METHODS OF MERGING TABLES USING STANDARD FUNCTIONS

Khaydarov Ozodbek Shamsi ugli,

Qarshi State Technical University,

Computer engineering student

Annotation. *Table merging is a fundamental operation in database management and data analysis, where multiple tables are combined to form a unified structure. Using standard functions in programming languages like SQL, Python, or R simplifies the process of merging tables efficiently. This paper explores various methods for merging tables using standard functions, focusing on SQL joins (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN) and the use of functions in Python (such as pandas merge()) and R. The article discusses the significance of these methods in relational databases and data manipulation tasks, highlighting the importance of data integrity, performance, and efficient query execution. Additionally, the paper addresses common challenges, such as handling missing data, managing large datasets, and ensuring accuracy during the merge process.*

Keywords. *Table merging, SQL joins, Python pandas, data analysis, relational databases, data integrity, data manipulation, LEFT JOIN, RIGHT JOIN, performance optimization.*

Аннотация. *Объединение таблиц — это фундаментальная операция в управлении базами данных и анализе данных, при которой несколько таблиц объединяются в единую структуру. Использование стандартных функций в языках программирования, таких как SQL, Python или R, упрощает процесс эффективного объединения таблиц. В этой статье рассматриваются различные методы объединения таблиц с использованием стандартных функций, уделяя особое внимание объединениям SQL (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN) и использованию функций в Python (например, pandas merge()) и R. В статье обсуждается значение этих методов в реляционных* *базах данных и задачах манипулирования данными, подчеркивая важность целостности данных, производительности и эффективного выполнения запросов. Кроме того, в статье рассматриваются распространенные проблемы, такие как обработка отсутствующих данных, управление большими наборами данных и обеспечение точности в процессе слияния.*

Ключевые слова. *Объединение таблиц, объединения SQL, Python pandas, анализ данных, реляционные базы данных, целостность данных, манипулирование данными, LEFT JOIN, RIGHT JOIN, оптимизация производительности.*

In modern data management and analysis, combining data from different sources or tables is a crucial task. Merging tables allows for the integration of information, making it possible to derive insights, perform complex analyses, and maintain an organized data structure. In relational databases, data is typically stored across multiple tables, each representing a specific entity. However, real-world applications require combining these tables to gather complete datasets for reporting, analysis, or business decision-making.

The process of merging tables is often achieved using standard functions provided by programming languages and database management systems. These functions enable seamless integration of tables based on common attributes, such as primary keys or foreign keys. This article will explore various methods for merging tables using standard functions, focusing on SQL joins and programming functions in languages such as Python and R. We will also discuss key considerations such as data integrity, handling missing values, and optimizing the merge process for performance.

Table merging is the operation of combining two or more tables into a single table based on related columns. It is essential when working with relational databases, where different tables are designed to hold different aspects of data but share common attributes. Merging tables typically involves aligning data from these tables by matching key columns, which is often done using SQL JOIN operations or data manipulation libraries in programming languages like Python and R.

The primary goal of merging tables is to consolidate data to create a unified dataset. Some common types of table merges include:

- **Inner Join**. Combines rows from two tables where there is a match on the key columns.
- **Left Join**: Combines all rows from the left table with matching rows from the right table, and fills missing values with NULLs if there's no match.
- **Right Join**: Combines all rows from the right table with matching rows from the left table.
- **Full Join**: Combines all rows from both tables, filling in NULLs where there are no matching rows.
- **Cross Join**: Combines every row from the left table with every row from the right table, generating a Cartesian product.

1. Using SQL Joins

SQL provides a robust set of functions for merging tables. These functions are based on the concept of a **join**, which links rows from two or more tables based on shared keys or conditions. The primary types of joins include:

1.1. INNER JOIN

The INNER JOIN operation merges tables by selecting only the rows where there is a match on the join condition, typically based on common columns like IDs or keys. If a row in one table does not have a matching row in the other table, it is excluded from the result.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.department_id;
```

In this example, only employees who belong to a department are returned in the result.

1.2. LEFT JOIN (LEFT OUTER JOIN)

The LEFT JOIN retrieves all records from the left table and the matched records from the right table. If there is no match in the right table, the result will contain NULL values.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments
ON employees.department_id = departments.department_id;
```

This will return all employees, including those who are not assigned to a department, with NULL in the department column for those employees.

