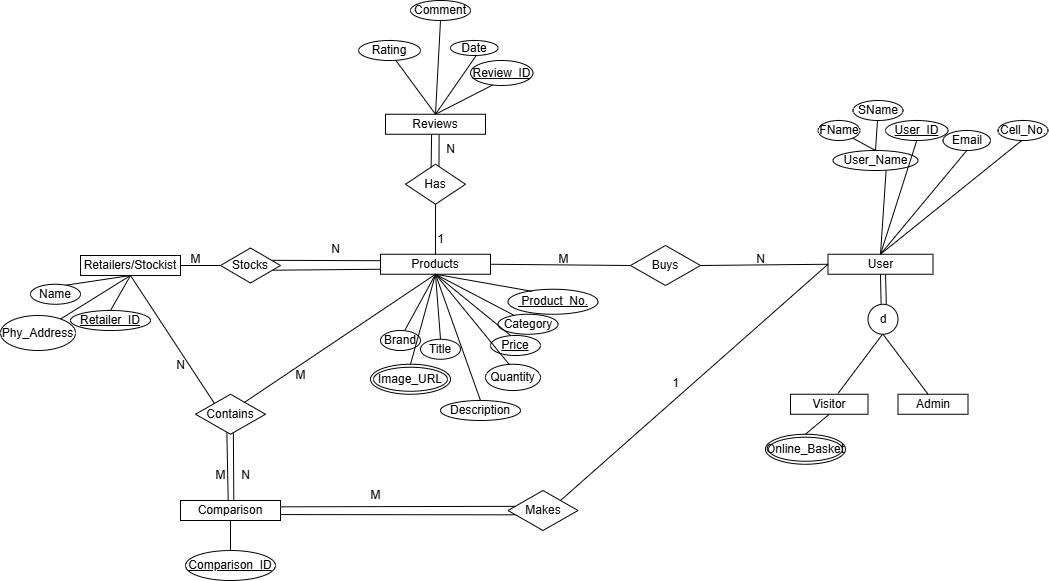
**COS 221 – Assignment 5**

**Satoshi Nakomoto**

**Github:** [**u24623131/Assignment-5**](https://github.com/u24623131/Assignment-5)

**(E)ER 1-diagram:**

**Changes to (E)ER 1-diagram:**

* Removed Buy relationship between Users and Products
* Relation between Users and Reviews added, as users make reviews
* Removed Comparison entity, as users query to make comparisons
* Replaced Online Basket with Favourites for better
* Added Admin key to Admin for differentiation (which is user specialisation)

**A diagram of a flowchart

AI-generated content may be incorrect.(E)ER 2-diagram:**

**Changes to (E)ER 2-diagram:**

* Changed Retailer/Stockist entity to Retailer for better readability
* Stocks relationship changed to Prices to better suit website purpose
* Removed Quantity attribute as that will be updated on retailer’s website
* Removed Comment attribute as we are comparing standardised products
* Added Password attribute for realism
* Price attribute moved from Products entity to Price relationship (which used to be Stocks) for easy relational mapping
* Relationship between User and Reviews changed from M:N to 1:M as users can each review can belong to exactly one user
* Removed the disjointedness between Visitor and Admin user as the user type makes disjointedness redundant
* New Favourites relationship added between Reviews and User, thus replacing Favourites multi-value attribute for easier relational mapping

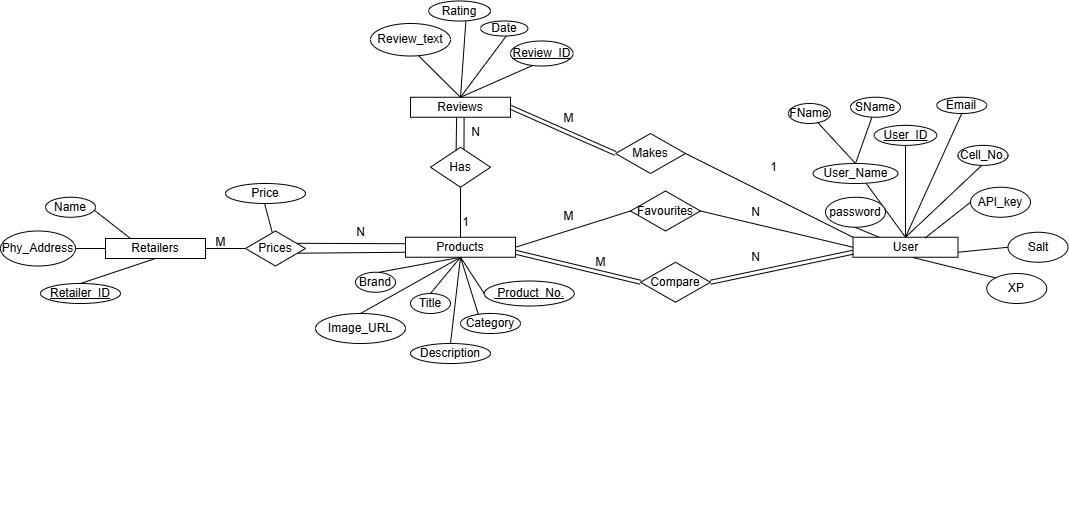
**(E)ER 3-diagram:**A diagram of a product

