

Database Structure Documentation

Database structure documentation for the Physician Personality Trait Annotation System.

Database Overview

This system uses PostgreSQL as the main database, containing the following major tables:

- **physicians** - Physician information table
 - **reviews** - Patient reviews table
 - **tasks** - Annotation tasks table
 - **model_annotations** - Model annotation results table
 - **human_annotations** - Human annotation results table
 - **machine_annotation_evaluations** - Human evaluations of machine annotations table
-

Table Structure Details

1. **physicians** — Physician Information Table

Stores basic physician information and biographies.

Field Name	Type	Description	Example
id	SERIAL	Primary key	1
phy_id	BIGINT	Physician original data unique ID	100745676
npi	BIGINT	National Provider Identifier	1659371102
first_name	TEXT	Physician first name	ALINA
last_name	TEXT	Physician last name	GRIGORE
gender	TEXT	Gender	F
credential	TEXT	Physician credential (e.g., MD)	MD
specialty	TEXT	Physician specialty	Anesthesiology Physician
practice_zip5	TEXT	Practice ZIP code	89134
business_zip5	TEXT	Business ZIP code	25304
biography_doc	TEXT	Biography (supports HTML format)	<p>Dr. Grigore is...</p>

Field Name	Type	Description	Example
education_doc	TEXT	Education history (XML format)	<education>Duke University...</education>
num_reviews	INTEGER	Number of reviews	7
doc_name	TEXT	Physician display name	Dr. Alina Grigore
zip3	TEXT	First 3 digits of ZIP code	891
zip2	TEXT	First 2 digits of ZIP code	89
zipcode	TEXT	Complete ZIP code	89134
state	TEXT	State	NV
region	TEXT	Regional distribution	Mountain

Special Notes:

- **biography_doc**: Supports HTML tags, frontend will render accordingly
- **education_doc**: Uses <education> XML tag format, frontend will parse automatically

2. reviews — Patient Reviews Table

Stores patient reviews for physicians.

Field Name	Type	Description	Example
id	SERIAL	Primary key	223
physician_id	INTEGER	Foreign key, references physicians	10
review_index	INTEGER	Review number (#0, #1 ...)	0
source	TEXT	Source (Vitals, HG, etc.)	Vitals
date	TIMESTAMP	Timestamp	2025-06-05 23:30:18
text	TEXT	Review content	I had an excellent experience...

Index Suggestions:

```
CREATE INDEX idx_reviews_physician_id ON reviews(physician_id);
CREATE INDEX idx_reviews_date ON reviews(date);
```

3. tasks — Annotation Tasks Table

Manages annotation task assignments and status.

Field Name	Type	Description	Example
id	SERIAL	Primary key	1
physician_id	INTEGER	Foreign key, references physicians	10
assigned_to	TEXT	Assigned annotator	user001
status	TEXT	Task status	in_progress
created_at	TIMESTAMP	Creation time	2025-06-05 23:30:18
updated_at	TIMESTAMP	Update time	2025-06-05 23:35:20

Status Enum:

- **pending** - Pending
- **in_progress** - In Progress
- **completed** - Completed
- **cancelled** - Cancelled

4. **model_annotations** — Model Annotation Results Table

Stores AI model analysis results for physician personality traits.

Field Name	Type	Description	Example
id	SERIAL	Primary key	1
physician_id	INTEGER	Foreign key, references physicians	10
model_name	TEXT	Model name	GPT-4
trait	TEXT	Personality dimension	Openness
score	TEXT	Score result	High
consistency	TEXT	Model consistency description	Very Consistent
sufficiency	TEXT	Model evidence sufficiency desc.	Sufficient
evidence	TEXT	Original evidence text from model	Based on the reviews...

Personality Trait Enum:

- **Openness** - Openness
- **Conscientiousness** - Conscientiousness
- **Extraversion** - Extraversion
- **Agreeableness** - Agreeableness
- **Neuroticism** - Neuroticism

Score Enum:

- **Low** - Low
- **Moderate** - Moderate
- **High** - High

5. **human_annotations** — Human Annotation Results Table

Stores human annotators' annotation results.

