CS352 – ADVANCED DATABASE SYSTEMS

UNIT 2 - DATABASE SYSTEM DEVELOPMENT LIFE CYCLE

Dr. John Conklin

Agenda

Database System Development Life Cycle

- Database System Development Life Cycle
- Enhanced Entity Relationship Diagram
- Advanced Normalization

Database System Development Life Cycle

Introducing the Database System Development Life Cycle (DSDLC)

- **Definition:**
  - The Database System Development Life Cycle (DSDLC) is a structured approach to designing, implementing, and maintaining a database system.
- **Purpose:**
  - It ensures systematic and efficient management of the database development process.
  - Facilitates alignment with organizational goals and user requirements.

Key Phases

- **Phase 1 - Planning**
  - Objectives of the planning phase
  - Activities involved (requirements gathering, feasibility study)
  - Deliverables (project plan, feasibility report)
- **Phase 2 - Requirements Analysis**
  - Understanding user needs
  - Data requirements and specifications
  - Use of tools like Entity-Relationship Diagrams (ERDs)
- **Phase 3 - Design**
  - Logical design vs. physical design
  - Schema design and normalization
  - Tools and techniques (e.g., SQL, data modeling tools)

Key Phases

- **Phase 4 - Implementation**
  - Database creation and population
  - Testing procedures (unit testing, integration testing)
  - Data migration strategies
- **Phase 5 - Deployment**
  - Rollout strategies (phased deployment, pilot testing)
  - User training and documentation
  - Transition to maintenance phase
- **Phase 6 - Maintenance**
  - Types of maintenance (corrective, adaptive, perfective)
  - Monitoring and performance tuning
  - Handling change requests and updates

SDLC Methodologies & Best Practices

- **SDLC Methodologies**
  - Waterfall vs. Agile approaches
  - Pros and cons of each methodology
  - Adaptation to database development
- **Best Practices in Database SDLC**
  - Collaboration between stakeholders
  - Documentation standards
  - Version control and backup strategies

Introducing the Database System Development Life Cycle (DSDLC)

- **Benefits:**
  - **Structured Approach**: Ensures systematic progression from concept to deployment.
  - **Risk Management**: Identifies and mitigates risks early in the development process.
  - **Quality Assurance**: Enhances data integrity, performance, and user satisfaction.
  - **Scalability**: Allows for future expansions and modifications as business requirements evolve.
- **Methodologies:**
  - Can be adapted to various methodologies like Waterfall, Agile, or DevOps depending on project requirements and organizational culture.

Real-world Example

Let's consider a real-world example of implementing the Software Development Life Cycle (SDLC) for a database system development project:

Scenario: Implementing SDLC for a Customer Relationship Management (CRM) Database System

1. Planning Phase:

- Objective: Develop a CRM database system to streamline customer interactions and improve customer service.
- Activities:
  - Define project scope and objectives.
  - Identify stakeholders and gather requirements.
  - Create a project plan outlining timelines and resources.

2. Analysis Phase:

- Objective: Understand the data requirements for managing customer information effectively.
- Activities:
  - Conduct interviews with sales, marketing, and customer service teams to gather requirements.
  - Analyze existing customer data sources and systems.
  - Define data entities such as customers, contacts, interactions, and transactions.

Real-world Example

3. Design Phase:

- Objective: Design a database schema and data model for the CRM system.
- Activities:
  - Design tables for storing customer information, contact details, sales data, etc.
  - Define relationships between different data entities.
  - Establish data integrity constraints and security measures.

4. Implementation Phase:

- Objective: Develop and implement the CRM database system based on the design.
- Activities:
  - Create database tables and columns using SQL.
  - Implement data access logic and business rules.
  - Load initial customer data into the database.

Real-world Example

5. Testing Phase:

- Objective: Ensure the CRM system functions correctly and meets performance requirements.
- Activities:
  - Perform unit testing to validate individual components.
  - Conduct integration testing to test interactions between different modules.
  - Test data integrity, security features, and performance under load.

6. Deployment Phase:

- Objective: Deploy the CRM database system for production use.
- Activities:
  - Migrate the database to the production environment.
  - Provide user training on how to use the CRM system effectively.
  - Set up monitoring and support mechanisms for ongoing maintenance.

