



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



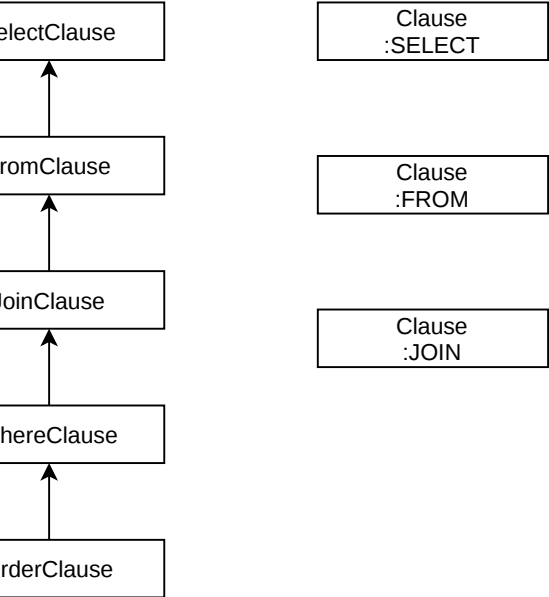
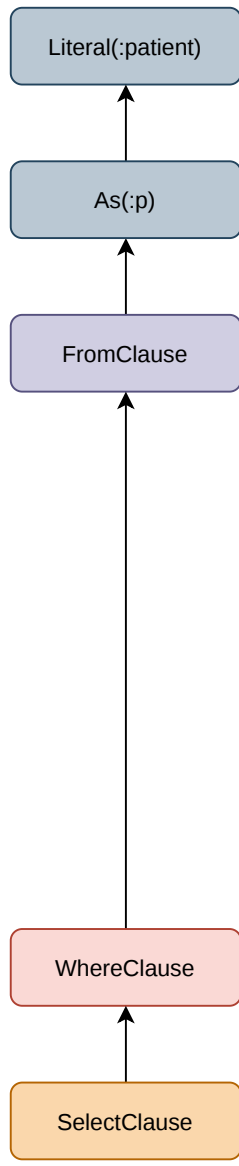
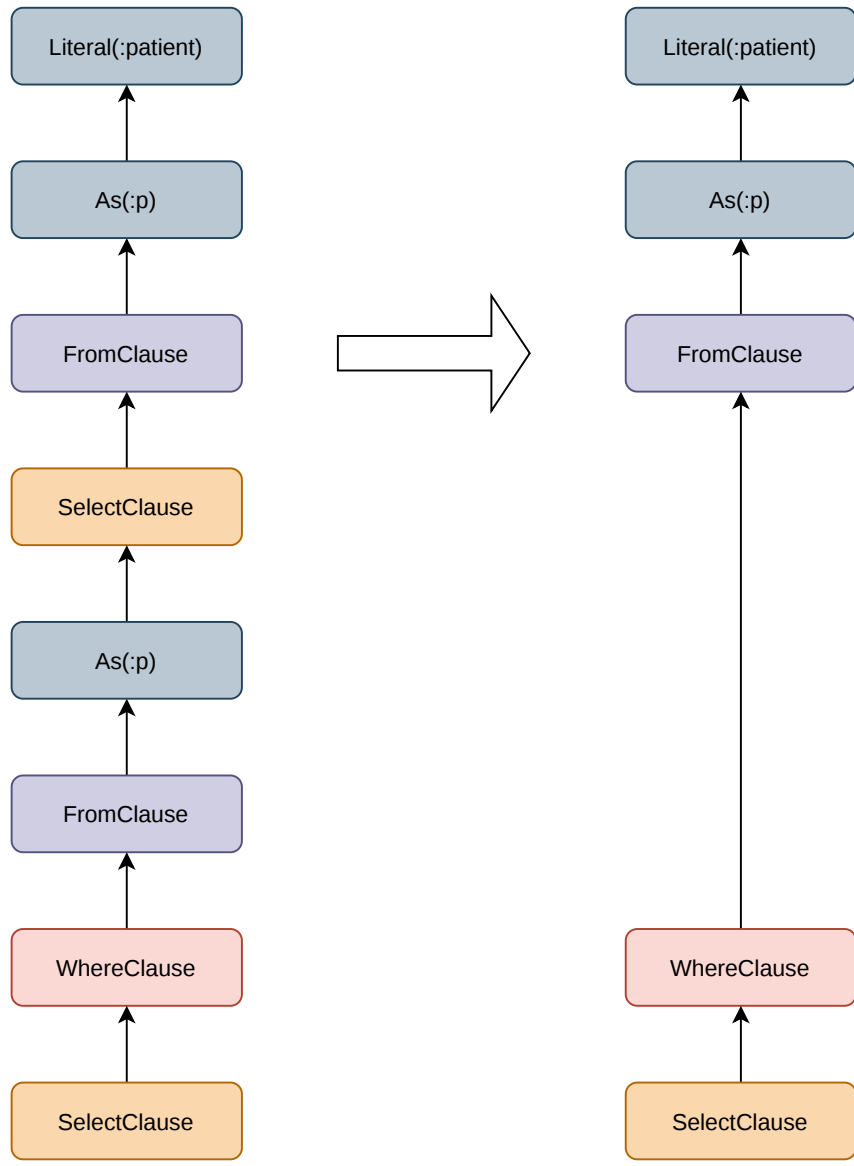
SELECT p.mrn
FROM patient p



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

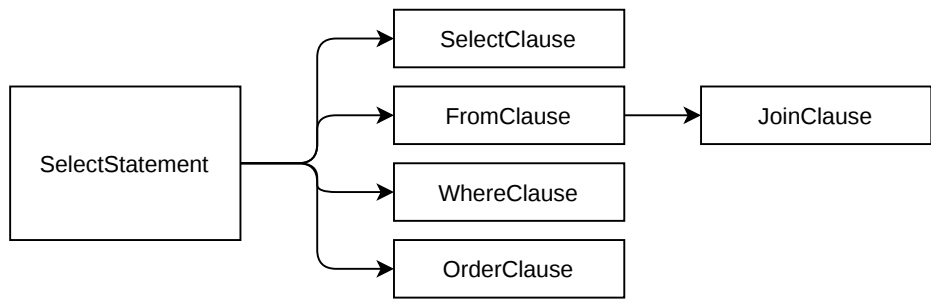
Clause
:JOIN

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)
```





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



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



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



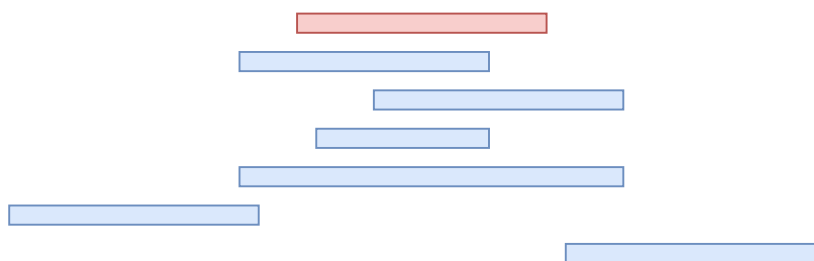
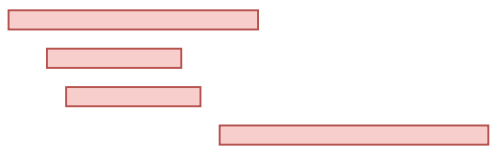
SELECT ... FROM (SELECT ... WHERE ...) AS ... WHERE ...



