How Medical Reports Get Stored in the Database

A Simple Guide to Understanding the Storage Process

The Big Picture

When you upload a medical report PDF, it goes through this journey:

```
📄 PDF Upload
    ↓
🔍 OCR Processing (Extract text & data)
    ↓
🧹 Normalize Data (Clean & organize)
    ↓
☁️ Save JSON to Cloud Storage
    ↓
💾 Save to Database ← YOU ARE HERE
```

What Gets Stored?

1. Report Record (1 per PDF)

Think of this as the "folder" for the report:

What	Example
Patient	John Doe (linked by patient_id)
File ID	abc-123-def-456
Type	Full Blood Count
Date	June 3, 2025 at 9:10 AM
Location	Link to PDF in cloud storage

2. Biomarker Records (Many per report)

These are the actual test results:

Test Name	Value	Unit	Normal Range
WBC	7,970	per cu mm	4,000 - 11,000
Hemoglobin	13.5	g/dL	13.0 - 17.0
Platelets	250,000	per cu mm	150,000 - 450,000

The Process in 5 Simple Steps

Step 1: Read the JSON File

The system reads the cleaned medical data from cloud storage:

```
Location: users/199512345678/processed/abc-123_normalized.json
```

This file contains:

- Patient name, age, gender
- Report type and date
- All biomarker values
- Reference ranges

Step 2: Create the Report Record

The system creates one main record in the `reports` table:

What goes in:

- Which patient (patient_id)
- Which file (file_id)
- What type (FBC, Lipid Profile, etc.)
- When collected (sample date)
- Where stored (cloud path)

What comes out:

- A unique report_id (automatically generated)

Step 3: Extract Biomarkers

The system reads the list of test results from the JSON.

For each biomarker, it extracts:

- Name (e.g., "Hemoglobin")
- Value (e.g., 13.5)
- Unit (e.g., "g/dL")
- Reference range (e.g., 13.0 to 17.0)
- Flag (e.g., "High" or "Low") if applicable

Step 4: Handle Reference Ranges

The JSON stores ranges as `[minimum, maximum]`, but the database needs two separate columns:

JSON format: `[13.0, 17.0]`
Database format:

- ref_min = 13.0
- ref_max = 17.0

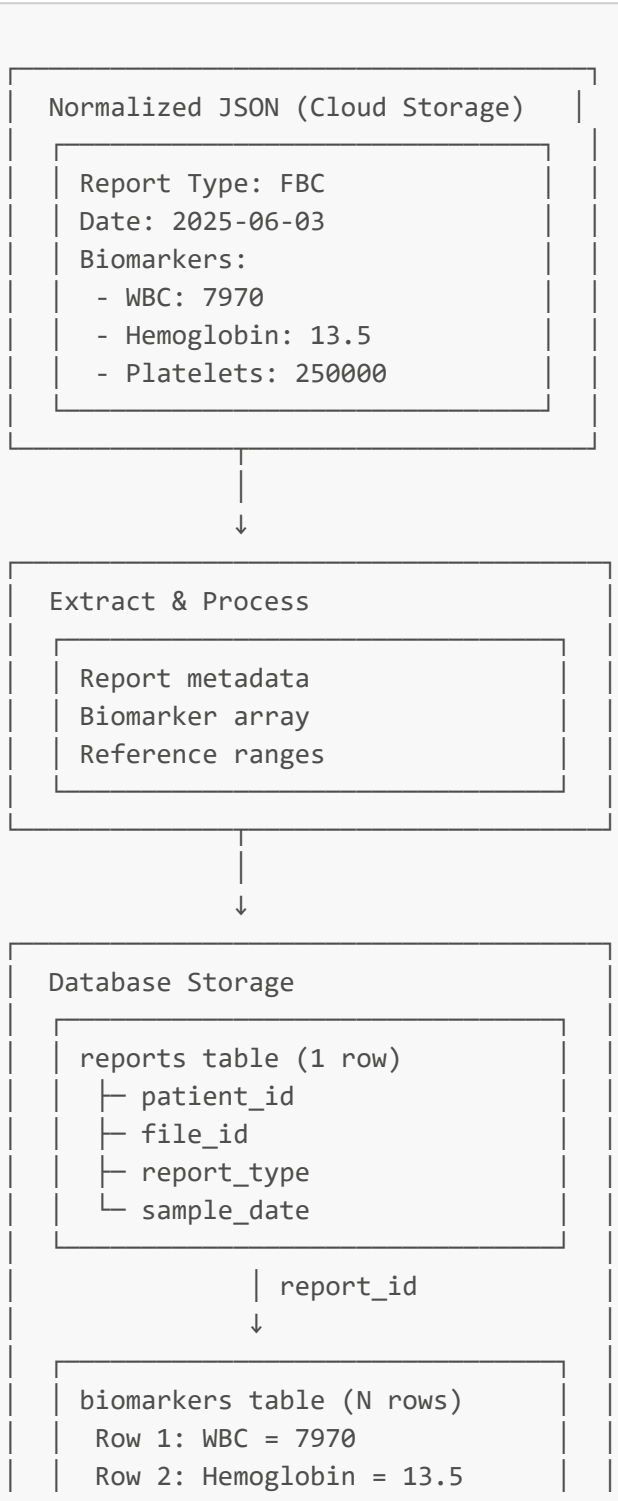
The system automatically converts this.

