



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\_alias("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(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))

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



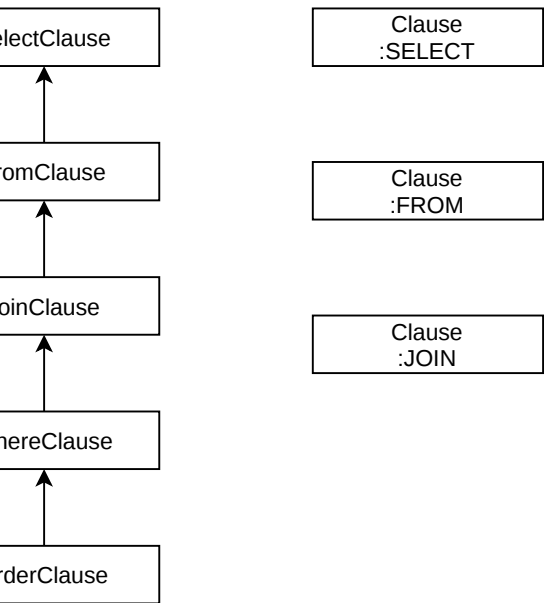
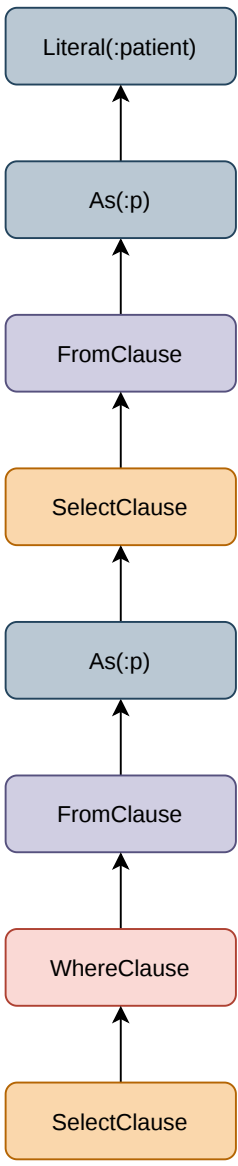
SELECT p.mrn  
FROM patient p



