BEMM459
Database Technologies for Business Analytics

Project Report on
Implementation of Polyglot Persistence for an Online Food Ordering Application
Assessment No: 1

Table of Contents

A1. Description of organisation and application conceptualisation

1.1. Organisation Description

EasyEats

FOOD AT YOUR FINGERTIPS

EasyEats is a food order application in the UK. Users can access the online food order platform, search for the right restaurant or right dishes and place the orders then pay easily with a good amount of discount. Orders will be sent to restaurants then the restaurants will prepare and deliver food to the customers. One of the notable features of EasyEats is to search for foods but exclude foods with the ingredient that they are allergic to or do not want in their foods.

1.2. Application Use Case

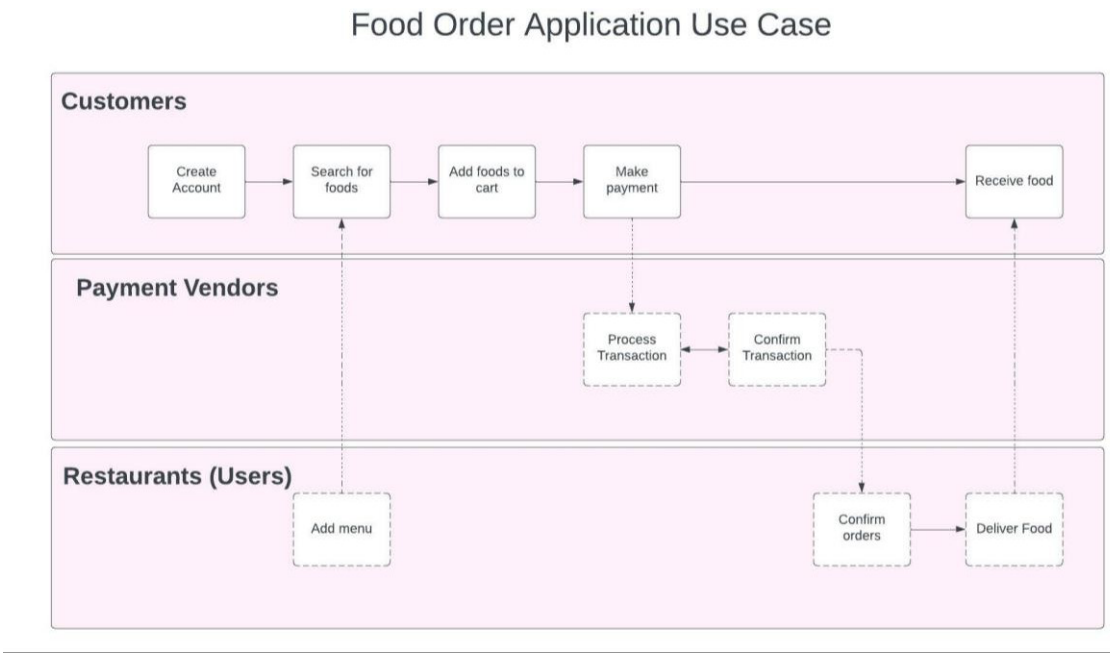


Fig 1: User Case of EasyEats - Food Order Application System

The application has the following functions:

- Customers can search for the restaurant they want, the restaurant will include the type of food, location, menu, price.
- Customers can search for the dish or the drink they want and know what dishes are available or not available now.

- Customers can add multiple dishes and drinks and add food to their cart, only from one restaurant

- Customers can place an order which only consists of food from one restaurant.

- Customers can make payments online with third-party vendors (maximum of 10 minutes to proceed with the payment). Online transaction gateways can be VISA, Mastercard, Apple Pay, and Paypal.

- Customers can create an account with their name, phone number, and email address at first.

- Customers can edit their address, add multiple addresses, and set one address as a primary address.

- Restaurant owners can create their accounts and add menus, cuisines, ingredients. The menu of restaurants can be changed depending on weekdays, seasons, availability of ingredients. Menus can contain various categories ("drinks", "starters", "salads", "sandwiches", "pizza" etc). Each category can include different dishes. Each dish can include different ingredients, allergy information, spicy level, diet notes (vegan, halal, vegetarian). The menu can be created, added, edited, and deleted by the restaurants frequently.

- A restaurant owner can own multiple restaurants and a restaurant can have multiple owners having many to many relationships.

1.3. Database Choices

EasyEats implements both SQL database and NoSQL (Redis) database. SQL database will be used to contain information on customers' login credentials and confidential data, restaurant owners, menus, orders, order details, ratings. While NoSQL database - Redis will be used to store and process data for food carts, menu status and time count for orders. We implemented both SQL database and NoSQL database due to the advantages of both types of databases for one application.

