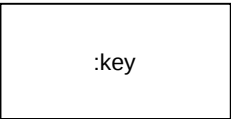
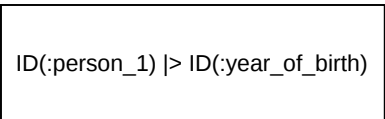
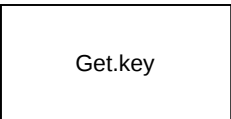
Vector{SQLClause}



Generate column aliases

req.refs::Vector{SQLNode}

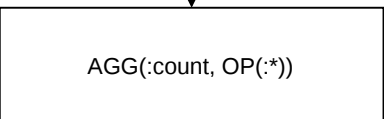
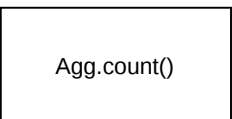
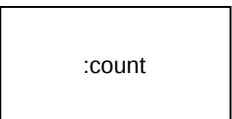
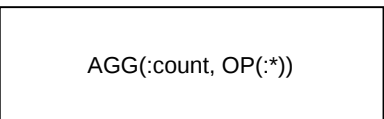
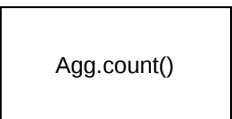
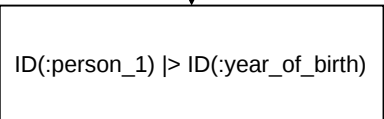
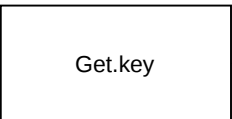
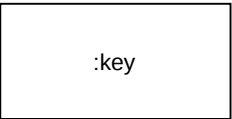
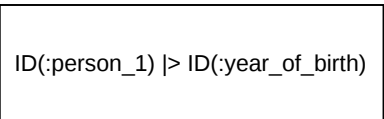
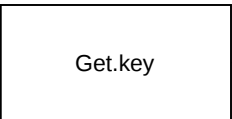
Vector{SQLClause}



