

FunSQL

A library for compositional construction
of SQL queries

<https://github.com/MechanicalRabbit/FunSQL.jl>

Clark C. Evans,
Kyrylo Simonov

JuliaCon 2021

Find all patients born in or after 1970.



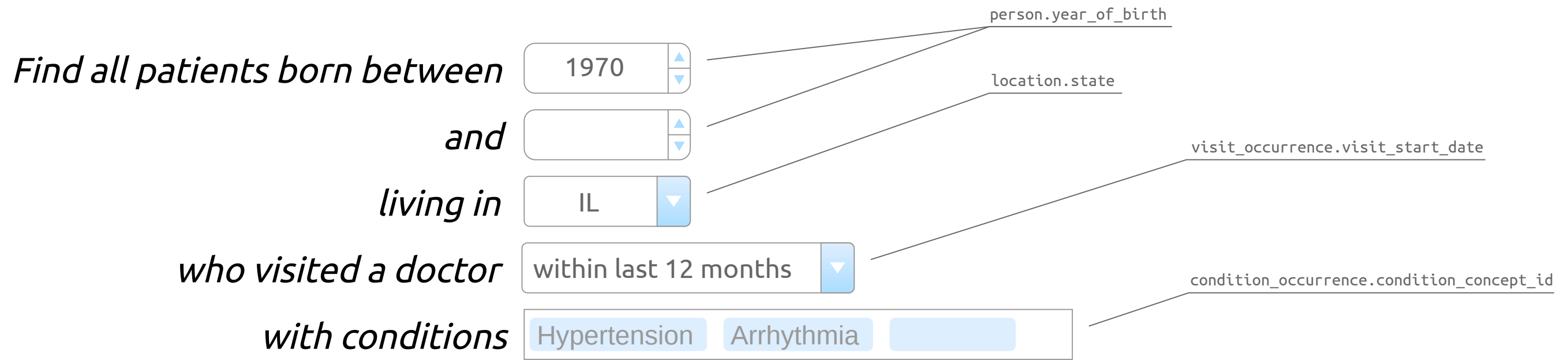
```
SELECT p.person_id  
FROM person p  
WHERE p.year_of_birth >= 1970
```



```
function find_patients(conn)  
  sql = ""  
  SELECT p.person_id  
  FROM person p  
  WHERE p.year_of_birth >= 1970  
  ""  
  DBInterface.execute(conn, sql)  
end
```

Find all patients born between *and*

```
function find_patients(conn; start_year = nothing, end_year = nothing)
  sql = ""
  SELECT p.person_id
  FROM person p
  ""
  predicates = String[]
  if start_year != nothing
    push!(predicates, "p.year_of_birth >= $start_year")
  end
  if end_year != nothing
    push!(predicates, "p.year_of_birth <= $end_year")
  end
  if !isempty(predicates)
    sql *= "\nWHERE " * join(predicates, " AND ")
  end
  DBInterface.execute(conn, sql)
end
```