SQLAlchemy library with Object Relational Mapper (ORM) tool to implement SQL database using Sqllite3 (SQLAlchemy, n.d). This library eases the communication between databases and Python. SQL database is a suitable option for EasyEats as follows:

- **Data integrity, Data consistency and Data redundancy:** Information about customers, restaurant owners, orders, ratings are crucial in terms of accuracy and consistency. The set of integrity rules will help to ensure that the data is accurate and control the data redundancy (SQLAlchemy, n.d).

- **Data Manipulation:** Using SQL ensured to perform CRUD operations (Create, Read, Update and Delete) which enhances data controlling, data sharing, and data analysing (SQLAlchemy, n.d).
- **Data independence:** SQL allows a high level of data independence. The benefits are affordable (SQLAlchemy, n.d).

- **Database system maintenance, improved data quality, enforced database security, ease of implementation** (SQLAlchemy, n.d)**.**

Redis is chosen to implement the food_baskets (TTL: Time to Live) and allergy information (pattern matching) in EasyEats. Redis is a key-value database and in-memory database. Redis is a good database choice for implementing food_baskets and pattern matching of customer's allergy because of the following reasons below:

- **Real-time data store:** EasyEats requires real-time features. The availability of foods is important throughout the food order and checkout processes. Redis supports a real-time data store that helps restaurant owners to update the availability of foods in the restaurants. The shopping cart feature requires more availability over consistency. Redis can meet this requirement with fast sub-millisecond response times and subsequently enhance user experience (Amazon, n.d).

- **Caching and session storage:** The delivery app provides the food cart feature. Data from food carts will be stored in the cache. This solution will reduce data access latency, enhance throughput, and increase lift the load off the relational database (Amazon, n.d).

- **Queuing and Messaging:** Orders with payment completed will be added to a queue and proceeded correspondingly. Redis data structure can support lightweight queues which increase data processing speed (Amazon, n.d). Redis with high performance in chat rooms and messages can be a good choice for further product improvement. Future support can develop messaging features between restaurants and customers.

1.4. Business Use Case

The SQL database and NoSQL database allow the internal team to retrieve data for business purposes. The business can have a variety of questions to support future business developments and decisions. To increase the successful transaction rate, the business wants to discover the relationship between discount percentages, ratings and the average time for successful transactions. Users can retrieve data from the Redis cache (order_id, menu_id, time, successful transactions) and SQL database for the Payment details table including order_id, menu_id, order_date, total_amount, discount_percentage and the Rating table including menu_id, score.