AI-generated content may be incorrect.

**Changes to (E)ER 3-diagram:**

* Total participation of Products relative to Favourites relationship is changed to partial participation as user is not required to favourite products
* M:N Compare relationship is added between User and Products to simulate website function
* API key is added as an attribute to User relation to uniquely identify user
* Salt is added as an attribute to User relation for security aspect of UX
* XP (experience attribute) added to User to gamify web application
* Review\_TXT is added back for helpful peer-to-peer recommendations
* Image\_URL changed from multivalued to standard attribute due to design reasons

**(E)ER 4-diagram:**



**Steps for mapping (E)ER-diagram to Relational Model:**

• Step 1: Mapping of regular (strong) entity types with standard and composite attributes

• Step 2: Mapping of binary 1:N relationships

• Step 3: Mapping of binary M:N relationships

• Step 4: Mapping of multivalued attributes

• Step 5: Mapping specialisation and generalization

• Step 6: Mapping of functional dependencies

**Assumptions made during model creation:**

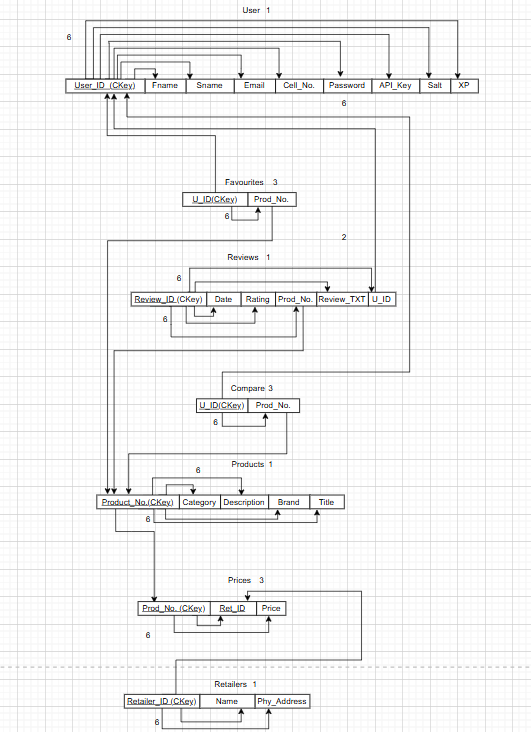
1. All identical items sold by the retailers look the same
2. The necessary entities are Users, retailers, products, reviews. Compare was initially an entity, but changed to a relationship, as comparison is an SQL query
3. Each product will have one image to prevent 1NF violation and ease design process
4. We designed relational model to avoid 1NF, 2NF, 3NF and BCDNF violations
5. No N-ary relationships were used to reduce complexity in relational mapping and database creation

