Assignment: Module 5 Final Project – Data Import

Nicholas Hooper; Amit Patel; Ganya Reddy; Trusha Sonawane; Yash Singh

05 DEC 2024

Northeastern University: College of Professional Studies

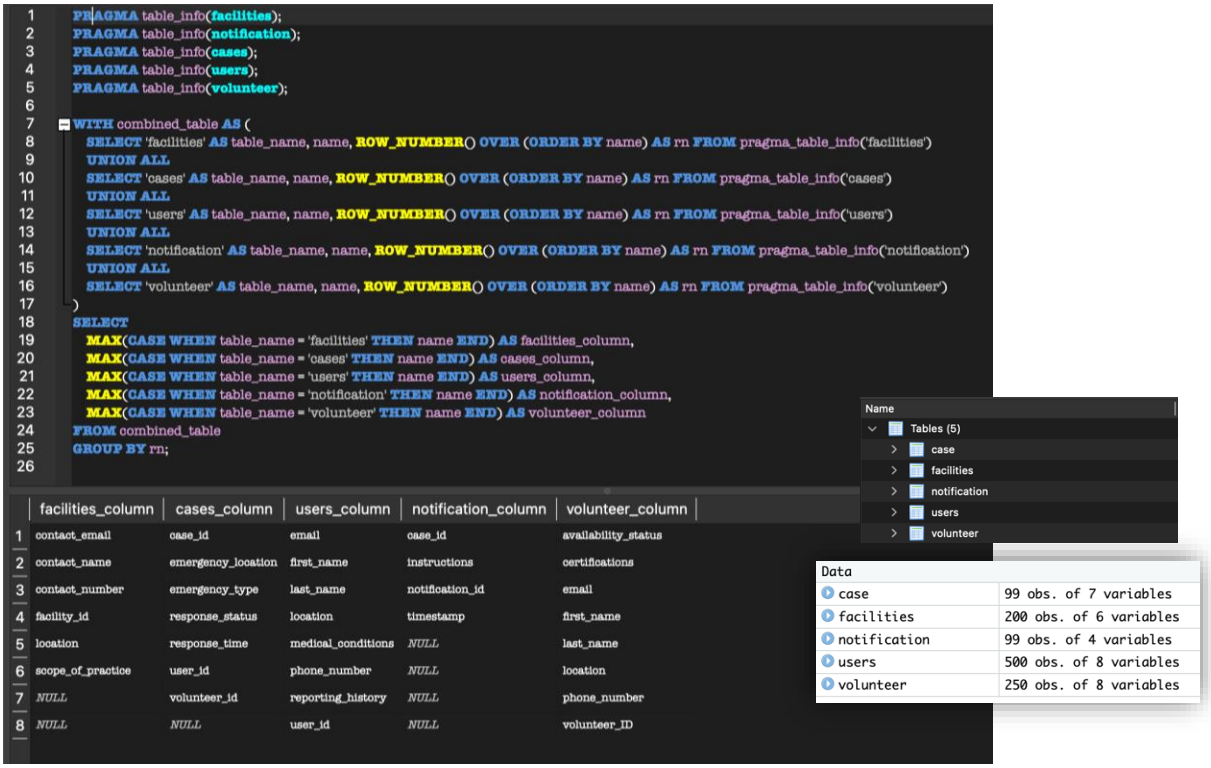
Introduction

The architecture of a client-server application plays a critical role in determining how data is managed, accessed, and stored. This report outlines the proposed solution for a client-server architecture, detailing the communication between the application and the database, the integration of cloud hosting services, and the anticipated storage requirements. The app will function in a client-based infrastructure connecting local data stores to a remote data warehouse, leveraging a third-party cloud service for scalability and reliability. Additionally, the chosen database model, relational, aligns with the application's need for complex queries, analytics, and transactions with strict consistency. This approach ensures efficient handling of expected storage demands, while maintaining performance and security.

