

# THESIS RESULTS PHASE 1

LUCA CASSENTI

# TABLE OF CONTENT

**01 - SUMMARY**

**04 - QUERIES**

**02 - DATABASE**

**05 - NEXT STEPS**

**03 - DATA & SCRIPTS**

# Summary



This first phase of the thesis focused on the collection, analysis, and refinement of data concerning mutations within the avian flu virus. At first only for the H5N1 strain, but then involving every strain available.

This phase was mostly about data cleaning and preliminary investigations to establish a robust foundation for the creation of the final tool with the objective to decipher the evolution of the virus, with focus on factors such as transmission dynamics, virulence traits, and possible implications for vaccine development.

Work started with an analysis of requirements given by the IZSVe, with which we collaborate, to clearly define the scope of the project. Then proceed with the necessary analysis of the data, the creation of a database schema and the aggregation of data with some example queries.



# TABLE OF CONTENT

**01 - SUMMARY**

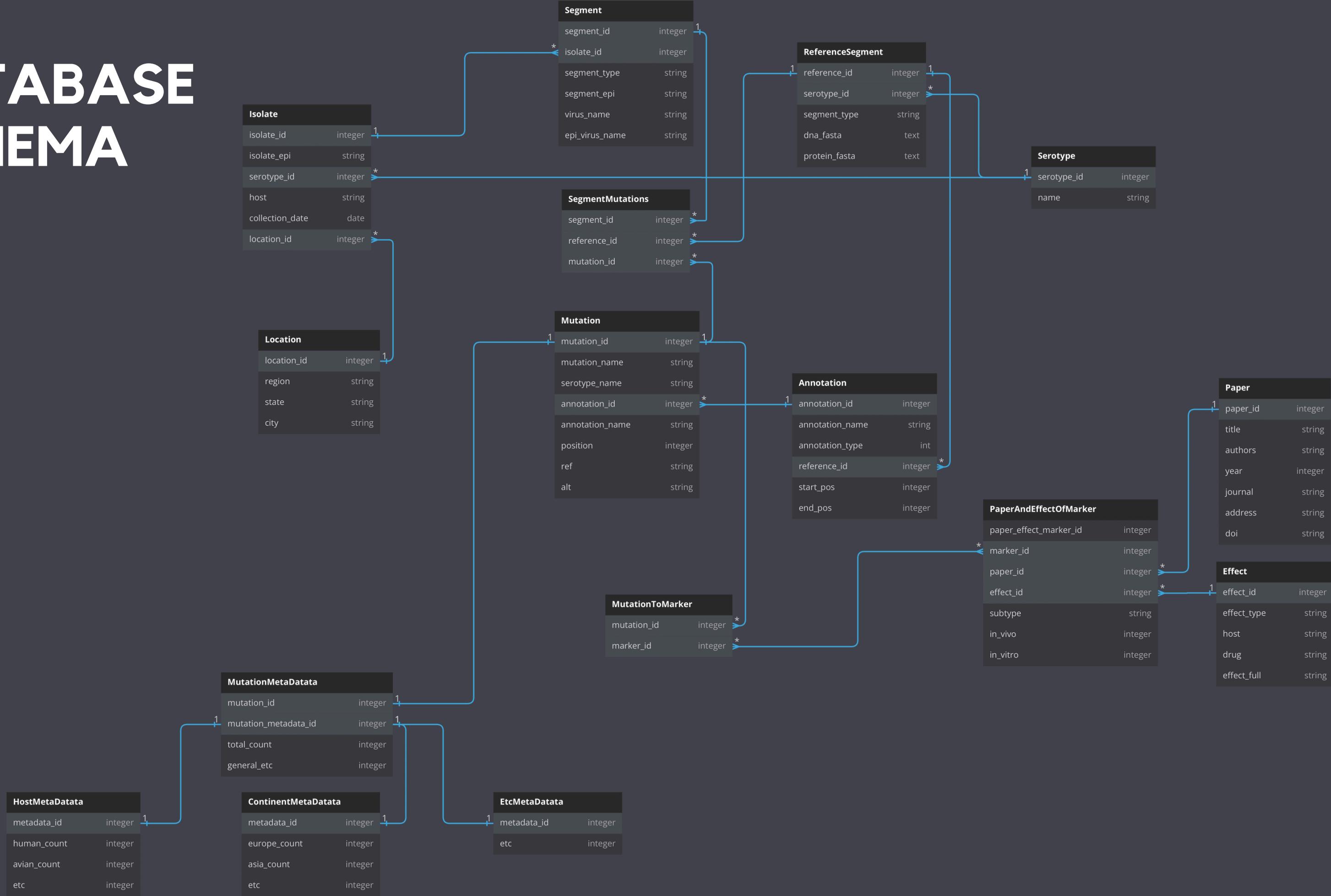
**04 - QUERIES**

**02 - DATABASE**

**05 - NEXT STEPS**

**03 - DATA & SCRIPTS**

# DATABASE SCHEMA



# MAIN FEATURES

- COMPLETE GRASP OF THE FLU DOMAIN
- SEARCH ON MUTATIONS, GROUPS OF, AND ANY MAIN CONCEPT
- QUERY WITH STATISTICAL RELEVANCE
- USE OF SECONDARY STRUCTURES TO HASTEN QUERIES
- COMPLETE INTEGRATION OF IZSVE INITIAL WORK

# EXCLUDED BRIDGE TABLE

WHEN DESIGNING THE DATABASE SCHEMA, THE FOLLOWING BRIDGE TABLE FOR SEGMENTS TO REFERENCES WAS NOT INCLUDED AS FROM A PRACTICAL STANDPOINT, IT WOULD HAVE UNNECESSARILY COMPLICATED THE QUERIES.

SegmentsToReferences	
segment_id	integer
reference_id	integer

DESPITE THAT IT IS IMPORTANT TO CONSIDER IT LOGICALLY AS TO BETTER REPRESENT THE DOMAIN.

IN FACT, ALTHOUGH IT WOULD BE POSSIBLE TO RECONSTRUCT THE TABLE FROM A JOIN CLAUSE, THE LOGICAL BOND BETWEEN THE 'SEGMENT' AND THE 'REFERENCE SEGMENTS' ENTITIES HAS A BIG SIGNIFICANCE IN THE RESEARCH OF MUTATIONS.

# TABLE OF CONTENT

**01 - SUMMARY**

**04 - QUERIES**

**02 - DATABASE**

**05 - NEXT STEPS**

**03 - DATA & SCRIPTS**

# Three Types of Data

annotation.xlsx

annotation_id	annotation_name
0	PB2
1	PB1
2	PB1-F2
3	PA
4	PA-X
5	PA-X
6	HA1
7	HA2
8	NP

01 - STATIC VIRUS DATA

>EPI1530142|PB2|A/ruddy\_turnstone/Delaware/2009  
caaatatattcaatataatggagagaataaaagaattaaga  
accactgttggaccacatggccataattaaaaagtacac  
gatgtatggcaatgaaatataccaatcacagcagacaage  
tcctctggagaaaacaaacgatgccgggtcagaccga  
ggaccaacaacaaggtaatcactatccaaaggtaata  
cttggccctgtacacttcagaaatcaagttaagataaa  
ccaaagaggcgccaggatgtaatcatggaaagtgttt  
ttgacaataacaaaggagaagaaggaagaactccaggaa  
agagctggtccgaaagacaagggtttctcccagtggctt  
aagggacatgctgggagcagatgtacactccgggaggaa  
gccaggaacatagtaagaaggcaacagtgtcagcaga  
tggaggaataagaatggtagacattttcggcaaaatc  
gcttgaggattagctcatccttcagcttgggttgtt  
gaagtgcattacggcaacccatccaacattgaaaataaa

02 - MUTATION DATA



03 - PAPERS DATA

# DATA FROM GISAID

We downloaded the flu data from GISAID, a global database that provides detailed genomic data. GISAID categorizes flu data by types (A, B, C), with our focus exclusively on type A. The database further distinguishes data by the hemagglutinin (H) and neuraminidase (N) segments, among other filters. For our analysis, we specifically filtered the data based on the H and N segment numbers, such as H5N1, ensuring our dataset was precise and relevant to our study objectives.

In particular the GISAID database has a total of ~450K Isolates for the type A, for a total of ~2.2 million Segment sequences, which roughly translates to 5 Segments per Isolates, despite each Isolate could have as much as 8 Segments (HA, NA, PB2, PB1, PA, NP, NS and MP).

The most common hemagglutinin and neuraminidase types are the H1N1, which accounts for ~150K Isolates, and the H3N2, with ~190K Isolates. Third place goes to H5N1 with ~20K Isolates, well beyond the other two.

The screenshot shows the GISAID search interface with various filters applied:

- Type:** A (selected)
- H:** 1 (selected)
- N:** 1 (selected)
- Lineage:** All
- Host:** Human
- Location:** All
- Clades:** 0

**Pathogenicity:** All

**Additional filters:**

- Collection date (YYYY-MM-DD):** From [ ] To [ ]
- Submission date (YYYY-MM-DD):** From [ ] To [ ]
- Originating Laboratory:** [Afghanistan, Kabul] National Public Health Laboratory, [Albania, Tirana] Institute of Public Health, [Algeria, Algiers] Institut Pasteur d'Algérie, [American Samoa, Faga'alu] LBJ Tropical Medicine Centre, [Andorra, a] a, [Argentina, Buenos Aires] Instituto Nacional de Enfermedades Infecciosas C.G. Malbrán
- Submitting Laboratory:** [Algeria, Algiers] Institut Pasteur d'Algérie, [Argentina, Buenos Aires] Instituto Nacional de Enfermedades Infecciosas C.G. Malbrán, [Argentina, Buenos Aires] Instituto Nacional de Enfermedades Infecciosas Dr. C.G. Malbrán

Help Total: 437,164 viruses (2,195,010 sequences)

Charts Reset S

# DATA FROM GISAID

Each Isolate can be downloaded in different ways to extract different information:

- It is possible to either extract its metadata, which includes data such as the available segments for that isolate, its subtype (ex: H1H5), where it was retrieved, from which host, etc. The totality of the metadata is composed by qualitative data, thus a quantitative analysis cannot be performed
- Otherwise it is possible to extract either the DNA (nucleotidic) or Protein sequences. This data is the exact sequence that was extracted from the isolate and is divided by the different segments. It is this kind of data that is then used to find mutations in the given Isolate by comparing the sequences with their respective references, divided by segment type and subtype.

