BİL 354 – Veritabanı Sistemleri

Entity-Relationship Model
Steps in building a DB application

1. Pick application domain
2. Conceptual design

What data do I need for my application domain?

How can I describe that data?
Steps in building a DB application

1. Pick application domain
2. Conceptual design
   - Convert ER diagram to the data model of your DBMS product
3. Implement application code & user interface
   - SQL & Java/C++/etc + user interface
Entity Relationship (E/R) Model

Entity-relationship (ER) data model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design.
(1) **Requirements Analysis**: What data to store, what the users want from the database

(2) **Conceptual Database Design**: high level description of the data (ER Model)

(3) **Logical Database Design**: convert an ER schema into a relational database schema.
Beyond the ER Model

- ER diagram is just an approximate description of the data, constructed through a very subjective evaluation of the information collected during requirements analysis.

- A more careful analysis can often refine the logical schema obtained at the end of Step 3. Once we have a good logical schema, we must consider performance criteria and design the physical schema.
(4) **Schema Refinement**: analyze the collection of relations in the relational database schema to identify potential problems, and refine it (normalization relations).

(5) **Physical Database Design**: consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria.

(6) **Security Design**: we identify different user groups and different **roles** played by various users
Entity Relationship (E/R) Model

- A popular data model – useful to database designers
  - Graphical representation of miniworld

- E/R design translated to a relational design
  - then implemented in an RDBMS

- Elements of model
  - Entities
  - Entity Sets
  - Attributes
  - Relationships
E/R Model: Entity Sets

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
E/R Model: Entity Sets

- **Entity**: object or “thing”
  - real-world object distinguishable from other objects
  - described by its *attributes*

- **Entity set**: collection of similar entities
  - Similar to a class in object-oriented languages
  - All with same attributes
  - Correspond to class of physical or business objects
    - E.g., Employees, products, accounts, grades, campaigns, etc.

- Represented by a rectangle: Students Courses
E/R Model: Attributes (Nitelikler)

- Attributes are:
  - Properties of entities
    - Like fields in a struct
    - Like columns in a table/spreadsheet
    - Like data members in an object
  - **Domain** – the set of permitted values for each attribute
  - Attribute types:
    - *Simple* and *composite* attributes.
    - *Single-valued* and *multi-valued* attributes
      - E.g. multivalued attribute: *phone-numbers*
    - *Derived* attributes
      - Can be computed from other attributes
      - E.g. *age*, given date of birth
#### Entity Sets *customer* and *loan*

<table>
<thead>
<tr>
<th>customer-id</th>
<th>customer-name</th>
<th>street</th>
<th>city</th>
<th>loan-amount</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>555-55-5555</td>
<td>Jackson</td>
<td>Dupont</td>
<td>Woodside</td>
<td></td>
<td></td>
</tr>
<tr>
<td>244-66-8800</td>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>963-96-3963</td>
<td>Williams</td>
<td>Nassau</td>
<td>Princeton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>335-57-7991</td>
<td>Adams</td>
<td>Spring</td>
<td>Pittsfield</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Slide from Silberschatz. et al.*
Relationships (İlişkiler – Bağıntılar)

- Connect two or more entity sets
  - e.g. students *enroll* in courses
  - Binary relationships: connect two entity sets
    - most common
  - Multiway relationships: connect several Entity Sets

- Represented by diamonds:
E/R Model: Relationships

- Students *Enroll* in courses
- Courses are *Held* in rooms
- The E/R data model:

A relationship can also have “descriptive attributes”.
### Relationship Set: borrower

#### Customer Table:
<table>
<thead>
<tr>
<th>Phone</th>
<th>Name</th>
<th>Street</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>Jackson</td>
<td>Dupont</td>
<td>Woodside</td>
</tr>
<tr>
<td>244-66-8800</td>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>963-96-3963</td>
<td>Williams</td>
<td>Nassau</td>
<td>Princeton</td>
</tr>
<tr>
<td>335-57-7991</td>
<td>Adams</td>
<td>Spring</td>
<td>Pittsfield</td>
</tr>
</tbody>
</table>

#### Loan Table:
<table>
<thead>
<tr>
<th>Loan ID</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>L-15</td>
<td>1500</td>
</tr>
<tr>
<td>L-14</td>
<td>1500</td>
</tr>
<tr>
<td>L-19</td>
<td>500</td>
</tr>
<tr>
<td>L-11</td>
<td>900</td>
</tr>
<tr>
<td>L-16</td>
<td>1300</td>
</tr>
</tbody>
</table>
Relationship Sets (Cont.)