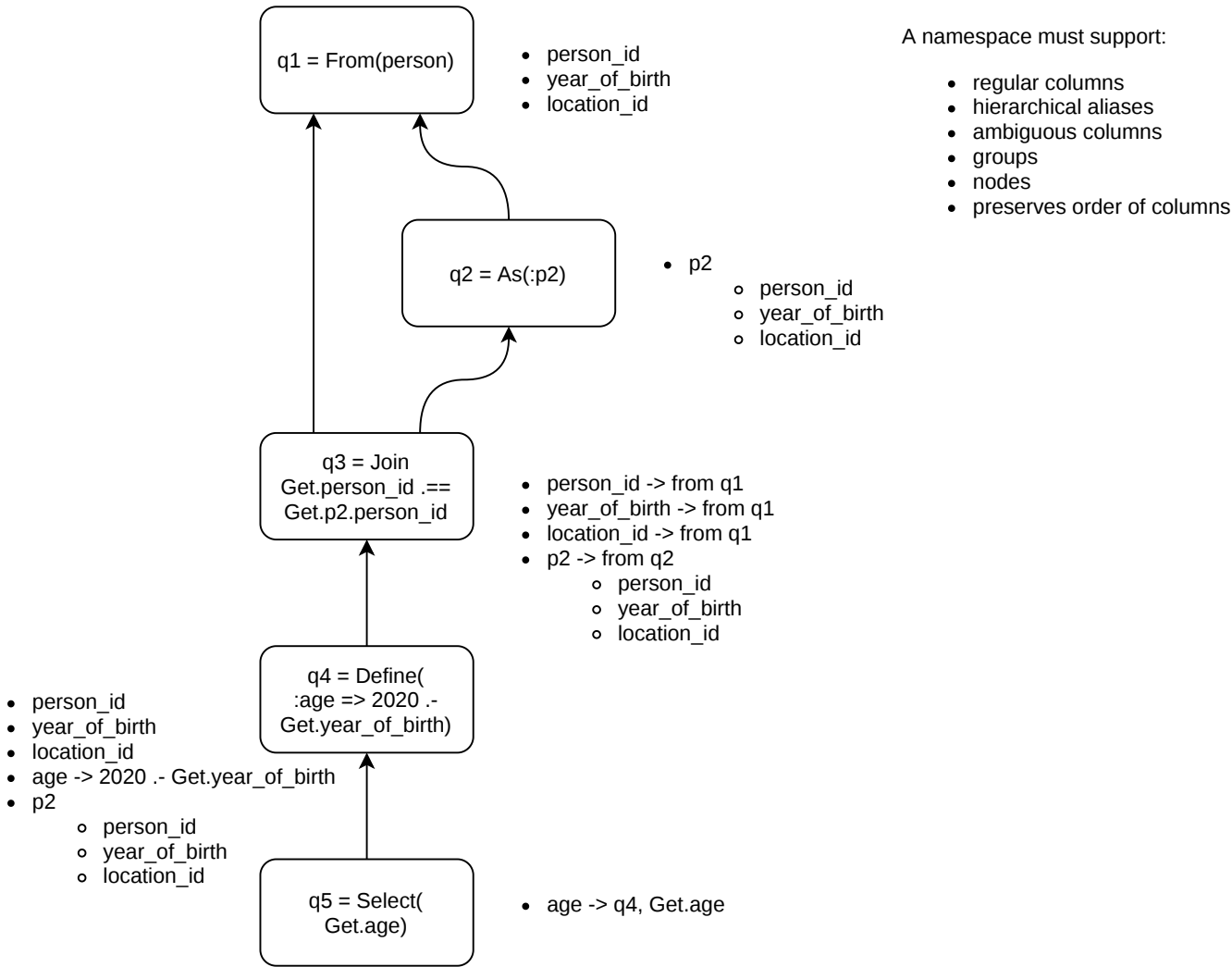
SELECT ... FROM (SELECT ... JOIN ...) AS ... WHERE ...



