How Business Administrators Utilize Databases

My chosen area of business for a database system is business administration. Business administrators have a variety of reasons and motivation for a database. Problems concur in businesses that are fixable, however you need the correct direction of a proper database. These problems can range anywhere from security, integrity, consistency, organization, available accessibility and retrieval of data, growth, flexibility, compliance, and operations.

The first need of a database is data organization. Business administration deals with large volumes of data whether it be inside the organization or through customers. Organizing this data into a system enables efficient access, analyzation, and management for information. Organizing enables easier accessibility to understand roles and acquisitions that are important responsibilities for this job. More access ensures an organized representation of the tasks they need to perform. Database systems are a given for efficient data retrieval, organizations need an easily digestible set up to access information.

Businesses are made to grow, with growth, data requirements increase. Implementing a system allows flexibility for this data representation without additional performance problems. Data security, as we know, is crucial in the business realm. Creating a system that requires user authentication, limits accessibility to only appropriate areas, and actively encrypting or ciphering data are a few ways the database works to keep the company safe. It ensures integrity to maintain consistency and an accurate measure of data stored.

Keeping an organized system puts your business at competitive advantage as it improves innovation, adaptation, and efficiently keeps up with its customers. Compliance related features of these systems are in accordance with established rules and expectations.

With the use of a proper database, all of these issues can be resolved. A structured way to store and organize data formatted consistently makes it easier for administrators to access. Instead of scatterplots, spreadsheets, and other separated, less organized diagrams, a database system provides the information that is important in one location. This lessens chances of inconsistency, loss of data, and irrelevant information. It intertwines data from separate sources, such as finance, inventory, sales, or operations, and overall makes it easier for an administrator to analyze and achieve correct business performance.

It is much more simple for these employees to perform complex analyses of data, trends, key performance indicators, and business intelligence. All factors enable accurate insights for making decisions to excel the company. These systems reduce the amount of manual effort and possibilities of errors by performing updates, back ups, and keeping up with the data entries. Customer relationship management is drastically improved through databases by managing and storing customer data such as preferences, purchase history, target market, and customer feedback.

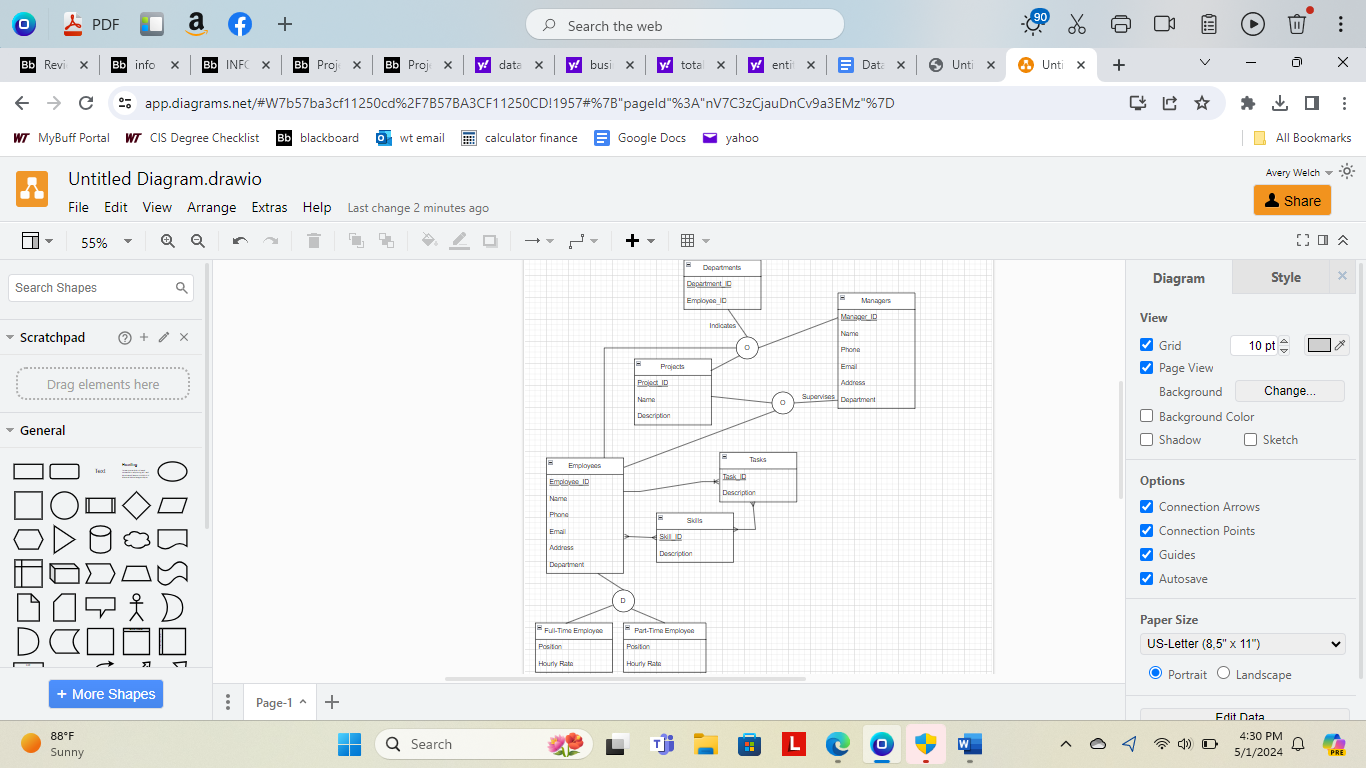
Use of this database will help future users, such as upcoming business administrators to easily digest and understand key components and operations. Overall, these systems play a mandatory role to help business administrators manage data, perform operations, and decide the route that the business needs to be successful.

