

# 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
```



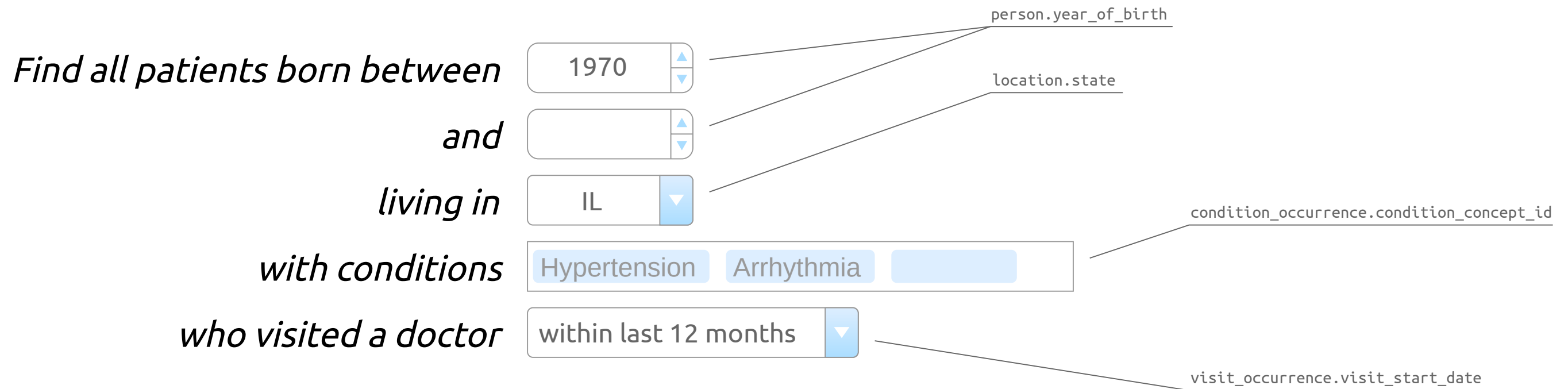
```
function find_patients(conn)  
  sql = ""  
  SELECT p.person_id  
  FROM person p  
  WHERE p.year_of_birth >= 1970  
  ""  
  DBInterface.execute(conn, sql)  
end
```

*Find all patients born between*

1970

*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>

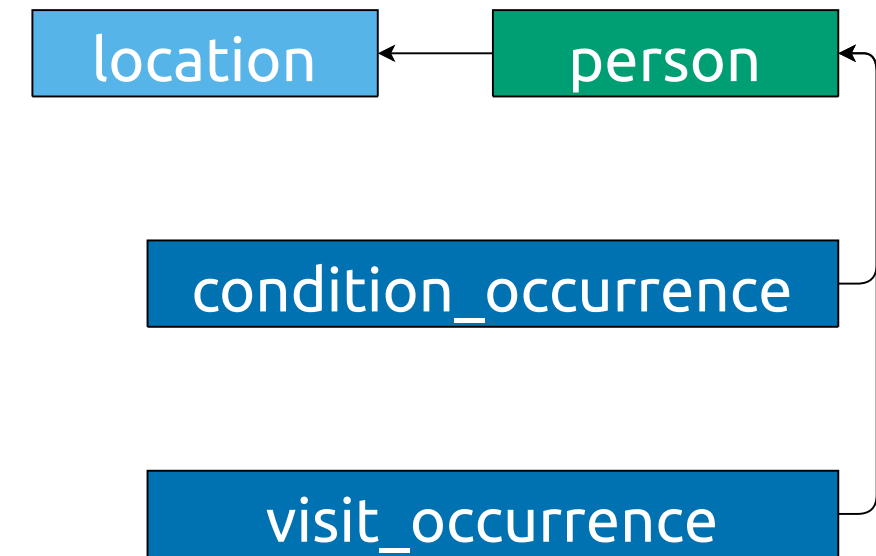
```
using FunSQL: SQLTable
```

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

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

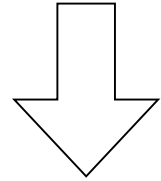
```
const condition_occurrence =  
  SQLTable(name = :condition_occurrence,  
    columns = [:condition_occurrence_id, :person_id, :condition_concept_id,  
      :condition_start_date, :condition_end_date])
```

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

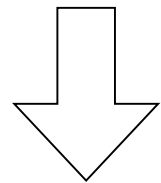


*Find all patients born in or after 1970.*

**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)



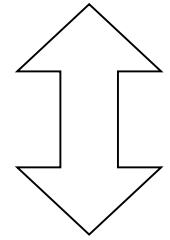
q = From(person) |>  
 Where(Get.year\_of\_birth .>= 1970) |>  
 Select(Get.person\_id)

sql = render(q, dialect = :postgresql)

