Name: Abenes, Enrico O.
Subject & Section: CC5 -2I
Date: October 30, 2024

ACTIVITY No. 8: RESEARCH

A. Normalization

Normalization in database management is a process aimed at optimizing data storage in tables by reducing redundancy and logically organizing information. This technique minimizes repetitive data by structuring tables so that each piece of information appears only where necessary. Imagine a bookshelf where each book is placed in a specific spot instead of appearing on multiple shelves. In a similar way, normalization ensures that data can be easily found, maintained, and updated without inconsistency across the database. Normalization works through different stages, or "normal forms," each refining the organization of data within tables for improved efficiency and scalability.

B. Benefits of Normalization:

1. Less Data Duplication. By eliminating repetitive values across tables, the database saves storage space. This streamlined structure prevents the need to store the same information multiple times, making the system more efficient.
2. Improved Data Integrity. Changes to data only need to be made in one location, reducing the risk of inconsistent or outdated information in multiple places. This ensures that updates are synchronized automatically across the database.
3. Efficient Queries. With a well-structured database, queries become faster and more efficient. Normalization enables databases to retrieve data quickly by minimizing the need to search through redundant information, which makes applications that rely on the database run faster and more smoothly.
4. Ultimately, the goal is to create an organized, scalable, and easy-to-maintain database that grows effectively alongside the expanding data. A normalized database structure ensures that it remains manageable over time, supporting efficient operations.

C. Normal Forms

- **First Normal Form (1NF)**:
  - For a table to be in 1NF, each column must contain atomic values, meaning there are no lists, arrays, or nested sets within a single cell. This ensures that every cell holds only a single, indivisible piece of data.
  - **Example**: Take a “Customer Orders” table with a column labeled “Products” where one cell might list several items as “Shoes, Shirt, Hat.” This setup is not in 1NF because it stores multiple items in one cell. To meet 1NF, each product would need to be separated into individual rows linked to a unique order identifier, ensuring that each cell contains only one product.

- **Second Normal Form (2NF)**:
  - A table is in 2NF if it is already in 1NF, and each non-key attribute depends on the entire primary key, not just part of it. This is particularly important in tables with composite primary keys (where the primary key consists of more than one column).
  - **Example**: Consider a “Student Courses” table with a composite key made up of StudentID and CourseID, along with a column for “StudentName.” If “StudentName” depends only on StudentID (and not on the combination of StudentID and CourseID), it breaks 2NF because “StudentName” is only partially dependent on the primary key. To resolve this, “StudentName” should be moved to a separate “Students” table where it can be linked by StudentID.

- **Third Normal Form (3NF)**:
  - To be in 3NF, a table must already satisfy 2NF, and all non-key columns must depend directly on the primary key. This eliminates transitive dependencies, where a non-key column depends on another non-key column instead of the primary key.
  - **Example**: Imagine an “Employees” table with columns for EmployeeID, DepartmentID, and

DepartmentLocation. Here, if DepartmentLocation relies on DepartmentID rather than directly on EmployeeID, this creates a transitive dependency that violates 3NF. Moving DepartmentLocation to a separate "Departments" table linked by DepartmentID would correct this structure.

- **Fourth Normal Form (4NF)**:
  - A table is in 4NF if it is in 3NF and contains no multi-valued dependencies. This means that each attribute pair within a row should have only one corresponding value in other columns.
  - **Example**: Consider a "Courses" table where each course might be associated with multiple instructors and multiple students. Such a table would create multi-valued dependencies, leading to redundancies. To comply with 4NF, separate tables for "CourseInstructors" (CourseID, InstructorID) and "CourseStudents" (CourseID, StudentID) should be created, removing these dependencies.

- **Fifth Normal Form (5NF)**:
  - A table is in 5NF if it is already in 4NF and does not contain any join dependencies, meaning the table cannot be divided into smaller tables without introducing redundancy or data loss. This level of normalization is particularly relevant for complex databases where maintaining data integrity in joins is critical.
  - **Example**: Suppose a "Contracts" table includes CustomerID, ProductID, and SupplierID, where each of these fields interacts with the others. To avoid redundancy and achieve 5NF, this table might be separated into smaller, more specific tables like "CustomerProducts" and "ProductSuppliers." This separation ensures the data can be recombined through joins without introducing unnecessary duplication, achieving an optimized, redundancy-free structure.

D. Summary

Normalization is a core database management principle focused on enhancing database structure and functionality by organizing data to reduce redundancy and improve data integrity. This involves a progression through normal forms (1NF to 5NF), each adding stricter requirements to refine the database structure. In 1NF, tables must contain atomic values with no repeating groups. In 2NF, tables eliminate partial dependencies on composite keys. 3NF removes transitive dependencies, ensuring all non-key columns depend solely on the primary key. 4NF addresses multi-valued dependencies, while 5NF resolves complex join dependencies, resulting in an organized, efficient, and easy-to-maintain database.

References

Amin, M., Gordon, R. W., Dey, P., & Sinha, B. (2019, October). Teaching relational database normalization in an innovative way. *Journal of Computing Sciences in Colleges*, 48-56. Retrieved from https://core.ac.uk/download/pdf/289189314.pdf#page=49

Demba, M. (2013, June). Algorithm for relational database normalization up to 3NF. *International Journal of Database Management Systems*, 39. Retrieved from https://d1wqtxts1xzle7.cloudfront.net/37950350/5313ijdms03-libre.pdf?1434779031=&response-content-disposition=inline%3B+filename%3DALGORITHM_FOR_RELATIONAL_DATABASE_NORMAL.pdf&Expires=1730218061&Signature=dJsK7WaGS-ribWrhgacdv-63bn8ieuKBmPYj2W9vgibw-3ypvL

Demba, M. (2013, April). An algorithmic approach to database normalization. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, 197-205. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9551fb9485381c799da9a3a3bc37a97f2c5fbe4c

Duggal, K., Srivastav, A., & Kaur, S. (2014, january). Gamified approach to database normalization. *nternational Journal of Computer Applications*. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e4e039b88bd9d63cbd63e21505762d3966bd462f

Eessaar, E. (2016, January). The Database Normalization Theory and the Theory of Normalized Systems: Finding a Common Ground. *Baltic Journal of Modern Computing*. Retrieved from https://www.researchgate.net/profile/Erki-Eessaar/publication/297731569_The_Database_Normalization_Theory_and_the_Theory_of_Normalized_Systems_Finding_a_Common_Ground/links/56e18d9508ae40dc0abf50a1/The-Database-Normalization-Theory-and-the-Theory-of-Norma

Lee, B. S. (1995, September). Normalization in OODB design. *ACM Sigmod Record*, 23-27. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2a9cdfca008f227ec547b89ad7d3cd2b8ed341a1

Pinto, Y. (2009). A framework for systematic database de-normalization. *Global Journal of Computer Science and Technology*, 44-151. Retrieved from http://irgu.unigoa.ac.in/drs/bitstream/handle/unigoa/2295/Global_J_Comput_Sci_Technol_9_44.pdf?sequence=1

Soler, J., Boada, I., Prados, F., & Poch, J. (2006, January). A web-based Problem-Solving Environment for database Normalization. *Proc. 8th Int. Symp. on Computers in Education SIIE*. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=96c83aa794d2a4f96885cddabd7c5f81b38cccc9