Week 4 - Conceptual Data Modelling and Logical Modelling

Tables

Relation/Table

- A relation denotes a titled, two-dimensional data table.
- A table is comprised of rows (records) and columns (attributes or fields)

Necessities for a table to meet the criteria of being a relation:

- *Must* possesses a distinct name
- Each attribute value *must be indivisible* (not multi-valued or composite)
- Every row must be distinct (*cannot contain two identical rows with all fields having the same values*)
- Attributes (*columns*) within tables *must hold* unique designations.
- *Sequence of columns must not matter*
- *Sequence of rows must not matter*

Connection to the ER Model

- *Relations* (tables) correspond to *entities*
- *Rows* correspond to *values*
- *Columns* correspond to *attributes*.

Key Fields

Primary Keys

- Ensure that each row in the table has a distinct and non-repeating identifier.
- Primary keys serve as the foundation for establishing relationships between tables.
- *The primary key could be the "employee number" field, which guarantees that every employee has a unique identifier.*

Foreign Key

- Identifiers within a table that establish a connection between two related tables in a database.
- They enable a table to reference the primary key of another table.
- *in a scenario where you have an "orders" table and a "customers" table, the "customer ID" in the "orders" table can be a foreign key, linking each order to a specific customer in the "customers" table.*

Types of Keys

Can be simple or composite:

- **Simple:** Single field that uniquely identifies a record
- **Composite:** Involves multiple input fields to form a single unique identifier. Useful when a single field may not guarantee uniqueness but a combination does.

Indexes and Query Optimisations

- Keys are often used as indexes in a database
- *An index is a data structure that enhances the speed of data retrieval by creating a quick reference to the location of data in a table*
- allow the database engine to locate relevant data more efficiently

Integrity Constraints

- This principle asserts that no primary key attribute can remain null; *every primary key field must contain valid data*.
- By enforcing entity integrity, we ensure that primary key values are present and distinct for every row.
- This uniqueness is essential for preventing duplicate or incomplete data entries and maintaining the overall integrity of the table.

Referential Integrity

- governs the relationships between tables
- Ensures the accuracy and consistency of connections between related tables.
- *Referential integrity states that any foreign key value in the "dependent" table (often referred to as the "many" side) must match a primary key value in the "parent" table (the "one" side) of the relationship.*
- Alternatively, the foreign key can be left null to indicate the absence of a relationship.

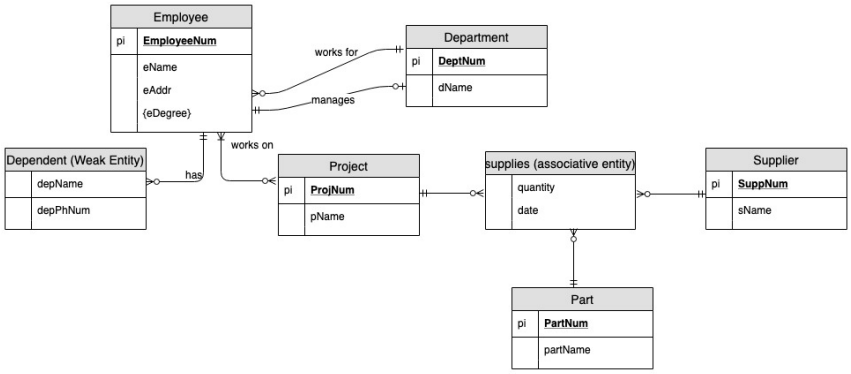
Delete Rules

Guidelines for handling the deletion of records in tables that have relationships with other tables. There are three common ones, these can vary by application

1. **Restrict:** Prevents the deletion of a record on the "parent" side if related records exist on the "dependent" side. It ensures that no record can be deleted if it's connected to other records.
2. **Cascade:** Automates the deletion process by removing all related records on the "dependent" side when a record on the "parent" side is deleted. Keeps the database consistent by removing records that rely on deleted data.
3. **Set-to-Null:** Sets the foreign key values in the "dependent" table to null when a record on the "parent" side is deleted. ***It's important to note that this rule is typically not suitable for weak entities, as it can lead to data integrity problems.***

Transforming ER Diagrams into Relations

There is an 7-step process to transform ER diagrams into tables/relations #todo Question 1: Convert this ER diagram into relations E-R to Relational Mapping will be explained here step by step.

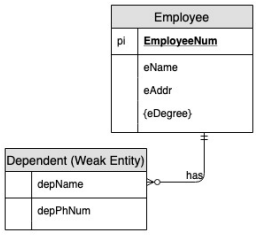


Step1: Strong entities

For each strong entity in the ER model, create a relation (i.e. a table that includes all the simple attributes). Make sure to identify the primary key for the relation (i.e. the PI of the entity becomes the PK of the table). If there is a composite attribute, you can expand them. Leave multi-valued attributes out (they will be dealt with later.)

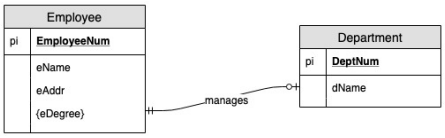
Step 2: Weak Entities

STEP 2: For each weak entity in the ER model, create a relation that includes all the simple attributes. The primary key of the relation is the combination of the primary key/s of the 'owner' and the main attribute of the weak entity itself.

