



Unit 3: Database Management Systems

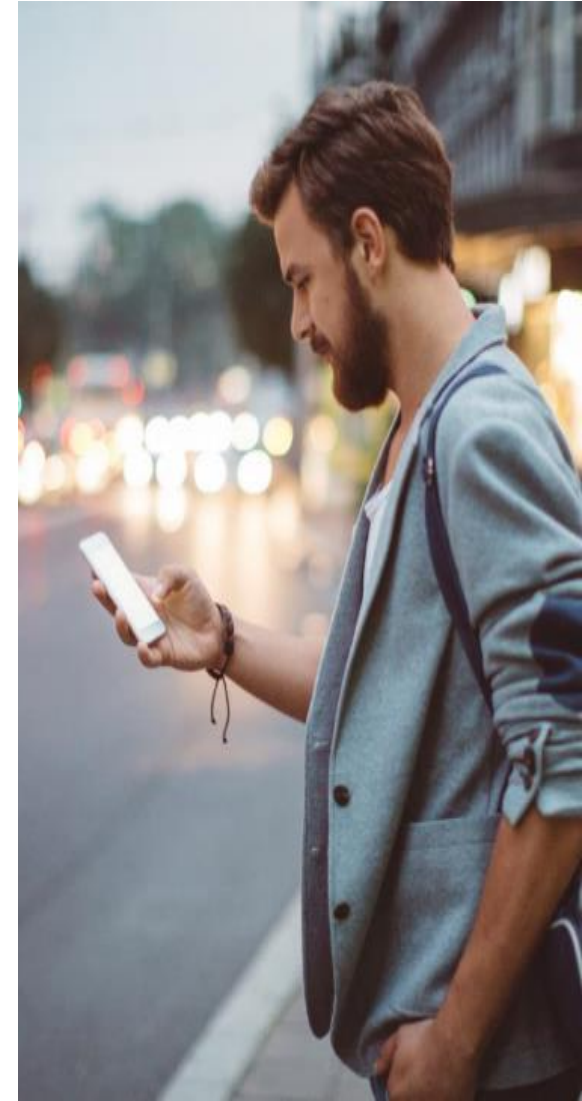
CS352-ADVANCED DATABASES

DR. JOHN CONKLIN

Agenda

Today's Topics:

1. Overview
2. Importance of Normalization
3. Database Dependencies
4. Step-by-Step Normalization
5. Logical Data Model Requirements
6. Subclass/Superclass Relationships
7. Associative Tables and M:N Relationships
8. Transaction Management Basics
9. Database Security Concepts
10. Summary of Expectations
11. Individual Project





Overview



- This unit focuses on transforming Enhanced ERDs into logical data models.
- You will apply normalization to move data from UNF to Boyce-Codd Normal Form (BCNF).
- Key areas include identifying primary and foreign keys, modeling subclass/superclass relationships, and understanding transaction and security principles.

Definition: Normalization is the process of organizing data in a database to reduce data redundancy and improve data integrity.

What is Normalization? (Briefly)

Analogy: Think of it like organizing a messy filing cabinet or a disorganized library. We want each piece of information in its logical place, without unnecessary duplicates.

Goal: To structure tables and relationships in a way that minimizes data anomalies and ensures consistency.

Key Concept: Achieved by adhering to "Normal Forms" (e.g., 1NF, 2NF, 3NF, BCNF) – a set of rules for database design.



Why Normalize?



1. Reason 1: Eliminates Data Redundancy
2. Reason 2: Ensures Data Integrity & Consistency (Addressing Anomalies)
3. Reason 3: Improves Data Quality and Reliability
4. Reason 4: Optimizes Database Performance (for certain operations)
5. Reason 5: Simplifies Database Design & Maintenance

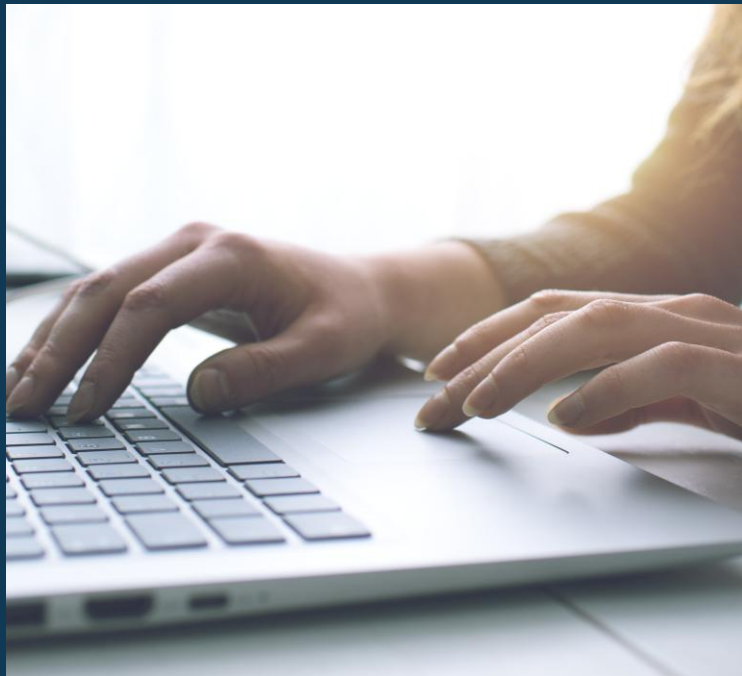


The Trade-off: Normalization vs. Denormalization (Briefly)



1. Normalization is Key For...
2. Denormalization (Controlled Redundancy) is Sometimes Used For...

Key Takeaways



1. Normalization is Fundamental
2. Ensures Trustworthy Data:
3. Drives Efficiency.
4. Simplifies Management
5. Enables Better Decisions:



Database Dependencies

Database dependencies are fundamental concepts in relational database design that help ensure data integrity and guide the normalization process. Here are the main types:

- Functional Dependencies (FD)
- Partial Dependencies
- Full Functional Dependencies
- Transitive Dependencies
- Multivalued Dependencies (MVD)
- Join Dependencies
- Inclusion Dependencies
- Key Dependencies



Functional Dependencies (FD)

A functional dependency $X \rightarrow Y$ means that for any two rows in a table, if they have the same value for attribute(s) X , they must have the same value for attribute(s) Y . In other words, X uniquely determines Y .

Example: In an Employee table, $\text{EmployeeID} \rightarrow \text{Name}$, because each employee ID corresponds to exactly one name.

Types of functional dependencies:

- **Trivial FD:** Y is a subset of X (like $\text{Name, Age} \rightarrow \text{Name}$)
- **Non-trivial FD:** Y is not a subset of X
- **Completely non-trivial FD:** X and Y have no attributes in common

Partial Dependencies

This occurs when a non-key attribute depends on only part of a composite primary key, rather than the entire key. This violates Second Normal Form (2NF).

Example: In a table with a composite key (StudentID, CourseID), if StudentName depends only on StudentID (not the full key), it's a partial dependency.

A non-key attribute depends on the entire primary key, not just part of it. This is required for 2NF compliance.

Full Functional Dependencies

Transitive Dependencies

When a non-key attribute depends on another non-key attribute, which in turn depends on the primary key ($A \rightarrow B \rightarrow C$). This violates Third Normal Form (3NF).

Example: $\text{EmployeeID} \rightarrow \text{DepartmentID} \rightarrow \text{DepartmentName}$ creates a transitive dependency.

Occurs when one attribute determines multiple independent sets of values.

Written as $X \twoheadrightarrow Y$, meaning X multidetermines Y .

Example: In a table tracking $\text{Employee} \rightarrow \text{Skills}$ and $\text{Employee} \rightarrow \text{Projects}$, an employee can have multiple skills and multiple projects independently.

Multivalued Dependencies (MVD)

Join Dependencies

A table can be decomposed into smaller tables and then rejoined without loss of information. This is relevant for higher normal forms like 4NF and 5NF.

Specify that values in one relation must also appear in another relation, similar to foreign key constraints.

Inclusion Dependencies

Key Dependencies

Candidate Key: Minimal set of attributes that uniquely identify each row

Primary Key: The chosen candidate key

Foreign Key: Attributes that reference the primary key of another table

Understanding these dependencies is crucial for database normalization, which eliminates redundancy and prevents anomalies during insert, update, and delete operations. Each normal form (1NF through 5NF) addresses specific types of dependencies to improve database design quality.



Step-by-Step Normalization



1. Normalization structures data efficiently to minimize redundancy.
2. Helps maintain data integrity across multiple tables.
3. Supports consistency in data updates, inserts, and deletes.
4. Ensures logical structure aligns with business rules.

UNF: Our Starting Point: Unnormalized Data

Data is grouped without structure, prone to anomalies.

Let's imagine a single table that stores information about Orders, Customers, and Products.

Observation: This table has many issues! Redundancy, repeating groups, and potential for inconsistencies.

| OrderID | OrderDate | CustomerID | CustomerName | CustomerAddress | ProductCode | ProductName | ProductPrice | Quantity |
|---------|------------|------------|--------------|-----------------|-------------|-------------|--------------|----------|
| 101 | 2024-06-01 | CUST001 | Alice Smith | 123 Main St | P001 | Laptop | 1200 | 1 |
| 101 | 2024-06-01 | CUST001 | Alice Smith | 123 Main St | P003 | Mouse | 25 | 2 |
| 102 | 2024-06-02 | CUST002 | Bob Johnson | 456 Oak Ave | P002 | Keyboard | 75 | 1 |
| 103 | 2024-06-03 | CUST001 | Alice Smith | 123 Main St | P001 | Laptop | 1200 | 1 |
| 103 | 2024-06-03 | CUST001 | Alice Smith | 123 Main St | P004 | Monitor | 300 | 1 |

Step 1: First Normal Form (1NF)

Removes repeating groups; each field contains atomic values.