Real-world Example

7. Maintenance Phase:

- Objective: Maintain and optimize the CRM database system to meet changing business requirements.
- Activities:
  - Monitor database performance and user feedback.
  - Implement updates and enhancements based on user needs.
  - Perform regular backups and ensure data security.

By following the SDLC for the CRM database system development project, the organization can effectively manage customer relationships, improve operational efficiency, and provide better customer service.

Enhanced Entity Relationship Diagram

Enhanced ER Diagrams (EER)

Enhanced Entity-Relationship (EER) diagrams are an extension of the traditional Entity-Relationship (ER) model used in database design. EER diagrams incorporate additional concepts and features to represent more complex relationships and constraints in a database schema. Here is the definition of Enhanced ER Diagrams (EER):

Definition:

Enhanced Entity-Relationship (EER) diagrams are a type of data modeling diagram that extends the concepts of the traditional Entity-Relationship (ER) model by incorporating additional modeling constructs to represent more complex relationships, constraints, and attributes in a database schema. EER diagrams include subtypes, supertypes, specialization, generalization, inheritance, and union types to provide a more detailed and expressive data model representation.

Enhanced ER Diagrams (EER)

Key Features of Enhanced ER Diagrams (EER):

1. **Subtypes and Supertypes**: EER diagrams allow entities to be grouped into subtypes and supertypes, enabling the modeling of hierarchical relationships between entities.

2. **Specialization and Generalization**: EER diagrams support the concepts of specialization and generalization, allowing entities to be specialized into more specific subtypes or generalized into broader supertypes.

3. **Inheritance**: EER diagrams include the notion of inheritance, where attributes and relationships defined in a supertype are inherited by its subtypes.

4. **Union Types**: EER diagrams can represent overlapping subtypes using union types, where an entity can belong to multiple subtypes simultaneously.

5. **Complex Relationships**: EER diagrams allow for the representation of complex relationships, such as recursive relationships, multi-valued attributes, and ternary relationships.

6. **Attribute Inheritance**: EER diagrams enable attributes to be inherited by subtypes from their supertypes, reducing redundancy and improving data consistency.

Enhanced ER Diagrams (EER)

- **Key Concepts of EER Diagrams**
  - **Subtypes and Supertypes**
    - Hierarchical categorization of entities
    - Example: Employee as a subtype of Person
  - **Specialization and Generalization**
    - Specialization: Creating new entities from existing ones
    - Generalization: Grouping entities into higher-level abstractions
- **Additional Features in EER Diagrams**
  - **Attributes Inheritance**
    - Subtypes inherit attributes from supertypes
    - Reduces redundancy and improves data consistency
  - **Union Types**
    - Combines multiple entity types into a single entity set
    - Useful for representing entities with similar attributes and relationships

Enhanced ER Diagrams (EER)

Relationship Types in EER Diagrams

- **Unary Relationships**
  - Relationship within the same entity type
  - Example: Employee manages Employee
- **Binary Relationships**
  - Relationship between two different entity types
  - Example: Student enrolls in Course

Cardinality Constraints

- **One-to-One (1:1), One-to-Many (1:M)**
- **Many-to-One (N:1), Many-to-Many (M:M)**
- Illustrations and examples for each cardinality type
- Importance of defining cardinality for database integrity and performance

Enhanced ER Diagrams (EER)

Participation Constraints

- **Total Participation vs. Partial Participation**
  - Total Participation: Every entity in a relationship must participate
  - Partial Participation: Participation is optional
- Implications on database design and data integrity

Example of Total Participation:

In a university database, the relationship between the "Department" entity and the "Professor" entity may have total participation, meaning that every department must have at least one professor associated with it.

Example of Partial Participation:

- In a library database, the relationship between the "Book" entity and the "Author" entity may have partial participation, meaning that not every book needs to have an associated author (e.g., for anonymous books).

Enhanced ER Diagrams (EER)

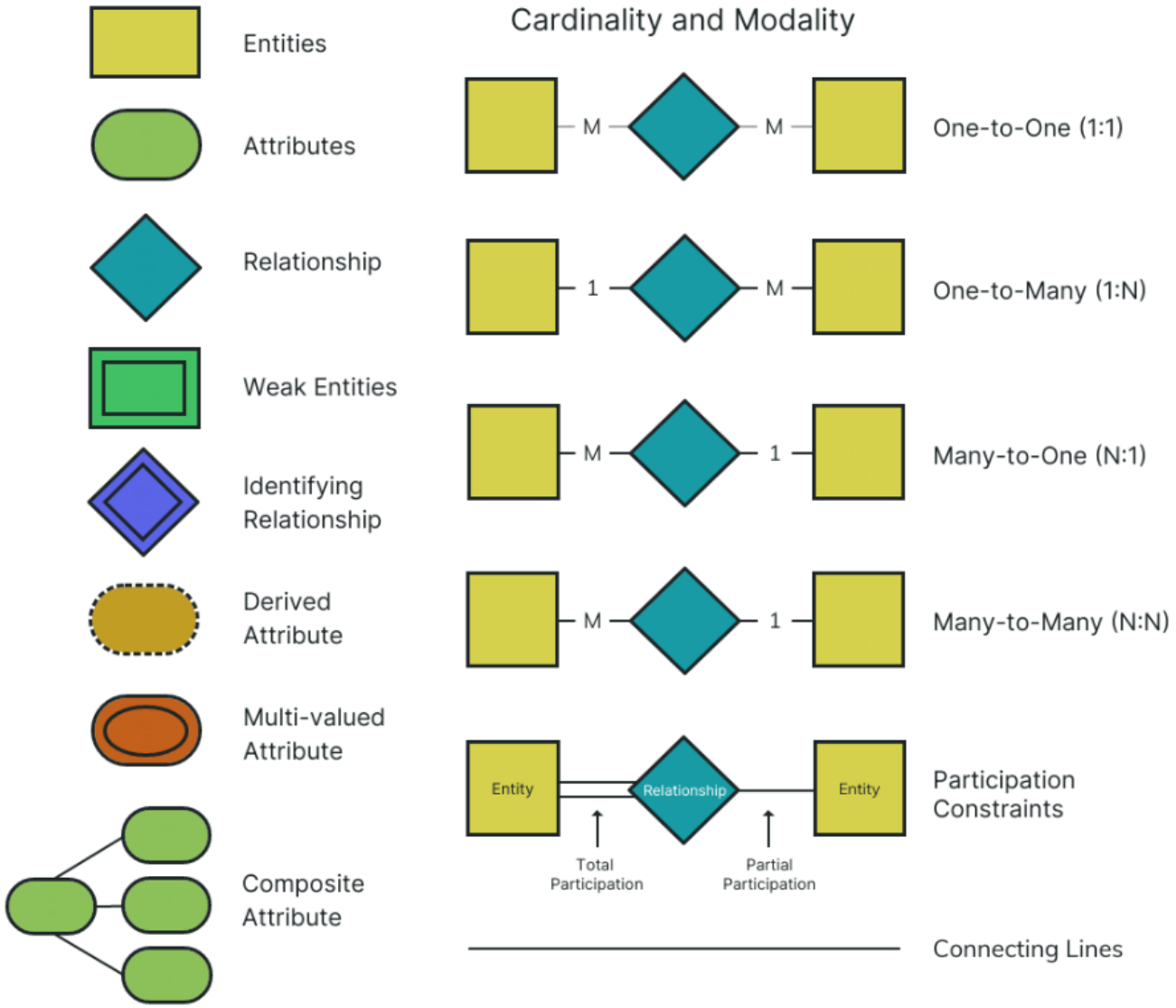
Advantages of EER Diagrams

- **Complexity Management**: Handle complex data relationships effectively
- **Flexibility**: Accommodate changes in data requirements and business rules
- **Visual Clarity**: Clear representation of data structure and constraints

Best Practices for Designing EER Diagrams

- Use of appropriate symbols and notation
- Maintain consistency in naming conventions and attribute definitions
- Regular review and validation with stakeholders for accuracy

ERD Symbols and Notations



https://venngage-wordpress.s3.amazonaws.com/uploads/2023/11/ERD_Symbols_and_Notations.png

Here are the primary symbols and notations used in ERDs:

1. Entities: Represented by rectangles.

2. Attributes: Represented by ovals.

3. Relationships: Depicted by diamonds.

4. Connecting lines: Solid lines are used to connect.

5. Cardinality and modality: These are the relationship indicators between entities.

•One-to-One (1:1): A single instance of an entity is associated with a single instance of another entity.

•One-to-Many (1:N): A single instance of an entity is associated with many instances of another entity.

•Many-to-One (N:1): Many instances of an entity are associated with a single instance of another entity.

•Many-to-Many (N:N): Many instances of an entity are associated with many instances of another entity.

6. Weak entities: Sometimes represented by a double rectangle.

7. Identifying relationships: Depicted by a double diamond.

8. Derived attribute: Represented by a dashed oval.

9. Composite attribute: shown as an oval with ovals inside it.

10. Multi-valued attribute: Depicted by a double oval.

11. Participation constraints: depicted by placing either “partial” or “total” on the lines that connect entities to relationships.

ADVANCED NORMALIZATION

Normalization

Normalization is the process of efficiently organizing data in a database. This process involves breaking a table into smaller tables and defining relationships to minimize redundancy and dependency.

Definition: Normalization is the process of structuring a relational database following a series of normal forms to reduce data redundancy and improve data integrity.

Key Concepts in Normalization

1. **First Normal Form (1NF):**
   - Ensures that each column in a table contains atomic (indivisible) values.
   - Eliminates repeating groups and ensures each field contains only one piece of data.
2. **Second Normal Form (2NF):**
   - It requires that the table is in 1NF and all non-key attributes are fully functional and dependent on the primary key.
   - Eliminates partial dependencies by moving non-key attributes to a separate table.
3. **Third Normal Form (3NF):**
   - Requires the table to be in 2NF and all transitive dependencies are removed.
   - Ensures that non-key attributes are not dependent on other non-key attributes.
4. **Boyce-Codd Normal Form (BCNF):**
   - A stricter form of 3NF that eliminates all anomalies related to functional dependencies.
   - Every determinant must be a candidate key.
5. **Fourth Normal Form (4NF), Fifth Normal Form (5NF), etc.:**
   - Higher normal forms that address more complex data integrity issues by further eliminating dependencies and anomalies.

Benefits of Normalization

- **Reduces data redundancy**: Organizing data into separate tables and avoiding duplicate information.
- **Improves data integrity**: By ensuring data consistency and accuracy through defined relationships and constraints.
- **Facilitates database maintenance**: This makes updating and modifying data easier without affecting the entire database.
- **Enhances database performance** By reducing storage requirements and improving query performance.

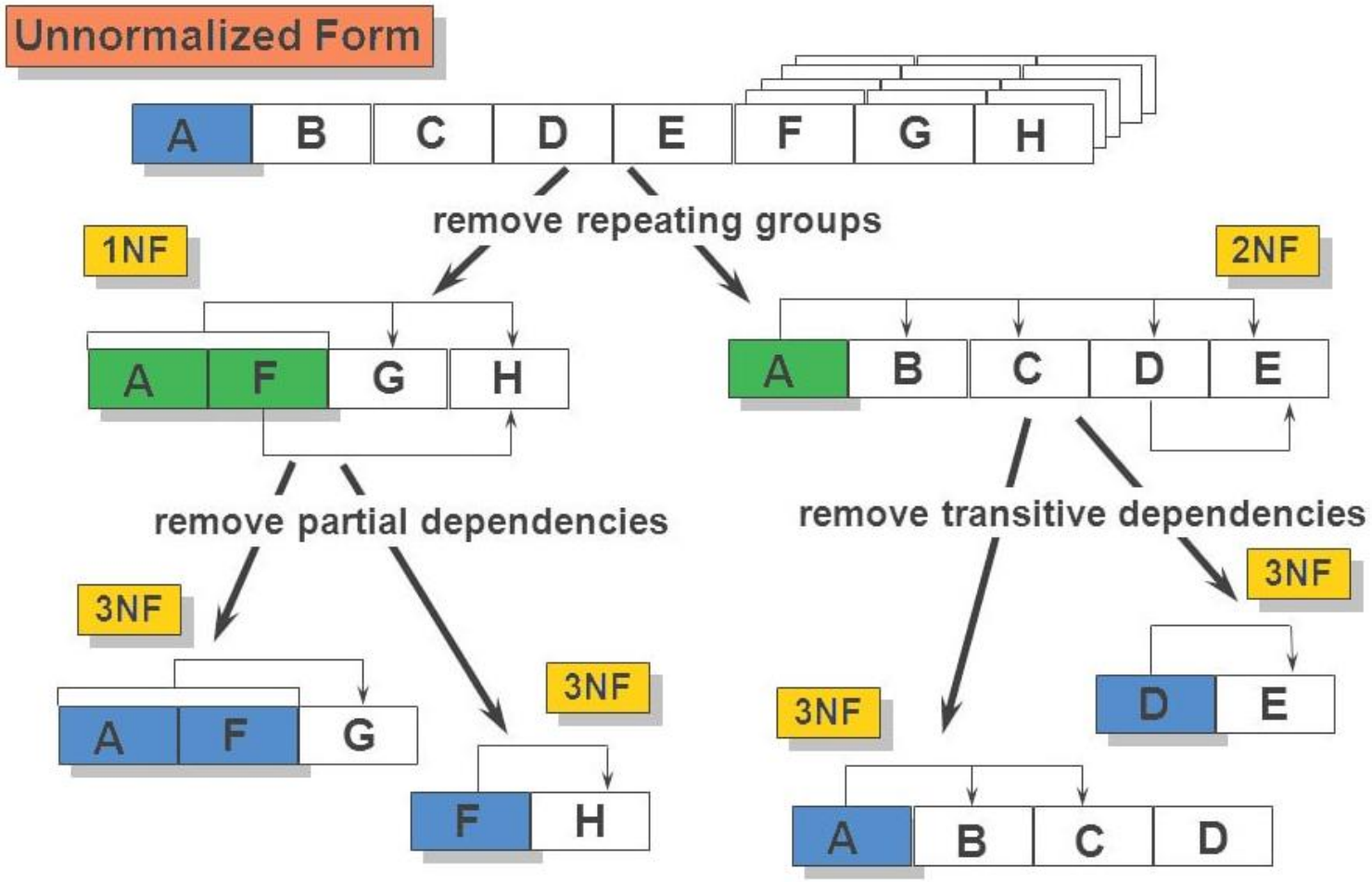
In conclusion, normalization is a crucial aspect of database design that helps optimize data storage, improve data integrity, and ensure efficient database operations.

Motivation for Advanced Normalization

1. **Primary Motivations**
   - **Data Integrity:** Advanced normalization reduces data redundancy and minimizes the risk of inconsistencies.
   - **Query Performance:** Well-normalized databases often lead to faster query execution.
   - **Scalability:** Properly normalized databases are easier to scale and maintain as the data grows.
   - **Simplifying Updates:** Updates and modifications become more straightforward with advanced normalization.
2. **Challenges Addressed**
   - **Anomalies:** Advanced normalization helps in reducing insertion, update, and deletion anomalies.
   - **Complex Relationships:** It facilitates managing complex relationships between entities effectively.
   - **Space Utilization:** Efficient use of storage space by eliminating redundant data.

Motivation for Advanced Normalization

3. **Use Cases**
   - **Online Transaction Processing (OLTP) Systems:** Benefit greatly from advanced normalization for fast and reliable transaction processing.
   - **Data Warehousing:** Proper normalization is crucial for organizing large volumes of data in data warehousing environments.
4. **Conclusion**
   - Advanced normalization is essential for maintaining data integrity, improving performance, and ensuring scalability in database systems.



https://estuary.dev/data-normalization/

Database Normalization

Pictorial representation of the first three levels of standard form.

Real World Examples

1. Third Normal Form (3NF):

Scenario: Imagine a company's database that stores employee information, including employee ID, employee name, department, and department manager. In a simplistic design, you might have a table like this:

Employee ID	Employee Name	Department	Manager Name
001	John Smith	Sales	Mary Johnson
002	Jane Doe	Marketing	Tom Brown

Normalization Issue:

- **Transitive Dependency:** The department manager's name (Manager Name) depends on the department, not directly on the employee. This creates a transitive dependency because Employee ID determines Department, and Department determines Manager Name.

Normalization Solution (3NF):

- Separate the data into two tables:
  - **Employees:** Employee ID, Employee Name, Department ID (foreign key)
  - **Departments:** Department ID (primary key), Department Name, Manager ID (foreign key)

Real World Examples

2. Boyce-Codd Normal Form (BCNF):

Scenario: Consider a university database where courses are offered. Each course can have multiple instructors, and each instructor can teach multiple courses. Initially, you might have a table like this:

Course ID	Course Name	Instructor ID	Instructor Name
101	Mathematics 101	001	Prof. Smith
102	Physics 201	002	Prof. Johnson

Normalization Issue:

- **Functional Dependency:** Instructor Name depends on Instructor ID, but Instructor ID also depends on Course ID. This creates a non-trivial functional dependency that violates BCNF.

