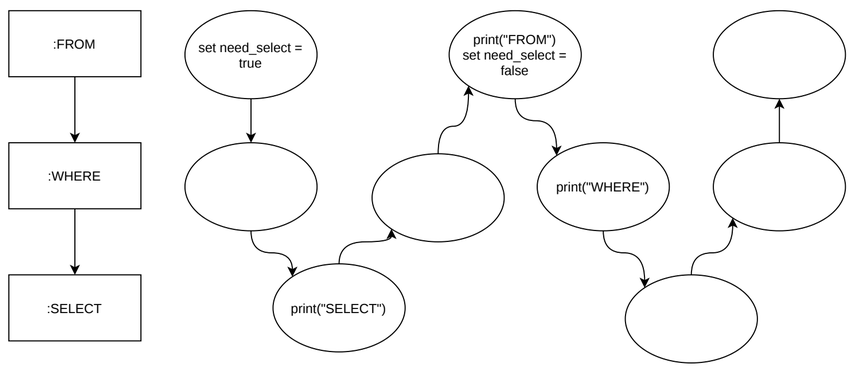
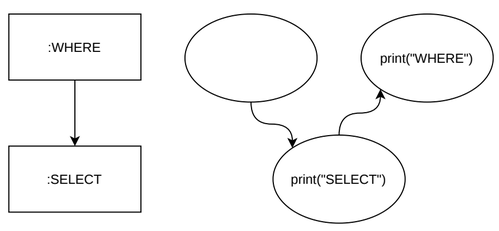
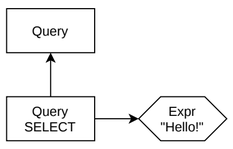

:FROM
:WHERE
:SELECT
set need_select = true
print("SELECT")
print("FROM")
set need_select = false
print("WHERE")


:WHERE
:SELECT
print("SELECT")
print("WHERE")

```
SELECT "Hello World!"
```

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

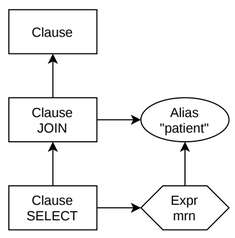



```
SELECT p.mrn FROM patient p
```

```
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)
```



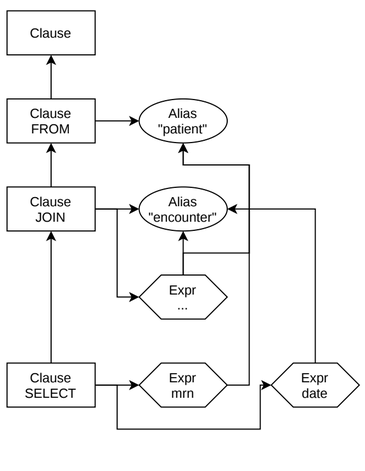

```
SELECT p.mrn, e.date
FROM patient p
JOIN encounter e ON (p.id = e.patient_id)
```

```
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)
```

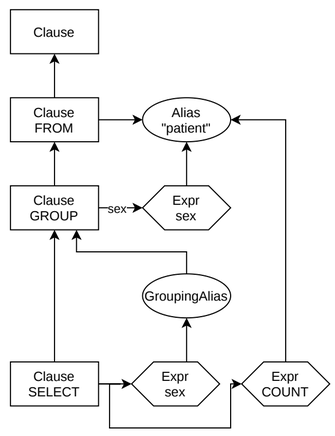
```
p = sql_alias("patient")
e = sql_alias("encounter")
sql_from(p) |> sql_join(e, p.id .== e.patient_id) |> sql_select(p.mrn) |> sql_select(e.date)
```




```
SELECT p.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(e, 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))
```

```
SELECT p.mrn
FROM patient p
WHERE p.sex = 'male'
```

```
p = From("patient")
w = Where(p, p.sex .== "male")
Select(w, p.mrn)
```

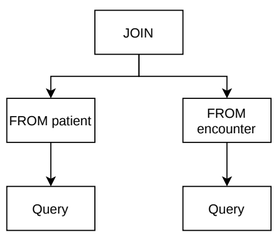
```
p = From("patient", columns=["mrn", "sex"])
w = Where(p, Ref(1, 2) .== "male", select=[Ref(1,1)])
Select(w, select=[Ref(1,1)])
```

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

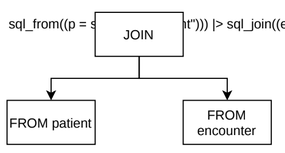
```
p = From("patient")
e = From("encounter")
g = Group(e, patient_id = e.patient_id)
gs = Select(g, patient_id = g.patient_id, n = Count(e))
j = LeftJoin(p, gs, p.id .== gs.patient_id)
Select(j, p.mrn, Coalesce(gs.n, 0))
```

```
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)
```

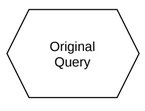
```
p = From("patient")
e = From("encounter")
j = Join(p, e, p.id .== e.patient_id)
Select(j, p.mrn, e.date)
```




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



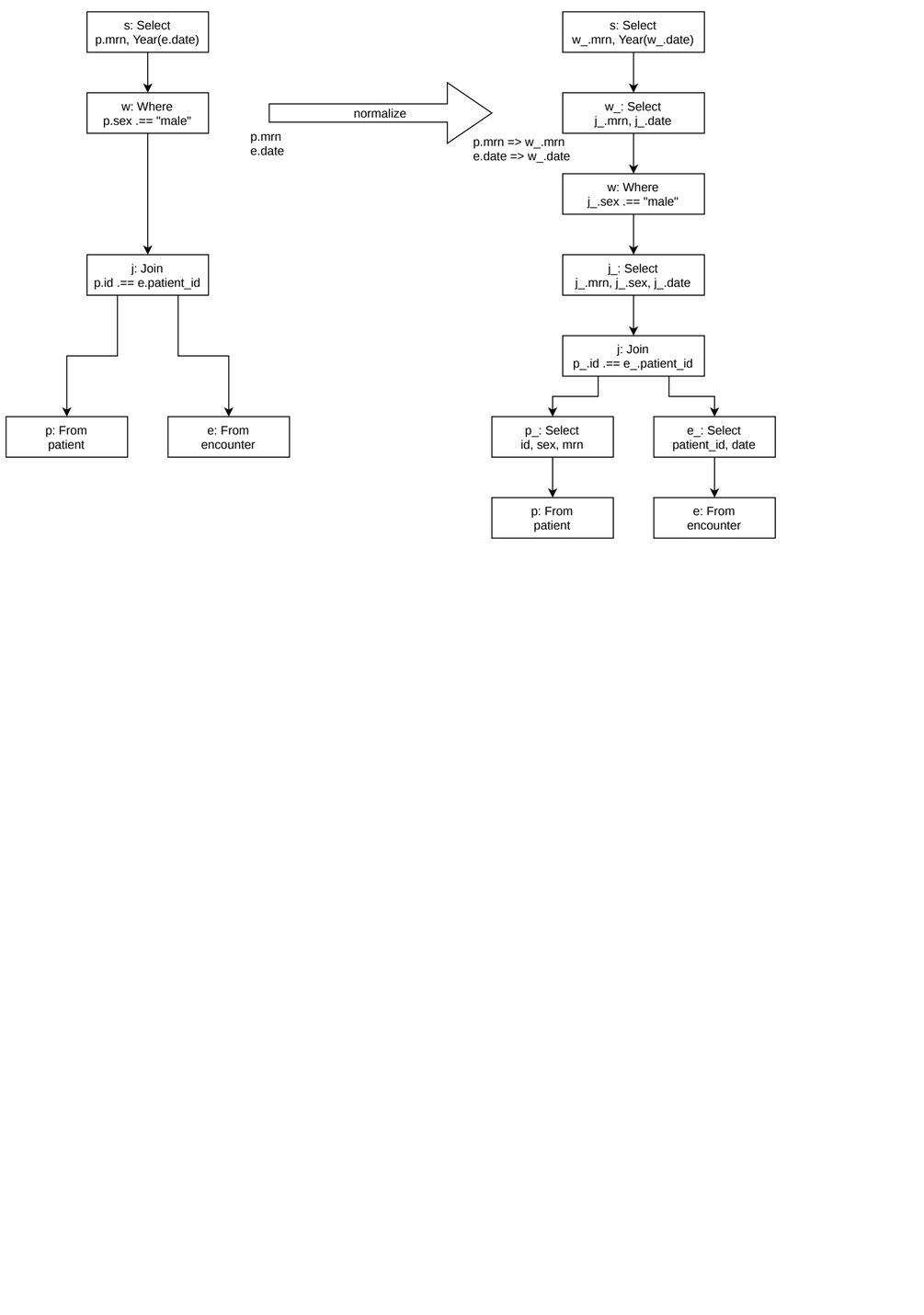

```
p = From("patient")
e = From("encounter")
g = Group(e, patient_id = e.patient_id, summarize=(; n = Count(e)))
j = LeftJoin(p, g, p.id .== g.patient_id)
Select(j, p.mrn, Coalesce(g.n, 0))
```



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

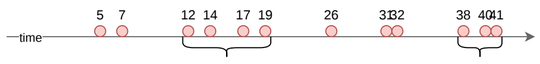
```
p = From(patient)
e = From(encounter)
j = Join(p, e, p.id .== e.patient_id)
w = Where(j, p.sex .== "male")
s = Select(w, mrn = p.mrn, year = Year(e.date))
```

```
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.

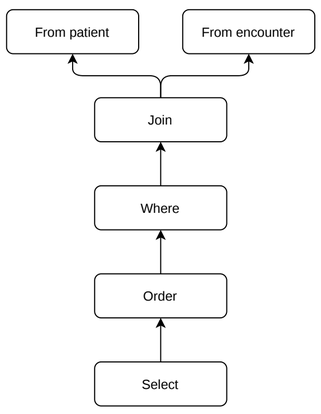


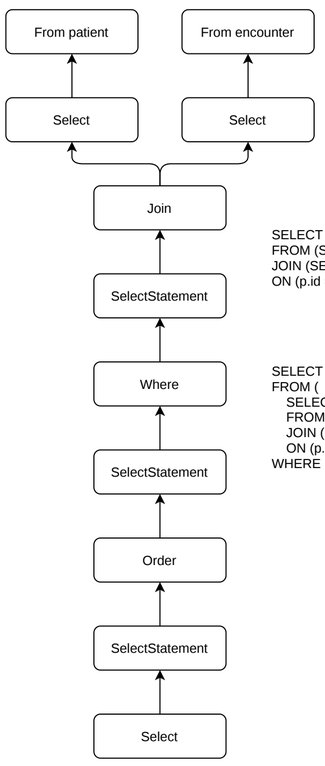
5	5	12	12	12	12	26	31	31	38	38	38												
0	2	0	2	5	7	0	0	1	0	2	3												

5		12		12		26	31		38														
2		2		7		0	1		3														

				12					38														
				7					3														

```
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
```

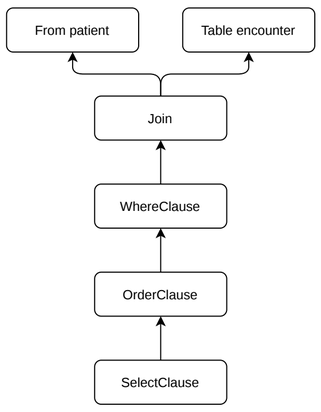


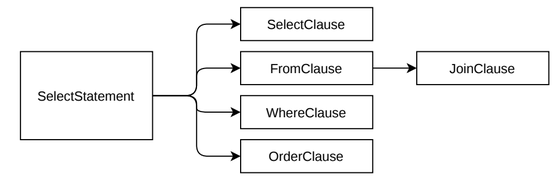


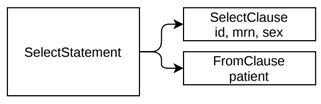