A1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
9807	EPI_ISL_653 EPI228907 A EPI228906 A EPI228905 A EPI228900 A EPI228903 A EPI228902 A EPI228901 A EPI228904 A	A/duck/South_A / HSN1	EA_nonGsGD	Africa / South Duck													
9808	EPI_ISL_653 EPI228789 A EPI228824 A EPI228819 A EPI228830 A EPI228801 A EPI228795 A EPI228813 A	A/duck/Vietnam_A / HSN1	1	Asia / Vietnam Duck													
9809	EPI_ISL_653	EPI228744 A	A/Duck/Hainan_A / HSN1	Asia / Thalain Duck													
9810	EPI_ISL_653	EPI228627 A	A/duck/Chon_A / HSN1	Asia / Thalain Duck													
9811	EPI_ISL_652	EPI228633 A EPI228737 A EPI228699 A	A/duck/Sabah_A / HSN1	1	Asia / Thalain Duck												
9812	EPI_ISL_652	EPI228625 A EPI228601 A EPI228735 A EPI228697 A	A/duck/Neblio_A / HSN1	1	Asia / Thalain Duck												
9813	EPI_ISL_652	EPI228635 A EPI228709 A EPI228671 A	A/duck/Chon_A / HSN1	1	Asia / Thalain Duck												
9814	EPI_ISL_652	EPI228554 A	A/duck/Iltara_A / HSN1	1	embryonated Asia / Thalain Duck												
9815	EPI_ISL_652	EPI228562 A	A/duck/Uthalai_A / HSN1	1	embryonated Asia / Thalain Duck												
9816	EPI_ISL_652	EPI228549 A EPI228576 A	A/duck/Nakor_A / HSN1	1	embryonated Asia / Thalain Duck												
9817	EPI_ISL_652	EPI228537 A	A/duck/Hainan_A / HSN1	1	embryonated Asia / Thalain Duck												
9818	EPI_ISL_652	EPI228535 A	EPI228559 A EPI229020 A	A/duck/Pitaha_A / HSN1	1	embryonated Asia / Thalain Duck											
9819	EPI_ISL_652	EPI228534 A	EPI228575 A EPI229027 A	A/duck/Pinch_A / HSN1	1	embryonated Asia / Thalain Duck											
9820	EPI_ISL_652	EPI228533 A	EPI228569 A EPI229025 A	A/duck/Chich_A / HSN1	1	embryonated Asia / Thalain Duck											
9821	EPI_ISL_652	EPI228532 A	A/duck/Chon_A / HSN1	1	embryonated Asia / Thalain Duck												
9822	EPI_ISL_652	EPI228521 A	A/duck/Phasa_A / HSN1	1	embryonated Asia / Thalain Duck												
9823	EPI_ISL_651	EPI228368 A EPI234634 A	A/muscovy_d_A / HSN1	1	Asia / Vietnamese Duck												
9824	EPI_ISL_324 EP1186124 8 EP1186398 8 EP1186123 8 EP1186118 8 EP1186120 8 EP1186119 8 EP1186122 8	A/duck/German_A / HSN1	2.2	e1 Europe / Gerr Duck													
9825	EPI_ISL_294 EP1156378 A EP1156380 A EP1156381 A EP1156382 A EP1156383 A EP1156384 A EP1156385 A EP1156386 A	A/mallard/Fra_A / HSN1	9	Europe / Fran/Asas plat													
9826	EPI_ISL_653 EP1228852 A EP1228851 A EP1228850 A EP1228853 A EP1228847 A EP1228846 A EP1228849 A EP1228848 A	A/mallard/Gus_A / HSN1	original	Asia / China Anas plat													
9827	EPI_ISL_325 EP1186397 sk EP1186399 sk EP1186395 sk	A/Anas platyrhynchos_A / HSN1	original	Europe / Slov/Anas plat													
9828	EPI_ISL_104 EP11847697 EP11847698 EP11847699 EP11847700 EP11847701 EP11847702 EP11847703 EP11847704 EP11847705 EP11847706 EP11847707 EP11847708 EP11847709 EP11847710 EP11847711 EP11847712 EP11847713 EP11847714 EP11847715 EP11847716 EP11847717 EP11847718 EP11847719 EP11847720 EP11847721 EP11847722 EP11847723 EP11847724 EP11847725 EP11847726 EP11847727 EP11847728 EP11847729 EP11847730 EP11847731 EP11847732 EP11847733 EP11847734 EP11847735 EP11847736 EP11847737 EP11847738 EP11847739 EP11847740 EP11847741 EP11847742 EP11847743 EP11847744 EP11847745 EP11847746 EP11847747 EP11847748 EP11847749 EP11847750 EP11847751 EP11847752 EP11847753 EP11847754 EP11847755 EP11847756 EP11847757 EP11847758 EP11847759 EP11847760 EP11847761 EP11847762 EP11847763 EP11847764 EP11847765 EP11847766 EP11847767 EP11847768 EP11847769 EP11847770 EP11847771 EP11847772 EP11847773 EP11847774 EP11847775 EP11847776 EP11847777 EP11847778 EP11847779 EP11847780 EP11847781 EP11847782 EP11847783 EP11847784 EP11847785 EP11847786 EP11847787 EP11847788 EP11847789 EP11847790 EP11847791 EP11847792 EP11847793 EP11847794 EP11847795 EP11847796 EP11847797 EP11847798 EP11847799 EP11847700 EP11847701 EP11847702 EP11847703 EP11847704 EP11847705 EP11847706 EP11847707 EP11847708 EP11847709 EP11847710 EP11847711 EP11847712 EP11847713 EP11847714 EP11847715 EP11847716 EP11847717 EP11847718 EP11847719 EP11847720 EP11847721 EP11847722 EP11847723 EP11847724 EP11847725 EP11847726 EP11847727 EP11847728 EP11847729 EP11847730 EP11847731 EP11847732 EP11847733 EP11847734 EP11847735 EP11847736 EP11847737 EP11847738 EP11847739 EP11847740 EP11847741 EP11847742 EP11847743 EP11847744 EP11847745 EP11847746 EP11847747 EP11847748 EP11847749 EP11847750 EP11847751 EP11847752 EP11847753 EP11847754 EP11847755 EP11847756 EP11847757 EP11847758 EP11847759 EP11847760 EP11847761 EP11847762 EP11847763 EP11847764 EP11847765 EP11847766 EP11847767 EP11847768 EP11847769 EP11847770 EP11847771 EP11847772 EP11847773 EP11847774 EP11847775 EP11847776 EP11847777 EP11847778 EP11847779 EP11847780 EP11847781 EP11847782 EP11847783 EP11847784 EP11847785 EP11847786 EP11847787 EP11847788 EP11847789 EP11847790 EP11847791 EP11847792 EP11847793 EP11847794 EP11847795 EP11847796 EP11847797 EP11847798 EP11847799 EP11847700 EP11847701 EP11847702 EP11847703 EP11847704 EP11847705 EP11847706 EP11847707 EP11847708 EP11847709 EP11847710 EP11847711 EP11847712 EP11847713 EP11847714 EP11847715 EP11847716 EP11847717 EP11847718 EP11847719 EP11847720 EP11847721 EP11847722 EP11847723 EP11847724 EP11847725 EP11847726 EP11847727 EP11847728 EP11847729 EP11847730 EP11847731 EP11847732 EP11847733 EP11847734 EP11847735 EP11847736 EP11847737 EP11847738 EP11847739 EP11847740 EP11847741 EP11847742 EP11847743 EP11847744 EP11847745 EP11847746 EP11847747 EP11847748 EP11847749 EP11847750 EP11847751 EP11847752 EP11847753 EP11847754 EP11847755 EP11847756 EP11847757 EP11847758 EP11847759 EP11847760 EP11847761 EP11847762 EP11847763 EP11847764 EP11847765 EP11847766 EP11847767 EP11847768 EP11847769 EP11847770 EP11847771 EP11847772 EP11847773 EP11847774 EP11847775 EP11847776 EP11847777 EP11847778 EP11847779 EP11847780 EP11847781 EP11847782 EP11847783 EP11847784 EP11847785 EP11847786 EP11847787 EP11847788 EP11847789 EP11847790 EP11847791 EP11847792 EP11847793 EP11847794 EP11847795 EP11847796 EP11847797 EP11847798 EP11847799 EP11847700 EP11847701 EP11847702 EP11847703 EP11847704 EP11847705 EP11847706 EP11847707 EP11847708 EP11847709 EP11847710 EP11847711 EP11847712 EP11847713 EP11847714 EP11847715 EP11847716 EP11847717 EP11847718 EP11847719 EP11847720 EP11847721 EP11847722 EP11847723 EP11847724 EP11847725 EP11847726 EP11847727 EP11847728 EP11847729 EP11847730 EP11847731 EP11847732 EP11847733 EP11847734 EP11847735 EP11847736 EP11847737 EP11847738 EP11847739 EP11847740 EP11847741 EP11847742 EP11847743 EP11847744 EP11847745 EP11847746 EP11847747 EP11847748 EP11847749 EP11847750 EP11847751 EP11847752 EP11847753 EP11847754 EP11847755 EP11847756 EP11847757 EP11847758 EP11847759 EP11847760 EP11847761 EP11847762 EP11847763 EP11847764 EP11847765 EP11847766 EP11847767 EP11847768 EP11847769 EP11847770 EP11847771 EP11847772 EP11847773 EP11847774 EP11847775 EP11847776 EP11847777 EP11847778 EP11847779 EP11847780 EP11847781 EP11847782 EP11847783 EP11847784 EP11847785 EP11847786 EP11847787 EP11847788 EP11847789 EP11847790 EP11847791 EP11847792 EP11847793 EP11847794 EP11847795 EP11847796 EP11847797 EP11847798 EP11847799 EP11847700 EP11847701 EP11847702 EP11847703 EP11847704 EP11847705 EP11847706 EP11847707 EP11847708 EP11847709 EP11847710 EP11847711 EP11847712 EP11847713 EP11847714 EP11847715 EP11847716 EP11847717 EP11847718 EP11847719 EP11847720 EP11847721 EP11847722 EP11847723 EP11847724 EP11847725 EP11847726 EP11847727 EP11847728 EP11847729 EP11847730 EP11847731 EP11847732 EP11847733 EP11847734 EP11847735 EP11847736 EP11847737 EP11847738 EP11847739 EP11847740 EP11847741 EP11847742 EP11847743 EP11847744 EP11847745 EP11847746 EP11847747 EP11847748 EP11847749 EP11847750 EP11847751 EP11847752 EP11847753 EP11847754 EP11847755 EP11847756 EP11847757 EP11847758 EP11847759 EP11847760 EP11847761 EP11847762 EP11847763 EP11847764 EP11847765 EP11847766 EP11847767 EP11847768 EP11847769 EP11847770 EP11847771 EP11847772 EP11847773 EP11847774 EP11847775 EP11847776 EP11847777 EP11847778 EP11847779 EP11847780 EP11847781 EP11847782 EP11847783 EP11847784 EP11847785 EP11847786 EP11847787 EP11847788 EP11847789																

# DB CREATION

---

## 1. DB SCHEMA

```
592 # Segment table
593 create_segment_table = """
594 CREATE TABLE IF NOT EXISTS Segment (
595     segment_id INTEGER PRIMARY KEY, isolate_id INTEGER,
596     segment_type TEXT, segment_epi TEXT, virus_name TEXT, epi_virus_name TEXT UNIQUE,
597     FOREIGN KEY (isolate_id) REFERENCES Isolate(isolate_id));"""
598
599 # Mutation table
600 create_mutation_table = """
601 CREATE TABLE IF NOT EXISTS Mutation (
602     mutation_id INTEGER PRIMARY KEY, mutation_name TEXT UNIQUE,
603     serotype_name TEXT, annotation_id, annotation_name TEXT, position INTEGER,
604     ref TEXT, alt TEXT,
605     FOREIGN KEY (annotation_id) REFERENCES Annotation(annotation_id));"""
606
607 # SegmentMutations table
608 create_segment_mutations_table = """
609 CREATE TABLE IF NOT EXISTS SegmentMutations (
610     segment_id INTEGER, reference_id INTEGER, mutation_id INTEGER,
611     FOREIGN KEY (segment_id) REFERENCES Segment(segment_id),
```

The first step in the creation of the database consists in translating the DB Schema created for the domain into a SQL schema using `CREATE TABLE` commands and by correctly linking the various tables using foreign keys.

## 2. GISAID DATA

Secondly, we need to download all the necessary data from the GISAID database. In particular we needed to download as many Isolates from type A, for a total of ~450K Isolates.

For each Isolate we downloaded the DNA Sequence and its metadata, in order to fill up the database with the necessary information both on the Isolate and its Segments, by using the metadata, and on the Mutations, by using the DNA sequences, by aligning them.

All the data was downloaded manually, as the subdivision by subtype on GISAID was only a slight problem for the H1N1 subtype and the H2N3 subtype.

## 3. REFERENCES

The third step was to obtain the references sequences necessary to obtain the mutation from each sample sequence.

These sequences are what could be considered the ancestors of all the analyzed data, from which all the flu samples have evolved and mutated.

Each sample can have multiple references in theory but usually only one is necessary. Furthermore these references are categorized by subtype and segment type.