Rules:

1. Eliminate repeating groups.
2. Each row must be unique

Our Data in 1NF

•**Primary Key:** (OrderID, ProductCode) - each combination is now unique.

| OrderID | ProductCode | OrderDate | CustomerID | CustomerName | CustomerAddress | ProductName | ProductPrice |
|---------|-------------|------------|------------|--------------|-----------------|-------------|--------------|
| 101 | P001 | 2024-06-01 | CUST001 | Alice Smith | 123 Main St | Laptop | 1200 |
| 101 | P003 | 2024-06-01 | CUST001 | Alice Smith | 123 Main St | Mouse | 25 |
| 102 | P002 | 2024-06-02 | CUST002 | Bob Johnson | 456 Oak Ave | Keyboard | 75 |
| 103 | P001 | 2024-06-03 | CUST001 | Alice Smith | 123 Main St | Laptop | 1200 |
| 103 | P004 | 2024-06-03 | CUST001 | Alice Smith | 123 Main St | Monitor | 300 |

Still Problems? Yes, we have a lot of redundancy for Customer and Product details!

Step 2: Second Normal Form (2NF)

Removes partial dependencies; every non-key attribute fully depends on the primary key.

Prerequisite: Must be in 1NF.

Our Data in 2NF (Split into 3 Tables)

Table 1: Orders (PK: OrderID)

| OrderID | OrderDate | CustomerID |
|---------|------------|------------|
| 101 | 2024-06-01 | CUST001 |
| 102 | 2024-06-02 | CUST002 |
| 103 | 2024-06-03 | CUST001 |

Table 2: Order_Items (PK: OrderID, ProductCode) - *Bridge Table*

| OrderID | ProductCode | Quantity |
|---------|-------------|----------|
| 101 | P001 | 1 |
| 101 | P003 | 2 |
| 102 | P002 | 1 |
| 103 | P001 | 1 |
| 103 | P004 | 1 |

Table 3: Products (PK: ProductCode)

| ProductCode | ProductName | ProductPrice |
|-------------|-------------|--------------|
| P001 | Laptop | 1200 |
| P003 | Mouse | 25 |
| P002 | Keyboard | 75 |
| P004 | Monitor | 300 |

Still Problems? CustomerName and CustomerAddress are still linked to CustomerID in the Orders table, but they don't depend on OrderID itself.

Step 3: Third Normal Form (3NF)

Removes transitive dependencies; attributes depend only on the key.

Our Data in 3NF (Split into 4 Tables)

Table 1: Orders (PK: OrderID)

| OrderID | OrderDate | CustomerID |
|---------|------------|------------|
| 101 | 2024-06-01 | CUST001 |
| 102 | 2024-06-02 | CUST002 |
| 103 | 2024-06-03 | CUST001 |

Table 2: Order_Items (PK: OrderID, ProductCode)

| OrderID | ProductCode | Quantity |
|---------|-------------|----------|
| 101 | P001 | 1 |
| 101 | P003 | 2 |
| 102 | P002 | 1 |
| 103 | P001 | 1 |
| 103 | P004 | 1 |

Table 3: Products (PK: ProductCode)

| ProductCode | ProductName | ProductPrice |
|-------------|-------------|--------------|
| P001 | Laptop | 1200 |
| P003 | Mouse | 25 |
| P002 | Keyboard | 75 |
| P004 | Monitor | 300 |

Table 4: Customers (PK: CustomerID)

| CustomerID | CustomerName | CustomerAddress |
|------------|--------------|-----------------|
| CUST001 | Alice Smith | 123 Main St |
| CUST002 | Bob Johnson | 456 Oak Ave |

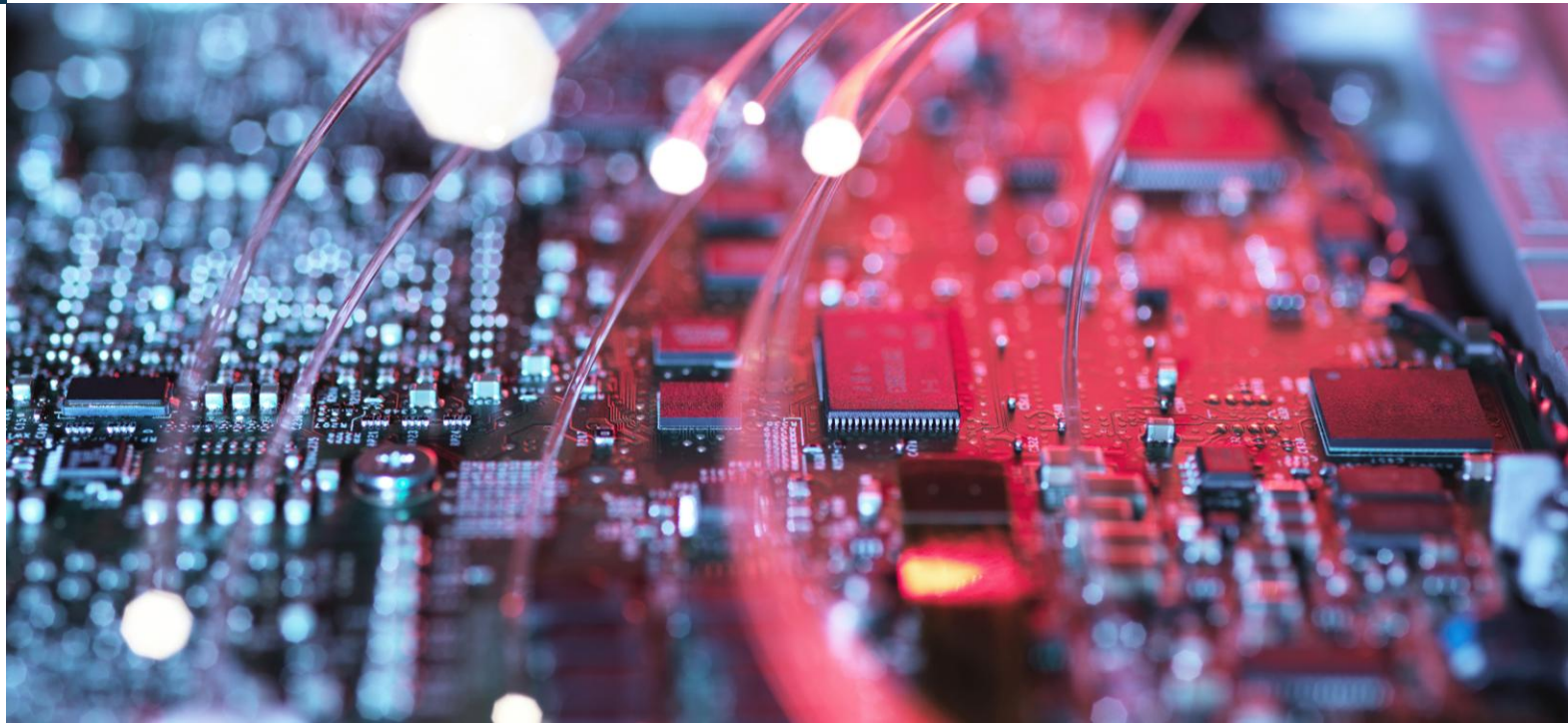
Beyond 3NF: Boyce-Codd Normal Form (BCNF)

Prerequisite: Must be in 3NF.

When it matters: BCNF addresses specific, less common Scenarios

Key difference from 3NF: 3NF allows a non-key attribute to determine part of a candidate key. BCNF is stricter, ensuring that *any* attribute that determines another attribute must be a full candidate key itself.

For most practical applications, reaching 3NF is often sufficient and provides a good balance between normalization benefits and query complexity.



Summary of Normal Forms & What They Achieve

The Normalized Advantage

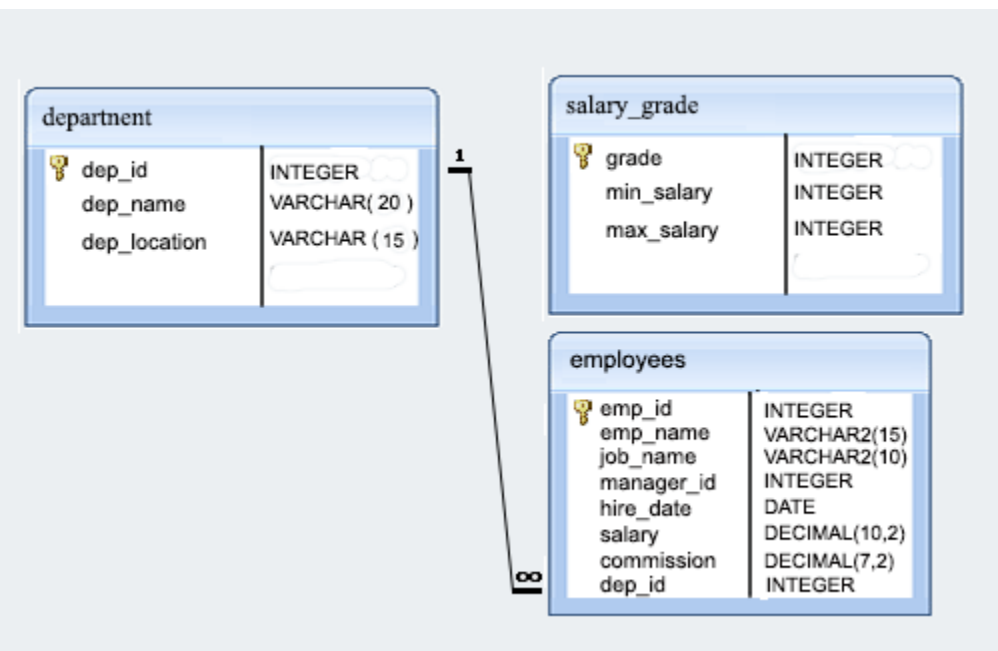
- Less Data Redundancy
- High Data Integrity
- Reduced Anomalies
- Improved Data Quality
- Easier Maintenance & Scalability
- Foundation for Trustworthy Data-Driven Decisions!

