WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Review Paper on Database Basics for Developers : Understanding CRUD Operations

Prof. K. M. Jadhav[1], Prof. T. S. Patil[2], Prof. P. S. Sutar[3], Prof. H. S. Bhore[4]

[1,2,3,4]Assistant Professor, General Sciences and Engineering Department, AITRC, Vita, India

Abstract

Databases are integral to modern software development, enabling the storage, retrieval, and management of data. CRUD operations—Create, Read, Update, and Delete—form the foundational interactions with databases. This paper explores the basics of CRUD operations, their implementation in various database systems, and their significance in software development. It also addresses best practices and while working with CRUD operations.Development and deployment technologies for data-intensive web applications have considerably evolvedin the last years. Domain specific frameworks or Model-Driven Web Engineering approaches are examplesof these technologies. They have made possible to face implicit problems of these systems such as quickevolving business rules or severe time-to-market requirements. Both approaches propose the automation ofredundant development tasks as the key factor for their success. The implementation of CRUD operations isa clear example of repetitive and recurrent task that may be automated. However, although web applicationframeworks have provided mechanisms to automate the implementation of CRUD operations, Model-Driven Web Engineering approaches have generally ignored them and its automation has not been properlyfaced yet. This paper presents AutoCRUD, a WebRatio plug-in that automates the generation of CRUDoperations in OMG IFML (Interaction Flow Modelling Language) standard. The suitability of this tool hasbeen evaluated by its application into several real projects developed by a software companyspecialized inmodel-driven web application development.

Keywords: Model Driven Web Engineering, Cost Reduction, CRUD, Model Driven Software Development

INTRODUCTION

Introduction to Database : An organization must have accurate and reliable data for effective decision making. To this end, the organization maintains records on the various facets maintaining relationships among them. Such related data are called a database. A database system is an integrated collection of related files, along with details of the interpretation of the data contained therein. Basically, database system is nothing more than a computer-based record keeping system i.e. a system whose overall purpose is to record and maintain information/data. A database management system (DBMS) is a software system that allows access to data contained in a database. The objective of the DBMS is to provide a convenient and effective method of defining, storing and retrieving the information contained in the database. The DBMS interfaces with the application programs, so that the data contained in the database can be used by multiple applications and users. In addition, the DBMS exerts centralized control of the database, prevents fraudulent or unauthorized users from accessing the data, and ensures the privacy of the data. Generally a database is an organized collection of related information. The organized information or database serves as a base from which desired information can be retrieved or decision made by further recognizing or processing the data. People use several databases in their day-to-day life. Dictionary, Telephone directory, Library catalog, etc are example for databases where the entries are arranged according to alphabetical or classified order. The term 'DATA' can be defined as the value of an attribute of an entity. Any collection of related data items of entities having the same attributes may be referred to as a 'DATABASE'. Mere collection of data does not make it a database; the way it is organized for effective and efficient use makes it a database. Database technology has been described as "one of the most rapidly growing areas of computer and information science". It is emerged in the late Sixties as a result of combination of various circumstances. There was a growing demand among users for more information to be provided by the computer relating to the day-to-day running of the organization as well as information for planning and control purposes. The technology that emerged to process data of various kinds is grossly termed as 'DATABASE MANAGEMENT TECHNOLOGY' and the resulting software are known as 'DATABASE MANAGEMENT SYSTEM' (DBMS) which they manage a computer stored database or collection of data.

LITERATURE REVIEW

1. CRUD Operation on WordPress Database Using C# SQL Client

In this paper, author shows how to create our data table inside the WordPress Website Database using a step-by-step procedure. They demonstrate the CRUD operation using a simple example. The researcher trying to integrate CRUD operation to store their Data inside the online Database can get

valuable references from this work. Using Flywheel Local, the development can be done free of charge. once the development is finalized, we can deploy it inside the online Database. They kept the example simple so new researchers or integrators could deploy it quickly.