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



SelectClause  
patient\_id, date

FromClause  
encounter



```
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 ...





\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

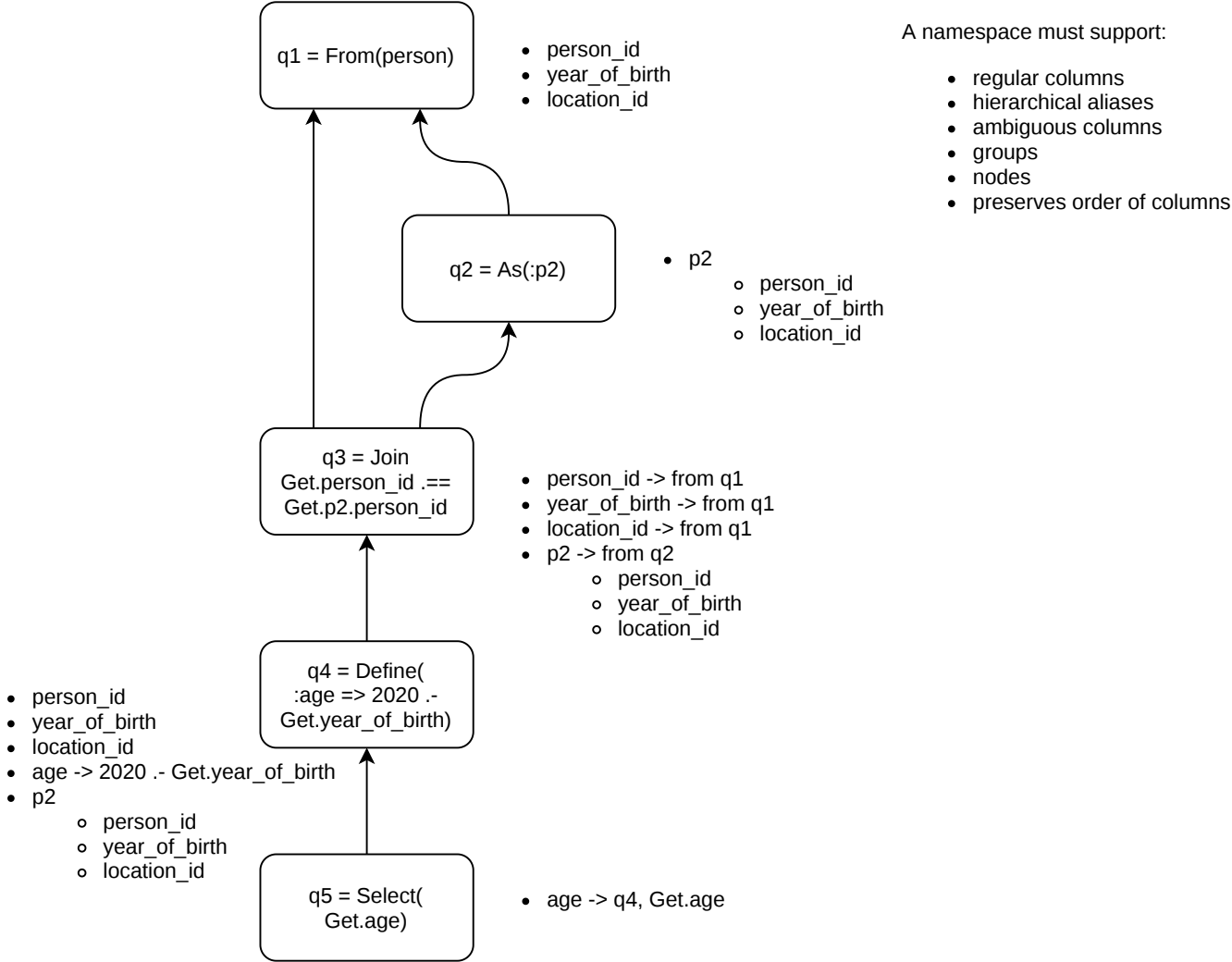
\_\_\_\_\_

\_\_\_\_\_

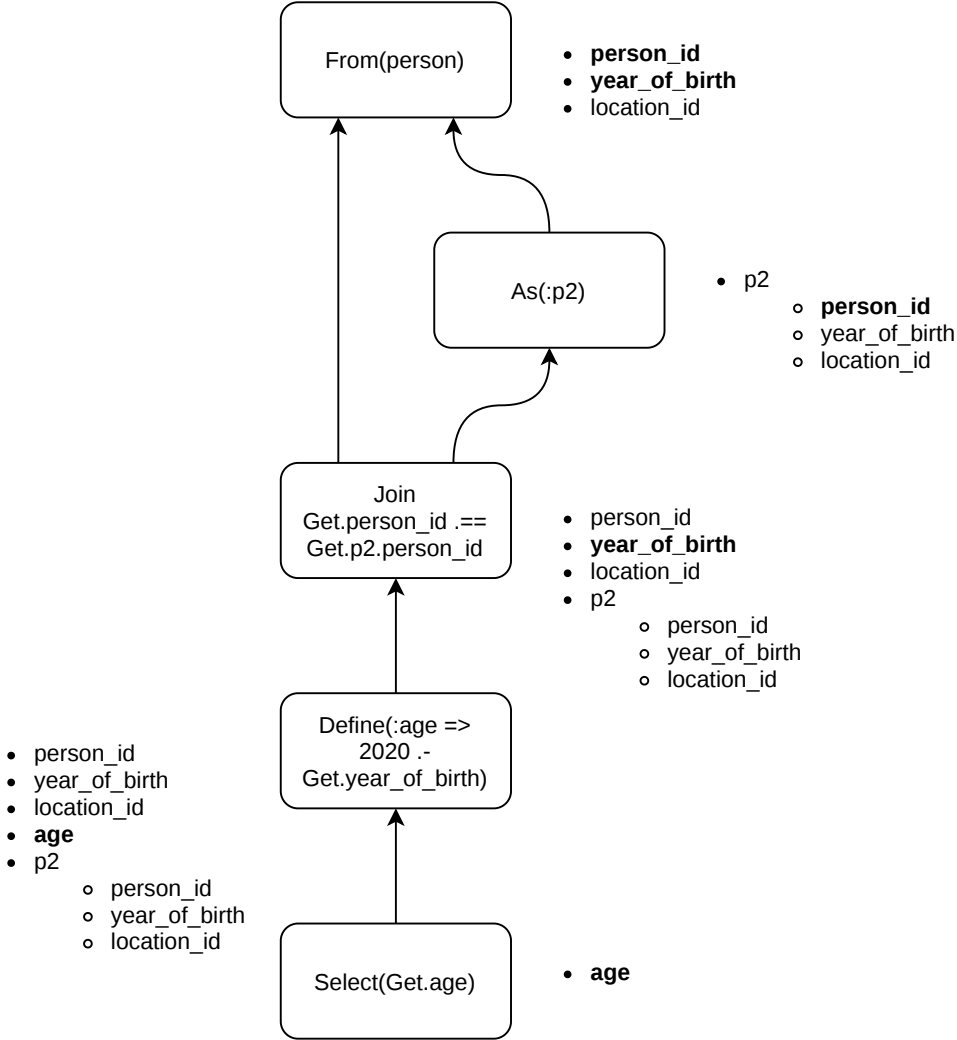
\_\_\_\_\_

\_\_\_\_\_

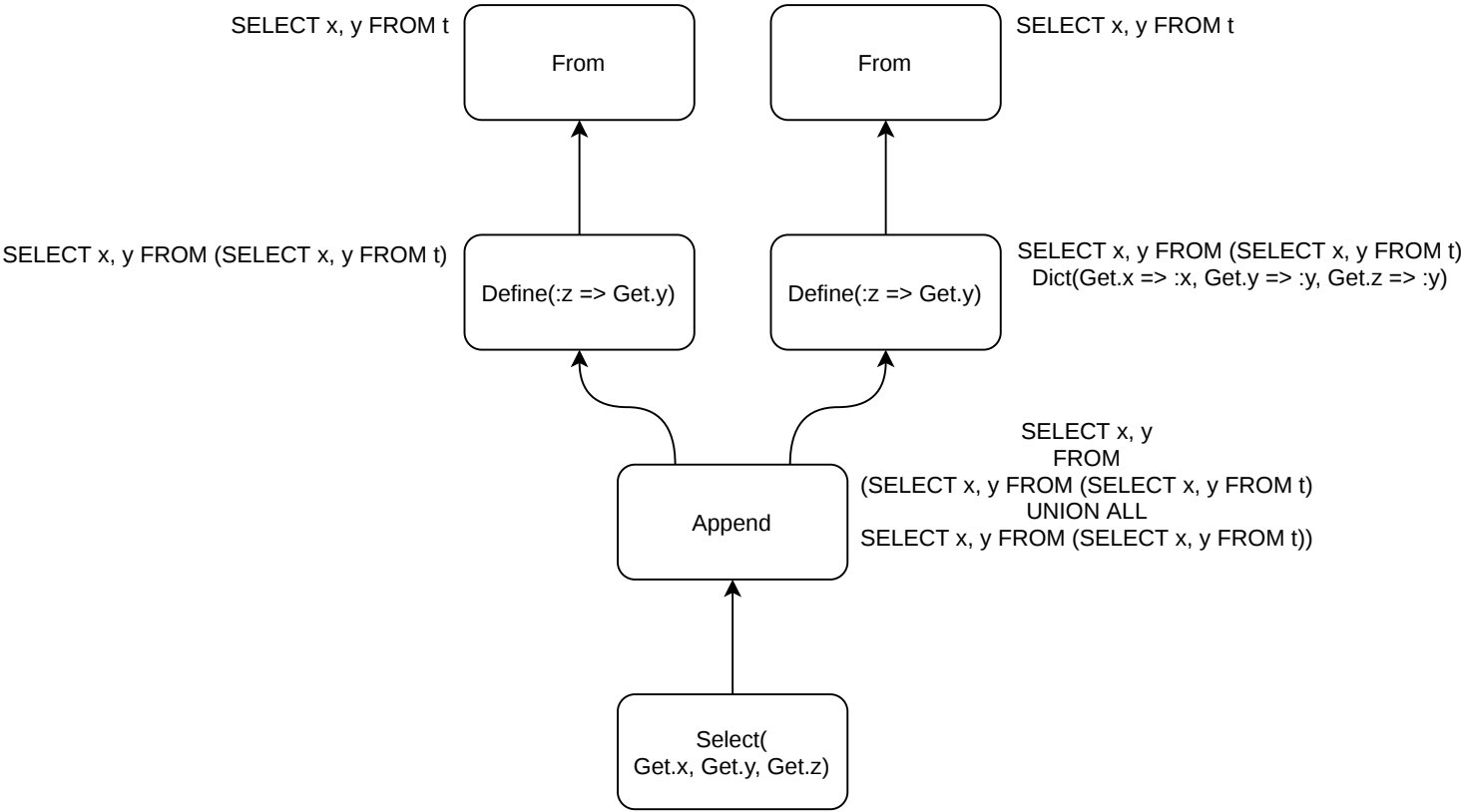