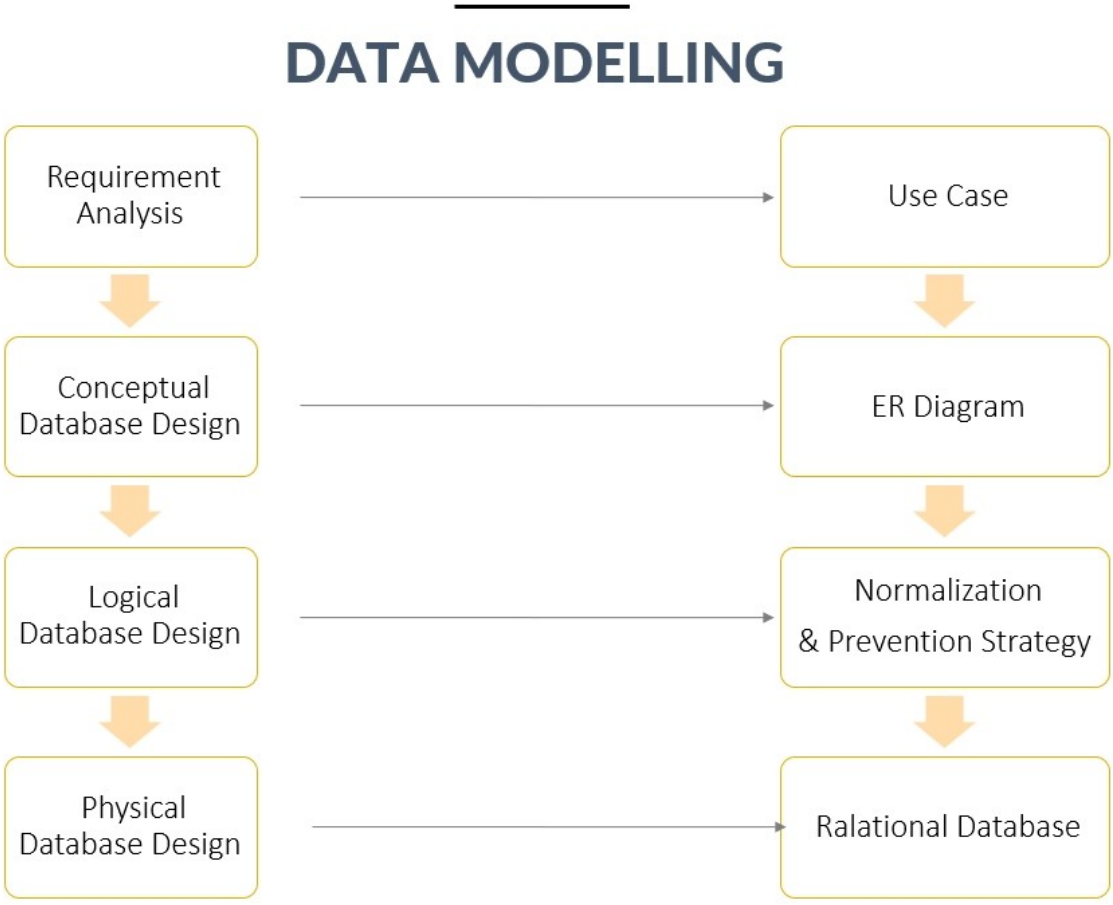


Fig 2: Flow Chart Summarising Relational Data Modelling

A2. Entity Relationship

Fig 3: ER Diagram

Entity Relationship Diagram is used for the design and representation of the relationship between the data. It is a top-down approach to database design and follows logical design to remove data redundancy. The main data objects are named Entities and the complete dataset of that entity is known as Entity Set. Those entities with their details characterized are Attributes, these attributes are important as they are used to identify the entity and different entities are connected through different relationships. An attribute can be of any type like a simple attribute that has atomic value, a composite attribute that has multiple values, a single-values attribute can be a single or composite attribute, a multi-valued attribute that has multiple values or a derived attribute which are not present in the database but derived using other attributes. Moreover, these attributes can have a large amount of data and to fetch a particular data we will be needing keys to identify specific data.

For that we have different relational keys: -
1). Primary key: that can uniquely identify each record.
2). Composite Primary key: Set of 2 primary keys in one table
3). Super key: a set of attributes that uniquely identifies any row
4). Alternate key: not a primary key, but a candidate key

5). Candidate key: a subset of super key, a key that has no redundant attribute
6). Unique key: key which identifies records uniquely
7). Foreign key: the key that provides a link between 2 tables.

As mentioned about relationships, relationships describe the relationship between the entities. Binary Relationship is the most common degree of relationship which is divided into three types:

a) One to One relationship (1:1): means one element of A can be linked to one element of B.
b) One to Many relationships (1:*): means one element of A can be linked to any element of B.
c) Many to Many relationships (*:*): means multiple records of table A associated with multiple records in table B, the direct implementation of this relationship is not allowed as we cannot find the correct result of it. So, we use Link Table for many too many relationships which breaks the relationship into one too many with the help of the third table.

Entities	Attribute	Description	Domain attribute and/or Data type	Null	Multi-values
Payment_info	**payment_id (PK)**	Uniquely identify each id	Numbers	No	No
	order_id (AK)	Uniquely identify each order	Numbers	No	No
	amount	Amount of the order	Numbers	No	No
	paid_by	Payment by the user	Person who paid the amount	No	No
	payment_date	Date of payment	Date format	Yes	No
	processed_by	Who processed the payment	User who processed	No	No
	means	Different modes of pament	With 4 types	No	No
Order_details	**order_details_id (PK)**	Uniquely identify order detail id	Numbers	No	No
	order_id (AK)	Uniquely identify order id	Numbers	No	No
	menu_id	Identify menu id	Numbers	No	No
	amount	Amount of each order	Numbers	No	No
	no_of_serving	Estimated number of servings	Numbers	No	No
	total_amount	Total amount to be paid	Numbers	No	No
Order	**order_id (PK)**	Uniquely identify order id	Numbers	No	No
	customer_id	Identify customer id	Numbers according to their id's	No	No
	order_date	Date of order	Date format	Yes	No
	total_amount	Amount to be paid	Numbers according to their id's	No	No
	order_status	Status of order	0 - pending 1- confirmed 2- cancelled	No	No
	processed_by	Order processed by people	User who processed	No	No
	discount_coupon	Disount coupon applied	Different coupons for discounts	No	No
	discount_percentage	Percentage of discount given	different discount percentage offered	No	No
	after_discount	After discount amount	Final payment to be made after discount	No	No

Customer	**customer_id (PK)**	Uniquely identify cusomer id	Number od customers	No	No
	customer_first_name	First name of customer	Text	No	No
	customer_last_name	Last name of customer	Text	No	
	customer_middle_name	Middle name of customer	Text	No	
	customer_email	Email address of customer	Text and include @		
	customer_phone_numbe	Phone number of customer	Numbers with particular code		
	profile_image				
	customer_username	Username of customer	Alphanumberic pattern		
	customer_password	Password of customer	Alphanumberic p		
	account_status	Status of customer	1- 0-		
Rating	**rating_id (PK)**	Uniquely identify id	N		
	menu_id				
	score	Scored by customers			
	remarks	Uniquely marke			
	date_recorded				
	customer_id	Un			
Menu	**menu_id (PK)**				
	menu_name				
	menu				

