



BIRMINGHAM CITY
University

Project Title:

Design and Development of Database System for Software Company

Name:

Oluwafisayomi Isola

Student_ID:

22185897

Course:

MSc Computer Science

School:

School of Computing and Digital Technology

Module Title:

Database Design & Development

Module Leader:

Dr. Essa Shahra

January, 2024

TABLE OF CONTENTS

1	Critical Evaluation of Database System	3
1.1	Relational Database Management System (RDBMS)	3
1.2	NoSQL Database Management System.....	4
1.3	NewSQL Database Management System	5
1.4	Object - Oriented Database Management System	6
1.5	Multimodel Database Management System	8
1.6	Best Database to be Used for Software Company	9
2	Database Design	10
2.1	Entity Relationship Diagram (ERD) (Logical Model)	10
2.2	Relational Schema Mapping	11
2.3	Assumptions	11
3	Database Development	12
3.1	Physical Design of the System	12
3.2	Tables and Relations.....	13
3.3	Queries Examples	22
4	Security Scenario.....	28
5	Conclusion.....	29
6	References	30

1.0 Critical Evaluation of Database System

The database management system is the beating heart of most operational and analytical business systems. The database management system (DBMS) is the main software for processing, storing and managing data, as well as serving it to applications as well as end users. However, there are numerous types of database management system (DBMS) products, each with its own set of advantages and disadvantages.

1.1 Relational Database Management System (RDBMS)

A relational database management system (RDBMS) is a type of software that enables users to create, manage, and interact with databases that are relational. It helps to effectively organise, store, and retrieve data through the use of a relational model, which structures data into tables with rows and columns. The columns are generally referred to as attributes while the rows are referred to as tuples, and every table or relation is managed through the use of primary key and foreign key. Microsoft SQL Server, Oracle Database, PostgreSQL and IBM are common examples of the relational database management system.

Advantages:

- Simple to use due to the fact that only simple queries must be entered to access the data, so it can be used with little to no training
- Fast Result: It allows the user to access data quickly since it has simple structures and it's also simple to use.
- Privacy: With this database system, only a database administrator has permission to grant users access to the data, making the data secure.
- Relational Databases are based on the ACID (Atomicity, Consistency, Isolation, Durability) model which provides consistency and availability as strong properties
- Relational model is higher to other models like hierarchal model because it does not use physical links to establish a relationship with the use of logical links; relationships are established through foreign keys rather than physical pointers.

Disadvantages:

- Storage Space: Due to a large number of rows and columns, this database requires a large amount of physical memory.

- The relational model supports only one type of relation database. Because of the strict schema, tasks will have to adapt to the database's requirements rather than the database conforming to the specifications of the application
- This database is capable of growing up, but its ability to grow is limited by upgrades to the hardware made to one server to improve its efficiency. As a result, administrators must expend more effort to upgrade their relational databases.
- Although relational databases include both open and closed origin platforms, many private platforms such as Oracle requires substantial funding from users, businesses and organisations in order to receive and use the benefits and advanced features. As a result, it is an expensive method of data storage

1.2 NoSQL Database Management System

NoSQL (Not Only SQL) database is a type of database management system that can manage numerous types of unstructured, semi – structures, or highly variable data. Comparing with standard relational databases (RDBMS), they provide benefits in terms of performance, scalability, and flexibility. They use various data models for data storing and retrieval, deviating from the relational model of relational database management systems (RDBMS). They come in various types such as document stores, key – value stores, column stores, etc. and are mainly used for managing substantial amounts of unstructured or semi – structures data. Examples are: MongoDB, Cassandra, Redis etc.

Advantages:

- Flexible and adaptable schema designs are possible with NoSQL databases. Flexible data structures allow for the management of a variety of data types because they do not require already established schemas.
- NoSQL databases extend across several servers or nodes with ease because of their horizontally extensible design. For processing large volumes of data and high volume applications, this makes them excellent.
- For specific use cases, they frequently offer high performance and quicker data access. Performance is improved for specific workloads by the read and writes optimisations of many NoSQL databases in distributed environments.

- NoSQL databases handle large datasets with efficiency and readily handle unstructured data types, which traditional relational databases may find difficult to handle.

Disadvantages:

- Limited Query Ability: NoSQL databases may have less querying power than RDBMS SQL depending on the type of database. Some NoSQL databases might not support complex queries.
- Final Uniformity: Availability and accepting partitions are given priority over strong consistency in certain NoSQL databases. Data may become temporarily inconsistent between elements as a result, potentially causing consistency problems along the line
- Lack of Development and Standards: In terms of features, devices and support, NoSQL databases may be less mature and regulated than RDBMS. Compared to well – established RDBMS solutions, this could present interface and support challenges.
- Not Ideal for All Use Cases: NoSQL databases work well for some use cases and data kinds, but they might not be appropriate for applications that need complicated transactions, tight relationships between data, or structured data

1.3 NewSQL Database Management System

NewSQL is a class of current relational database management systems (RDBMS) that aims to combine the benefits of traditional RDBMS with the scalability and performance improvement offered by NoSQL databases. These systems are designed to address the shortcomings of traditional RDBMS while maintaining the ACID (Atomicity, Consistency, Isolation, and Durability) properties crucial for transactional applications.

Advantages:

- NewSQL databases maintain the ACID characteristics that are essential for operational credibility, ensuring consistency and dependability of data in distributed settings.

- When compared to conventional RDBMs, they offer better scalability, especially in distributed environments. To improve scalability, NewSQL databases use strategies like replication, splitting and enhanced agreement management.
- Performance can be improved by query execution, searching and storage mechanism optimisation in NewSQL systems, particularly for workloads that involve a lot of studying. They have a low delay and can handle massive amounts of data with efficiency.
- Developers used to working with SQL – based querying will find NewSQL databases familiar and easy to use because they are SQL – complaint. The integration and transition into current applications are made easier by this compatibility
- They provide support for complex queries and interactions, which is important for applications that call for detailed adjustments to data or operations involving several tables.

Disadvantages:

- NewSQL database implementation can be challenging, particularly in distributed systems. It may take specialised knowledge to design and manage a system that strikes a balance between scalability and ACID compliance
- Compared to traditional RDBMS, NewSQL database might have fewer users and fewer well – developed solutions available.
- While scalability and ACID compliance are goals of NewSQL databases, in some circumstances, reaching both at scale may require trade – offs in performance when compared to specialised NoQL databases
- Working with NewSQL databases may require some learning for developers, particularly if they are more used to working with traditional RDBMS or NoSQL databases.

1.4 Object - Oriented Database Management System

Object Programming languages can incorporate database functionality with the aid of object database management systems. Programmers that work with languages like Java and C++ can create a fully functional database while maintaining compatibility with their preferred language. Real world entities and data must be arranged in the

database as objects. A collection of objects, each consisting of values kept in instance variables, make up the Object – Oriented model. Additionally, the object has “methods”, which are collections of code that run inside the object. Classes are collections of objects that share common methods and value types.

Advantages:

- The native object structure found in programming languages is preserved by OODBMS, which stores objects directly. This makes object – relational mapping (ORM) less necessary and improves user ease of use by streamlining the mapping between application code and the database.
- Complex relationships between objects are possible with OODBMS, better modelling real – world scenarios. It makes complicated data models and relationships possible by supporting packaging, variability, and recurrence
- OODBMS can shorten development cycles for object oriented applications by enabling developers to work with a database that is more structured like the application code, hence minimising efficiency error.
- When compared to relational databases, retrieving all objects from the database can be more efficient because it does not require the complex form activities that are frequently needed in RDBMSs.
- Schema Flexibility: OODBMS often provide schema flexibility, enabling adjustments to object structures without leading to modifications to the database schema as a whole.

Disadvantages:

- In comparison to RDBMS, OODBMS has not been as widely adopted, which has led to a lack of community support, resources, and mature tools for developers.
- OODBMS are not organised, which results in exclusive products with different features and implementations. Conversion and usability may suffer as a result.
- Optimised relational queries may perform better than complex queries and data retrieval operations involving relationships, even though accessing complete objects may be faster.
- The distinct data structures and query languages of OODBMS may present an adjustment process for developers used to relational databases.