I am a business owner and have decided to implement a new database system. The database developer has asked me to give a clear definition of the entities, attributes, data types of the attributes, relationships, and cardinalities. As the owner, it is important that I provide a clear description to the developer to ensure that our system meets our needs efficiently.

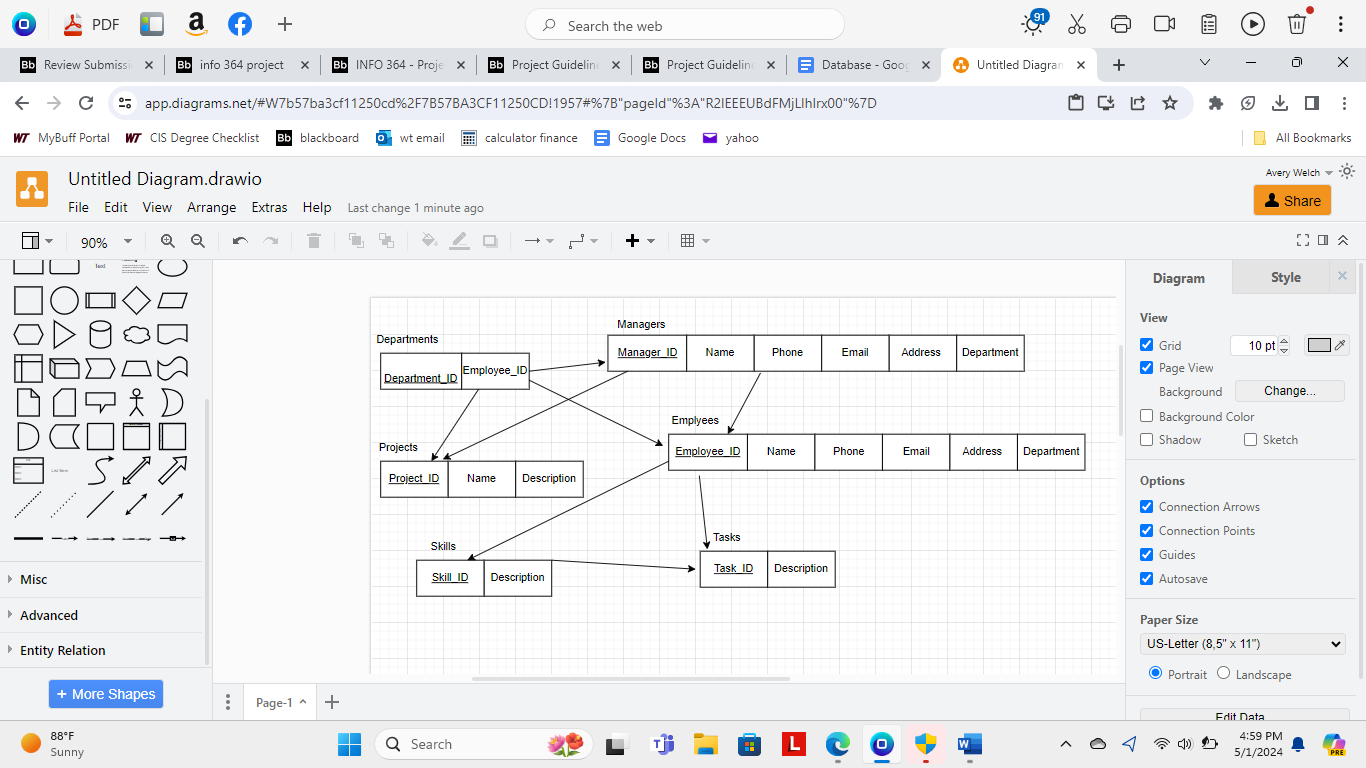
* One important entity is products, attributes include product\_ID, name, quantity, price, and description.
* Next is customers and attributes include customer\_ID, name, phone, email, address. Each customer can place none to many orders(one-to-many).
* Next is purchases or orders with attributes order\_ID, customer\_ID, order\_date, status. Each order is placed by only one customer (many-to-one). Each order consists of one or many products (many-to-many).
* employees is the next entity, attributes include employee\_ID, name, phone, email, address, Department. Each employee can handle zero to many orders (one-to-many) but an order can be handled by only one employee (many-to-one). Each employee can belong to one or many departments (many-to-many). You must include the table employee\_departments for the relationship.
* Lastly, suppliers, with attributes including supplier\_ID, name, phone, email, address. Each product is supplied by one or many suppliers (many-to-many).