In particular, for the H5N1 subtype we used the references provided by the IZSVe

## 4. METADATA INTEGRATION

Before processing the mutations, each sequence is correlated with its metadata, in order to ensure consistency in the database by classifying each sample once and into the proper categories for future queries.

```
>EPI1848284|PB2|A/Chongqing/00013/2021|EPI_ISL_1081369||A / H5N6  
agcgaaaggcaggtaaatatattcagtatgaacagaataaaaagaactaagagatctaattgcacagt  
gatactgacaaagaccactgtggaccatcatggccataatcaaaaaatacacatcaggaagacaggaa  
tcaggatgaaatggatgtggcaatgaaataccgatcacagccgacaaaaagataatggagatgat  
gaacaaggtaaactcttggagcaagacaatgtatgcgggtcagacagagataatggatcaccc  
gttggaaacggaaatggccacaacaaggcacagtccattatccaaagggtgtacaaaacctactttgaa  
taaaaacaaggaaaccttggccgttcattcaggagtcaagtcaaaaatacgcgcagggttgacat  
gccccatctcgtgcggaaaggcacaagatgttataatggagggttggccaaacgaaagttaggag  
ttcagagtccacaatgacaatacaaaggaaaagaaagaagaactccaggattgtaaaggattgtcc  
atatgttggaaagagaactggtcgaagacaagattccatccaggctggctggcgggacaaggcagcgt
```

166	EPI_ISL_2081527	EPI1916407 A/red fox/England/AVP-M1-21-(EPI1916407)
167	EPI_ISL_4063607	EPI1973694 A/mink/China/F0130m/2018 EPI1973694
168	EPI_ISL_16466440	EPI2287045 A/Changsha/1/2022 EPI2287045
169	EPI_ISL_1081370	EPI1848292 A/Anhui/2021-00011/2020 EPI1848292
170	EPI_ISL_1081369	EPI1848284 A/Chongqing/00013/2021 EPI1848284
171	EPI_ISL_180669	
172	EPI_ISL_19055420	EPI3210827 A/large-billed crow/Iwate/0303G EPI3210827
173	EPI_ISL_19055419	EPI3210819 A/large-billed crow/Iwate/0302G EPI3210819
174	EPI_ISL_19055418	EPI3210818 A/large-billed crow/Iwate/0302G EPI3210818

## 5. VARIANT CALLING

The final step in the creation of the database data consists in finding the mutations of each sample using variant calling.

Variant calling is a crucial process in genomic analysis, used to identify genetic differences between sequences to detect mutations. The workflow begins with Augur, a toolkit that aligns the sequences in their DNA form, creating a standardized reference framework. Once aligned, the coding sequences for each protein (ex: M1 and M2 for segment MP) are extracted from these DNA sequences. These coding sequences are then translated into their corresponding amino acid sequences, reflecting the actual proteins produced by the organism.

The next step involves further aligning these amino acid sequences. This detailed alignment allows for the precise identification of mutations by highlighting differences from a reference sequence.

By focusing on the amino acid level, we can better understand the functional impacts of these mutations, as changes in protein structure correspond to biological consequences.

# DB FILL

While the sample data and associated metadata are utilized by the software to create entries for the mutations, the system simultaneously checks and creates entries for the Isolate and Segment tables if they do not already exist.

These tables contain comprehensive information about each sample and its respective genomic segments.

Additionally, the data correlating each sample segment with its identified mutations and the reference sequence used is systematically organized into a bridge table. This bridge table effectively links all relevant information, allowing for efficient tracking and analysis of the relationship between samples, segments, and their mutations.

# **[SOLATES**

EPI_ISL_332135	EPI1324826 A/Barry/
EPI_ISL_332136	EPI1324834 A/Cardiff/
EPI_ISL_332137	
EPI_ISL_332138	EPI1324846 A/Paris/246
EPI_ISL_332139	EPI1324854 A/Dijon/240
EPI_ISL_332140	
EPI_ISL_332141	EPI1338304 A/Krasnoyarsk/
EPI_ISL_332142	
EPI_ISL_332143	EPI1338294 A/Novosibirsk/
EPI_ISL_332144	
EPI_ISL_332145	EPI1337480 A/Berlin/
EPI_ISL_332146	EPI1324844 A/Paris/

# + CROSS REFERENCING WITH METADATA

# SEGMENTS

```
# Check if segment exists
try:
    self.cursor.execute(_sql: "SELECT * FROM Segment WHERE epi_virus_name = ?",
                       _parameters: (f'{header_dict['segment_id']}',))
    segment = self.cursor.fetchone()
    if segment:
        return segment["segment_id"] # return id for SegmentMutations
except sqlite3.OperationalError:
    pass

# Insert Segment into DB
command = f"""INSERT INTO Segment VALUES (?, ?, ?, ?, ?, ?, ?)"""
self.cursor.execute(command,
                    _parameters: (None,
                                  header_dict["isolate_id"], header_dict["segment_type"],
                                  header_dict["segment_epi"], header_dict["virus_name"], header_dict["segment_name"]),
                    _commit: True)

# Return id for SegmentMutations
self.conn.commit()
return self.cursor.lastrowid
```

# MUTATIONS

# NOTE: VIEW CREATION FOR DATA TRANSFER

```
1  SELECT
2    row_number() OVER (ORDER BY pwi.n)
3    paper_effect_marker_id,
4    me.marker_id, pwi.n as paper_id,
5    ewi.n as effect_id,
6    me.subtype, me.in_vivo, me.in_vitro
7  FROM markers_effects me JOIN
8    papers_with_ids pwi ON me.paper_id = pwi.id JOIN
9    effects_with_ids ewi ON me.effect_name = ewi.effect_full
```

	paper_effect_marker_id	marker_id	paper_id	effect_id	subtype
1		1	149	4	24 H5N1
2		2	149	4	80 H5N1
3		3	149	4	82 H5N1
4		4	134	12	65 H5N1
5		5	34	13	34 H5N1
6		6	34	13	66 H5N1
7		7	34	13	74 H5N1
8		8	107	14	14 H5N1
9		9	107	14	18 H5N1
10		10	107	14	22 H5N1
11		11	131	14	62 H5N1
12		12	131	14	77 H5N1
13		13	71	15	62 H9N2
14		14	71	15	77 H9N2
15		15	75	16	73 H1N2

In our database migration strategy, I've prioritized data consistency between IZSVe's database and our new system. To achieve this, I've developed the following views:

- 1. PEM View:** This view combines paper, effect, and marker data, ensuring seamless integration.
- 2. Markers Summary View:** Summarizing markers and their associated mutations, this view provides a comprehensive overview.
- 3. Papers with IDs View:** By assigning unique numerical identifiers to papers and filtering out those without a DOI, this view establishes consistency in paper indexing.
- 4. Effects with IDs View:** Assigning unique identifiers to effects streamlines data organization.

These views collectively enhance data integrity, simplifying querying and analysis while bolstering overall database management efficiency.

