ADBMS PROJECT DOCUMENTATION

EMPLOYEE PAYROLL MANAGEMENT SYSTEM

USING

MICROSOFT AZURE SQL

Group – 6

Lavanya Peddireddy

Shwetha Sunkara

Siddhartha Reddy Gundluru

Yashwanth Bharadwaj Nandamuru

ADBMS PROJECT DOCUMENTATION

A Detailed description of the DBMS

Microsoft Azure SQL Database is a cloud-based relational database service built on the Microsoft SQL Server engine. It's a fully managed platform as a service (PaaS) that provides scalability, high availability, and security while handling most of the database management functions such as upgrading, patching, backups, and monitoring without user involvement.

Here is a detailed description of the service:

Characteristics of Microsoft Azure SQL Database:

1.Fully Managed Service: Azure SQL's fully managed service means you don't need to worry about routine database administration tasks. Microsoft takes care of performance tuning, backups, and high-availability setups, allowing your team to focus on application development and innovation.

2.Dynamic Scalability: With Azure SQL, you can easily scale your database resources up or down based on your application's demands. This dynamic scalability ensures your database can handle increased workloads without causing downtime, providing a seamless user experience.

3.High Availability: Azure SQL offers a 99.99% uptime SLA and handles high availability through automated data replication and failover. This minimizes the risk of service disruptions and data loss, ensuring your database is reliable and accessible.

4.Security and Compliance: Azure SQL provides advanced security measures, including network security, threat detection, access controls, and data encryption. It also complies with a wide range of regulatory standards, making it suitable for industries with stringent data protection and compliance requirements.

5.Performance Optimization: Azure SQL's built-in features like Query Performance Insights and Automatic Tuning help you optimize the performance of your queries. These tools identify and address performance issues, enhancing the efficiency of your database operations.

ADBMS PROJECT DOCUMENTATION

6.Backup and Disaster Recovery: Azure SQL automates database backups and enables you to restore your data to any point within the retention period. Additionally, geo-replication ensures data resilience in the face of unforeseen disasters, safeguarding your critical data.

7.Integration with Azure Services: Azure SQL seamlessly integrates with various Azure services, enabling you to extend your applications' functionality. This integration allows you to leverage services like Azure Logic Apps, Azure Functions, and Azure Machine Learning to enhance your applications.

8.Serverless Compute Tier: Azure SQL's serverless compute tier offers cost savings by allowing you to pay for compute resources on a per-second basis. This is particularly beneficial for workloads with fluctuating resource demands, making your database more cost-effective.

9.Advanced Data Security: Azure SQL offers robust data security with features like Always Encrypted, Transparent Data Encryption (TDE), and SQL Database Auditing. These features protect data both at rest and in transit, ensuring the confidentiality and integrity of your data.

10.Intelligent Protection: Azure Defender for SQL employs machine learning to detect unusual activities that may indicate security threats. This proactive threat protection helps you identify and respond to potential security issues, bolstering the security of your databases and applications.

Practical Uses:

- Web and Mobile Applications: Azure SQL is ideal for web and mobile apps that need a scalable, secure, and managed database service.
- Microservices Architecture: Fits well into a microservices architecture, where each microservice can have its own Azure SQL database.
- Modernization of Existing Applications: It allows organizations to modernize their existing SQL Server applications by migrating to the cloud with minimal changes.
- Data Warehousing and Big Data: It can also serve as a relational layer for big data processing or as a data source for Azure Synapse Analytics.

ADBMS PROJECT DOCUMENTATION

Azure SQL Database provides a rich environment for building robust, enterprise-grade applications. Its seamless integration with other Azure services makes it a strong contender for any organization looking to fully leverage cloud capabilities in their database management strategy.

Database Capabilities

1.Data Storage: Azure SQL allows you to store structured data in a relational database format. You can define tables with columns, data types, and constraints to organize and store your data efficiently.

2.Data Querying: You can perform data retrieval and manipulation using SQL (Structured Query Language). SQL enables you to write queries to filter, sort, join, and aggregate data, making it easy to work with your database.

