

FunSQL

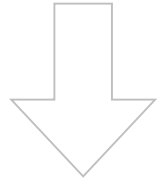
A library for compositional construction
of SQL queries

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

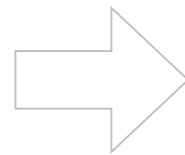
Clark C. Evans,
Kyrylo Simonov

JuliaCon 2021

Find all patients born in or after 1970.



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

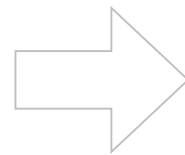


```
function find_patients(conn)  
  sql = ""  
  SELECT p.person_id  
  FROM person p  
  WHERE p.year_of_birth >= 1970  
  ""  
  DBInterface.execute(conn, sql)  
end
```

Find all patients born in or after 1970.

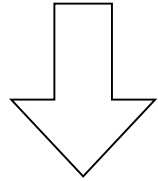


```
SELECT p.person_id  
FROM person p  
WHERE p.year_of_birth >= 1970
```

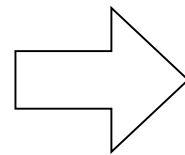


```
function find_patients(conn)  
  sql = ""  
  SELECT p.person_id  
  FROM person p  
  WHERE p.year_of_birth >= 1970  
  ""  
  DBInterface.execute(conn, sql)  
end
```

Find all patients born in or after 1970.



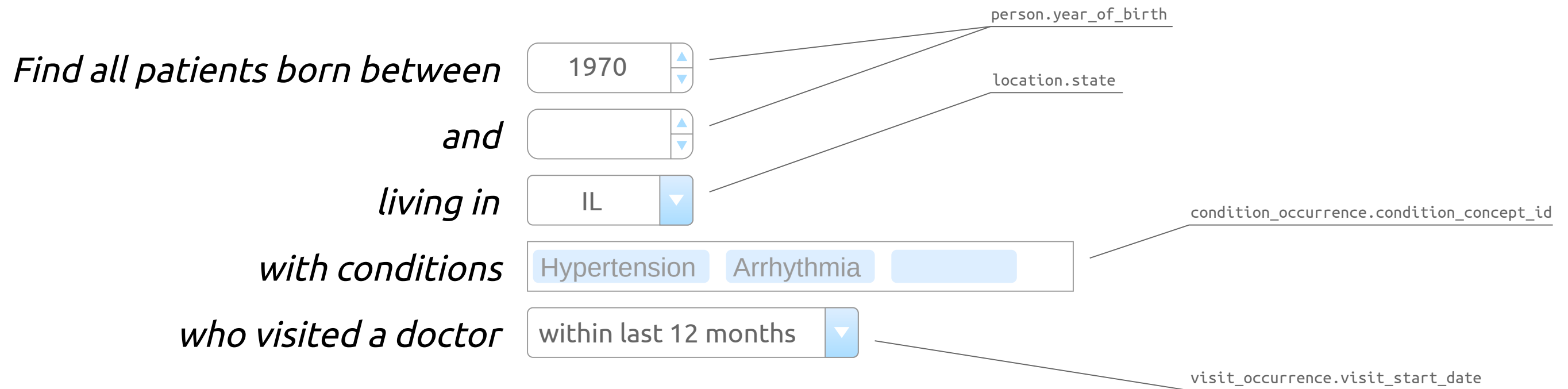
```
SELECT p.person_id  
FROM person p  
WHERE p.year_of_birth >= 1970
```



```
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
```