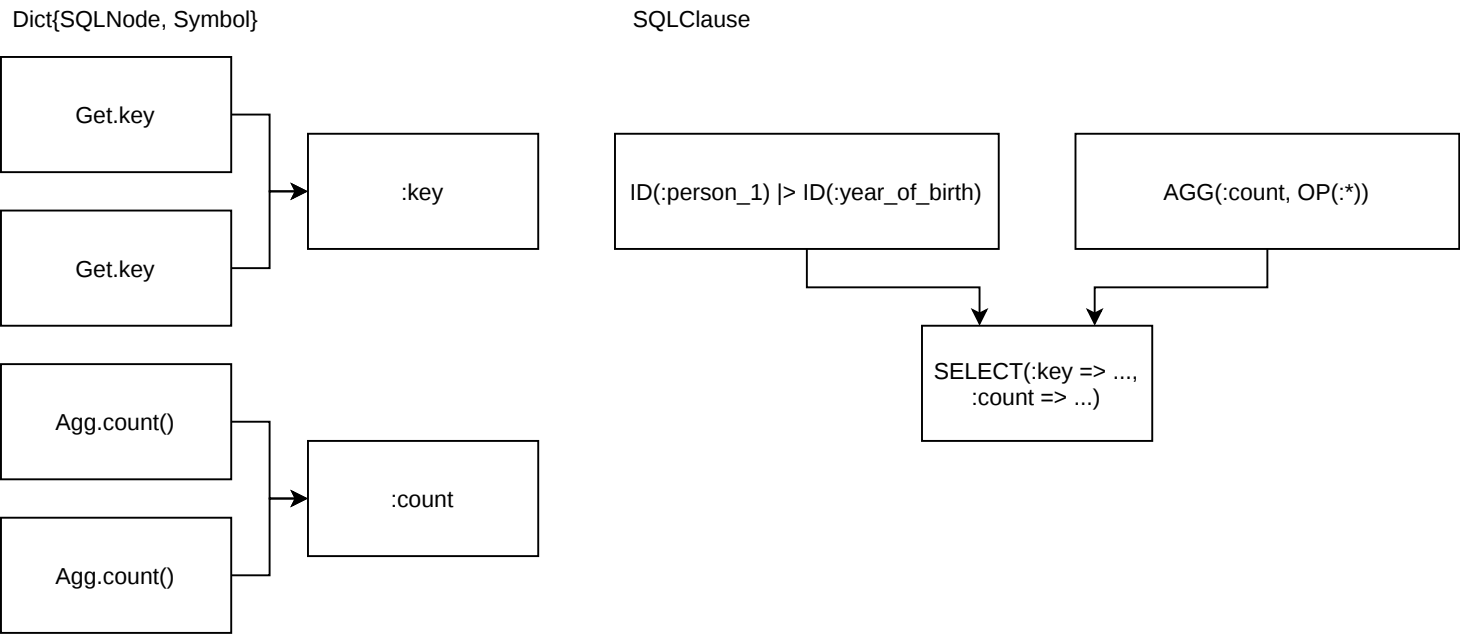
Make column aliases unique

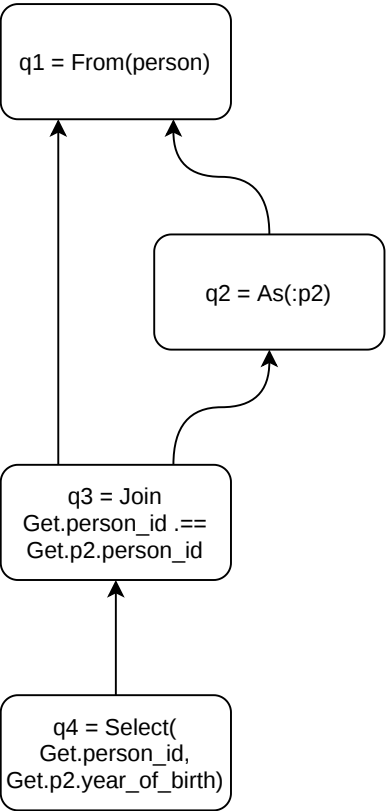
req.refs::Vector{SQLNode}

Vector{SQLClause}



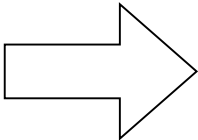
Generate a subquery object and replacement map





[q4, q3, q2, q1]

```
for q in [q4, q3, q2, q1]
  collect_references(q)
end
```



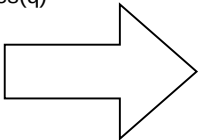
Requests

```
q1 => [Get.person_id (q3), Get.p2.person_id (q3), Get.person_id (q4), Get.p2.year_of_birth (q4), Get.person_id (q2), Get.year_of_birth (q2)]
q2 => [Get.person_id (q3), Get.p2.person_id (q3), Get.person_id (q4), Get.p2.year_of_birth (q4)]
q3 => [Get.person_id (q4), Get.p2.year_of_birth (q4)]
q4 => []
```

Remaps

```
q1 => Dict()
q2 => Dict(Get.p2.person_id (q3) => Get.person_id (q2), Get.p2.year_of_birth (q4) => Get.year_of_birth (q2))
q3 => Dict()
q4 => Dict()
```

```
for q in [q1, q2, q3, q4]
  build_clauses(q)
end
```