Slide from Silberschatz. et al.
Relationship Sets with Attributes

- **customer-name**
- **customer-street**
- **access-date**
- **account-number**
- **balance**

![Diagram of relationship sets with attributes](image-url)

Slide from Silberschatz. et al.
A little set theory

- A mathematical *set* is a collection of *members*
- A set is defined by its members
  - No other structure, no order, no duplicates allowed
- Sets specified by listing:
  - \( \{1, 2, 3, \ldots\} = S \)
- Or by “set-builder” notation:
  - \( \{ x \in S : 2 \text{ divides } x \} = ? \)
  - \( \{ x \in \text{Presidents} \mid \text{reelected}(x) \} = ? \)

- One set can be a *subset* of another (which is a *superset* of it)

- Given two sets \( X \) and \( Y \), the *cross product* or *Cartesian product* is \( X \times Y = \{(x,y) : x \in X, y \in Y\} = \text{the set of all ordered pairs} \)
  - Important: \((x,y) \neq \{x,y\}\)
  - In an order pair or tuple
    - Order matters; duplicates are allowed
A little set theory

- Mathematically, a relation between X and Y is just a subset of \( X \times Y \) = all those pairs \((x,y)\) s.t. \(x\) is related to \(y\)

- Example: owner-of \(O\) on People, Cats
  - \(O(\text{Ali, Tekir})\) holds

- The equals relation \(E\) on \(N, N\):
  - \(E(3,3)\) holds
  - \(E(3,4)\) does not hold
  - \(E\) is still a set: \(E = \{(1,1), (2,2), (3,3), \ldots\}\)

- Father-of relation \(F\) on People, People:
  - \(F(\text{father, son})\) holds
  - \(F(\text{son, father})\) does not hold
  - Relations aren’t necessarily symmetric
Relationships

If $A$, $B$ are sets, then a **relation** $R$ is a subset of $A \times B$.

$A = \{1, 2, 3\}$  $B = \{a, b, c, d\}$  
$R = \{(1,a), (1,c), (3,b)\}$

**makes** is a subset of **Product** $\times$ **Company**:
Multiplicity of Relationships

Representation of relationships

- No arrow: many-to-many
- Sharp arrow: many-to-one
- Rounded arrow: “exactly one”
  - “key constraint”
- One-one:
Representing “Multiplicity”

- Show a many-one relationship by an arrow entering the “one” side.

- Show a one-one relationship by arrows entering both entity sets.

- Rounded arrow = “exactly one,” i.e., each entity of the first set is related to exactly one entity of the target set.
Multiplicity of Relationships

Many-to-many:

Many-to-one: a student living in a residence hall

Many-to-exactly-one: a student must live in a residence hall
One-To-Many Relationship

In the one-to-many relationship a loan is associated with at most one customer via borrower, a customer is associated with several (including 0) loans via borrower.

Slide from Silberschatz. et al.
Many-To-One Relationships

In a many-to-one relationship a loan is associated with several (including 0) customers via borrower, a customer is associated with at most one loan via borrower
Many-To-Many Relationship

- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower

Slide from Silberschatz. et al.
E-R Diagram with a Ternary Relationship

Slide from Silberschatz. et al.
We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint.

E.g. an arrow from *works-on* to *job* indicates each employee works on at most one job at any branch.

If there is more than one arrow, there are two ways of defining the meaning.

- E.g a ternary relationship $R$ between $A$, $B$ and $C$ with arrows to $B$ and $C$ could mean
  1. each $A$ entity is associated with a unique entity from $B$ and $C$ or
  2. each pair of entities from $(A, B)$ is associated with a unique $C$ entity, and each pair $(A, C)$ is associated with a unique $B$ entity,

Each alternative has been used in different formalisms. To avoid confusion we outlaw more than one arrow.
Degree of a Relationship Set

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are *binary* (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
  
  **E.g.** Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets employee, job *and branch*
  
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
E/R Diagram

- Students
  - Name
  - ID

- Enrolls
  - TA

- Courses
  - Name
  - ID

- Assisting

- TA
  - ID
  - Name
E/R Diagrams

- OK if each TA is a TA of all students
  - Student and TA connected only through Course

- But what if students were divided among multiple TAs?
  - Then a student in C20.0046 would be related to only one of the TA's for C20.0046—which one?
  - Schema doesn’t store enough info

- 3-way relationship is helpful here
Multiway Relationships

Enrolls entries:

<table>
<thead>
<tr>
<th>Students</th>
<th>Courses</th>
<th>TAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condi</td>
<td>C20.0046</td>
<td>Donald</td>
</tr>
<tr>
<td>George</td>
<td>C20.0046</td>
<td>Dick</td>
</tr>
<tr>
<td>Alberto</td>
<td>C20.0046</td>
<td>Colin</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Enrolls determines TA: 

(student, course) \(\rightarrow\) at most one TA
What if we need an entity set twice in one relationship?

Product

Purchase

salesperson

buyer

Person

Person

the “role”

Store
Roles

- Entity sets of a relationship need not be distinct.
- The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship.
Roles in relationships

- Entity set appears more than once in a relationship
  - Generally distinct entities
- Each appearance is in a different role
- Edges labeled by roles

<table>
<thead>
<tr>
<th>Pre-req</th>
<th>Course</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting</td>
<td>Finance-I</td>
<td></td>
</tr>
<tr>
<td>Finance-I</td>
<td>Derivatives</td>
<td></td>
</tr>
<tr>
<td>Finance-I</td>
<td>Finance-II</td>
<td></td>
</tr>
<tr>
<td>Calculus</td>
<td>Derivatives</td>
<td></td>
</tr>
</tbody>
</table>
Keys

A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.

A *candidate key* of an entity set is a minimal super key.

*Customer-id* is candidate key of *customer*

*account-number* is candidate key of *account*

Although several candidate keys may exist, one of the candidate keys is selected to be the *primary key*.

-We must designate a key for every entity set.
Underline the key for each entity set

Product

- name
- category
- price

multi-attribute keys are okay!

Person

- name
- ssn
- address

Multiple “candidate keys”? Pick just one to be the key.

Is this a good key?
Example: a Multi-attribute Key

- Note that hours and room could also serve as a key, but we must select only one key.
Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g. A ternary relationship parents, relating a child to his/her father and mother, is best replaced by two binary relationships, father and mother
  - Using two binary relationships allows partial information (e.g. only mother being know)

- But there are some relationships that are naturally non-binary
  - E.g. works-on
Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace $R$ between entity sets $A$, $B$ and $C$ by an entity set $E$, and three relationship sets:
    1. $R_A$, relating $E$ and $A$
    2. $R_B$, relating $E$ and $B$
    3. $R_C$, relating $E$ and $C$
  - Create a special identifying attribute for $E$
  - Add any attributes of $R$ to $E$
  - For each relationship $(a_i, b_i, c_i)$ in $R$, create
    1. a new entity $e_i$ in the entity set $E$
    2. add $(e_i, a_i)$ to $R_A$
    3. add $(e_i, b_i)$ to $R_B$
    4. add $(e_i, c_i)$ to $R_C$
Converting multiway relationships to binary

- Binary relationships are as strong as multiway
- Replace relationship with *connecting entity set* and multiple binary relationships

Enrolls has no attributes!
Second multiway e.g.: renting movies

- Scenario: a *Customer Rents a Movie from a VideoStore on a certain date*

- date should belong to the *fact* of the renting
  - Relationship attribute
Second multiway e.g.: renting movies

- Convert to binary?

```
<table>
<thead>
<tr>
<th>VideoStore</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rental</td>
<td></td>
</tr>
<tr>
<td>Customer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MovieOf</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>StoreOf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BuyerOf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Store</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
</tr>
</thead>
</table>
```
Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.

- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - Identifying relationship depicted using a double diamond

- The **discriminator (or partial key)** of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.

- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set’s discriminator.
Weak Entity Sets

- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.
- *payment-number* – discriminator of the *payment* entity set
- Primary key for *payment* – (loan-number, payment-number)
More Weak Entity Set Examples

- In a university, a *course* is a strong entity and a *course-offering* can be modeled as a weak entity.

- The discriminator of *course-offering* would be *semester* (including year) and *section-number* (if there is more than one section).

- If we model *course-offering* as a strong entity we would model *course-number* as an attribute. Then the relationship with *course* would be implicit in the *course-number* attribute.
Sometimes your entity might not seem to have a key

**Weak entity set**: some or all of its key attributes come from other classes to which it is related.
Example: Weak Entity Set

- **name** is almost a key for football players, but there might be two with the same name.
- **number** is certainly not a key, since players on two teams could have the same number.
- But **number**, together with the team **name** related to the player by **Plays-on** should be unique.
In E/R Diagrams

- Double diamond for supporting many-one relationship.
- Double rectangle for the weak entity set.

Note: must be rounded because each player needs a team to help with the key.
Design a database for a bank, including information about customers and their accounts. Information about a customer includes their name, address, phone, and Social Security number. Accounts have numbers, types (e.g., saving, checking) and balances. Also record the customer(s) who own an account.