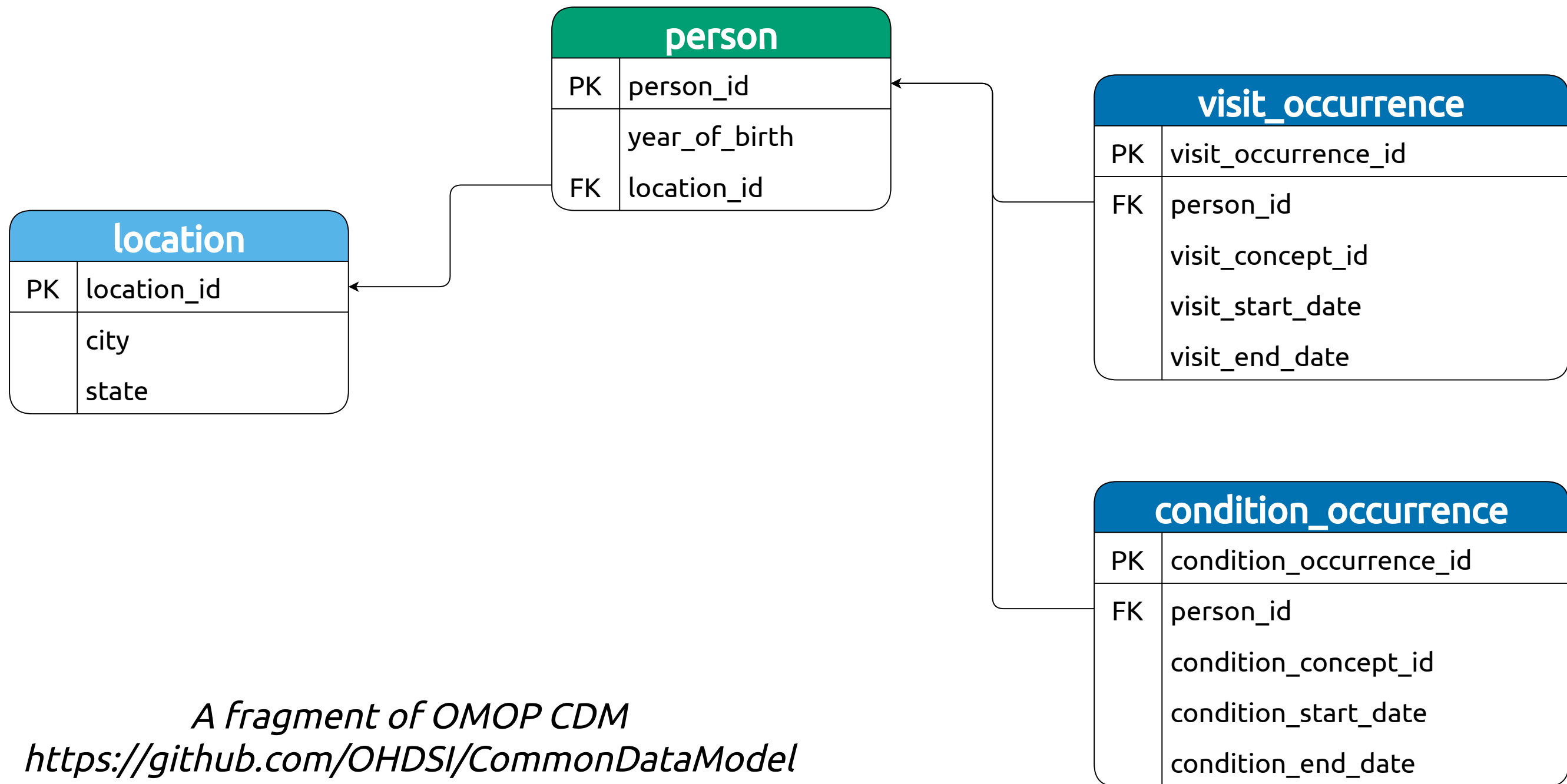


```
function find_patients(conn; start_year = nothing,  
                        end_year = nothing,  
                        state = nothing,  
                        latest_visit_threshold = nothing,  
                        condition_concepts = [])
```

```
    sql = ???
```

```
    DBInterface.execute(conn, sql)
```

```
end
```



A fragment of OMOP CDM
<https://github.com/OHDSI/CommonDataModel>

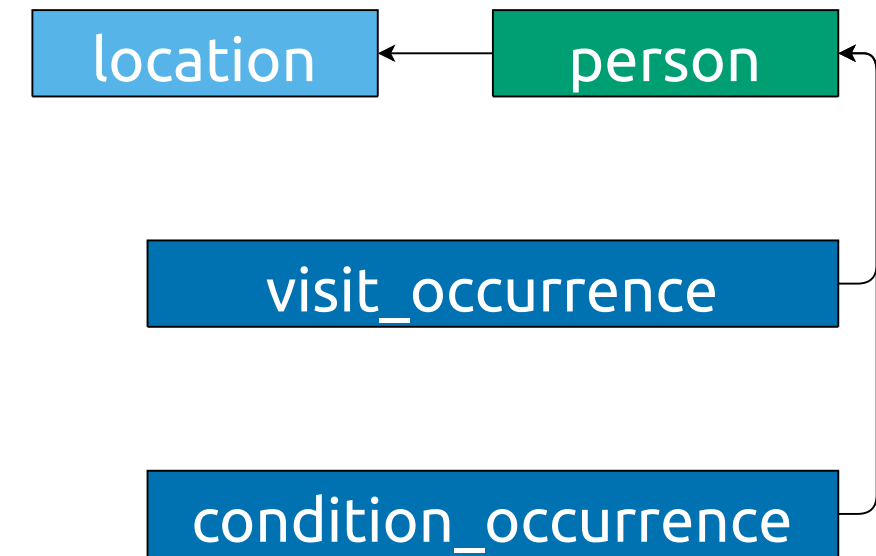
```
using FunSQL: SQLTable
```

```
const person =  
  SQLTable(name = :person,  
    columns = [:person_id, :year_of_birth, :location_id])
```

```
const location =  
  SQLTable(name = :location,  
    columns = [:location_id, :city, :state, :zip])
```

```
const visit_occurrence =  
  SQLTable(name = :visit_occurrence,  
    columns = [:visit_occurrence_id, :person_id, :visit_concept_id,  
      :visit_start_date, :visit_end_date])
```

```
const condition_occurrence =  
  SQLTable(name = :condition_occurrence,  
    columns = [:condition_occurrence_id, :person_id, :condition_concept_id,  
      :condition_start_date, :condition_end_date])
```