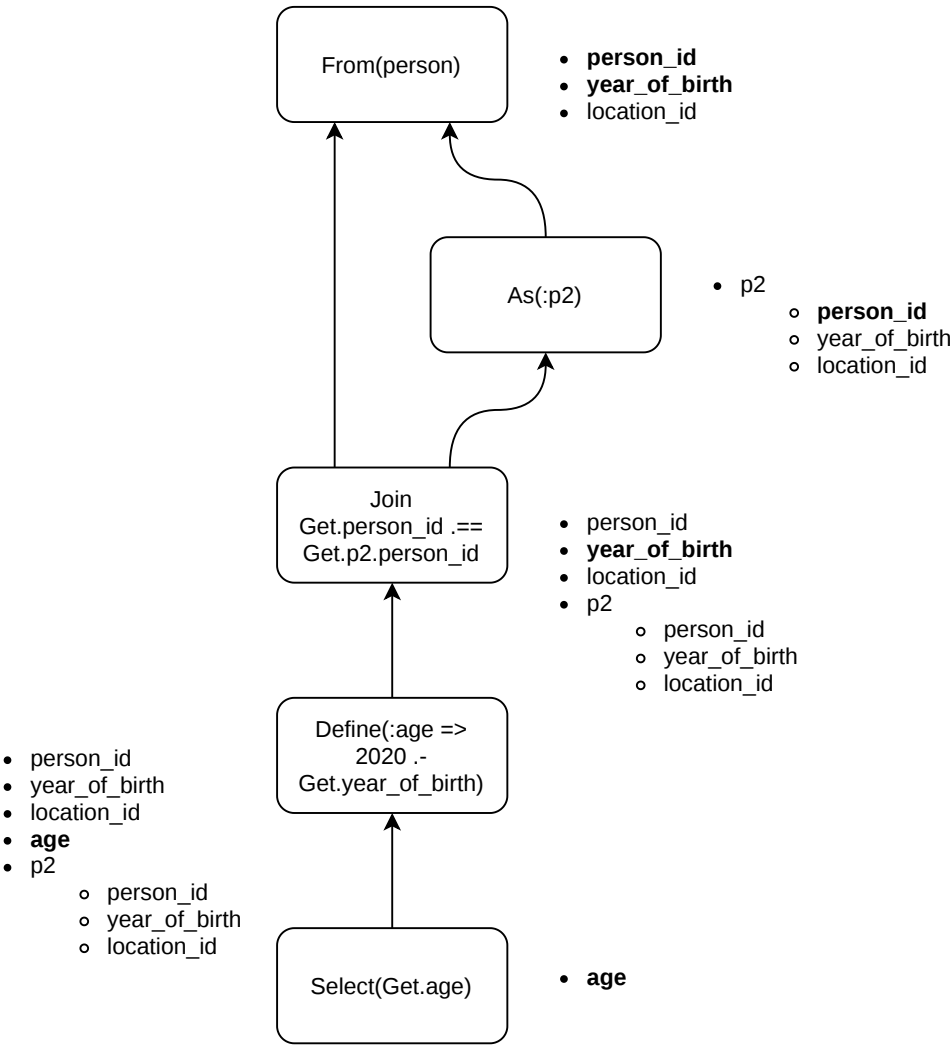
# Generate a namespace for each node

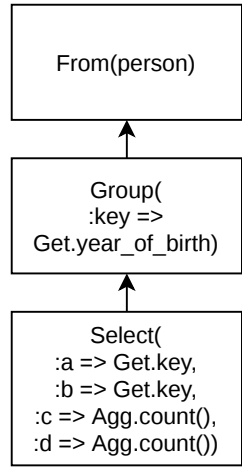


# Generate an order for each node



# Generate an order for each node





## Generate clauses

req.refs::Vector{SQLNode}

Get.key

Get.key

Agg.count()

Agg.count()

req.refs::Vector{SQLNode}

Get.key

Get.key

Agg.count()

Agg.count()

Vector{SQLClause}