1NF: Ensures atomicity of values and unique rows.

2NF: Eliminates partial dependencies (non-key attributes depend on the entire primary key).

3NF: Eliminates transitive dependencies (non-key attributes do not depend on other non-key attributes).

BCNF (Stricter 3NF): Addresses specific cases of overlapping candidate keys.





Logical Data Model Requirements

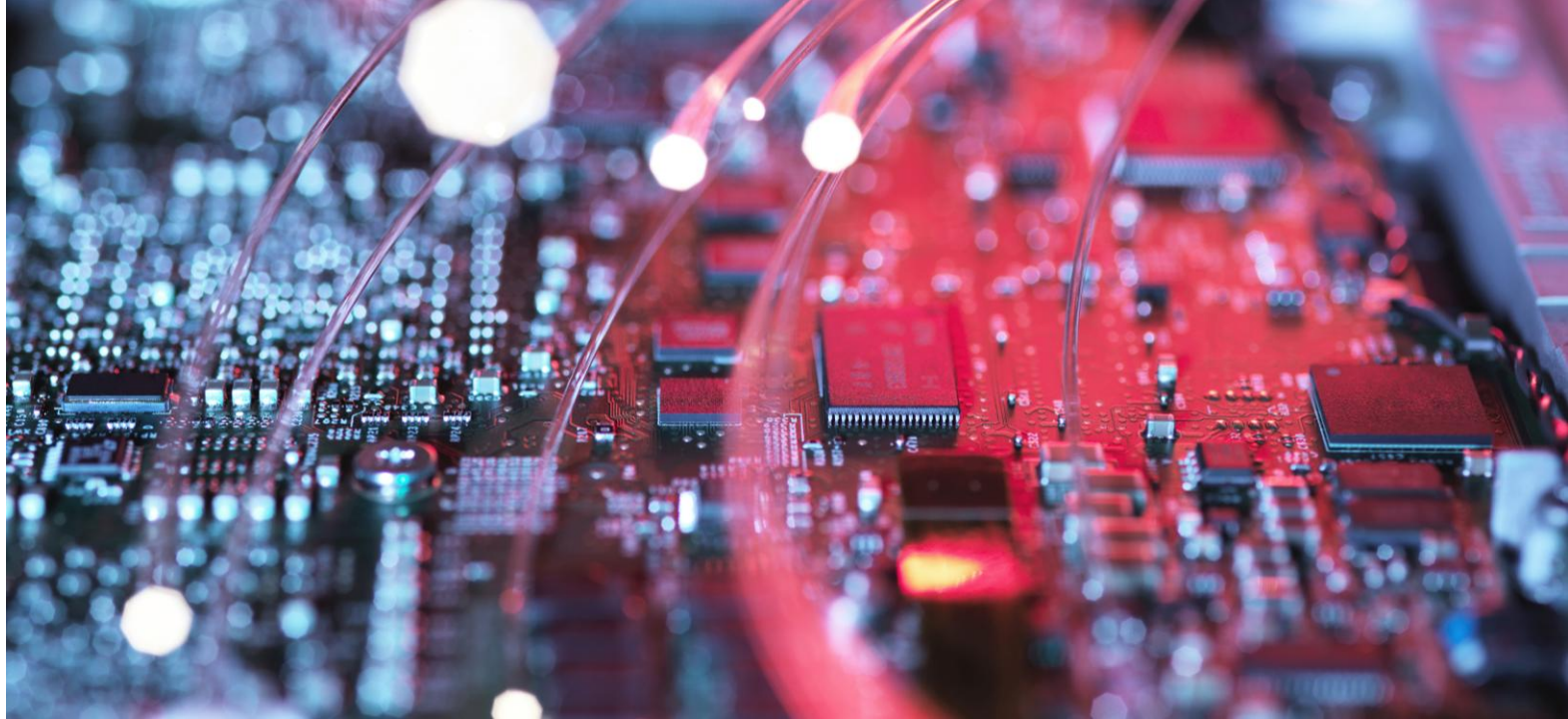


1. Tables should reflect the normalized form to at least 3NF or BCNF.
2. Primary Keys must be identified (bolded and underlined).
3. Foreign Keys should be indicated (italicized and underlined).
4. Ensure all relationships maintain referential integrity.

Logical Data Model (LDM) Requirements

1. Core Purpose & Definition
2. Key Components of a Logical Data Model
3. Requirements for a *Good* Logical Data Model
4. Inputs to Developing an LDM
5. Outputs of an LDM

By adhering to these requirements, an organization can create an LDM that serves as a robust blueprint for database design, ensuring the resulting system effectively supports business operations and provides reliable, high-quality data.





Subclass/Supersclass Relationships

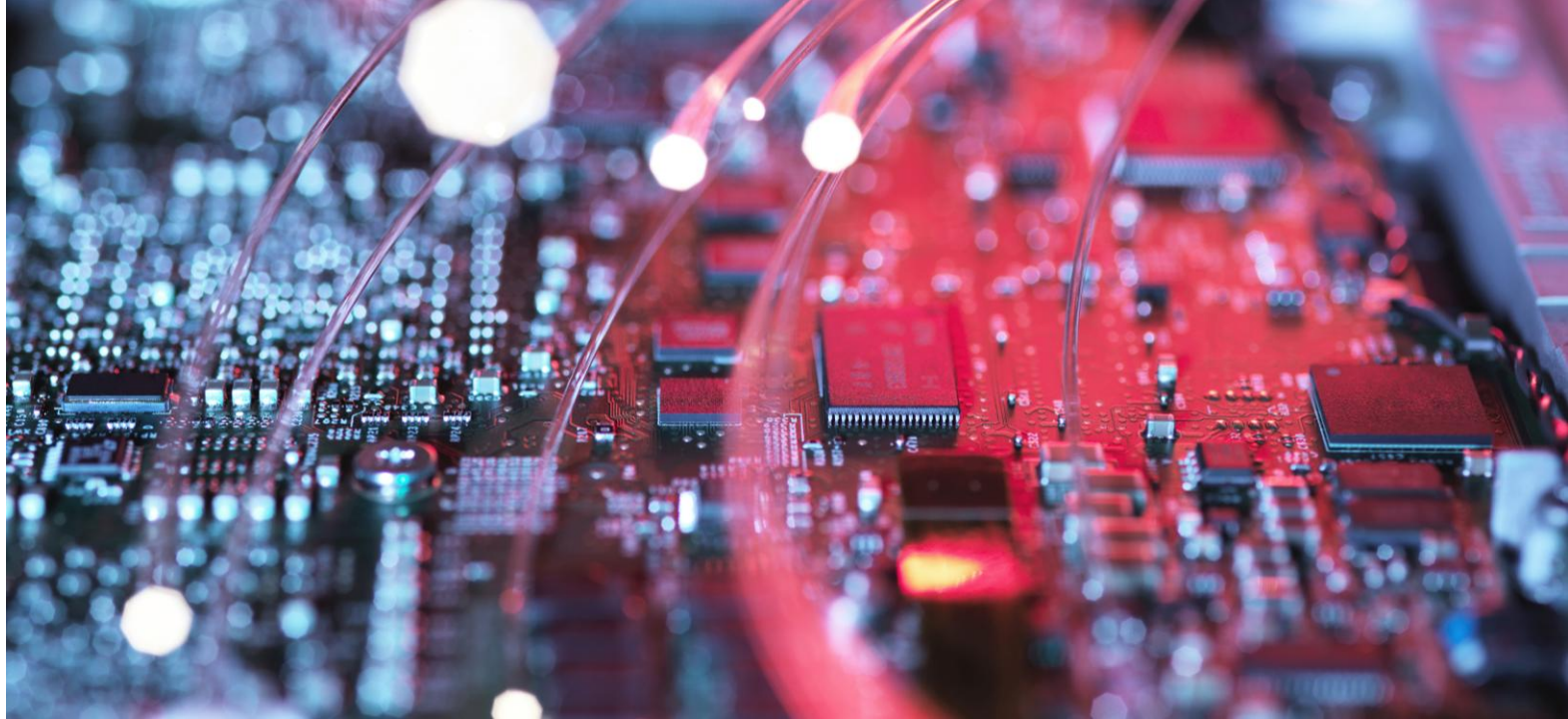


1. Superclass: General entity (e.g., Customer, Employee).
2. Subclasses: Specialized roles (e.g., Supplier, Purchaser, Internal Support).
3. Use 1:1 relationships where subclass shares PK with superclass.
4. Enforces consistent data modeling across inherited attributes.