Clauses

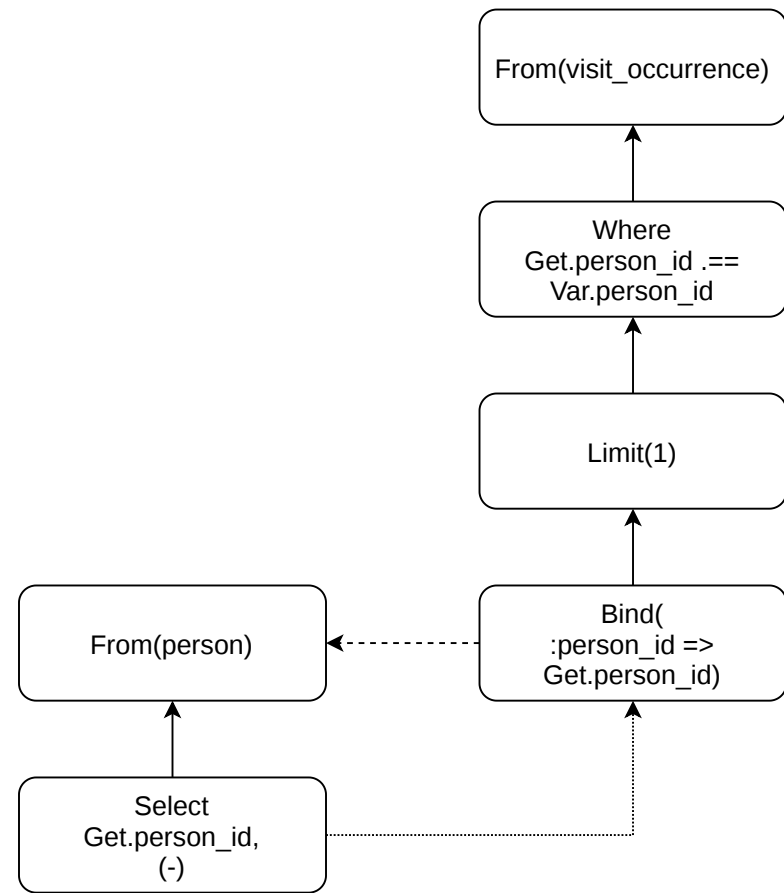
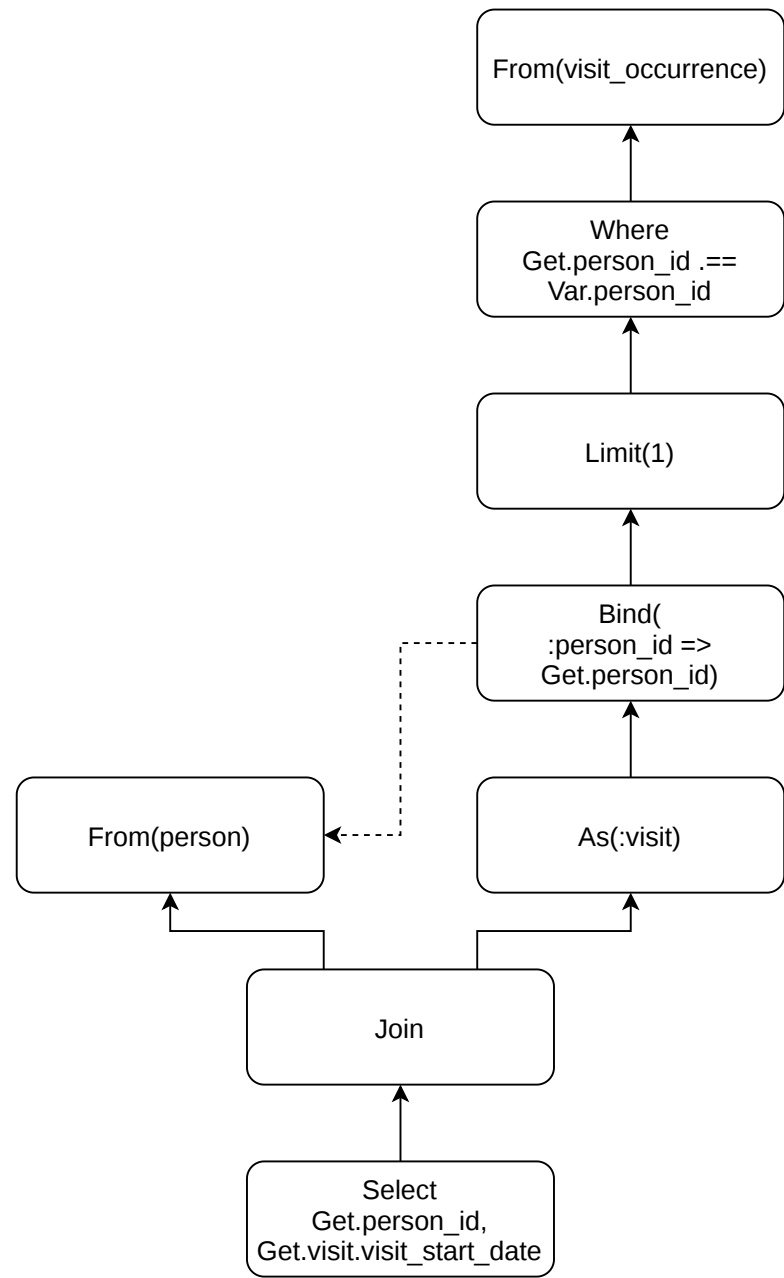
```
q1 => SELECT person_id AS person_id, year_of_birth AS year_of_birth FROM person
q2 => SELECT person_id AS person_id, year_of_birth AS year_of_birth FROM person
q3 => SELECT p1.person_id AS person_id, p2.year_of_birth AS year_of_birth FROM (clauses[q1]) AS p JOIN (clauses[q2]) ON p1.person_id = p2.person_id
q4 => SELECT p3.person_id, p3.year_of_birth FROM (clauses[q3]) p3
```