Find all patients born in or after 1970.

FROM person p



FROM person p
WHERE p.year_of_birth >= 1970



SELECT p.person_id
FROM person p
WHERE p.year_of_birth >= 1970

using FunSQL: From, Get, Select, Where, render

q = From(person)



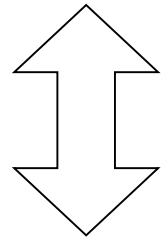
q = From(person) |>
 Where(Get.year_of_birth .>= 1970)



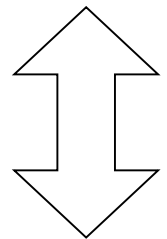
q = From(person) |>
 Where(Get.year_of_birth .>= 1970) |>
 Select(Get.person_id)

sql = render(q, dialect = :postgresql)

```
q1 = From(person)
q2 = q1 |> Where(q1.year_of_birth .>= 1970)
q = q2 |> Select(q2.person_id)
```



```
q = From(person) |>
  Where(Get.year_of_birth .>= 1970) |>
  Select(Get.person_id)
```



```
BornInOrAfter(Y) = Get.year_of_birth .>= Y
```

```
q = From(person) |>
  Where(BornInOrAfter(1970)) |>
  Select(Get.person_id)
```

bound references

unbound references

person	
PK	person_id
	year_of_birth
FK	location_id


```
SELECT p.person_id  
FROM person p  
WHERE p.year_of_birth >= 1970
```

"Fun" notation

using FunSQL: Fun

Fun.">="(Get.year_of_birth, 1970)

or

Get.year_of_birth .>= 1970

broadcasting

```
WHERE p.year_of_birth >= 1970 AND  
      p.year_of_birth <= 2000
```

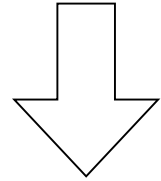
Fun.and(Get.year_of_birth .>= 1970,
 Get.year_of_birth .<= 2000)

```
WHERE p.year_of_birth  
      BETWEEN 1970 AND 2000
```

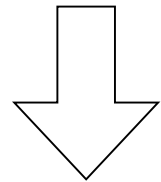
Fun.between(Get.year_of_birth, 1970, 2000)

Show patients with their state of residence.

FROM person p



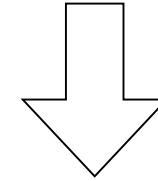
FROM person p
JOIN location l
 ON (p.location_id = l.location_id)



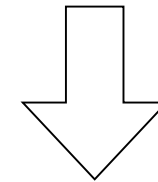
SELECT p.person_id, l.state
FROM person p
JOIN location l
 ON (p.location_id = l.location_id)

using FunSQL: Join

q = From(person)



q = From(person) |>
 Join(:location => location,
 Get.location_id .==
 Get.location.location_id)



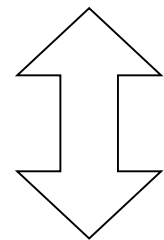
q = From(person) |>
 Join(:location => location,
 Get.location_id .==
 Get.location.location_id) |>
 Select(Get.person_id, Get.location.state)



```

qp = From(person)
ql = From(location)
q  = qp |> Join(ql, qp.location_id .== ql.location_id)

```



Get

- person_id
- year_of_birth
- location_id*
- city
- state

```

q = From(person) |>
  Join(:location => From(location),
       Get.location_id .== Get.location.location_id)

```

Get

- person_id
- year_of_birth
- location_id
- location**
 - location_id
 - city
 - state

person	
PK	person_id
	year_of_birth
FK	location_id

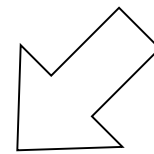
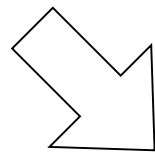
location	
PK	location_id
	city
	state