As a reminder: Our project concentrates on developing a *Community Health Support and Emergency Response App* aimed at connecting trained local volunteers with people in need during medical emergencies. *The goal is to bridge the critical gap between the incident and professional medical response, reducing response times and providing basic support during crises.*

Database Architecture

Data Sources
Using both SQL and R we can see the strucuture of the sample data below.



```
PRAGMA table_info(facilities);
PRAGMA table_info(notification);
PRAGMA table_info(cases);
PRAGMA table_info(users);
PRAGMA table_info(volunteer);

WITH combined_table AS (
    SELECT 'facilities' AS table_name, name, ROW_NUMBER() OVER (ORDER BY name) AS rn FROM pragma_table_info('facilities')
    UNION ALL
    SELECT 'cases' AS table_name, name, ROW_NUMBER() OVER (ORDER BY name) AS rn FROM pragma_table_info('cases')
    UNION ALL
    SELECT 'users' AS table_name, name, ROW_NUMBER() OVER (ORDER BY name) AS rn FROM pragma_table_info('users')
    UNION ALL
    SELECT 'notification' AS table_name, name, ROW_NUMBER() OVER (ORDER BY name) AS rn FROM pragma_table_info('notification')
    UNION ALL
    SELECT 'volunteer' AS table_name, name, ROW_NUMBER() OVER (ORDER BY name) AS rn FROM pragma_table_info('volunteer')
)
SELECT
    MAX(CASE WHEN table_name = 'facilities' THEN name END) AS facilities_column,
    MAX(CASE WHEN table_name = 'cases' THEN name END) AS cases_column,
    MAX(CASE WHEN table_name = 'users' THEN name END) AS users_column,
    MAX(CASE WHEN table_name = 'notification' THEN name END) AS notification_column,
    MAX(CASE WHEN table_name = 'volunteer' THEN name END) AS volunteer_column
FROM combined_table
GROUP BY rn;
```

	facilities_column	cases_column	users_column	notification_column	volunteer_column
1	contact_email	case_id	email	case_id	availability_status
2	contact_name	emergency_location	first_name	instructions	certifications
3	contact_number	emergency_type	last_name	notification_id	email
4	facility_id	response_status	location	timestamp	first_name
5	location	response_time	medical_conditions	*NULL*	last_name
6	scope_of_practice	user_id	phone_number	*NULL*	location
7	*NULL*	volunteer_id	reporting_history	*NULL*	phone_number
8	*NULL*	*NULL*	user_id	*NULL*	volunteer_ID

The database will need to be able to handle 5 primary data sources. Users, volunteers, facilities, cases, and notifications.

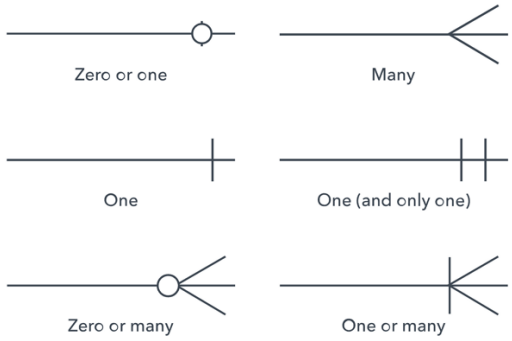


Figure 2: Crows Feet Definitions

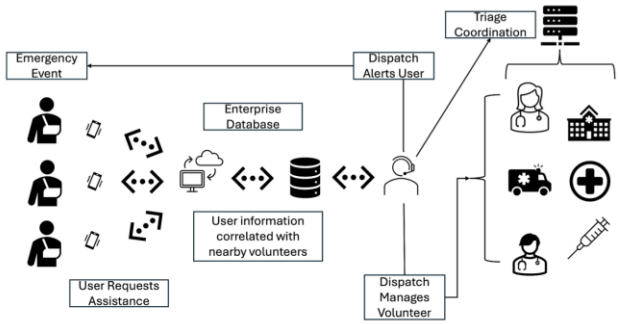


Figure 3: Communication and Relationships Flow

Client-Server Relationship

Appendix 1 is the ERD and illustrates the totality of the system architecture. It is generally a star schema with protocols built in for security, governance, and vertical scalability. Figure 3 shows a simplified illustration for how most of the application will comminate data across entities. The communication hub is the dispatch center. Dispatch holds administrative rights to a primary secure cloud, which is where entity data is amalgamated, and new case files are created. Initially, each entity stores their data on a local device, like their cell phone, and then uploads it to the hosting center. However, before the data can populate the data lake, filters check for quality and governance.