# TABLE OF CONTENT

**01 - SUMMARY**

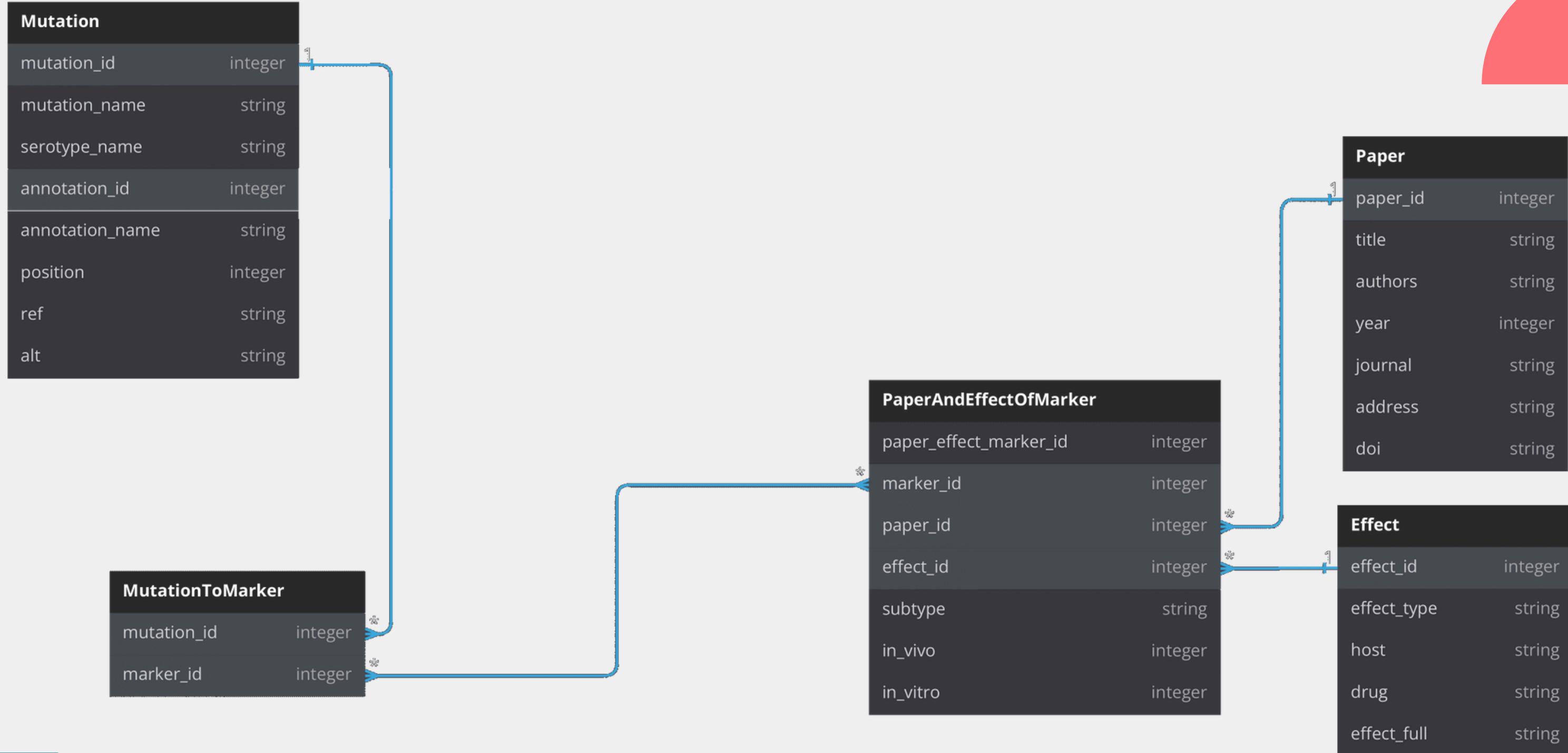
**04 - QUERIES**

**02 - DATABASE**

**05 - NEXT STEPS**

**03 - DATA & SCRIPTS**

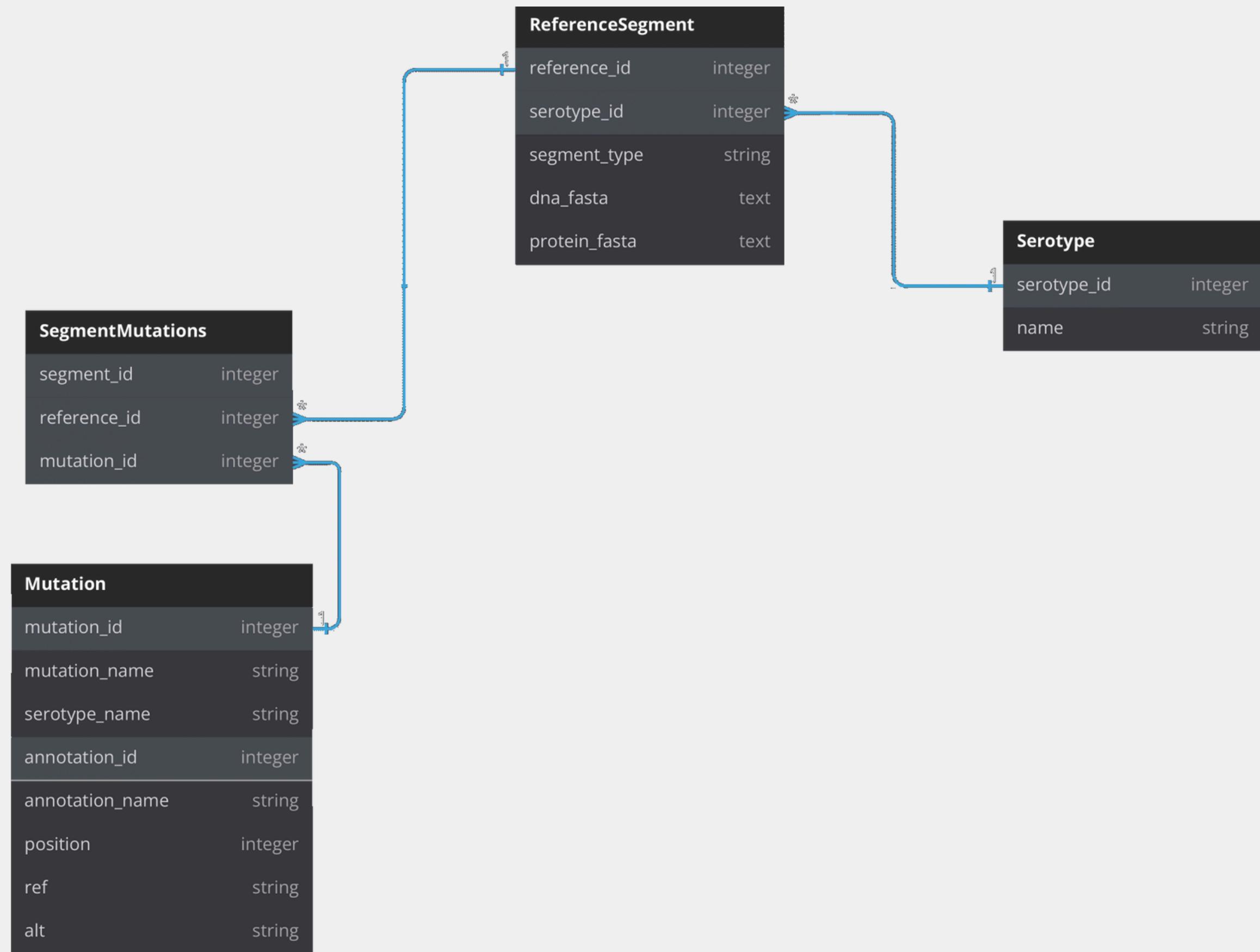
# QUERY 1: GET ALL EFFECTS ASSOCIATED WITH A PARTICULAR MUTATION



## **QUERY 1: GET ALL EFFECTS ASSOCIATED WITH A PARTICULAR MUTATION**

```
SELECT * FROM Effect e
WHERE e.effect_id in (
    SELECT pem.effect_id
    FROM MutationsToMarker mtm
    JOIN Marker m ON mtm.marker_id = m.marker_id
    JOIN PaperAndEffectOfMarker pem ON m.marker_id = pem.marker_id
    WHERE mtm.mutation_id = 6501)
```

## QUERY 2: GET ALL MUTATIONS ASSOCIATED WITH ONE SEROTYPE



## **QUERY 2: GET ALL MUTATIONS ASSOCIATED WITH ONE SEROTYPE**

```
SELECT DISTINCT mut.mutation_name, sm.mutation_id
FROM SegmentMutations sm
JOIN ReferenceSegment rs on sm.reference_id = rs.reference_id
JOIN Serotype sero ON sero.serotype_id = rs.serotype_id
JOIN Mutation mut ON mut.mutation_id = sm.mutation_id
WHERE sero.name = "H5N1"
```

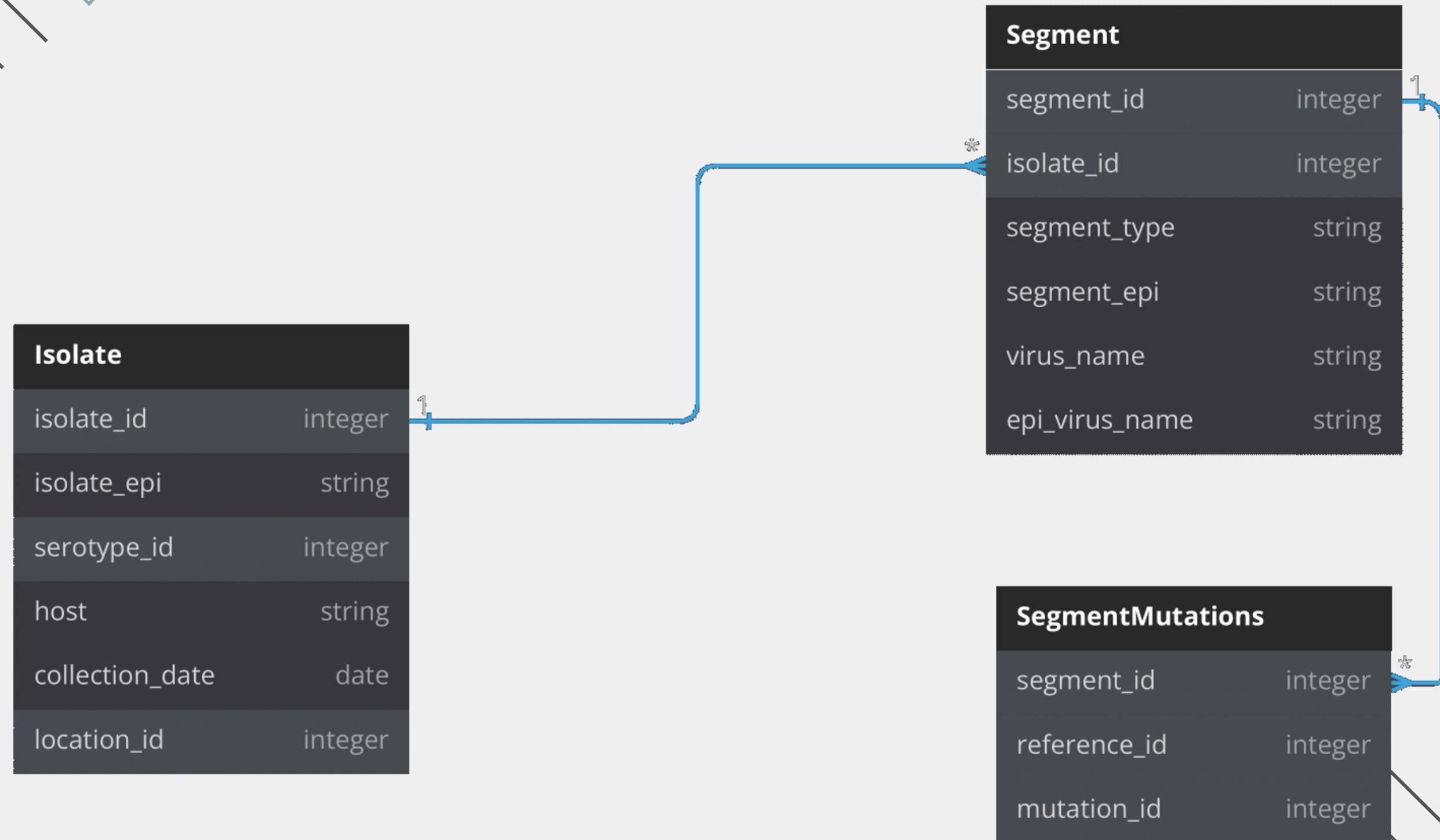
```

1 | SELECT DISTINCT mut.mutation_name, sm.mutation_id, mut.mutation_name
2 | FROM SegmentMutations sm
3 | JOIN ReferenceSegment rs ON sm.reference_id = rs.reference_id
4 | JOIN Serotype sero ON sero.serotype_id = rs.serotype_id
5 | JOIN Mutation mut ON mut.mutation_id = sm.mutation_id
6 | WHERE sero.name = "H5N1"

```

	mutation_name	mutation_id	mutation_name
1	H5N1:HA1:K3R	1	H5N1:HA1:K3R
2	H5N1:HA1:K5R	2	H5N1:HA1:K5R
3	H5N1:HA1:E7N	3	H5N1:HA1:E7N
4	H5N1:HA1:R8K	4	H5N1:HA1:R8K
5	H5N1:HA1:N8S	5	H5N1:HA1:N8S
6	H5N1:HA1:G8D	6	H5N1:HA1:G8D
7	H5N1:HA1:S9N	7	H5N1:HA1:S9N
8	H5N1:HA1:L9F	8	H5N1:HA1:L9F
9	H5N1:HA1:L1Q	9	H5N1:HA1:L1Q
10	H5N1:HA1:P1S	10	H5N1:HA1:P1S
11	H5N1:HA1:E1D	11	H5N1:HA1:E1D

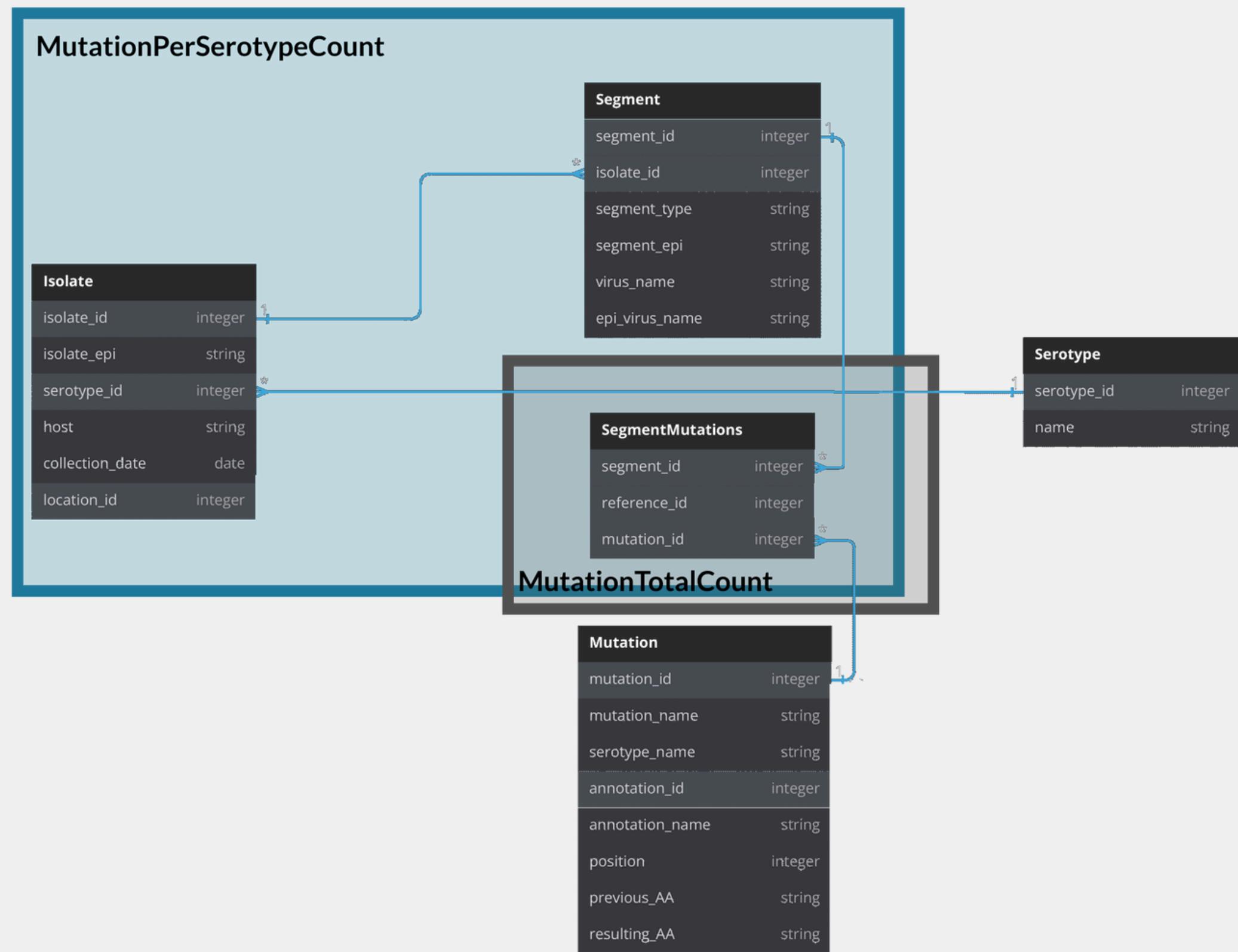
# QUERY 3: GET ALL MUTATIONS ASSOCIATED WITH ONE SEROTYPE AND NOT OTHERS



## **QUERY 3: GET ALL MUTATIONS ASSOCIATED WITH ONE SEROTYPE AND NOT OTHERS**

```
SELECT sm.mutation_id, iso.serotype_id
FROM SegmentMutations sm
JOIN Segment s ON sm.segment_id = s.segment_id
JOIN Isolate iso ON s.isolate_id = iso.isolate_epi
WHERE sm.mutation_id NOT IN
    (SELECT sm.mutation_id FROM SegmentMutations sm2
     JOIN Segment s2 ON sm2.segment_id = s2.segment_id
     JOIN Isolate iso2 ON s2.isolate_id = iso2.isolate_epi
     WHERE iso2.serotype_id <> iso.serotype_id)
```