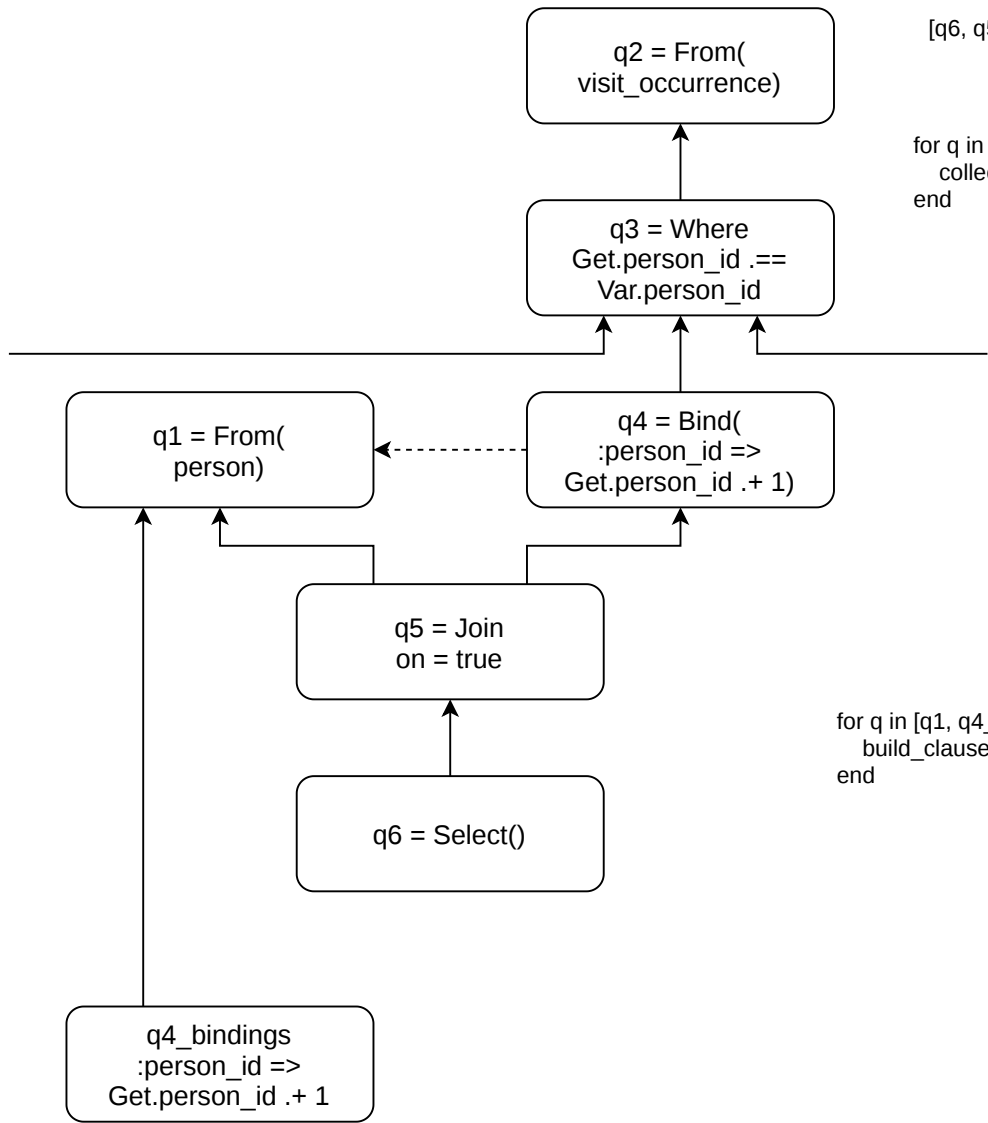
Repl

```
q1 => Dict(Get.person_id (q3) => :person_id, Get.person_id (q4) => :person_id, Get.person_id (q2) => :person_id, Get.year_of_birth (q2) => :year_of_birth)
q2 => Dict(Get.p2.person_id (q3) => :person_id, Get.p2.year_of_birth (q4) => :year_of_birth)
q3 => Dict(Get.person_id (q4) => :person_id, Get.p2.year_of_birth => :year_of_birth)
q4 => Dict()
```