3.Data Indexing: Azure SQL supports the creation of indexes, which are data structures that improve the speed of data retrieval operations. Indexes allow you to quickly locate specific rows within large datasets.

4.Data Security: Azure SQL provides robust data security features, including role-based access control to limit who can access and modify data, firewall rules to control network access, and encryption mechanisms to protect data in transit and at rest.

5.Data Backup and Recovery: Automatic backups are taken by Azure SQL, and you have the option to restore your data to a specific point in time. This is essential for recovering from accidental data changes or system failures.

6.Data Scalability: Azure SQL allows you to adjust the resources allocated to your database dynamically, ensuring that your application can handle varying workloads without downtime. You can scale up or down based on demand.

ADBMS PROJECT DOCUMENTATION

7.Query Performance Optimization: Features like Query Performance Insight and Automatic Tuning help identify and address performance bottlenecks in your database queries, ensuring efficient query execution.

8.Data Integration: Azure SQL seamlessly integrates with other Azure services, enabling you to extend your application's functionality. For example, you can trigger functions or workflows in response to database events.

9.Monitoring and Management: Azure SQL offers tools for monitoring and managing your database. You can track performance metrics, set up alerts, and manage database configurations to meet your application's needs.

10.Disaster Recovery: Geo-replication allows you to replicate your data to different Azure regions, providing disaster recovery capabilities. This helps protect your data from region-specific outages and disasters.

11.Data Compliance: Azure SQL complies with various data security and compliance standards, such as GDPR, HIPAA, and ISO 27001, making it suitable for industries with strict data protection and regulatory requirements.

12.Scalable Architectures: You can design scalable architectures using features like elastic pools, which allow you to efficiently share and manage resources across multiple databases, optimizing resource utilization.

13.Data Partitioning: Data partitioning allows you to divide large datasets into smaller, manageable segments, which improves query performance and simplifies data management.

14.Data Transformations: Azure SQL supports ETL operations, allowing you to extract, transform, and load data. This is valuable for preparing data for analysis and reporting by applying data transformations and aggregations.

ADBMS PROJECT DOCUMENTATION

These core functions collectively make Microsoft Azure SQL a versatile and powerful platform for managing relational data and are essential for building and maintaining a wide range of applications, from small-scale projects to large enterprise solutions.

A Detailed description of the KDD Nuggets referenced data

Employee

The "Employee" dataset provides insight into a company's workforce characteristics. It includes details such as their highest level of education, the date they began working for the company, the city in which they are employed, their pay grade, age, gender, and the amount of time they have been working in their current position. Since most workers hold a college degree or above, the workforce is highly educated. The fact that this data covers several years and includes workers from major Indian cities like Bengaluru, Pune, and New Delhi suggests that the company employs people in a variety of geographical areas.

We can see a little bit into the company's system of pay by looking at the different pay levels in the dataset, which correspond to the various employment rolls and revenue at the organization. We can get a sense of the diversity of individuals who work there by looking at the ages and genders of the staff. Checking out if workers have been "benched" gives us insight into the way the business assigns tasks and manages its staff. The number of years of experience a worker has in their sector may impact on their prospects for growth and compensation.

In general, the "employeegpt.xlsx" file has useful information that can assist the business in making better plans, managing its workforce more skillfully, and discovering more about the people who work for them and their career paths.

Payroll

A full set of data regarding the payroll of a company may be found in the "Payroll" dataset. Payroll figures, job departments, full-time or part-time employment status, hourly wages, annual salary expectations, and annual bonus payments are all included. The fact that this data spans multiple years and pertains to various organizations including the port authority, sanitary services, and police department raises the prospect that it was taken from official city or local government documents.

ADBMS PROJECT DOCUMENTATION

The dataset provides a detailed breakdown of employee pay, including hourly, annual, and quarterly earnings. Additionally, agreements known as Memoranda of Understanding (MOUs), which have the power to affect compensation and working conditions, are mentioned. Upon closer inspection, it becomes clear that workers receive a variety of rewards in addition to their payment, such as life, dental, and health insurance.