ID(:person\_1) |> ID(:year\_of\_birth)

ID(:person\_1) |> ID(:year\_of\_birth)

AGG(:count, OP(:\*))

AGG(:count, OP(:\*))

## Find duplicate clauses

req.refs::Vector{SQLNode}

Get.key

Get.key

Agg.count()

Agg.count()

Vector{SQLClause}

ID(:person\_1) |> ID(:year\_of\_birth)

ID(:person\_1) |> ID(:year\_of\_birth)

AGG(:count, OP(:\*))

AGG(:count, OP(:\*))

## Generate column aliases

req.refs::Vector{SQLNode}

Get.key

Get.key

Agg.count()

Agg.count()

ID(:person\_1) |> ID(:year\_of\_birth)

ID(:person\_1) |> ID(:year\_of\_birth)

AGG(:count, OP(:\*))

AGG(:count, OP(:\*))

:key

:count

## Make column aliases unique

req.refs::Vector{SQLNode}

Get.key

Get.key

Agg.count()

Agg.count()

ID(:person\_1) |> ID(:year\_of\_birth)

ID(:person\_1) |> ID(:year\_of\_birth)

AGG(:count, OP(:\*))

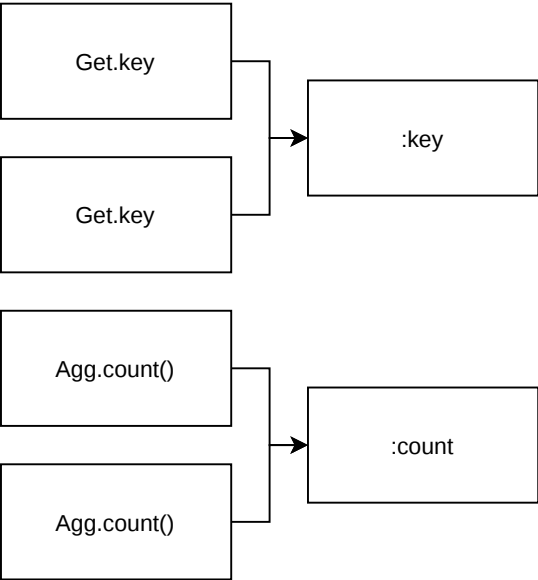
AGG(:count, OP(:\*))

:key

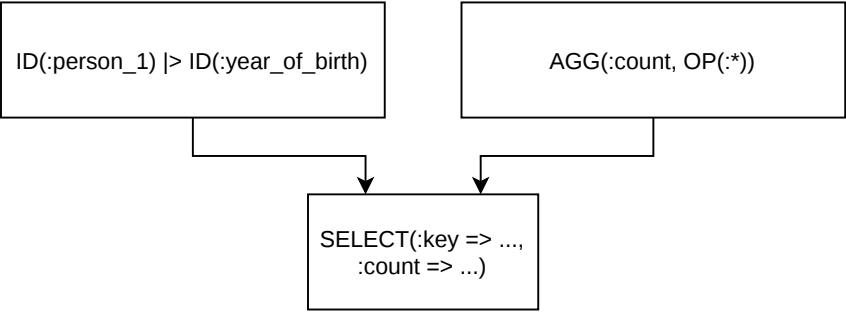
:count

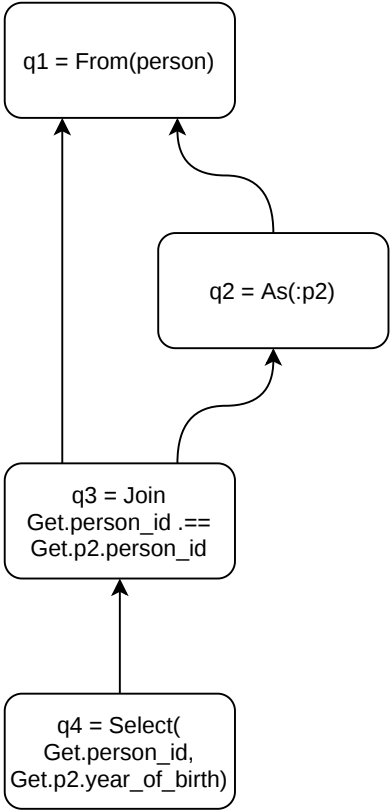
# Generate a subquery object and replacement map

Dict{SQLNode, Symbol}



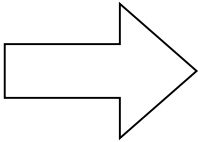
SQLClause





[q4, q3, q2, q1]

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



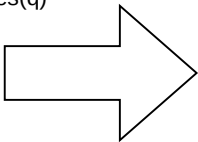
**Requests**