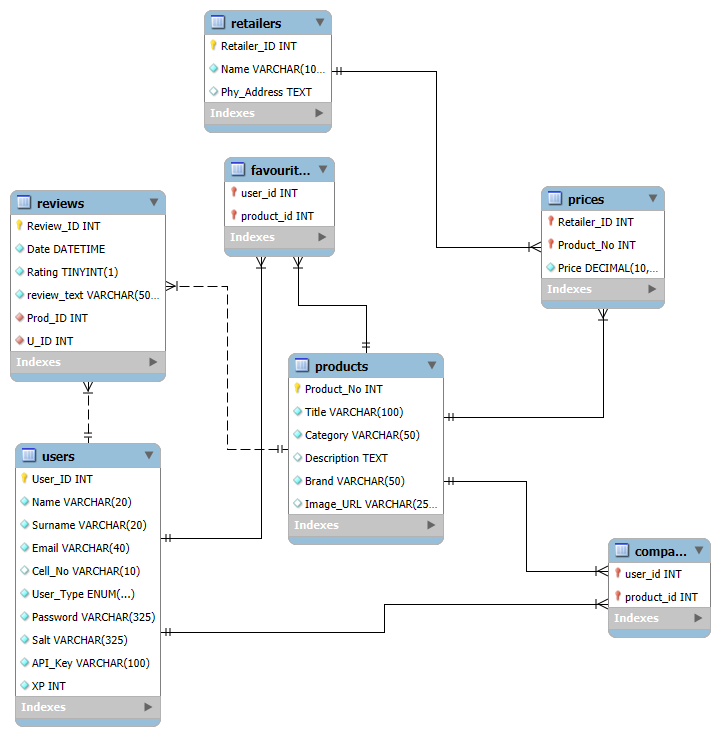
A diagram of a company

AI-generated content may be incorrect.**Relational Model (1):**

**Changes Model (1):**

* Compare table is added between User and Products
* Changed Prod\_ID and P\_ID to Prod\_No.
* API key, Salt and XP were added to User table
* Review\_TXT is added to Reviews table
* Image\_URL table is removed as Image\_URL attribute was changed from multivalued to standard attribute due to design reasons

**Relational Model (2):** 

**Task 4: Database visualization and SQL:** 

**SQL Code:**

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO"; START TRANSACTION; SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT */; /*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS */; /*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION */; /*!40101 SET NAMES utf8mb4 \*/;

-- -- Database: u24623131\_satoshi\_nakamoto\_db --

CREATE DATABASE IF NOT EXISTS u24623131\_satoshi\_nakamoto\_db DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci; USE u24623131\_satoshi\_nakamoto\_db;

-- -- Table structure for table Compare --

CREATE TABLE Compare ( user\_id int(11) NOT NULL, product\_id int(11) NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Table structure for table favourites --

CREATE TABLE favourites ( user\_id int(11) NOT NULL, product\_id int(11) NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Table structure for table Prices --

CREATE TABLE Prices ( Retailer\_ID int(11) NOT NULL, Product\_No int(11) NOT NULL, Price decimal(10,2) NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Table structure for table Products --

CREATE TABLE Products ( Product\_No int(11) NOT NULL, Title varchar(100) NOT NULL, Category varchar(50) NOT NULL, Description text DEFAULT NULL, Brand varchar(50) NOT NULL, Image\_URL varchar(255) DEFAULT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Table structure for table Retailers --

CREATE TABLE Retailers ( Retailer\_ID int(11) NOT NULL, Name varchar(100) NOT NULL, Phy\_Address text DEFAULT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Table structure for table Reviews --

CREATE TABLE Reviews ( Review\_ID int(11) NOT NULL, Date datetime NOT NULL DEFAULT current\_timestamp(), Rating tinyint(1) NOT NULL CHECK (Rating between 1 and 5), review\_text varchar(500) NOT NULL, Prod\_ID int(11) NOT NULL, U\_ID int(11) NOT NULL ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Table structure for table Users --

CREATE TABLE Users ( User\_ID int(11) NOT NULL, Name varchar(20) NOT NULL, Surname varchar(20) NOT NULL, Email varchar(40) NOT NULL, Cell\_No varchar(10) DEFAULT NULL, User\_Type enum('normal','admin') NOT NULL DEFAULT 'normal', Password varchar(325) NOT NULL, Salt varchar(325) NOT NULL, API\_Key varchar(100) NOT NULL, XP int(11) NOT NULL DEFAULT 0 COMMENT 'EXPERIENCE POINTS AUTOMATICALLY 0\r\n' ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- -- Indexes for dumped tables --

-- -- Indexes for table Compare --

ALTER TABLE Compare ADD PRIMARY KEY (user\_id,product\_id), ADD KEY product\_id (product\_id);

-- -- Indexes for table favourites --

ALTER TABLE favourites ADD PRIMARY KEY (user\_id,product\_id), ADD KEY product\_id (product\_id);

-- -- Indexes for table Prices --

ALTER TABLE Prices ADD PRIMARY KEY (Retailer\_ID,Product\_No), ADD KEY Product\_No (Product\_No);

-- -- Indexes for table Products --

ALTER TABLE Products ADD PRIMARY KEY (Product\_No);

-- -- Indexes for table Retailers --

ALTER TABLE Retailers ADD PRIMARY KEY (Retailer\_ID);

-- -- Indexes for table Reviews --

ALTER TABLE Reviews ADD PRIMARY KEY (Review\_ID), ADD KEY Prod\_ID (Prod\_ID), ADD KEY U\_ID (U\_ID);