- When working with large data sets or distributed structures, some OODBMS may experience scalability issues in comparison to specific NoSQL or extended relational databases.

1.5 Multimodel Database Management System

A database system that supports several data models on a single integrated platform is known as Multimodel database management system. Working with different data models within the same database management system is made possible by this method, including relational, document – oriented, graph – based, and key – value pairs.

Advantages:

- Based on particular application requirements, MDBMS provides the flexibility to use different data models. With a single system, it supports multiple data kinds and access patterns.
- By enabling several data models to coexist within the same system, MDBMS eliminates data duplication by doing away with the need for distinct databases for each type of data model, improving data consistency and management.
- MDBMS facilitates data sharing and integration by supporting multiple data models, which encourages connection amongst various applications or systems that may use different data models
- Without having to switch between different database systems, developers can work with familiar data models, which can expedite development and simplify maintenance.
- By utilising the advantage of different data models for different kinds of data, MDBMS can maximise performance and enable scalability for a variety of workloads and data access patterns.

Disadvantages:

- Essential and effective planning is necessary for the implementation and management of a system that supports multiple data models, as this could result in an increase in system complexity and management overhead.
- Despite the fact that MDBMS is flexible, some use cases may benefit more from a particular data model than others. When using different data models in the same system, some operations may not perform as well as they should.

- The MDBMS supports a wide range of data models, which users and developers may need to become familiar with and adjust to. This could provide a learning process and make it difficult to fully understand and utilise all of the features that are provided.
- Various MDBS solutions might have exclusive applications and features for supporting multiple data models, which could limit accessibility and compatibility across platforms
- Linkage problems and restrictions on feature parity between models may arise from MDBMS's struggles with regulations and interaction with various data models.

1.6 Best Database to be Used for Software Company

From the research on the different types of databases we have, I'd conclude that the Relational Database Management System (RDBMS) is the best for use in a software company because when RDBMS is compared to other databases, it is ideal for structured data and complex queries compared to NoSQL that handles unstructured or semi – structured data and offer better scalability for certain use cases. It has also has a longer history and more established tools and resources compared to NewSQL databases that aims to combine the scalability of NoSQL with ACID compliance, which is often suitable for distributed systems.

2.0 Database Design

2.1 Entity Relationship Diagram (ERD) (Logical Model)

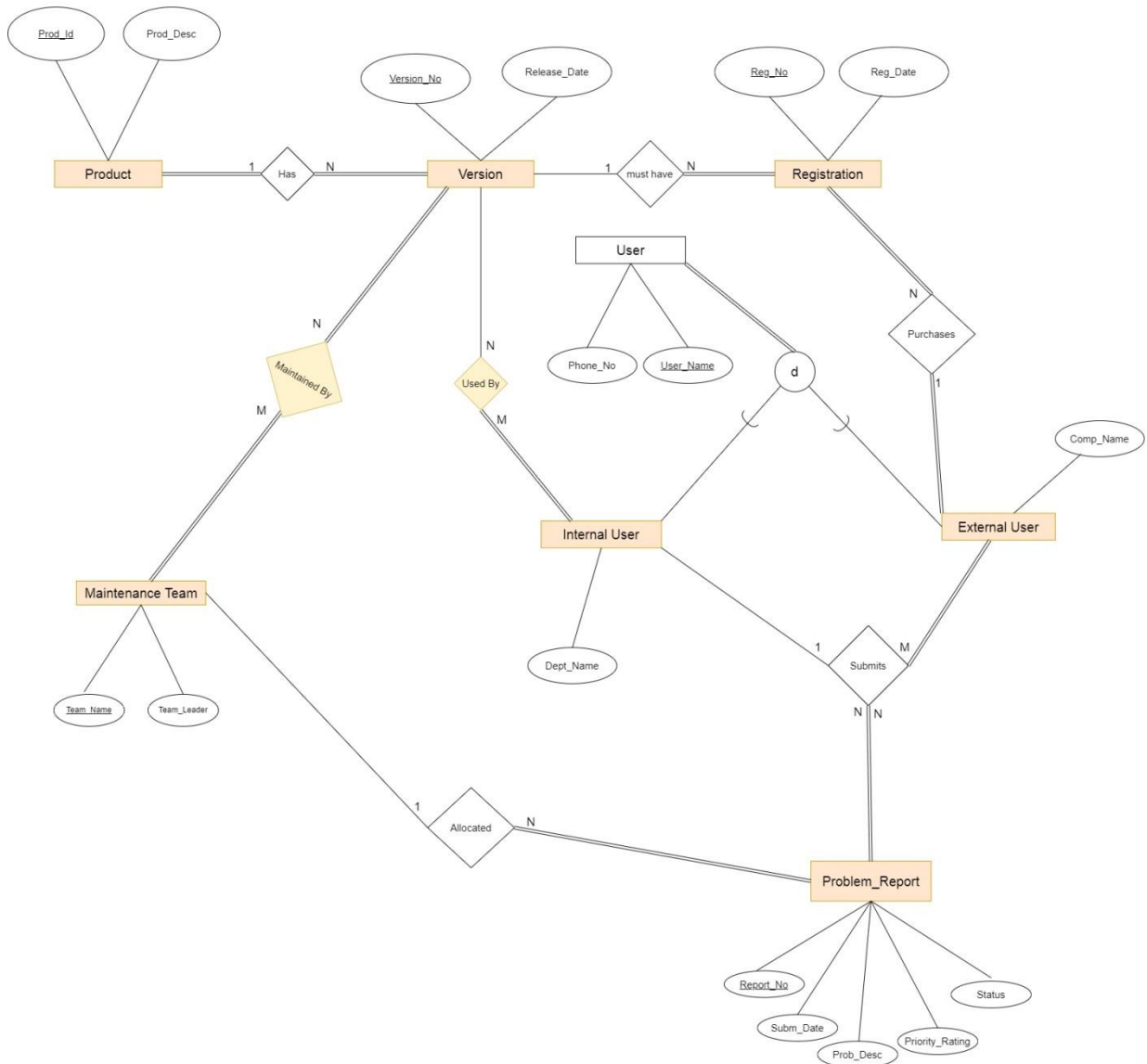
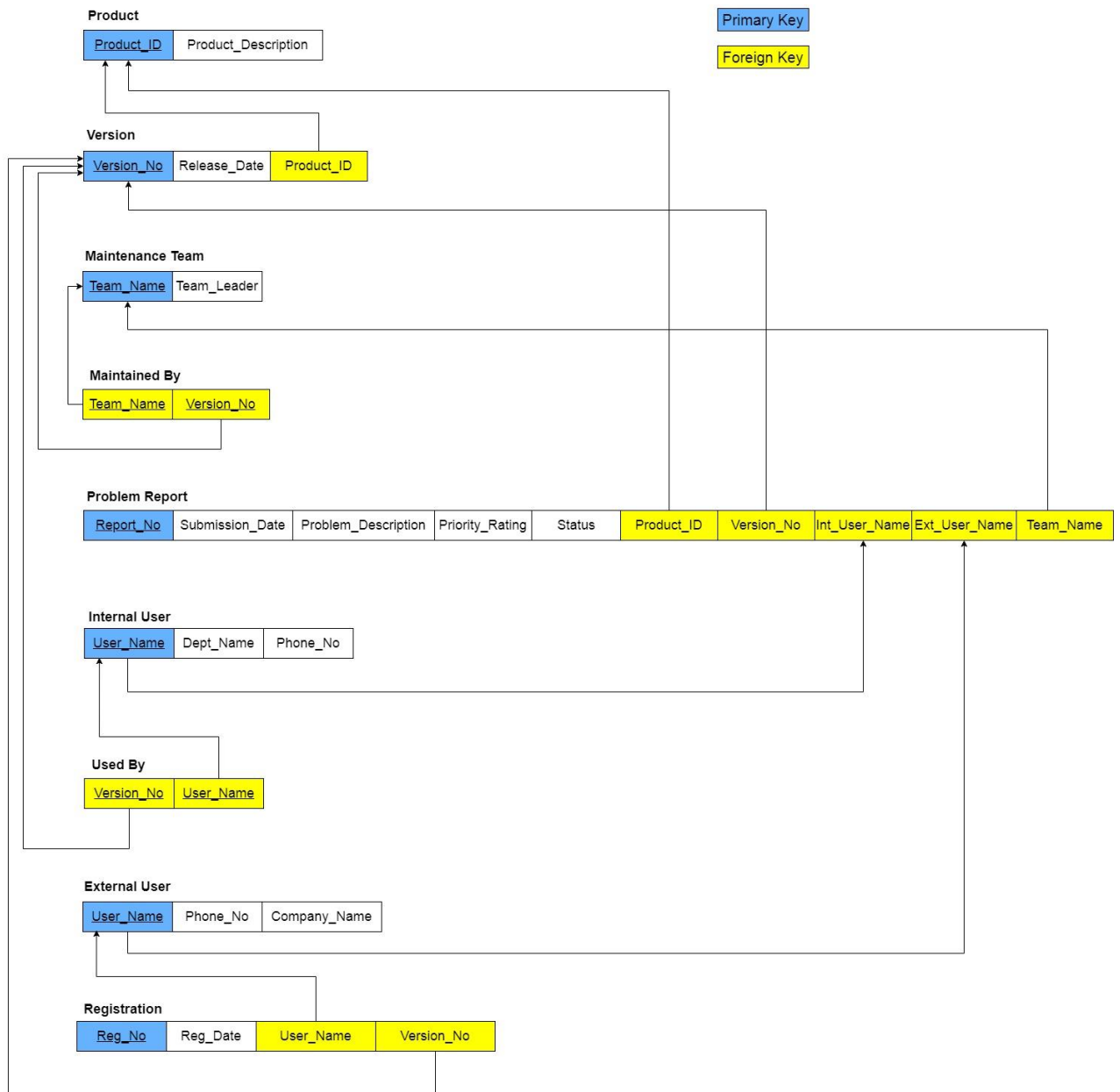