1.3. RIGHT JOIN (RIGHT OUTER JOIN)

The RIGHT JOIN works similarly to the LEFT JOIN but retrieves all records from the right table and the matching records from the left table.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.department_id;
```

This will return all departments, including those without any employees, with NULL in the employee name column.

1.4. FULL OUTER JOIN

The FULL OUTER JOIN combines the results of both the LEFT JOIN and RIGHT JOIN. It returns all rows from both tables, with NULLs in places where there is no match.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
FULL OUTER JOIN departments
ON employees.department_id = departments.department_id;
```

This will return all employees and all departments, even if there is no match between them.

1.5. CROSS JOIN

A CROSS JOIN combines every row from the left table with every row from the right table, resulting in a Cartesian product. It does not require any condition for merging tables.

Example:

```
SELECT employees.name, departments.department_name
FROM employees
CROSS JOIN departments;
```

This generates a result where every employee is paired with every department.

2. Using Python (Pandas Library)

In Python, one of the most widely used libraries for data manipulation is **pandas**. The merge() function in pandas is used to combine data from multiple DataFrames, similar to SQL JOIN operations. Pandas supports various types of joins, including inner, outer, left, and right joins.

2.1. Merging DataFrames in Pandas

The merge() function allows you to merge DataFrames based on common columns or indexes.

Example:

```
import pandas as pd
# Create DataFrames
employees = pd.DataFrame({
   'employee_id': [1, 2, 3],
   'name': ['John', 'Anna', 'Mike'],
   'department_id': [101, 102, 103]
})departments = pd.DataFrame({
   'department_id': [101, 102, 104],
   'department_name': ['HR', 'Engineering', 'Marketing']
})
```

# Merge DataFrames using 'department_id' as the key

merged_df = pd.merge(employees, departments, on='department_id', how='inner')

print(merged_df)

This example performs an **inner join** on the department_id column, similar to the SQL INNER JOIN.

2.2. Types of Joins in Pandas

The how parameter in the merge() function allows the user to specify the type of join:

- 'inner': Performs an inner join.
- 'left': Performs a left join.
- 'right': Performs a right join.
- 'outer': Performs a full outer join.

Example of a **left join**:

merged_df = pd.merge(employees, departments, on='department_id', how='left')

3. Using R (dplyr Package)

In R, the **dplyr** package provides functions such as left_join(), right_join(), inner_join(), and full_join() for merging data frames.

3.1. Merging Data Frames in R

Example using the dplyr package to merge two data frames:

```
library(dplyr)
# Create data frames
employees <- data.frame(employee_id = c(1, 2, 3),
                        name = c('John', 'Anna', 'Mike'),
                        department_id = c(101, 102, 103))
departments <- data.frame(department_id = c(101, 102, 104),
                          department_name = c('HR', 'Engineering', 'Marketing'))
# Merge data frames using left_join
merged_df <- left_join(employees, departments, by = "department_id")
```

```
print(merged_df)
```

This performs a **left join**, combining the data frames on the department_id column.

Challenges in Table Merging

1. **Handling Missing Data**: When merging tables, there may be rows with no matching entries. Deciding how to handle missing data (i.e., whether to fill it with NULLs or exclude the rows) is an important consideration.
2. **Performance Issues**: Merging large datasets can be computationally expensive. Optimizing queries and using efficient algorithms is crucial when working with big data.
3. **Ensuring Data Integrity**: Merging tables requires careful handling of keys to ensure data integrity. Duplication, misalignment of keys, and loss of data can occur if joins are not implemented properly.

Merging tables is a vital aspect of database management, data analysis, and business intelligence. Whether using SQL joins or the merge() function in Python or R, standard functions make it easier to combine data from multiple sources effectively. By understanding and applying the appropriate type of join, developers and data analysts can create unified datasets that enable comprehensive analysis, better decision-making, and more efficient data management.