```
SELECT p.id, p.mrn, p.sex FROM patient p
    SELECT p.id, p.mrn FROM patient p
SELECT e.patient_id, e.date FROM encounter e
```

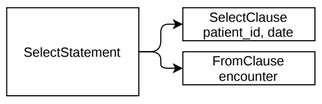
```
SELECT p.mrn, p.sex, e.date
FROM (SELECT p.id, p.mrn, p.sex FROM patient p) p
JOIN (SELECT e.patient_id, e.date FROM encounter e) e
ON (p.id = e.patient_id)
```

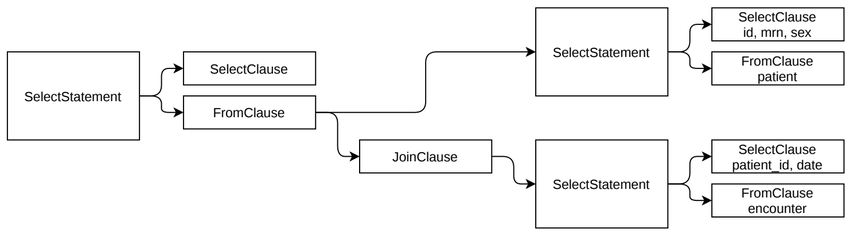
```
SELECT p.mrn, e.date
FROM (
   SELECT p.mrn, p.sex, e.date
   FROM (SELECT p.id, p.mrn, p.sex FROM patient p) p
   JOIN (SELECT e.patient_id, e.date FROM encounter e) e
   ON (p.id = e.patient_id)) p
WHERE p.sex = 'male'
```

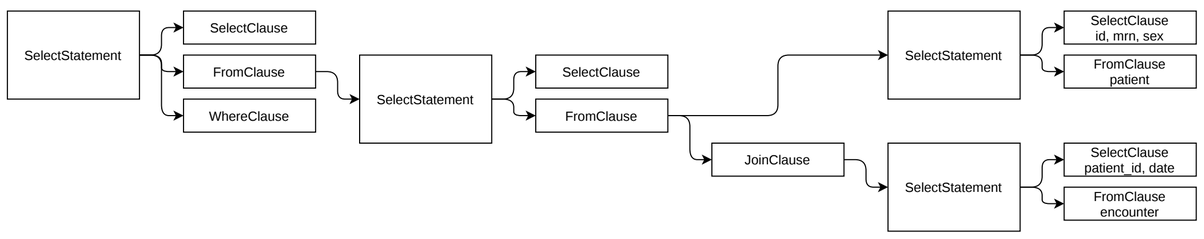


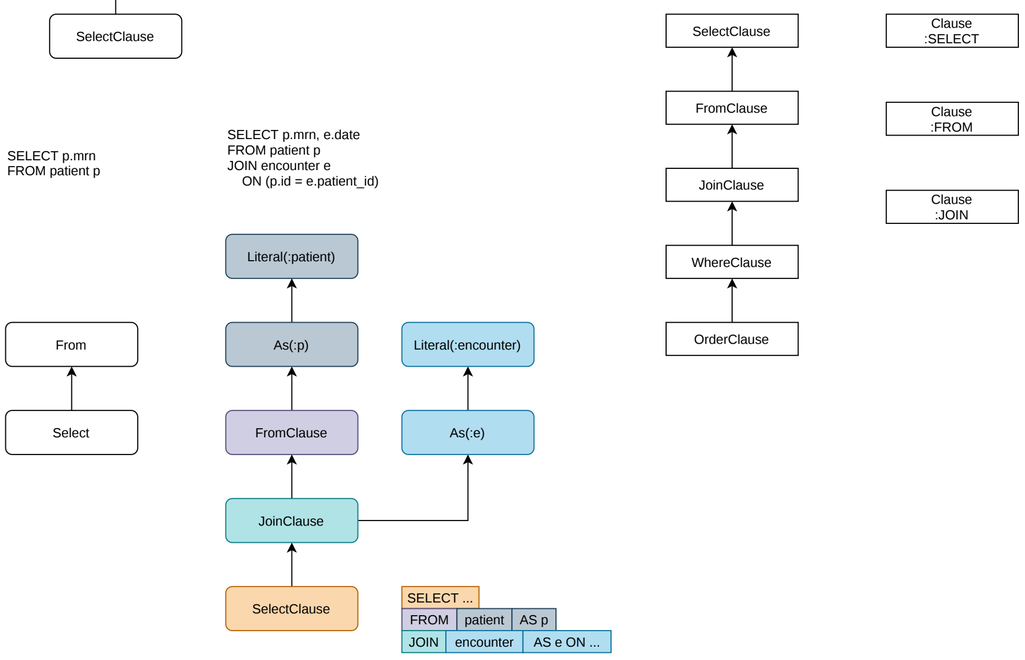


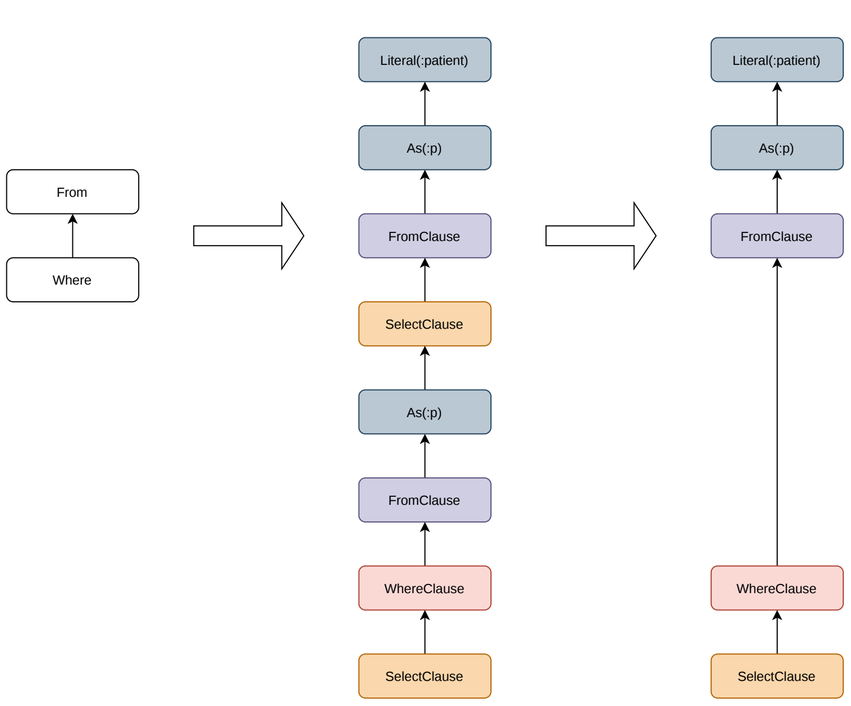


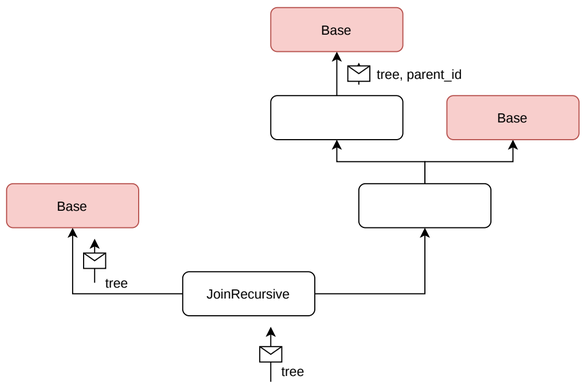




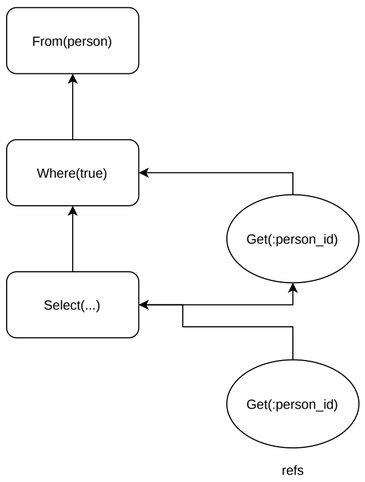


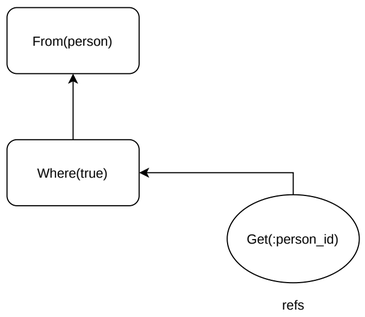


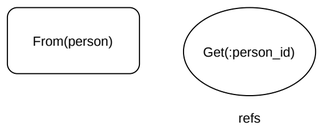


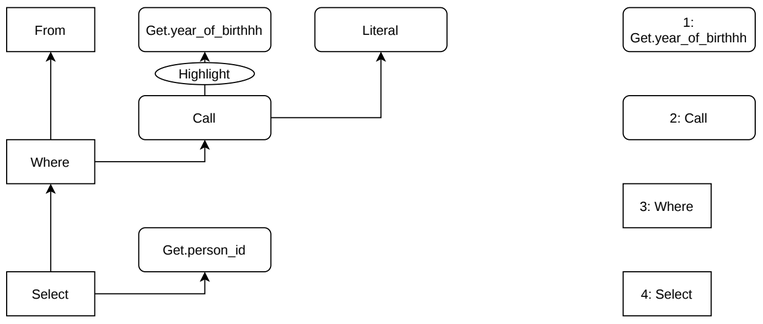