2. Automating IFML Specification of CRUD Operations

In this paper, Development and deployment technologies for data-intensive web applications have considerably evolvedin the last years. Domain specific frameworks or Model-Driven Web Engineering approaches are examplesof these technologies. They have made possible to face implicit problems of these systems such as quickevolving business rules or severe time-to-market requirements. Both approaches propose the automation ofredundant development tasks as the key factor for their success. The implementation of CRUD operations isa clear example of repetitive and recurrent task that may be automated. However, although web applicationframeworks have provided mechanisms to automate the implementation of CRUD operations, Model-Driven Web Engineering approaches have generally ignored them and its automation has not been properlyfaced yet. This paper presents AutoCRUD, a WebRatio plug-in that automates the generation of CRUDoperations in OMG IFML (Interaction Flow Modelling Language) standard. The suitability of this tool hasbeen evaluated by its application into several real projects developed by a software company specialized inmodel-driven web application development. The results obtained present evidences of the significantproductivity improvement obtained by the tool, which almost completely removes the developer timededicated to CRUD operation implementation.

3. CRUD Application Using ReactJS Hooks

This Paper is aimed to implement CRUD operations using the library called React.js which is the advanced version of the language Javascript. The question which pops out everyone's mind now is why React? There's lot of open – source framework/ library/ language that is available in today's internet. The purpose of using react is anyone with the little knowledge of javascript can easily learn react and implementation of this library is user-friendly. Also, there's many more packages react can easily get installed in NPM and run the application whatever changes in the code can do. React also delivers good UI for the developer and to create/develop new apps for future generation. There are more than 2000 developers and over one lakh sites making use of react.js among some of the most powerful are Instagram, Netflix, Whatsapp, Uber. It also integrates with any other javascript libraries or framework and speciality of react can make changes in the web browser without doing subsequent refresh of the pages. React can allow the developer to build any complex web application and more importantly mobile application. The CRUD is an acronym for Create, Read, Update and Delete which is very much essential for implementing any strong application with relational Database. Thus, the project results in Creating the user/users with their details, read the users task, updating if any new user is created through 'Create' operation, and finally delete the user/users list if not wanted for an application. Though the 'crud' functions for creating an application can be done in any language, the author specifically uses Javascript library called react.js which is one of the best platforms to create the single page user application interfaces. The overarching goal of this application is to bring easiness for any product development company in need of crud-user functions can stick to this.

4. CRUD Operations in MongoDB

In this paper author will examine the key features of thedatabase management system MongoDB. We will focus on the basicoperations of CRUD and indexes. For our example we will createtwo databases one using MySQL and one in MongoDB. We will alsocompare the way that data will be created, selected, inserted anddeleted in both databases. For the index part we will talk about thedifferent types used in MongoDB comparing them with the indexesused in a relational database.

5. Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA,Hibernate, Spring Data JPA

The role of databases is to allow for the persistence of data, no matter if they are of the SQLor NoSQL type. In SQL databases, data are structured in a set of tables in the relational databasemodel, grouped in rows and columns. CRUD operations (create, read, update, and delete) are usedto manage the information contained in relational databases. Several dialects of the SQL languageexist, as well as frameworks for mapping Java classes (models) to a relational database. A comparison of the most frequentlyused relational database management systems, mixed with the most frequently used frameworksshould give us some guidance about when to use what. The evaluation is conducted based on thetime taken for each CRUD operation to run, from thousands to hundreds of thousands of entries,using the possible combinations in the relational database system and the framework.