# QUERY 4: GET ALL MUTATIONS ASSOCIATED 95% OF THE TIMES WITH ONLY ONE SEROTYPE



## QUERY 4: GET ALL MUTATIONS ASSOCIATED 95% OF THE TIMES WITH ONLY ONE SEROTYPE

```
WITH MutationPerSerotypeCount AS (
    SELECT sm.mutation_id, iso.serotype_id, COUNT(*) AS mutation_count
    FROM SegmentMutations sm
    JOIN Segment s ON sm.segment_id = s.segment_id
    JOIN Isolate iso ON s.isolate_id = iso.isolate_epi
    GROUP BY sm.mutation_id, iso.serotype_id),

MutationTotalCount AS (
    SELECT sm.mutation_id as mutation_id, COUNT(*) AS total_mutation_count
    FROM SegmentMutations sm
    GROUP BY sm.mutation_id)

SELECT m.mutation_name, mpc.mutation_id, sero.name AS serotype_name, mpc.mutation_count,
    mpc.mutation_count * 100.0 / mtc.total_mutation_count AS mutation_percentage
    FROM MutationPerSerotypeCount mpc
    JOIN MutationTotalCount mtc ON mpc.mutation_id = mtc.mutation_id
    JOIN Mutation m ON m.mutation_id = mpc.mutation_id
    JOIN Serotype sero ON sero.serotype_id = mpc.serotype_id
    WHERE mpc.mutation_count * 100.0 / mtc.total_mutation_count >= 95
```

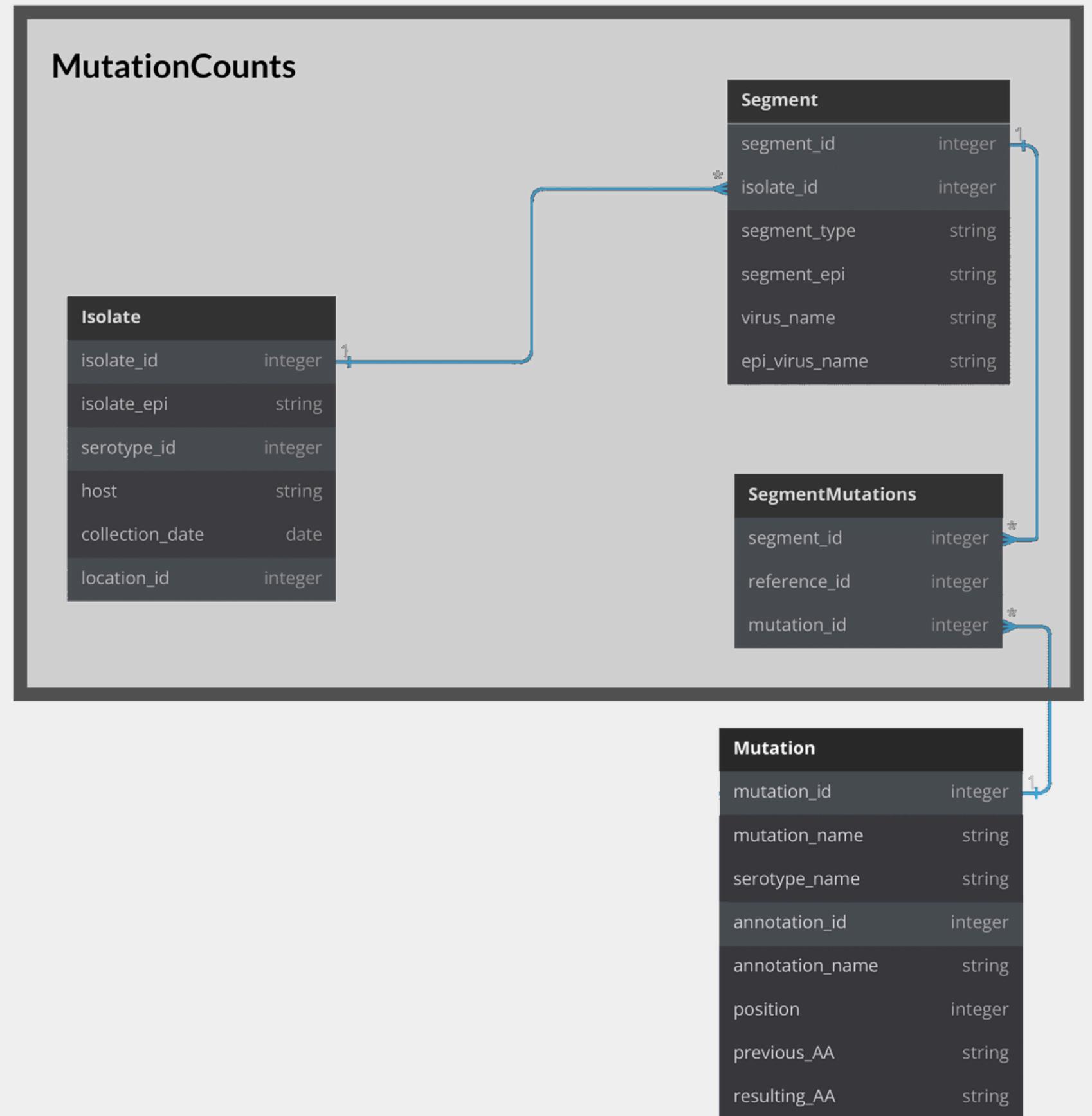
```

1 WITH MutationPerSerotypeCount AS (
2     SELECT sm.mutation_id, iso.serotype_id, COUNT(*) AS mutation_count
3     FROM SegmentMutations sm
4     JOIN Segment s ON sm.segment_id = s.segment_id
5     JOIN Isolate iso ON s.isolate_id = iso.isolate_epi
6     GROUP BY sm.mutation_id, iso.serotype_id),
7 MutationTotalCount AS (
8     SELECT sm.mutation_id as mutation_id, COUNT(*) AS total_mutation_count
9     FROM SegmentMutations sm
10    GROUP BY sm.mutation_id)
11    SELECT m.mutation_name, mpc.mutation_id, sero.name AS serotype_name, mpc.mutation_count,
12          mpc.mutation_count * 100.0 / mtc.total_mutation_count AS mutation_percentage
13    FROM MutationPerSerotypeCount mpc
14    JOIN MutationTotalCount mtc ON mpc.mutation_id = mtc.mutation_id
15    JOIN Mutation m ON m.mutation_id = mpc.mutation_id
16    JOIN Serotype sero ON sero.serotype_id = mpc.serotype_id
17    WHERE mpc.mutation_count * 100.0 / mtc.total_mutation_count >= 95

```

	mutation_name	mutation_id	serotype_name	mutation_count	mutation_percentage	
1	H5N1:HA1:K3R	1	H5N1	23	100.0	
2	H5N1:HA1:K5R	2	H5N1	5	100.0	
3	H5N1:HA1:E7N	3	H5N1	23	100.0	
4	H5N1:HA1:R8K	4	H5N1	23	100.0	
5	H5N1:HA1:N8S	5	H5N1	3	100.0	

# QUERY 5A: GET ALL MUTATIONS BY % OF HUMAN HOSTS



## QUERY 5A: GET ALL MUTATIONS BY % HUMAN HOSTS

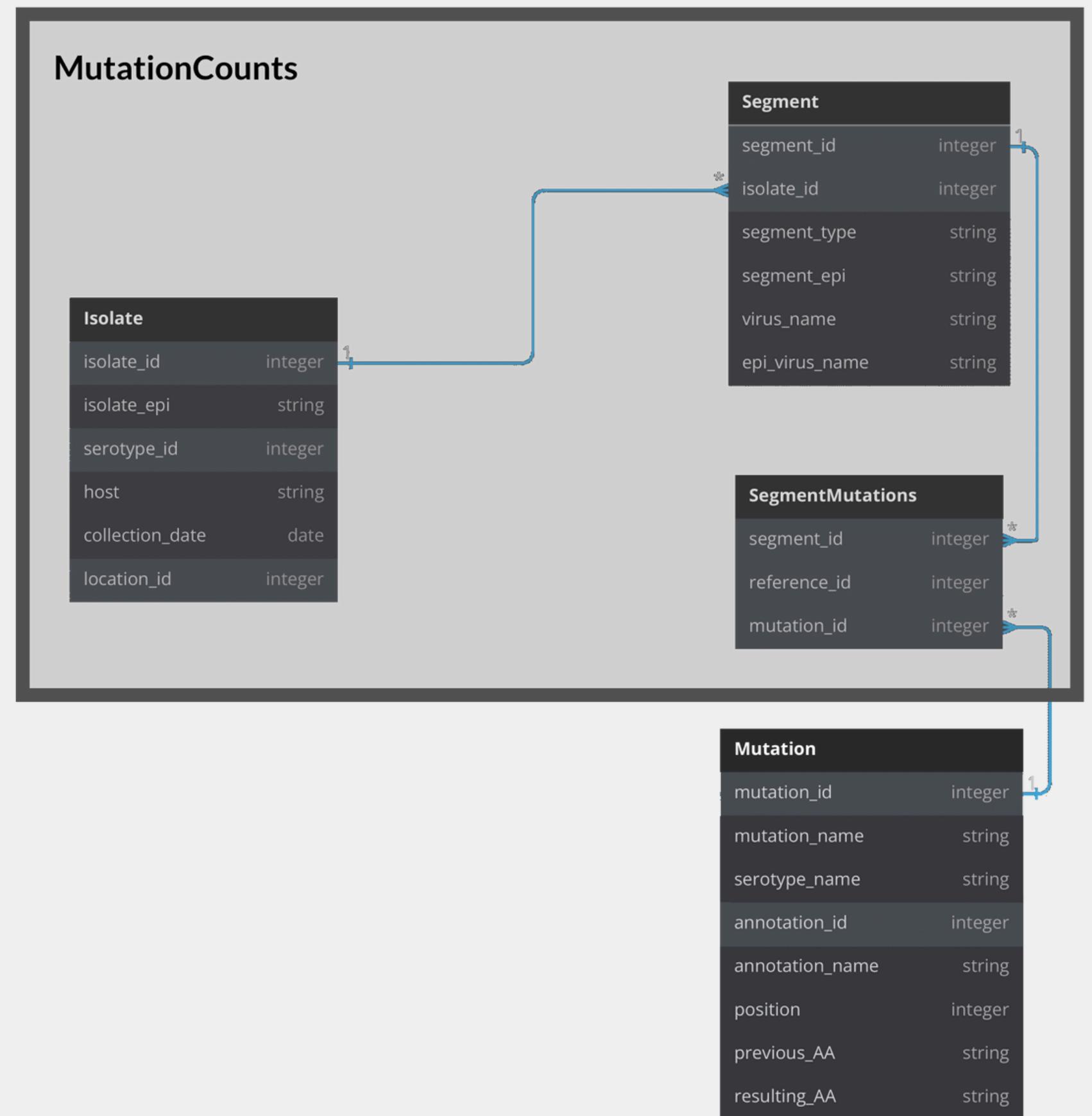
```
CREATE VIEW MutationCounts AS
SELECT sm.mutation_id, COUNT(*) AS total_count,
SUM(CASE WHEN iso.host = 'Human' THEN 1 ELSE 0 END) AS human_count
FROM SegmentMutations sm JOIN Segment seg ON sm.segment_id =
seg.segment_id
JOIN Isolate iso ON seg.isolate_id = iso.isolate_epi
GROUP BY sm.mutation_id

SELECT m.mutation_name, mutation_id, total_count, human_count,
human_count * 100.0 / total_count AS human_percentage
FROM MutationCounts mc
JOIN Mutation m on m.mutation_id = mc.mutation_id
ORDER BY human_count * 100.0 / total_count DESC
```

```
1 | SELECT m.mutation_name, mc.mutation_id, total_count, human_count, human_count * 100.0 / total_count AS human_percentage
2 | FROM MutationCounts mc
3 | JOIN Mutation m ON m.mutation_id = mc.mutation_id
4 | ORDER BY human_count * 100.0 / total_count DESC
```

	mutation_name	mutation_id	total_count	human_count	human_percentage
1	H5N1:HA1:K5R	2	5	5	100.0
2	H5N1:HA1:N8S	5	3	3	100.0
3	H5N1:HA1:E1D	11	3	3	100.0
4	H5N1:HA1:L1S	13	3	3	100.0
5	H5N1:HA1:A1S	14	5	5	100.0
6	H5N1:HA1:Q1L	15	3	3	100.0
7	H5N1:HA1:A1R	16	2	2	100.0
8	H5N1:HA1:S2P	27	3	3	100.0
9	H5N1:HA1:V2M	35	5	5	100.0
10	H5N1:HA2:D5N	38	5	5	100.0
11	H5N1:HA1:N1S	44	5	5	100.0
12	H5N1:HA1:A1K	45	5	5	100.0
13	H5N1:HA1:S3A	48	1	1	100.0
14	H5N1:HA2:K3Q	49	3	3	100.0
15	H5N1:HA1:A8V	52	2	2	100.0
16	H5N1:HA1:S9D	53	3	3	100.0
17	H5N1:HA1:P1L	54	2	2	100.0

