JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 13 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v13i2.6354

IMPLEMENTASI BASIS DATA TERSTRUKTUR DENGAN PENCEGAHAN SQL INJECTION PADA SISTEM MANAJEMEN PENJUALAN

Galih Rafianto[1*], Apriade Voutama[2]

[1,2]Universitas Singaperbangsa Karawang, Jl. HS. Ronggo Waluyo, Puserjaya, Telukjambe Timur, Karawang, Jawa Barat, 41361.

Received: 2 Maret 2025
Accepted: 27 Maret 2025
Published: 14 April 2025

Keywords:
Normalisasi Basis Data;
SQL Injection;
Stored Procedure;
Parameterized Queries.

Corespondent Email:
galihrafianto179@gmail.com

Abstrak. Dalam era digital, basis data memainkan peran penting dalam efisiensi operasional perusahaan, terutama dalam pengelolaan informasi pelanggan dan transaksi. Namun, banyak sistem mengalami permasalahan akibat struktur basis data yang tidak terorganisir dengan baik, menyebabkan redundansi data, performa yang buruk, serta potensi ancaman keamanan seperti SQL *Injection*. Penelitian ini bertujuan untuk merancang basis data yang lebih terstruktur dengan menerapkan normalisasi hingga tingkat *Third Normal Form* (3NF) serta meningkatkan keamanan menggunakan *Stored Procedure* dengan *parameterized queries*. Metode yang diterapkan dalam penelitian ini mengacu pada *Database System Development Lifecycle* (DSDLC), yang meliputi tahap pengumpulan kebutuhan, perancangan, implementasi, serta pengujian sistem. Hasil yang diperoleh menunjukkan bahwa proses normalisasi secara efektif mengurangi redundansi data dan meningkatkan efisiensi sistem. Selain itu, penerapan *Stored Procedure* dengan *parameterized queries* terbukti mampu memberikan perlindungan terhadap serangan *SQL Injection*, memastikan keamanan serta integritas data tetap terjaga. Pengujian keamanan yang dilakukan membuktikan bahwa sistem tetap aman dari *input* berbahaya, memastikan integritas data tetap terjaga. Penelitian ini berkontribusi dalam meningkatkan efisiensi dan keamanan basis data, serta membuka peluang untuk pengembangan lebih lanjut, seperti penerapan validasi *input* di tingkat aplikasi dan sistem pemantauan aktivitas basis data guna meningkatkan ketahanan terhadap ancaman siber di masa depan.

Abstract. *In the digital era, databases play a crucial role in enhancing the operational efficiency of companies, particularly in managing customer information and transactions. However, many systems face issues due to poorly structured databases, leading to data redundancy, poor performance, and security threats such as SQL Injection. This study aims to design a more structured database by applying normalization up to the Third Normal Form (3NF) and enhancing security through the implementation of Stored Procedures with parameterized queries. The method applied in this study follows the Database System Development Lifecycle (DSDLC), which includes the stages of requirements gathering, design, implementation, and system testing. The results indicate that the normalization process effectively reduces data redundancy and enhances system efficiency. Additionally, the implementation of Stored Procedures with parameterized queries has been proven to provide strong protection against SQL Injection attacks, ensuring both data security and integrity.. Security testing confirms that the system*

remains protected from malicious input, ensuring data integrity. This research contributes to optimizing database performance and security while opening opportunities for further developments, such as implementing input validation at the application level and monitoring database activity to enhance resilience against future cybersecurity threats.

1. PENDAHULUAN

Kegiatan pada berbagai bidang saat ini ditunjang oleh teknologi yang berkembang dengan sangat pesat [1][2]. Perkembangan teknologi yang begitu cepat mengharuskan perusahaan untuk mampu mengalokasikan sumber daya mereka dengan cara yang efektif dan efisien [3]. Dalam hal ini, pengelolaan data dalam skala besar menjadi sangat penting bagi perusahaan untuk menjaga efisiensi operasional dan meningkatkan pelayanan kepada pelanggan. Hal ini membuat penggunaan basis data dibutuhkan demi keberlangsungan operasional dan proses bisnis perusahaan. Basis data berperan sebagai wadah penyimpanan yang mampu mengelola serta menyusun data dalam format yang terstruktur, sehingga mempermudah akses informasi dengan cepat dan efisien [4]. Perusahaan bergantung pada sistem basis data yang efisien untuk mengelola informasi pelanggan, transaksi, dan analisis bisnis yang mendukung pengambilan keputusan strategis.

Namun, perusahaan sering kali menghadapi masalah di mana basis data mereka tidak ternormalisasi dengan baik, yang mengakibatkan data menjadi redundan, tidak konsisten, dan sulit dikelola [5]. Struktur basis data yang tidak optimal ini dapat memengaruhi performa aplikasi, menghambat analisis data, serta meningkatkan risiko kesalahan operasional [6].