Links to job descriptions demonstrate how the company keeps everything structured and is informed about the specifics of the positions. Planning, determining how much money goes to employees, and making major choices about finances and employee rewards in large enterprises or government agencies all benefit from this kind of in-depth financial data.

A detailed description of the Product (Transactional or Analytical).

Depending on what the tool is intended to do, the "employeegpt.xlsx" and "payrollgpt.xlsx" datasets may be a part of a daily business tool or a tool for making strategic business choices.

Regarding Daily Transactional Business:

The tool would be referred to as transactional if it were created for daily tasks. Such a tool might be beneficial for:

Paychecks: Totaling an employee's hours worked and adding any extra bonuses to figure their pay.

Employee Records: Recording every detail about an employee, including their employment history and schooling.

Assigning Work: Deciding which workers, particularly those without tasks at the present, should focus on specific projects.
This tool would be all about making sure the day-to-day operations of the firm run smoothly and maintaining accurate records.

Regarding Analytical Decisions in Large Business:

The tool would be considered analytical if it were built with data analysis and decision-making in mind. It would be employed in:

Recognizing the Team: Examining factors such as educational background and length of service to assist in selecting candidates or organizing training.

Analyzing Pay: Establishing that a company pays enough to retain employees as well as assessing whether or not all are receiving the perks, they have a right to.

ADBMS PROJECT DOCUMENTATION

By reviewing trends and patterns in the data, an analytical tool would assist the business in seeing the wider picture and making future.

Reasons It Could Be Either:

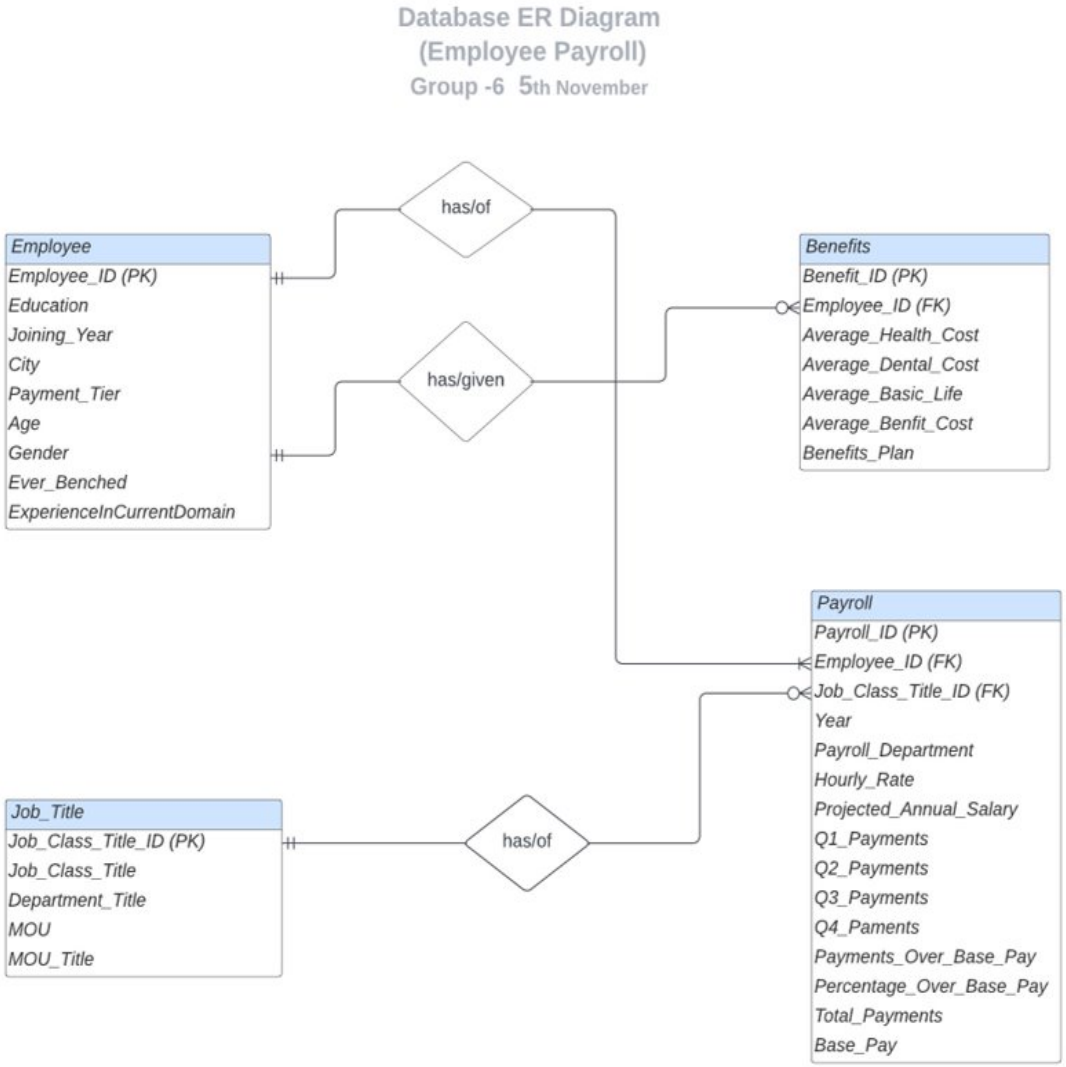
If it is for Daily Use: The datasets contain all the information needed for frequent updates, making them ideal for use as a tool for routine business operations.

If it is for Big Decisions: A decision-making tool would use the same data to analyze trends across time.

In summary, the tool has the potential to support the company in two ways: first, it could be transactional, allowing with tasks like tracking employees' salaries and hours worked; or second, it may be analytical, helping the business look at data over time and make better choices. The tool's actual application would rely on what the business needs it for: managing day-to-day operations or helping long-term strategy and planning.

ADBMS PROJECT DOCUMENTATION

A detailed description of the Product Data Structures.



The product, as shown in the Entity-Relationship Diagram (ERD), is an Employee Payroll System that manages employee payrolls and benefits.

ADBMS PROJECT DOCUMENTATION

Below is a detailed description of the data structures that comprise the system:

Entity Descriptions:

1. Employee Entity:

- Employee_ID (int, Identity, NOT NULL): This is an integer column and serves as the primary key of the "Employee" table. It cannot be NULL, ensuring that each employee has a unique identifier.
- Education (varchar(50), NULL): This column stores information about the education level of the employee. It allows NULL values, indicating that an employee's education level may not be known or specified.
- Joining_Year (int, NULL): This column stores the year in which the employee joined the company.
- City (varchar(50), NULL): This column holds the city in which the employee is located.
- Payment_Tier (int, NULL): This column stores the payment tier or level associated with the employee.
- Age (int, NULL): This column represents the age of the employee.
- Gender (varchar(50), NULL): This column stores the gender of the employee.
- Ever_Benched (varchar(50), NULL): This column indicates whether the employee has ever been benched or assigned to non-productive work periods.
- ExperienceInCurrentDomain (int, NULL): This column records the number of years of experience the employee has in their current domain or field.

This "Employee" table is structured to store information about employees, including their educational background, joining year, location, payment tier, age, gender, benching history, and domain experience. The use of NULL values in several columns allows for flexibility in recording data when certain attributes are not available.

2. Job Title Entity:

- Job_Class_Title_ID (int): This is an integer column that serves as the primary key of the "Job_Title" table.
- Job_Class_Title (varchar(200)): This column stores the job class title, which typically represents the name or designation of a job or position.
- Department_Title (varchar(200)): This column holds information about the department or division associated with the job title.
- MOU (varchar(200), NULL): This column is for storing details related to a Memorandum of Understanding (MOU) related to the job title.
- MOU_Title (varchar(200), NULL): This column may store the title or name of a specific MOU related to the job title.