6. Performance evaluation for CRUD operations inasynchronously replicated documentorienteddatabase

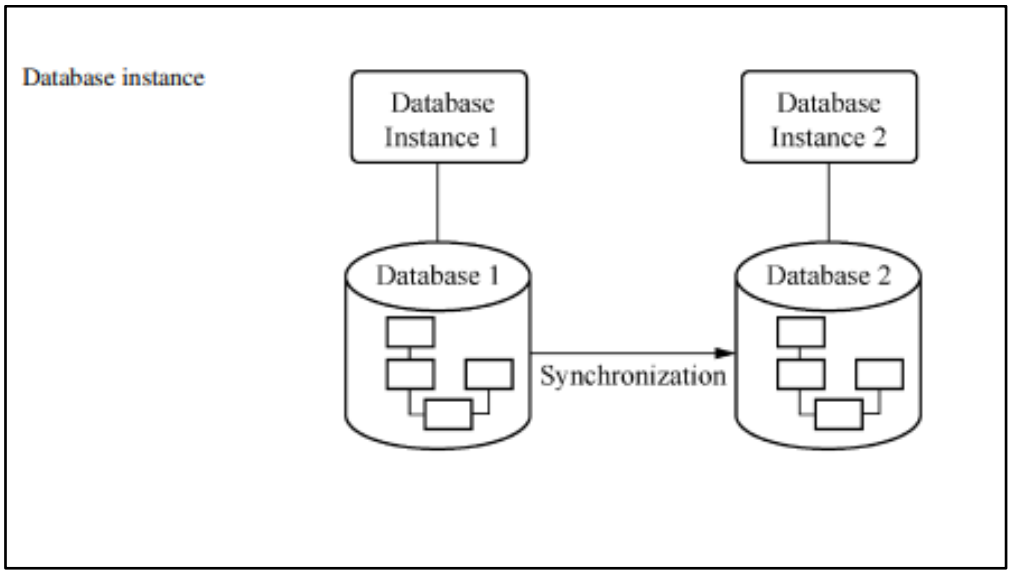
NoSQL databases are becoming increasingly popular as more developers seek new ways for storing information.The popularity of these databases has risen due to their flexibilityand scalability needed in domains like Big Data and CloudComputing. This paper examines asynchronous replication, oneof the key features for a scalable and flexible system. Threeof the most popular Document-Oriented Databases, MongoDB,CouchDB, and Couchbase, are examined. For testing, the execution time for CRUD operations for a single database instance andfor a distributed environment with two nodes is taken into accountand the results are compared with tests outcomes obtained forthree relational database management systems: Microsoft SQLServer, MySQL, and PostgreSQL.

7. Performance Evaluation for CRUD Operations In NoSQLDatabases

With the Web growing rapidly and increase in user-generated content websites such as Facebook and Twitter, there is aneed of fast databases that can handle huge amounts of data. For this purpose, new databases management systemscollectively called NoSQL are being developed. There are many NoSQL database types with different performances,and thus it is important to evaluate performance. To check the performance, three major NoSQL databases calledMongoDB, Cassandra and Couchbase have been considered. For performance analysis, different workloads weredesigned. The evaluation has been done on the basis of read and update operations. This evaluation enables users tochoose the most appropriate NoSQL database according to the particular mechanisms and application needs.

METHODOLOGY

Database system is made for managing data, and database is actually a collection of data, which is expressed as a collection of data files, data blocks, physical operating system files or disk data blocks, such as data files, index files and structure files. But not all database systems are file-based, there are also databases that write data directly into memory. Database instance refers to a series of processes in the operating system and the memory blocks allocated for these processes, which are the channels to access the database. Generally speaking, a database instance corresponds to a database. A database is a collection of physically stored data, and a database instance is the collection of software processes, threads and memory that access data. Oracle is process-based, so its instance refers to a series of processes; and a MySQL instance is a series of threads and the memory associated with the threads. Multi-instance is to build and run multiple database instances on a physical server, each using a different port to listen through a different socket, and each having a separate parameter profile. Multi-instance operation can make full use of hardware resources and maximize the service performance of the database. Distributed database presents unified instances, and generally does not allow users to directly connect to instances on data nodes. A distributed cluster is a set of mutually independent servers that form a computer system through a high-speed network. Each server may have a complete copy or a partial copy of the database, and all servers are connected to each other through the network, together forming a complete global large-scale database that is logically centralized and physically distributed. Multi-instance and distributed cluster.



Key Takeaways

- CRUD operations form the foundation of database interactions in software development.
- CRUD operations map directly to RESTful API methods, enhancing their applicability in web development.
- Best practices like data validation, security measures, and performance optimization are essential for efficient CRUD operations.
- Developers face challenges like concurrency issues, scalability, and data integrity, which require careful planning and implementation.
- Understanding CRUD operations is crucial for building reliable and maintainable applications across various domains.