```
WITH RECURSIVE X AS (
      SELECT 1 AS N
   UNION ALL
      SELECT ...
      ...
         FROM X
      ...
         FROM X)
```


From(person)
Where(true)
Get(:person_id)
Select(...)
Get(:person_id)
refs


From(person)
Where(true)
Get(:person_id)
refs


From(person)
Get(:person_id)
refs

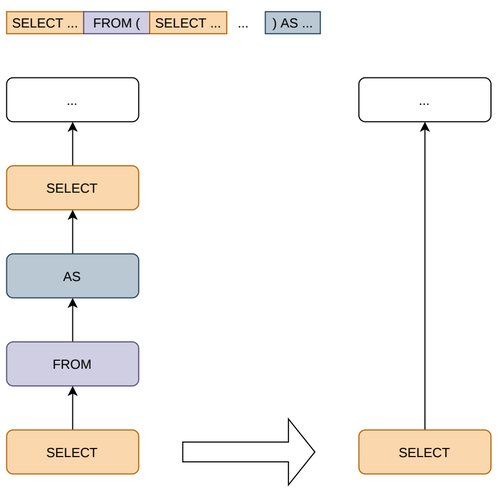


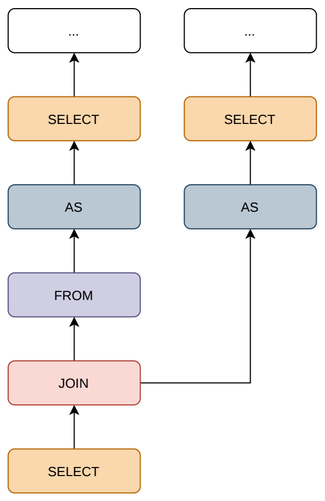
1: Get.year_of_birthhh

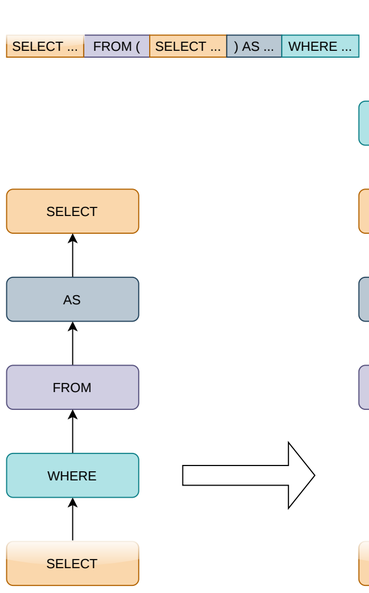
2: Call

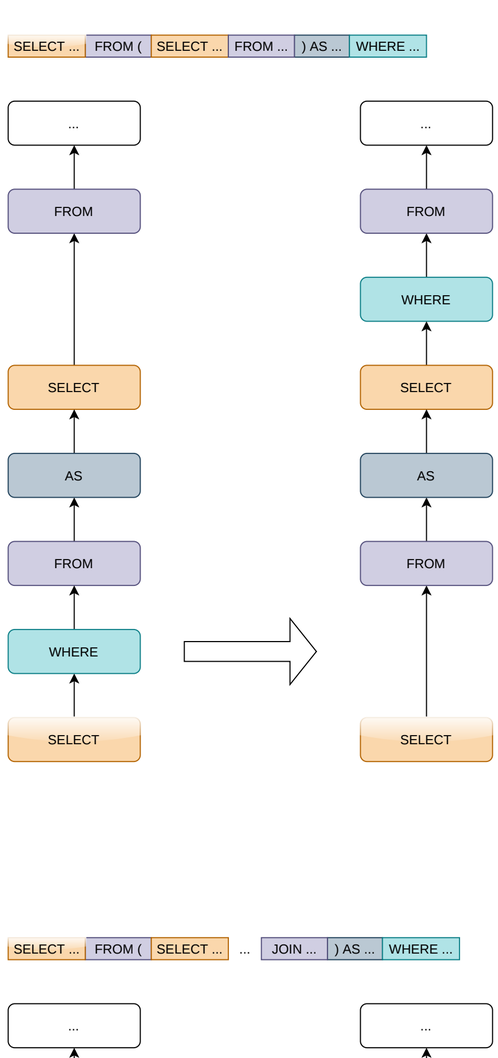
3: Where

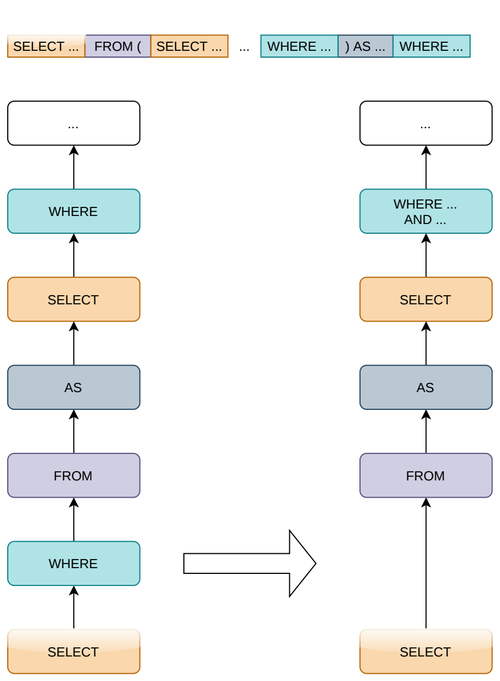
4: Select

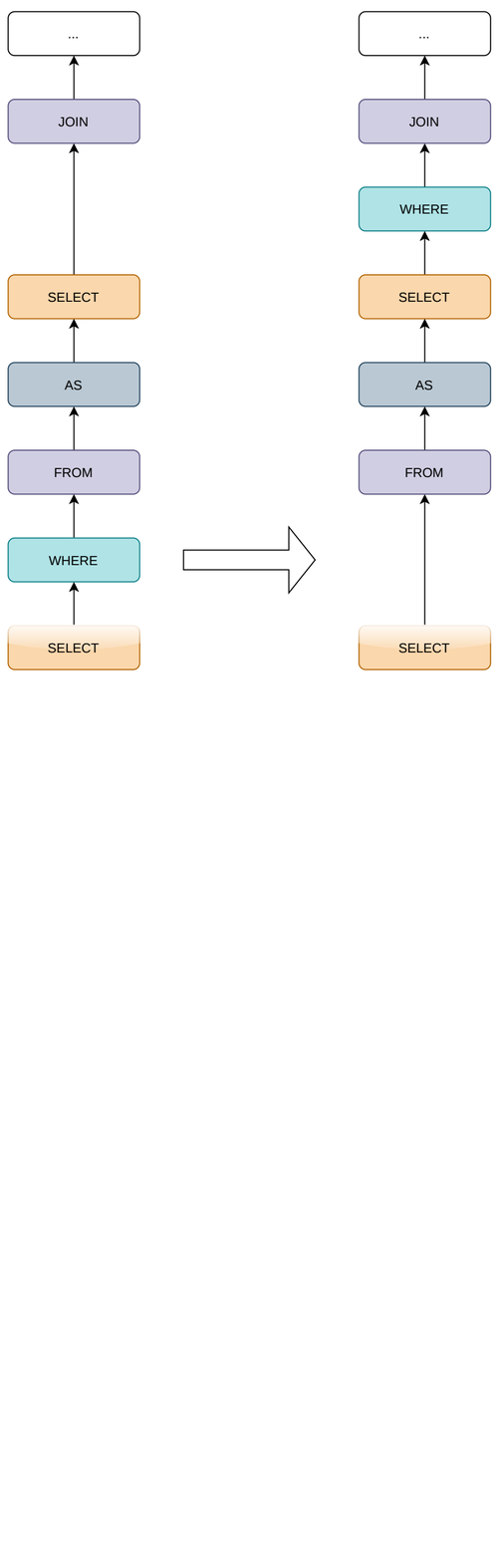

SELECT ... FROM (SELECT ...) AS ...
...
SELECT
AS
FROM
WHERE
SELECT
...
SELECT
AS
FROM
SELECT

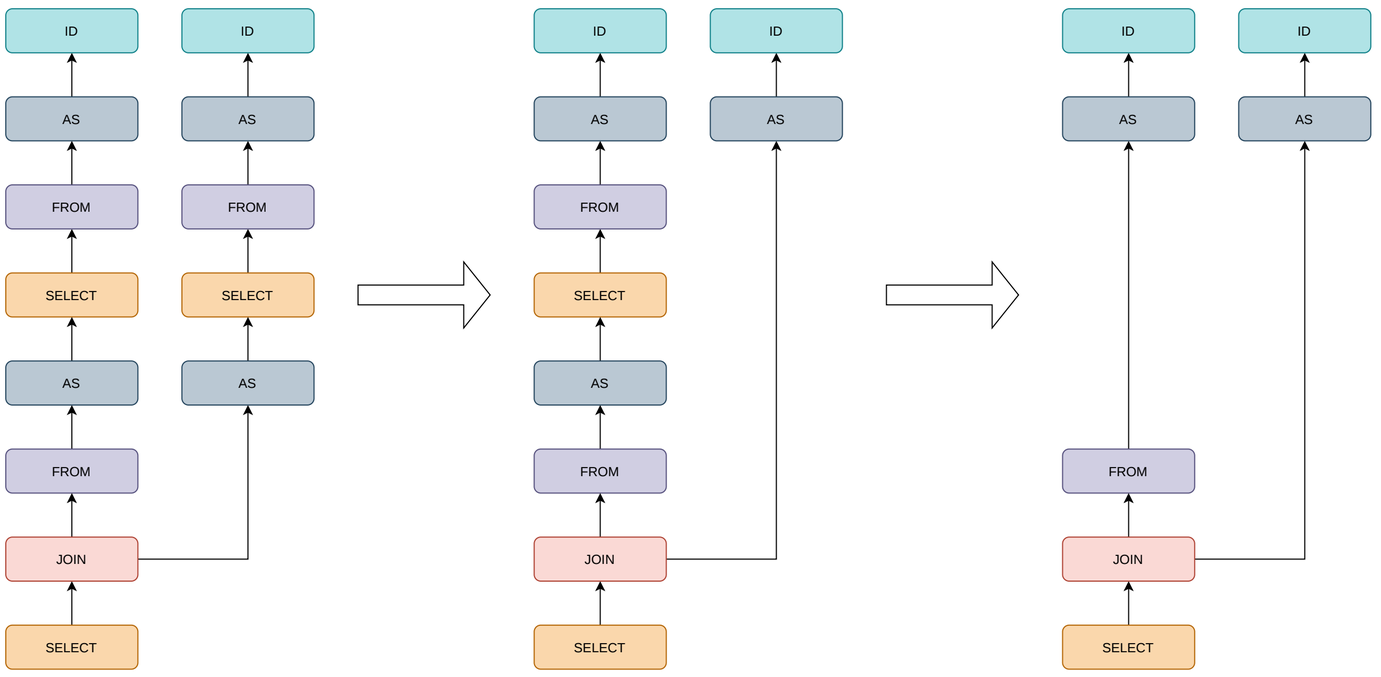

...
...
SELECT
SELECT
AS
AS
FROM
JOIN
SELECT

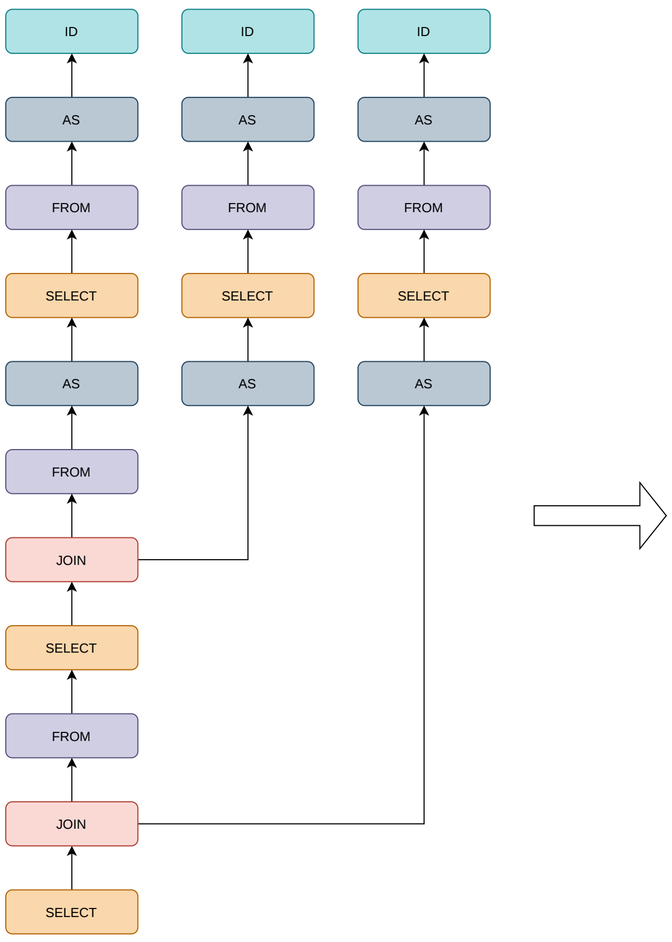

SELECT ... FROM (SELECT ...) AS ... WHERE ...
SELECT
AS
FROM
WHERE
SELECT
WHERE
SELECT
AS
FROM
SELECT

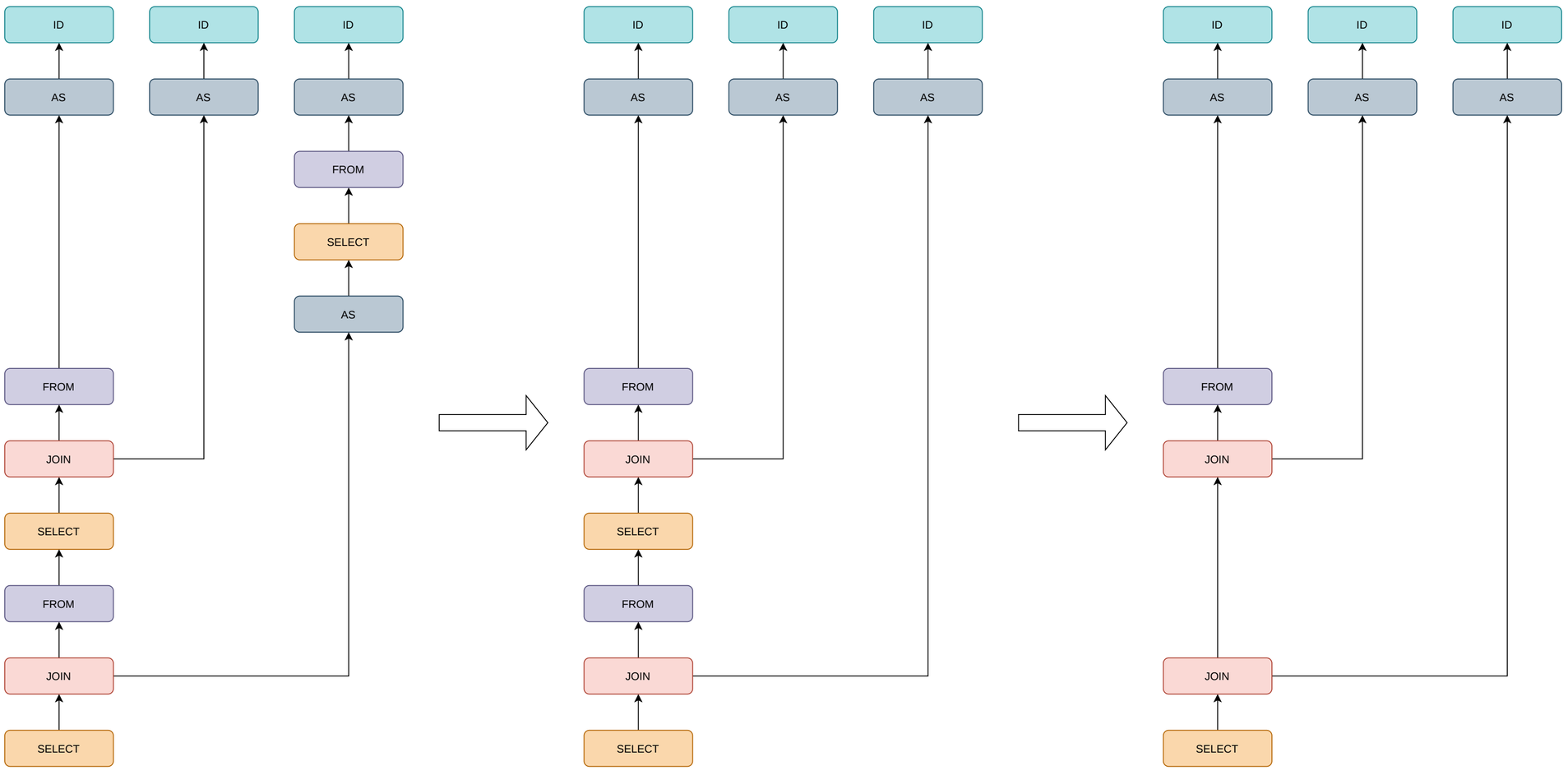

SELECT ... FROM (SELECT ... FROM ...) AS ... WHERE ...
...
FROM
SELECT
AS
FROM
WHERE
SELECT
...
FROM
WHERE
SELECT
AS
FROM
SELECT

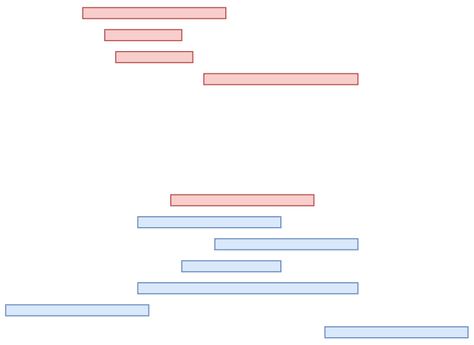

SELECT ... FROM (SELECT ... ... WHERE ...) AS ... WHERE ...
...
WHERE
SELECT
AS
FROM
WHERE
SELECT
...
WHERE ... AND ...
SELECT
AS
FROM
SELECT


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

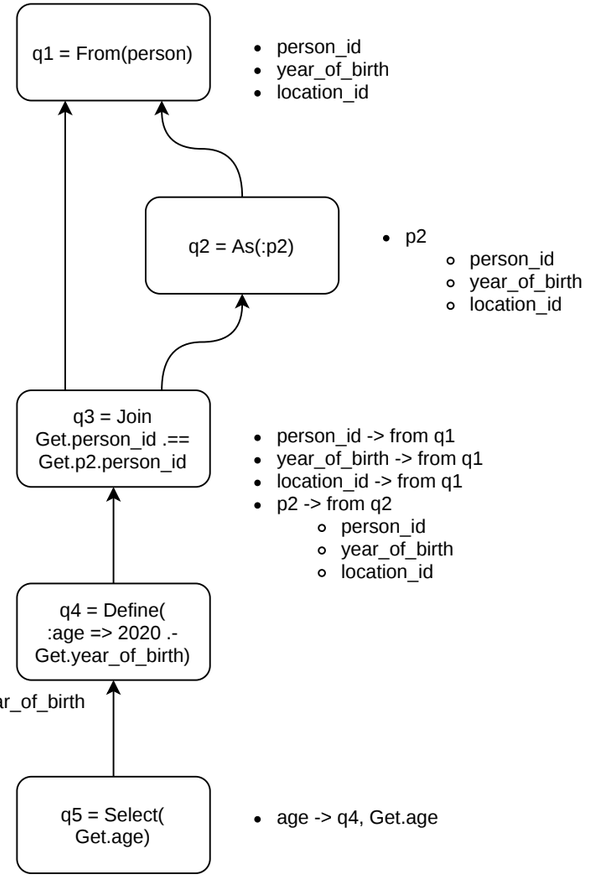

ID
ID
AS
AS
FROM
FROM
SELECT
SELECT
AS
AS
FROM
JOIN
SELECT
ID
ID
AS
AS
FROM
SELECT
AS
FROM
JOIN
SELECT
ID
ID
AS
AS
FROM
JOIN
SELECT


ID
ID
ID
AS
AS
AS
FROM
FROM
FROM
SELECT
SELECT
SELECT
AS
AS
AS
FROM
JOIN
SELECT
FROM
JOIN
SELECT


ID
ID
ID
ID
ID
ID
ID
ID
ID
AS
AS
AS
AS
AS
AS
AS
AS
AS
FROM
SELECT
AS
FROM
JOIN