Fig 1: Entity Relationship Diagram for Software Company

2.2 Relational Schema Mapping



2.3 Assumptions

- Every product must have at least one version and no version exists without being a product hence total participation on both sides
- A version must be registered before it can be sold to external users hence the total participation on the side of registration
- One external user can purchase more than one registered version of a product
- Many internal users can use many versions of a product
- Versions don't have to be registered before they can be used by internal users
- Many problem reports can be allocated to one maintenance team but many maintenance team cannot resolve one problem report
- Many maintenance teams can maintain many versions of a product and vice versa.

3 Database Development

3.1 Physical Design of the System

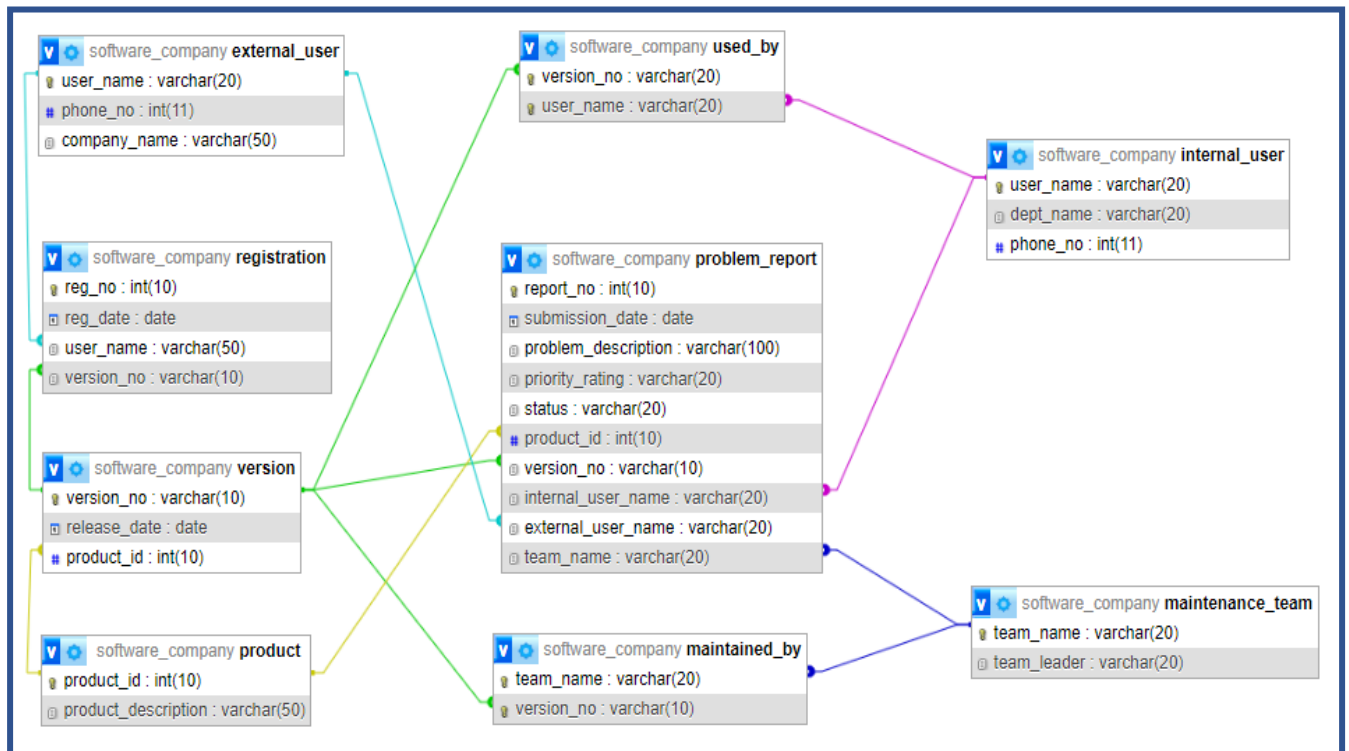


Fig 3: Screenshot of the Software Company Physical Database Schema and Relationship Representation

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> external_user	★ Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> internal_user	★ Browse Structure Search Insert Empty Drop	14	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> maintained_by	★ Browse Structure Search Insert Empty Drop	54	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> maintenance_team	★ Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> problem_report	★ Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_general_ci	96.0 KiB	-
<input type="checkbox"/> product	★ Browse Structure Search Insert Empty Drop	25	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> registration	★ Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> used_by	★ Browse Structure Search Insert Empty Drop	15	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> version	★ Browse Structure Search Insert Empty Drop	54	InnoDB	utf8mb4_general_ci	32.0 KiB	-
9 tables	Sum	218	InnoDB	utf8mb4_general_ci	304.0 KiB	0 B

