Case Study of the Failures of Relational Database Management Systems

Team Number #1

Kris Beckman, Bachelor of Science Information Systems (BSIS)

Jennifer Jamiel, Bachelor of Science Information Systems (BSIS)

Thaddeus Thomas, Bachelor of Science Information Systems (BSIS)

Deepti Venkatesh, Bachelor of Science Applied Computing (BSAPC)   
Professor Donna Hoffman, IS 456 Database Systems Management,

School of Technology and Computing, City University of Seattle

[kristina060585@cityuniversity.edu](mailto:kristina060585@cityuniversity.edu), [jamieljennifer@cityuniversity.edu](mailto:jamieljennifer@cityuniversity.edu), [thomasthaddeus@cityuniversity.edu](mailto:Thomasthaddeus@cityuniversity.edu), [venkateshdeepti@cityuniversity.edu](mailto:venkateshdeepti@cityuniversity.edu)

**Abstract**

This paper will contain a case study of the failures of RDBMS (relational database management systems). Specifically, it will focus on what processes are restrictive to ensuring database integrity. Furthermore, it will show the methods that forcing ACID (Atomic Consistency Isolation Durability) operations restrict the usability of RDBMS. Lastly, this case study will show how scaling vertically wastes enormous space and processing power when compounded across thousands of databases.

**Keywords:** RDBMS, ACID, CRUD (Create Read Update Delete), Data Management, SQLite

**1.** **DATABASE DESIGN SUMMARY**

In approaching the database design for our company employee database, we chose a relational data model due to the highly structured nature of employee data. In addition, high ACID guarantees are needed because of the sensitivity of data to avoid corruption and out-of-sync data. Finally, the relational model strikes the right balance because we will not need to scale thousands to millions of reads and writes, where NoSQL models shine.

For this case study, the database is designed with three tables, initially: employees, jobs, and departments. The way they relate to each other can be seen in figure 1. We have also included stored procedures for manipulating data in a controlled, governed fashion. To work around the query performance bottlenecks of JOINS, we have created query-optimized VIEWS for many of our core reads. This simple model still gives elegant flexibility as employees can be managers with self-referential keys without introducing the fourth table. This model can be extended in the future if needed without a complete re-design of the database. For example, we could add an employee-type table with values such as individual contributor, manager, executive, board member, and foreign key into the employee table.

**Related Work**

Numerous studies have been written about the differences between relational databases and their NoSQL counterparts (Fowler, 2015). What this case study endeavors to support is where they both excel and where they fail. The database built for this project will be used for examples to show how the data can be oriented using multiple systems.

**2.** **ENTITY RELATIONSHIP DIAGRAM**

The following is the ERD (Entity Relationship Diagram) explaining the data model & its built-in relations to form the core of a company employee database. Three tables populate the database.

**Employee**

The employee table of the database is where all the critical information is stored for employees. [EmployeeID] is the primary key and [JobID] is the foreign key.

**Job**

This table contains Job details and salary compensation levels. [JobID] is the primary key in this table. [DeptID] is the foreign key. [CommPerc] is shorthand for a commission percentage.

**Department**

The Department table has the primary key [DeptID] also the foreign key on the Job table.

Graphical user interface, application

Description automatically generated

Figure 1-Entity Relationship Diagram

**3.** **REFERENCES**

Coronel, C., & Morris, S. (2018). *Database Systems: Design, implementation, & management* (13th ed.). Cengage Learning.

Fowler, A. (2015). *NoSQL For Dummies*. John Wiley and Sons.

Petkovic, D. (2020). *Microsoft SQL Server 2019: A beginner's guide, seventh edition* (7th ed.). McGraw Hill.

Zhang, W. (2012). *A Suite of Case Studies in Relational Database Design* (Doctoral dissertation).

**Appendices**

Tasks

**#1. Write a query to display the names (first\_name, last\_name) using alias name "First Name", "Last Name"**

SELECT

first\_name AS 'First Name',

last\_name AS 'Last Name'

FROM employee;

**#2. Write a query to get unique department ID from employee table.**

SELECT DISTINCT dept\_id

FROM employee;

**#3. Write a query to get all employee details from the employee table order by first name, descending.**

SELECT \*

FROM employee

ORDER BY first\_name DESC;

**#4. Write a query to get the names (first\_name, last\_name), salary, PF of all the employees (PF is calculated as 12% of salary).**

SELECT

(first\_name + last\_name) AS Names,

salary,

PF

FROM employee

WHERE PF = (salary \* .12)

ORDER BY salary ASC;

**#5. Write a query to get the employee ID, names (first\_name, last\_name), salary in ascending order of salary.**

SELECT

emp\_id,

(first\_name + last\_name) AS Names,

salary

FROM employee

GROUP BY emp\_id

ORDER BY salary ASC;

**#6. Write a query to get the total salaries payable to employees.**

SELECT SUM(salary)

FROM employee;

**#7. Write a query to get the maximum and minimum salary from the employees' table.**

SELECT

MAX(salary) AS 'Maximum',

MIN(salary) AS 'Minimum'

FROM employee;

**#8 Write a query to get the average salary and number of employees in the employees' table**

SELECT

ROUND(AVG(salary)) AS avg\_salary,

COUNT(DISTINCT emp\_id) AS 'Total Employees'

FROM employee;