Global Unity Realty Services: Database Requirements Document

Lloyd Thomas

In my CRUD application, I have decided to use Redis as an in-memory key-value storage system for caching data from the Users and Portfolios collections. This strategy is implemented to enhance the efficiency of data retrieval operations and to optimize the caching mechanisms.

Moreover, other documents provide detailed descriptions of how I plan to implement additional functionalities, such as tenant interaction logs and financial report summaries, utilizing Redis to optimize data processing and enhance system efficiency

Objective:
Global Unity Realty Services (GURS) aims to provide comprehensive property management solutions on a global scale, fostering unity and collaboration in the real estate sector.

Problem Description:
The absence of centralized and comprehensive management systems inhibits investors from engaging in diverse property ventures, especially in third-world countries.

Challenges such as navigating complex regulatory environments, overcoming communication barriers with tenants, tracking maintenance tasks and expenses, generating detailed financial reports, mitigating compliance risks, and managing documents heighten these concerns.

Database Rules:

Client Users
Each document in the Users collection has a unique identifier (user_id).
Contains attributes such as first_name, last_name, and last_login for tracking purposes.
A user can manage multiple property portfolios, reflected by a portfolios array containing portfolio_ids of portfolios managed by the user.
A user can have multiple transactions, reflected by a transactions array containing transaction_ids that pertain to the user.

Property Portfolios
Each document in the Portfolios collection has a unique identifier (portfolio_id).
Contains attributes such as name, total_value, and status.
Each portfolio displays a list of properties, leases, financial reports, and transactions made by the managing client user, included as arrays of property_ids, lease_ids, report_ids, and transaction_ids, respectively.
A portfolio must be associated with a client user and cannot exist independently, represented by a user_id attribute linking to the respective user.
If a portfolio is deleted, all properties and transactions linked to it should also be deleted.

Properties
Schema Rules:
Each document in the Properties collection has a unique identifier (property_id).
Contains attributes such as value, address, acquisition_date, square_footage, and property_type.
A property is associated with one portfolio, represented by a portfolio_id.
A property can host multiple financial reports and leases, included as arrays of report_ids and lease_ids, respectively.
If a property is deleted, all leases and potentially related financial reports should also deleted.

Financial Reports
Schema Rules:
Each document in the Financial_reports collection has a unique identifier (report_id).
Contains attributes such as report_type, period_start_date, period_end_date, income, expenses, and report_summary.
Each financial report is attached to a property, represented by a property_id.
A financial report can exist independently from its property, but the relationship is significant as each report pertains to one property only.

Transactions
Schema rules:
Each document in the transactions collection has a unique identifier(transaction_id).
Contains attributes such as transaction_date, description, transaction_type.id, transaction_type.name, amount, portfolio_id
A transaction is linked to a portfolio by the portfolio_id, and if a portfolio is deleted, attached transactions should be deleted as well.
There can be numerous transactions in one portfolio, but each transaction can only belong to one portfolio.

Leases
Schema Rules:
Each document in the Leases collection has a unique identifier (lease_id).
Contains attributes such as lease_start_date, lease_end_date, rent_amount, renewal_date, and summary.
A lease is associated with one property, represented by a property_id.
If the property associated with a lease is deleted, the lease should also be deleted.
A lease can have zero or more tenants, included as an array of tenant_ids under tenants.

Tenants
Schema Rules:
Each tenant has its own unique identifier (tenant_id). Each tenant table includes the tenant's first and last name alongside their email. The Tenants are also connected to the Lease table by having a unique lease id. Furthermore, a tenant can have one to many leases.

Possible Nouns and Verbs

Nouns:
Client User
Property Portfolio
Lease
Tenant
Property
Property Type
Financial Report
Report Type
Transaction
Transaction Type

Verbs:
Manage (e.g., manage property portfolios)
Owns (e.g., owns a property)
Initiates (e.g., initiates a lease agreement)
Create (e.g., create a new client user)
Update (e.g., update property information)
Delete (e.g., delete a financial report)
View (e.g., view property portfolio details)
Assign (e.g., assign a tenant to a lease)
Generate (e.g., generate a financial report)
Calculate (e.g., calculate total income)
Filter (e.g., filter transactions by type)
Search (e.g., search for properties by address)
Analyze (e.g., analyze lease renewal dates)