Fig 4: Screenshot of the Software Company Physical Design of the Company System

3.2 Tables and Relations

3.2.1 Product Entity

This table illustrates a list of different products (25) designed in a software company. The table consists of the 'product_id' which is identified as the primary key and is unique to every product and the product_description which gives a brief detail about a product.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 product_id	int(10)			No	None			Change Drop More
<input type="checkbox"/>	2 product_description	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns Remove from central columns

```
SELECT * FROM `product`
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows: | Sort by key: None

Extra options

				product_id	product_description
<input type="checkbox"/>				10001	Microsoft Windows
<input type="checkbox"/>				10002	Adobe Photoshop
<input type="checkbox"/>				10003	Google Chrome
<input type="checkbox"/>				10004	Salesforce CRM
<input type="checkbox"/>				10005	Oracle Database
<input type="checkbox"/>				10006	Apple iPhone
<input type="checkbox"/>				10007	Amazon Web Services
<input type="checkbox"/>				10008	IBM Watson
<input type="checkbox"/>				10009	Autodesk AutoCAD
<input type="checkbox"/>				10010	Intuit QuickBooks
<input type="checkbox"/>				10011	VMware vSphere
<input type="checkbox"/>				10012	Atlassian Jira
<input type="checkbox"/>				10013	Unity
<input type="checkbox"/>				10014	Facebook Messenger
<input type="checkbox"/>				10015	Zoom
<input type="checkbox"/>				10016	Twitter
<input type="checkbox"/>				10017	Whatsapp
<input type="checkbox"/>				10018	Instagram
<input type="checkbox"/>				10019	Zendesk
<input type="checkbox"/>				10020	Xampp
<input type="checkbox"/>				10021	Splunk
<input type="checkbox"/>				10022	Dropbox
<input type="checkbox"/>				10023	WordPress
<input type="checkbox"/>				10024	Cisco WebEx
<input type="checkbox"/>				10025	HubSpot CRM

☐ Check all With selected: Edit Copy Delete Export

3.2.2 Version Entity

When a product has different versions, it retains that information within this table. The 'version_no' is identified as the primary key along with details regarding the release date of the versions while the 'product_id' is identified as the foreign key which has a relationship with the product entity.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	version_no	varchar(10)	utf8mb4_general_ci	No	None			Change Drop More
<input type="checkbox"/>	2	release_date	date		No	None			Change Drop More
<input type="checkbox"/>	3	product_id	int(10)		No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns

SELECT * FROM `version`

Profiling

[Edit inline]

[Edit]

[Explain SQL]

[Create PHP code]

[Refresh]

Show all

Number of rows: 100

Filter rows:

Search this table

Sort by key: None

Extra options

Edit

Copy

Delete

version_no

release_date

product_id

Edit

Copy

Delete

10001.1

2022-12-03

10001

Edit

Copy

Delete

10001.2

2023-05-08

10001

Edit

Copy

Delete

10001.3

2023-12-12

10001

Edit

Copy

Delete

10002.1

2021-03-15

10002

Edit

Copy

Delete

10003.1

2021-06-20

10003

Edit

Copy

Delete

10003.2

2022-01-10

10003

Edit

Copy

Delete

10004.1

2021-09-25

10004

Edit

Copy

Delete

10004.2

2022-02-28

10004

Edit

Copy

Delete

10004.3

2022-07-14

10004

Edit

Copy

Delete

10005.1

2021-11-30

10010

Edit

Copy

Delete

10005.2

2022-04-18

10005

Edit

Copy

Delete

10005.3

2022-08-22

10005

Edit

Copy

Delete

10005.4

2023-01-05

10005

Edit

Copy

Delete

10005.5

2023-06-10

10005

Edit

Copy

Delete

10006.1

2021-04-10

10006

Edit

Copy

Delete

10007.1

2021-07-02

10007

Edit

Copy

Delete

10007.2

2022-03-11

10007

Console

Edit

Copy

Delete

version_no

release_date

product_id

Edit

Copy

Delete

10008.2

2023-02-19

10008

Edit

Copy

Delete

10009.1

2021-08-15

10009

Edit

Copy

Delete

10009.2

2022-01-22

10009

Edit

Copy

Delete

10009.3

2022-05-30

10009

Edit

Copy

Delete

10010.1

2021-12-08

10010

Edit

Copy

Delete

10010.2

2022-06-14

10010

Edit

Copy

Delete

10010.3

2023-03-20

10010

Edit

Copy

Delete

10011.1

2022-09-07

10011

Edit

Copy

Delete

10011.2

2023-04-01

10011

Edit

Copy

Delete

10012.1

2022-11-12

10012

Edit

Copy

Delete

10013.1

2023-01-30

10013

Edit

Copy

Delete

10013.2

2023-07-25

10013

Edit

Copy

Delete

10014.1

2023-05-19

10014

Edit

Copy

Delete

10015.1

2023-09-28

10015

Edit

Copy

Delete

10015.2

2023-12-31

10015

Edit

Copy

Delete

10016.1

2022-01-05

10016

Edit

Copy

Delete

10017.1

2022-05-20

10017

Edit

Copy

Delete

10018.1

2022-08-07

10018

Edit

Copy

Delete

10018.2

2023-02-14

10018

Edit

Copy

Delete

10018.3

2023-07-18

10018

Edit

Copy

Delete

10019.1

2022-03-25

10019

Edit

Copy

Delete

10019.2

2022-09-03

10019

Edit

Copy

Delete

10019.3

2023-04-10

10019

3.2.3 External User Entity

A version could be used by external users; this table retains the basic details of the external user. The 'user_name' is identified as the primary key along with details regarding their phone number and their company name which must be recorded when they submit a report.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	user_name	varchar(20)	utf8mb4_general_ci	No	None			Change Drop More
<input type="checkbox"/>	2	phone_no	int(11)		No	None			Change Drop More
<input type="checkbox"/>	3	company_name	varchar(50)	utf8mb4_general_ci	No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns

```
SELECT * FROM `external_user`
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: Filter rows: Sort by key:

Extra options

				user_name	phone_no	company_name
<input type="checkbox"/>	Edit	Copy	Delete	Annie Jackson	795332131	IBM
<input type="checkbox"/>	Edit	Copy	Delete	Bekky Daniels	792553101	Salesforce
<input type="checkbox"/>	Edit	Copy	Delete	Bryan Rolland	791802185	Amazon
<input type="checkbox"/>	Edit	Copy	Delete	Christiannah Joe	792443105	Oracle
<input type="checkbox"/>	Edit	Copy	Delete	Donald Fred	795212201	Adobe
<input type="checkbox"/>	Edit	Copy	Delete	James Stones	712345678	Microsoft
<input type="checkbox"/>	Edit	Copy	Delete	John Kennedy	791356544	Intel
<input type="checkbox"/>	Edit	Copy	Delete	Kings Smith	781325231	Google
<input type="checkbox"/>	Edit	Copy	Delete	Rio Jack	795112065	SAP
<input type="checkbox"/>	Edit	Copy	Delete	Robert Green	791458133	Apple

☐ Check all With selected: Edit Copy Delete Export

3.2.4 Internal User Entity

A version could also be used by internal users; this table retains the basic details of the internal user. The 'user_name' is identified as the primary key along with details regarding their phone number and their department name which must be recorded when they submit a report.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	user_name	varchar(20)	utf8mb4_general_ci	No	None			Change Drop More
<input type="checkbox"/>	2	dept_name	varchar(20)	utf8mb4_general_ci	No	None			Change Drop More
<input type="checkbox"/>	3	phone_no	int(11)		No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns

```
SELECT * FROM `internal_user`
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 ▼ Filter rows: Sort by key: None ▼

Extra options

⬅️ ➡️

▼

				user_name	dept_name	phone_no
<input type="checkbox"/>				Adrian Jones	Communications	792917865
<input type="checkbox"/>				Charles Darwin	Workplace Management	793506834
<input type="checkbox"/>				Charles Godwin	Analytics	794217328
<input type="checkbox"/>				David Mark	Development	795644131
<input type="checkbox"/>				Francisca Joe	Project Management	791889320
<input type="checkbox"/>				Goodness Rocky	Quality Assurance	734723781
<input type="checkbox"/>				Grace Thompson	Engineering & Develo	791517231
<input type="checkbox"/>				Joss Richard	Software Development	795652381
<input type="checkbox"/>				Junior Darlington	Human Resources	792005173
<input type="checkbox"/>				Leonard Benson	Software Analysis	793256584
<input type="checkbox"/>				Luke Cornell	Sales and Business	793257551
<input type="checkbox"/>				Paul Thompson	Engineering	791345763
<input type="checkbox"/>				Prince Maxwell	Data Science	791226551
<input type="checkbox"/>				Victoria Spencer	User Experience	792711728

⬆️

☐ Check all

With selected:

Edit

Copy

Delete

Export

3.2.5 Maintenance Team Entity

Different problem reports are sent to maintenance team and each team has a unique name 'team_name' which is identified as the primary key and also a team leader.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	team_name	varchar(20)	utf8mb4_general_ci	No	None			Change Drop More
<input type="checkbox"/>	2	team_leader	varchar(20)	utf8mb4_general_ci	No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns

SELECT * FROM `maintenance_team`

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

team_name

team_leader

<input type="checkbox"/>	Edit	Copy	Delete	Accurate_Teams	Annabel Fisher
<input type="checkbox"/>	Edit	Copy	Delete	Alpha_Teams	Shawn Tiffany
<input type="checkbox"/>	Edit	Copy	Delete	Emirate_Teams	James Wills
<input type="checkbox"/>	Edit	Copy	Delete	Gladiators_Teams	Queen Lehman
<input type="checkbox"/>	Edit	Copy	Delete	Mastermind_Teams	Stephen Craig
<input type="checkbox"/>	Edit	Copy	Delete	Mavericks_Teams	Anthony Wiliams
<input type="checkbox"/>	Edit	Copy	Delete	Star_Teams	Reid Wilmer
<input type="checkbox"/>	Edit	Copy	Delete	Ultimate_Teams	Anita Bassey
<input type="checkbox"/>	Edit	Copy	Delete	Universal_Teams	Victor Griggs
<input type="checkbox"/>	Edit	Copy	Delete	Warriors_Teams	Stephen Young

☐ Check all With selected: Edit Copy Delete Export

3.2.6 Maintained by Relationship

Different Versions of a product are maintained by different teams, and this is represented in this table. Different versions (uniquely identified with 'version_no' which is the primary key) are maintained by different teams (each team has a unique name 'team_name' which is identified as the primary key). Both attributes are also identified as foreign keys because they have relationships with maintenance team entity and version entity.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 team_name	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	2 version_no	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns

```
SELECT * FROM `maintained_by` ORDER BY `maintained_by`.`version_no` ASC
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

1 > >>

☐ Show all

Number of rows: 25

Filter rows:

Sort by key: None

Extra options

	team_name	version_no
<input type="checkbox"/>	Universal_Teams	10001.1
<input type="checkbox"/>	Emirate_Teams	10001.2
<input type="checkbox"/>	Accurate_Teams	10001.3
<input type="checkbox"/>	Alpha_Teams	10002.1
<input type="checkbox"/>	Alpha_Teams	10003.1
<input type="checkbox"/>	Mavericks_Teams	10003.2
<input type="checkbox"/>	Star_Teams	10004.1
<input type="checkbox"/>	Mastermind_Teams	10004.2
<input type="checkbox"/>	Emirate_Teams	10004.3
<input type="checkbox"/>	Mavericks_Teams	10005.1
<input type="checkbox"/>	Warriors_Teams	10005.2
<input type="checkbox"/>	Mastermind_Teams	10005.3
<input type="checkbox"/>	Emirate_Teams	10005.4
<input type="checkbox"/>	Gladiators_Teams	10005.5
<input type="checkbox"/>	Emirate_Teams	10006.1
<input type="checkbox"/>	Mastermind_Teams	10007.1
<input type="checkbox"/>	Mavericks_Teams	10007.2
<input type="checkbox"/>	Ultimate_Teams	10008.1
<input type="checkbox"/>	Alpha_Teams	10008.2
<input type="checkbox"/>	Warriors_Teams	10009.1
<input type="checkbox"/>	Universal_Teams	10009.2
<input type="checkbox"/>	Accurate_Teams	10009.3
<input type="checkbox"/>	Universal_Teams	10010.1
<input type="checkbox"/>	Gladiators_Teams	10010.2
<input type="checkbox"/>	Star_Teams	10010.3
<input type="checkbox"/>	Star_Teams	10011.1
<input type="checkbox"/>	Gladiators_Teams	10011.2
<input type="checkbox"/>	Alpha_Teams	10012.1
<input type="checkbox"/>	Mastermind_Teams	10013.1
<input type="checkbox"/>	Emirate_Teams	10013.2
<input type="checkbox"/>	Accurate_Teams	10014.1
<input type="checkbox"/>	Warriors_Teams	10015.1
<input type="checkbox"/>	Emirate_Teams	10015.2
<input type="checkbox"/>	Universal_Teams	10016.1
<input type="checkbox"/>	Mastermind_Teams	10017.1
<input type="checkbox"/>	Star_Teams	10018.1
<input type="checkbox"/>	Accurate_Teams	10018.2
<input type="checkbox"/>	Warriors_Teams	10018.3

3.2.7 Used by Relationship

Versions used by internal users are recorded in this table. It has version_no as the primary key and user_name as a foreign key which has relationship with the internal user entity.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	version_no	varchar(10)	utf8mb4_general_ci	No	None			Change Drop More
<input type="checkbox"/>	2	user_name	varchar(20)	utf8mb4_general_ci	No	None			Change Drop More

☐ Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns

SELECT * FROM `used_by`

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

				version_no	user_name
<input type="checkbox"/>	Edit	Copy	Delete	10001.1	Adrian Jones
<input type="checkbox"/>	Edit	Copy	Delete	10002.1	Charles Darwin
<input type="checkbox"/>	Edit	Copy	Delete	10003.2	Charles Godwin
<input type="checkbox"/>	Edit	Copy	Delete	10004.1	David Mark
<input type="checkbox"/>	Edit	Copy	Delete	10005.5	Francisca Joe
<input type="checkbox"/>	Edit	Copy	Delete	10006.1	Goodness Rocky
<input type="checkbox"/>	Edit	Copy	Delete	10007.2	Grace Thompson
<input type="checkbox"/>	Edit	Copy	Delete	10008.2	Joss Richard
<input type="checkbox"/>	Edit	Copy	Delete	10009.3	Junior Darlington
<input type="checkbox"/>	Edit	Copy	Delete	10010.3	Leonard Benson
<input type="checkbox"/>	Edit	Copy	Delete	10011.2	Luke Cornell
<input type="checkbox"/>	Edit	Copy	Delete	10012.1	Paul Donald
<input type="checkbox"/>	Edit	Copy	Delete	10013.2	Prince Maxwell
<input type="checkbox"/>	Edit	Copy	Delete	10014.1	Victoria Spencer
<input type="checkbox"/>	Edit	Copy	Delete	10023.1	Victoria Spencer

☐ Check all With selected: Edit Copy Delete Export

3.2.8 Problem Report Entity

Different problems of different versions of a software product are being submitted by the internal and external user. This table has report_no as the primary key; it also has details of submission date, problem description, priority rating, status (fixed, in progress or not a problem), product_id, version_no, internal_user_name, external_user_name and team_name. The foreign keys of the table are:

Product_id which has relationship with the product entity to tell which product was reported.

Version_no which has a relationship with the version entity to tell which particular version of a product was reported.

Internal_user_name (NULL) which has a relationship with the internal_user entity to give the name, department and phone number of the internal user that submitted the report.

External_user_name (NULL) which has a relationship with the external_user entity to give the name, phone number and company name of the external user that submitted the report.

Team_name which has a relationship with the maintenance team entity to give the name of the team that is maintaining the particular version of a product.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 report_no	int(10)			No	None			Change Drop More
<input type="checkbox"/>	2 submission_date	date			No	None			Change Drop More
<input type="checkbox"/>	3 problem_description	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 priority_rating	int(5)			No	None			Change Drop More
<input type="checkbox"/>	5 status	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	6 product_id	int(10)			No	None			Change Drop More
<input type="checkbox"/>	7 version_no	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	8 internal_user_name	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	9 external_user_name	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	10 team_name	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More