Menu_type	**menu_type_id (PK)**	Uniquely identify menu type id	Number from 1-5 identified	No	No
	type_name	Uniquely identify type	Different name for different type	No	No
	description	Discription of menu	Detailed discription of menu	No	No
User	**user_id (PK)**	Uniquely identify user id.	5 user identified	No	No
	full_name	Full name of user	Text- Name of the user	No	No
	contact_number	Contact number of user	Number	No	No
	email_address	Email address of user	Text and include @	No	No
	username	Username of user	Name of the user - alphanumeric	No	No
	password	Password of user	Alphanumeric passwords	No	No
Site_info	**site_info_id (PK)**	Uniquely identify site info id	Numbers	No	No
	site_name	Name of the site	Text	No	No
	description	Discription of site	Detailed discription of site	No	No
	contact_info	Contact number of site	Numbers with particular code	No	No
	address	Address of site	Full address of site	No	No
	last_update			Yes	No
	user_id	Identify the user id	Numbers based on user id	No	No

PK – Primary Key

AK- Alternate Key

Fig 4: Entities and Attributes of the Database

User (has) Site Info - * : * relationship

User (user_id, full_name, contact_number, email_address, username, password)
Primary key: user_id
Foreign key: user_id references **User**(user_id)

Site info (site_info_id, site_name, description, conatact_info, addresss, last_update, user_id)
Primary key: site_info_id

A3. Normalization

Database Normalization is a technique for organizing the data in the database. Normalization is done for eliminating redundant data and ensuring data dependencies make sense. If the dataset is not normalized and has data redundancy, it may take a lot of memory and have difficulty in updating the dataset. Tables that contain excess data may have the effect of updating anomalies, further, they are classified into Insertion anomalies, delete anomalies, and modify anomalies.

a) **Insertion Anomalies:** - this happens when inserting vital data into the database is not possible due to missing other data.
b) **Delete Anomalies:** this happens when the deletion of crucial information is caused by the deletion of unwanted information.
c) **Modification Anomalies:** this happens when we need to update a particular attribute of an entity due to some changes. If modification is missed, it will lead to data inconsistency.

Afterwards, with the help of 3 Normal Forms, i.e., 1NF, 2NF and 3NF, we will identify the tables that have desirable properties from those that may suffer from update anomalies.

- **First Normal Form (1NF):** - The tables which represent no multi-valued attribute in any column are in 1NF. Also, having unique names in every column in which ordering is not necessary.

Table: Customer

TABLE : customerid_number

customer_id	customer_phone_number
1	(398)-141-3015
2	(568) 321-4313
2	1-241-637-7576
3	1-353-143-8156,
3	1-960-534-2130
4	(870) 774-9521
5	1-361-726-3713,
5	1-862-737-5979
6	1-919-888-8542
7	1-852-642-6485
8	(668) 345-0371,
8	1-894-983-8873
9	1-280-755-0157,
9	(238) 641-4811
10	1-683-236-7664,
10	(373) 498-7104

TABLE : Customer

customer_id	customer_first_name	customer_last_name	customer_middle_name	customer_email	profile_image	customer_username	customer_passwo	account_st
1	Adria	Snow	Lamberg	vulputate.ullamcorper@aol.org		JPC63CSG1NX	TPC17IEM0RX	1
2	Dorothy	Vance	Underwood	morbi@google.net		MPS39WQU8HX	EWL29JMF3RV	1
3	Uta	Benedict	Espinoza	molestie.sed@icloud.couk		XID94WKH8OH	KGT87PCU7LG	0
4	Colette	Amery	Williams	molestie.sed@protonmail.org		SGN12FAZ0HC	UFF38BHH7FX	0
5	Arthur	Christopher	Jimenez	felis.eget@icloud.couk		EMC65AGT4ZP	VVB46XRB5CU	0
6	Noelle	Emmanuel	Gilbert	neque.sed@aol.ca		QHC57CDC2EW	VLB62PFO4BG	1
7	Calista	Simon	Mitchell	aliquam.vulputate.ullamcorper@hotmail.ca		IFR17RWN3HX	MAR34CKI2UX	1
8	Gwendolyn	Walker	Harmon	justo@icloud.couk		URG16LTK5UT	RBY71JDW4UI	1
9	Rama	Brendan	Hays	felis.donec@icloud.ca		QDG21VDE2BM	CAI75MHW2TL	1
10	Arden	Jordan	Johnson	et@protonmail.org		TNU34PUT5SC	BRE52XHD5HM	1