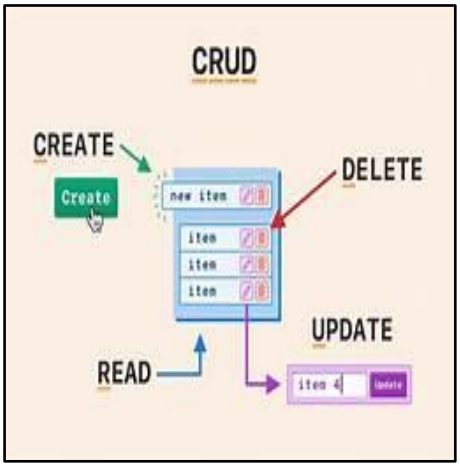
What are crud Operations

CRUD operations refer to the four basic functions that a database must provide to allow for the full manipulation of its data. These operations form the basis of most persistent storage systems, enabling users to interact with data in a meaningful way. Here's a breakdown of each CRUD operation:

- **Create**: This operation allows you to insert or add new records to a database. For example, in an application, this might involve registering a new user or adding a new product to a catalog.
- **Read**: The Read operation is used to retrieve or view data from the database. This is arguably the most frequently performed operation, as it involves fetching stored information, whether that's querying user details, fetching product listings, or generating reports.

- **Update**: When existing records need modification, the Update operation comes into play. This allows changes to be made to already stored data, such as editing user profiles or updating product prices.
- **Delete**: Finally, the Delete operation is responsible for removing data that is no longer needed. This could involve deleting a user account, removing outdated product listings, or clearing temporary records.

In databases, these operations can be executed through SQL queries or via programming interfaces in applications. They are essential for maintaining and manipulating any form of data, ensuring that the data remains up to date and relevant.



REST APIs

The Representational State Transfer (REST) architectural style defines a set of rules for the design of distributed hypermedia systems that have guided the design and development of the Web as we know it. Web services following the REST architectural style are referred to as RESTful Web services, and the programmatic interfaces of these services as REST APIs. The principles governing the design of REST APIs are in big part the result of architectural choices of the Web aimed at fostering scalability and robustness of networked, resourceoriented systems based on HTTP . The core principles are :

Resource addressability. APIs manage and expose resources representing domain concepts; each resource is uniquely identified and addressable by a suitable Uniform Resource Identifier (URI).

– Resource representations. Clients do not directly know the internal format and state of resources; they work with resource representations (e.g., JSON or XML) that represent the current or intended state of a resource. The declaration of content-types in the headers of HTTP messages enables clients and servers to properly process representations.

– Uniform interface. Resources are accessed and manipulated using the standard methods defined by the HTTP protocol (Get, Post, Put, etc.). Each method has its own expected, standard behavior and standard status codes.

– Statelessness. Interactions between a client and an API are stateless, meaning that each request contains all the necessary information to be processed by the API; no interaction state is kept on the server.

– Hypermedia as the engine of state. Resources as domain concepts can be related to other resources. Links between resources (included in their representations) allow clients to discover and navigate relationships and to maintain interaction state.

Together, these principles explain the name "representational state transfer": interaction state is not stored on the server side; it is carried (transferred) by each request from the client to the server and encoded inside the representation of the resource the request refers to.

Best practices for development

Best practices for development Along with the general principles introduced above, a set of implementation best practices have emerged to guide the design of quality APIs . These best practices address the main design aspects in REST APIs: (i) the modelling