SELECT * FROM `problem_report`										
Profiling [Edit Inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]										
Show all Number of rows: 100 Filter rows: Search this table Sort by key: None										
Extra options										
	report_no	submission_date	problem_description	priority_rating	status	product_id	version_no	internal_user_name	external_user_name	team_name
<input type="checkbox"/>	Edit Copy Delete	1	2023-08-08	Performance Issue	8	In Progress	10001 10001.3	Charles Godwin	NULL	Alpha_Teams
<input type="checkbox"/>	Edit Copy Delete	2	2023-04-12	Integration Issue	7	In Progress	10002 10002.1	NULL	Bekky Daniels	Gladiators_Teams
<input type="checkbox"/>	Edit Copy Delete	3	2022-11-25	Software Bug	5	Fixed	10003 10003.2	NULL	Annie Jackson	Emirate_Teams
<input type="checkbox"/>	Edit Copy Delete	4	2022-09-30	Software Crash	9	In Progress	10004 10004.2	NULL	Luke Cornell	Accurate_Teams
<input type="checkbox"/>	Edit Copy Delete	5	2023-02-17	Compatibility Error	6	Not a Problem	10005 10005.4	Paul Donald	NULL	Warriors_Teams
<input type="checkbox"/>	Edit Copy Delete	6	2022-07-10	UI Glitch	4	In Progress	10006 10006.1	NULL	Bryan Rolland	Mastermind_Teams
<input type="checkbox"/>	Edit Copy Delete	7	2022-05-05	Functionality Flaw	3	Fixed	10007 10007.1	Prince Maxwell	NULL	Star_Teams
<input type="checkbox"/>	Edit Copy Delete	8	2023-01-21	Performance Degradation	7	In Progress	10008 10008.1	NULL	Rio Jack	Star_Teams
<input type="checkbox"/>	Edit Copy Delete	9	2023-09-03	Installation Problem	6	Not a Problem	10009 10009.3	NULL	Christiannah Joe	Emirate_Teams
<input type="checkbox"/>	Edit Copy Delete	10	2022-12-12	Software Configuration Issue	8	In Progress	10010 10010.3	Charles Darwin	NULL	Mavericks_Teams
<input type="checkbox"/>	Edit Copy Delete	11	2022-08-18	Database Error	9	Fixed	10011 10011.1	NULL	John Kennedy	Star_Teams
<input type="checkbox"/>	Edit Copy Delete	12	2023-03-24	Security Vulnerability	5	In Progress	10012 10012.1	NULL	Rio Jack	Accurate_Teams
<input type="checkbox"/>	Edit Copy Delete	13	2022-06-30	Software Update Issue	7	Not a Problem	10013 10013.1	Goodness Rocky	NULL	Accurate_Teams
<input type="checkbox"/>	Edit Copy Delete	14	2022-03-14	Network Connectivity Problem	6	In Progress	10014 10014.1	Prince Maxwell	NULL	Mavericks_Teams
<input type="checkbox"/>	Edit Copy Delete	15	2023-07-01	User Permission Issue	8	Fixed	10015 10015.1	NULL	James Stones	Alpha_Teams
<input type="checkbox"/>	Edit Copy Delete	16	2023-05-09	Software Configuration Issue	4	In Progress	10016 10016.1	NULL	Donald Fred	Emirate_Teams
<input type="checkbox"/>	Edit Copy Delete	17	2022-10-22	Performance Issue	6	Not a Problem	10017 10017.1	Prince Maxwell	NULL	Gladiators_Teams
<input type="checkbox"/>	Edit Copy Delete	18	2022-02-27	Integration Issue	9	In Progress	10018 10018.3	NULL	Annie Jackson	Mastermind_Teams
<input type="checkbox"/>	Edit Copy Delete	19	2023-04-18	Software Crash	7	Fixed	10019 10019.1	Junior Darlington	NULL	Mavericks_Teams
<input type="checkbox"/>	Edit Copy Delete	20	2022-08-11	UI Glitch	3	In Progress	10020 10020.2	NULL	Robert Green	Star_Teams
<input type="checkbox"/>	Edit Copy Delete	21	2022-12-29	Functionality Flaw	8	Not a Problem	10021 10021.2	Francisca Joe	NULL	Ultimate_Teams
<input type="checkbox"/>	Edit Copy Delete	22	2023-06-07	Software Bug	5	In Progress	10022 10022.1	David Mark	NULL	Universal_Teams
<input type="checkbox"/>	Edit Copy Delete	23	2022-04-05	Performance Degradation	6	Fixed	10023 10023.1	NULL	James Stones	Warriors_Teams
<input type="checkbox"/>	Edit Copy Delete	24	2023-01-13	Software Configuration Issue	7	In Progress	10024 10024.2	Charles Darwin	NULL	Alpha_Teams
<input type="checkbox"/>	Edit Copy Delete	25	2023-10-30	Compatibility Error	9	Not a Problem	10025 10025.2	Charles Godwin	NULL	Accurate_Teams
<input type="checkbox"/>	Edit Copy Delete	26	2023-08-08	Performance Issue	8	In Progress	10015 10015.2	Paul Donald	NULL	Ultimate_Teams

3.2.9 Registration Entity

When an external user purchases a product, each version of that product is allocated a registration number and registration date and the reg_no is identified as the primary_key. The information is represented on this table with user_name and version_no identified foreign keys which has relationship with external_user entity and version entity respectively.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 reg_no	int(10)			No	None			Change Drop More
<input type="checkbox"/>	2 reg_date	date			No	None			Change Drop More
<input type="checkbox"/>	3 user_name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 version_no	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns									

```
SELECT * FROM `registration`
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 Filter rows: Search this table | Sort by key: None

Extra options

		reg_no	reg_date	user_name	version_no
<input type="checkbox"/>	Edit Copy Delete	101	2023-11-01	Annie Jackson	10003.2
<input type="checkbox"/>	Edit Copy Delete	102	2023-07-26	Bekky Daniels	10005.4
<input type="checkbox"/>	Edit Copy Delete	103	2023-12-25	Bryan Rolland	10025.1
<input type="checkbox"/>	Edit Copy Delete	104	2023-01-07	Christiannah Joe	10008.1
<input type="checkbox"/>	Edit Copy Delete	105	0000-00-00	Donald Fred	10007.1
<input type="checkbox"/>	Edit Copy Delete	106	2023-07-14	James Stones	10018.3
<input type="checkbox"/>	Edit Copy Delete	107	2023-03-29	John Kennedy	10010.1
<input type="checkbox"/>	Edit Copy Delete	108	2023-04-25	Kings Smith	10025.2
<input type="checkbox"/>	Edit Copy Delete	109	2023-10-09	Rio Jack	10021.3
<input type="checkbox"/>	Edit Copy Delete	110	2023-12-29	Robert Green	10019.3

☐ Check all With selected: Edit Copy Delete Export

3.3 Queries Examples

Query 1: Print the list of problem reports submitted by external users

Answer:

```
SELECT external_user.user_name AS ext_user_name, external_user.phone_no, external_user.company_name, problem_report.report_no, problem_report.submission_date, problem_report.problem_description, problem_report.priority_rating, problem_report.status, problem_report.product_id, problem_report.version_no, problem_report.team_name FROM external_user INNER JOIN problem_report ON external_user.user_name = problem_report.external_user_name;
```

```
SELECT external_user.user_name AS ext_user_name, external_user.phone_no, external_user.company_name, problem_report.report_no, problem_report.submission_date, problem_report.problem_description, problem_report.priority_rating, problem_report.status, problem_report.product_id, problem_report.version_no, problem_report.team_name FROM external_user INNER JOIN problem_report ON external_user.user_name = problem_report.external_user_name;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 | Filter rows: Sort by key: None

Extra options

ext_user_name	phone_no	company_name	report_no	submission_date	problem_description	priority_rating	status	product_id	version_no	team_name
Annie Jackson	795332131	IBM	3	2022-11-25	Software Bug	5	Fixed	10003	10003.2	Emirate_Teams
Annie Jackson	795332131	IBM	18	2022-02-27	Integration Issue	9	In Progress	10018	10018.3	Mastermind_Teams
Bekky Daniels	792553101	Salesforce	2	2023-04-12	Integration Issue	7	In Progress	10002	10002.1	Gladiators_Teams
Bryan Rolland	791082185	Amazon	6	2022-07-10	UI Glitch	4	In Progress	10006	10006.1	Mastermind_Teams
Christiannah Joe	792443105	Oracle	9	2023-09-03	Installation Problem	6	Not a Problem	10009	10009.3	Emirate_Teams
Donald Fred	795212201	Adobe	16	2023-05-09	Software Configuration Issue	4	In Progress	10016	10016.1	Emirate_Teams
James Stones	712345678	Microsoft	15	2023-07-01	User Permission Issue	8	Fixed	10015	10015.1	Alpha_Teams
James Stones	712345678	Microsoft	23	2022-04-05	Performance Degradation	6	Fixed	10023	10023.1	Warriors_Teams
John Kennedy	791356544	Intel	11	2022-08-18	Database Error	9	Fixed	10011	10011.1	Star_Teams
Rio Jack	795112065	SAP	8	2023-01-21	Performance Degradation	7	In Progress	10008	10008.1	Star_Teams
Rio Jack	795112065	SAP	12	2023-03-24	Security Vulnerability	5	In Progress	10012	10012.1	Accurate_Teams
Robert Green	791458133	Apple	20	2022-08-11	UI Glitch	3	In Progress	10020	10020.2	Star_Teams

Query 2: Print the list of problem reports submitted by internal users

Answer:

```
SELECT internal_user.user_name AS int_user_name, internal_user.phone_no, internal_user.dept_name, problem_report.report_no, problem_report.submission_date, problem_report.problem_description, problem_report.priority_rating, problem_report.status, problem_report.product_id, problem_report.version_no, problem_report.team_name FROM internal_user INNER JOIN problem_report ON internal_user.user_name = problem_report.internal_user_name;
```

```
SELECT internal_user.user_name AS int_user_name, internal_user.phone_no, internal_user.dept_name, problem_report.report_no, problem_report.submission_date, problem_report.problem_description, problem_report.priority_rating, problem_report.status, problem_report.product_id, problem_report.version_no, problem_report.team_name FROM internal_user INNER JOIN problem_report ON internal_user.user_name = problem_report.internal_user_name;
```