Ambs

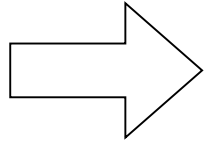
```
q1 => Set()
q2 => Set()
q3 => Set()
q4 => Set()
```



[q6, q5, q4, q3, q2, q4_bindings, q1]

for q in [q6, q5, q4, q3, q2, q1]
collect_references(q)
end



Requests

q1 => [Get.person_id (q4)]
q2 => [Get.person_id (q3)]
q3 => []
q4 => []
q5 => []
q6 => []

Environment

q1 => nothing
q2 => q4_bindings
q3 => q4_bindings
q4 => nothing
q5 => nothing
q6 => nothing

Bindings

q1 => Dict()
q2 => Dict(:person_id => (q1, Get.person_id .+ 1))
q3 => Dict(:person_id => (q1, Get.person_id .+ 1))
q4 => Dict()
q5 => Dict()
q6 => Dict()

Clauses

q1 => (SELECT person_id AS person_id FROM person) AS q1
q2 => (SELECT person_id AS person_id FROM visit_occurrence) AS q2
q3 =>
q4 =>
q5 =>
q6 =>

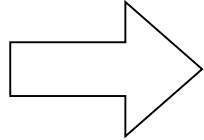
Repl

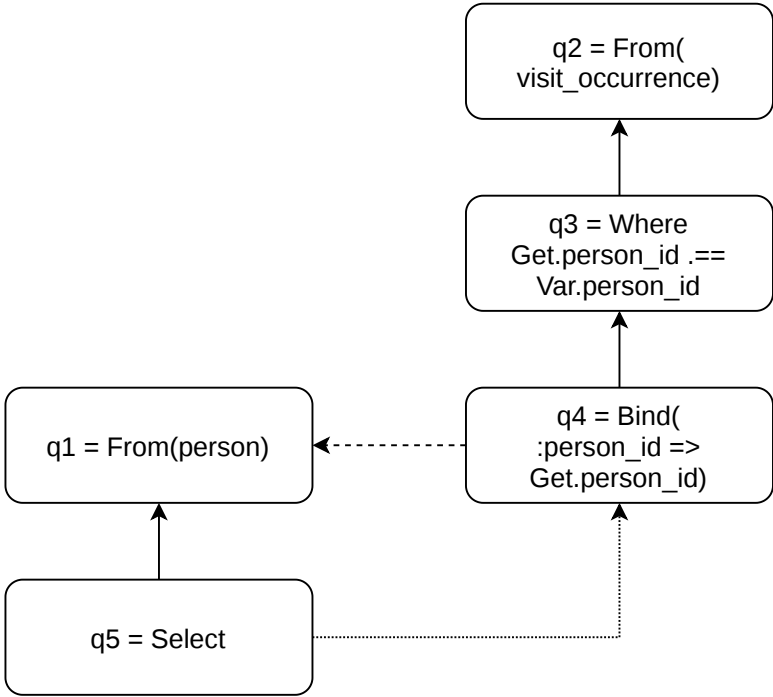
q1 => Dict(Get.person_id (q4) => :person_id)
q2 => Dict(Get.person_id (q3) => :person_id)
q3 => Dict()
q4 => Dict()
q5 => Dict()
q6 => Dict()

Vars

q4_bindings => Dict(:person_id => (q1.person_id + 1))

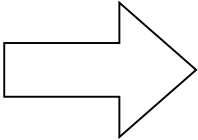
for q in [q1, q4_bindings, q2, q3, q4, q5, q6]
build_clauses(q)
end





[q5, q4, q3, q2, q1]

