**COMM108 Data Systems - Coursework**

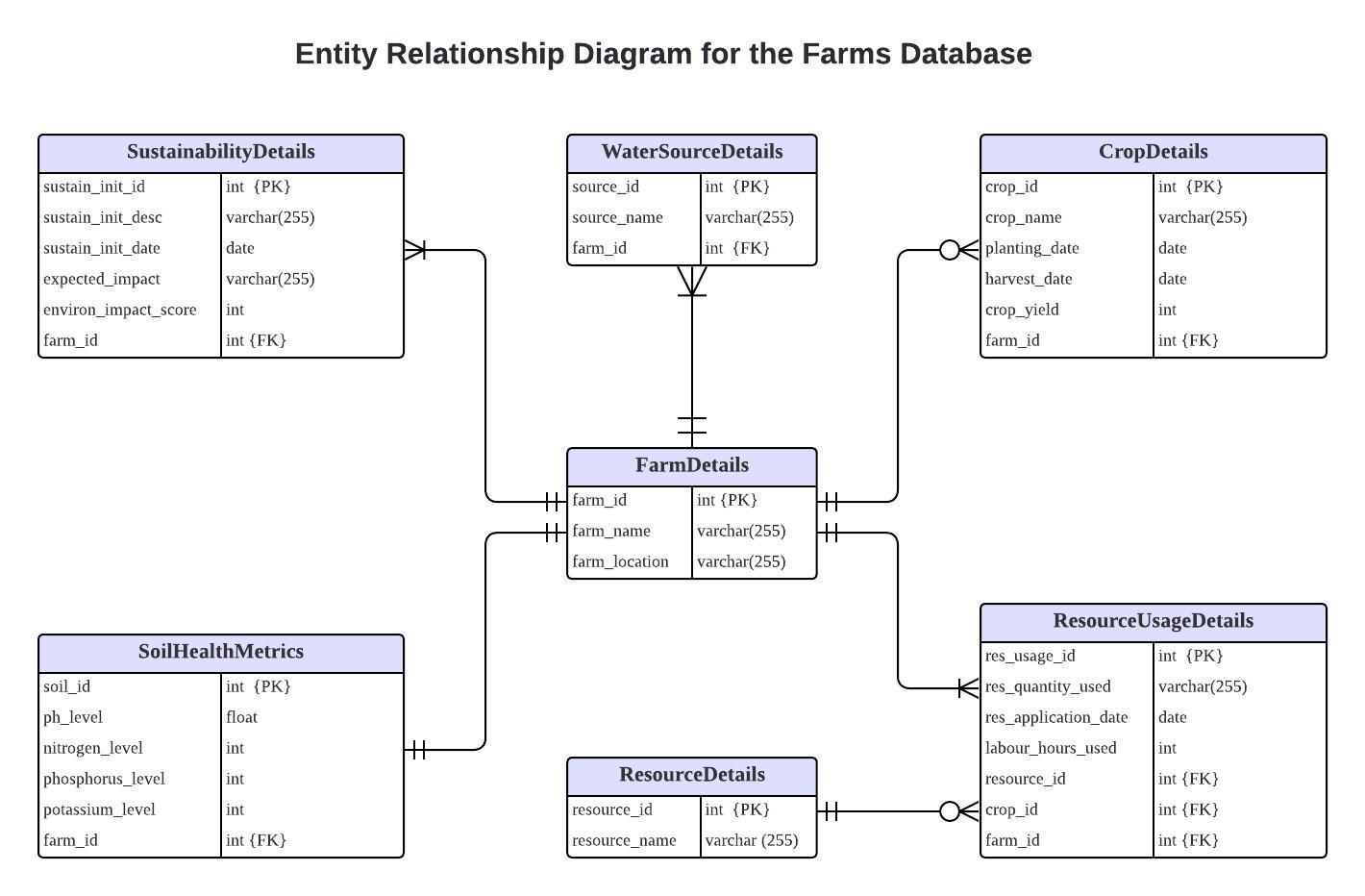
**Student Id :740003228 MSc Data Science Academic Year:2024/25**

**Introduction:**

In the UK, agriculture is a significant contributor to the economy, but it also plays a role in the country’s energy consumption and environmental impact. As part of a broader commitment to sustainability and reducing carbon emissions, farmers are increasingly adopting innovative practices aimed at enhancing energy efficiency and minimizing environmental harm. In this coursework, we will be creating a MySQL database project consisting of entities and various relationships between them along alternate database designs discussing the advantages and disadvantages of both.

**Relational Database Design for UK Farm Management System:**

Let’s design a relational database named “farms” for the given farms data in the UK. Below is the Entity Relationship diagram for the design with the explanation for each entity as follows.



|  |  |
| --- | --- |
| **Entity** | **Attributes** |
| ***FarmDetails:***  This entity contains basic descriptive and identifying information for farms | *farm\_id* (PK): A unique identifier for each farm.  *farm\_name:* The name of the farm.  *farm\_location:* The geographic location of the farm. |
| ***CropDetails:***  This entity contains attributes that capture the essential information about individual crops. | *crop\_id* (PK): A unique identifier for each crop.  *crop\_name:* The name of the crop.  *planting\_date*: The date the crop was planted.  *harvest\_date*: The date the crop is harvested.  *crop\_yield*: The total yield of the crop in specific units.  *farm\_id* (FK): Links the crop with a specific farm. |
| ***SoilHealthMetrics:***  This entity captures the essential properties and metrics that define the quality of soil on a farm. | *soil\_id* (PK): A unique identifier for each soil record.  *ph\_level*: The pH level of the soil.  *nitrogen\_level*: The nitrogen content in the soil.  *phosphorus\_level*: The phosphorus content in the soil.  *potassium\_level*: The potassium content in the soil.  *farm\_id* (FK): Associates soil data with a specific farm. |
| ***WaterSourceDetails:***  This entity stores information about water sources used on farms. | *source\_id* (PK): A unique identifier for each water source.  *source\_name*: The name of the water source.  *farm\_id* (FK): Links to the farm\_details table |
| ***SustainabilityDetails:***  This entity is designed to track eco-friendly and sustainable farming practices implemented on various farms in the UK undertaken to improve environmental impact. | *sustain\_init\_id* (PK): A unique identifier for each sustainability initiative.  *sustain\_init\_name*: The name of the sustainability initiative.  *sustain\_init\_date*: The date when the initiative was started.  *expected\_impact*: A description of the expected outcome of the initiative.  *environ\_impact\_score*: A numeric score representing the environmental impact of the initiative.  *farm\_id* (FK): Links the initiative to a specific farm. |
| ***ResourceDetails:***  This entity is designed to catalog resources used in farm operations. | *resource\_id* (PK): A unique identifier for each resource.  *resource\_name*: The name of the resource. |
| ***ResourceUsageDetails:***  This entity records the usage of resources (e.g., fertilizers, water) for farm operations. | *res\_usage\_id* (PK): A unique identifier for each resource usage record.  *res\_quantity\_used*: The quantity of the resource used.  *res\_application\_date*: The date the resource was applied.  *labour\_hours\_used*: The number of labor hours used for the resource application.  *resource\_id* (FK): Links to the resource\_details table.  *crop\_id* (FK): Links to the crop\_details table.  *farm\_id* (FK): Links to the farm\_details table. |

**Association between entities:**

1. ***FarmDetails and CropDetails*:**

A farm can have zero or many crops, while each crop is linked to a specific farm via farm\_id.

**Cardinality:** *FarmDetails*: Zero to Many (0 .. \*), *CropDetails*: One to one (1 .. 1)

**Rationale:** Farms may grow multiple crops at different times or have resting periods with no crops, so each farm can have zero or many crops, while each crop is linked to one farm.

1. ***FarmDetails and SoilHealthMetrics*:**

Each row in the FarmDetails table corresponds to one row in SoilHealthMetrics, linked by farm\_id, if testing is completed.

**Cardinality:** *FarmDetails*: One to one (1 .. 1), *SoilHealthMetrics*: One to one (1 .. 1)

**Rationale:** Each farm can have one or more soil profiles. In this example, we assume each farm has one, with its own corresponding soil health metrics.

1. ***FarmDetails and WaterSourceDetails*:**

Each farm in FarmDetails can have multiple water sources in WaterSourceDetails, with each source linked to a farm via farm\_id.

**Cardinality:** *FarmDetails*: One to Many (1 .. \*), *WaterSourceDetails*: One to one (1 .. 1)

**Rationale:** Each farm must have at least one water source, as water is essential for plant growth. Thus, each farm can have multiple water sources, but each water source is linked to only one farm.

1. ***FarmDetails and SustainabilityDetails*:**

Each farm in FarmDetails can have multiple sustainability initiatives in SustainabilityDetails, linked via farm\_id.

**Cardinality:** *FarmDetails*: One to many (1 .. \*), *SustainabilityDetails*: One to one (1 ..1)

**Rationale:** Each farm should implement sustainability initiatives, such as organic farming, to boost crop yield. Thus, each farm can have multiple initiatives, each linked to an environmental impact score.

1. ***ResourceDetails and ResourceUsageDetails*:**

Each resource in ResourceDetails can have multiple usage details in ResourceUsageDetails, linked via resource\_id.

**Cardinality:** *ResourceDetails*: Zero to Many (0 .. \*), *ResourceUsageDetails* : One to one (1..1)

**Rationale:** Each resource, such as water, may be stored in reservoirs but not necessarily used for crop planting. Each resource can have multiple usage details, with each usage linked to a resource.

1. ***FarmDetails and ResourceUsageDetails*:**

Each farm in FarmDetails must have at least one usage detail in ResourceUsageDetails, linked via farm\_id.

**Cardinality:** *FarmDetails* :One to Many (1 .. \*), *ResourceUsageDetails* : One to one (1..1)

**Rationale:** Each farm must use one or more energy sources for crop planting. Thus, each farm has multiple usage details, with each linked to a specific resource and source.

1. ***CropDetails and ResourceUsageDetails:***

Each crop in CropDetails must have at least one usage detail in ResourceUsageDetails, linked via crop\_id.

**Cardinality:** *CropDetails*:One to Many (1 .. \*), *ResourceUsageDetails* : One to one (1..1)

**Rationale:** Each crop in the farm requires one or more energy sources for harvesting. Thus, each crop has multiple resource usage details, linked to the specific crop in a specific farm.

***RESTFUL API DESIGN for the designed MySQL Database:***

To design a RESTful API for this system, the following key endpoints and HTTP methods is used for each table created as part of the relational database model designed already.

***FarmDetails:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add a new farm | /farm | POST | farmId, farmName, farmLocation |
| Retrieve the details of all farms | /farm | GET |  |
| Retrieve a specific farm | /farm/{farmId} | GET | farmId |
| Update a specific farm detail | /farm/{farmId} | PUT | farmId, farmName, farmLocation |
| Delete a farm | /farm/{farmId} | DELETE | farmId |

***CropDetails:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add a new crop | /crop | POST | cropId, cropName, plantingDate, harvestDate, cropYield, farmId |
| Retrieve the details of all crops | /crop | GET |  |
| Retrieve a specific crop | /crop/{cropId} | GET | cropId |
| Update a specific crop detail | /crop/{cropId} | PUT | cropId, cropName, plantingDate, harvestDate, cropYield, farmId |
| Delete a crop | /crop/{cropId} | DELETE | cropId |

***SoilHealthMetrics:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add new soil health metrics for a farm | /soilHealthMetrics | POST | soilId,phLevel,nitrogenLevel, phosphorusLevel, potassiumLevel,farmId |
| Retrieve the details of all soil health metrics | /soilHealthMetrics | GET |  |
| Retrieve a specific soil health metric for a farm | /soilHealthMetrics/{soilId} | GET | soilId |
| Update a specific soil health metrics detail | /soilHealthMetrics/{soilId} | PUT | soilId,phLevel,nitrogenLevel, phosphorusLevel, potassiumLevel, farmId |
| Delete a soil health metric | /soilHealthMetrics/{soilId} | DELETE | soilId |

***SustainabilityDetails:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add new sustainability initiative for a farm | /sustainabilityInitiative | POST | sustainInitId,sustainInitName,sustainInitDate, expectedImpact, environImpactScore, farmId |
| Retrieve the details of all sustainability initiatives | /sustainabilityInitiative | GET |  |
| Retrieve a specific sustainability initiative for a farm | /sustainabilityInitiative/{sustainInitId} | GET | sustainInitId |
| Update a specific sustainability initiative detail | /sustainabilityInitiative/{sustainInitId} | PUT | sustainInitId,sustainInitName,sustainInitDate, expectedImpact, environImpactScore, farmId |
| Delete a sustainability initiative | /sustainabilityInitiative/{sustainInitId} | DELETE | sustainInitId |

***WaterSourceDetails:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add new water source for a farm | /waterSource | POST | sourceId, sourceName, formId |
| Retrieve the details of water sources | /waterSource | GET |  |
| Retrieve a specific waterSource for a farm | /waterSource/{sourceId} | GET | sourceId |
| Update a specific waterSource detail | /waterSource/{sourceId} | PUT | sourceId, sourceName, formId |
| Delete a water source | /waterSource/{sourceId} | DELETE | sourceId |

***ResourceDetails:***

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add a new resource | /resource | POST | resourceId, resourceName |
| Retrieve the details of all resources | /resource | GET |  |
| Retrieve a specific resource | /resource/{resourceId} | GET | resourceId, resourceName |
| Update a specific resource detail | /resource/{resourceId} | PUT | resourceId, resourceName |
| Delete a resource | /resource/{resourceId} | DELETE | resourceId, resourceName |

**ResourceUsageDetails:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Endpoint** | **Method** | **Parameters** |
| Add a new resource usage | /resourceUsage | POST | usageId,quantityUsed, applicationDate, labourHours, resourceId, farmId, cropId |
| Retrieve the details of all resource usage | /resourceUsage | GET |  |
| Retrieve a specific resource usage for a farm | /resourceUsage/{usageId} | GET | usageId |
| Update a specific resource usage detail | /resourceUsage/{usageId} | PUT | usageId,quantityUsed, applicationDate, labourHours, resourceId, farmId, cropId |
| Delete a resource usage detail | /resourceUsage/{usageId} | DELETE | usageId |

***Alternate Document based database model for the designed relational model:***

For the designed relational Farms database above, we are going to design an alternate document-based database model. Below is the sample document in a Farms collection in the document-based database like MongoDB. In this below design, we have embedded the crop details, resource usage and sustainability details inside the same farm document. Attached as ***farms.json*** file in the zip.

{"\_id": "673dad31472c989751e38b27",

"farm\_name": "South Farm",

"farm\_location": "Kent",

"crops": [**{**

"crop\_id": "673dad31472c989751e38b28",

"crop\_name": "Wheat",

"planting\_date": "2023-03-15T00:00:00.000Z",

"harvest\_date": "2023-08-15T00:00:00.000Z",

"crop\_yield": 3000,

"resource\_usage": [**{**

"resource\_name": "Water",

"quantity\_used": 1000,

"application\_date": "2023-10-03T00:00:00.000Z",

"labour\_hours\_used": 150

**}]}**,

**{**"crop\_id": "673dad31472c989751e38b29",

"crop\_name": "Barley",

"planting\_date": "2023-03-16T16:00:00.000Z",

"harvest\_date": "2023-08-20T16:00:00.000Z",

"crop\_yield": 2800,

"resource\_usage": [**{**

"resource\_name": "Fertilizer",

"quantity\_used": 200,

"application\_date": "2023-03-12T00:00:00.000Z",

"labour\_hours\_used": 120

**}]}**],

"soil\_metrics": {

"ph\_level": 6.5,

"nitrogen\_level": 50,

"phosphorus\_level": 20,

"potassium\_level": 180 },

"farm\_water\_source": ["River"],

"sustainability\_initiatives": [**{**

"initiative\_id": "673dad31472c989751e38b33",

"initiative\_desc": "Organic Farming",

"initiated\_date": "2023-01-01T00:00:00.000Z",

"expected\_impact": "Increase in yield",

"environ\_impact\_score": 4

**}**]}

**Advantages of Relational Database Approach:**

* Data is structured using tables and views with required columns and tuple rows.
* Data integrity can be enforced thorough constraints relationships.
* Redundant data is avoided by using normalization and updates are consistent across the database.
* Complex queries can be used against relational model using joins, aggregations.

**Disadvantages of Relational Database Approach:**

* Schema is fixed and not flexible. If we need to add a new column to the tables, we need to take backup of the data and make changes to tables, views.
* Horizontal scaling (distributing data across multiple servers) is more challenging due to its ACID properties and data dependency between tables.

**Advantages of Document based Database Approach:**

* Document based database like MongoDB supports a flexible schema allowing to add new attributes without altering the structure of the existing documents
* Data retrieval is faster in a single query reducing the overhead with the embedded documents. Thus, entire data can be retrieved without complex queries or joins.
* Data can be distributed across multiple servers which is horizontal scaling. Thus, making it well-suited for large-scale systems if the data increase over time.

**Disadvantages of Relational Database Approach:**

* Since the data is being embedded in a single document, there is a chance for data duplication which results in redundant data.
* Document based database doesn’t use any standardized SQL structure for complex operations like analytics or joins between collections.
* Without constraints and relationships increases the risk of inconsistencies.
* If the data within a single document keep growing, it might result in a document size limit (16 MB) issue.