```
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
```

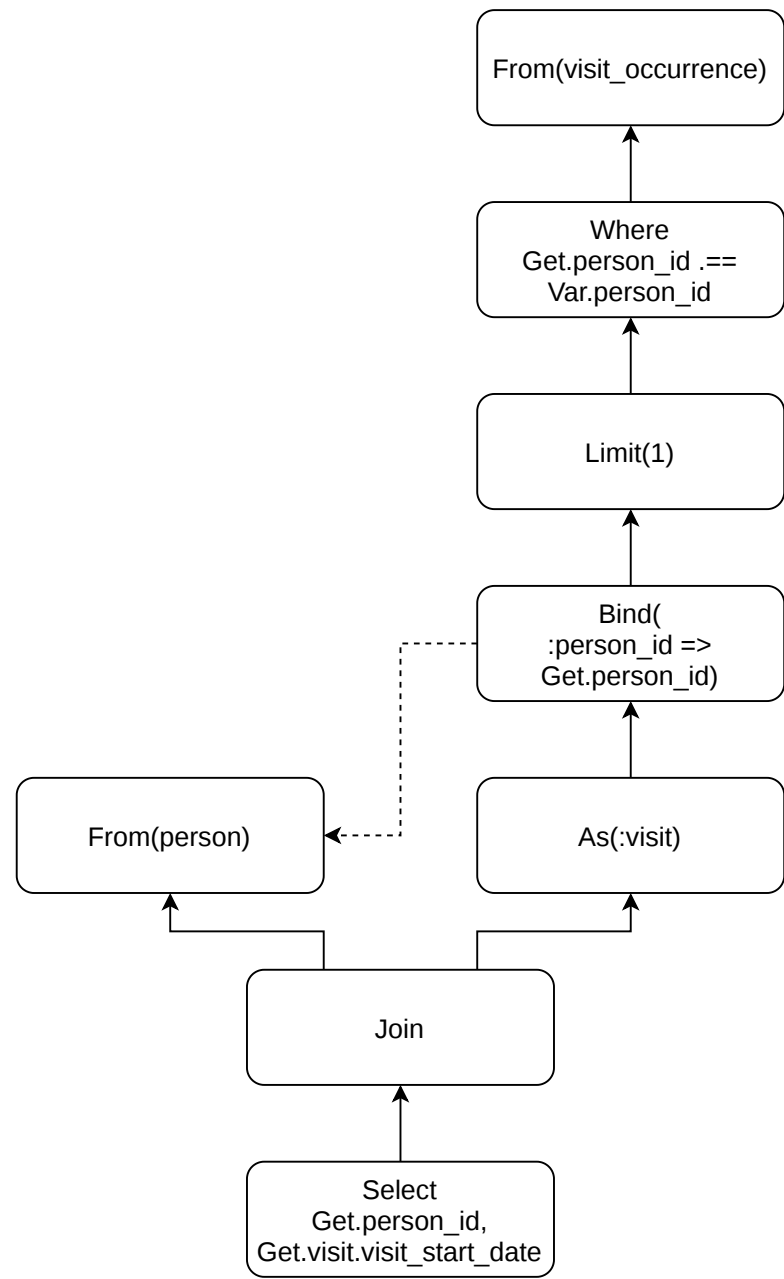
**Repl**

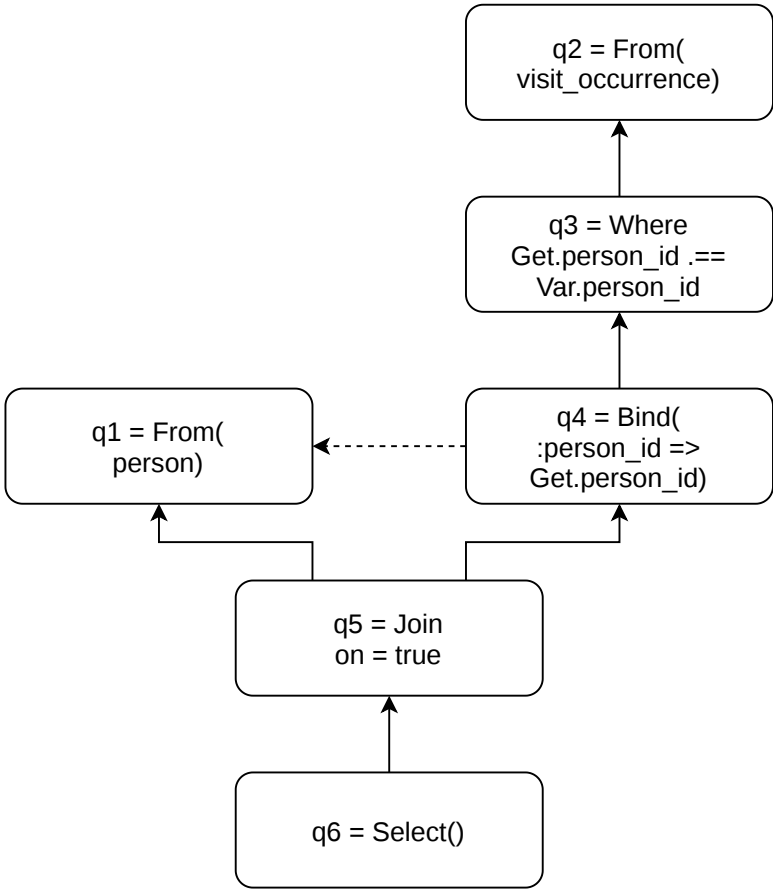
```
q1 => Dict(Get.person_id (q3) => :person_id, Get.person_id (q4) => :person_id, Get.person_id (q2) => :person_id, Get.year_of_birth (q2) => :year_of_birth)
q2 => Dict(Get.p2.person_id (q3) => :person_id, Get.p2.year_of_birth (q4) => :year_of_birth)
q3 => Dict(Get.person_id (q4) => :person_id, Get.p2.year_of_birth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