Forcing each entity to interact with a very small portion of the data will limit latency and variability issues. It also enables a clean local user interface with the host application handling the backend as a service, BaaS, requirements.

Figure 2 shows a key for how 'crows feet' connections are utilized within the ERD in appendix 1. Overall, the dispatch center has exclusively many connections in and out, as it handles multiple entities each with its own data mart while also creating case files and coordinating message packages. Notice that the message packages are one and only one illuding to the strict security protocols surrounding these instructions. Also notice that many users can be associated with a single event, allowing for multi-patient events.

Hosting and Storage Requirements

The cloud hosting service fits into two places within the data flow architecture. The first, as previously described, is in the data lake. This will be a relational solution, hosting user, volunteer, and facility data. A relational storage solution fits this use case as the data is large, uniform, and static. The second cloud host will only manage coordinating messaging via a NoSQL solution as this data will be dynamic and communicating with a wide range of recipients and unknown hardware.

Separating the coordinating feature serves two primary functions: data latency and security. The primary data lake will not be inhibited by the changing communication structure and bespoke solutions to coordinating emergencies. The second aspect is in the ability to add additional security protocols to all message packages to ensure sensitive data is kept secure and access is highly restricted.

Growth is an expected aspect of the application. For this use case, large files are not anticipated as a requirement; however, high volume is a potential concern. The afore mentioned partitioning of locally created content separated by two clouds formatted to meet their unique needs should address both the vertical and horizontal growth in the data warehouse and data mart respectively.

The final storage consideration is with indexing. The data is partitioned within the host cloud; however, the data lake is correlated with primary and secondary keys. Upon passing the filter into the data warehouse the keys will be assigned and indexed for faster reference by dispatch. As illustrated in appendix 1, each entity: user, volunteer, event, facility, and message has its own unique primary key, which acts as a secondary key across most tables.

Security Considerations

Security concerns have been referenced throughout this synopsis. Given the nature of health data security and privacy will be paramount to the architecture.

Access to the primary secure cloud, previously referenced as both the data lake and data warehouse, will be accessible in 3 levels: Admin, explorer, and read-only. Dispatch will have the only admin rights and be able to see and manipulate everything within the cloud. Explorer will allow users to access information that is connected to allocated primary and secondary keys. e.g. medical staff at the receiving hospital will be able to see the information they require. Finally, read-only allows an entity to only see the information on its primary key, e.g. the data they uploaded themselves.

Access to the secondary cloud is severely restricted. It will house the most sensitive data, data that needs to make it to its destination un-interrupted and un-manipulated. e.g. a hacker cannot change a victim's location in a coordinating instruction. Once the coordinating package is sent from dispatch it enters a series of firewalls. Only a very select few will have admin access to this cloud and that is the only type of access available.

Conclusion

The proposed system architecture employs a robust client-server relationship with an emphasis on security, scalability, and efficient data handling. The star schema design, illustrated through the ERD in Appendix 1, ensures effective data communication and governance, with the dispatch center acting as the central hub. By limiting each entity's interaction to a minimal data subset, latency and variability issues are mitigated, while maintaining a user-friendly interface. The cloud hosting strategy separates the data lake and message coordination functions, addressing both storage requirements and high-volume data handling. Security is prioritized through multi-level access controls for both the primary and secondary cloud environments, safeguarding sensitive health data and ensuring seamless communication. The architecture is designed for future growth, with vertical and horizontal scalability incorporated to manage increasing data and messaging demands.

References

Coronel, C., & Morris, S. (2016). *Database systems: Design, implementation, and management* (12th ed.). Cengage Learning.

Oracle. (n.d.). *What is a data warehouse?* Retrieved December 4, 2024, from https://www.oracle.com/database/what-is-a-data-warehouse/

Appendix

Appendix 1: