Chapter 3 - Data Modeling using ER model

Unified Modeling Language (UML)

3.1 High level conceptual models for DB Design

→ Steps in designing a database for an application

↳ Requirements gathering

↳ Conceptual design: concise description of the data requirements

↳ Logical design (data model mapping (Database specific)

↳ Physical design → indexes, file organizations, etc.

3.2 Sample Database Application

→ Sample database application is called COMPANY: Keeps track of a company's employees, departments, and projects

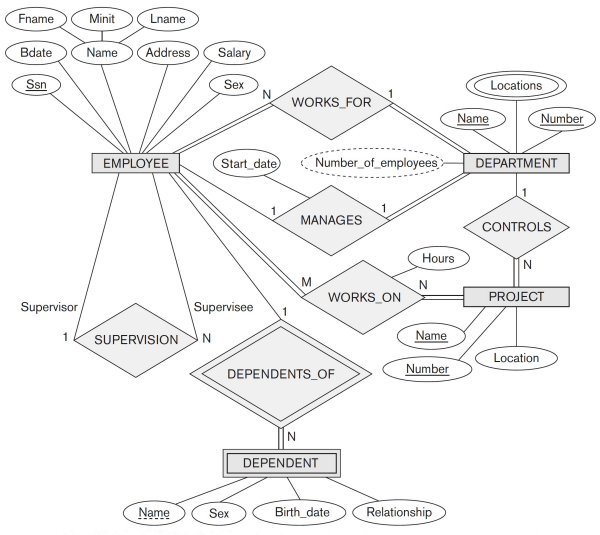
↳ Company organized into departments; each department has unique name, number, and employee who manages the department.

↳ Department controls a number of projects, each with unique name, number, and single location.

↳ DB will track employee's personal details, the (one) department they're assigned to, and the projects they're working on.

↳ DB will keep track of dependents of each employee for tax purposes.

Diagram for entire schema used/described throughout chapter



Summary: DB design phase consists of 4 steps: requirements gathering, conceptual design, logical design, and physical design

Chapter 3 - Data Modeling using ER Model

2018-09-02

→ Some entity types have more than one key attribute

→ An entity type with no key is called a weak entity type.

→ Pure ER notation has no concept of a primary key; that comes later on when discussing relational models.

→ Value set of attributes: range of values a particular entity can have

↳ Attribute "A" of entity set "E" of value set "V" can be expressed as a function like so: $A : E \rightarrow P(V)$

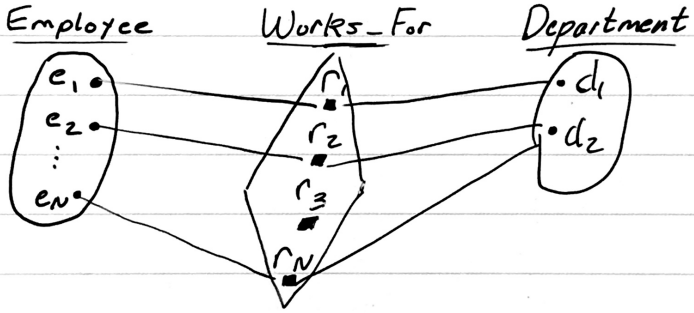
3.4 / 3.4.1 - Relationship Types/Sets/Roles, and Structural Constraints

Relationships: References / Connections between entity types

↳ Relationship types can also have attributes

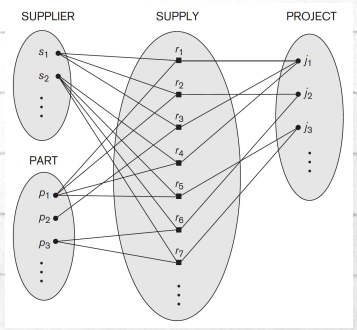
↳ In ER diagrams, relationship types are displayed as diamond-shaped boxes

Example:



3.4.2. Relationship Degree / Role Names / Recursive Relationship

Degree: Number of participating entity types. Degree two is binary, and degree three is ternary.



→ Example of ternary relationship

→ Role name signifies the role that a participating entity plays

→ Self-referencing / recursive relationships require the role to be specified since it can change

Summary: entity can have one or multiple keys, or no keys if its a weak entity type. Relationships between entities are represented by diamonds, and have degrees (number of types included), but are usually binary or ternary.

2018-09-02

Chapter 3 - Data Modeling using ER model

3.3 / 3.3.1 - Entities and Attributes

* Entity: thing or object in the real world with independent existence

Attributes: describe entities (ex: EMPLOYEE is described by name, age, etc.)

Composite / Simple attributes: Composite attributes are divided into smaller parts. E.g. "address" can be divided into City, State, ZIP code, etc.

↳ Attributes that are not divisible are called simple / atomic attributes.

Multi-value vs. single-value attributes: most attributes are single-valued, which means they have only one value for the attribute. Other things are multi-valued, which means they can have multiple options for a given attribute.

Stored vs. derived values: Age and birth-date are examples of derived and stored values, respectively. This is because age can be calculated from the birth-date, whereas birth-date needs to be specified.

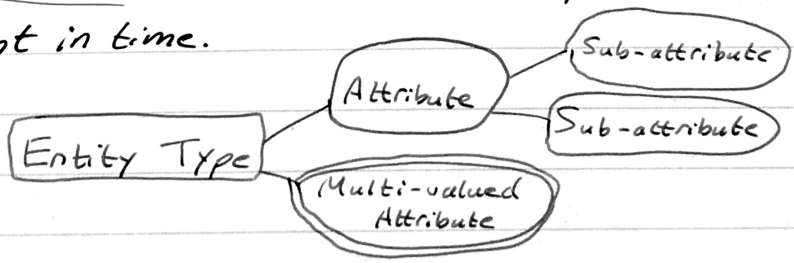
NULL Values: placeholder for when an attribute is either: (1) unknown, or (2) not applicable to every entry in the table.

Complex attributes: Composite or multivalued attributes that are nested arbitrarily.

3.3.2 - Entity Types / Sets / Keys / Value Sets

Entity type: Collection or set of entities with the same attributes

Entity set / collection: Collection of all entities of particular entity type at any point in time.



→ In ER diagrammatic notation, each key attribute has its name underlined inside of the oval

Summary: entities are real-world things with attributes. Attributes can be composite vs. simple, multivalued vs. single-valued, and stored/derived. Collections of entities with same attributes are entity types, and the collection of all entities are entity sets.