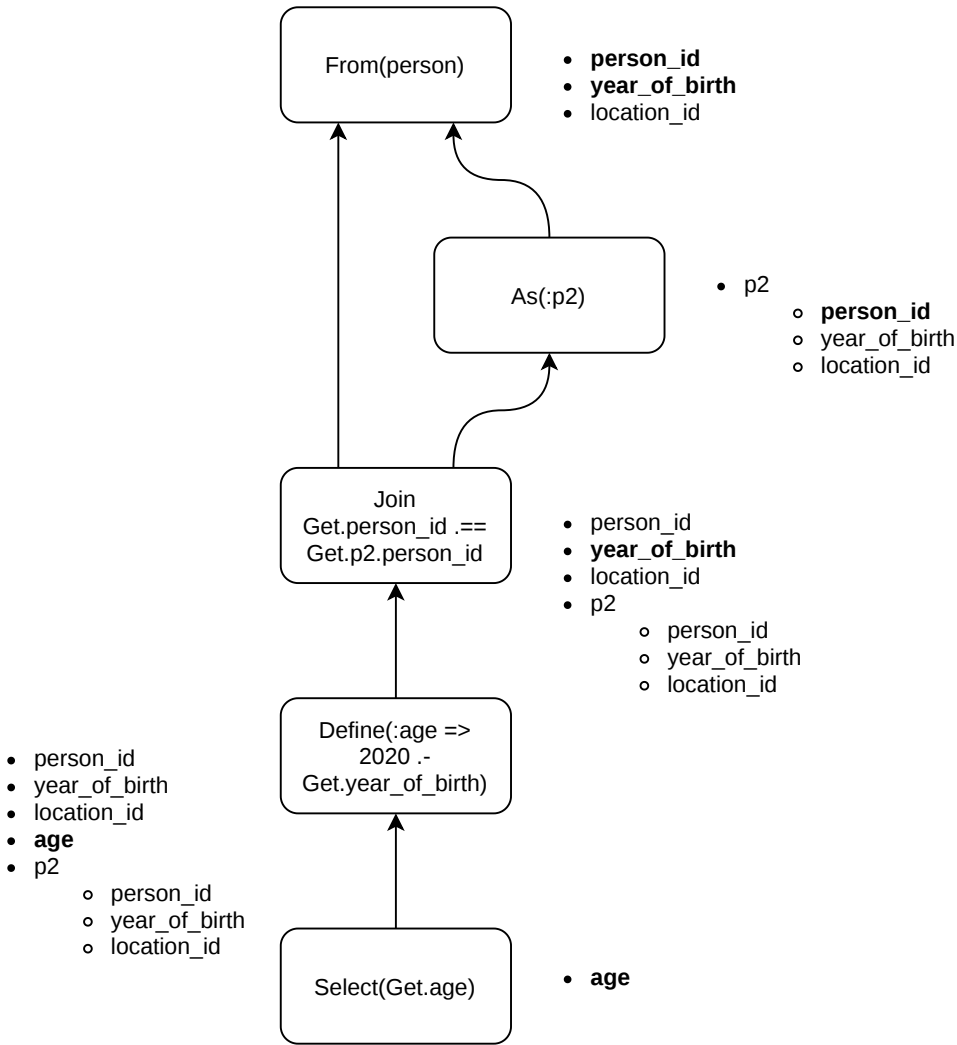




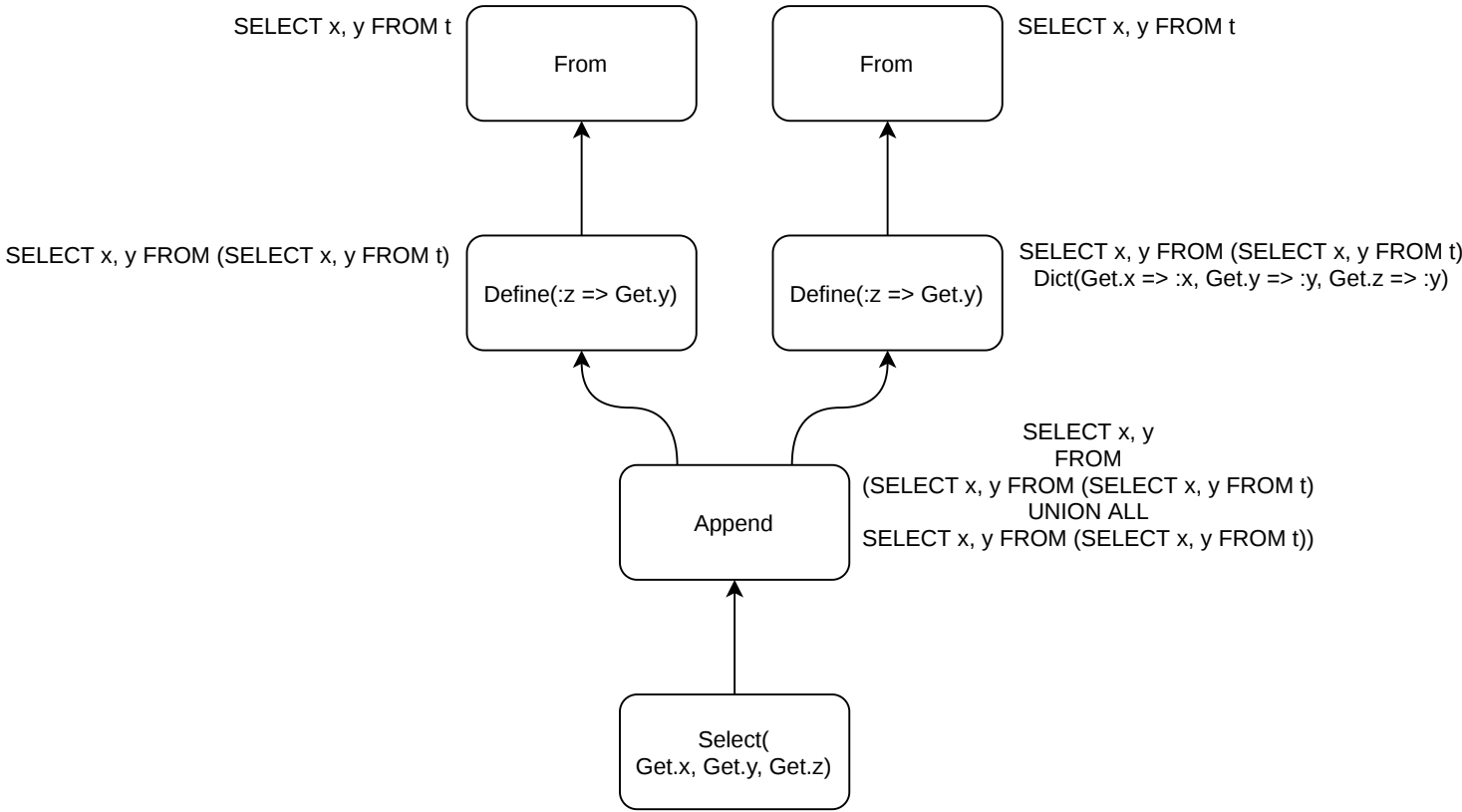
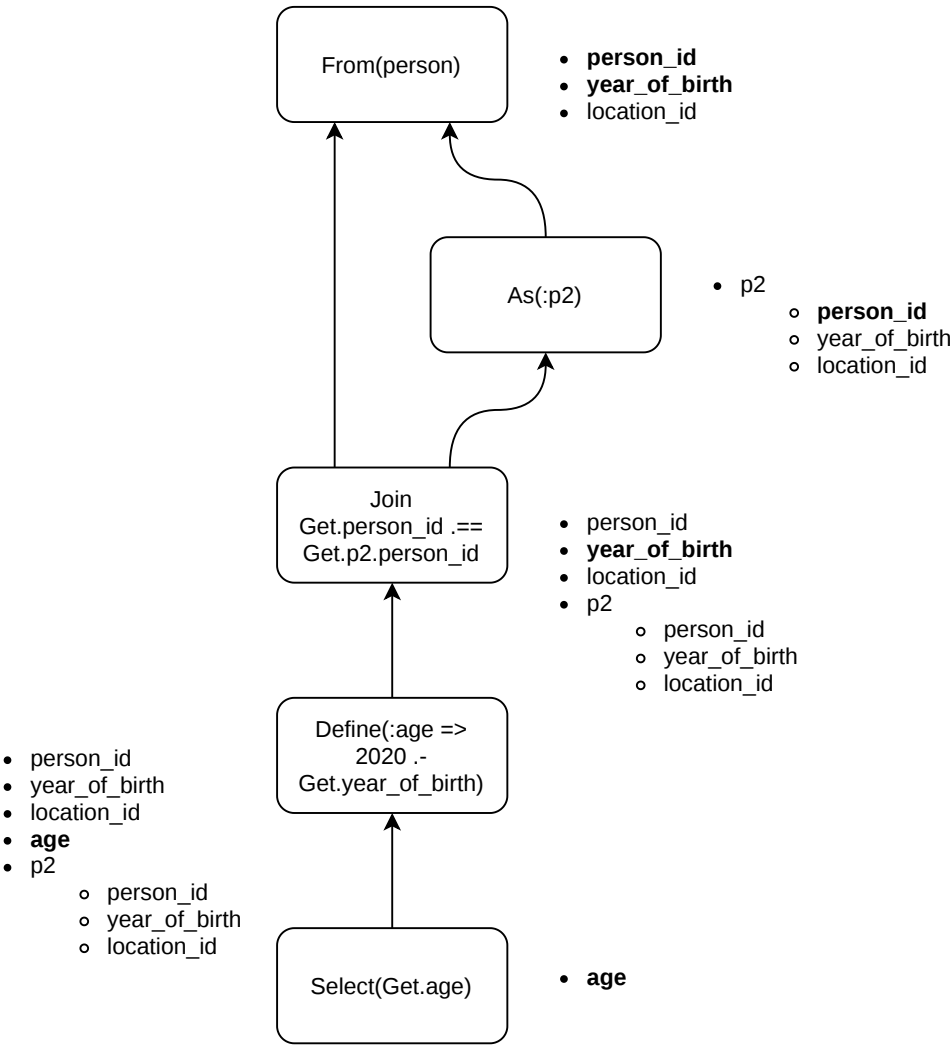
Generate a namespace for each node