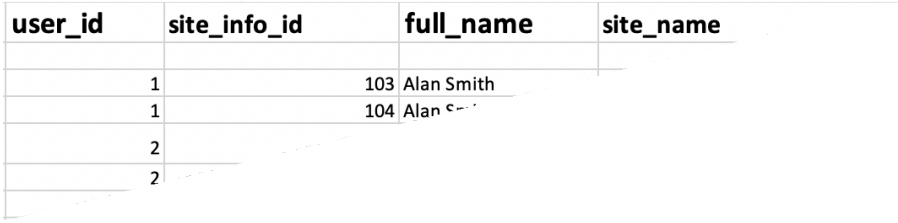
Fig 5: First Normal Form Example

- **Second Normal Form (2NF):** - For tables having in 2NF, two conditions must be satisfied.

    - It should be in 1NF.
    - It should not have a partial dependency.

Also, the table should be having a composite primary key.
Partial dependency means if column B can be defined by any subset of column A, said column B is fully determined by column A. However, if the tables are not having composite primary keys, then we can directly shift to the third normal form, i.e., it is already in the second normal form.

- **Third Normal Form (3NF): -**
  When tables are already in 2NF and have no transitive dependencies. Transitive Dependencies mean non-key attribute depends on an another non key attribute. For e.g., if column A is determined by column B and column B is determined by column C, then column C is transitively dependent on column C.



user_id	site_info_id	full_name	site_name
1	103	Alan Smith	
1	104	Alan	
2			
2			

Fig 6: Third Normal Form Example

This process is done by undertaking all the tables, and now all the tables are in Third Normal Form.

Referential Integrity

Referential Integrity means if a foreign key contains a value, that value must be an existing record in the parent table as the foreign key links the child table record to the parent table record on the matching primary key. To ensure, we examine existing constraints which means considering one too many relationships under which a primary or foreign key can be inserted, updated, or deleted.

For this we are considering two cases:

a) **UPDATE CASCADE:** If we update a specific record in the parent table which has a primary key, the UPDATE CASCADE function will automatically update the record of the child table with the help of a foreign key.

b) **DELETE CASCADE:** If we delete any record from the parent table it will automatically delete referencing record in a child table.

Related tables:

Payment_info (payment_id, order_id, amount, paid_by, payment_date, payment_date, processed_by, means)
Primary key: payment_id
Alternate key: order_id
Foreign key: processed_by references **User** (user_id) **ON Update Cascade on Delete Cascade**

Order_details (order_deatils_id, order_id, menu_id, amount, no_of_servings, total_amount)
Primary key: order_details_id
Alternate key: order_id
Foreign key: menu_id references **Menu** (menu_id) **ON Update Cascade on Delete Cascade**

Order (order_id, customer_id, order_date, total-amount, order_status, processed_by, discount_coupon,
discount_percentage, after_discount)
Primary key: order_id
Foreign key: customer_id references **Customer** (customer_id) **ON Update Cascade on Delete Cascade**

Customer (customer_id, customer_first_name, customer_last_name, customer_middle_name, profile_image,
customer_email, customer_phone_number, customer_username, customer_password)
Primary key: customer_id

Rating (rating_id, menu_id, score, remarks, date_recorded, customer_id)
Primary key: rating_id
Foreign key: menu_id references **Menu**(menu_id) **ON Update Cascade on Delete Cascade**

Menu (menu_id, menu_name, menu_type_id, pmenu_status, price)
Primary key: menu_id
Foreign key: menu_type_id references **Menu_type**(menu_type_id) **ON Update Cascade on Delete Cascade**

Menu_type (menu_type_id, type_name, description)
Primary key: menu_type_id

User (user_id, full_name, contact_number, email_address, username, password)
Primary key: user_id

Site_info (site_info_id, site_name, description, conatact_info, addresss, last_update, user_id)
Primary key: site_info_id

A4. NoSQL Element

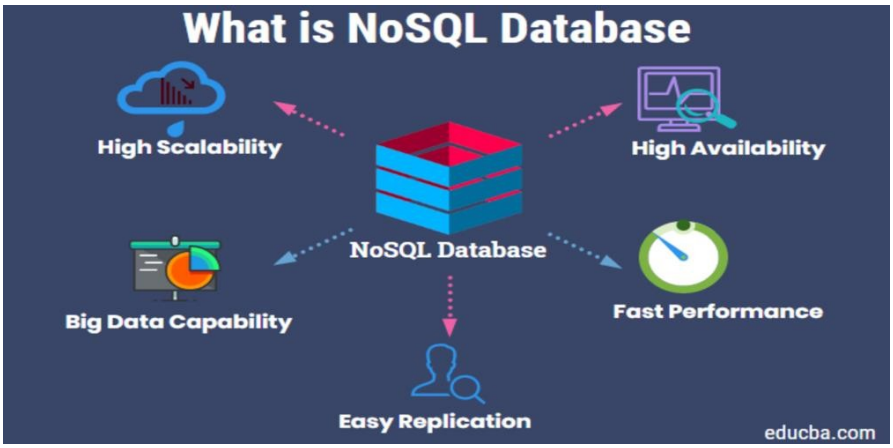


Fig 7 : Advantages of NoSQL Database

NoSQL databases are frequently more scalable and give better performance than relational databases. Furthermore, in comparison to the relational model, the flexibility and ease of use of their data models can speed up development, especially in the cloud computing environment.