REFERENCES:

1. Zarif o‘g‘li K. F. CREATING A TEST FOR SCHOOL EDUCATIONAL PROCESSES IN THE ISPRING SUITE PROGRAM //BOSHLANG ‘ICH SINFLARDA O ‘ZLASHTIRMOVCHILIKNI. – C. 84.
2. O‘G‘Li K. F. Z. CREATING A TEST FOR SCHOOL EDUCATIONAL PROCESSES IN THE ISPRING SUITE PROGRAM //Yosh mutaxassislar. – 2023. – T. 1. – №. 8. – C. 84-87.
3. Kaynarov F. Z. THEORETICAL FOUNDATIONS FOR THE CREATION OF ELECTRONIC TEXTBOOKS FOR DISTANCE EDUCATION //Экономика и социум. – 2024. – №. 2-2 (117). – C. 169-175.

4. Kaynarov F. APPLICATION OF MODERN INFORMATION TECHNOLOGIES IN MEDICINE //International Scientific and Practical Conference on Algorithms and Current Problems of Programming. – 2023.

5. Кайнаров Ф. З. ИННОВАЦИОННЫЕ МЕТОДЫ ПРЕПОДАВАНИЯ ПРИКЛАДНОЙ МАТЕМАТИКИ //Экономика и социум. – 2023. – №. 1-2 (104). – С. 619-622.

6. Daminova B. ACTIVATION OF COGNITIVE ACTIVITY AMONG STUDENTS IN TEACHING COMPUTER SCIENCE //CENTRAL ASIAN JOURNAL OF EDUCATION AND COMPUTER SCIENCES (CAJECS). – 2023. – Т. 2. – №. 1. – С. 68-71.

7. Esanovna D. B. Modern Teaching Aids and Technical Equipment in Modern Educational Institutions //International Journal of Innovative Analyses and Emerging Technology. – Т. 2. – №. 6.

8. Рахимов Н., Эсановна Б., Примкулов О. Ахборот тизимларида мантиқий хулосалаш самарадорлигини ошириш ёндашуви //International Scientific and Practical Conference on Algorithms and Current Problems of Programming. – 2023

9. Даминова Б. Э. СОДЕРЖАНИЕ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ И ТЕНДЕНЦИИ ЕГО ИЗМЕНЕНИЯ ПОД ВЛИЯНИЕМ НОВЫХ СОЦИАЛЬНО-ЭКОНОМИЧЕСКИХ УСЛОВИЙ //Yosh mutaxassislar. – 2023. – Т. 1. – №. 8. – С. 72-77.

10. Кувандиков Ж., Даминова Б., Хафизадинов У. АВТОМАТЛАШТИРИЛГАН ЭЛЕКТРОН ТАЪЛИМ ТИЗИМИНИ ЛОЙИҲАЛАШДА ЎҚУВ ЖАРАЁНИНИ МОДЕЛЛАШТИРИШ //International Scientific and Practical Conference on Algorithms and Current Problems of Programming. – 2023.

11. Даминова Б. Э. Сравнительный анализ состояния организации многоуровневых образовательных процессов //Экономика и социум. – 2023. – №. 1-2 (104). – С. 611-614.

12. Daminova B. Algorithm of education quality assessment system in secondary special education institution (on the example of guzor industrial technical college)

//International Scientific and Practical Conference on Algorithms and Current Problems of Programming. – 2023.

13. Daminova B. FORMATION OF THE MANAGEMENT STRUCTURE OF EDUCATIONAL PROCESSES IN THE HIGHER EDUCATION SYSTEM //Science and innovation. – 2023. – Т. 2. – №. A6. – С. 317-325.

14. Даминова Б. Э., Якубов М. С. Развития познавательной и творческой активности слущателей //Международная конференция" Актуальные проблемы развития инфокоммуникаций и информационного общества. – 2012. – С. 26-27.06.

15. Якубов М., Даминова Б., Юсупова С. Формирование и повышение качества образования с помощью образовательных информационных технологий //International Scientific and Practical Conference on Algorithms and Current Problems of Programming.-2023.

16. Даминова Б. Э. и др. ОБРАБОТКА ВИДЕОМАТЕРИАЛОВ ПРИ РАЗРАБОТКЕ ОБРАЗОВАТЕЛЬНЫХ РЕСУРСОВ //Экономика и социум. – 2024. – №. 2-2. – С. 117.