Find patients

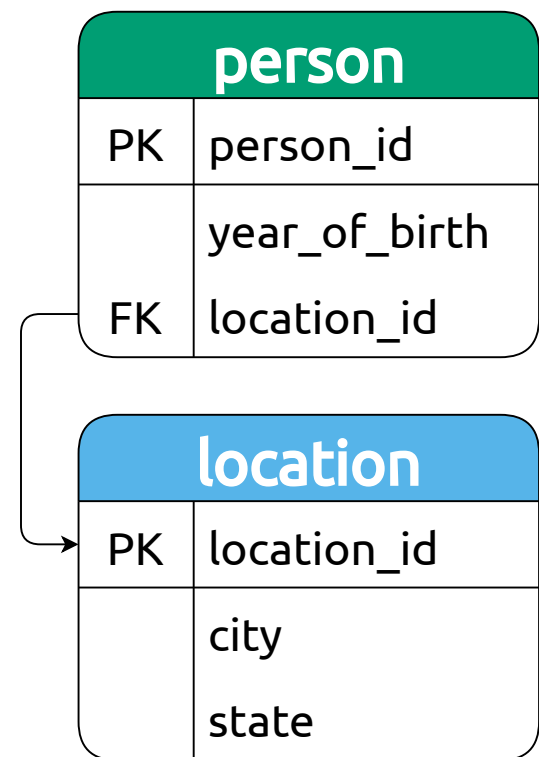
- *born in or after 1970*
- *living in Illinois*

```
qp = From(person) |>  
  Where(Get.year_of_birth .>= 1970)
```

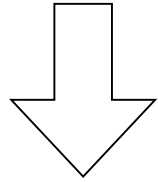
```
ql = From(location) |>  
  Where(Get.state .== "IL")
```



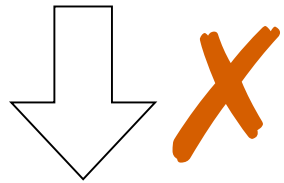
```
qp |> Join(:location => ql,  
           Get.location_id .==  
           Get.location.location_id)
```



FROM person p

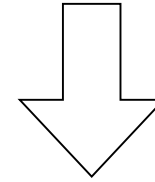


FROM person p
WHERE p.year_of_birth >= 1970

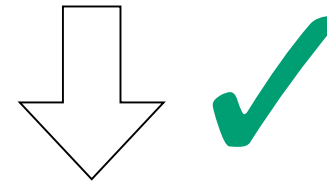


FROM person p
WHERE p.year_of_birth >= 1970
JOIN location l
 ON (p.location_id = l.location_id)

From(person)



From(person) |>
Where(Get.year_of_birth .>= 1970)



From(person) |>
Where(Get.year_of_birth .>= 1970) |>
Join(:location => From(location),
 Get.location_id .==
 Get.location.location_id)



From(*table*)

```
SELECT ???  
FROM $table
```

|>
Where(*condition*)

```
SELECT ???  
FROM ()  
WHERE $condition
```

|>
Join(, *condition*)

```
SELECT ???  
FROM ()  
JOIN ()  
ON $condition
```

|>
Select(*list...*)

```
SELECT $(list...)  
FROM ()
```

SELECT ???
FROM person

SELECT ???
FROM location

SELECT ???
FROM () p
WHERE p.year_of_birth >= 1970

SELECT ???
FROM () l
WHERE l.state = 'IL'

```
q1 = From(person)
q2 = q1 |> Where(q1.year_of_birth .>= 1970)
q3 = From(location)
q4 = q3 |> Where(q3.state .== "IL")
q5 = q2 |> Join(q4, q2.location_id .==
               q4.location_id)
q6 = q5 |> Select(q5.person_id)
```

SELECT ???
FROM () p
JOIN () l
ON p.location_id = l.location_id

SELECT p.person_id
FROM () p


```
SELECT person_id, year_of_birth, location_id
FROM person
```

```
SELECT location_id, state
FROM location
```