SELECT
FROM
JOIN
SELECT
FROM
JOIN
SELECT



Generate a namespace for each node



- person_id
- year_of_birth
- location_id

- p2
  - person_id
  - year_of_birth
  - location_id

- person_id -> from q1
- year_of_birth -> from q1
- location_id -> from q1
- p2 -> from q2
  - person_id
  - year_of_birth
  - location_id

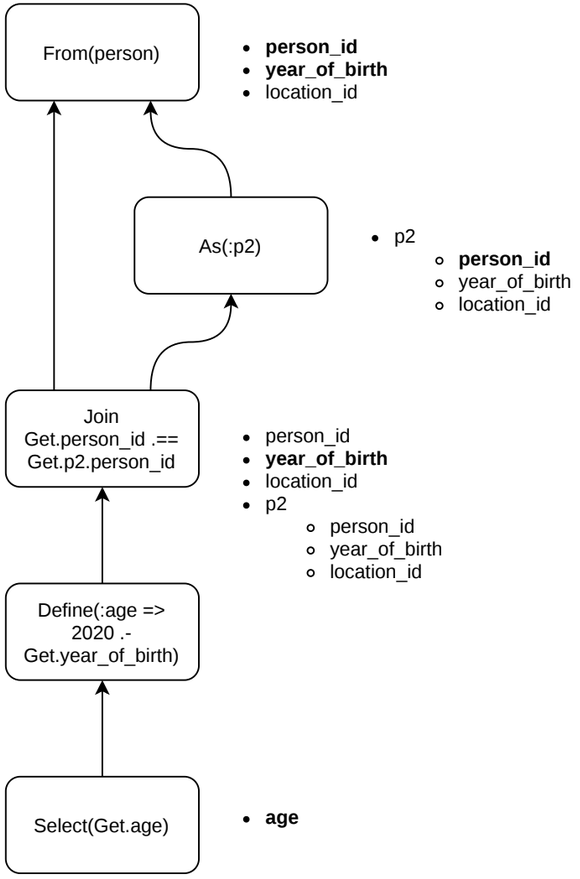
- person_id
- year_of_birth
- location_id
- age -> 2020 .- Get.year_of_birth
- p2
  - person_id
  - year_of_birth
  - location_id

- age -> q4, Get.age

A namespace must support:

- regular columns
- hierarchical aliases
- ambiguous columns
- groups
- nodes
- preserves order of columns

Generate an order for each node



- **person_id**
- **year_of_birth**
- location_id

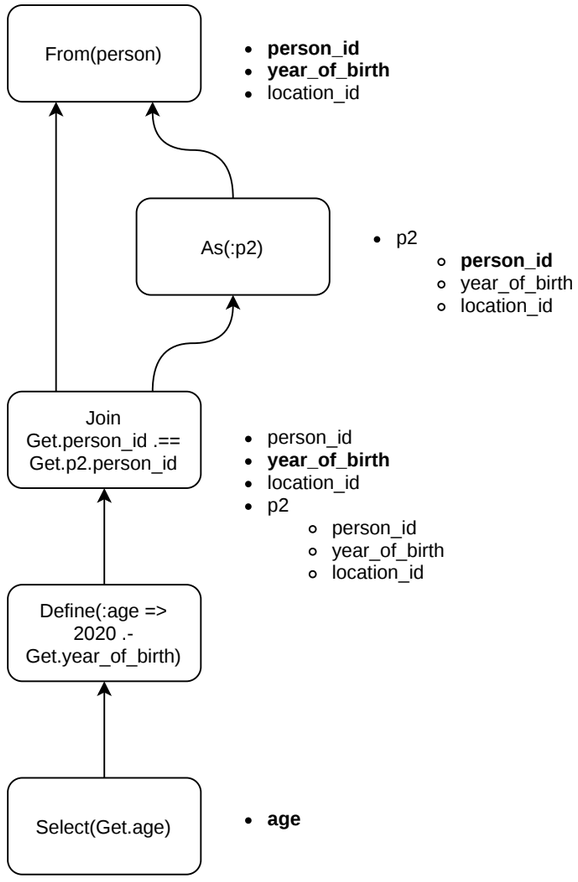
- p2
  - **person_id**
  - year_of_birth
  - location_id

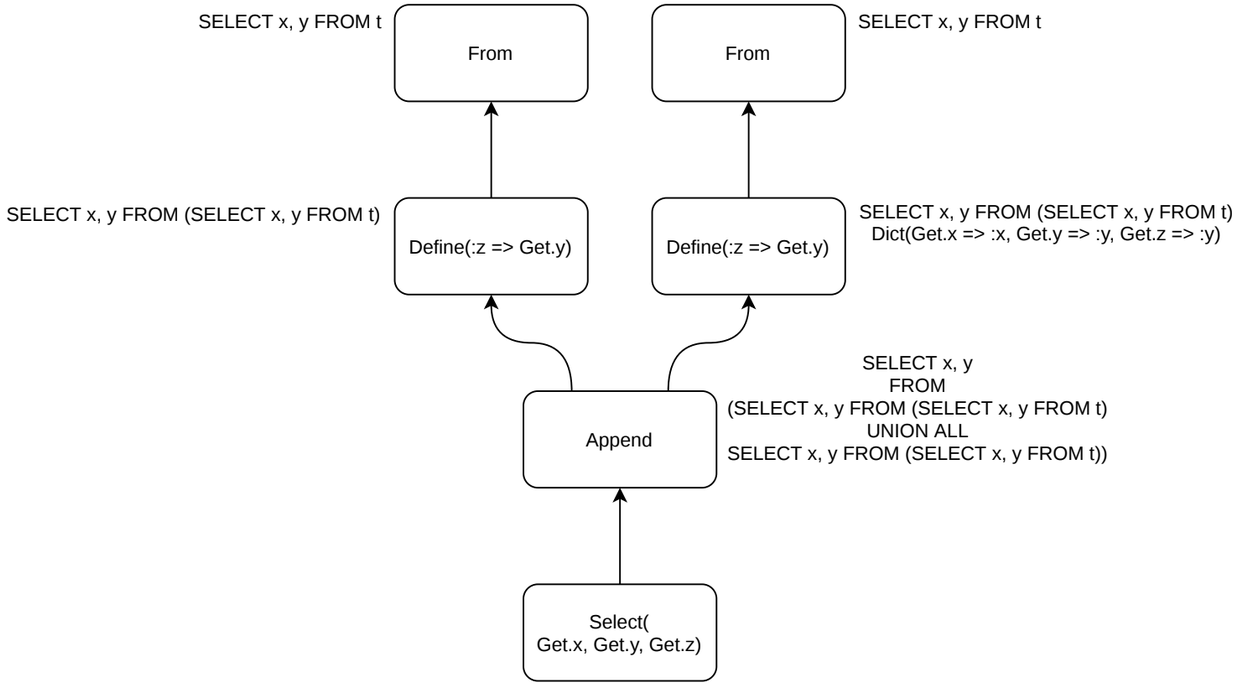
- person_id
- **year_of_birth**
- location_id
- p2
  - person_id
  - year_of_birth
  - location_id

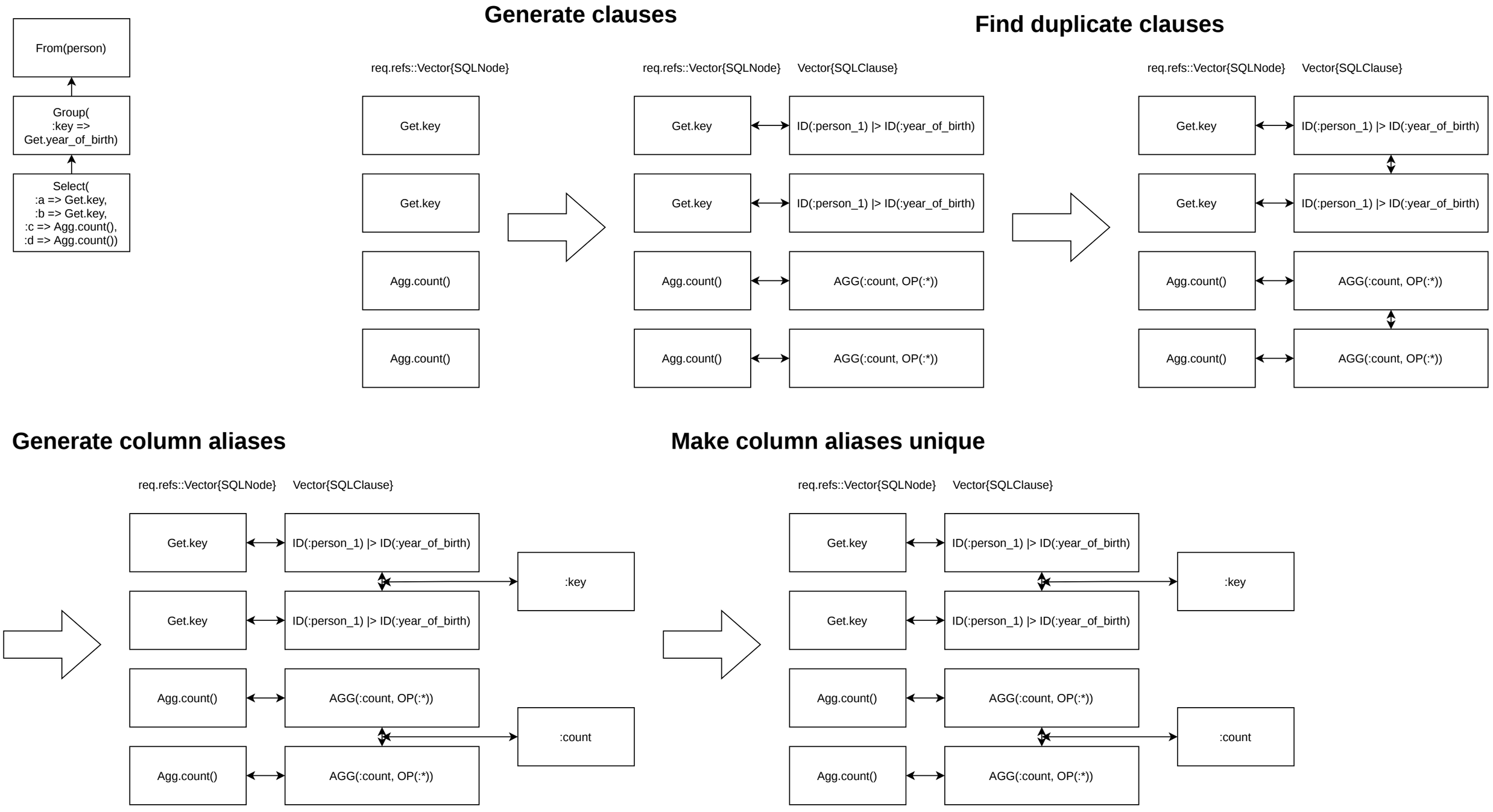
- person_id
- year_of_birth
- location_id
- **age**
- p2
  - person_id
  - year_of_birth
  - location_id

- **age**

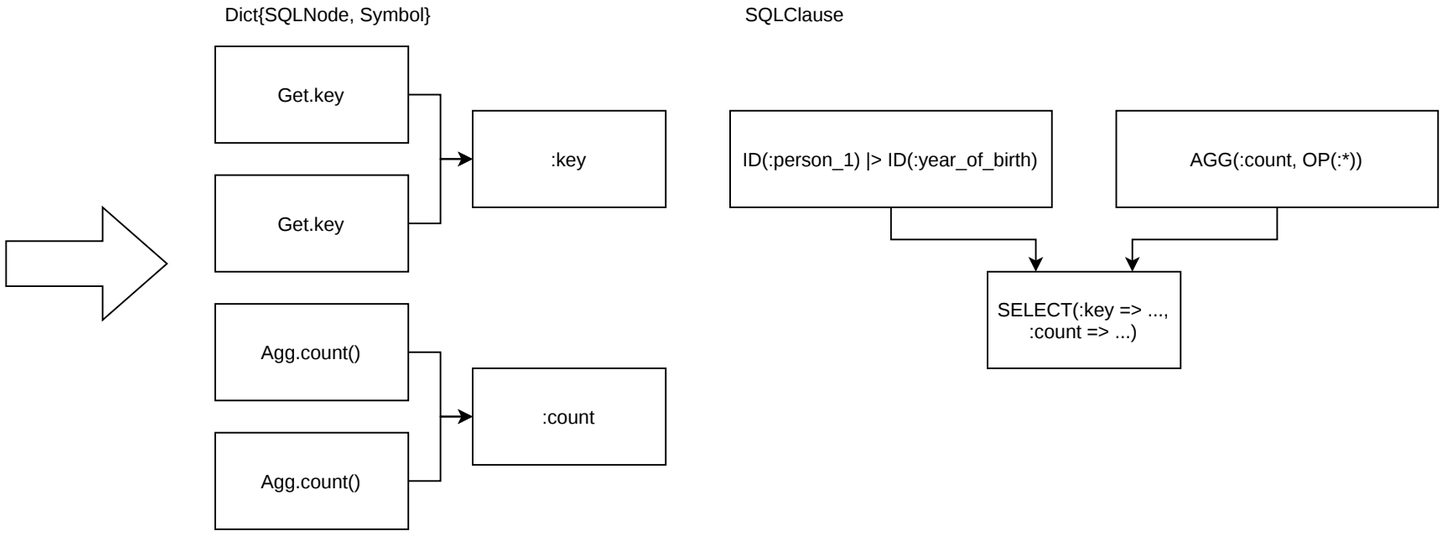
Generate an order for each node

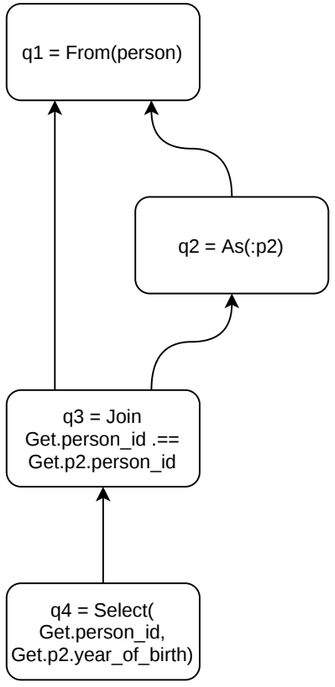





From(person)
Group(
:key =>
Get.year_of_birth)
Select(
:a => Get.key,
:b => Get.key,
:c => Agg.count(),
:d => Agg.count())
Generate clauses
req.refs::Vector{SQLNode}
Get.key
Get.key
Agg.count()
Agg.count()
req.refs::Vector{SQLNode}
Vector{SQLClause}
Get.key
ID(:person_1) |> ID(:year_of_birth)
Get.key
ID(:person_1) |> ID(:year_of_birth)
Agg.count()
AGG(:count, OP(:*))
Agg.count()
AGG(:count, OP(:*))
Find duplicate clauses
req.refs::Vector{SQLNode}