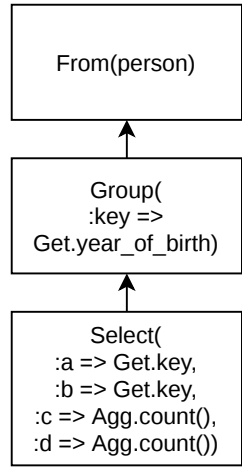


Generate an order for each node



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(:*))

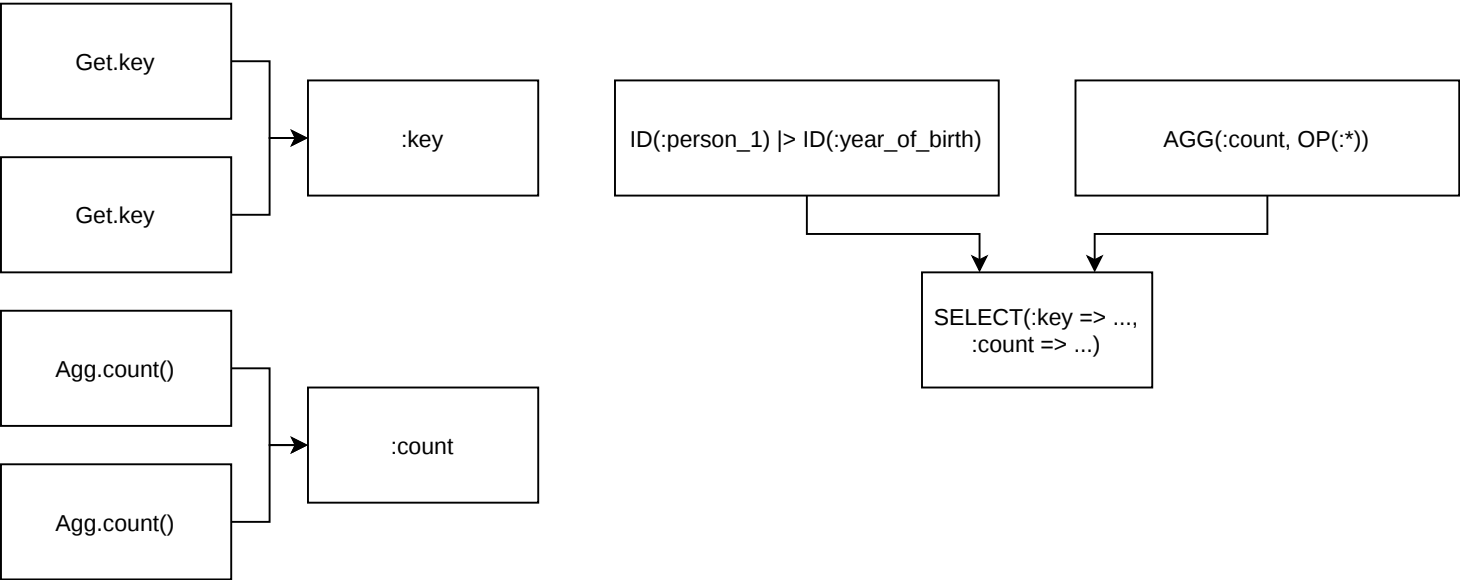
:key

:count