Untuk mengatasi masalah tersebut, normalisasi basis data menjadi pendekatan yang digunakan untuk menyusun kembali struktur data agar lebih efisien dan terorganisir. Normalisasi bertujuan untuk menghilangkan redundansi data, memastikan konsistensi informasi, serta meningkatkan kecepatan dalam proses *query database*. Dengan penerapan prinsip normalisasi, data dapat diorganisir lebih baik, mengurangi duplikasi, dan memastikan integritas data terjaga [7].

Selain itu, keamanan basis data turut menjadi perhatian utama dalam pengembangan aplikasi berbasis data. Serangan basis data yang kerap dialami adalah SQL *Injection*. Serangan ini memanfaatkan kelemahan dalam pengkodean aplikasi, di mana peretas dapat menyisipkan perintah SQL, seperti melalui *form login* [8]. Ini memungkinkan pihak yang tidak berwenang untuk dapat mengakses dan memanipulasi basis data aplikasi, baik untuk mencuri informasi maupun mengubah atau menghapus data yang valid yang terdapat dalam aplikasi [9]. Aplikasi yang tidak memiliki validasi *input* dan langkah keamanan yang cukup saat berkomunikasi dengan basis data *Structured Query Language* cenderung rentan terhadap serangan SQL *Injection* [10].

Salah satu pendekatan yang dapat diterapkan untuk meningkatkan keamanan basis data adalah dengan memanfaatkan *Stored Procedures*. *Stored Procedure* adalah gabungan beberapa pernyataan SQL yang sudah dikompilasi ke dalam satu rencana eksekusi, yang kemudian dibuat dan disimpan ke dalam *database* yang dapat digunakan kembali sesuai kebutuhan [11]. *Stored Procedures* memungkinkan eksekusi *query* dilakukan secara lebih aman dengan mengurangi kemungkinan manipulasi perintah SQL dari *input* pengguna.

Penelitian ini bertujuan untuk merancang basis data yang terstruktur tanpa mengesampingkan aspek keamanannya. Dalam penelitian ini, konsep normalisasi diterapkan untuk membangun struktur basis data yang lebih optimal, sementara *Stored Procedure* dengan *Parameterized Query* digunakan sebagai langkah perlindungan terhadap serangan SQL *Injection*. Dengan menerapkan metodologi DSDLC (*Database System Development Lifecycle*), proses perancangannya akan lebih terstruktur.

Penelitian terdahulu pada sebuah jurnal berjudul "Perancangan Basis Data Menggunakan Normalisasi Tabel pada Perusahaan Dagang Barokah Abadi", penelitian ini bertujuan untuk merancang basis data dengan struktur yang baik [12]. Penelitian kedua yang berjudul "*A Method to Prevent* SQL

Injection Attack using an Improved Parameterized Stored Procedure", penelitian ini bertujuan untuk mengusulkan metode baru untuk mencegah SQL *Injection* dengan menggunakan *Stored Procedure*. Hasilnya menunjukkan bahwa penggunaan *Stored Procedure* efektif dalam mencegah serangan SQL *Injection* [13]. Pada artikel ketiga yang berjudul "SQL *Injection Attack: Quick View*" membahas bahwa validasi *input* dan *parameterized queries* adalah cara paling dasar namun efektif dalam mencegah serangan SQL *Injection* [10].

2. TINJAUAN PUSTAKA

2.1. Basis Data

Basis data adalah sekumpulan data yang terintegrasi dan dapat didesain sehingga mendapatkan data yang dibutuhkan oleh perusahaan atau organisasi [14]. Dalam perancangan basis data, dibutuhkan metode normalisasi demi memperoleh bentuk basis data yang optimal dan terstruktur.

2.2. Normalisasi

Normalisasi basis data adalah sebuah proses bertahap yang dilakukan dalam perancangan basis data untuk mencegah terjadinya inkonsistensi dan redundansi data [15]. Proses ini memiliki beberapa bentuk tahapan yaitu 1NF, 2NF, 3NF, BCNF, dan 5NF. Namun pada kasus yang umum terjadi, tahapan normalisasi hanya mencapai bentuk 3NF. Setelah melakukan normalisasi, diperlukan sebuah rancangan dalam bentuk diagram yang disebut ERD.

2.3. *Entity-Relationship Diagram*

Entity-Relationship Diagram (ERD) merupakan representasi grafis yang digunakan untuk menggambarkan hubungan antar data dalam proses perancangan basis data [16]. ERD berfungsi untuk menggambarkan struktur data serta hubungan antar entitas dalam basis data, sehingga mempermudah proses perancangan dan pengembangannya dalam sistem perangkat lunak [17]. Selain itu, ERD berfungsi untuk mendokumentasikan dan memberikan pemahaman tentang basis data yang sudah ada, serta berperan sebagai alat bantu dalam merancang ulang suatu proses bisnis [18].