```

q1 = From(person)
q2 = q1 |> Where(q1.year_of_birth .>= 1970)
q3 = q2 |> Select(q2.person_id)

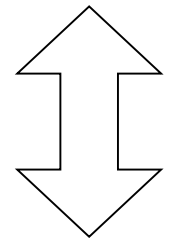
```



```

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
```

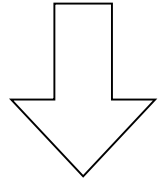
Fun.and(Get.year\_of\_birth .>= 1970,  
 Get.year\_of\_birth .<= 2000)

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

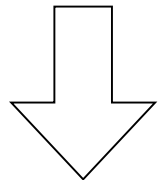
Fun.between(Get.year\_of\_birth, 1970, 2000)

*Show patients with their state of residence.*

**FROM** person p



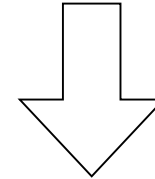
**FROM** person p  
**JOIN** location l  
  **ON** (p.location\_id = l.location\_id)



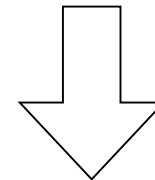
**SELECT** p.person\_id, l.state  
**FROM** person p  
**JOIN** location l  
  **ON** (p.location\_id = l.location\_id)

**using** FunSQL: Join

q = From(person)



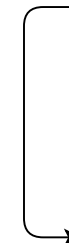
q = From(person) |>  
  Join(:location => location,  
      Get.location\_id .==  
      Get.location.location\_id)



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

person	
PK	person_id
	year_of_birth
FK	location_id

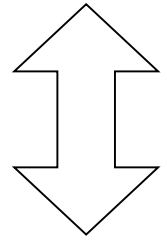
location	
PK	location_id
	city
	state



```

q1 = From(person)
q2 = From(location)
q3 = q1 |> Join(q2, q1.location_id .== q2.location_id)

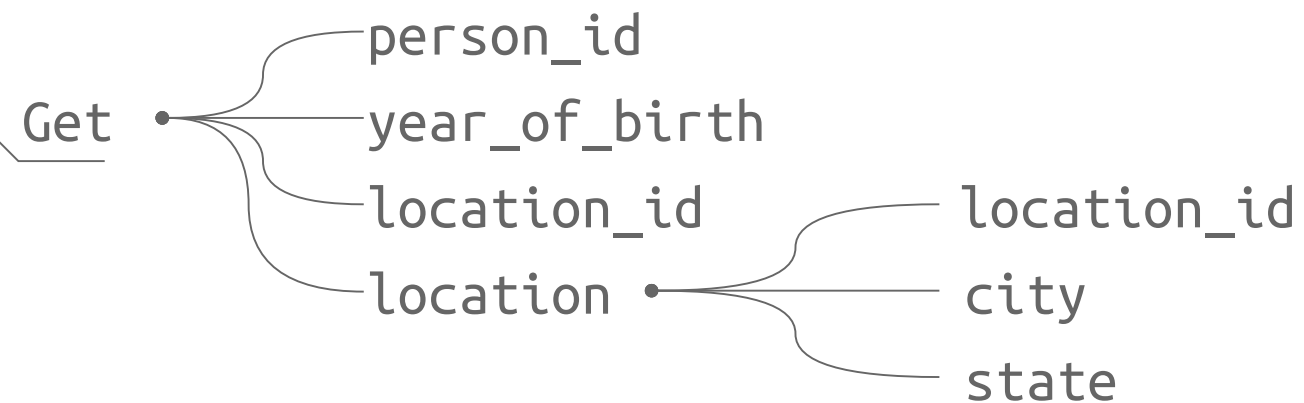
```



```

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

```



person	
PK	person_id
	year_of_birth
FK	location_id

location	
PK	location_id
	city
	state