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: | Filter rows: | Sort by key:

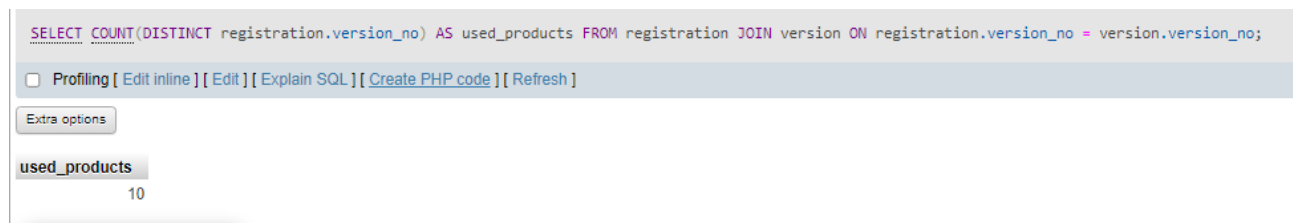
Extra options

int_user_name	phone_no	dept_name	report_no	submission_date	problem_description	priority_rating	status	product_id	version_no	team_name
Charles Darwin	793506834	Workplace Management	10	2022-12-12	Software Configuration Issue	8	In Progress	10010	10010.3	Mavericks_Teams
Charles Darwin	793506834	Workplace Management	24	2023-01-13	Software Configuration Issue	7	In Progress	10024	10024.2	Alpha_Teams
Charles Godwin	794217328	Analytics	1	2023-08-08	Performance Issue	8	In Progress	10001	10001.3	Alpha_Teams
Charles Godwin	794217328	Analytics	25	2023-10-30	Compatibility Error	9	Not a Problem	10025	10025.2	Accurate_Teams
David Mark	795644131	Development	22	2023-06-07	Software Bug	5	In Progress	10022	10022.1	Universal_Teams
Francisca Joe	791889320	Project Management	21	2022-12-29	Functionality Flaw	8	Not a Problem	10021	10021.2	Ultimate_Teams
Goodness Rocky	734723781	Quality Assurance	13	2022-06-30	Software Update Issue	7	Not a Problem	10013	10013.1	Accurate_Teams
Junior Darlington	792005173	Human Resources	19	2023-04-18	Software Crash	7	Fixed	10019	10019.1	Mavericks_Teams
Luke Cornell	793257551	Sales and Business	4	2022-09-30	Software Crash	9	In Progress	10004	10004.2	Accurate_Teams
Paul Donald	791345763	Engineering	5	2023-02-17	Compatibility Error	6	Not a Problem	10005	10005.4	Warriors_Teams
Paul Donald	791345763	Engineering	26	2023-08-08	Performance Issue	8	In Progress	10015	10015.2	Ultimate_Teams
Prince Maxwell	791226551	Data Science	7	2022-05-05	Functionality Flaw	3	Fixed	10007	10007.1	Star_Teams
Prince Maxwell	791226551	Data Science	14	2022-03-14	Network Connectivity Problem	6	In Progress	10014	10014.1	Mavericks_Teams
Prince Maxwell	791226551	Data Science	17	2022-10-22	Performance Issue	6	Not a Problem	10017	10017.1	Gladiators_Teams

Query 3: Print the total number of products used by the external users

Answer:

```
SELECT COUNT(DISTINCT registration.version_no) AS used_products FROM registration JOIN version ON registration.version_no = version.version_no;
```



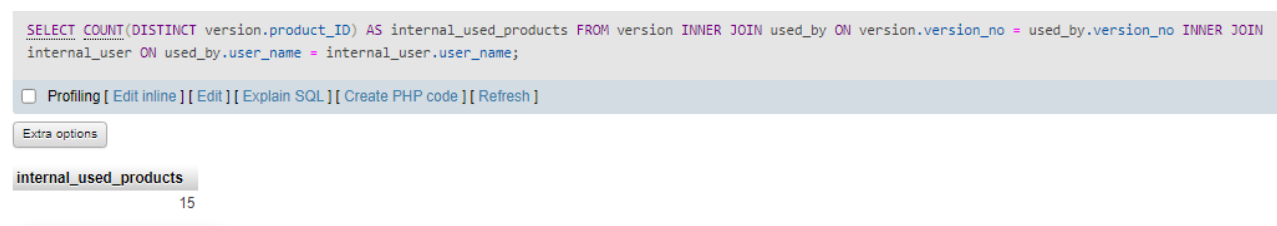
The screenshot shows a database query interface. At the top, the SQL query is displayed: `SELECT COUNT(DISTINCT registration.version_no) AS used_products FROM registration JOIN version ON registration.version_no = version.version_no;`. Below the query, there are buttons for `Profiling`, `Edit inline`, `Edit`, `Explain SQL`, `Create PHP code`, and `Refresh`. A button labeled `Extra options` is also present. The result of the query is shown in a table with the header `used_products` and a single value `10`.

used_products
10

Query 4: Print the total number of products used by internal users

Answer:

```
SELECT COUNT(DISTINCT version.product_ID) AS internal_used_products FROM version INNER JOIN used_by ON version.version_no = used_by.version_no INNER JOIN internal_user ON used_by.user_name = internal_user.user_name;
```



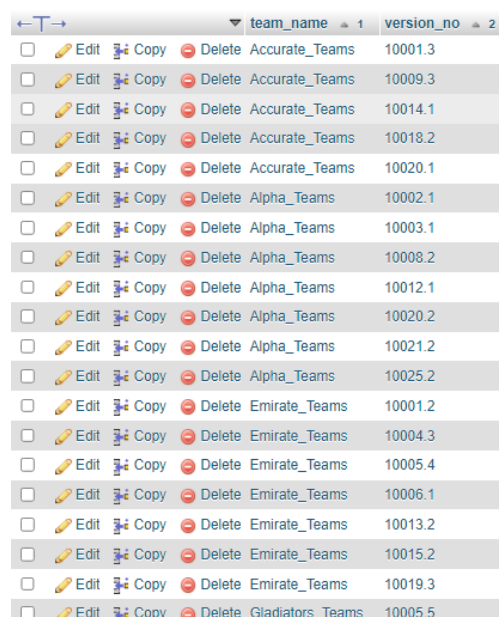
The screenshot shows a database query interface. At the top, the SQL query is displayed: `SELECT COUNT(DISTINCT version.product_ID) AS internal_used_products FROM version INNER JOIN used_by ON version.version_no = used_by.version_no INNER JOIN internal_user ON used_by.user_name = internal_user.user_name;`. Below the query, there are buttons for `Profiling`, `Edit inline`, `Edit`, `Explain SQL`, `Create PHP code`, and `Refresh`. A button labeled `Extra options` is also present. The result of the query is shown in a table with the header `internal_used_products` and a single value `15`.

internal_used_products
15

Query 5: Print the first 20 in ascending order of team name the versions maintained by team and their team names

Answer:

```
SELECT maintained_by.team_name, maintained_by.version_no FROM maintained_by ORDER BY `team_name` ASC, `version_no` ASC LIMIT 20;
```





































The screenshot shows a database query interface. At the top, the SQL query is displayed: `SELECT maintained_by.team_name, maintained_by.version_no FROM maintained_by ORDER BY `team_name` ASC, `version_no` ASC LIMIT 20;`. Below the query, there are buttons for `Edit`, `Copy`, and `Delete`. The result of the query is shown in a table with two columns: `team_name` and `version_no`. The table contains 20 rows of data, sorted by team name and then by version number.