2.4. SQL *Injection*

Serangan SQL *Injection* adalah jenis serangan yang menargetkan aplikasi berbasis web yang terhubung dengan basis data, di mana kode berbahaya disisipkan, diproses, dan dieksekusi [19].

2.5. *Stored Procedure*

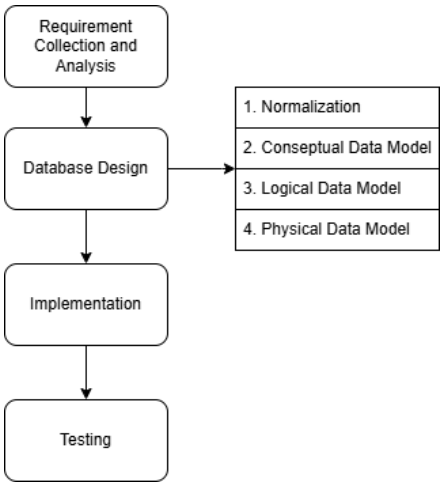
Stored procedure merupakan kumpulan instruksi yang disimpan di dalam basis data. Biasanya, *stored procedure* ditulis menggunakan SQL. Eksekusi *stored procedure* dilakukan berdasarkan perintah yang diberikan oleh pengguna dan dapat menerima berbagai parameter untuk *input* dan *output* [20].

2.6. *Parameterized Query*

Parameterized Query adalah sebuah *query* yang menggunakan sebuah parameter atau penanda, yang mana parameter ini akan diberikan sebuah nilai pada saat *query* dieksekusi. Teknik ini mencegah eksekusi perintah SQL yang berbahaya dengan memisahkan logika SQL dari data yang diberikan oleh pengguna. Melalui penggunaan *parameterized query*, setiap *input* yang dimasukkan dianggap sebagai nilai parameter, bukan bagian dari perintah SQL, sehingga menghilangkan potensi terjadinya injeksi kode[21].

3. METODE PENELITIAN

Pada metode penelitian ini mengacu pada pendekatan DSDLC (*Database System Development Lifecycle*) sebagai kerangka kerja utama dalam pengembangan basis data, dengan susunan tahapan sebagai berikut:



Gambar 1. Kerangka Penelitian

Berikut ini adalah tahapan penelitian dengan metode DSDLC berdasarkan gambar 1 sebagai berikut:

3.1. *Requirement Collection and Analysis*

Pada tahap ini pengumpulan dan analisis data untuk mengidentifikasi kebutuhan sistem basis data yang akan dirancang. Proses ini mencakup studi terhadap data transaksi, struktur tabel awal, serta analisis kebutuhan pengguna dalam pengelolaan informasi yang lebih efisien dan aman. Identifikasi kebutuhan ini menjadi dasar dalam menentukan strategi normalisasi dan teknik pengamanan *database* yang tepat guna.

3.2. *Database Design*

Tahap ini dilakukan pembuatan desain basis data yang dibutuhkan oleh sistem basis data untuk menunjang kebutuhan perusahaan. Penelitian ini mencakup empat tahap utama dalam perancangan basis data, yaitu Normalisasi, *Conceptual Data Model*, *Logical Data Model*, dan *Physical Data Model*. Normalisasi dilakukan hingga tingkat *Third Normal Form* (3NF) untuk menghilangkan redundansi data dan menghasilkan data yang terintegrasi dengan baik. Selanjutnya, ERD dirancang dalam tiga model untuk memastikan kejelasan hubungan antar entitas sebelum implementasi dilakukan. Hasil dari desain basis data menjadi panduan pada tahap implementasi.

3.3. Implementasi

Tahap implementasi basis data dilakukan dengan menggunakan SQL Server, di mana struktur tabel yang telah dirancang diterapkan ke dalam sistem. Pada tahap ini juga dilakukan pengembangan *Stored Procedures* serta *parameterized query* untuk meningkatkan keamanan terhadap serangan SQL *Injection*. Implementasi ini dilakukan secara sistematis dengan memastikan bahwa setiap perintah SQL yang dieksekusi memenuhi standar keamanan yang ditetapkan.

3.4. Pengujian

Langkah terakhir dalam penelitian ini adalah melakukan pengujian dan evaluasi terhadap sistem basis data yang telah diimplementasikan. Pengujian dilakukan dengan aspek utama, yaitu tingkat keamanan terhadap serangan SQL *Injection*.

4. HASIL DAN PEMBAHASAN

Bagian ini membahas hasil implementasi perancangan basis data yang telah dilakukan serta evaluasi terhadap efisiensi dan keamanannya. Berikut adalah pembahasan lebih lanjut mengenai hasil implementasi yang telah dilakukan.