Proper user requirements include the ability to manage products, update current inventory, processing customer orders, and updating order status. It includes the storage of accurate customer, and employee information. It maintains supply relationships and updates supplier information. Overall, the database needs to keep a general report on performance, inventory, customers, employees, and suppliers.

Entity Relation Diagram



Third Normal Form



SQL Script

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

EmployeeName VARCHAR(50),

DepartmentID INT,

FOREIGN KEY (DepartmentID)

REFERENCES Departments (DepartmentID)

);

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(50),

ManagerID INT,

FOREIGN KEY (ManagerID) REFERENCES Employees(EmployeeID)

);

CREATE TABLE Tasks (

TaskID INT PRIMARY KEY,

TaskName VARCHAR(50),

EmployeeID INT,

ProjectID INT,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)

);

CREATE TABLE Projects (

ProjectID INT PRIMARY KEY,

ProjectName VARCHAR(50)

);

CREATE TABLE Skills (

SkillID INT PRIMARY KEY,

SkillName VARCHAR(50)

);

CREATE TABLE EmployeeSkills (

EmployeeID INT,

SkillID INT,

PRIMARY KEY (EmployeeID, SkillID),

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

FOREIGN KEY (SkillID) REFERENCES Skills(SkillID)

);

INSERT INTO Employees (EmployeeID, EmployeeName, DepartmentID) VALUES

(1, 'Napoleon Dynamite', 1),

(2, 'Mike Myers', 2),

(3, 'John Cena', 1);

INSERT INTO Departments (DepartmentID, DepartmentName, ManagerID) VALUES

(1, 'HR', 1),

(2, 'IT', 2);

INSERT INTO Tasks (TaskID, TaskName, EmployeeID, ProjectID) VALUES

(1, 'Task 1', 1, 1),

(2, 'Task 2', 2, 2),

(3, 'Task 3', 3, 1);

INSERT INTO Projects (ProjectID, ProjectName) VALUES

(1, 'Project A'),

(2, 'Project B');

INSERT INTO Skills (SkillID, SkillName) VALUES

(1, 'Programming'),

(2, 'HR Management');

INSERT INTO EmployeeSkills (EmployeeID, SkillID) VALUES

(1, 1),

(2, 2),

(3, 1);