Field Name	Type	Description	Example
id	SERIAL	Primary key	1
physician_id	INTEGER	Foreign key, references physicians	10
evaluator	TEXT	Annotator username	user001
task_id	INTEGER	Foreign key, references tasks	1
trait	TEXT	Personality dimension	Openness
score	TEXT	Score result	High
consistency	TEXT	Consistency evaluation	Very Consistent
sufficiency	TEXT	Evidence sufficiency evaluation	Sufficient
evidence	TEXT	Evidence text provided by annotator	The patient reviews show...
timestamp	TIMESTAMP	Annotation time	2025-06-05 23:30:18

6. **machine_annotation_evaluations** — Machine Annotation Evaluation Table

Stores human annotators' evaluations of AI model outputs.

Field Name	Type	Description	Example
id	SERIAL	Primary key	1
model_annotation_id	INTEGER	Foreign key, references model_annotations	1
evaluator	TEXT	Evaluator username	user001
task_id	INTEGER	Foreign key, references tasks	1
ranking	INTEGER	Model ranking (1 is best)	1
accuracy_score	TEXT	Accuracy evaluation	Good
comment	TEXT	Subjective evaluation text	This model provides accurate...
timestamp	TIMESTAMP	Evaluation time	2025-06-05 23:30:18

Accuracy Score Enum:

- **Excellent** - Excellent
 - **Good** - Good
 - **Fair** - Fair
 - **Poor** - Poor
-

Relationship Diagram

```
physicians (1) ↔ (N) reviews
      ↓
tasks (1) ↔ (N) human_annotations
      ↓
physicians (1) ↔ (N) model_annotations
      ↓
model_annotations (1) ↔ (N) machine_annotation_evaluations
```

Database Initialization

Create Database

```
CREATE DATABASE physicians;
\c physicians;
```

Create Table Structure

```
-- Physician information table
CREATE TABLE physicians (
  id SERIAL PRIMARY KEY,
  phy_id BIGINT,
  npi BIGINT UNIQUE,
  first_name TEXT,
  last_name TEXT,
  gender TEXT,
  credential TEXT,
  specialty TEXT,
  practice_zip5 TEXT,
  business_zip5 TEXT,
  biography_doc TEXT,
  education_doc TEXT,
  num_reviews INTEGER,
  doc_name TEXT,
  zip3 TEXT,
  zip2 TEXT,
```

```

        zipcode TEXT,
        state TEXT,
        region TEXT
    );

-- Reviews table
CREATE TABLE reviews (
    id SERIAL PRIMARY KEY,
    physician_id INTEGER REFERENCES physicians(id),
    review_index INTEGER,
    source TEXT,
    date TIMESTAMP,
    text TEXT
);

-- Tasks table
CREATE TABLE tasks (
    id SERIAL PRIMARY KEY,
    physician_id INTEGER REFERENCES physicians(id),
    assigned_to TEXT,
    status TEXT DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Model annotations table
CREATE TABLE model_annotations (
    id SERIAL PRIMARY KEY,
    physician_id INTEGER REFERENCES physicians(id),
    model_name TEXT,
    trait TEXT,
    score TEXT,
    consistency TEXT,
    sufficiency TEXT,
    evidence TEXT
);

-- Human annotations table
CREATE TABLE human_annotations (
    id SERIAL PRIMARY KEY,
    physician_id INTEGER REFERENCES physicians(id),
    evaluator TEXT,
    task_id INTEGER REFERENCES tasks(id),
    trait TEXT,
    score TEXT,
    consistency TEXT,
    sufficiency TEXT,
    evidence TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Machine annotation evaluations table
CREATE TABLE machine_annotation_evaluations (

```

```
id SERIAL PRIMARY KEY,  
model_annotation_id INTEGER REFERENCES model_annotations(id),  
evaluator TEXT,  
task_id INTEGER REFERENCES tasks(id),  
ranking INTEGER,  
accuracy_score TEXT,  
comment TEXT,  
timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Create Indexes