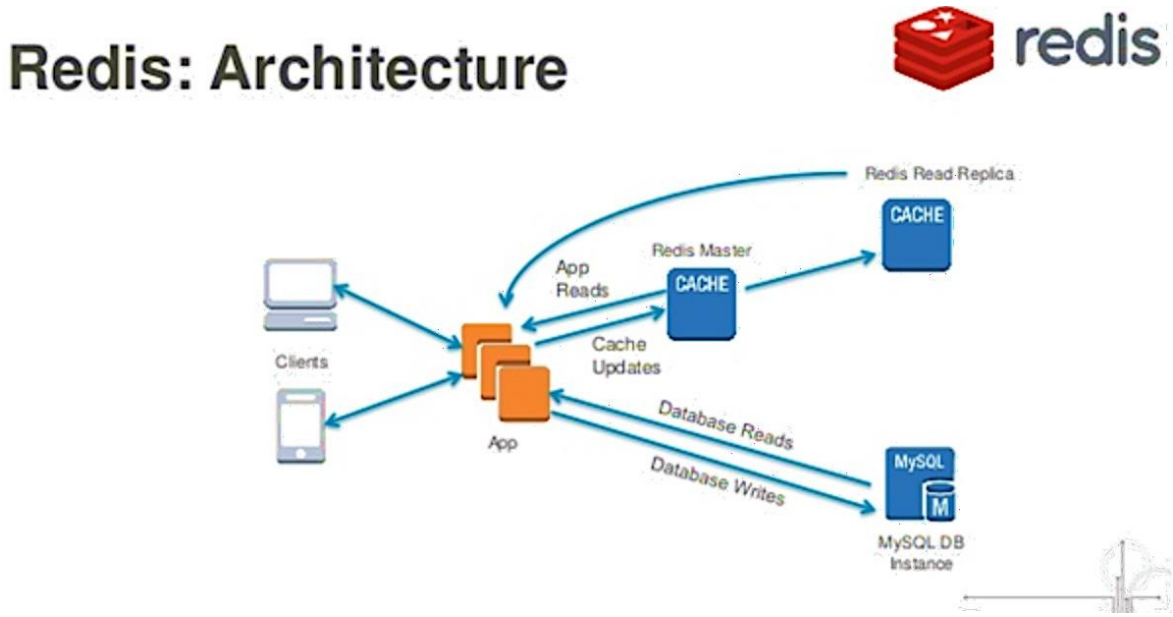


Fig 8: Redis Architecture

Redis (REmote DIctionary Server) is a key-value data store that can be used as a database, cache, or message broker. Strings, hashes, lists, sets, sorted sets, bitmaps, and hyperlogs are among the keys in this NoSQL advanced key-value data storage, which is also known as a data structure server. Because Redis keeps data in

memory, read and write operations are extremely quick. Data can also be written back to the memory or stored on the disk. It is perfect for building scalable, high-performance online applications.

```
import random
# Redis hash of field-value pairs is used. Eash hash has a key that with an integer (we are importing random()
random.seed(203)
```

```
food_baskets = {f"food_basket:{random.getrandbits(32)}": i for i in (
    {
        'customer_id':1,
        'order_id':1,
        'total_amount':4000,
        'quantity:1':4,
        'quantity:2':2,
        'menu_name:1':'banana',
        'menu_status:1':1,
        'menu_status:2':0,
        'menu_name:2':'apples',
        'site_name' : 'Party Fowl',
        'date':'22 may, 2020',
        'time':'14:00',
        'discount_coupon': 'ADIDAS25',
        'discount_percentage':25,
        'amount_after_discount':3000
    },
    {
        'customer_id':2,
        'order_id':2,
        'total_amount':4000,
        'quantity:1':4,
        'quantity:2':2,
        'menu_name:1':'banana',
        'menu_name:2':'apples',
        'menu:1_status':1,
        'menu:2_status':0,
        'site_name' : 'Party Fowl',
        'date':'21 may, 2020',
        'time':'14:00',
        'discount_coupon': 'ADIDAS25',
        'discount_percentage':25,
        'amount_after_discount':3000
    })
```

Fig 9: Implemented code for Hashes from Python code

Hashes work similarly to nested Redis objects in that they can hold any number of key-value pairs. To keep track of users who sign up for our URL-Shortening service, we will utilise a hash. Hashes are useful because they prevent the usage of artificial key prefixes when storing data. We can generate a hash that has its key value pairs instead of using distinct keys. To retrieve all hash values, we only need to keep track of one Redis key.