## QUERY 5B: GET ALL MUTATIONS WHERE HUMAN HOSTS % >= 95%



## **QUERY 5B: GET ALL MUTATIONS WHERE HUMAN HOSTS % >= 95%**

```
CREATE VIEW MutationCounts AS
SELECT sm.mutation_id, COUNT(*) AS total_count,
SUM(CASE WHEN iso.host = 'Human' THEN 1 ELSE 0 END) AS human_count
FROM SegmentMutations sm JOIN Segment seg ON sm.segment_id =
seg.segment_id
JOIN Isolate iso ON seg.isolate_id = iso.isolate_epi
GROUP BY sm.mutation_id
```

```
SELECT m.mutation_name, mc.mutation_id, total_count, human_count,
human_count * 100.0 / total_count AS human_percentage
FROM MutationCounts mc
JOIN Mutation m on m.mutation_id = mc.mutation_id
WHERE human_count * 100.0 / total_count >= 95.0
```

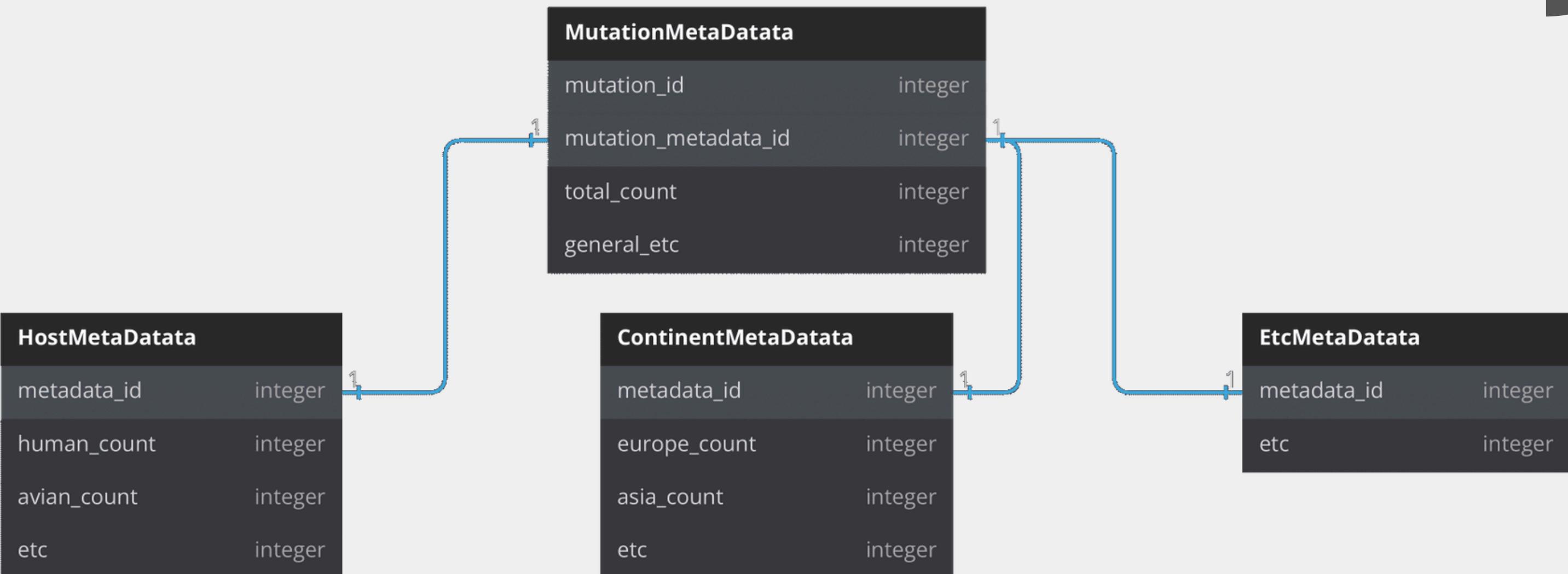
```

1 | SELECT m.mutation_name, mc.mutation_id, total_count, human_count, human_count * 100.0 / total_count AS human_percentage
2 | FROM MutationCounts mc
3 | JOIN Mutation m ON m.mutation_id = mc.mutation_id
4 | WHERE human_count * 100.0 / total_count >= 95.0

```

	mutation_name	mutation_id	total_count	human_count	human_percentage
1	H5N1:HA1:K5R	2	5	5	100.0
2	H5N1:HA1:N8S	5	3	3	100.0
3	H5N1:HA1:E1D	11	3	3	100.0
4	H5N1:HA1:L1S	13	3	3	100.0
5	H5N1:HA1:A1S	14	5	5	100.0
6	H5N1:HA1:Q1L	15	3	3	100.0
7	H5N1:HA1:A1R	16	2	2	100.0
8	H5N1:HA1:S2P	27	3	3	100.0
9	H5N1:HA1:V2M	35	5	5	100.0
10	H5N1:HA2:D5N	38	5	5	100.0
11	H5N1:HA1:N1S	44	5	5	100.0
12	H5N1:HA1:A1K	45	5	5	100.0
13	H5N1:HA1:S3A	48	1	1	100.0
14	H5N1:HA2:K3Q	49	3	3	100.0
15	H5N1:HA1:A8V	52	2	2	100.0
16	H5N1:HA1:S9D	53	3	3	100.0
17	H5N1:HA1:P1L	54	2	2	100.0

# QUERY 5C: GET ALL MUTATIONS BY % HUMAN HOSTS WHERE >= 95%



## **QUERY 5C: GET ALL MUTATIONS BY % HUMAN HOSTS WHERE >= 95%**

```
SELECT mutation_id,  
hmd.human_count / md.total_count AS human_percentage  
FROM MutationMetaData md  
JOIN HostMetaData hmd ON md.id = hmd.metadata_id  
WHERE md.total_count > 0  
AND hmd.human_count / md.total_count >= 0.95  
ORDER BY hmd.human_count * 100.0 / md.total_count  
DESC
```

## QUERY 6, 7, 8

Mutation	
mutation_id	integer
mutation_name	string
serotype_name	string
annotation_id	integer
annotation_name	string
position	integer
previous_AA	string
resulting_AA	string

## QUERY 6: GET THE MOST COMMON AA MUTATIONS

```
CREATE VIEW MostCommonAAMutation AS  
SELECT previous_AA, resulting_AA, COUNT(*) AS count  
FROM Mutation  
GROUP BY previous_AA, resulting_AA
```

```
SELECT * FROM MostCommonAAMutation  
ORDER BY count DESC
```

	previous_AA	resulting_AA	count
1	K	R	23
2	R	K	16
3	V	I	15
4	I	V	14
5	T	A	13
6	D	N	12
7	S	N	12
8	N	S	11
9	A	T	10
10	A	V	9
11	E	G	9
12	I	L	9
	F	Q	8

## **QUERY 7: GET THE MOST COMMON AA MUTATIONS PER ANNOTATION**

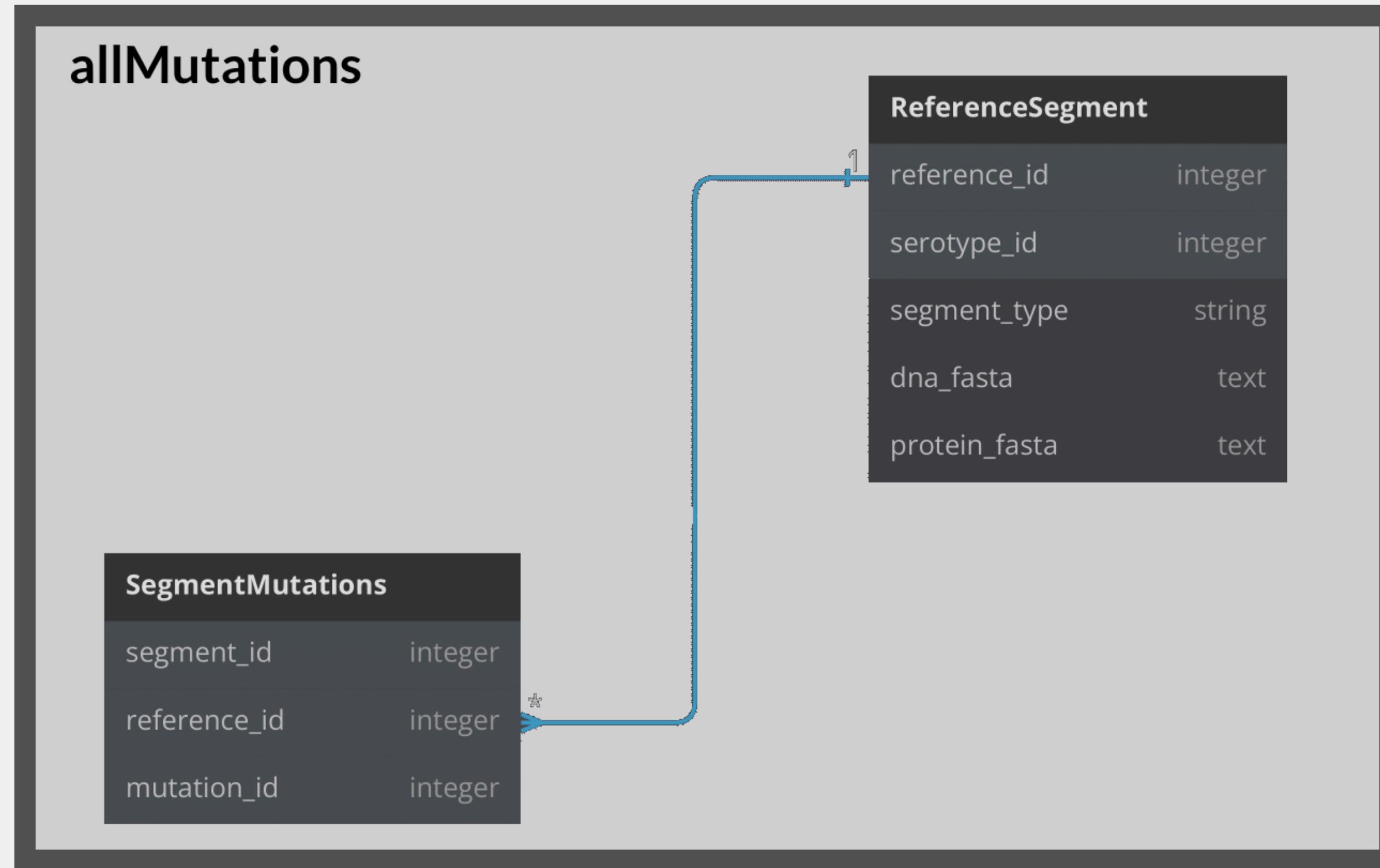
```
CREATE VIEW MostCommonAnnotationMutation AS  
SELECT previous_AA, resulting_AA, COUNT(*) AS count  
FROM Mutation  
GROUP BY annotation_name, previous_AA, resulting_AA  
  
SELECT * FROM MostCommonAnnotationMutation  
ORDER BY count DESC
```