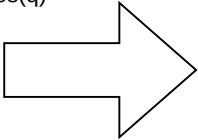
```
for q in [q5, q4, q3, q2, q1]
  collect_references(q)
end
```



Requests

```
q1 => [Get.person_id (q4)]
q2 => [Get.person_id (q3)]
q3 => []
q4 => []
q5 => []
q6 => []
```

```
for q in [q1, q2, q3, q4, q5]
  build_clauses(q)
end
```

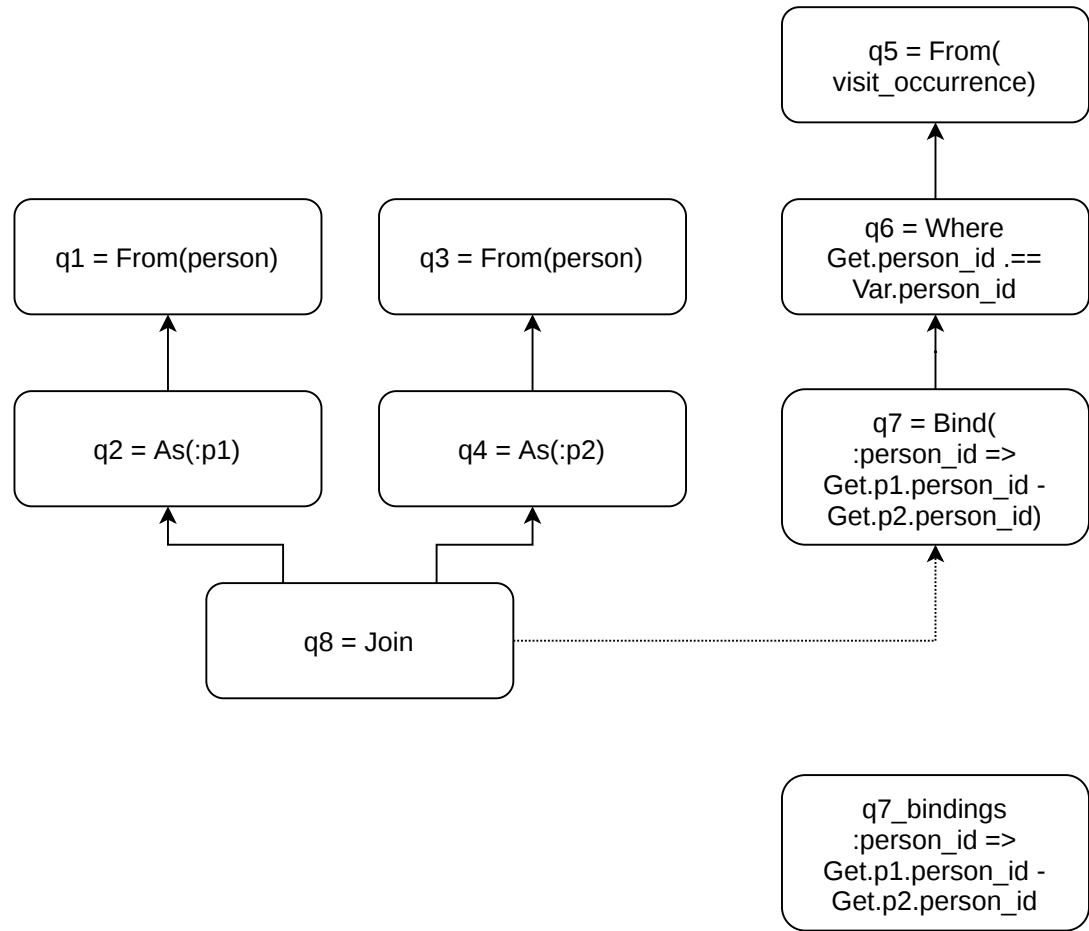


Clauses

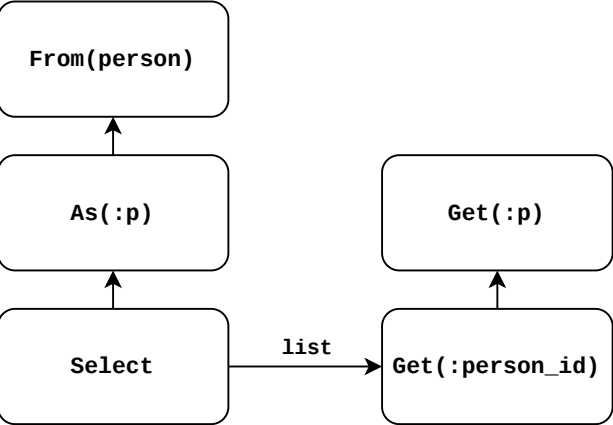
```
q1 => SELECT person_id AS person_id FROM person