As shown in the above figure, '**food_baskets'** represents a **namespace** (collection of hashes).

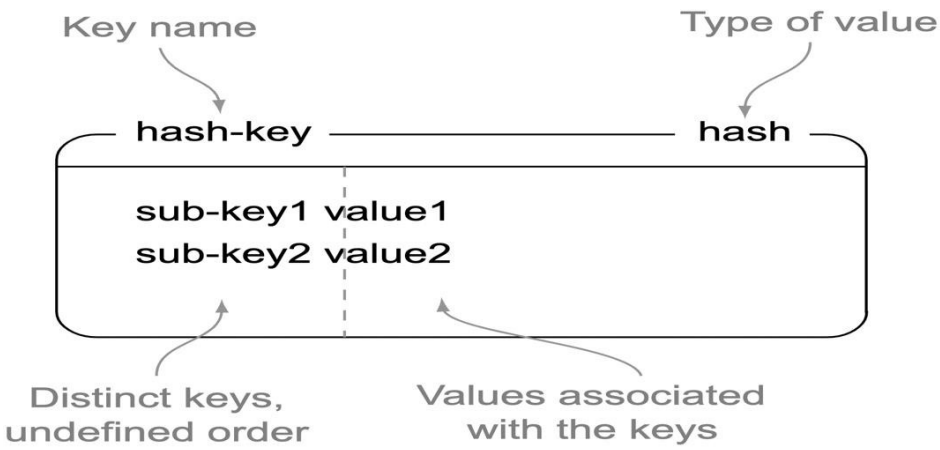


Fig 10: An example pseudo code of a HASH with two keys/values under the key hash-key

Redis hashes cannot nest, unlike the document databases MongoDB and CouchDB (nor can any other complex datatype such as lists). In other words, hashes can only store string values, not sets or nested hashes, for example.

A5. Polyglot Persistence

Polyglot persistence refers to using different data storage technologies to support various data types and their storage needs. Polyglot persistence is the idea that an application can use more than one core database. Various kinds of data are best dealt with in different databases. In short, it means picking up the right database for the right data. It is the same idea behind polyglot programming, which is the practice of writing applications using a mix of languages because different languages are suitable for tackling different problems. One of the main problems faced by implementing polyglot is complexity. Each data storage introduces a new interface to be learned. Also, the thing to remember is that, data storage requires high level of performance, so it is important understand how the database works to get a decent performance out of it. Using the right persistence technology will help, but the complexity and challenge still remains.

So, an organisation should think about different scenarios before implementing the polyglot persistence, some of the questions would be;

- How the data flow will be?
- How to data will be stored for the polyglot persistence?
- Will the organisation be able to manage the complexity that comes with it.?

So, in the application, we have developed, instead of storing data in a single database base and hinder the performance. The data is stored in databases that are best suited for it.

Functionality	Consideration	Database Used
User & Site Details	Stores the details about the restaurant owner and the site and requires data security.	RDMS
Payment Details	Requires transactional updates and data security.	RDMS
Order	Stores the details about the order being made and requires data security.	RDMS
Customer Details	Stores the details of the customer, this requires data privacy and security.	RDMS
Menu Type	Details regarding types of Menus. Requires occasional updating.	RDMS
Restaurant Details	Details of Restaurant and its contact info. It requires occasional updating.	RDMS
Menu Details	Has all the details of the ingredients and customers selected the required dietary preference and allergen information	Redis
Basket Details	It has a timer and the data stored is not important	Redis