Subclass / Superclass Relationships

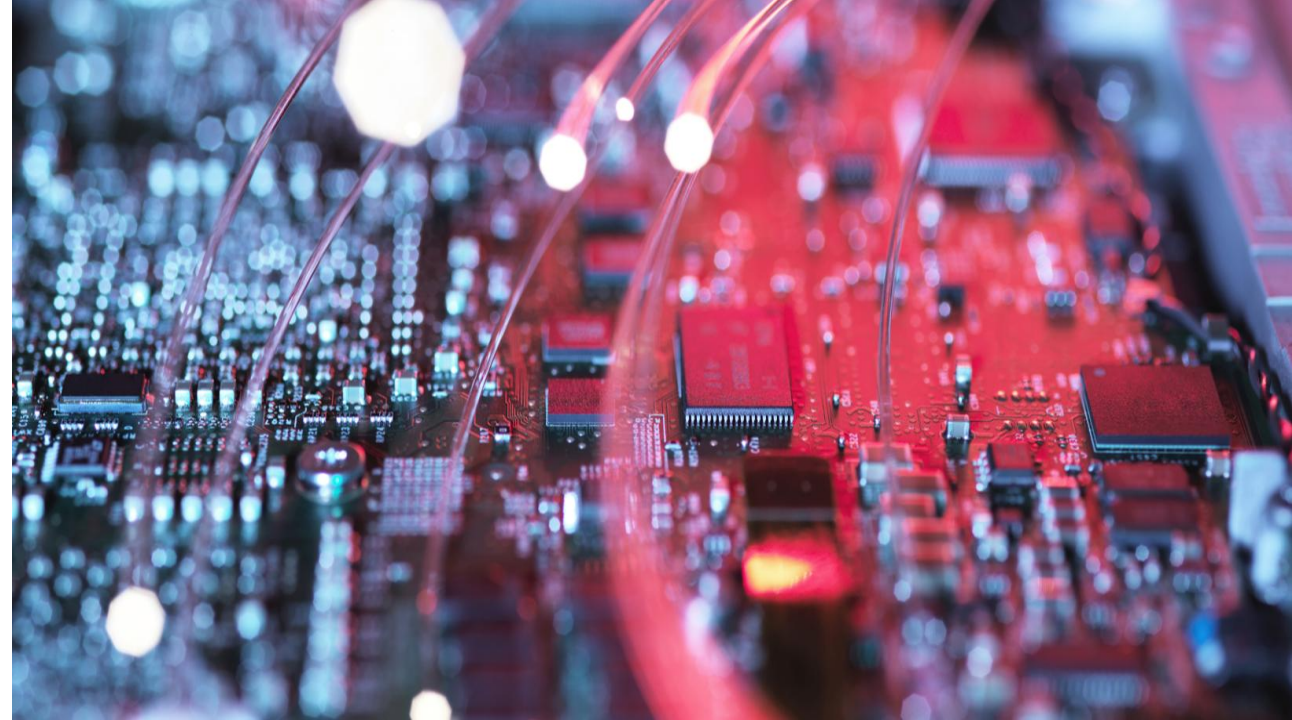
- Generalization vs. Specialization
 - Generalization (Bottom-Up)
 - Specialization (Top-Down)

- Core Concepts
 - Superclass (Parent/Base Class)
 - Subclass (Child/Derived Class)
 - "Is-a" Relationship



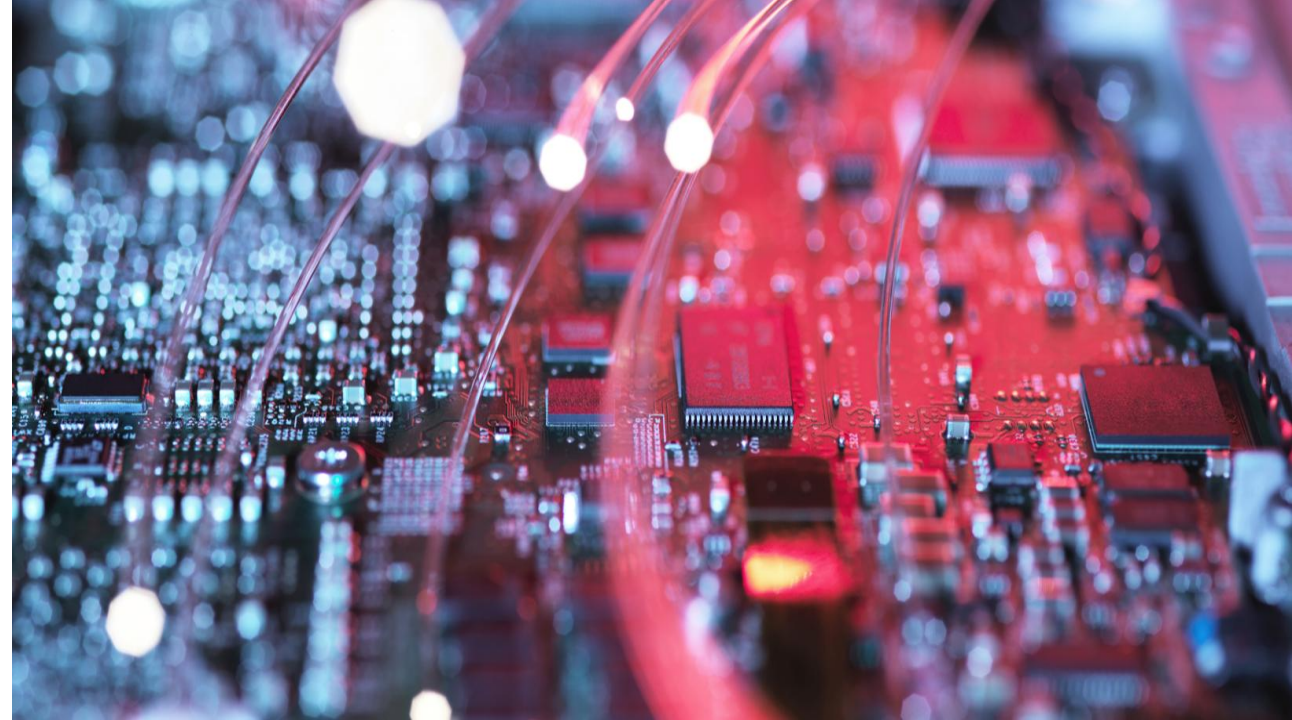
Key Characteristics & Inheritance

- Attribute Inheritance
 - Method/Behavior Inheritance
 - Overriding
 - Polymorphism (Briefly)
-
- Types of Specialization (in Data Modeling / EER)
 - Disjoint vs. Overlapping Constraint (Completeness/Participation)
 - Total vs. Partial Constraint (Cardinality/Completeness)



Benefits of Subclass/Superclass Relationships

- **Maintainability:** Changes to shared logic in the superclass automatically apply to all subclasses, simplifying updates.
- **Clearer Data Modeling:** Represents real-world hierarchies intuitively, improving the clarity and understanding of the data structure.
- **Polymorphism (OOP):** Enables flexible and powerful code that can operate on objects of different types through a common interface.
- **Code/Model Reusability:** Inheriting common attributes and behaviors reduces duplication, making models and code more concise.
- **Modularity:** Breaks down complex systems into smaller, manageable, and logically organized units.
- **Extensibility:** Easier to add new subclasses without modifying existing superclass code, promoting future growth.



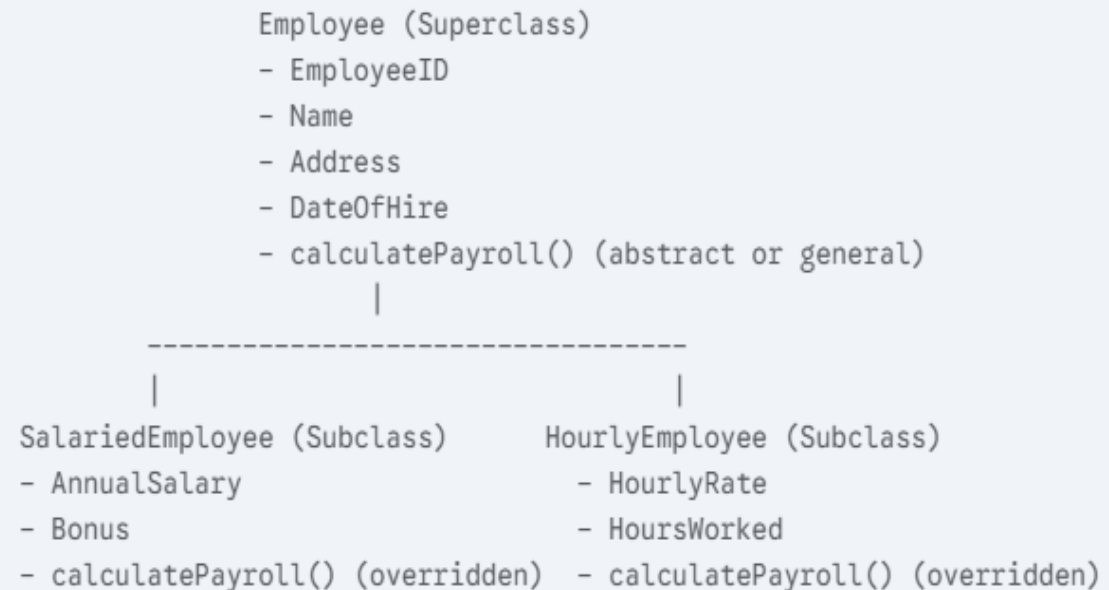
Example: Employee Hierarchy

Specialization: Each subclass adds specific attributes (AnnualSalary, HourlyRate) and provides its own calculatePayroll() logic.

Constraints (e.g., Disjoint, Total): An employee can be either salaried or hourly (Disjoint). Every employee must be one of these types (Total).

Relationship: SalariedEmployee is an Employee, HourlyEmployee is an Employee.

Inheritance: Both subclasses inherit EmployeeID, Name, Address, DateOfHire.





Associative Tables and M:N Relationships



1. Use associative tables like Orders to resolve many-to-many relationships.
2. Composite Primary Key ensures unique records (e.g., Customer_ID + Product_ID).
3. Foreign Keys link back to related parent tables (e.g., Customers, Products).
4. Maintains transaction traceability.

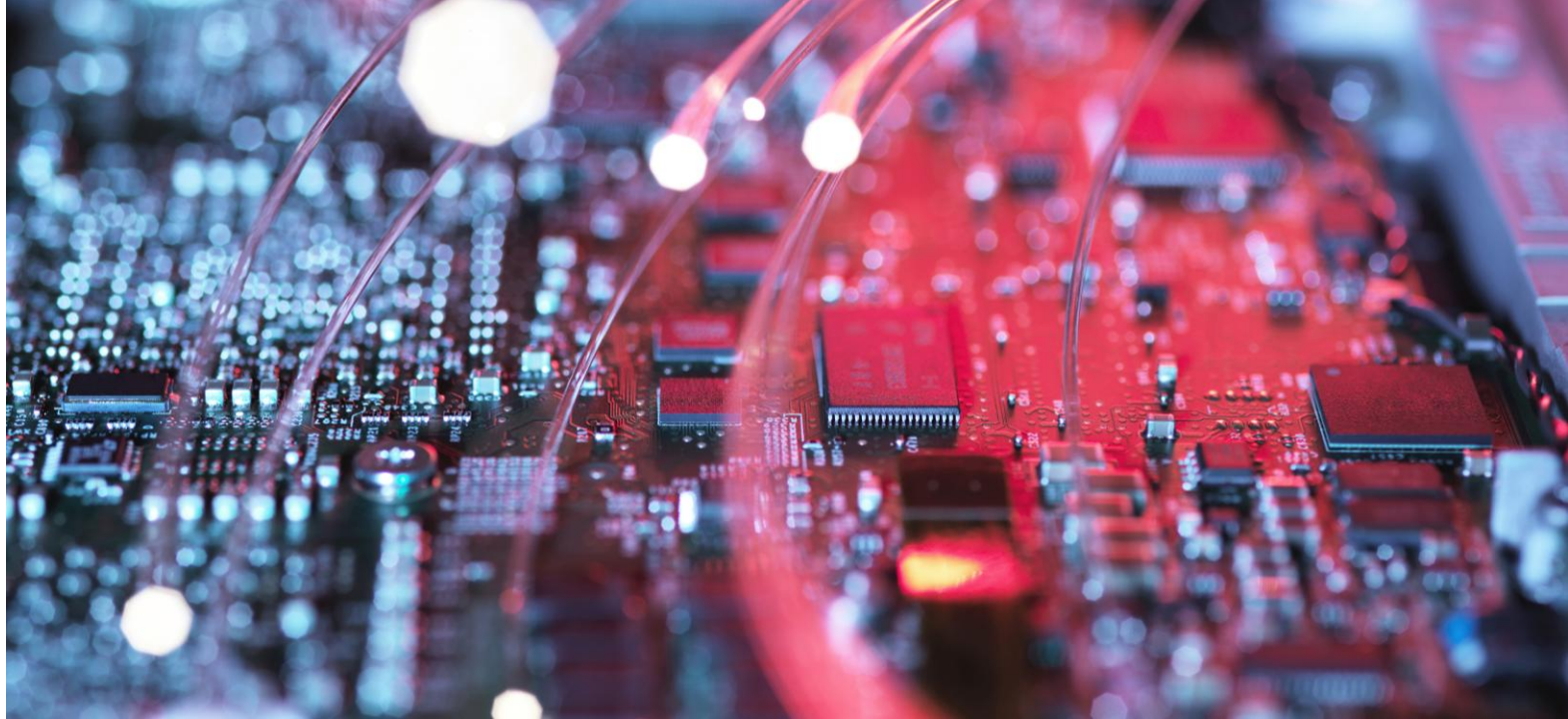
Understanding Database Relationships (Recap)

One-to-One (1:1)

One instance in Table A relates to exactly one instance in Table B.

One-to-Many (1:N or 1:M)

One instance in Table A can relate to *many* instances in Table B. Each instance in Table B relates to *one* instance in Table A.

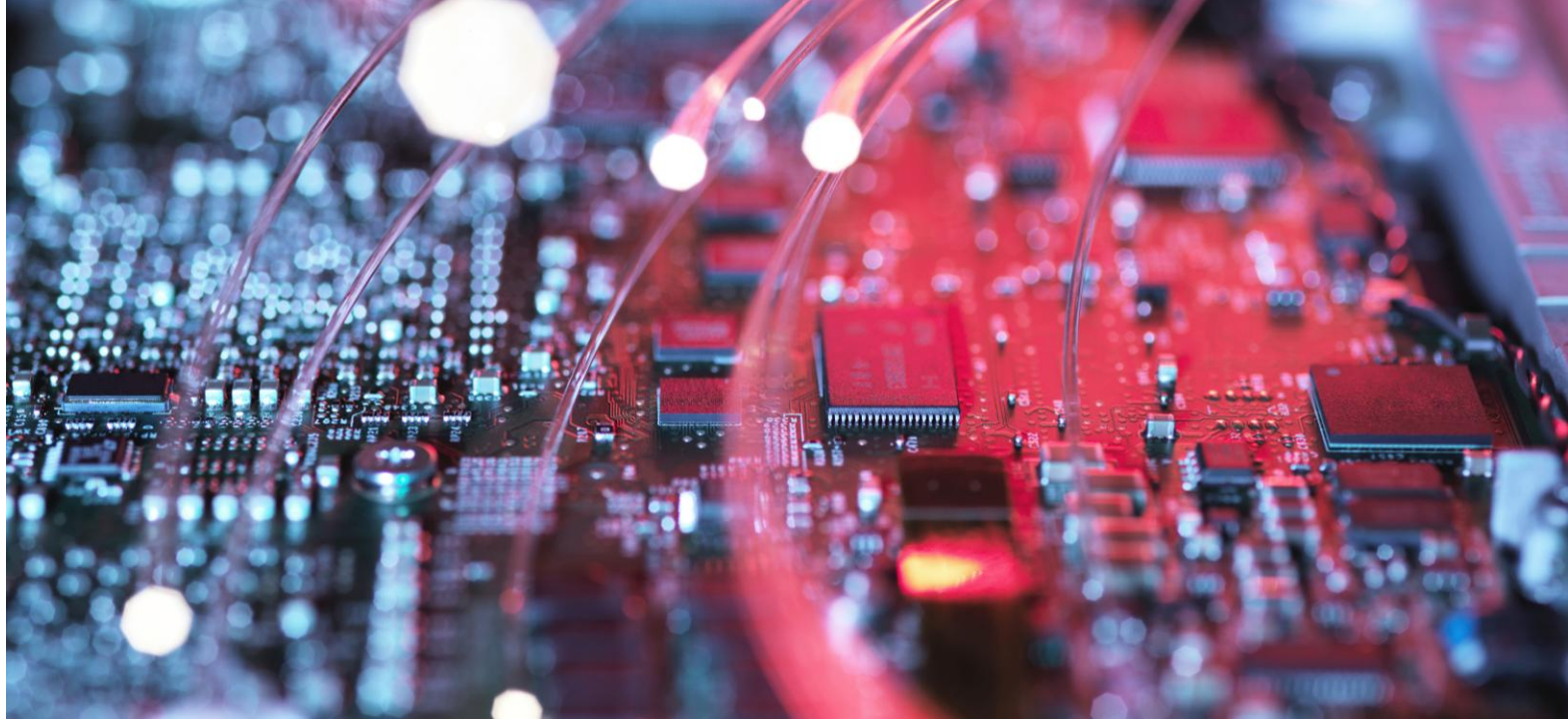


The Challenge: Many-to-Many (M:N) Relationships

An M:N relationship exists when one instance of Entity A can relate to *many* instances of Entity B, AND one instance of Entity B can relate to *many* instances of Entity A

Common Examples:

1. Students and Courses
2. Books and Authors
3. Orders and Products



Why M:N is a Problem in Relational Databases

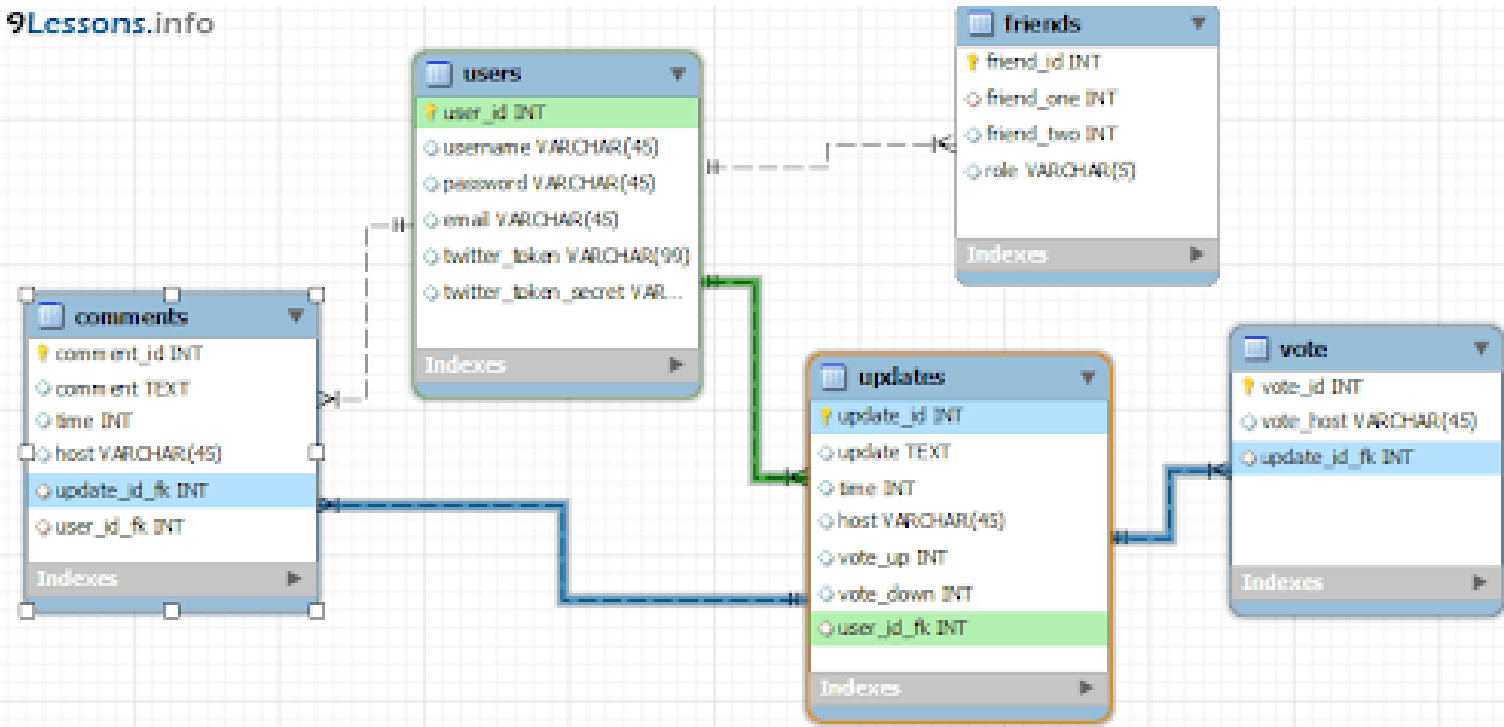
Relational database tables are designed for unique rows and direct relationships (1:1, 1:N).

You cannot directly represent an M:N relationship within two tables without severe issues.

Attempting Direct M: N Leads To:

1. Redundancy
2. Data Inconsistency
3. Difficulty Querying/Joining

9Lessons.info



The Solution: Associative Tables

Also Known As: Junction Table, Bridge Table, Linking Table, Join Table.

What it is...

How It Works...

Reports Table

| | ReportName ▾ | Creator ▾ | PrintLayout ▾ | IsAltShaded ▾ | IsSummary ▾ | FontSize ▾ |
|---|---------------------|-----------|---------------|-------------------------------------|--------------------------|------------|
| + | Accrual Report | | 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 7 |
| + | Amortization Report | | 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 8 |
| + | Callable Status | | 1 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 8 |
| + | CMR ARM Detail | | 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 8 |

Templates Table

| | ReportName ▾ | ControlID ▾ | Sequence ▾ | GroupOn ▾ | DisplaySum ▾ | Sum ▾ | Sort ▾ |
|--|---------------------|-------------|------------|--------------------------|--------------------------|--------------------------|--------|
| | Amortization Report | 24 | 3 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | Amortization Report | 30 | 4 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Asc |
| | Amortization Report | 1 | 5 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| | Amortization Report | 31 | 6 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |

Anatomy of an Associative Table

Consists of:

1. Composite Primary Key
2. Foreign Keys
3. Optional Element: Additional Attributes

The diagram illustrates the structure of an associative table. It features a table with four columns and four rows. Above the table, the text 'Fields (Columns)' is centered, with a vertical line connecting it to the column headers. To the left of the table, the text 'Primary key' is positioned above the first column, with a vertical line connecting it to the 'StudentID' header. To the right of the table, the text 'Rows' is positioned vertically, with a bracket connecting it to the row data.

| StudentID | StudentName | StudentMajor | StudentEmail |
|-----------|----------------|--------------|----------------------|
| 1234 | Jonh Smith | Marketing | jsmith@university.ed |
| 2345 | Robert Jackson | MIS | rjackson@university. |
| 3456 | Anne Sun | Accounting | asun@university.edu |
| 4567 | Mary Brown | Finance | mbrown@university. |

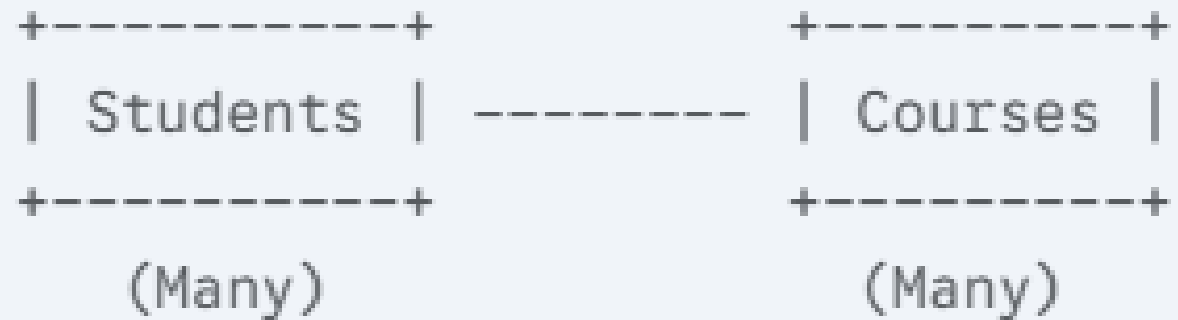
Step-by-Step

Example:

Students & Courses

Problematic Direct Implementation: If you try to put CourseIDs in Student table or StudentIDs in Course table, you'd need multiple columns or repeating groups, leading to redundancy and anomalies.

Conceptual M:N Relationship: A Student can enroll in many Courses. A Course can have many Students



Step-by-Step Example: Students & Courses (Solution)

Solution: Introduce an
Associative Table
(e.g., Enrollments)

1. **Students Table (PK: StudentID)**

| StudentID | StudentName | ... |
|-----------|-------------|-----|
| :----- | :----- | :-- |
| S001 | Alice | |
| S002 | Bob | |

2. **Courses Table (PK: CourseID)**

| CourseID | CourseName | ... |
|----------|------------|-----|
| :----- | :----- | :-- |
| CS101 | Intro CS | |
| MA201 | Calculus | |

3. **Enrollments (Associative) Table (Composite PK: StudentID , CourseID)** [↗](#)

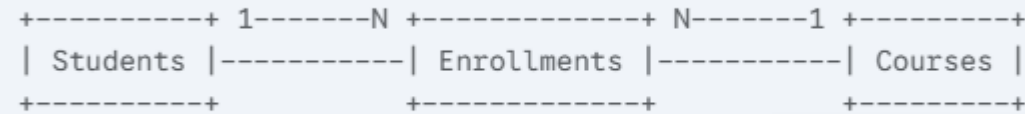
- This table links students to courses. Each row represents one student's enrollment in one course.
- It also contains an attribute specific to the enrollment: **EnrollmentDate**.

| StudentID (FK) | CourseID (FK) | EnrollmentDate | |
|----------------|---------------|----------------|--|
| :----- | :----- | :----- | |
| S001 | CS101 | 2024-09-01 | |
| S001 | MA201 | 2024-09-01 | |
| S002 | CS101 | 2024-09-02 | |

The Resulting 1:N Relationships

The original M:N relationship is now replaced by two 1:N relationships:

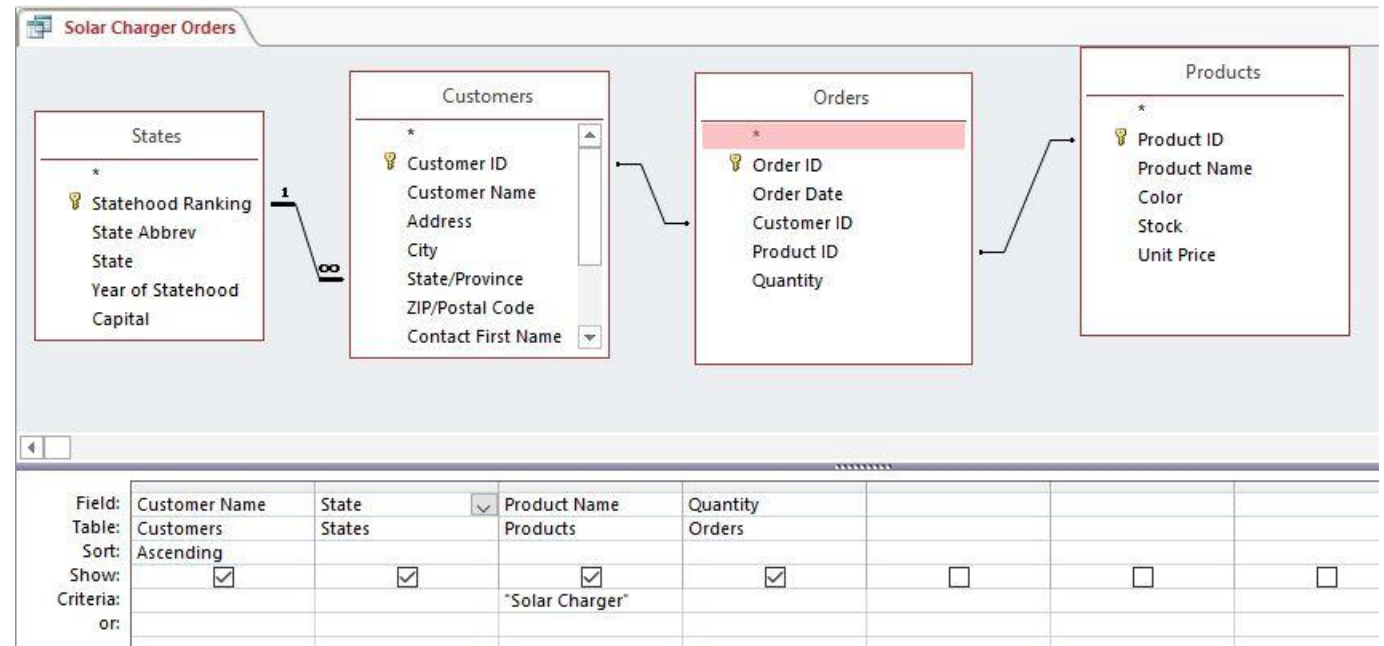
- One Student can have many Enrollments. (Students 1:N Enrollments)
- One Course can have many Enrollments. (Courses 1:N Enrollments)



The Resulting 1:N Relationships

Benefits:

1. Resolves M:N Relationships
2. Eliminates Redundancy
3. Ensures Data Integrity
4. Allows for Relationship Attributes
5. Facilitates Querying





Transaction Management Basics



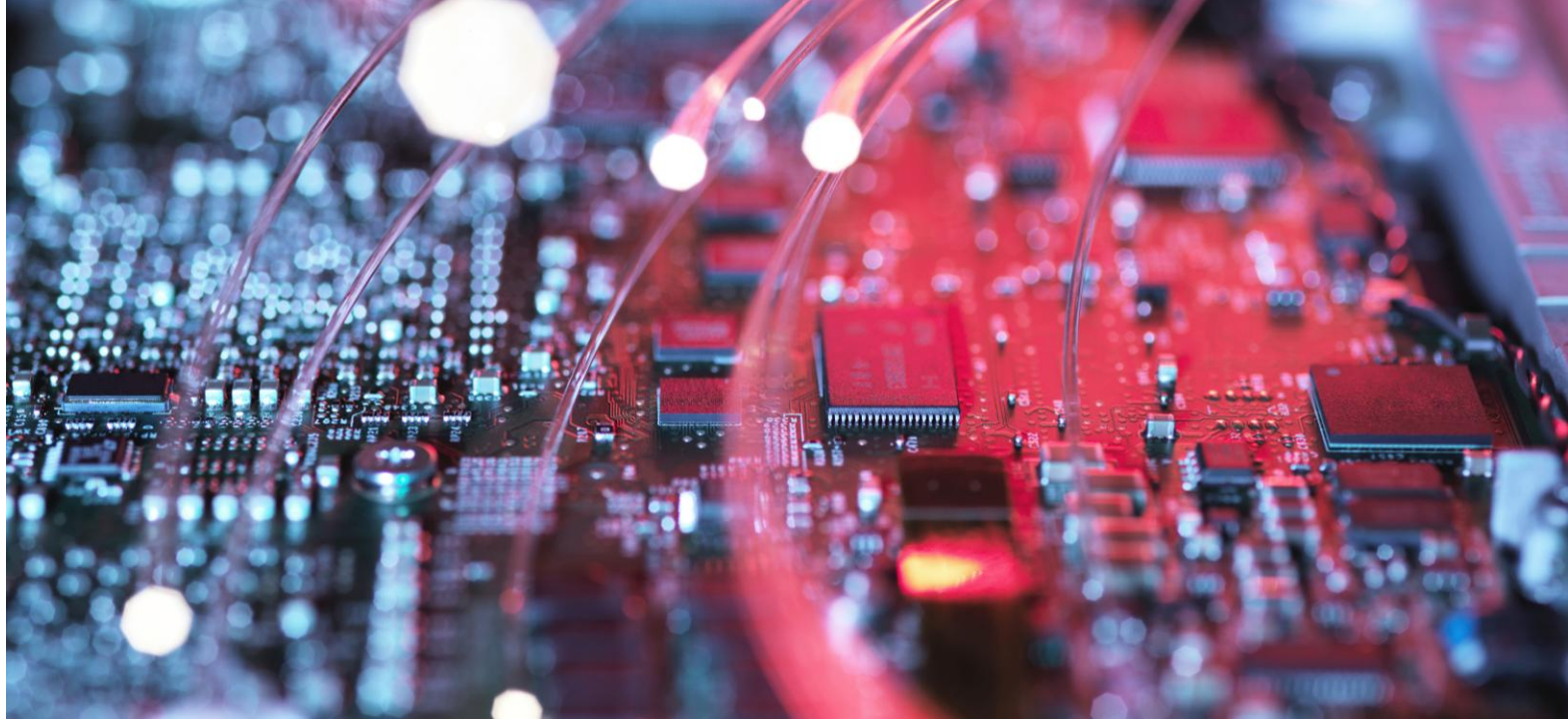
- ACID properties:
- Vital for multi-user environments and concurrent access.

What is a Transaction?

- Definition...
- Purpose...
- Analogy...

Why is Transaction Management Crucial?

1. Data Integrity
2. Reliability
3. Consistency
4. Concurrently
5. Recovery



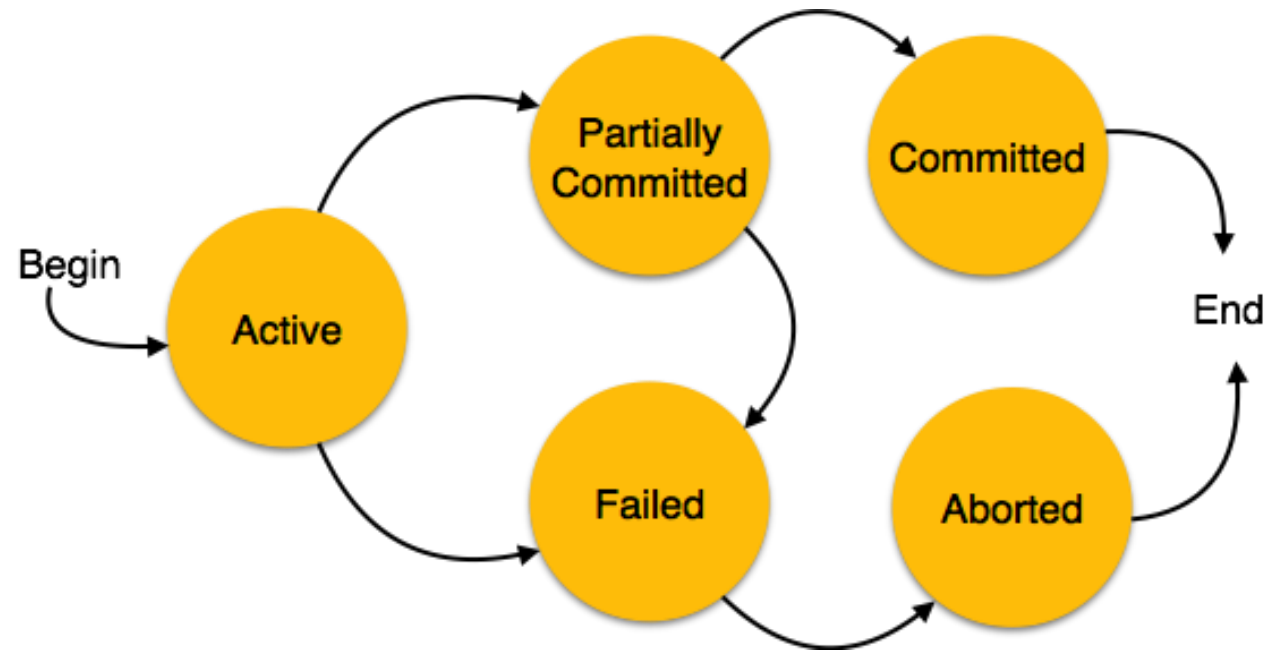
The ACID Properties

A = Atomicity

- All or Nothing

C = Consistency

- From one consistent state to another.



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

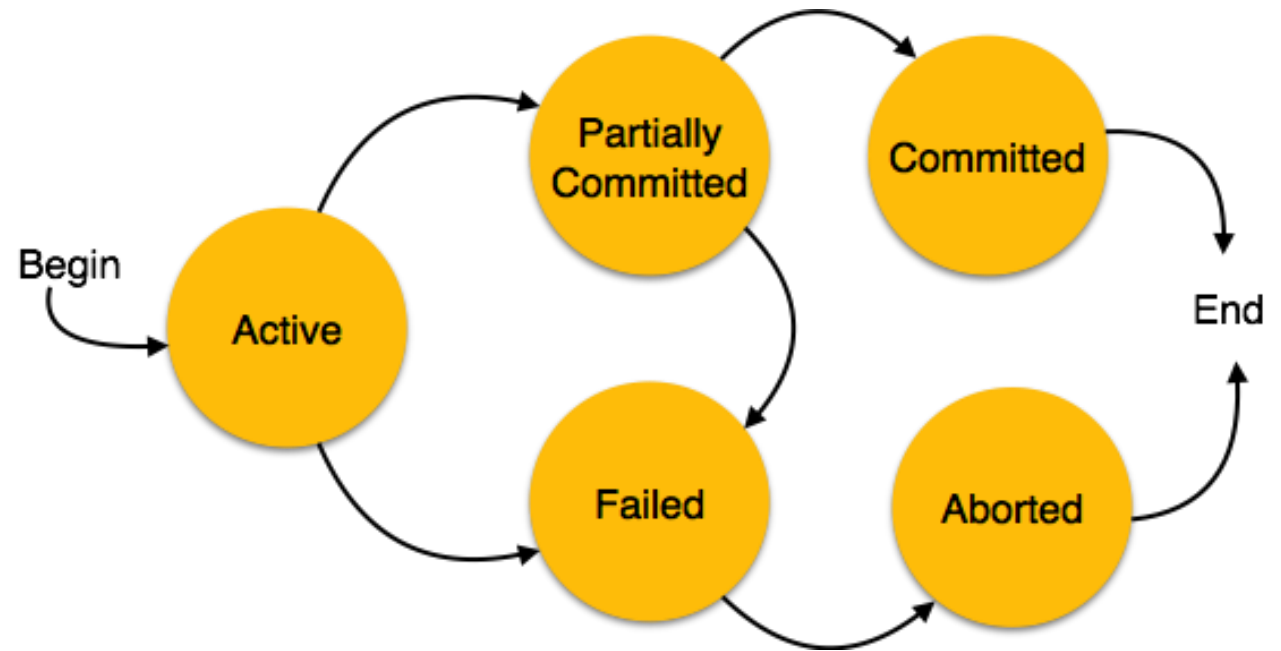
The ACID Properties

D = Durability

- Once successfully committed, changes are permanently stored.

