Conceptual Design.
The Entity-Relationship (ER) Model

Slides based on “Database Management Systems” 3rd ed, Ramakrishnan and Gehrke
Database Design Overview

- Conceptual design
  - The Entity-Relationship (ER) Model, UML
  - High-level, close to human thinking
  - Semantic model, intuitive, rich constructs
    - Not directly implementable

- Logical Design
  - The relational data model
  - Machine-implementable, fewer and more basic constructs
  - Logical design translates ER into relational model (SQL)

- Physical Design (not in this course)
  - Storage and indexing details
Conceptual Design – ER Model

- What are the *entities* and *relationships* in a typical application?
  - What information about these entities and relationships should we store in the database?

- What are the *integrity constraints* or *business rules*
  - Key constraints
  - Participation constraints

- Representation through *ER diagrams*
  - ER diagrams are then mapped into relational schemas
  - Conversion is fairly mechanical in simple cases, but can be tricky
**Entities and Entity Sets**

- **Entity**: represents a real-world object
  - Characterized using set of attributes
  - Each attribute has a **domain** – similar to variable types

- **Entity Set**: represents collection of similar entities
  - E.g., all employees in an organization
  - All entities in an entity set share same set of attributes

Example from Murach Chap 9 we looked at last time:
- Entities: Vendor, Invoice, LineItem
- Entity Sets: Vendors, Invoices, LineItems
- design → Tables: vendors, invoices, invoice_line_items
Keys

- Each entity set has a **key**

- Set of attributes that uniquely identify an entity (one entity in the entity set)

- Multiple *candidate keys* may exist

- *Primary key* selected among them
Representation Convention:
- Entity sets: rectangles
- Attributes: ovals, with key attributes underlined
- Edges connect entity sets to attributes
- Note that each attribute represents a single value: use of “lot” here means each employee has only one parking lot assigned.
Relationships and Relationship Sets

- **Relationship**: Association among two (or more) entities
  - “Gabriel works in the CS department”
  - Entities here: Gabriel in Employees, CS in Departments
  - Can have descriptive attributes: e.g., “since 9/1/2011”
    - But relationship must be fully determined by entities!
    - i.e., the descriptive attributes can’t disqualify a relationship, only add info about it
  - Binary, ternary or multi-way (n-way) relationships

- **Relationship Set**: Collection of similar relationships
  - Contains $n$-tuples $(e_1, \ldots, e_n)$, where $e_i$ belongs to entity set $E_i$
  - **Instance**: “snapshot” of relationship set at some point in time
    - $(e_1, \ldots, e_n)$, where $e_1 = (Gabriel, CS)$, $e_2 = (Betty, CS)$, …
Visualizing Relationships and Rel. Sets

Edge = Relationship
Set of Edges = Relationship Set

(A, 1)  (B, 1)  (B, 2)  (D, 3)
Relationship Set Representation

Representation Convention:
- Relationship sets: diamonds
- Edges connect relationship sets to entity sets, and relationship sets to relationship set attributes
- E-R with this notation was published by P. Chen in 1976 (at M.I.T. at the time)
- Other notations are also in use: Murach uses “crow’s foot” notation
From Chen’s 1976 paper
A Special Case of Relationship

- An entity set can participate in a relationship set with itself
  - Entities in same set play different *roles* in the relationship
  - *Role indicators* express the role

[Diagram of an entity-relationship model showing the relationship between employees and their supervisors]
Key Constraints

- How many other entities can an entity have a relationship with? (case of binary relationships)

Note: many practitioners lump one-to-many and many-to-one into one category “many-to-one” or N-1 for short, i.e. the many side is not specified by this name. On page 33, “sometimes said to be one-to-many” … “is said to be many-to-one”
Example 1

- **Works_In** relationship: An employee can work in many departments; a dept can have many employees.

  *many-to-many*
Example 2, like Murach’s, in Chen notation

- Member_of relationship: an employee can be in many committees; a committee can have many employee members. *many-to-many*
Example 3

- **Manages relationship:** each dept has *at most one* manager

  *one-to-many*

  from *Employees* to *Departments*, or

  *many-to-one*

  from *Departments* to *Employees*

Arrow for “to-one” direction of key constraint
Participation Constraints
Participation Constraints

- **Total vs Partial Participation**
  - **Total**: every department must have a manager
    - “Departments” entity set has total participation in relationship
    - Represented as thickened line
    - Here we don’t assume each department has exactly one manager, so there is no arrow for a key constraint from Departments to Manages
  
  - **Partial**: not every employee is a manager
    - “Employees” entity set has partial participation, thin line

---

17
Participation and Key Constraints

- **Key constraint**: each dept has *at most one* manager
- **Total participation**: every dept must have a manager
  - “Departments” entity set has total participation in relationship
  - Represented as thickened line
  - The key constraint is represented by the arrow

- **Partial participation**: not every employee is a manager, and no key constraint, since an employee may manage multiple departments here.
  - “Employees” entity set has partial participation, thin line
Manages relationship visualized

Key constraint: each dept has at most one manager
Design Choices in the ER Model

- Should a concept be modeled as an entity or an attribute?
  - Will look at an example: address as entity, attribute

- Should a concept be modeled as an entity or a relationship?
  - Example of enrolls:
    - A student enrolls in a class (enrolls is a relationship) or
    - change enrolls to enrollment and consider an enrollment as an entity itself, related to students and classes
  - Similarly with catalog, etc.
  - Hard cases: hierarchies and inheritance
    - Outside the scope of this class
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
Entity vs. Attribute

- Sometimes **address** may have to be an entity:
  - If we have several addresses per employee (since attributes need to represent single values, i.e., cannot be set-valued)
  - Typically we split address up into parts, either multiple attributes of Employee or attributes of an entity of its own, to allow queries on them
Example

Design a database for a bank, including information about customers and their accounts. Information about customers includes their name, address, phone and SSN. Accounts have numbers, types (e.g., savings/checking) and balances.