## **QUERY 8: GET THE MOST COMMON AA MUTATIONS COMBINATION OF ANNOTATION AND SEROTYPE**

```
CREATE VIEW MostCommonSerotypeAnnotationMutation AS
SELECT previous_AA, resulting_AA, COUNT(*) AS count
FROM Mutation
GROUP BY serotype_id, annotation_name, previous_AA, resulting_AA
```

```
SELECT * FROM MostCommonSerotypeAnnotationMutation
ORDER BY count DESC
```

# QUERY 9: FOR EACH MUTATION GET THE NUMBER OF SEGMENTS THAT EXHIBIT IT, AND THE PERCENTAGE OVER THE TOTAL NUMBER OF SEGMENTS



## **QUERY 9: FOR EACH MUTATION GET THE NUMBER OF SEGMENTS THAT EXHIBIT IT, AND THE PERCENTAGE OVER THE TOTAL NUMBER OF SEGMENTS**

```
SELECT m.mutation_name, sm.mutation_id, sm.reference_id, COUNT(*) as count,
       COUNT(*) / (SELECT COUNT(DISTINCT sm.segment_id) FROM SegmentMutations WHERE
reference_id = sm.reference_id) as percentage
FROM SegmentMutations sm
JOIN Mutation m ON m.mutation_id = sm.mutation_id
GROUP BY sm.mutation_id, sm.reference_id
ORDER BY count DESC
```

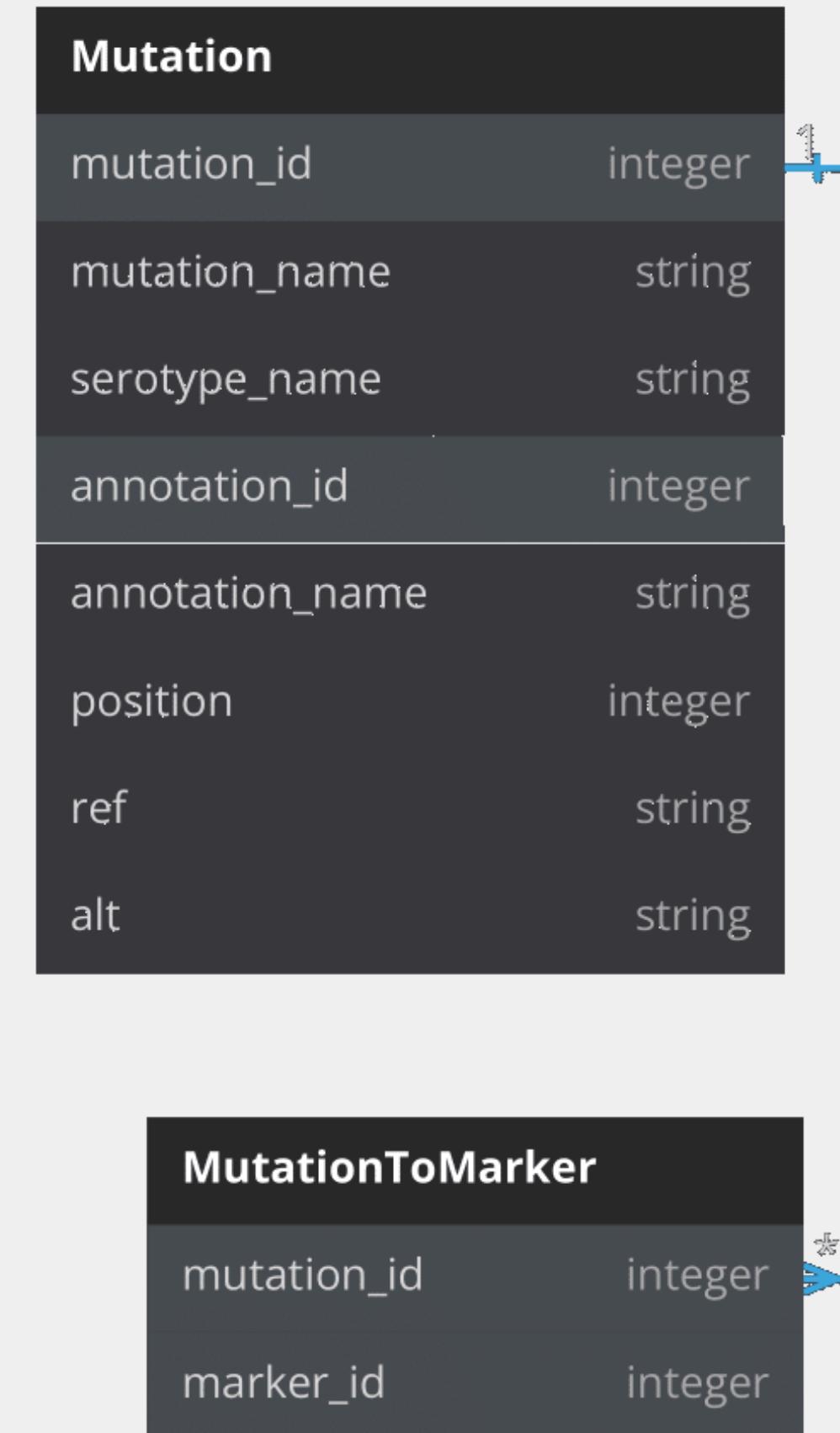
```

1 | SELECT m.mutation_name, sm.mutation_id, sm.reference_id, COUNT(*) as count,
2 |     COUNT(*) * 1.0 / (SELECT COUNT(*) FROM SegmentMutations WHERE reference_id = sm.reference_id) as percentage
3 | FROM SegmentMutations sm
4 | JOIN Mutation m ON m.mutation_id = sm.mutation_id
5 | GROUP BY sm.mutation_id, sm.reference_id
6 | ORDER BY count DESC

```

	mutation_name	mutation_id	reference_id	count	percentage
1	H5N1:HA1:K3R	1	3	23	0.0153027278775782
2	H5N1:HA1:E7N	3	3	23	0.0153027278775782
3	H5N1:HA1:R8K	4	3	23	0.0153027278775782
4	H5N1:HA1:L9F	8	3	23	0.0153027278775782
5	H5N1:HA1:P1S	10	3	23	0.0153027278775782
6	H5N1:HA1:D1S	17	3	23	0.0153027278775782
7	H5N1:HA1:S1P	21	3	23	0.0153027278775782
8	H5N1:HA1:N1D	22	3	23	0.0153027278775782
9	H5N1:HA1:E1A	23	3	23	0.0153027278775782
10	H5N1:HA1:N1K	24	3	23	0.0153027278775782
11	H5N1:HA1:K1Q	25	3	23	0.0153027278775782
12	H5N1:HA1:K2E	26	3	23	0.0153027278775782
13	H5N1:HA1:D2N	31	3	23	0.0153027278775782
14	H5N1:HA1:H2N	32	3	23	0.0153027278775782
15	H5N1:HA1:G2E	33	3	23	0.0153027278775782
16	H5N1:HA1:V2L	34	3	23	0.0153027278775782
17	H5N1:HA1:S3T	36	3	23	0.0153027278775782

# QUERY 10: RETRIEVE ALL MARKERS ASSOCIATED WITH A PARTICULAR MUTATION



## **QUERY 10: RETRIEVE ALL MARKERS ASSOCIATED TO GROUPS THAT CONTAIN A PARTICULAR MUTATION**

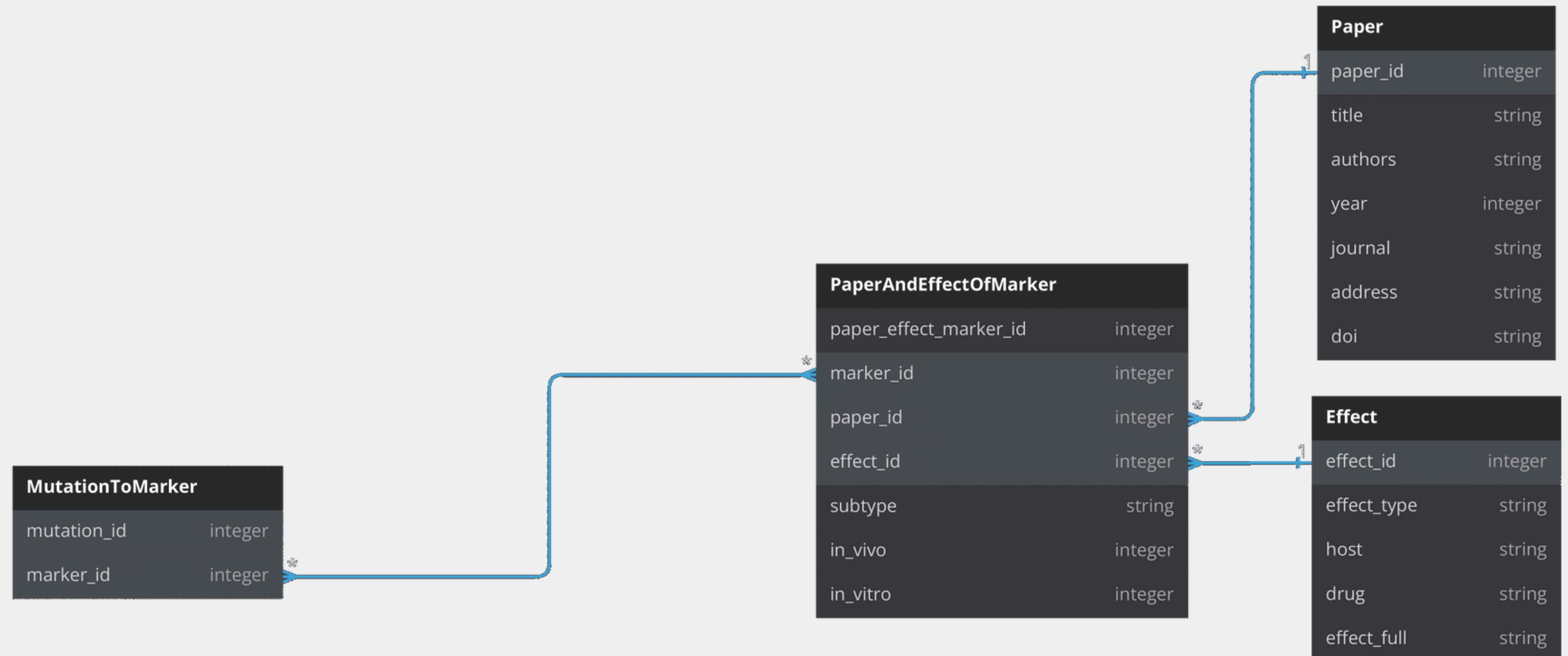
```
SELECT marker_id FROM MutationsToMarker mtm  
JOIN Mutation mut ON mut.mutation_id = mtm.mutation_id  
WHERE mut.mutation_name = "H5N1:M2:L26F"
```

## **QUERY 10B: RETRIEVE THE MARKER (IF ANY) ASSOCIATED TO A GROUP OF MUTATIONS**

```
WITH mutation_group AS
  (SELECT mutation_id
   FROM Mutation
   WHERE mutation_name IN (???)),
   


SELECT marker_id FROM MutationsToMarker
WHERE mutation_id IN (SELECT mutation_id FROM mutation_group)
GROUP BY marker_id
HAVING COUNT(CASE WHEN mutation_id IN (SELECT mutation_id FROM mutation_group ) THEN 1
END) = COUNT(*)
AND COUNT(CASE WHEN mutation_id IN (SELECT mutation_id FROM mutation_group ) THEN 1 END)
= (SELECT COUNT(*) FROM mutation_group)
```