```
-- Performance optimization indexes  
CREATE INDEX idx_physicians_npi ON physicians(npi);  
CREATE INDEX idx_reviews_physician_id ON reviews(physician_id);  
CREATE INDEX idx_tasks_physician_id ON tasks(physician_id);  
CREATE INDEX idx_tasks_assigned_to ON tasks(assigned_to);  
CREATE INDEX idx_human_annotations_physician_id ON  
human_annotations(physician_id);  
CREATE INDEX idx_human_annotations_evaluator ON  
human_annotations(evaluator);  
CREATE INDEX idx_model_annotations_physician_id ON  
model_annotations(physician_id);  
CREATE INDEX idx_machine_evaluations_model_annotation_id ON  
machine_annotation_evaluations(model_annotation_id);
```

Data Import

Using Go Import Tool

```
cd backend/cmd/import  
go run main.go
```

Using Python ETL Script

```
cd database  
python etl.py
```

Query Examples

Get Complete Physician Information

```
SELECT p.*,
       COUNT(r.id) as review_count
FROM physicians p
LEFT JOIN reviews r ON p.id = r.physician_id
WHERE p.npi = 1659371102
GROUP BY p.id;
```

Get Annotation Progress

```
SELECT
  p.doc_name,
  t.assigned_to,
  COUNT(DISTINCT ha.trait) as completed_traits,
  COUNT(DISTINCT ma.trait) as total_traits
FROM physicians p
JOIN tasks t ON p.id = t.physician_id
LEFT JOIN human_annotations ha ON p.id = ha.physician_id
  AND t.id = ha.task_id
LEFT JOIN model_annotations ma ON p.id = ma.physician_id
WHERE t.assigned_to = 'user001'
GROUP BY p.id, p.doc_name, t.assigned_to;
```

Get Model Evaluation Statistics

```
SELECT
  ma.model_name,
  mae.accuracy_score,
  COUNT(*) as evaluation_count,
  AVG(mae.ranking::numeric) as avg_ranking
FROM model_annotations ma
JOIN machine_annotation_evaluations mae ON ma.id =
mae.model_annotation_id
GROUP BY ma.model_name, mae.accuracy_score
ORDER BY ma.model_name, avg_ranking;
```

Backup and Recovery

Backup Database

```
pg_dump physicians > physicians_backup.sql
```

Restore Database


```
psql physicians < physicians_backup.sql
```

Performance Optimization Suggestions

1. **Index Optimization:** Create indexes for frequently queried fields
 2. **Partitioning:** Consider time-based partitioning for large tables (e.g., reviews)
 3. **Connection Pooling:** Use connection pooling to manage database connections
 4. **Query Optimization:** Use EXPLAIN to analyze slow queries
 5. **Caching:** Use Redis caching for hot data
-

Data Migration

When upgrading database structure, refer to migration SQL scripts in the `backend/db/` directory:

- `migration_new_workflow.sql` - New workflow migration
 - `rebuild_database.sql` - Rebuild database
 - `clean_database.sql` - Clean database
-

Monitoring and Maintenance

Regular Maintenance Tasks

```
-- Update table statistics
ANALYZE;

-- Clean up unused data
VACUUM;

-- Rebuild indexes (if needed)
REINDEX DATABASE physicians;
```

Monitoring Queries




```
-- View current active connections
SELECT * FROM pg_stat_activity WHERE datname = 'physicians';

-- View table sizes
SELECT
    schemaname,
    tablename,
    pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename))
as size
```



```
FROM pg_tables
WHERE schemaname = 'public'
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC;
```

Version History


v1.2.0 (Current Version)

-  Added machine_annotation_evaluations table
-  Improved index structure
-  Optimized query performance

v1.1.0

-  Added tasks and human_annotations tables
-  Refactored model_annotations table structure

v1.0.0

-  Basic physicians and reviews table structure