q2 => SELECT person_id AS person_id FROM visit_occurrence
q3 =>
q4 =>
q5 =>
q6 =>
```

Repl

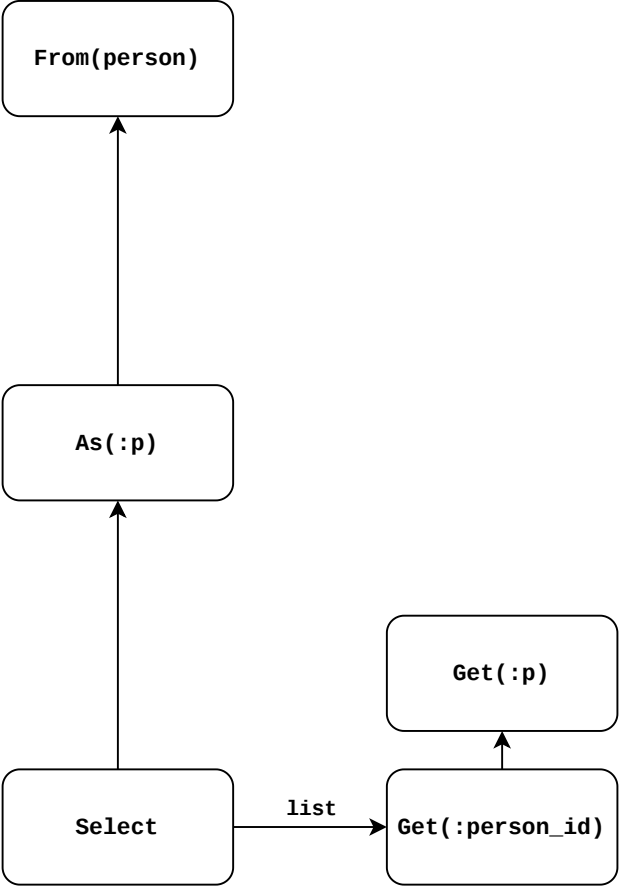
```
q1 => Dict(Get.person_id (q4) => :person_id)
q2 => Dict(Get.person_id (q3) => :person_id)
q3 => Dict()
q4 => Dict()
q5 => Dict()
q6 => Dict()
```



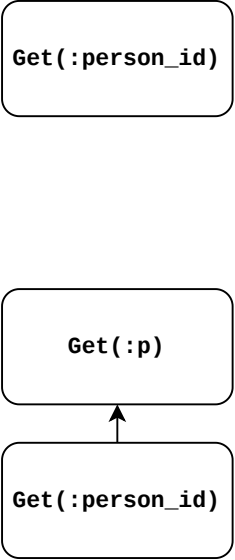
```
From(person) |>  
As(:p) |>  
Select(Get.p.person_id)
```



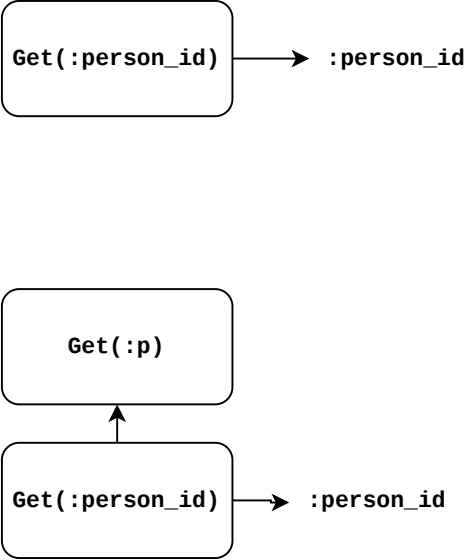
Resolve



refs



repl



New Resolve