of resources, (ii) the identification of resources and the design of resource identifiers (URIs), (iii) the representation of resources, (iv) the definition of (HTTP) operations on resources, and (v) the interlinking of resources. We overview these best practices in the following; a summary with examples is shown in Table 1. Resource modeling. REST APIs can manage different types of resources: documents for single instances of resources, collections for groups of resources, and controllers for actions that cannot logically be mapped to the standard HTTP methods . While modeling resources for REST APIs is not fundamentally different from modeling classes in OO programming or entities in data modeling, there are a couple of recommended naming practices that are typical of REST APIs: singular nouns for documents, plural nouns for collections, and verbs only for controllers , no CRUD names in URLs , no transparency of server-side implementation technologies (e.g., PHP, JSP) (http://www.ibm.com/developerworks/library/ws-restful/). Resource identification. Resource identifiers should conform with the URI format, consisting of a scheme, authority, path, query, and fragment . In the case of Web-accessible REST APIs, the URIs are typically URLs (Uniform Resource Locators) that tell clients how to locate the APIs. In order to improve the readability of URLs, it is recommended to use hyphens instead of underscores, lowercase letters in paths, "api" as part of the domain, and avoid the trailing forward slash . In addition, in its purest form, REST services should avoid declaring API versions in the URL. Resource representation. Resources can support alternative representations (e.g., XML, JSON) and serve different clients with different formats. Which representation to serve should be negotiated at runtime, with the client expressing its desired representation using the HTTP Accept header instruction.

This fosters reusability, interoperability and loose-coupling. APIs should therefore use content negotiation instead of file extensions to specify formats (e.g., .json or .xml). In addition, it is recommended that APIs support (valid) JSON among their representation alternatives. Operations. To manage resources, REST APIs should rely on the uniform set of operations (Post, Get, Put, Delete, Options, Head) defined by the HTTP standard and comply with their standardized semantics:

– Post should be used to create new resources within a collection.

– Get should be used to retrieve a representation of a resource.

– Put should be used to update or create resources.

– Delete should be used to remove a resource from its parent.

– Options should be used to retrieve the available interactions of a resource.

– Head should be used to retrieve metadata of the current state of a resource. REST APIs should thus never tunnel requests through Get or Post, e.g., by specifying the actual operation as a parameter or as part of the resource name.

Conclusion:

CRUD operations are the foundation of database interactions, providing the essential methods for data management. Mastery of CRUD enables developers to build robust, efficient, and scalable applications, regardless of the underlying database technology. By adhering to best practices and addressing challenges, developers can ensure data reliability and system performance, fostering seamless user experiences.In conclusion, if a developer wants to build a webapplication that is fast and flexible, than MongoDB is the rightchoice. If the application designer's main concern is therelation between data and to have a normalized database thatuses ACID transactions, then the right choice is a classicrelational database.

References:

1. [1] Pratik Sharad Maratkar, Protibha Adkar. ReactJS – An Emerging Frontend Javascript Library,issued Iconic Research And Engineering Journals. 2021 ; 4(12) : 99-102.
2. [2] Bhupati venkat sai indla, yogeshchandra puranik. Review on ReactJS" International journal of trend in scientific research and development(IJTSRD). 2021 ; 5(4) : 1137-1139.
3. [3] Ritwik, Anitha Sandeep.: ReactJS and Front End Development International Research Journal of Engineering and Technology (IRJET) 2020 ; 7(4) : 3676-3679.
4. [4] Prateek Rawat, Archana N. Mahajan - ReactJS: A Modern Web Development Framework 2020 ; 5(11) : 698-702.
5. [5] Alok Kumar Srivastava1, Vaishnavi Laxmi2 , Payal Singh3 , Km Pratima4 , Vibha Kirti5. React JS (Open Source JavaScript Library). IJIRT. 2022 ; 8(9) : 342-346.
6. [6] Mayelson de Sousa, Alexandrino Gonçalves.: Human portal – A reactjs case study 15th Iberian Conference on Information Systems and Technologies. 2020 .
7. [7] Arshad Javeed.: Performance Optimization Techniques for Reactjs, IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT) 2019 ; 1067-1073.
8. [8] Xingwei Zhou, Wenshan Hu, Guo-Ping Liu.: React-Native based Mobile App for online Experimentation-39th Chinese Control Conference (CCC) 2020.
9. [9] Fu Kaifang.: Design and implementation of an instant messaging architecture for mobile collaborative learning" Computing, Communication, Control, and Management, CCCM 2009. ISECS International Colloquium on, 2009 ; 3: 287-290.
10. [10] Basques.K.Performance analysis reference https://developers.google.com/web /tools/chrome-devtools/evaluate-performance/ reference - 2020.