Vector{SQLClause}
Get.key
ID(:person_1) |> ID(:year_of_birth)
Get.key
ID(:person_1) |> ID(:year_of_birth)
Agg.count()
AGG(:count, OP(:*))
Agg.count()
AGG(:count, OP(:*))
Generate column aliases
req.refs::Vector{SQLNode}
Vector{SQLClause}
Get.key
ID(:person_1) |> ID(:year_of_birth)
Get.key
ID(:person_1) |> ID(:year_of_birth)
Agg.count()
AGG(:count, OP(:*))
Agg.count()
AGG(:count, OP(:*))
:key
:count
Make column aliases unique
req.refs::Vector{SQLNode}
Vector{SQLClause}
Get.key
ID(:person_1) |> ID(:year_of_birth)
Get.key
ID(:person_1) |> ID(:year_of_birth)
Agg.count()
AGG(:count, OP(:*))
Agg.count()
AGG(:count, OP(:*))
:key
:count

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

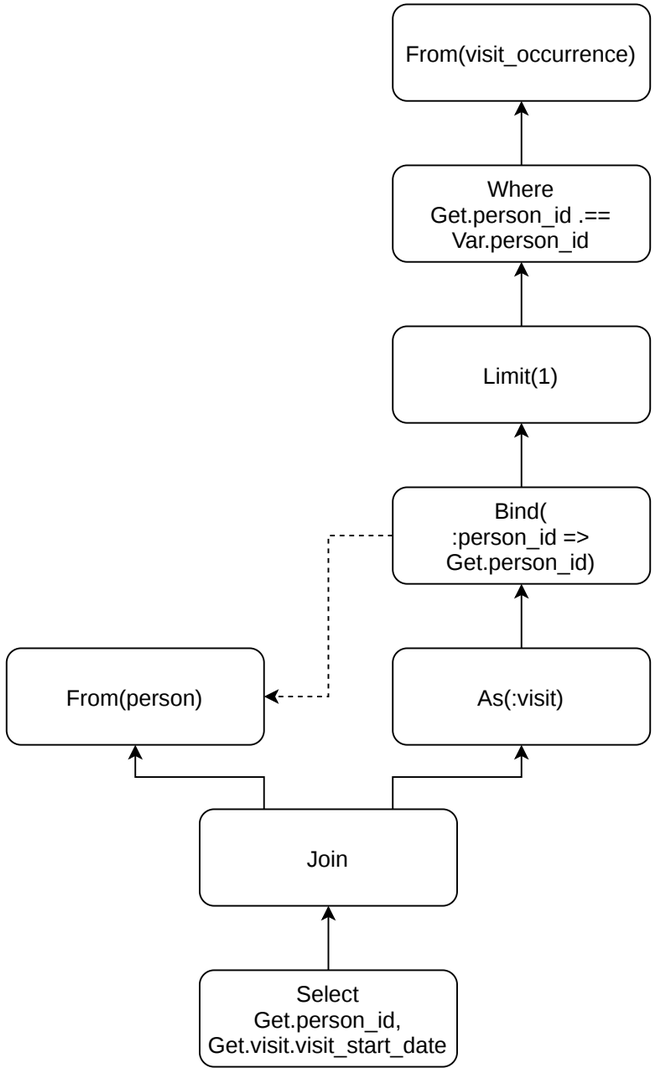
q1 => SELECT person_id AS person_id, year_of_birth AS year_of_birth FROM person
q2 => SELECT person_id AS person_id, year_of_birth AS year_of_birth FROM person
q3 => SELECT p1.person_id AS person_id, p2.year_of_birth AS year_of_birth FROM (clauses[q1]) AS p JOIN (clauses[q2]) ON p1.person_id = p2.person_id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

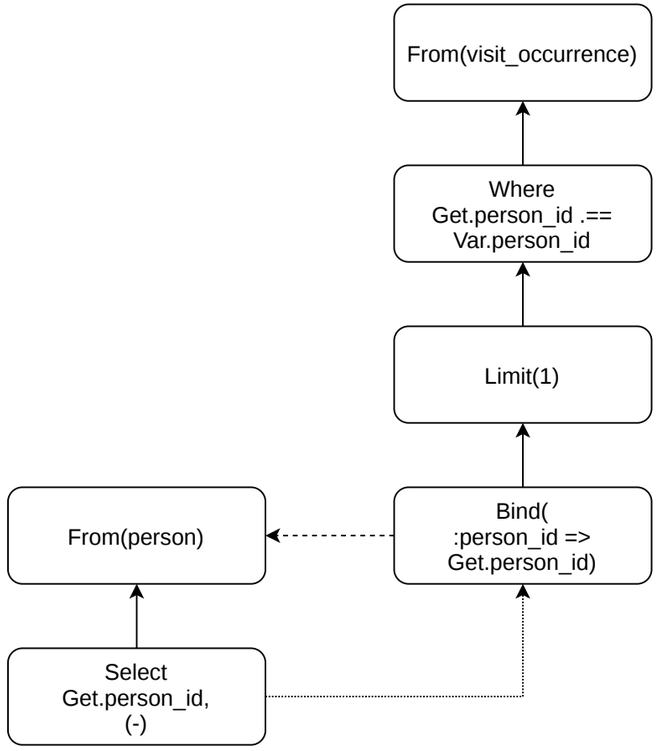
Repl

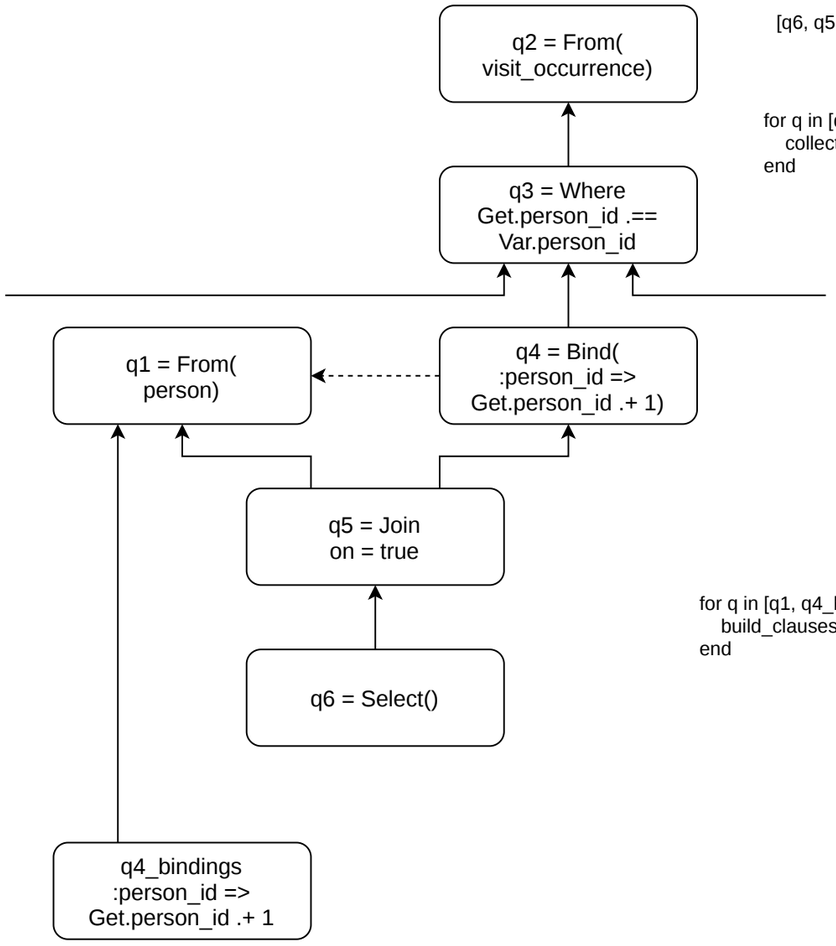
q1 => Dict(Get.person_id (q3) => :person_id, Get.person_id (q4) => :person_id, Get.person_id (q2) => :person_id, Get.year_of_birth (q2) => :year_of_birth)
q2 => Dict(Get.p2.person_id (q3) => :person_id, Get.p2.year_of_birth (q4) => :year_of_birth)
q3 => Dict(Get.person_id (q4) => :person_id, Get.p2.year_of_birth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

Ambs

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


From(visit_occurrence)
Where
Get.person_id .==
Var.person_id
Limit(1)