team_name	version_no
Accurate_Teams	10001.3
Accurate_Teams	10009.3
Accurate_Teams	10014.1
Accurate_Teams	10018.2
Accurate_Teams	10020.1
Alpha_Teams	10002.1
Alpha_Teams	10003.1
Alpha_Teams	10008.2
Alpha_Teams	10012.1
Alpha_Teams	10020.2
Alpha_Teams	10021.2
Alpha_Teams	10025.2
Emirate_Teams	10001.2
Emirate_Teams	10004.3
Emirate_Teams	10005.4
Emirate_Teams	10006.1
Emirate_Teams	10013.2
Emirate_Teams	10015.2
Emirate_Teams	10019.3
Gladiators_Teams	10005.5

Query 6: Print versions of products maintained by teams that starts with A

Answer:

```
SELECT maintained_by.team_name, maintained_by.version_no FROM maintained_by WHERE maintained_by.team_name LIKE 'A%';
```

		team_name	version_no
<input type="checkbox"/>	 Edit  Copy  Delete	Accurate_Teams	10001.3
<input type="checkbox"/>	 Edit  Copy  Delete	Accurate_Teams	10009.3
<input type="checkbox"/>	 Edit  Copy  Delete	Accurate_Teams	10014.1
<input type="checkbox"/>	 Edit  Copy  Delete	Accurate_Teams	10018.2
<input type="checkbox"/>	 Edit  Copy  Delete	Accurate_Teams	10020.1
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10002.1
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10003.1
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10008.2
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10012.1
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10020.2
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10021.2
<input type="checkbox"/>	 Edit  Copy  Delete	Alpha_Teams	10025.2

Query 7: Find the names of users, company name, phone number and product description managed by team leader Annabel Fisher

Answer:

```
SELECT external_user.user_name AS name, external_user.company_name AS company, external_user.phone_no, product.product_description FROM external_user INNER JOIN problem_report ON external_user.user_name = problem_report.external_user_name INNER JOIN maintenance_team ON problem_report.team_name = maintenance_team.team_name INNER JOIN product ON problem_report.product_id = product.product_ID WHERE maintenance_team.team_leader = 'Annabel Fisher';
```

name	company	phone_no	product_description
Rio Jack	SAP	795112065	Atlassian Jira

Query 8: Find the versions of a product that are not registered Limit 15

Answer:

```
SELECT version.* FROM version LEFT JOIN registration ON version.version_no = registration.version_no WHERE registration.reg_no IS NULL;
```

version_no	release_date	product_id
10001.1	2022-12-03	10001
10001.2	2023-05-08	10001
10001.3	2023-12-12	10001
10002.1	2021-03-15	10002
10003.1	2021-06-20	10003
10004.1	2021-09-25	10004
10004.2	2022-02-28	10004
10004.3	2022-07-14	10004
10005.1	2021-11-30	10010
10005.2	2022-04-18	10005
10005.3	2022-08-22	10005
10005.5	2023-06-10	10005
10006.1	2021-04-10	10006
10007.2	2022-03-11	10007
10008.2	2023-02-19	10008
10009.1	2021-08-15	10009
10009.2	2022-01-22	10009
10009.3	2022-05-30	10009
10010.2	2022-06-14	10010
10010.3	2023-03-20	10010
10011.1	2022-09-07	10011
10011.2	2023-04-01	10011
10012.1	2022-11-12	10012
10013.1	2023-01-30	10013
10013.2	2023-07-25	10013
10014.1	2023-05-19	10014
10015.1	2023-09-28	10015

Query 9: Find the versions of a product that are registered

Answer:

```
SELECT version.* FROM version LEFT JOIN registration ON version.version_no = registration.version_no WHERE registration.reg_no IS NOT NULL;
```

version_no	release_date	product_id
10003.2	2022-01-10	10003
10005.4	2023-01-05	10005
10007.1	2021-07-02	10007
10008.1	2022-10-05	10008
10010.1	2021-12-08	10010
10018.3	2023-07-18	10018
10019.3	2023-04-10	10019
10021.3	2023-11-02	10021
10025.1	2023-04-30	10025
10025.2	2023-09-05	10025

Query 10: Print out the report id, submission date, problem description, priority rating, status, product id, version number of reports maintained by Ultimate Teams

Answer:

```
SELECT * FROM problem_report WHERE team_name = 'Ultimate_Teams';
```

report_no	submission_date	problem_description	priority_rating	status	product_id	version_no	internal_user_name	external_user_name	team_name
21	2022-12-29	Functionality Flaw	8	Not a Problem	10021	10021.2	Francisca Joe	NULL	Ultimate_Teams
26	2023-08-08	Performance Issue	8	In Progress	10015	10015.2	Paul Donald	NULL	Ultimate_Teams

4 Security Scenario

Databases are prone to various threats such as loss of confidentiality, loss of integrity, loss of availability etc.) and to protect the database from such threats, we have to implement countermeasures like access control (account creation), discretionary access control (privilege granting and revoking) and mandatory authorisation (security level assignment).

So as to secure the database, we must create various users with different controls and access to the database. The steps to achieve this are to firstly create a new role, then grant privileges to the role and lastly grant the roles to its users.

4.1 Create Roles

Step 1: This is done to easily and efficiently assign privileges to its users to improve efficiency.

```
CREATE ROLE 'admin', 'users', 'development', 'quality';
```

Step 2: Assign privileges to the roles

```
GRANT SELECT ON software_company.* TO 'admin';
GRANT SELECT, INSERT ON software_company.* TO 'users';
GRANT ALL ON software_company.* TO 'development';
GRANT ALL ON software_company.* TO 'quality';
```

4.2 Create Users

This is done for users to have different tasks and to make access to the system easy.

```
CREATE USER 'david'@'localhost' IDENTIFIED BY 'admin101';
CREATE USER 'brown'@'localhost' IDENTIFIED BY 'user101';
CREATE USER 'shawn'@'localhost' IDENTIFIED BY 'devel01';
CREATE USER 'prime'@'localhost' IDENTIFIED BY 'qual01';
```

4.3 Access and read the companies database as an admin

```
GRANT SELECT ON software_company.* TO 'david'@'localhost';
```

4.4 Access the database of registered versions and to also submit problem report as a user

```
GRANT SELECT ON problem_report.* TO 'brown'@'localhost';
GRANT INSERT ON problem_report.* TO 'brown'@'localhost';
```

4.5 For efficient enhancement/improvement in development of new products, new versions and registering products based on the problem reports submitted by users as a software developer

```
GRANT INSERT, UPDATE, DELETE ON product.* TO 'shawn'@'localhost';
GRANT INSERT, UPDATE, DELETE ON registration.* TO 'shawn'@'localhost';
GRANT INSERT, UPDATE, DELETE ON version.* TO 'shawn'@'localhost';
|
GRANT SELECT ON problem_report.* TO 'shawn'@'localhost';
```

- 4.6 For efficient maintenance of software products and their versions. To read problem reports submitted by both internal and external users as a quality assurance member

```
GRANT INSERT, UPDATE, DELETE ON maintained_by.* TO 'prime'@'localhost';
GRANT INSERT, UPDATE, DELETE ON maintenance_team.* TO 'prime'@'localhost';
GRANT INSERT, UPDATE, DELETE ON problem_report.* TO 'prime'@'localhost';

GRANT SELECT ON used_by.* TO 'prime'@'localhost';
GRANT SELECT ON external_user.* TO 'prime'@'localhost';
GRANT SELECT ON internal_user.* TO 'prime'@'localhost';
```

5 Conclusion

There are different types of databases that can be used in a software company with each of them having their unique merits and demerits. However, the most commonly used type of database the relational database due to its flexibility and simplicity is when organising the data into tables and creating relational links between tables through their primary and foreign keys.

Also, I designed the structure of the case study using an Entity Relational Diagram and a Relational Schema Mapping to better visualise the relationships and the entities created. Finally, sample data was created and used to populate the table and also ran some SQL queries to see how the database would link and work with each other. Lastly, various roles and users were created to demonstrate the accessibility and security of the database by controlling the privileges of each user.

6 References

Plattner, H. and Zeier, A. (2011). *In-Memory Data Management*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:<https://doi.org/10.1007/978-3-642-19363-7>.

Sisense. (2019). *Different Types of Databases for Modern Data Challenges* | Sisense. [online] Available at: <https://www.sisense.com/blog/different-types-of-databases-for-modern-data-challenges/>.

V Vidhya, G Jeyaram and Ishwarya, K.R. (2016). *Database management systems*. Oxford, U.K.: Alpha Science International Ltd.