**Ambbs**

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

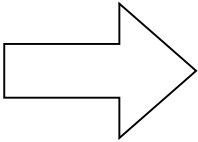






[q6, q5, q4, q3, q2, q1]

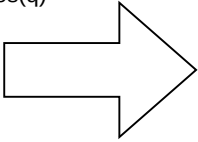
```
for q in [q6, q5, q4, q3, q2, q1]
  collect_references(q)
end
```



**Requests**

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

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



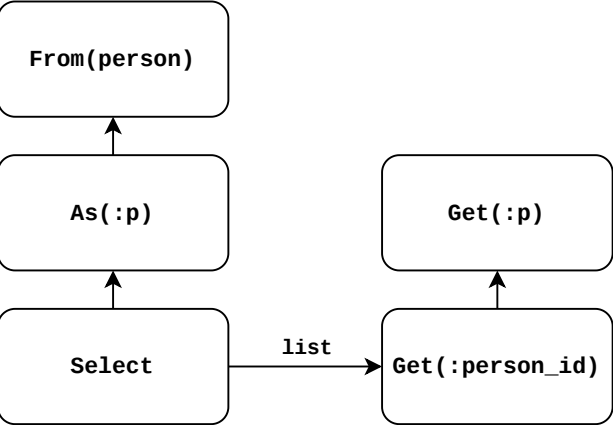
**Clauses**

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

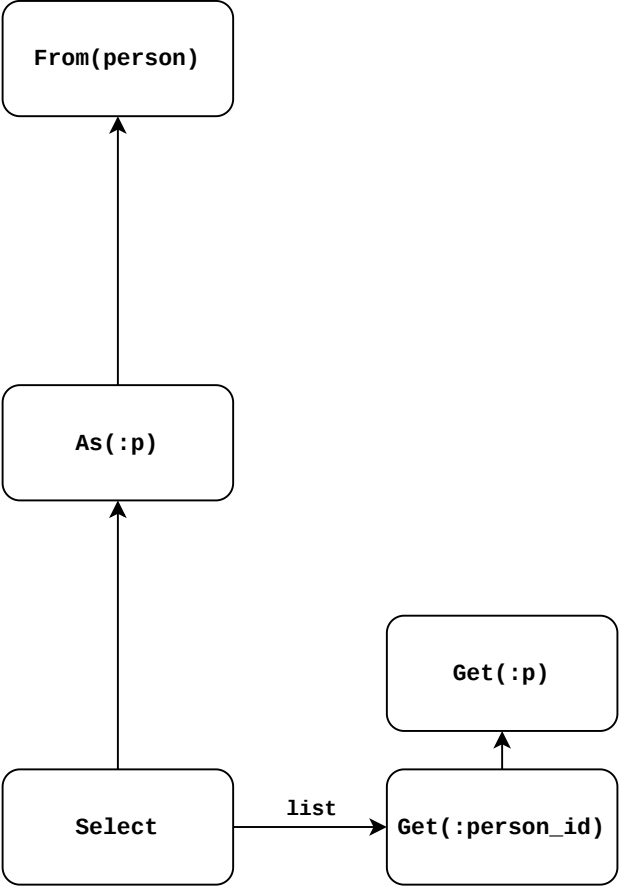
**Repl**

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

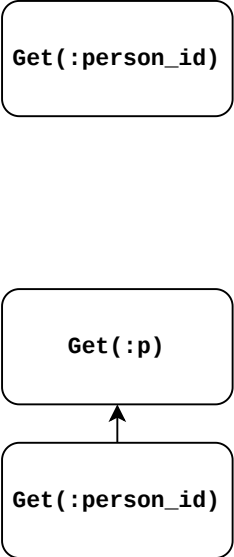
```
From(person) |>  
As(:p) |>  
Select(Get.p.person_id)
```



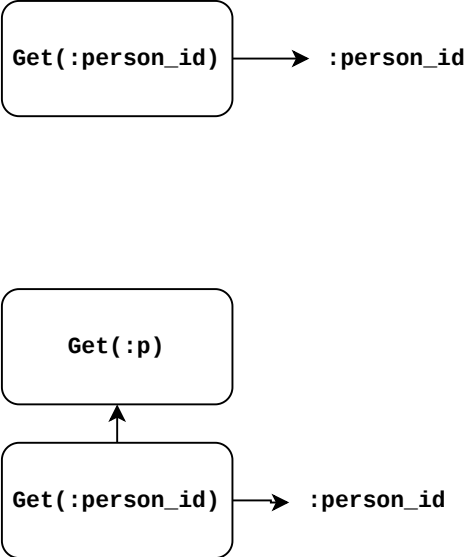
Resolve



refs



repl



New Resolve

refs

repl