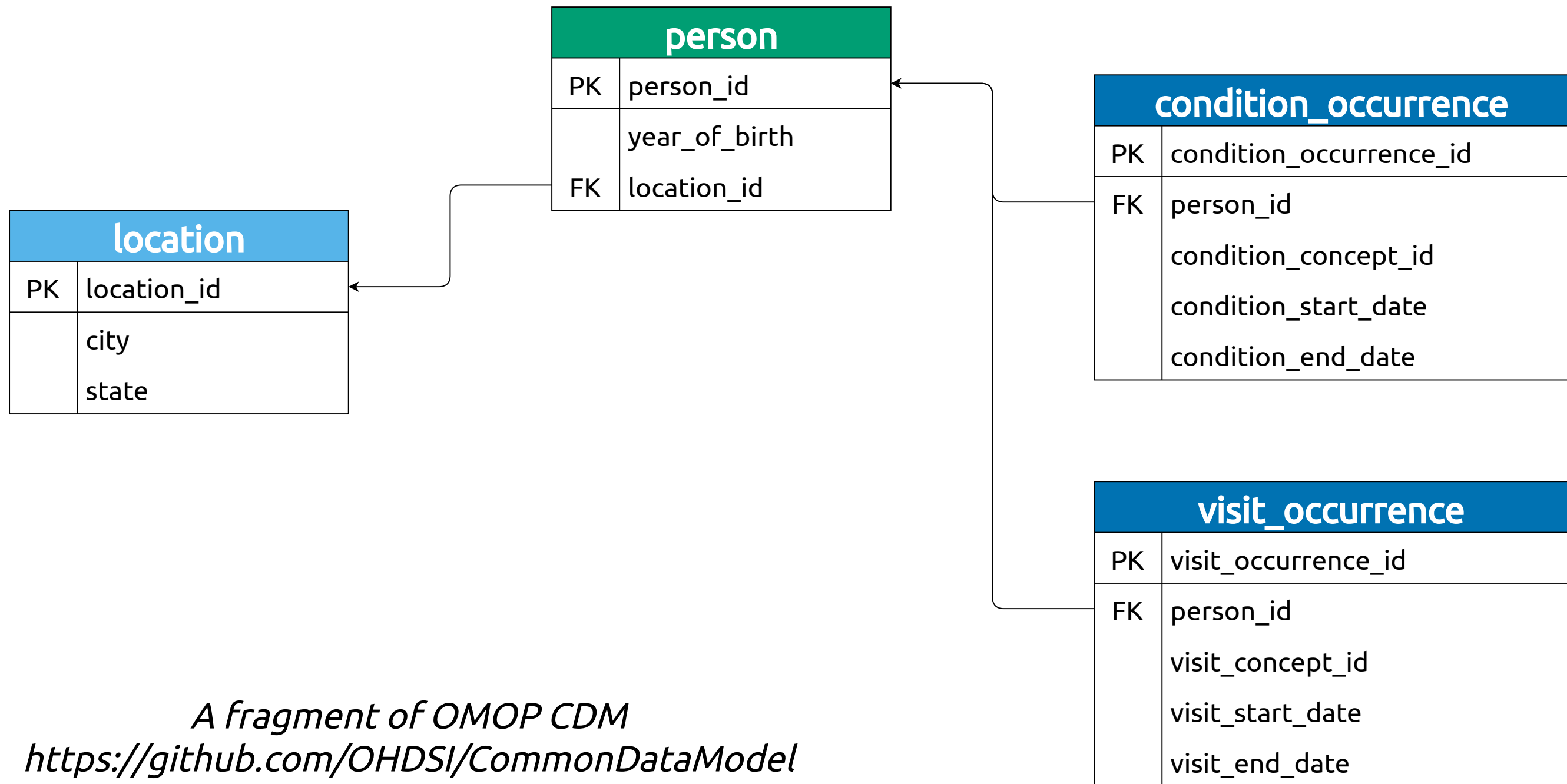


```
function find_patients(conn; start_year = nothing,  
                        end_year = nothing,  
                        state = nothing,  
                        conditions = [],  
                        latest_visit = nothing)
```

```
    sql = ???
```