ADBMS PROJECT DOCUMENTATION

3. Payroll Entity:

- Payroll_ID (varchar(200)): This is a character column that serves as the primary key of the "Payroll" table.
- Employee_ID (int, NOT NULL): This column represents the identifier of the employee associated with the payroll entry.
- Job_Class_Title_ID (int, NOT NULL): This column indicates the identifier of the job class title related to the payroll entry.
- Year (int, NULL): This column records the year for which the payroll information is related.
- Payroll_Department (varchar(200), NULL): This column may store information about the department responsible for the payroll.
- Hourly_Rate (varchar(200), NULL): This column may represent the hourly rate associated with the payroll entry.
- Projected_Annual_Salary (varchar(200), NULL): This column may hold the projected annual salary for the employee.
- Q1_Payments, Q2_Payments, Q3_Payments, Q4_Payments (varchar(200), NULL): These columns may store the payments made to the employee during the four quarters of the year.
- Payments_Over_Base_Pay (varchar(200), NULL): This column may record payments made over and above the employee's base pay.
- Percentage_Over_Base_Pay (varchar(200), NULL): This column may store the percentage by which payments exceed the base pay.
- Total_Payments (varchar(200), NULL): This column may represent the total payments made to the employee.
- Base_Pay (varchar(200), NULL): This column may indicate the employee's base pay.

The "Payroll" table is structured to store payroll-related information for employees, including their identifiers, job titles, year, department, payments, and base pay, with some columns allowing for flexibility in data entry by allowing NULL values.

4. Benefits Entity:

- Benefit_ID (int, Identity, NOT NULL): This is an integer column that serves as the primary key of the "Benefits" table.
- Employee_ID (int, NOT NULL): This column represents the identifier of the employee associated with the benefits entry.
- Average_Health_Cost (money, NULL): This column may store the average cost of health-related benefits for the employee.
- Average_Dental_Cost (money, NULL): This column may represent the average cost of dental-related benefits for the employee.

- Average_Basic_Life (money, NULL): This column may record the average cost of basic life insurance benefits for the employee.
- Average_Benefit_Cost (money, NULL): This column may store the average total benefit cost for the employee, which could include health, dental, and basic life costs.
- Benefits_Plan (varchar(200), NULL): This column may indicate the benefits plan or package associated with the employee.

The "Benefits" table is structured to store information about employee benefits, including costs related to health, dental, basic life insurance, and the benefits plan. The primary key constraint ensures the uniqueness of benefit identifiers in the table.

Data Structure Characteristics:

The data structure for the "Employee Payroll System" is designed to reflect a relational database model, exhibiting characteristics that are tailored to meet the specific needs of payroll management within an organization.

Relational Model: The use of primary keys (e.g., "Payroll_ID," "Employee_ID") and the potential for foreign keys (e.g., "Employee_ID" and "Job_Class_Title_ID") indicates a relational database model. This model is ideal for managing complex relationships between different aspects of payroll data, such as employees, job titles, and payment information.

Normalization: The database structure is likely normalized to minimize data redundancy and maintain data integrity. Normalization is crucial in a payroll system to ensure that information about employees, their job titles, and payments is consistent and accurate. This reduces the risk of errors and inconsistencies in payroll calculations.

Operational Efficiency: The focus on individual employee transactions, including details like "Hourly_Rate," "Q1_Payments," and "Base_Pay," suggests that the system is optimized for operational efficiency. It allows for the efficient tracking of payment details, facilitating the processing of employee salaries, bonuses, and deductions.

Transaction Handling: The structure is designed to handle a high volume of payroll transactions, which is essential for businesses with a large workforce. It can efficiently manage individual payment records, making it well-suited for tasks like calculating and disbursing salaries.

CRUD Operations: The presence of attributes like "Projected_Annual_Salary," "Payments_Over_Base_Pay," and "Total_Payments" indicates a focus on facilitating Create, Read, Update, and Delete (CRUD) operations. These are essential for managing and maintaining employee payroll records, making the system user-friendly for day-to-day payroll activities.