# QUERY 11: GET ALL EFFECTS ASSOCIATED WITH A SPECIFIC MUTATION



## **QUERY 11: GET ALL EFFECTS ASSOCIATED WITH A SPECIFIC MUTATION**

```
SELECT * FROM Effect e
WHERE e.effect_id in (
    SELECT pem.effect_id
    FROM MutationsToMarker mtm
    JOIN PaperAndEffectOfMarker pem ON mtm.marker_id = pem.marker_id
    WHERE mtm.mutation_id = 6501)
```

## QUERY 12: IDENTIFY MUTATIONS AT A SPECIFIC POSITION.

Mutation	
mutation_id	integer
mutation_name	string
serotype_name	string
annotation_id	integer
annotation_name	string
position	integer
previous_AA	string
resulting_AA	string

## **QUERY 12: IDENTIFY MUTATIONS AT A SPECIFIC POSITION.**

```
SELECT mutation_name  
FROM Mutation  
WHERE position > 0  
AND position < 1000
```

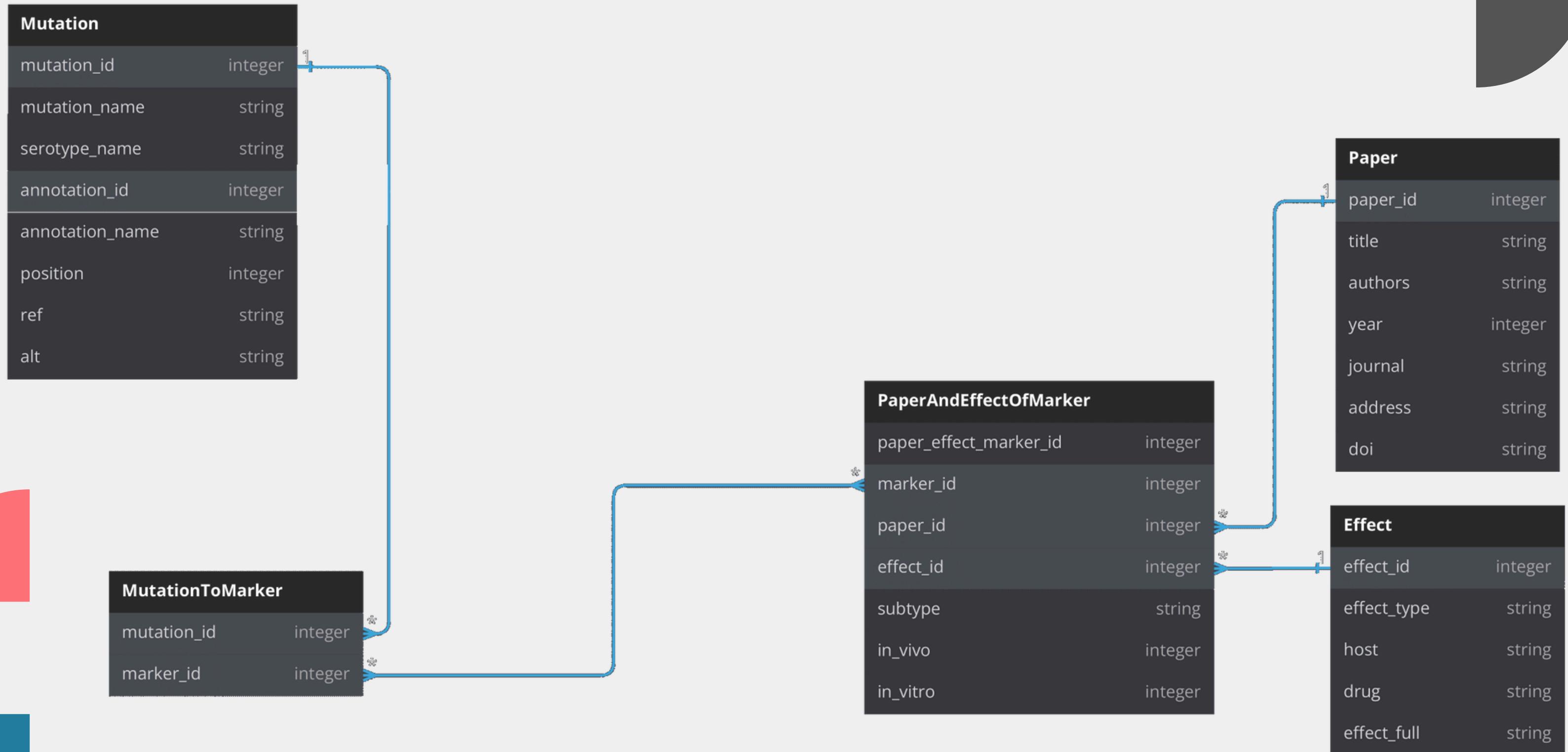
## QUERY 13: RETRIEVE ALL EFFECTS BY SPECIFIC TYPE

Effect	
effect_id	integer
effect_type	string
host	string
drug	string
effect_full	string

## QUERY 13: RETRIEVE ALL EFFECTS BY SPECIFIC TYPE

```
SELECT effect_full  
from Effect  
WHERE effect_type LIKE '%Increased%'  
OR effect_type LIKE '%Decreased%'
```

# QUERY 14: IDENTIFY MUTATIONS ASSOCIATED WITH A PARTICULAR EFFECT TYPE



## **QUERY 14: IDENTIFY MUTATIONS ASSOCIATED WITH A PARTICULAR EFFECT TYPE**

```
SELECT DISTINCT mut.mutation_name
FROM Mutation mut
JOIN MutationsToMarker mtm ON mut.mutation_id = mtm.mutation_id
JOIN PaperAndEffectOfMarker pem ON mtm.marker_id = pem.marker_id
JOIN Effect e ON pem.effect_id = e.effect_id
WHERE e.effect_type LIKE '%Increased%'
```

# TABLE OF CONTENT

**01 - SUMMARY**

**04 - QUERIES**

**02 - DATABASE**

**05 - NEXT STEPS**

**03 - DATA & SCRIPTS**

# NEXT STEPS

## ARCHITECTURE

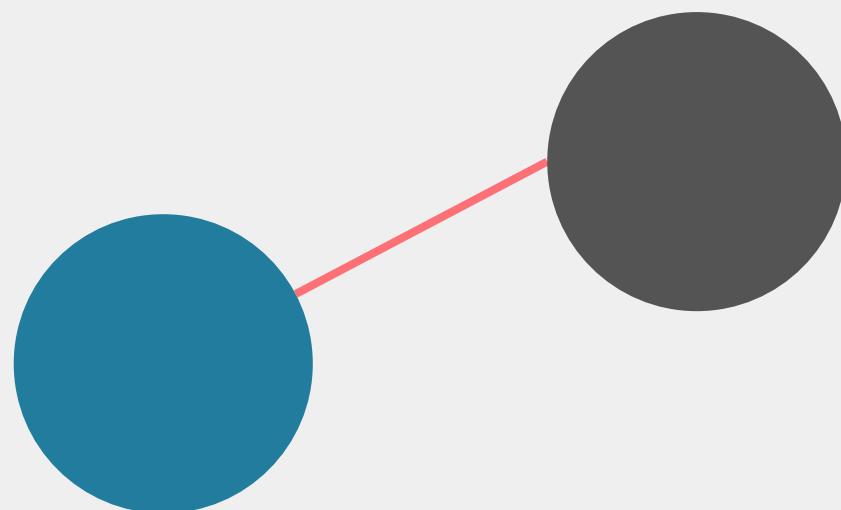
DESIGN AND CONCEPTUALIZE THE ARCHITECTURE OF THE ANALYTICAL FRAMEWORK, ENSURING SCALABILITY, EFFICIENCY

## DOCKER

IMPLEMENT DOCKER CONTAINERS TO DEPLOY THE PROJECT, ENSURING CONSISTENCY ACROSS DIFFERENT ENVIRONMENTS

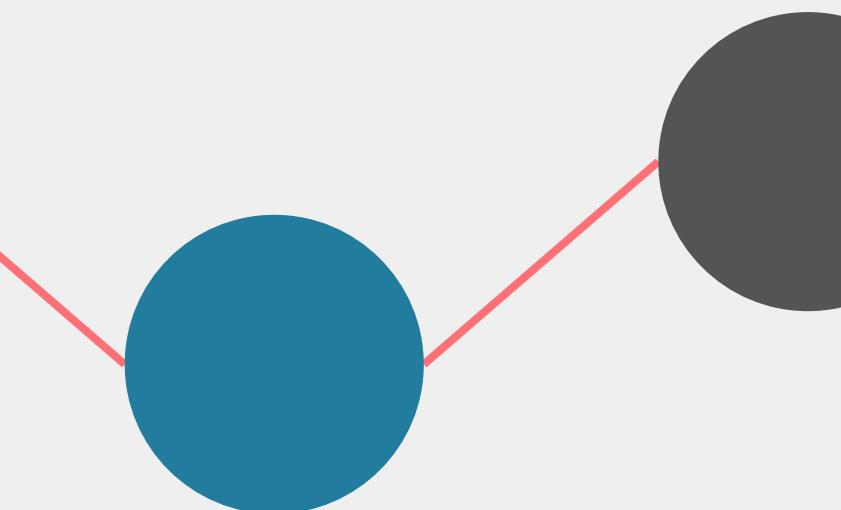
## UPDATES

CONTINUOUSLY REFINE AND UPDATE THE WEBSITE, INCORPORATING NEW DATA AND INSIGHTS OVER TIME



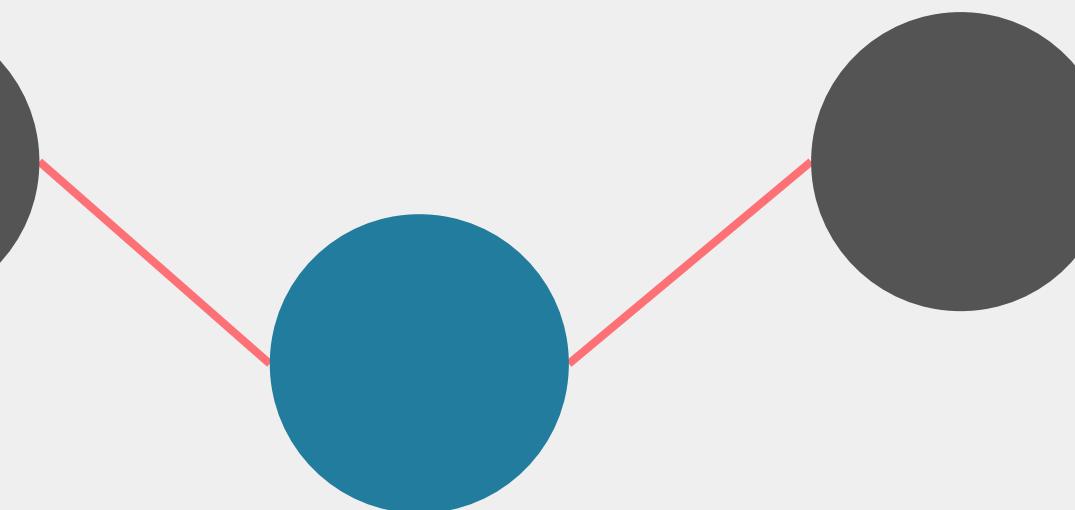
## NEW QUERIES

EXPAND THE SCOPE OF DATA COLLECTION BY FORMULATING AND EXECUTING REFINED QUERIES TO GATHER ADDITIONAL PERTINENT DATA ON FLU MUTATIONS



## DEVELOPMENT

DEVELOPMENT PHASE, TRANSLATING ARCHITECTURAL BLUEPRINTS INTO FUNCTIONAL CODE, ITERATIVELY REFINING ALGORITHMS AND FUNCTIONALITIES



## PUBLIC WEBSITE

PUBLIC-FACING WEBSITE TO PUBLISH RESEARCH FINDINGS, PROVIDING A USER-FRIENDLY INTERFACE FOR ACCESSING VISUALIZATIONS



# THANK YOU

ANY QUESTION?