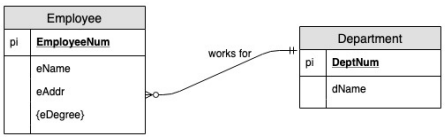


Step 3: 1:1 relationship

For each 1 TO 1 Relationship identify the two relations corresponding to the entities participating in the relationship. Choose the PK of the Relation (usually the one with mandatory constraint) and make it as the foreign key of the other relation.

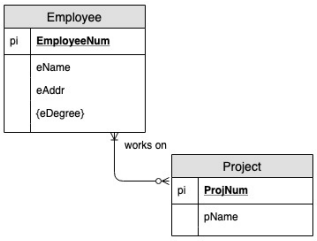


Step 4: 1:N relationship For each binary 1 TO N Relationship identify the relations that represent the participating entity at the N (i.e many) side of the relationship. Include as the foreign key in the relation that holds the N side, the primary key of the other entity (that holds the 1 side)



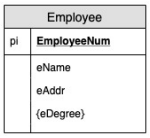
Step 5: M:N relationship

For each binary M:N Relationship create a new relation to represent the relationship. The primary key of the new relation is the combination of the primary keys of the two connected entities. This is an associative entity. If there are any attributes on the relationship, then include them.



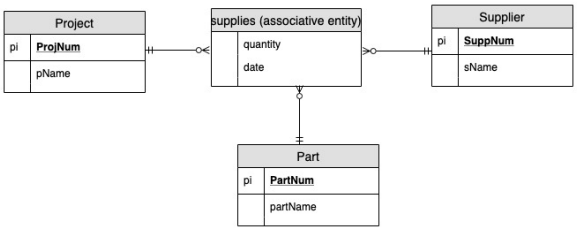
Step 6: Multi-valued attributes

For each multivalued attribute, create a new relation that includes the multivalued attribute and the primary key of the entity where the multivalued attribute is attached.



Step 7: Associative entities

For each ternary relationship create a new relation to represent the relationship. The primary key of the new relation is the combination of the primary keys of the participating entities that hold the N (many) side. In most cases of an ternary relationship, all the participating entities hold a many side.



Final Table List

Use the updated version of all tables and combine them to get the result. Take the most up-to-date version of each table. A useful habit is to cross out any tables if you create an updated version of it.

Physical Database Design - Extra

PDBD's primary purpose is to translate the logical data model, which represents the database's structure and relationships, into technical specifications for actually storing and retrieving data within a database management system (DBMS).

Key Objectives

1. **Performance**: Design should be optimised for efficient data storage and retrieval operations, considering factors such as query speed and response time.
2. **Integrity**: Should incorporate measures to prevent data corruption or inconsistencies.
3. **Security**: Should implement security mechanisms to control access to the data.
4. **Recoverability**: should support data recovery in case of system failures.

Designing Fields

1. **Choosing Data Type**: Different data types (such as integers, strings, dates) have specific storage requirements and capabilities. Essential for efficient storage and accurate data representation.
2. **Coding , Compression and Encryption**: Depending on the application, encoding, compressing, or encrypting data can be part of the design strategy to save space and enhance security.
3. **Controlling Data Integrity**: Should include mechanisms to enforce data integrity constraints, such as ensuring that certain fields are mandatory, or that certain values adhere to predefined rules.

File Organisations

- Refers to the technique of physically arranging records within a file on secondary storage devices such as hard drives or solid-state drives.
- The way data is organized within files can significantly impact data retrieval speed, storage efficiency, data integrity, and system performance.

Factors for Selecting File Organization:\

1. **Fast Data Retrieval and Throughput**: Different file organizations have varying efficiencies when it comes to fetching records quickly.
2. **Efficient Storage Space Utilization**: Reducing storage costs and maximizing available resources.
3. **Protection from Failure and Data Loss**: File organizations should support data recovery and minimize the risk of data loss.
4. **Minimizing Need for Reorganization**: Choosing an appropriate file organization can reduce the frequency of reorganization.
5. **Accommodating Growth**: Should be scalable, allowing the file to expand without significant performance degradation.'
6. **Security from Unauthorized Use**: File organization should offer security measures to prevent unauthorized access. \

Types of File

There are three primary types of File organisation

1. **Sequential**: Records are stored in a linear fashion, one after the other. This arrangement is simple and suitable for batch processing, but it can be slower for direct access to specific records.
2. **Indexed**: Involves maintaining a separate data structure, called an index, which stores pointers to the actual records. This enables faster access to records based on the index key. Common index types include B-trees and hash indexes.
3. **Hashed**: Uses a hash function to determine the storage location of records. This method provides quick access when the hash function's distribution is even, but it can be challenging to handle collisions (when multiple records map to the same location).

Sequential organization is suitable for scenarios where records are processed in order, indexed organization is well-suited for direct access, and hashed organization provides quick access based on keys. The choice depends on the specific requirements and characteristics of the data and application.