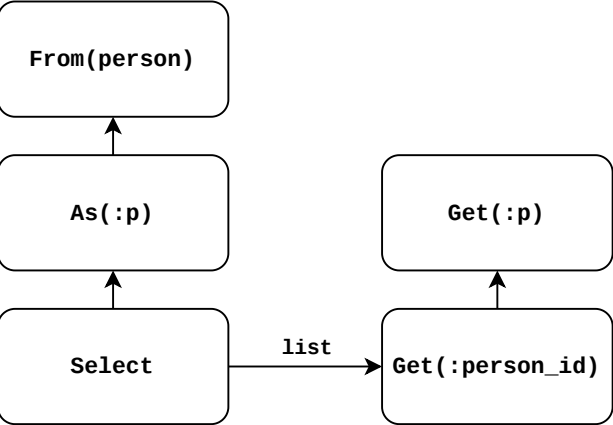
Generate a subquery object and replacement map

Dict{SQLNode, Symbol}

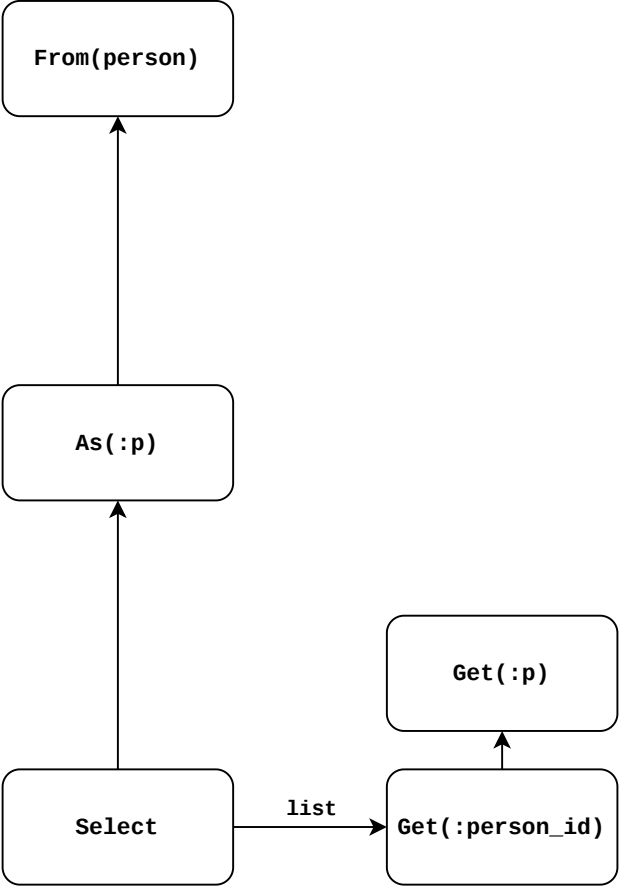
SQLClause