4.1. Implementasi Normalisasi

Berikut hasil implementasi normalisasi hingga tingkat 3NF untuk mendukung perancangan basis data terstruktur.

Hasil implementasi menunjukkan bahwa sistem basis data yang telah didesain dengan normalisasi hingga tingkat 3NF berhasil meningkatkan efisiensi pengelolaan data. Tabel-tabel yang sebelumnya memiliki banyak redundansi telah dikonversi menjadi lebih terstruktur dan efisien dalam penyimpanan serta pemrosesan data.

4.1.1. First Normal Form (1NF)

Pada tahap *First Normal Form* (1NF), setiap kolom harus berisi nilai atomik tanpa duplikasi. Oleh karena itu, kolom *PhoneNumber* dipisahkan agar setiap pelanggan hanya memiliki satu nomor telepon, *OrderID* diatur agar merepresentasikan satu transaksi per baris, serta *PaymentMethod* diubah agar hanya menyimpan satu metode pembayaran per transaksi.

FirstName	LastName	Address	City	State

ZipCode	PhoneNumber	Email	OrderID	OrderDate

ProductID	ProductName	Category	ProductDescription	PricePerUnit

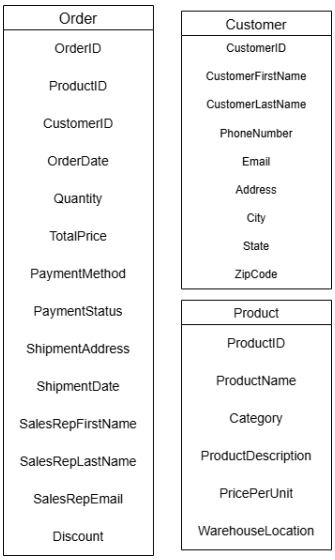
TotalPrice	PaymentMethod	PaymentStatus	ShipmentAddress	ShipmentDate

CustomerID	Discount	SalesRepFirstName	SalesRepLastName	SalesRepEmail	WarehouseLocatior

Gambar 2. Hasil Normalisasi pada *First Normal Form*

4.1.2. Second Normal Form (2NF)

Pada tahap *Second Normal Form* (2NF), dilakukan pemisahan atribut yang memiliki ketergantungan parsial terhadap *primary key* ke dalam tabel yang lebih spesifik. Tabel *Customer* menyimpan informasi pelanggan dengan *CustomerID* sebagai *primary key*, sementara tabel *Product* memiliki *ProductID* sebagai *primary key* dengan atribut seperti *ProductName*, *Category*, *PricePerUnit*, dan *WarehouseLocation*. Tabel *Order* tetap menyimpan *OrderID* sebagai *primary key*, dengan *ProductID* dan *CustomerID* sebagai *foreign keys* untuk menghubungkan pemesanan dengan produk serta pelanggan yang bersangkutan.



Gambar 3. Hasil Normalisasi pada *Second Normal Form*

4.1.3. Third Normal Form (3NF)

Pada tahap *Third Normal Form* (3NF), dilakukan eliminasi ketergantungan transitif dengan memisahkan informasi yang tidak secara langsung bergantung pada *primary key*. Tabel *Location* dibuat untuk menyimpan *ZipCode* sebagai *primary key* dengan atribut *City* dan *State*, yang sebelumnya berada dalam tabel *Customer*. Tabel *Category* diperkenalkan untuk menyimpan *CategoryID* sebagai *primary key*, menggantikan penyimpanan langsung dalam tabel *Product*, sementara tabel *SalesRep* dibuat untuk menampung informasi *sales representative* dengan *SalesRepID* sebagai *primary key* yang terhubung ke tabel *Order* melalui *foreign key*.

Guna mengelola pengiriman dan gudang secara lebih efisien, tabel *Shipping* diciptakan dengan *ShipmentID* sebagai *primary key*, serta *OrderID* dan *WarehouseID* sebagai *foreign keys* untuk menghubungkannya dengan tabel *Order* dan *Warehouse*. Tabel *Transaction* dibentuk untuk mencatat transaksi pembayaran secara terpisah, dengan *TransactionID* sebagai *primary key* serta *OrderID* sebagai *foreign key*, yang menghubungkannya dengan setiap pesanan yang dilakukan.

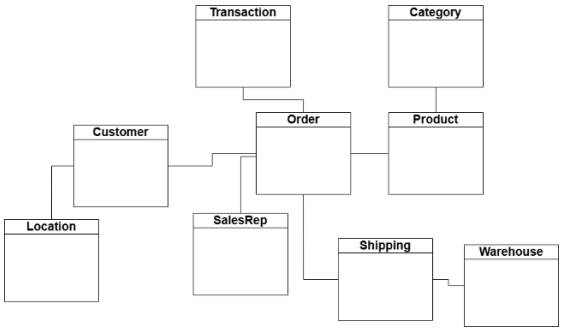
Gambar 4. Hasil Normalisasi pada *Third Normal Form*

4.2. Desain ERD

Berikut adalah beberapa model desain basis data yang diterapkan dalam perancangan sistem ini guna membangun struktur basis data yang lebih terorganisir dan efisien.

4.2.1. Conceptual Data Model

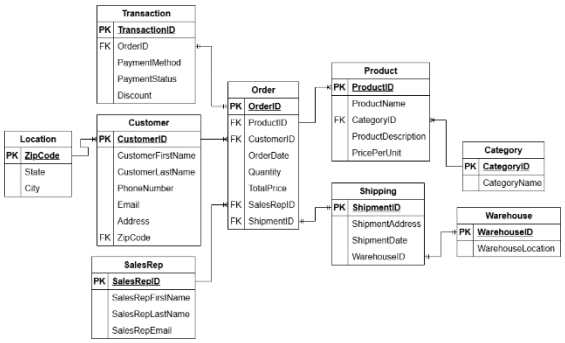
Conceptual Data Model mendeskripsikan hubungan antar entitas secara abstrak tanpa mempertimbangkan detail atribut dan tipe data. Model ini berfokus pada representasi utama dari data yang dibutuhkan oleh sistem, sehingga lebih mudah dipahami oleh pemangku kepentingan non-teknis. Pada sistem manajemen penjualan ini, entitas utama yang direpresentasikan dalam C*onceptual Data Model* meliputi *Customer*, *Product*, *Order*, *Payment*, dan *Shipping*.



Gambar 5. *Conceptual Data Model*

4.2.2. Logical Data Model

Logical Data Model (LDM) merupakan pengembangan dari *Conceptual Data Model* dengan menambahkan detail atribut, *primary key* (PK), dan *foreign key* (FK) untuk mendefinisikan relasi antar tabel secara lebih sistematis dan terstruktur.

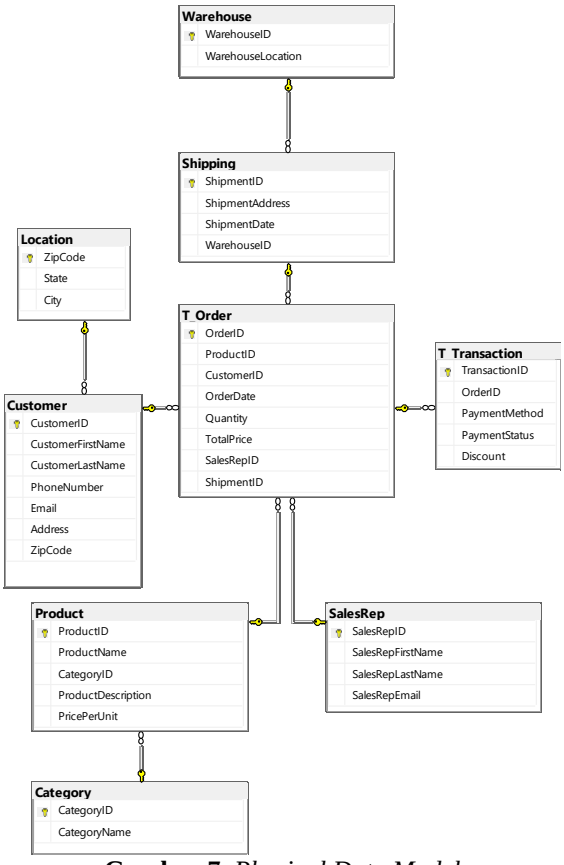


Gambar 6. *Logical Data Model*

Dalam *Logical Data Model* untuk sistem ini, tabel *Customer* menyimpan informasi pelanggan dengan atribut utama *CustomerID* sebagai *primary key*, serta *CustomerName*, *Email*, dan *PhoneNumber*. Tabel *Product* berisi data produk dengan *ProductID* sebagai *primary key*, serta *ProductName*, *Category*, dan *Price*. Tabel *Order* mencatat transaksi dengan *OrderID* sebagai *primary key*, *CustomerID* sebagai *foreign key*, serta *OrderDate* dan *TotalAmount* untuk mencatat waktu dan total pembelian. Untuk menghubungkan pesanan dengan produk, tabel *OrderDetails* digunakan dengan *OrderDetailID* sebagai *primary key*, serta *OrderID* dan *ProductID* sebagai *foreign key*, yang mencatat jumlah produk dalam setiap pesanan melalui atribut *Quantity*. Sementara itu, tabel *Payment* mencatat transaksi pembayaran dengan *PaymentID* sebagai *primary key*, *OrderID* sebagai *foreign key* yang menghubungkan dengan pesanan terkait, serta *PaymentMethod* dan *PaymentStatus* untuk mencatat metode dan status pembayaran pelanggan.