Bind(
:person_id =>
Get.person_id)
From(person)
As(:visit)
Join
Select
Get.person_id,
Get.visit.visit_start_date


From(visit_occurrence)
Where
Get.person_id .==
Var.person_id
Limit(1)
From(person)
Bind(
:person_id =>
Get.person_id)
Select
Get.person_id,
(-)



[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()

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

Clauses

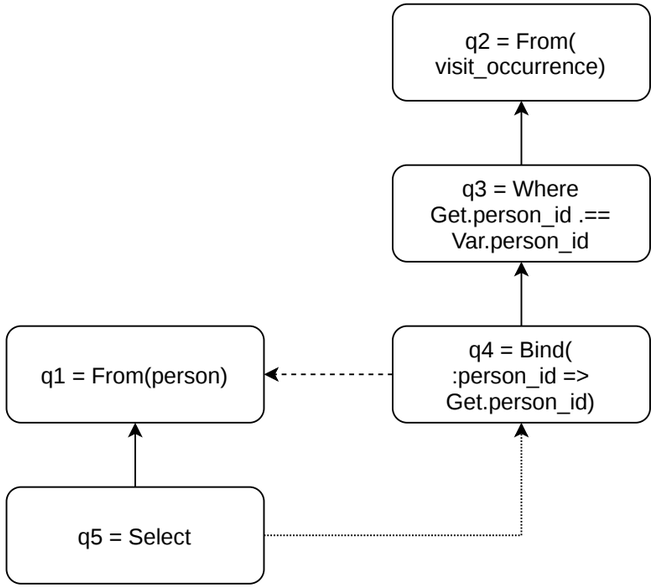
q1 => (SELECT person_id AS person_id FROM person) AS q1
q2 => (SELECT person_id AS person_id FROM visit_occurrence) AS q2
q3 =>
q4 =>
q5 =>
q6 =>

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))



[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 => []

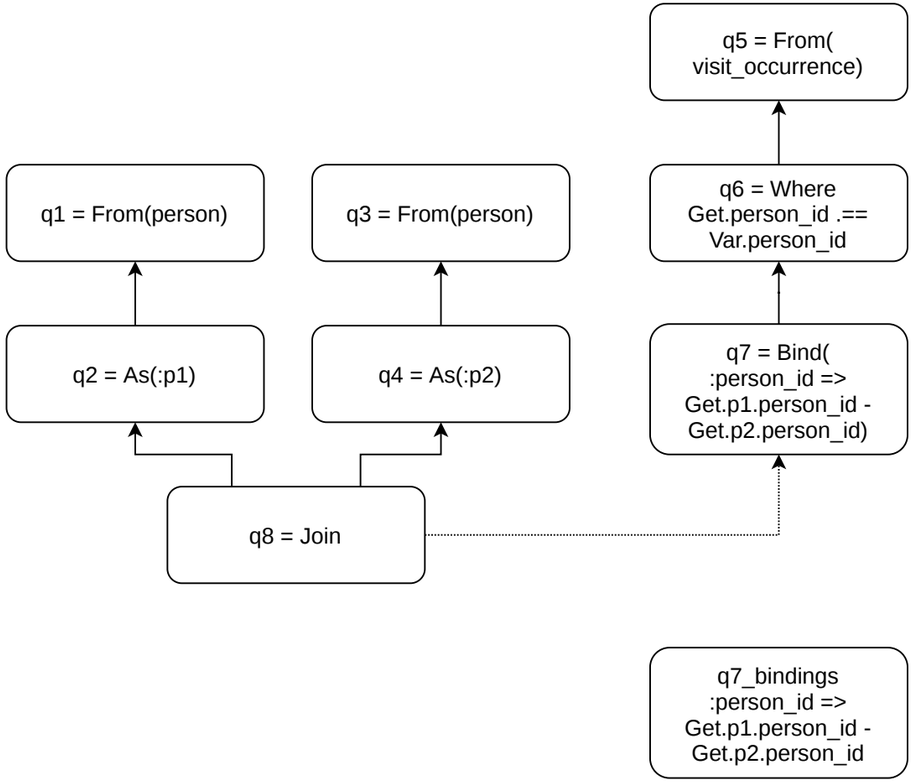
```
for q in [q1, q2, q3, q4, q5]
    build_clauses(q)
end
```

Clauses

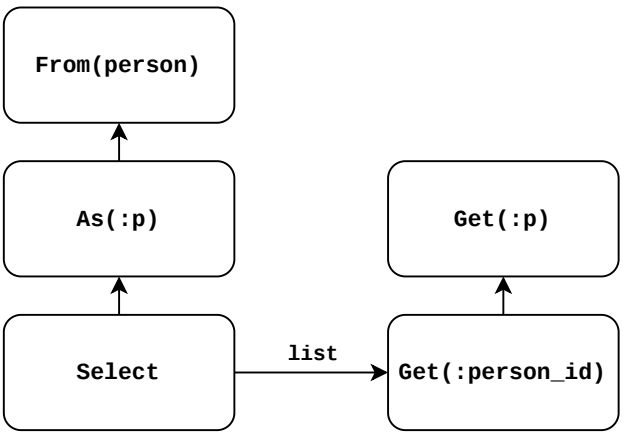
q1 => SELECT person_id AS person_id FROM person
q2 => SELECT person_id AS person_id FROM visit_occurrence
q3 =>
q4 =>
q5 =>
q6 =>

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()


q5 = From(
visit_occurrence)
q1 = From(person)
q3 = From(person)
q6 = Where
Get.person_id .==
Var.person_id
q2 = As(:p1)
q4 = As(:p2)
q7 = Bind(
:person_id =>
Get.p1.person_id -
Get.p2.person_id)
q8 = Join
q7_bindings
:person_id =>
Get.p1.person_id -
Get.p2.person_id

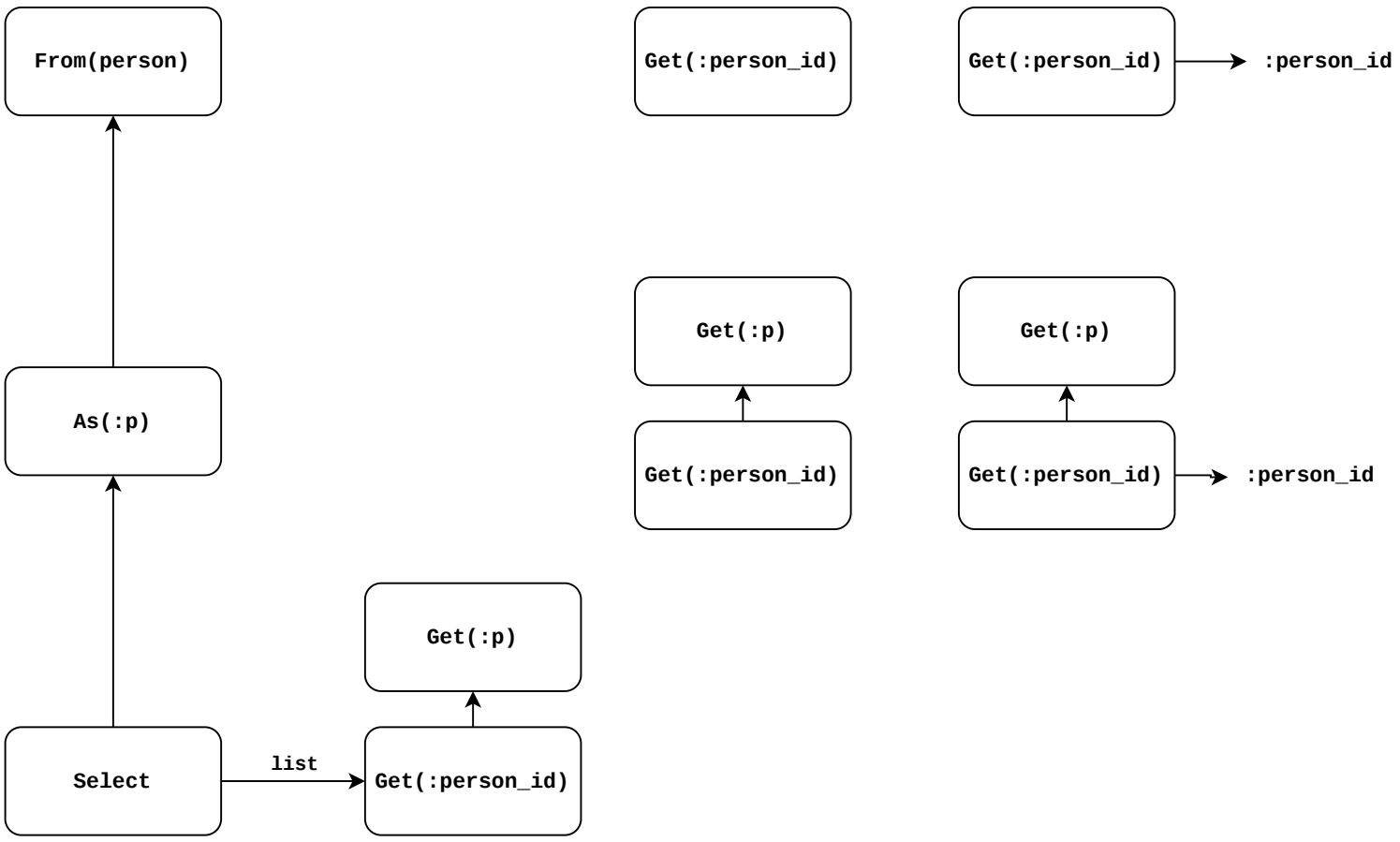
```
From(person) |>
As(:p) |>
Select(Get.p.person_id)
```



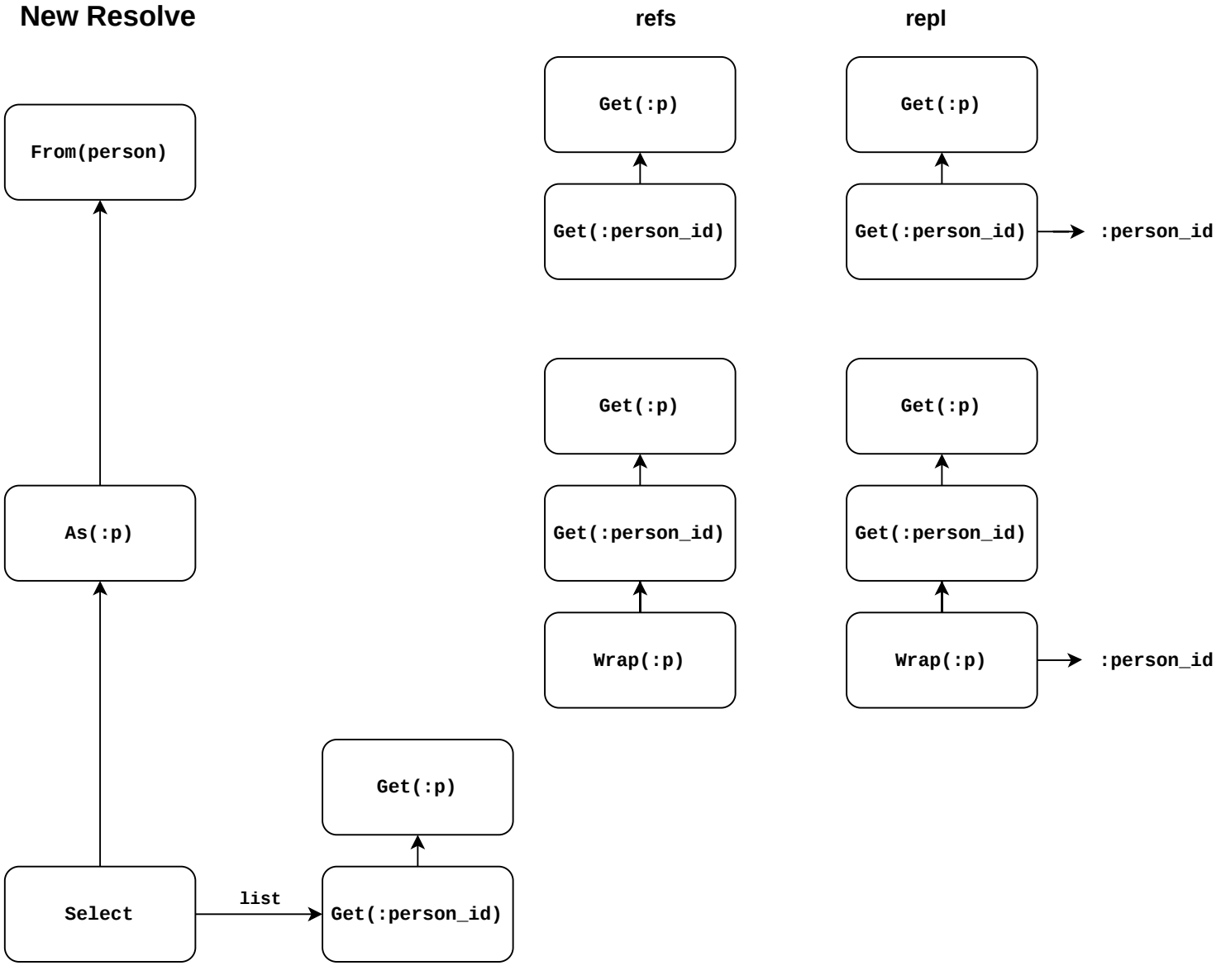
Resolve

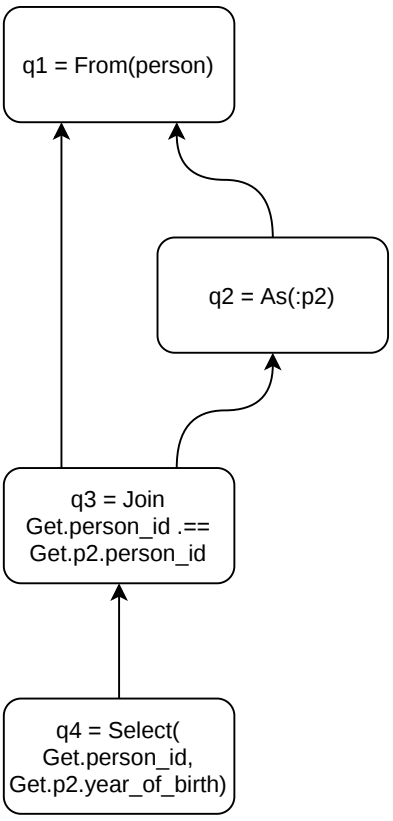
refs

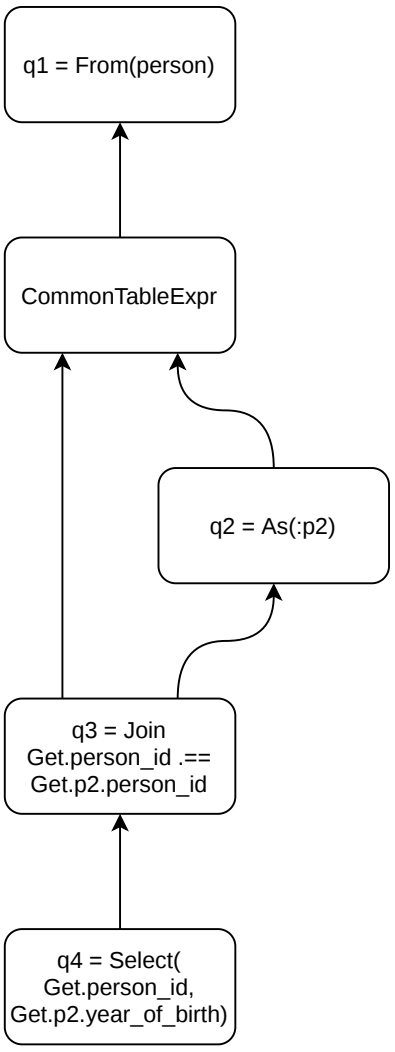
repl

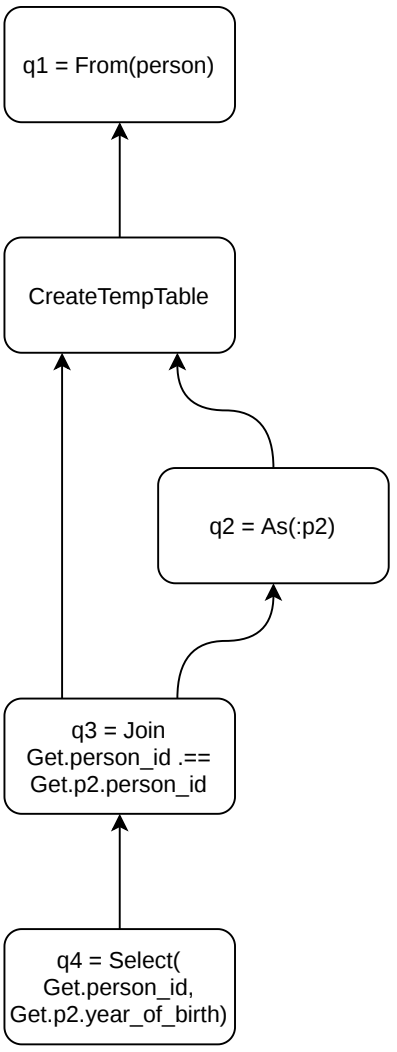


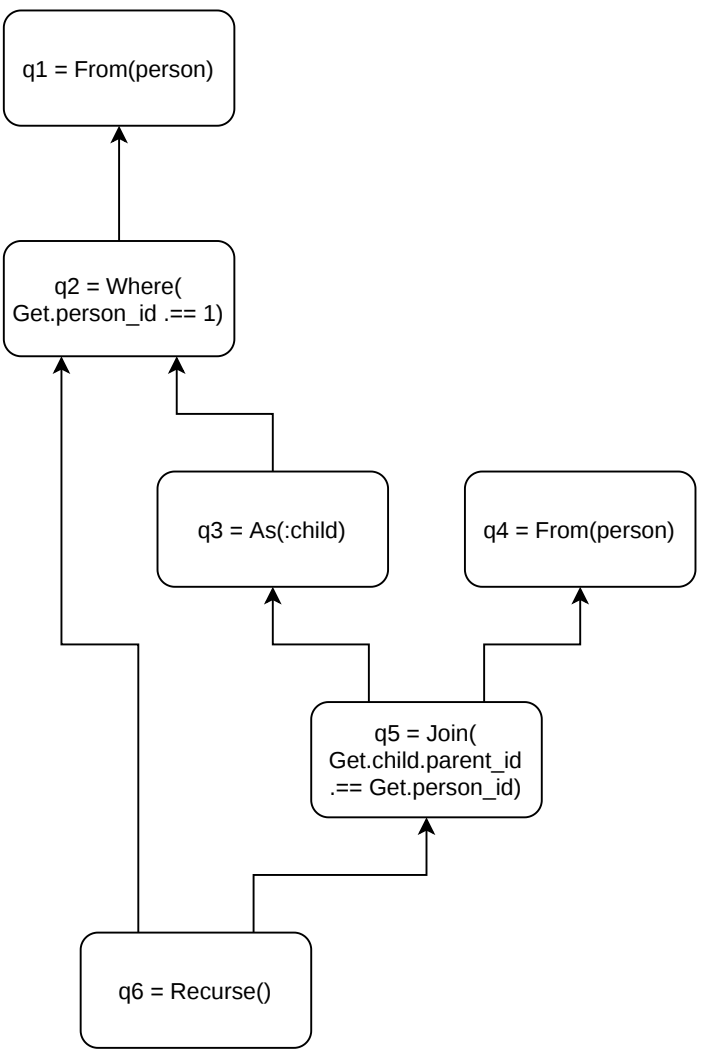
New Resolve

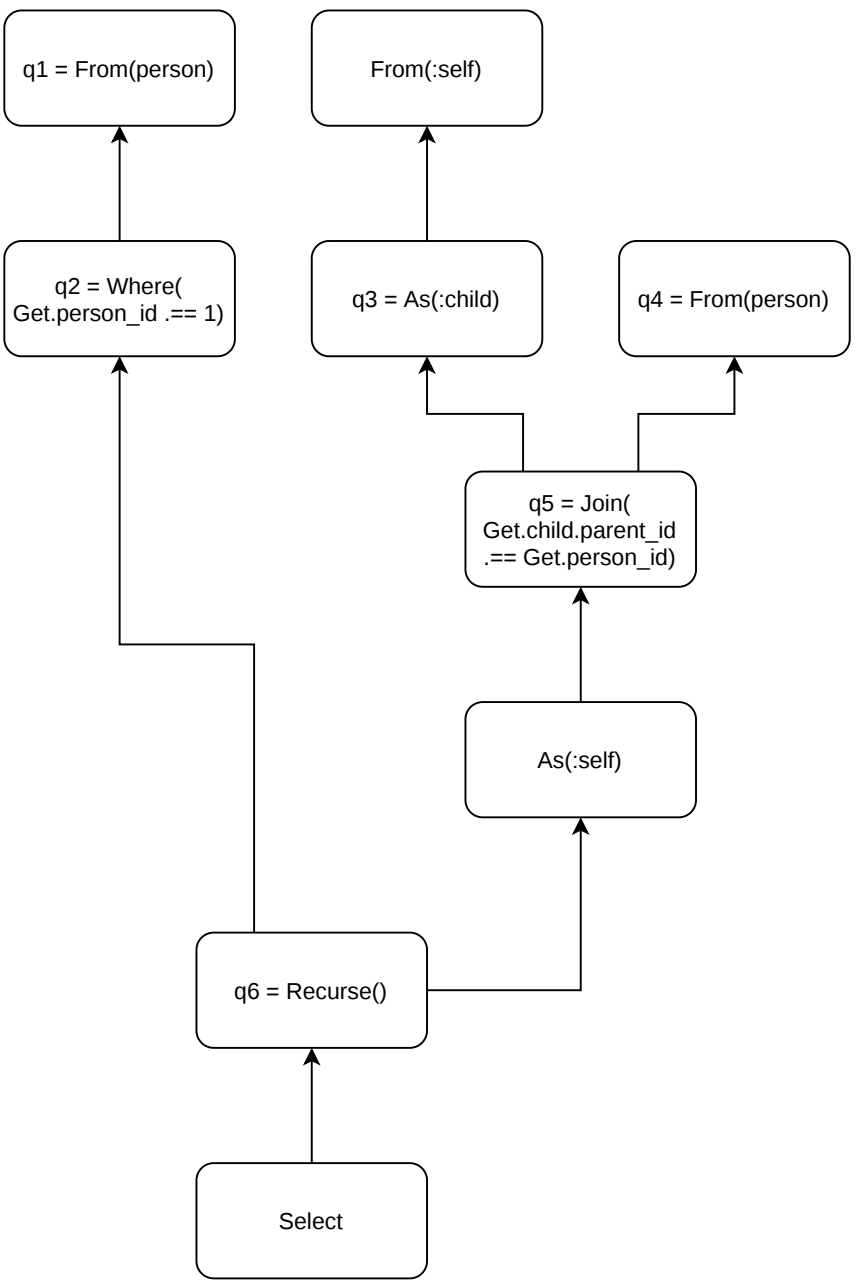