Step 5: Save All Biomarkers

All biomarkers are saved at once (bulk insert) into the `biomarkers` table.

Each biomarker is linked to the report using the report_id from Step 2.

Visual Flow



```
│  │  Row 3: Platelets = 250000     │  │
│  └────────────────────────────────┘  │
└──────────────────────────────────────┘
```

Example: FBC Report

Input (JSON from Cloud)

```
Report Type: Full Blood Count
Sample Date: June 3, 2025 at 9:10 AM

Biomarkers:
1. WBC = 7,970 (range: 4,000 - 11,000)
2. Hemoglobin = 13.5 (range: 13.0 - 17.0)
3. Platelets = 250,000 (range: 150,000 - 450,000)
```

Output (Database)

reports table:

Field	Value
patient_id	550e8400-...
file_id	fbc-2025-06-03
report_type	Full Blood Count
sample_collected_at	2025-06-03 09:10:00
created_at	2026-02-07 14:00:00

biomarkers table:

biomarker_name	value	unit	ref_min	ref_max
WBC	7970	per cu mm	4000	11000
Hemoglobin	13.5	g/dL	13	17
Platelets	250000	per cu mm	150000	450000

Why This Design?

☑ Benefits

One Report, Many Biomarkers

- Easy to add new biomarkers

- Each test result is a separate row
- Can query individual biomarkers

Connected Data

- Reports are linked to patients
- Biomarkers are linked to reports
- Delete a report → biomarkers automatically deleted

Fast Queries

- "Show me all WBC values for this patient"
- "Find reports with high cholesterol"
- "List all tests from June 2025"

Common Questions

Q: What if the JSON is missing data?

A: The system uses safe defaults:

- Missing reference range → stored as NULL
- Missing value → uses 0 (rare)
- Missing biomarkers → report created with 0 biomarkers

Q: What if something fails?

A: Partial success is allowed:

- Report saved ☑, but biomarkers failed ❌
- System returns a warning
- Data in cloud storage is still intact

Q: How are reference ranges handled?

A: The system automatically converts:

- JSON: `"ref_range": [12.0, 16.0]`
- Database: `ref_min: 12.0, ref_max: 16.0`

Q: Can I query the data?

A: Yes! Use the API:

- `GET /reports/nic/{nic}` - List all reports for a patient
- `GET /report/id/{report_id}/complete` - Get report + biomarkers
- `GET /report/id/{report_id}/biomarkers` - Just biomarkers

Summary

The Journey

1. **JSON files** are created during OCR processing
2. **Saved to cloud** for permanent storage
3. **Read by worker** after processing completes
4. **Extracted** into report metadata + biomarker array
5. **Stored in database** for fast queries

What Gets Stored

- **1 report record** per PDF uploaded
- **N biomarker records** per report (N = number of tests)
- **Reference ranges** split into min/max columns
- **Clinical flags** (High/Low) for lipid profiles

Why It Matters

- **Fast searches** - Find specific tests quickly
- **Data integrity** - Connected records with foreign keys
- **Flexibility** - Can query in many ways
- **Scalability** - Millions of biomarkers no problem

File Locations

Where the code lives:

- Main function: `app/services/reportService.py`
- Called by: `app/workers/ocr_worker.py`
- Database connection: `app/db/supabase.py`

Where the data lives:

- Cloud storage: `users/{nic}/processed/{file_id}_normalized.json`
- Database tables: `reports` and `biomarkers`

That's it! Medical reports → Database in 5 simple steps. ✨