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

```
end
```



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

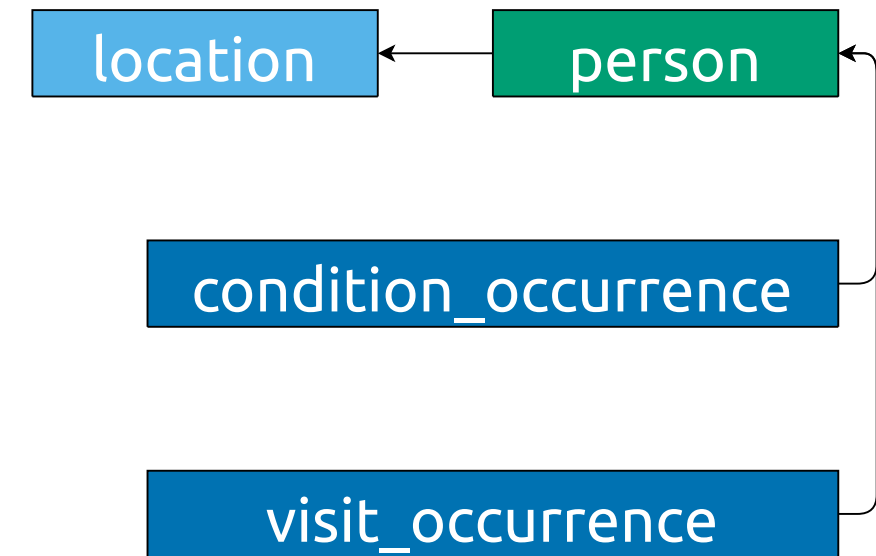
```
using FunSQL: SQLTable
```

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

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

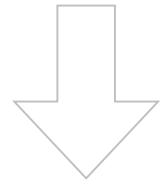
```
const condition_occurrence =  
  SQLTable(name = :condition_occurrence,  
    columns = [:condition_occurrence_id, :person_id, :condition_concept_id,  
      :condition_start_date, :condition_end_date])
```

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

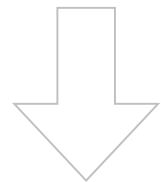


Find all patients born in or after 1970.

FROM person p



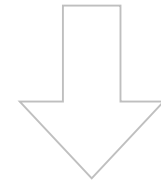
FROM person p
WHERE p.year_of_birth >= 1970



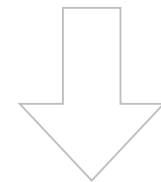
SELECT p.person_id
FROM person p
WHERE p.year_of_birth >= 1970

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

q = From(person)



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

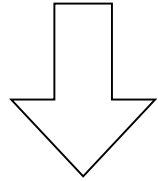


q = From(person) |>
 Where(Get.year_of_birth .>= 1970) |>
 Select(Get.person_id)

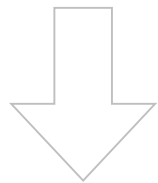
sql = render(q, dialect = :postgresql)

Find all patients born in or after 1970.

FROM person p



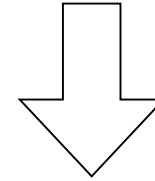
FROM person p
WHERE p.year_of_birth >= 1970



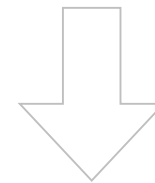
SELECT p.person_id
FROM person p
WHERE p.year_of_birth >= 1970

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)

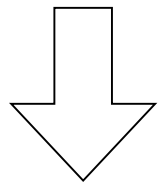
sql = render(q, dialect = :postgresql)

Find all patients born in or after 1970.

FROM person p



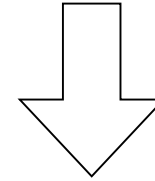
FROM person p
WHERE p.year_of_birth >= 1970



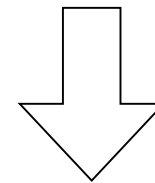
SELECT p.person_id
FROM person p
WHERE p.year_of_birth >= 1970

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)