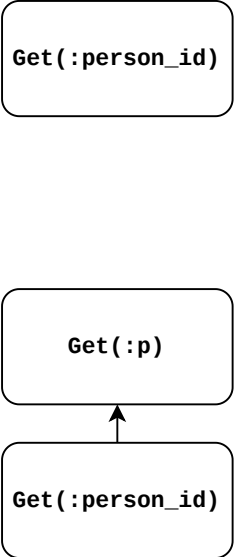
```
From(person) |>  
As(:p) |>  
Select(Get.p.person_id)
```



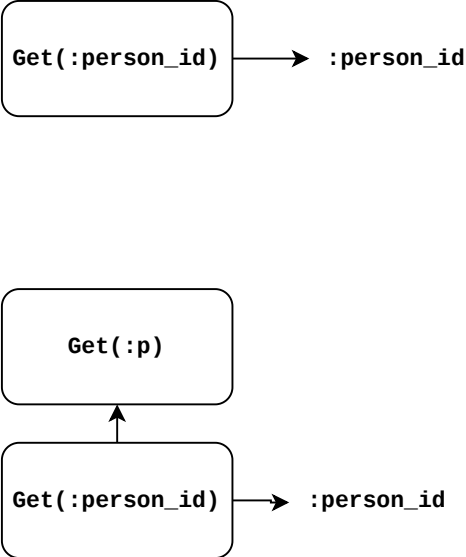
Resolve



refs



repl



New Resolve