1) Draw the E/R diagram for this database.
2) Change your diagram so an account can have only one customer.
3) Further change your diagram so a customer can have only one account
4) Change your original diagram in (1) so that a customer have a set of addresses.
Subclasses in the E/R model

- Some ESs are special cases of others
- Conversely: some are generalizations
  - Mammals, humans, students, grad students
  - NB: These aren’t *members* but subclasses

- Subclass A isa B
  - Represented by a triangle
  - Root is more general
Subclasses

movies

stars

Voices

length

title

year

isa

Cartoons

Movies
Keys in ISA Hierarchy

- In an Isa hierarchy, only the root entity set has a key, and it must serve as the key for all entities in the hierarchy.
Aggregation

Consider the ternary relationship works-on, which we saw earlier.

Suppose we want to record managers for tasks performed by an employee at a branch.

Slide from Silberschatz. et al.
Aggregation (Cont.)

- Relationship sets *works-on* and *manages* represent overlapping information
  - Every *manages* relationship corresponds to a *works-on* relationship
  - However, some *works-on* relationships may not correspond to any *manages* relationships
    - So we can’t discard the *works-on* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity
- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager
E-R Diagram With Aggregation

- **employee**
- **job**
- **branch**
- **manages**
- **manager**

Slide from Silberschatz. et al.
E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.
DB Design Principles

1. Model your domain faithfully
2. Avoid redundancy.
3. Limit the use of weak entity sets.
4. Don’t use an entity set when an attribute will do.
Faithfulness

- Is the relationship many-many or many-one?
- Are the attributes appropriate?
- Are the relationships applicable to the entities?

Examples:
- Courses & instructors
  - maybe many-one, maybe many-many
- Bosses & subordinates
  - maybe one-many, maybe many-many
ER Design Principle #1: Model your domain faithfully
Principle #2: Avoid redundancy

- **Redundancy** = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.
Example: Good

This design gives the address of each manufacturer exactly once.
Example: Bad

This design states the manufacturer of a toy twice: as an attribute and as a related entity.
Example: Bad

This design repeats the manufacturer’s address once for each toy and loses the address if there are temporarily no toys for a manufacturer.
Principle #3: Don’t overuse entity sets

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.
    or
  - It is the “many” in a many-one or many-many relationship.
**Example: Good**

- **Manfs** deserves to be an entity set because of the nonkey attribute **addr**.
- **Toys** deserves to be an entity set because it is the “many” of the many-one relationship **ManfBy**.
Example: Good

If there is no manf address, there is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.
Example: Bad

Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.
Principle #4: Don’t Overuse Weak Entity Sets

- Beginning database designers often make most entity sets weak, supported by all other entity sets to which they are linked.

- In reality, we usually create unique ID’s for entity sets.
  - Examples include social-security numbers, automobile VIN’s etc.
When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID’s.

- **Example**: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.
More Examples
Avoiding redundancy

- Example: spot the redundancy

Redundancy: Movies “knows” the studio two ways
### Spot more redundancy

Different redundancy: studio info listed for every movie!
Don’t add relationships that are implied

Suppose each course again has $\leq 1$ TA

Q: Is this good design?

A: probably not

…unless the Assist relationship is *not* implied by Enrolls+TA-of
Still more redundancy

Q: What’s wrong with this design?

A:
- Repeating TA names & IDs – redundant
- TA is not TAing any course now ➔ lose TA’s data!
- TA should get its own ES
Related issue: entity or attribute?

- Some E/Rs improved by removing entities

- Can convert Entity E into attributes of F if
  1. R: F → E is many-one (or 1-1)
  2. Attributes for E are *mutually independent*
     - knowing one att val doesn’t tell us another att val

- Then
  - remove E
  - add all attributes of E to F
Entity ➔ attribute

Students → Enrolls → Courses

TA-Name → TA → Assists

Course-ID
CName
Room

Students → Enrolls → Courses

Course-ID
CName
Room

TA-Name
Convert TA entity again?

- No! Multiple TAs allowed \(\rightarrow\) redundant course data
- Violates condition (1)

<table>
<thead>
<tr>
<th>CName</th>
<th>CID</th>
<th>Room</th>
<th>TA-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS</td>
<td>46</td>
<td>123</td>
<td>Howard</td>
</tr>
<tr>
<td>DBMS</td>
<td>46</td>
<td>123</td>
<td>Wesley</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
No! TA has dependent fields $\rightarrow$ redundant TA data

Violates condition (2)

- How can it tell?

<table>
<thead>
<tr>
<th>CName</th>
<th>TA-Name</th>
<th>TA-ID</th>
<th>TA-Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS</td>
<td>Ralph</td>
<td>678</td>
<td>Green</td>
</tr>
<tr>
<td>A.Soft.</td>
<td>Ralph</td>
<td>678</td>
<td>Green</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>