refs
repl
From(person)
As(:p)
Select
list
Get(:p)
Get(:person_id)
Get(:p)
Get(:person_id)
Wrap(:p)
Get(:p)
Get(:person_id)
:person_id
Get(:p)
Get(:person_id)
Wrap(:p)
:person_id

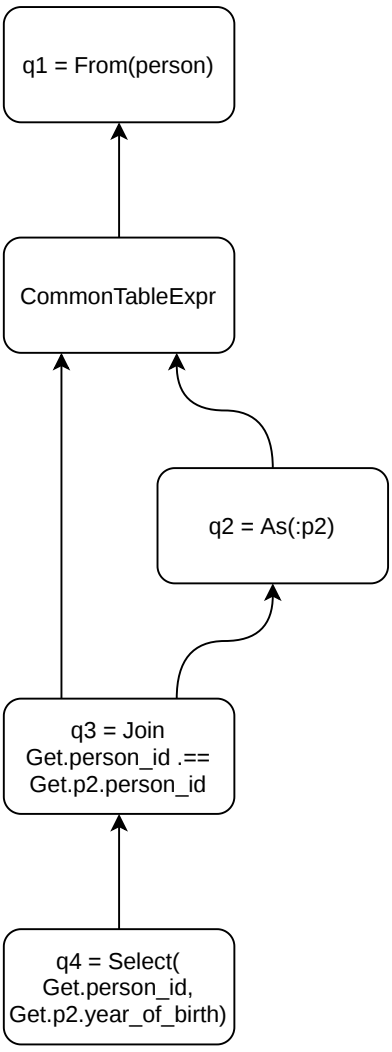

q1 = From(person)
q2 = As(:p2)
q3 = Join
Get.person_id .==
Get.p2.person_id
q4 = Select(
Get.person_id,
Get.p2.year_of_birth)

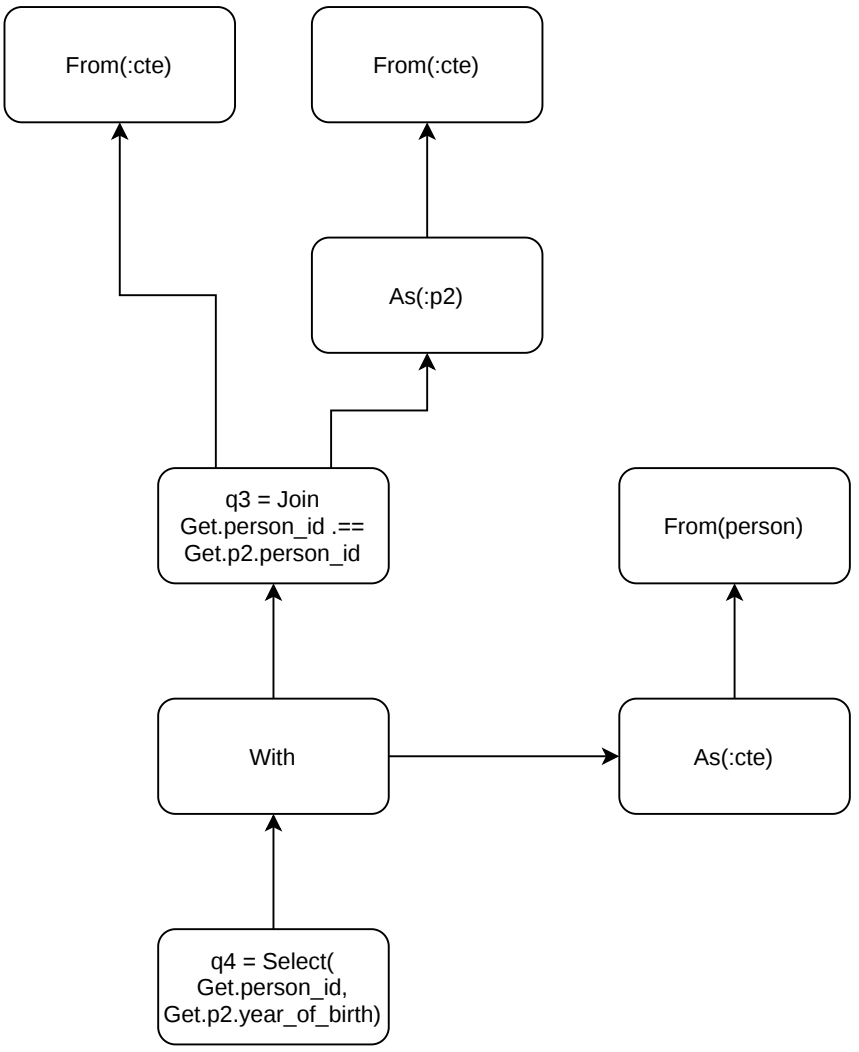

q1 = From(person)
CommonTableExpr
q2 = As(:p2)
q3 = Join
Get.person_id .==
Get.p2.person_id
q4 = Select(
Get.person_id,
Get.p2.year_of_birth)

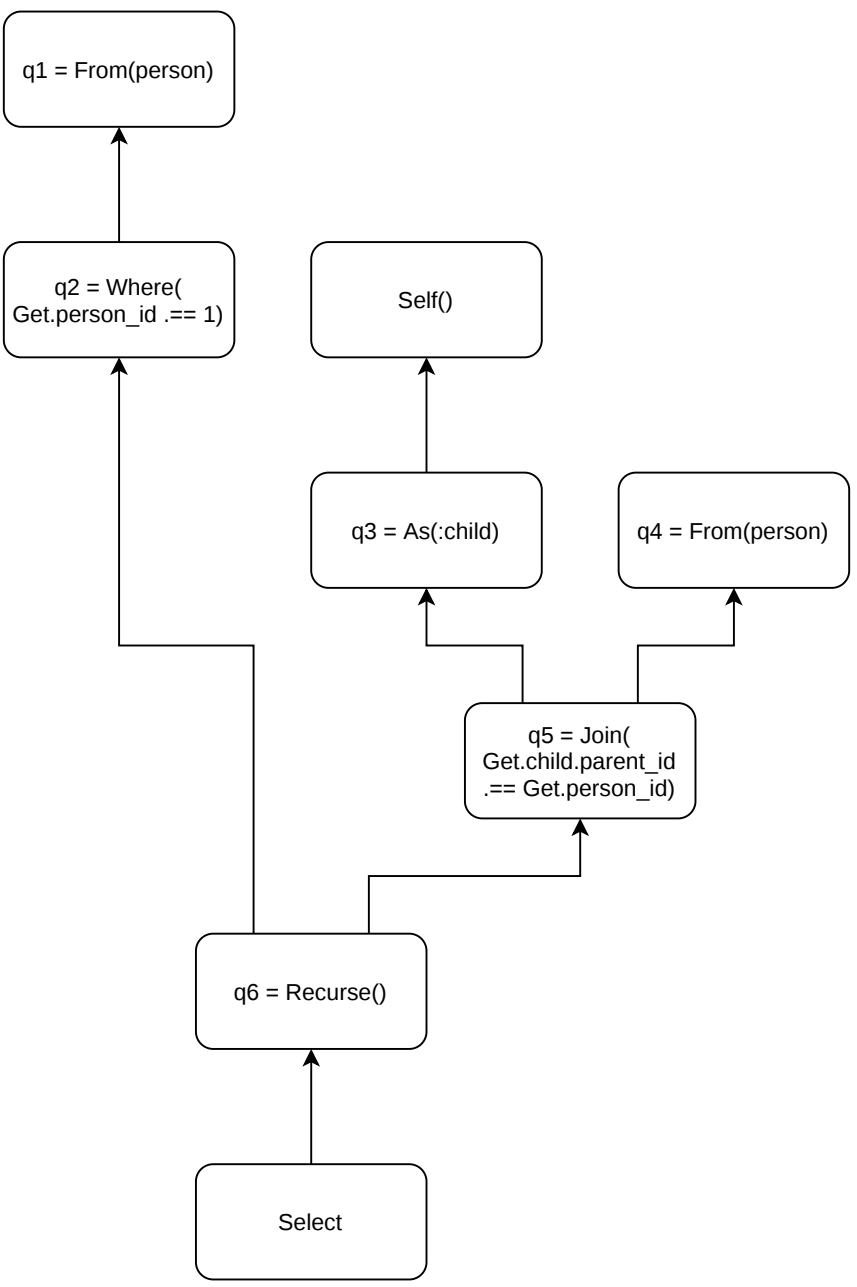

q1 = From(person)
CreateTempTable
q2 = As(:p2)
q3 = Join
Get.person_id .==
Get.p2.person_id
q4 = Select(
Get.person_id,
Get.p2.year_of_birth)

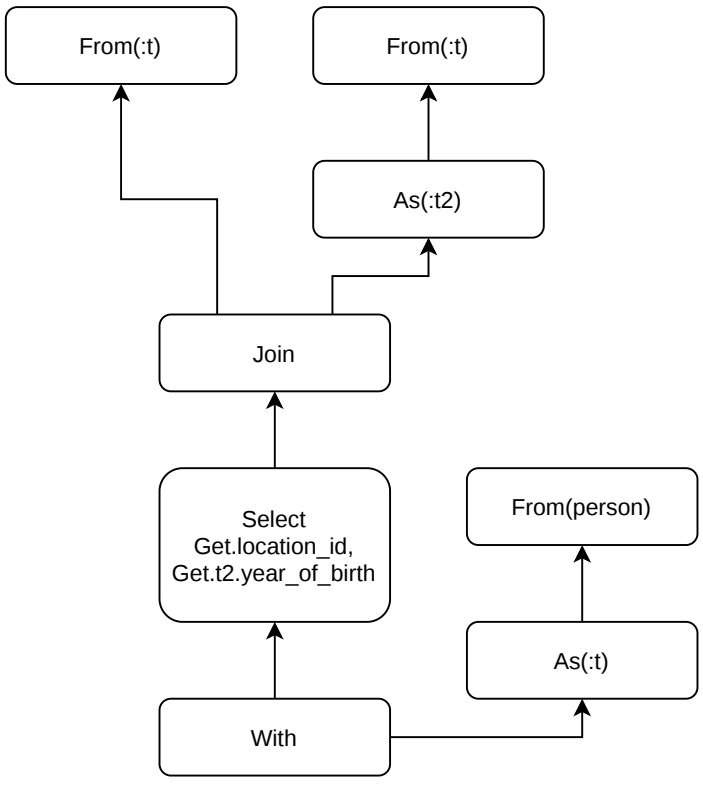

q1 = From(person)
q2 = Where(
Get.person_id .== 1)
q3 = As(:child)
q4 = From(person)
q5 = Join(
Get.child.parent_id
.== Get.person_id)
q6 = Recurse()

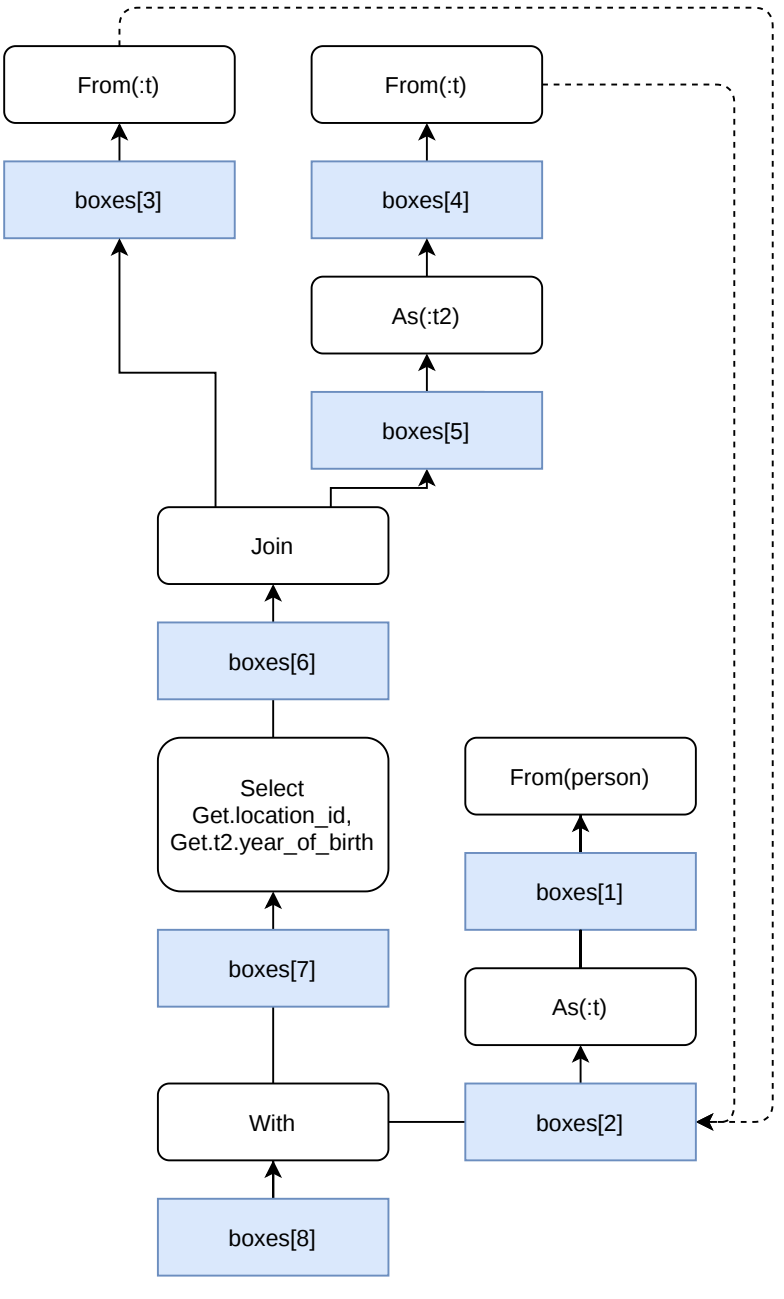

q1 = From(person)
From(:self)
q2 = Where(
Get.person_id .== 1)
q3 = As(:child)
q4 = From(person)
q5 = Join(
Get.child.parent_id
.== Get.person_id)
As(:self)
q6 = Recurse()
Select

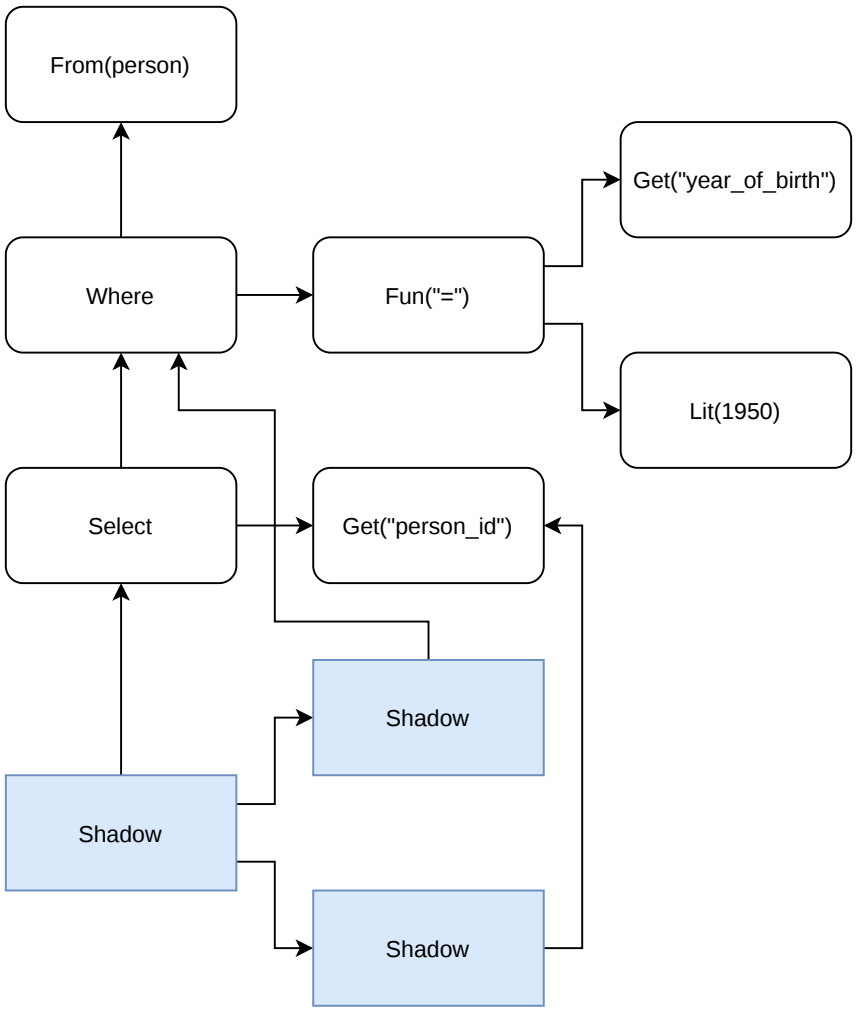

q1 = From(person)
CommonTableExpr
q2 = As(:p2)
q3 = Join
Get.person_id .==
Get.p2.person_id
q4 = Select(
Get.person_id,
Get.p2.year_of_birth)

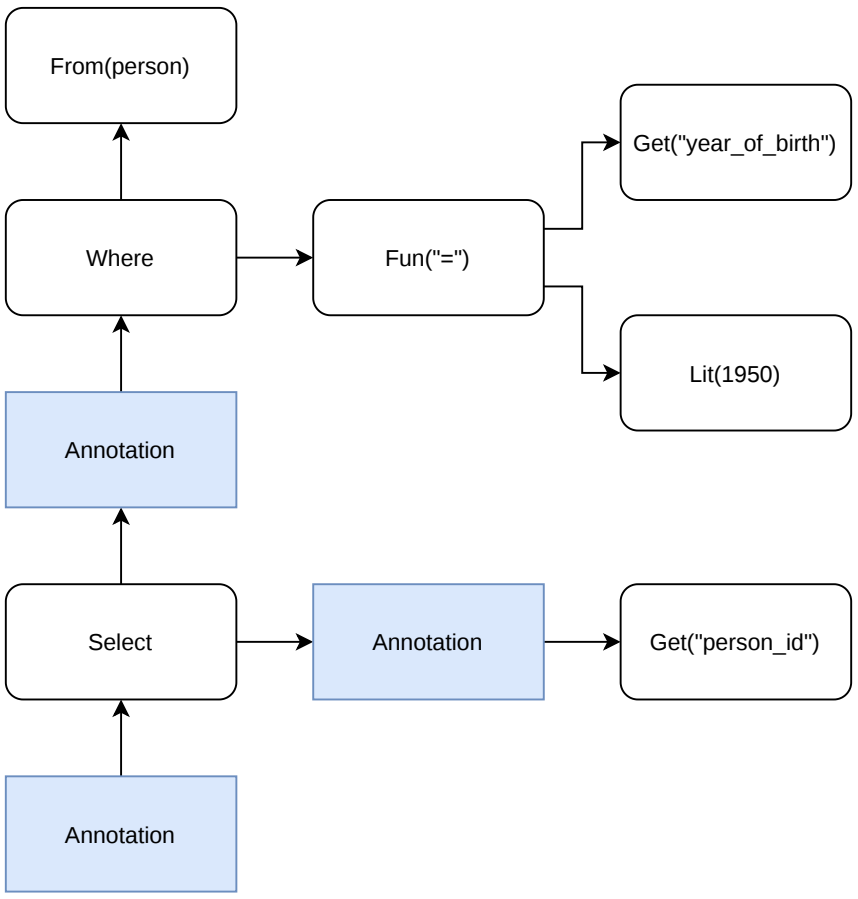

From(:cte)
From(:cte)
As(:p2)
q3 = Join
Get.person_id .==
Get.p2.person_id
From(person)
With
As(:cte)
q4 = Select(
Get.person_id,
Get.p2.year_of_birth)

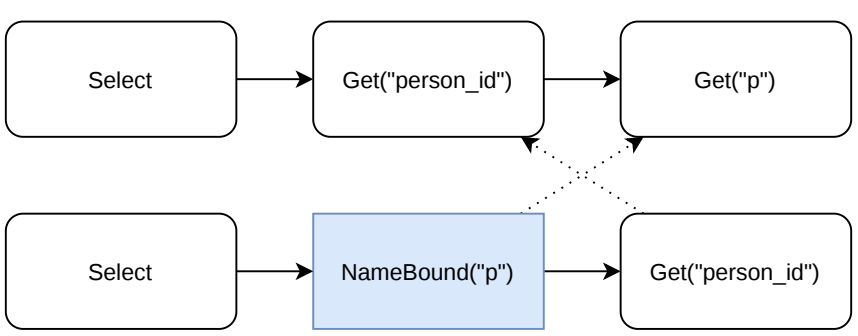

q1 = From(person)
q2 = Where(
Get.person_id .== 1)
Self()
q3 = As(:child)
q4 = From(person)
q5 = Join(
Get.child.parent_id
.== Get.person_id)