4.2.3. Physical Data Model

Physical Data Model (PDM) merepresentasikan implementasi nyata dari *Logical Data Model* dalam basis data yang digunakan pada sistem ini, dengan *SQL Server* sebagai platform pengelolaannya.



Gambar 7. *Physical Data Model*

4.3. Implementasi *Stored Procedure*

Setiap *query* yang dieksekusi dalam sistem ini dilakukan melalui *Stored Procedure* dengan

penggunaan parameter yang telah ditentukan (*parameterized queries*), sehingga mencegah input langsung yang dapat dimanipulasi oleh pengguna berbahaya. Seperti pada proses memasukkan data pelanggan, sistem tidak menggunakan pernyataan SQL langsung melainkan menjalankan *Stored Procedure* yang telah terdefinisi dengan baik, di mana setiap parameter divalidasi sebelum data dimasukkan ke dalam basis data.

```
CREATE PROCEDURE sp_InsertCustomer
    @CustomerID INT,
    @CustomerFirstName VARCHAR(20),
    @CustomerLastName VARCHAR(20),
    @PhoneNumber VARCHAR(13),
    @Email VARCHAR(30),
    @Address VARCHAR(255),
    @ZipCode INT
AS
BEGIN
    INSERT INTO Customer (CustomerID, CustomerFirstName,
CustomerLastName, PhoneNumber, Email, Address, ZipCode)
    VALUES (@CustomerID, @CustomerFirstName, @CustomerLastName,
@PhoneNumber, @Email, @Address, @ZipCode);
END;
```

Gambar 8. *Stored Procedure* untuk *Insert* Data *Customer*

```
CREATE PROCEDURE sp_UpdateCustomer
    @CustomerID INT,
    @CustomerFirstName VARCHAR(20),
    @CustomerLastName VARCHAR(20),
    @PhoneNumber VARCHAR(13),
    @Email VARCHAR(30),
    @Address VARCHAR(255),
    @ZipCode INT
AS
BEGIN
    UPDATE Customer
    SET CustomerFirstName = @CustomerFirstName,
        CustomerLastName = @CustomerLastName,
        PhoneNumber = @PhoneNumber,
        Email = @Email,
        Address = @Address,
        ZipCode = @ZipCode
    WHERE CustomerID = @CustomerID;
END;
```

Gambar 9. *Stored Procedure* untuk *Update* Data *Customer*

```
CREATE PROCEDURE sp_DeleteCustomer
    @CustomerID INT
AS
BEGIN
    DELETE FROM Customer
    WHERE CustomerID = @CustomerID;
END;
```

Gambar 10. *Stored Procedure* untuk *Delete* Data *Customer*

4.4. Pengujian Keamanan

Pengujian keamanan dilakukan dengan menjalankan *query* menggunakan *input* berbahaya untuk mengevaluasi efektivitas *Stored Procedure* yang menerapkan *parameterized queries* dalam menangkal serangan SQL *Injection*.

```
BEGIN TRANSACTION;

-- Coba jalankan Stored Procedure dengan SQL Injection
EXEC sp_InsertCustomer 202, 'Hacker', 'Exploit', '081234567891',
    'hacker@example.com''; DROP TABLE Customer; --',
    'Jl. Rahasia No. 99', 10110;
```

(1 row affected)

Gambar 11. Percobaan *Input* yang Berbahaya

Pada gambar 11, dilakukan pengujian *input* berbahaya pada kolom email dengan tujuan percobaan untuk menghapus tabel *Customer*.

```
-- Periksa apakah tabel masih ada
SELECT * FROM Customer;
```

200 % Results Messages

	CustomerID	CustomerFirstName	CustomerLastName	PhoneNumber	Email	Address	ZipCode
1	1	Andi	Wijaya	081234567890	andi.wijaya@example.com	Jl. Merdeka 1	10110
2	2	Budi	Santoso	081234567891	budi.santoso@example.com	Jl. Merdeka 2	60119
3	3	Citra	Amalia	081234567892	citra.amalia@example.com	Jl. Merdeka 3	40123
4	4	Dedi	Saputra	081234567893	dedi.saputra@example.com	Jl. Merdeka 4	80225
5	5	Eka	Wulandari	081234567894	eka.wulandari@example.com	Jl. Merdeka 5	70122
6	6	Feri	Irawan	081234567895	feri.irawan@example.com	Jl. Merdeka 6	50233
7	7	Gina	Lestari	081234567896	gina.lestari@example.com	Jl. Merdeka 7	20212
8	8	Hendra	Susilo	081234567897	hendra.susilo@example.com	Jl. Merdeka 8	60232
9	9	Indah	Permata	081234567898	indah.permata@example.com	Jl. Merdeka 9	10245
10	10	Joko	Purnomo	081234567899	joko.purnomo@example.com	Jl. Merdeka 10	30312
11	202	Hacker	Exploit	081234567891	hacker@example.com'; DROP TABL	Jl. Rahasia No. 99	10110