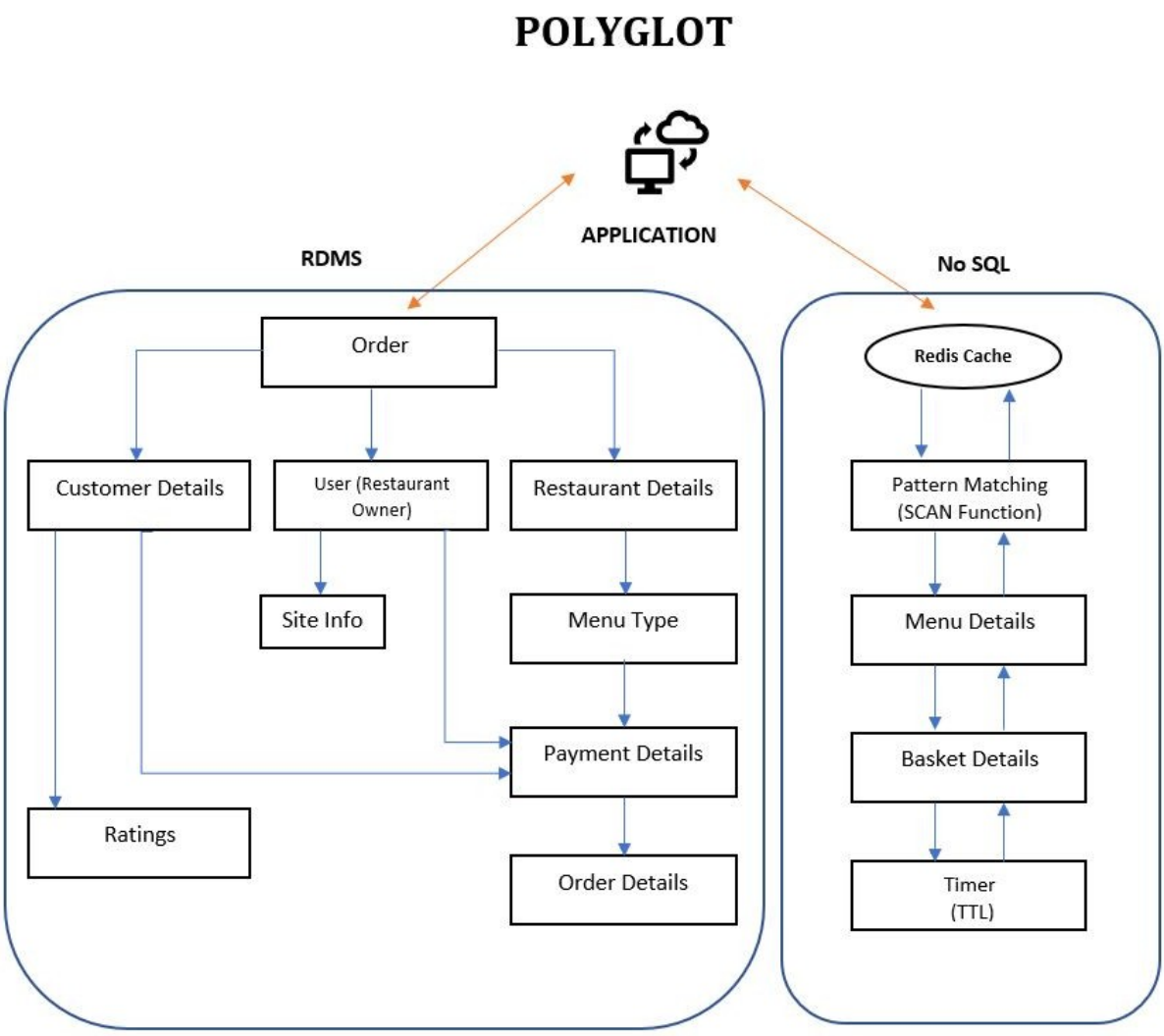


Fig 11: Implementation of Polyglot persistence

Here, the data is stored in RDMS and Redis. When a customer signs up for the service, the required data is collected and stored in the RDMS and when the customer initiates the service, the application requests the data such as Restaurant, Menu Type, Rating etc from the database. Then when the customer gives any dietary preference or allergen information for the order, after Pattern Matching using the SCAN function, the data is shared back from the Redis server to the application. Once the items are selected and moved to the shopping basket and the payment is initiated, a timer will be started in the Redis server using the TTL function. If the payment is not made within the time ends, the data stored in the basket will be deleted and if the payment is made, then the application will confirm the order.

Reference:

- *1.2.4 hashes in Redis*. (2014, October 26). Redis. https://redis.com/ebook/part-1-getting-started/chapter-1-getting-to-know-redis/1-2-what-redis-data-structures-look-like/1-2-4-hashes-in-redis/

- Features - SQLAlchemy. (n.d.). SQLAlchemy - The Database Toolkit for Python. https://www.sqlalchemy.org/features.html

- *Redis tutorial*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/redis-tutorial

- Redis: In-memory data store. How it works and why you should use it. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/redis/

- Seven databases in seven weeks: a guide to modern databases and the NoSQL movement. (n.d.). Exeter Learning Environment. https://vle.exeter.ac.uk/pluginfile.php/3050708/mod_resource/content/3/BEMM459_Week%207%20Lecture_NoSQL_Key-Value.pdf

- *What is NoSQL database*. (2022, March 17). EDUCBA. https://www.educba.com/what-is-nosql-database/

- Connolly, T., Begg, C., & Holowczak, R. (2010). *Business Database Systems*. Addison Wesley.

- Sullivan, D. (2015). *NoSQL for mere mortals*. Addison-Wesley.