-- -- Indexes for table Users --

ALTER TABLE Users ADD PRIMARY KEY (User\_ID), ADD UNIQUE KEY Email (Email), ADD UNIQUE KEY Password (Password), ADD UNIQUE KEY API\_Key (API\_Key);

-- -- AUTO\_INCREMENT for dumped tables --

-- -- AUTO\_INCREMENT for table Products --

ALTER TABLE Products MODIFY Product\_No int(11) NOT NULL AUTO\_INCREMENT;

-- -- AUTO\_INCREMENT for table Retailers --

ALTER TABLE Retailers MODIFY Retailer\_ID int(11) NOT NULL AUTO\_INCREMENT;

-- -- AUTO\_INCREMENT for table Reviews --

ALTER TABLE Reviews MODIFY Review\_ID int(11) NOT NULL AUTO\_INCREMENT;

-- -- AUTO\_INCREMENT for table Users --

ALTER TABLE Users MODIFY User\_ID int(11) NOT NULL AUTO\_INCREMENT;

**-- Constraints for dumped tables**

-- -- Constraints for table Compare --

ALTER TABLE Compare ADD CONSTRAINT Compare\_ibfk\_1 FOREIGN KEY (user\_id) REFERENCES Users (User\_ID) ON DELETE CASCADE, ADD CONSTRAINT Compare\_ibfk\_2 FOREIGN KEY (product\_id) REFERENCES Products (Product\_No) ON DELETE CASCADE;

-- -- Constraints for table favourites --

ALTER TABLE favourites ADD CONSTRAINT favourites\_ibfk\_1 FOREIGN KEY (user\_id) REFERENCES Users (User\_ID) ON DELETE CASCADE, ADD CONSTRAINT favourites\_ibfk\_2 FOREIGN KEY (product\_id) REFERENCES Products (Product\_No) ON DELETE CASCADE;

-- -- Constraints for table Prices --

ALTER TABLE Prices ADD CONSTRAINT Prices\_ibfk\_1 FOREIGN KEY (Retailer\_ID) REFERENCES Retailers (Retailer\_ID) ON DELETE CASCADE, ADD CONSTRAINT Prices\_ibfk\_2 FOREIGN KEY (Product\_No) REFERENCES Products (Product\_No) ON DELETE CASCADE;

-- -- Constraints for table Reviews --

ALTER TABLE Reviews ADD CONSTRAINT Reviews\_ibfk\_1 FOREIGN KEY (Prod\_ID) REFERENCES Products (Product\_No) ON DELETE CASCADE, ADD CONSTRAINT Reviews\_ibfk\_2 FOREIGN KEY (U\_ID) REFERENCES Users (User\_ID) ON DELETE CASCADE; COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT */; /*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS */; /*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

**Task 7: Analyse and Optimise**

**SQL Query Performance and Optimisation Report:**

This report analyses the performance of a SQL query that retrieves product details, prices, and retailer information using joins across three tables: Products, Prices, and Retailers. The goal is to identify inefficiencies and apply optimisations for improved performance.

**Original Query:**

SELECT Products.Product\_No, Products.Title, Products.Category,

Products.Description, Products.Brand, Products.Image\_URL,

Prices.Price, Retailers.Name AS Retailer\_Name

FROM Products

JOIN Prices ON Products.Product\_No = Prices.Product\_No

JOIN Retailers ON Prices.Retailer\_ID = Retailers.Retailer\_ID;

**Identified Issues:**

- Lack of indexes with the use of join columns, leads to inefficient lookups.

- Full table scans are performed, increasing CPU and I/O load.

- Products with Product\_No 25000001 and above do not have price entries, leading to wasted joins.

- Overall query execution is slow, especially with large datasets.

**4. Optimisation Strategy:**

**A. Indexing**

**Creating indexes on commonly joined columns to speed up lookups:**

CREATE INDEX idx\_prices\_product\_no ON Prices(Product\_No);

CREATE INDEX idx\_prices\_retailer\_id ON Prices(Retailer\_ID);

CREATE INDEX idx\_products\_product\_no ON Products(Product\_No);

CREATE INDEX idx\_retailers\_retailer\_id ON Retailers(Retailer\_ID);

**B. Optimised Query with Filtering:**

Using a WHERE clause to skip products without prices:

SELECT p.Product\_No, p.Title, p.Category, p.Description,

p.Brand, p.Image\_URL, pr.Price, r.Name AS Retailer\_Name

FROM Products p

JOIN Prices pr ON p.Product\_No = pr.Product\_No

JOIN Retailers r ON pr.Retailer\_ID = r.Retailer\_ID

WHERE p.Product\_No < 25000001;

**C. Alternative: Include All Products:**

**Using LEFT JOINs to show all products, even if they have no price:**

SELECT p.Product\_No, p.Title, p.Category, p.Description,

p.Brand, p.Image\_URL, pr.Price, r.Name AS Retailer\_Name

FROM Products p

LEFT JOIN Prices pr ON p.Product\_No = pr.Product\_No

LEFT JOIN Retailers r ON pr.Retailer\_ID = r.Retailer\_ID

ORDER BY p.Product\_No;

**Performance Improvements:**

**Expected gains after applying optimisations:**

- Execution time reduced from 2–5 seconds to under 0.5 seconds.

- Drastic reduction in rows examined due to index seek.

- Lower memory and CPU usage during execution.

- More efficient join processing.

Maintenance Recommendations:

- Regularly update table statistics:

ANALYZE TABLE Products, Prices, Retailers;

**- Monitor index usage and query performance with:**

SHOW PROFILES;

- Consider populating missing price data for consistency:

INSERT INTO Prices (Retailer\_ID, Product\_No, Price)

SELECT 1, Product\_No, 0.00

FROM Products

WHERE Product\_No >= 25000001

AND Product\_No NOT IN (SELECT DISTINCT Product\_No FROM Prices);

**Useful Query Variations**

**- To retrieve products by category:**

SELECT p.Product\_No, p.Title, pr.Price, r.Name AS Retailer\_Name

FROM Products p

JOIN Prices pr ON p.Product\_No = pr.Product\_No

JOIN Retailers r ON pr.Retailer\_ID = r.Retailer\_ID

WHERE p.Category = 'Electronics';

**- To compare prices across retailers:**

SELECT p.Product\_No, p.Title,

MIN(pr.Price) AS Min\_Price,

MAX(pr.Price) AS Max\_Price,

COUNT(pr.Retailer\_ID) AS Retailer\_Count

FROM Products p

JOIN Prices pr ON p.Product\_No = pr.Product\_No

GROUP BY p.Product\_No, p.Title;

**By applying indexes and restructuring the query, significant performance gains are achievable. These optimisations help ensure the system can scale and remain responsive as data volumes grow.**

**Developer Project Contributions:**

**Mosi (u24663370):** Mosi’s job was to implement high-level designs of the project, by creating the (E)ER-diagram, converting this to a relational model and guiding the creation of the relational models using myPHP admin. General (non-programming) tasks such as Task 1 and the creation of the management of the PDF file was also Mosi’s responsibility.

**Delleshan (u24623131):** Delleshan shared a similar role to Mosi, as they both worked on the (E)ER-diagram and relational model. However, he had a greater role to play in creating the relational models, by populating the database tables using mockaroo, and an API to specifically populate the product’s table. Delleshan’s role also extends to the creation and management of the project Github repository.

**Mutombo (u24957102) –** Mutombo contributed to the development of the compare, favorites, header, and home pages. He implemented styling using CSS, designed the layout using HTML and made the pages dynamic using JavaScript. He assisted in making the pages interactive with the database and helped update the database with new records, following established guidelines, and uploaded images to Cloudinary URLs within the database to ensure seamless integration. Furthermore, Mutombo designed and implemented the filter and search functionality, primarily focusing on the home and favorites pages, to enable efficient information retrieval.

**Paul (u24618391) –** Paul contributed to the development of the login and signup pages. He implemented and assisted in styling the header component to dynamically adjust accessible pages based on the user's authentication status and role (admin or regular user).

I developed and styled the profile page and created as well as styled the manager page in addition to a basic CSS layout for the compare page container. A footer component was created, and CSRF protection was integrated across the existing pages to enhance the application's security.

**Bandile (u24675394) –** Bandile was responsible for the backend API which was developed to support core features like account management(Login, Register, Admin functions , product handling(Add Product) , reviews, and user interactions (Add review, Update Review) . Users can add to their personal favorites page as well as Retailers insertion, I created and continually updated the api doc. Txt file and finialised the outputs after quite a time. Creation and allocation of .env file as well as git ignore files.

**Karabo (u24675394) –** Karabo Implemented the following functionality:

1. Change password

2. Retrieving all users

3. Updating product details

4. Product comparison

5. Filtering products