Gambar 12. Pengecekan Hasil Eksekusi *Query* dengan *Input* Berbahaya

Lalu dilakukan pengecekan terhadap tabel *Customer* dengan mengambil seluruh data dari tabel *Customer* untuk memeriksa apakah tabel *Customer* terhapus. Hasilnya tabel *Customer* masih ada namun dengan penambahan data *Customer* dengan *CustomerID* 202 yang merupakan data *customer* untuk pengujian ini, seperti yang ada pada gambar 12. Hal ini berarti bahwa *input* berbahaya yang coba dimasukkan hanya dianggap sebagai *string* dan tidak mengeksekusi perintah tambahan untuk menghapus tabel *Customer*.

Hal ini membuktikan bahwa penggunaan *Stored Procedure* dengan *Parameterized Queries* mampu mencegah upaya SQL *Injection* yang dilakukan dengan menambahkan *input* berbahaya. Dengan adanya *parameterized query*, *input* yang berupa perintah untuk memanipulasi data hanya dianggap sebagai *string* biasa.

5. KESIMPULAN

Hasil penelitian yang telah dilakukan menghasilkan beberapa kesimpulan berikut:

a. Normalisasi hingga tingkat *Third Normal Form* (3NF) telah meningkatkan efisiensi pengelolaan data. Redundansi berkurang, hubungan antar tabel lebih jelas, dan proses pencarian serta pengolahan data menjadi lebih cepat dan akurat.

b. *Stored Procedure* dengan *parameterized queries* terbukti efektif dalam mencegah SQL *Injection*. *Input* pengguna diproses sebagai parameter, bukan bagian langsung dari perintah SQL, sehingga serangan berbasis injeksi kode tidak dapat dieksekusi.
c. Hasil pengujian keamanan menunjukkan bahwa sistem mampu menangkal serangan SQL *Injection* dengan baik. Percobaan dengan *input* berbahaya tidak berdampak pada eksekusi *query* maupun integritas tabel *Customer*, membuktikan bahwa metode yang diterapkan telah berjalan dengan baik.
d. Pengembangan lebih lanjut masih diperlukan untuk meningkatkan ketahanan sistem. Validasi *input* di tingkat aplikasi, sistem *logging* untuk mendeteksi aktivitas mencurigakan, serta optimalisasi indeks dan strategi eksekusi *query* dapat diterapkan guna memastikan sistem tetap efisien dan aman dalam skala yang lebih besar.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Singaperbangsa Karawang atas dukungan akademik dan fasilitas yang diberikan, serta Metrodata *Academy* atas kesempatan dan wawasan yang berharga dalam pengembangan proyek ini. Penghargaan juga diberikan kepada rekan-rekan tim proyek atas kerja samanya dalam membangun sistem ini, serta rekan-rekan mahasiswa yang turut memberikan masukan dan dukungan selama penelitian berlangsung. Tak lupa, apresiasi kepada semua pihak yang telah membantu dan memberikan motivasi, sehingga penelitian ini dapat terselesaikan dengan baik..