ADBMS PROJECT DOCUMENTATION

In summary, the way data is organized in the "Employee Payroll System" is planned very thoughtfully to make sure employee payments are correct and are handled efficiently and quickly. They use a system that keeps all the data connected and avoids mistakes. This is important for making sure employees get paid the right amount on time. It's like having a well-organized filing system that ensures everything is in the right place, so you can easily figure out what each person should be paid.

Relationship Descriptions:

In Entity-Relationship Diagram (ERD), cardinality and modality are used to describe the relationships between entities.

Cardinality:

Cardinality defines the number of instances of one entity that can be associated with the number of instances of another entity.

Modality:

Modality indicates whether the presence of an entity in a relationship is mandatory or optional.

- Mandatory: The relationship must have an associated entity. For example, in the account-transaction relationship, every transaction must be linked to an account.
- Optional: The entity may or may not have an associated entity in the relationship. For example, a client might not have any accounts, which would make the client's side of the client-account relationship optional.

To determine the cardinality and modality of each relationship in your database, we need to analyze the tables and their associations. Below are the cardinality and modality for the relationships based on the table descriptions you provided:

Employee to Payroll:

- Cardinality: One-to-Many (1:N)
- Modality:

  For Every Payroll generated there is an employee associated with it. So the Modality here is Mandatory (1).

Job Title to Payroll:

- Cardinality: One-to-Many (1:N)
- Modality:

For Every Job Title, it is not mandatory to have a Payroll. So, the Modality here is Optional (0).

Employee to Benefits:

- Cardinality: One-to-Many (1:N)
- Modality:

  Every Employee may not have Benefits. So the Modality here is Optional (0).

These relationships are based on the understanding that:

Each employee can have multiple payroll entries (1:N).

Each Job Title can have multiple Payroll entries (1:N).

Each employee can have multiple benefit entries (1:N).

A detailed description of the CRUD Operations.

CRUD operations (Create, Read, Update, Delete) can be performed on the four entities ("Employee," "Job Title," "Payroll," and "Benefits") in the following ways:

Employee Entity:

- Create (C): Add new employee records to the database, specifying details such as education, joining year, city, payment tier, age, gender, benching history, and experience in the current domain.
- Read (R): Retrieve employee information, including attributes like education, joining year, city, payment tier, age, gender, benching history, and experience.
- Update (U): Modify employee details, such as updating their education, joining year, city, or other attributes.
- Delete (D): Remove employee records from the database when they are no longer employed.

Job_Title Entity:

- Create (C): Add new job titles to the database, specifying the job class title, department title, and any associated Memorandum of Understanding (MOU) details.

ADBMS PROJECT DOCUMENTATION

- Read (R): Retrieve job title information, including job class title, department title, and MOU details.
- Update (U): Modify job title records, such as updating the department title or MOU information.
- Delete (D): Remove job titles from the database if they become obsolete.

Payroll Entity:

- Create (C): Add new payroll entries for employees, specifying details like the year, payroll department, hourly rate, quarterly payments, payments over base pay, percentage over base pay, total payments, and base pay.
- Read (R): Retrieve payroll information for employees, including all payment-related attributes, to calculate salaries, bonuses, and deductions.
- Update (U): Modify payroll records, such as updating the hourly rate, quarterly payments, or other payment details.
- Delete (D): Remove payroll entries if errors are detected or if they are no longer needed.

Benefits Entity:

- Create (C): Add new benefit entries for employees, specifying details like average health cost, dental cost, basic life insurance cost, average benefit cost, and the benefits plan associated with each employee.
- Read (R): Retrieve benefit information, including all cost-related attributes and the benefits plan, for employee compensation and reporting.
- Update (U): Modify benefit records, such as updating the average health cost, dental cost, or benefits plan details.
- Delete (D): Remove benefit entries if errors are detected or if they are no longer relevant for an employee.

These CRUD operations are essential for managing and maintaining data related to employees, job titles, payroll, and benefits in an organization. They allow for data entry, retrieval, updates, and removal as necessary to keep the database accurate and up to date.