q6 = Recurse()
Select

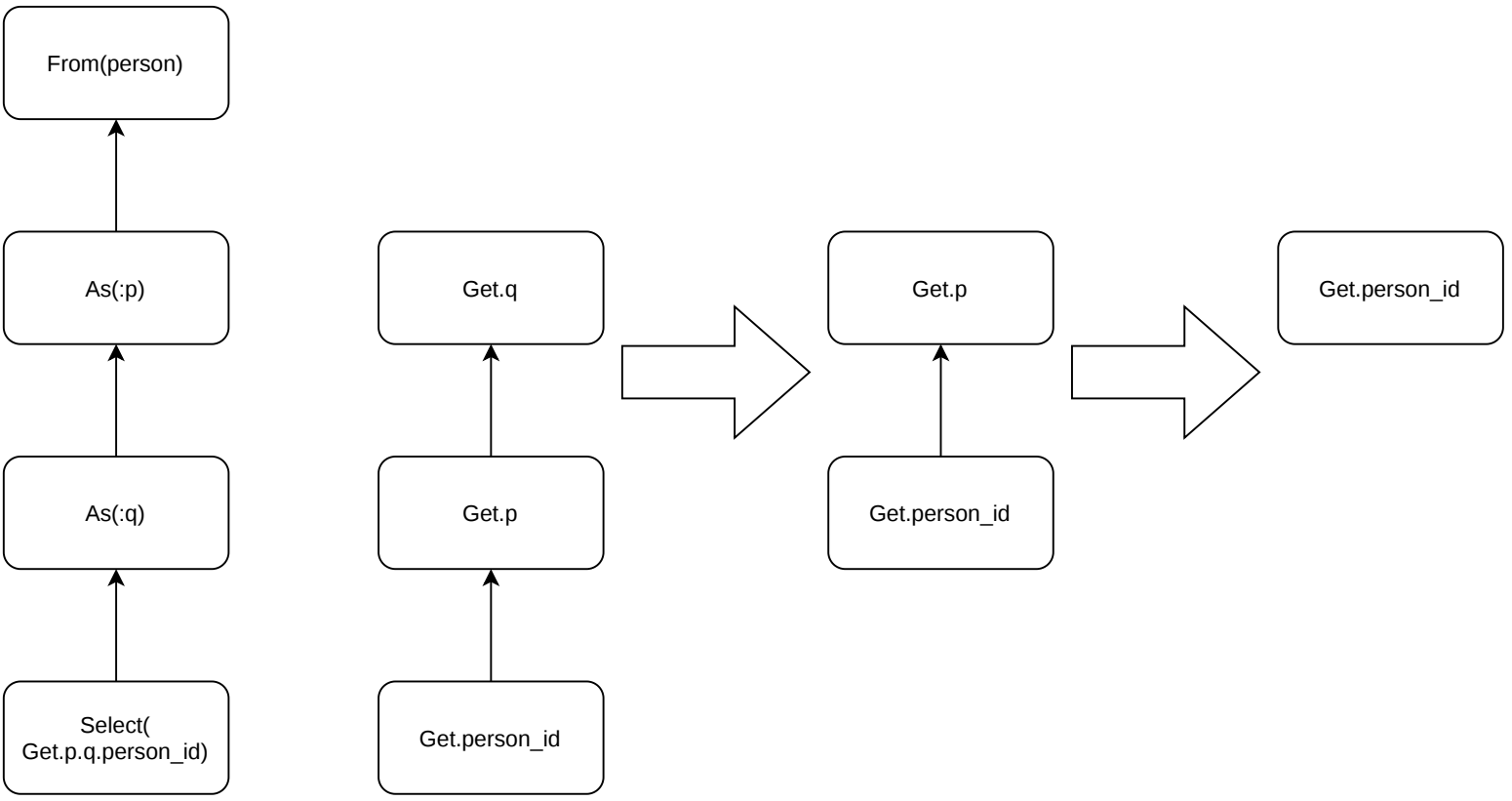

From(:t)
From(:t)
As(:t2)
Join
Select
Get.location_id,
Get.t2.year_of_birth
From(person)
As(:t)
With

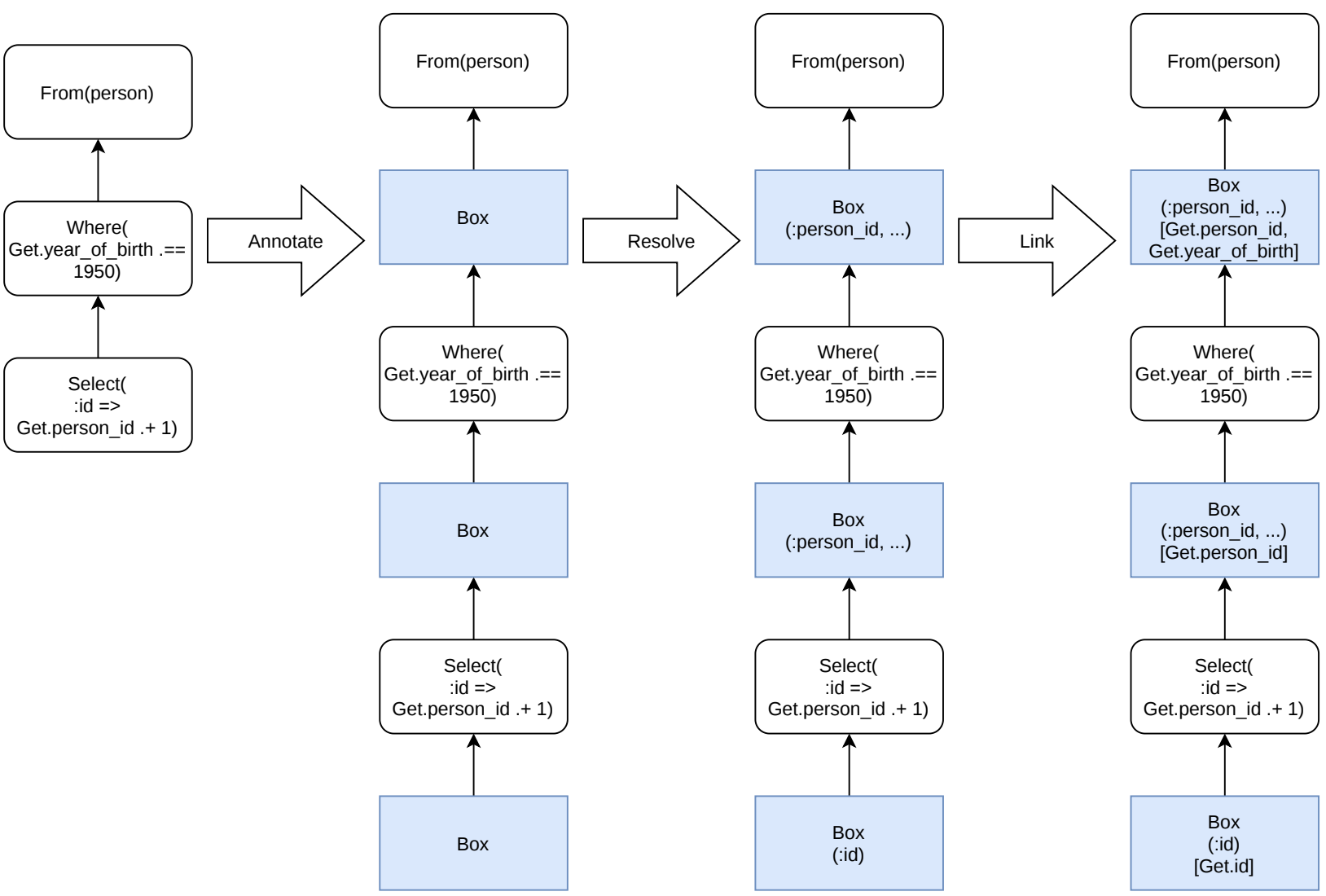

From(:t)
From(:t)
boxes[3]
boxes[4]
As(:t2)
boxes[5]
Join
boxes[6]
Select
Get.location_id,
Get.t2.year_of_birth
From(person)
boxes[1]
boxes[7]
As(:t)
With
boxes[2]
boxes[8]

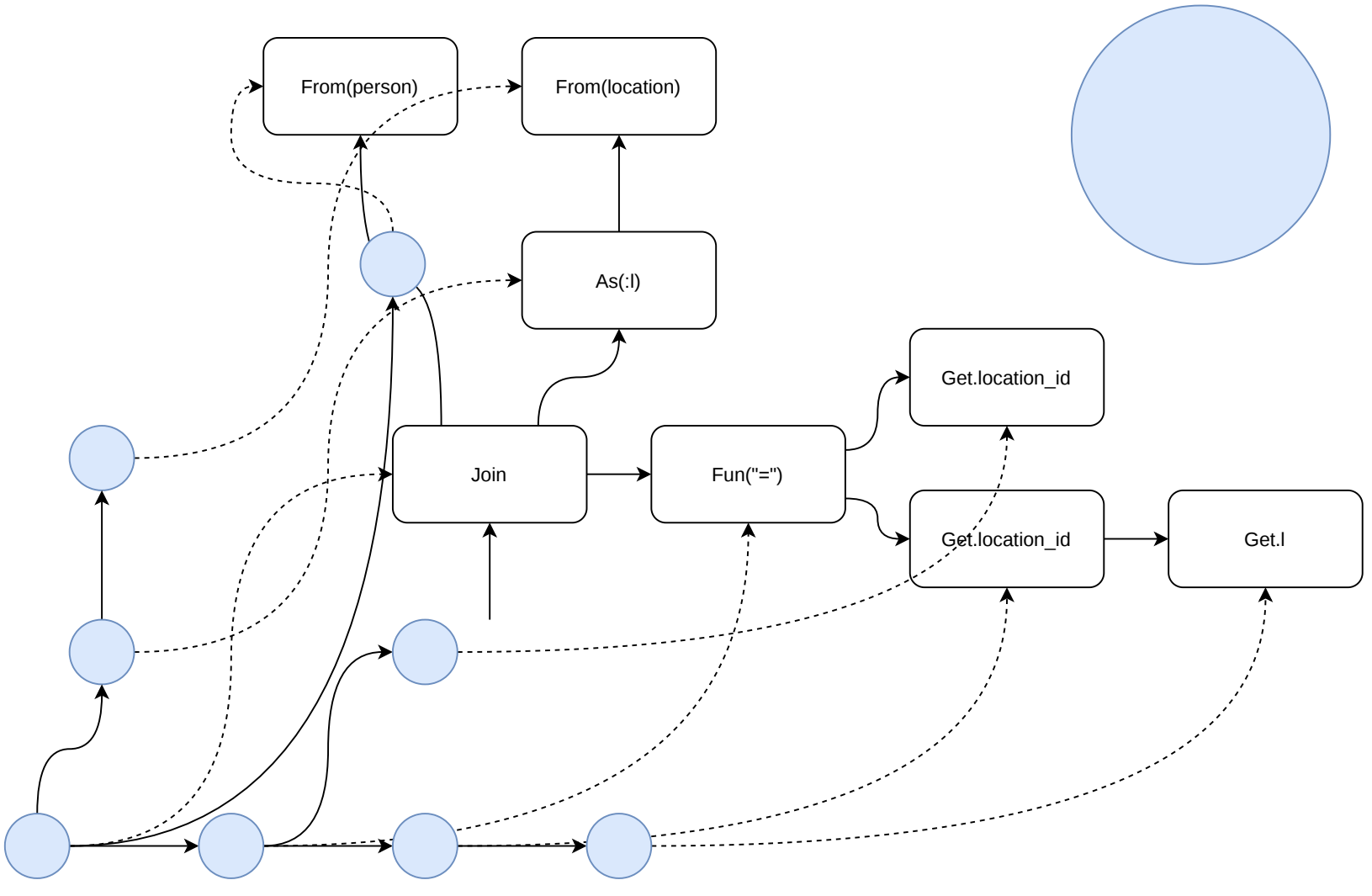

From(person)
Where
Fun("=")
Get("year_of_birth")
Lit(1950)
Select
Get("person_id")
Shadow
Shadow
Shadow

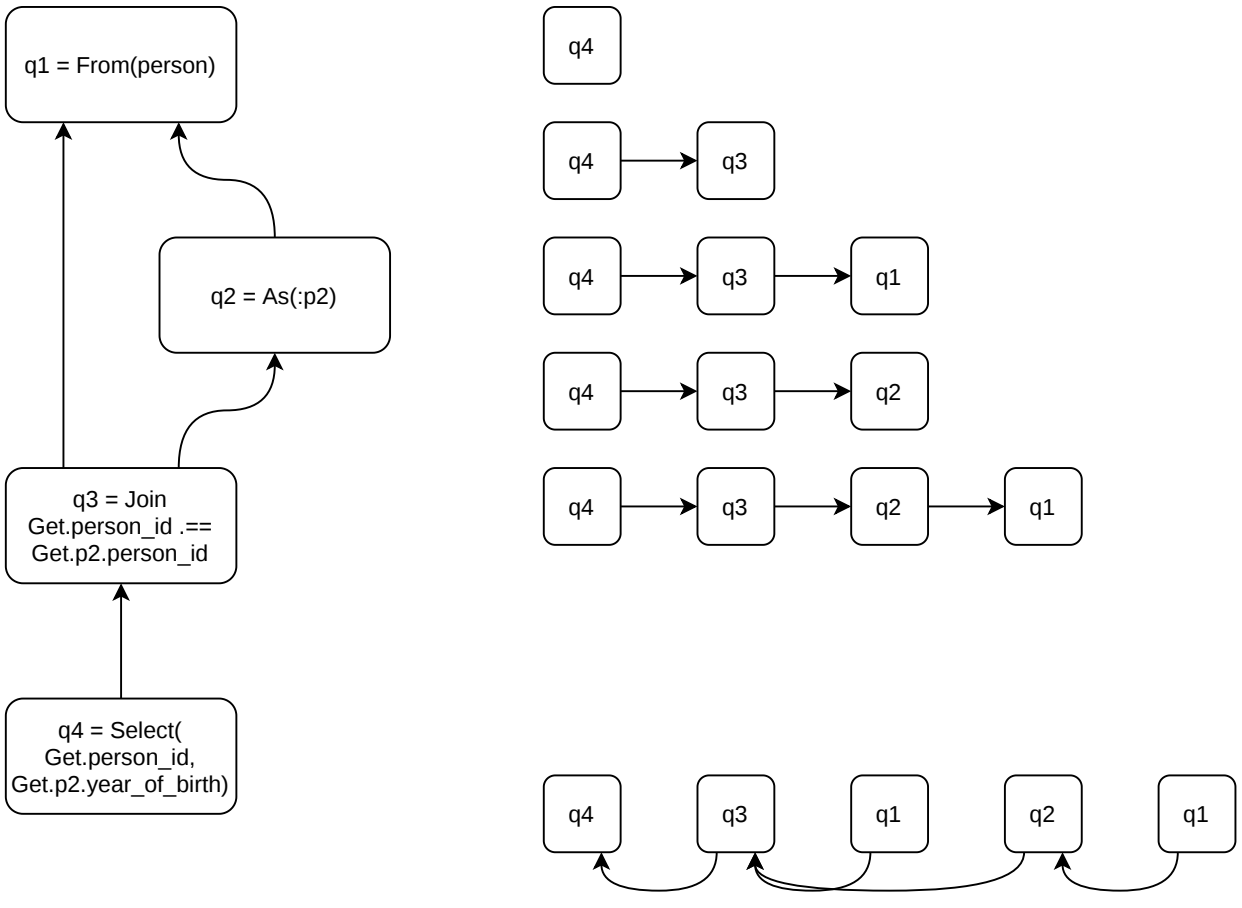

From(person)
Where
Fun("=")
Get("year_of_birth")
Lit(1950)
Annotation
Select
Annotation
Get("person_id")
Annotation


Select
Get("person_id")
Get("p")
Select
NameBound("p")
Get("person_id")


From(person)
As(:p)
As(:q)
Select(
Get.p.q.person_id)
Get.q
Get.p
Get.person_id
Get.p
Get.person_id
Get.person_id


From(person)
Where(
Get.year_of_birth .==
1950)
Select(
:id =>
Get.person_id .+ 1)
Annotate
From(person)
Box
Where(
Get.year_of_birth .==
1950)
Box
Select(
:id =>
Get.person_id .+ 1)
Box
Resolve
From(person)
Box
(:person_id, ...)
Where(
Get.year_of_birth .==
1950)
Box
(:person_id, ...)
Select(
:id =>
Get.person_id .+ 1)
Box
(:id)
Link
From(person)
Box
(:person_id, ...)
[Get.person_id,
Get.year_of_birth]
Where(
Get.year_of_birth .==
1950)
Box
(:person_id, ...)
[Get.person_id]
Select(
:id =>
Get.person_id .+ 1)
Box
(:id)
[Get.id]


From(person)
From(location)
As(:l)
Join
Fun("=")
Get.location_id
Get.location_id
Get.l


q1 = From(person)
q2 = As(:p2)
q3 = Join
Get.person_id .==
Get.p2.person_id
q4 = Select(
Get.person_id,
Get.p2.year_of_birth)
q4
q4
q3
q4
q3
q1
q4
q3
q2
q4
q3
q2
q1
q4
q3
q1
q2
q1