DAFTAR PUSTAKA

[1] A. Voutama and E. Novalia, "Perancangan Sistem Informasi Plakat Wisuda Berbasis Web Menggunakan UML dan Model Waterfall," 2022.
[2] C. Ayu Binangkit, A. Voutama, and N. Heryana, "PEMANFAATAN UML (UNIFIED MODELING LANGUAGE) DALAM PERENCANAAN SISTEM PENGELOLAAN SEWA ALAT MUSIK BERBASIS WEBSITE," 2023.
[3] R. Yulia Ekadianti, A. Voutama, and A. A. Ridha, "RANCANG BANGUN SISTEM INFORMASI PENDAFTARAN PASIEN BERBASIS WEBSITE DI RUMAH SAKIT PERMATA," *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, 2024.
[4] B. W. Aulia, M. Rizki, P. Prindiyana, and S. Surgana, "Peran Krusial Jaringan Komputer dan Basis Data dalam Era Digital," *JUSTINFO | Jurnal Sistem Informasi dan Teknologi Informasi*, vol. 1, no. 1, pp. 9–20, Dec. 2023, doi: 10.33197/justinfo.vol1.iss1.2023.1253.
[5] Nurgustin and M. I. P. Nasution, "STRATEGI MANAJEMEN DATA YANG EFEKTIF UNTUK MENINGKATKAN KINERJA BASIS DATA," *Jurnal Ilmiah Nusantara (JINU)*, vol. 1, no. 4, pp. 705–710, 2024, doi: 10.61722/jinu.v1i4.18890.
[6] A. Handijono and Z. Suhatman, "USULAN SOLUSI UNTUK MEMBANGUN CUSTOMER MASTER FILE PADA PT. ABC," *15 |Jurnal Ilmu Komputer JIK*, vol. VI, no. 03, 2023.
[7] D. Franciska Mey Dina, "Normalisasi Database Rancangan Sistem Penyewaan Buku Berbayar," 2022.
[8] A. A. Onyekachi, A. O. Agbakwuru, and D. O. Njoku, "SQL Injection Attack on Web Base Application: Vulnerability Assessments and Detection Technique," *International Research Journal of Engineering and Technology*, 2021, [Online]. Available: https://www.researchgate.net/publication/353257660
[9] I. S. Crespo-Martínez, A. Campazas-Vega, Á. M. Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez-Aparicio, and C. Fernández-Llamas, "SQL injection attack detection in network flow data," *Comput Secur*, vol. 127, Apr. 2023, doi: 10.1016/j.cose.2023.103093.
[10] V. Abdullayev and A. S. Chauhan, "SQL Injection Attack: Quick View," 2023, *Mesopotamian Academic Press*. doi: 10.58496/MJCS/2023/006.
[11] A. Dudu, A. Rohmana, H. Mubarok, and R. Gunawan, "PENGUKURAN KINERJA STORED PROCEDURE PADA DATABASE RELASIONAL," *Jurnal Siliwangi*, vol. 5, no. 2, 2019.
[12] S. Yakan Khomsi Pane, N. Ghaniaviyanto Ramadhan, and F. Dharma Adhinata, "Perancangan Basis Data Menggunakan Normalisasi Tabel Pada Perusahaan Dagang Barokah Abadi," *Journal of Dinda: Data Science, Information Technology, and Data*

Analytics, vol. 2, no. 2, pp. 90–96, 2022, [Online]. Available: http://journal.ittelkom-pwt.ac.id/index.php/dinda

[13] K. Ahmad and M. Karim, "A Method to Prevent SQL Injection Attack using an Improved Parameterized Stored Procedure," 2021. [Online]. Available: www.ijacsa.thesai.org

[14] A. D. Hardiansyah, D. C. Nugrahaeni, P. Dewi, and M. Kom, "PERANCANGAN BASIS DATA SISTEM INFORMASI PERWIRA TUGAS BELAJAR (SIPATUBEL) PADA KEMENTERIAN PERTAHANAN," in *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA)*, Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA), 2020.

[15] I. Listiawan and Y. Marsongko, "MODEL DATABASE REVITALISASI LUMBUNG PANGAN DESA (STUDI KASUS DESA PENEN KECAMATAN NGAGLIK SLEMAN YOGYAKARTA)," in *Seminar Nasional UNRIYO*, Seminar Nasional UNRIYO, 2020.

[16] K. ' Afiifah, Z. Fira Azzahra, and A. D. Anggoro, "Analisis Teknik Entity-Relationship Diagram dalam Perancangan Database: Sebuah Literature Review," *JURNAL INTECH*, vol. 3, no. 2, pp. 18–22, 2022.

[17] D. D. Ramadhan, R. Mumpuni, and A. N. Sihananto, "IMPLEMENTASI METODE RAPID APPLICATION DEVELOPMENT (RAD) DALAM PENGEMBANGAN SISTEM ENTERPRISE INDUSTRI TEKSTIL BERBASIS WEBSITE," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 3S1, Oct. 2024, doi: 10.23960/jitet.v12i3S1.5222.

[18] F. Surya Halim, T. Gantini, and K. Maranatha Jl drg Surya Sumantri No, "Model Perancangan Aplikasi Konsultasi Pengobatan Herbal," *Jurnal Strategi*, vol. 3, p. 332, 2021.

[19] A. Mutedi and B. Tjahjono, "Systematic Literature Review: Preventing SQL Injection Attacks Using Tools OWASP CSR Web Application Firewall," *Jurnal Informatika Universitas Pamulang*, vol. 7, no. 1, pp. 151–156, 2022, doi: 10.32493/informatika.v7i1.17590.

[20] N. Hartono, "Comparison of Stored Procedures on Relational Database Management Systems," *Journal TECH-E*, 2021, [Online]. Available: http://bsti.ubd.ac.id/e-jurnal

[21] I. Bisnis, M. Bekasi, M. Fadillah, and Y. Servanda, "ANALISIS EFEKTIVITAS TEKNIK PARAMETERIZED QUERIES DALAM MENCEGAH SERANGAN SQL INJECTION MENGGUNAKAN DVWA," *JUPITER Teknologi Informatika & Komputer*, vol. 5, no. 2, 2024.