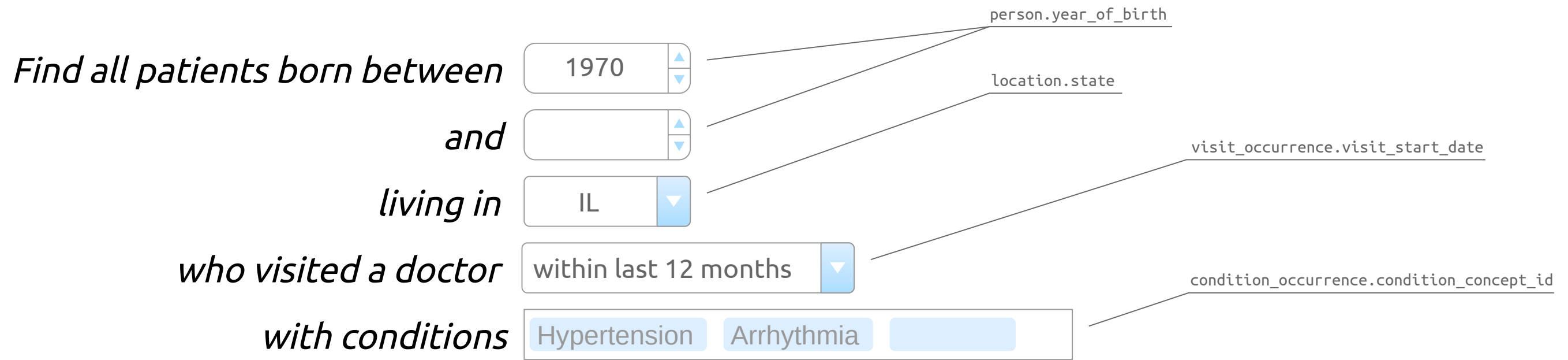
```
SELECT p.person_id, p.location_id
FROM ( ) p
WHERE p.year_of_birth >= 1970
```

```
SELECT l.location_id
FROM ( ) l
WHERE l.state = 'IL'
```

```
q1 = From(person)
q2 = q1 |> Where(q1.year_of_birth .>= 1970)
q3 = From(location)
q4 = q3 |> Where(q3.state .== "IL")
q5 = q2 |> Join(q4, q2.location_id .==
               q4.location_id)
q6 = q5 |> Select(q5.person_id)
```

```
SELECT p.person_id
FROM ( ) p
JOIN ( ) l
  ON p.location_id = l.location_id
```

```
SELECT p.person_id
FROM ( ) p
```



```
function find_patients(conn; start_year = nothing,  
                        end_year = nothing,  
                        state = nothing,  
                        latest_visit_threshold = nothing,  
                        condition_concepts = [])  
  q = FindPatients(; start_year, end_year, state,  
                    latest_visit_threshold, condition_concepts)  
  sql = render(q, dialect = :postgresql)  
  DBInterface.execute(conn, sql)  
end
```

```

FindPatients(; start_year = nothing, end_year = nothing,
              state = nothing,
              latest_visit_threshold = nothing,
              condition_concepts = []) =
  From(person) |>
  FilterByYearOfBirth(; start_year, end_year) |>
  FilterByState(; state) |>
  FilterByLatestVisit(; latest_visit_threshold) |>
  FilterByConditions(; condition_concepts)

FilterByYearOfBirth(; start_year, end_year) =
  if start_year !== nothing && end_year !== nothing
    Where(Fun.between(Get.year_of_birth, start_year, end_year))
  elseif start_year !== nothing
    Where(Get.year_of_birth .>= start_year)
  elseif end_year !== nothing
    Where(Get.year_of_birth .<= end_year)
  else
    identity
  end

```

```

FilterByState(; state) =
  if state != nothing
    Join(:location => From(location) |>
      Where(Get.state .== state),
      Get.location_id .== Get.location.location_id)
  else
    identity
  end

```

```

FilterByLatestVisit(; latest_visit_threshold) =
  if latest_visit_threshold != nothing
    Join(:visit_group => From(visit_occurrence) |>
      Group(Get.person_id),
      Get.person_id .== Get.visit_group.person_id) |>
    Define(:latest_visit_date => Agg.max(Get.visit_start_date, over = Get.visit_group)) |>
    Where(Fun.today() .- Get.latest_visit_date .<= latest_visit_threshold)
  else
    identity
  end

```

```
ConditionsByPerson(person_id; condition_concepts) =  
  From(condition_occurrence) |>  
  Where(Fun.and(Fun.in(Get.condition_concept_id, condition_concepts...),  
                Get.person_id .== Var.person_id)) |>  
  Bind(:person_id => person_id)  
  
FilterByConditions(; condition_concepts) =  
  if !isempty(condition_concepts)  
    Where(Fun.exists(ConditionsByPerson(Get.person_id, condition_concepts)))  
  else  
    identity  
  end
```