Normalization Solution (BCNF):

- Separate the data into two tables:
  - **Courses:** Course ID (primary key), Course Name
  - **Instructors:** Instructor ID (primary key), Instructor Name
  - **Teaches:** Course ID (foreign key), Instructor ID (foreign key)

Real World Examples

3. Fourth Normal Form (4NF):

Scenario: Imagine an online store where customers can make purchases. Each purchase can contain multiple products, and each product can be purchased multiple times. Initially, you might have a table like this:

Purchase ID	Customer Name	Product ID	Product Name	Quantity
001	John Doe	101	Laptop	1
001	John Doe	102	Mouse	2

Normalization Issue:

- **Multi-valued Dependency:** Customer Name determines multiple products (Product ID and Product Name). This leads to redundancy and violates 4NF.

Normalization Solution (4NF):

- Separate the data into multiple tables:
  - **Purchases:** Purchase ID (primary key), Customer ID (foreign key), Purchase Date
  - **Products:** Product ID (primary key), Product Name
  - **Purchase Details:** Purchase ID (foreign key), Product ID (foreign key), Quantity

Real World Examples

4. Fifth Normal Form (5NF):

Scenario: Consider a social media platform where users can create posts and comment on posts. Each post can have multiple comments, and each comment can reference multiple posts. Initially, you might have a table like this:

Post ID	Post Content	Comment ID	Comment Text	User ID
101	Hello World!	001	Nice post!	201
101	Hello World!	002	I agree!	202

Normalization Issue:

- **Join Dependency:** Decomposing tables in a way that allows them to be joined back together without losing information.

INDIVIDUAL PROJECT

Individual Project

Description

After you have analyzed the existing material used by the company for their day-to-day duties, the current Access database, and the additional requirements that the current system does not meet, the following requirements entity/attributes have been compiled:

- Customer information is tracked using ‘standard’ attributes.
- A customer may purchase products or services; if they purchase products or services, the following is tracked:
    - Product purchased
    - Date of purchase
    - Total of purchase
- A supplier may sell products or services; if they sold products or services, the following is tracked:
    - Product sold
    - Date of sale
    - Total of sale
    - Is item purchased available for resale
- Your company wants to keep the number of tables storing address information to a bare minimum (read this as 1).
- Customers can be both a "supplier" and "someone that purchased services," and it is not required that they be either.
- Employee information is tracked using standard attributes.
- An employee will be considered either customer-interfacing or internal support.

Individual Project

- If the employee is customer facing, the following information is tracked:
  - Customer for interaction (note that a customer will only interface with a single employee)
  - Product specialty
  - Hours of training
  - Commission rate
- If the employee is internal support, then the following is tracked:
  - Salary
  - Support area
- An employee can be either customer facing or internal support, but not both.
- Your company wants to keep the number of tables storing generic employee information to a bare minimum (read this as 1).
- Finally, the company wants to track products or services offered. This should be a single table with typical attributes that describe inventory.
- Every customer that either makes a purchase or sells goods to the company must be associated with an employee.
- Every transaction that a customer makes with the company is stored/tracked, a customer may buy or sell many products, and a product is sold to more than one customer.

Individual Project

- Create an enhanced ERD to meet these requirements. Ensure that entities are properly defined, and appropriate attributes are listed for each entity. Also, ensure that all entities are properly related.
- Add your ERD as a screenshot to a Word document and provide supporting discussion about the need for the enhanced diagram tools and the reasoning behind the multiplicity for the relationships.
- Add the enhanced ERD and the discussion about the relationship multiplicity to your project template to the section titled "Database System Development Life Cycle."
- Name the document **CS352_<First and Last Name>_IP2**.docx and submit the document for grading.
- Please submit your assignment.
- For assistance with your assignment, please use your textbook and all course resources.

Contact Information

Email:	Jconklin@coloradotech.edu
Phone:	602.796.5972
Website:	http://drjconklin.com
Office Hours:	Wednesdays: 6:00 PM – 7:00 PM (CST)
	Saturdays: 11:00 AM – 12:00 PM (CST)
Live Chats:	Wednesdays: 7:00 PM – 8:00 PM (CST)