1. Draw the E/R diagram for this database.
2. Modify the E/R diagram such that each customer must have at least one account.
3. Modify the E/R diagram further such that an account can have at most one customer.
Mapping ER to Relational Schemas

- For most part, process is mechanical
  - Some special cases arise in the presence of constraints

- Translation from ER to SQL requires:
  - Mapping entity sets to tables
  - Mapping relationship sets to tables
  - Capturing key constraints
  - Capturing participation constraints (not always possible)
CREATE TABLE Employees
    (ssn CHAR(11),
     name CHAR(20),
     lot INTEGER,
     PRIMARY KEY (ssn))
“No-constraints” case follows simple rules

- This means primary keys for the entities are known, but no rules such as “only one manager for a department”

Relationship set becomes a relation, attributes include:

- Keys for each participating entity set (as foreign keys pointing to respective entity table)
- All descriptive attributes for relationship
- Primary key of relationship set table is the composite of primary keys for the entity sets
CREATE TABLE Works_In(
    ssn CHAR(11),
    did INTEGER,
    since DATE,
    PRIMARY KEY (ssn, did),
    FOREIGN KEY (ssn)
        REFERENCES Employees,
    FOREIGN KEY (did)
        REFERENCES Departments)
CREATE TABLE Works_In2(
    ssn CHAR(11), did INTEGER,
    address CHAR(20), since DATE,
    PRIMARY KEY (ssn, did, address),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments,
    FOREIGN KEY (address) REFERENCES Locations)

This means a certain employee can work in various departments and various locations, and a certain employee, when working in a certain department and at a certain location, has been doing so since the since-value.
What if there are Key Constraints?

- Each department has at most one manager, according to the key constraint on Manages represented by the arrow.
Variant 1 for “Each department has at most one manager” key constraint on Manages

- **Variant 1: Map relationship to its own table:**
  - Note that did is the key now!
    - That means we can’t have two rows here for one department, pointing to two managers for that department.
  - Note this allows a department without a manager: it would simply be missing from this table.

```sql
CREATE TABLE Manages(
    ssn CHAR(11), -- manager’s ssn
    did INTEGER,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```
Variant 2 for “Each department has at most one manager” key constraint on Manages

- Since each department has at most one manager, we could instead combine Manages and Departments, i.e., expand Departments a little.
- And use a nullable FK on ssn.
  - A department without a manager would have a null ssn

```sql
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget INTEGER,
    ssn CHAR(11), -- note nullable
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees
)
```

- Note: this is the more common approach in applications because it’s simpler. Normally we’d call this table simply Departments.
Variant 2 for “Each department has exactly one manager” key+participation constraint

- Since each department has at most one manager, we can again combine Manages and Departments.
- i.e., add a column to Departments for the FK to Employees.
- We use a not-null FK on ssn to specify the one manager.

```sql
CREATE TABLE Dept_Mgr ( -- or Departments
  did INTEGER,
  dname CHAR(20),
  budget INTEGER,
  ssn  CHAR(11) NOT NULL,  -- total participation since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```
Participation Constraints and Variant 1

- Does every department have a manager?
  - If yes, the participation of Departments in Manages is *total*
    - Variant 1: has Manages table with all the \((did, ssn)\) links
    - For total participation, every did value in Departments table must appear in a row of the Manages table, but this cannot be controlled in SQL (unless we use complex constraints not available in Oracle)
  - So it turns out that it is **NOT** possible to capture total participation with Variant 1.
  - The Dept_Mgr variant (Variant 2) is the only way to express total participation, as shown on just-previous slide.
  - With powerful CHECK, could put CHECK (exists (select did from manages)) on Departments—but not available on Oracle
Participation Constraints in SQL: “every department has one manager”: the winning design...

CREATE TABLE Dept_Mgr(  -- or simply Departments
did INTEGER,
dname CHAR(20),
budget INTEGER,
ssn CHAR(11) NOT NULL,
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (ssn) REFERENCES Employees
  ON DELETE NO ACTION) --the default
Participation Constraints Summary

- **General case (no key constraints)**
  - Total participation cannot be enforced unless we use complex constraints not supported in Oracle

- **What if there is also a key constraint in place?**
  - If the entity set with total participation also has a key constraint, then it is possible to capture total participation
  - But only if “combined” table construction (Variant 2) is used!
A **weak entity** can be identified uniquely only by considering the key of another (**owner**) entity.

- Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
- Weak entity set must have total participation in this **identifying** relationship set.
- The weak entity set and the identifying relationship set are marked with thickened lines.
- The diamond should have a verb in it: supports? Has? Supports?
- [Wikipedia article](https://en.wikipedia.org) with Orders, OrderItems example
Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
- When the owner entity is deleted, all owned weak entities must also be deleted.

```sql
CREATE TABLE Dep_Policy (  
pname  CHAR(20),
age  INTEGER,
cost  REAL,
ssn  CHAR(11) NOT NULL,
  PRIMARY KEY (pname, ssn),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE CASCADE)
```