I = Isolation

- Execute concurrent transactions independently

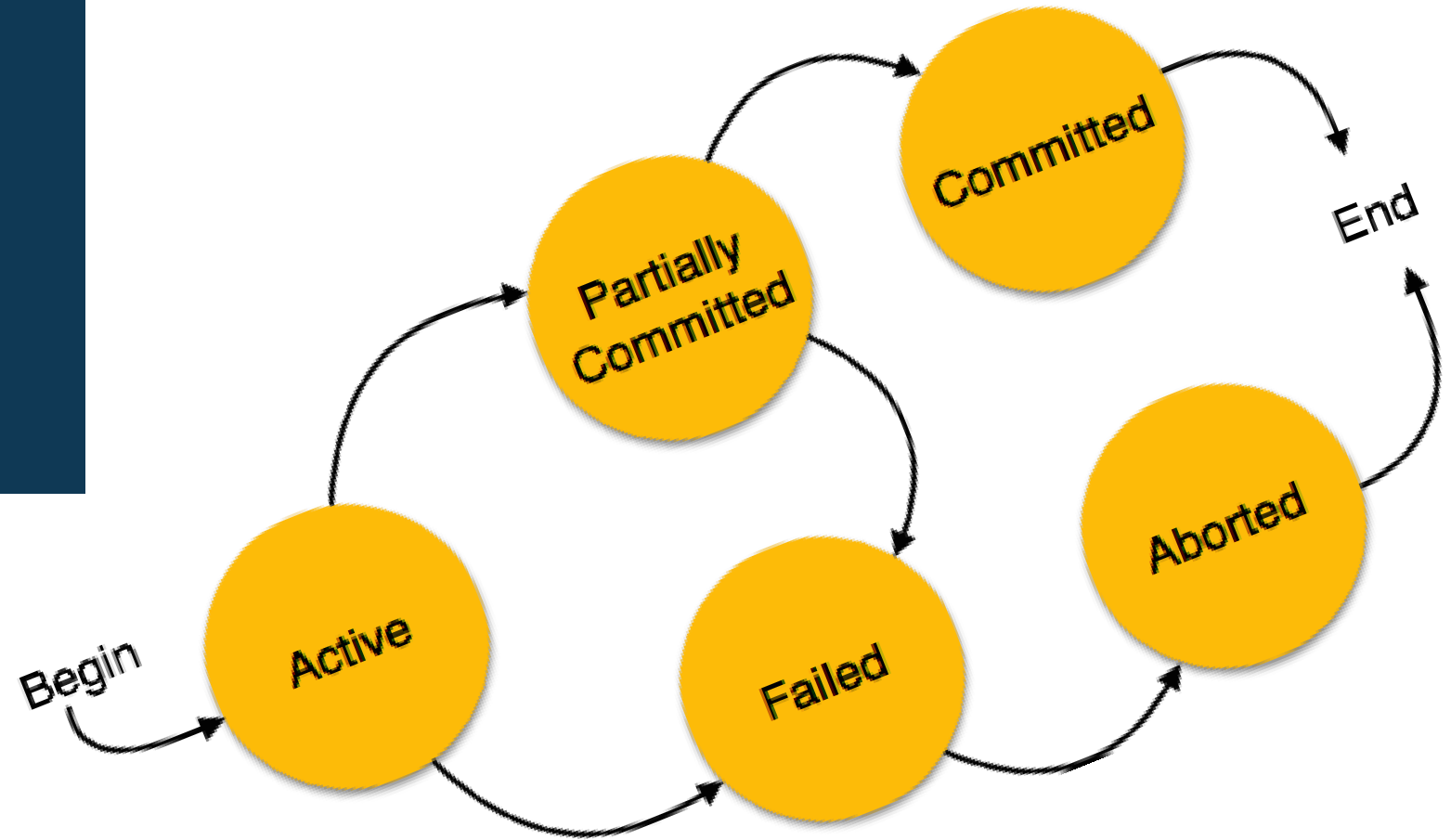


[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

Transaction States (Lifecycle)

A transaction typically progresses through several states:

- Active
- Partially Committed
- Committed
- Failed
- Aborted
- Terminated





Database Security Concepts



1. Define roles and grant permissions based on the principle of least privilege.
2. Access control enforces user-level security.
3. Encryption protects data at rest and in transit.
4. Auditing tracks changes and access to sensitive information.

Why Database Security Matters

- Data is gold
- There are consequences of breaches...
- Treat Landscape

Core Pillars of Database Security:

1. Authentication - Who Are You?
2. Authorization - What Are You Allowed to Do?
3. Data Encryption - Protecting Confidentiality
4. Auditing & Monitoring - Who Did What, When?
5. Data Masking and Redaction - Limiting Exposure
6. Database Hardening - Securing the Environment
7. Backup & Recovery - Ensuring Availability



Layered Security (Defense-in-Depth)

- No Single Solution...
- Combined Approach...
- Continuous Improvement...






Summary and Expectations



- Deliver two logical models:
 - One for the provided unnormalized table.
 - One transformed from your Enhanced ERD.
- Use correct notation for PKs and FKs.
- Normalize to BCNF and explain each step.
- Consider how your model supports security and transaction reliability.

Individual Project

Description

 Phase 3 IP has 2 parts:

Part 1: Analyze the following table (see the Word document called [CS352 - IP3](#)), and reorganize the table into Boyce-Codd Normal Form, at each step describing what is needed to move to the next Normal Form and why each step meets the Normal Form requirements.

Show un-normalized table given and progression through the normal forms up to Boyce Codd in logical data models. Include explanation of how each normal form is met as you progress through the process of breaking down this un-normalized table to tables meeting Boyce Codd normal form.

Part 2: In addition, transform your data model (your EERD created in phase 2 IP) into a logical model, to third normal form. Describe why each table is in third Normal Form. In your logical data model identify the primary keys in each table as **bolded and underlined** and each foreign key as *italicized and underlined*.

Individual Project

Description

Submission for phase 3 IP includes:

- Logical Data Model for the supplied table (Part 1) with a description of how it moved through UNF to 1NF to 2NF to 3NF and Boyce-Codd.
 - Logical Data Model for Part 2 with a description of how each table is in third normal form.
- Add both parts described to the project template section titled "Database Management Systems."
Name the document CS352_<First and Last Name>_IP2.docs, and submit the document for grading.

****ALL SUBMISISON SHOULD BE IN THE FORMAT OF A WORD DOCUMENT (.docx) AND NOT PDF. ALSO, ALL SUBMISSIONS SHOULD BE ENTIRELY DOUBLE-SPACED AND FOLLOW ALL APA 7 RULES FOR FORMATTING, INCLUDING HEADERS, TITLE PAGE, TABLE OF CONTENTS, REFERENCES, AND IN-TEXT CITATIONS.**

Resources

Database Management Systems

A Database Management System (DBMS) is software designed to define, create, maintain, and control access to databases. It acts as an intermediary between users and the database, ensuring that data is consistently organized and remains easily accessible. A DBMS allows for efficient data manipulation through Structured Query Language (SQL), enabling operations such as data retrieval, insertion, updating, and deletion. It also supports key features like concurrency control, transaction management, security enforcement, and data integrity. By centralizing data access and offering tools to manage large volumes of structured information, a DBMS plays a critical role in supporting data-driven decision-making across various industries and applications.

Here are some helpful online resources for students:

Normalization & Logical Modeling Resources

1. IBM's Database Normalization Tutorial

IBM Developer provides real-world examples of normalization through BCNF. It's especially good for reinforcing dependencies and data anomalies with visuals.


2. Stanford University's "Introduction to Databases"

A free online course that includes lectures and quizzes on ER modeling and normalization. The interactive format helps reinforce theory with application.

 <https://databases.stanford.edu>

3. "Database Design" by Adrienne Watt (Open Textbook Library)

Chapter 5 specifically covers normalization. It's well-structured for undergraduate audiences and offers review questions.

 <https://open.umn.edu/opentextbooks/textbooks/database-design>

4. SQLBolt – Visual SQL and Schema Practice

Though focused on SQL, SQLBolt's ER diagram and schema tutorials help students conceptualize relationships before writing code.

 <https://sqlbolt.com/>

INSTRUTOR

DR. JOHN CONKLIN



PHONE

602.796.5972



EMAIL

jconklin@coloradotech.edu



WEBSITE

drjconklin.com