WEEK 1

Advantages of Database Approach

- program-data independence
- planned data redundancy
- improved data consistency
- improved data sharing
- increased application development productivity
- enforcement of standarts
- improved data quality
- improved data accessibility and responsiveness
- reduced program maintenance
- improved decision support

Database Types: OLTP (Online Transaction Processing) - Operational (Relative), OLAP (Online Analtytical Processing) - Informational (Analysis Reporting)
Operational: Dynamic (Constantly)
Informational: Static (Weekly)
Software Development Life Cycle: Planning, Analysis, Design, Implementation, Maintenance

Evolution of Database Systems

- Need for program data-independence in order to reduce maintenance
- Desire to manage more complex data types and structures
- Ease of data access for less techinal personnel
- Need for more powerful decision support platforms

Entitity (Example, Student, Course)
Relation (One student can take many courses)
Attribute (Student Name)
Entitity Types

- Strong (one line), independent, has its own unique identifier
- Weak (two line), dependent on a strong entitiy (identifying owner); cannot exists on its own, does not have a unique identifier (only a partial identifier), connected to strong entity with double lines

Associative (rounded line)
Relationship Degrees

- Unary (relation to itself)
- Binary (two entities)
- Ternary (multiple entities)

Attributes

- Identifier
- Partial Identifier (Double underline) Partial identifier = identifier + some attribute
- Optional
- [Derived]
- {Multivalued}
- Composite (..)

A Good Business Rule Is:

- Declerative - what
- Precise - clear
- Atomic - one statement
- Consistent - internally and externally
- Expressible - structured, natural language
- Distinct - non-redundant
- Business-oriented - understood by business people

A Good Data Name is:

- Related to buisness, not technical, characteristics
- Meaningful and self-documenting
- Unique
- Readable
- Composed of words from an approved list
- Repetable
- Written in standart syntax

An Entitiy Should Be

- An object that will have many instances in the database
- An object that will be composed of multiple attributes
- An object that we are trying to model

An Entitiy Should Not Be

- A user of the database system
- An output of the database system (e.g., a report)

Guidelines for Naming Entities

- Singular noun
- Specific to organization
- Concise, or abbreviation
- For event entities, the result not the process
- Name consistent for all diagrams

Guidelines for Defining Entities

"An X is..."

- Describe unique characteristics of each instance
- Explict about what is and is not the entity
- When an instance is created or destroyed
- Changes to other entitiy types
- History that should be kept

Week 2

Attribute: property or characteristic of an entitiy or relationship type

Classifications of Attributes:

1. Required versus Optional
2. Simple versus Composite
3. Single-Valued versus Multivalued
4. Stored versus Derived
5. Identifier

Required Attribute: Must have a value for every entity

Optional Attribute: May not have a value for every entity

Composite Attribute: An attribute that has maningful component parts (sub-attributes) (ex: address)
Multi Valued Attribute: Has values more than one
Derived: Values can be calculated from related attribute values (not physically stored in the database), Years employed calculated from date employed and current date.
Identifier: an attribute (or combination of attributes) that uniquely identifies individual instances of an enitity type, Simple versus Composite Identifier.
Composite Identifier: more attributes together (Full name + age...)
Candidate Identifier: an attribute that could be an identifier; it satisfies the requirements for being an identifier.
Criteria for Identifiers

1. Choose Identifiers that: Will not change in value, Will not be null
2. You should avoid using intelligent identifiers (ex: containing locations or people that might change)
3. Substitute new, simple keys for long, composite keys

Naming Attributes

1. Name should be a singular noun or noun phrase
2. Name should be unique
3. Name should follow a standart format
4. Similar attributes of different entitity types should use the same qualifiers and classes.

Defining Attributes

1. State source of values
2. State whether attribute value can change once set
3. Specify whether required or optional
4. State min and max number of occurences allowed
5. Indicate relationships with other attributes

Modeling Relationships

1. Relationship Types vs Relationship Instances: The relationship type is modeled as lines between entity types, the relationship instance is between specific entity instances.
2. Relationships can have attributes: These describe features pertaining to the association between the entities in the relationship
3. Two entities can have more than one type of relationship between them (multiple relationships)
4. Associative Entity: combination of relationship and entity

Cardinality of Relationships

1. One to One
2. One to Many
3. Many to Many

Relationships can be defined as optional or mandatory. (O for optional to the line, I for mandatory to the line)
If you start with zero, it has to be "An x may has"
If you start with one, it has to be "An x has"
If it is "one to one", it has to be "an x has exactly one"
If it is "zero to one", it has to be "an x may have one"
Cardinality Constraints: the number of instances of one entity that can or must be associated with each instance of another entity
Minimum Cardinality:

1. If zero, then optional
2. If one or more, then mandatory

Associative Entities

1. All relationships for the associative entitity should be many
2. The associative entity could have meaning independent of the other entities
3. The associative entity preferably has a unique identifier, and should also have other attributes
4. The associatve entity may participate in other relationhips other than entities of the associated relationship
5. Convert tenrary relationships to associative entities

Uppercase naming for the entities

Lowercase naming for the attributes

Independent identifier is really important for the associative entities

Time stamp: a time value that is associated with a data value, often indicating when some event occured that affected the data value.

Week 3

EER Model: extends original E-R model with new modelling constructs

Subtype: A subgrouping of the entities in an entity type that has attributes distinct from those in other subgroupings

Supertype: A generic entity type that has a relationhip with one or more subypes

Attribute Inheritance:

1. Subtype entities inherit values of all attributes and relationships of the supertype
2. An instance of a subtypes also an instance of the supertype

Relationships at the supertype level indicate that all subtypes will participte in the relationship

The instances of a subtype may participate in a relationship unqiue to the subtype. In this situation, the relationship is shown at the subtype level.

Generalization: The process of defining a more general entity type from a set of more specialized entity types. Bottom-up

Specilization: The process of defining one or more subtypes of the supertype and forming supertype/subtype relationships. Top-down

Completeness Constrains: Whether an instance of a supertype must also be a member of at least one subtype

1. Total Specialization Rule: Yes (double line): you cannot be super type, you have to be sub type of the super type.
2. Partial Specialization Rule: No (single line): you can be super type

Disjointness Constraints: Whether an instance of a supertype may simultaneously be a member of two (or more) subtypes.

1. Disjoint Rule: An instance of the supertype can be only ONE of the subtype (d)
2. Overlap Rule: An instance of the supertype could be more than one of the subtypes (o)

Subtype Discriminator: An attribute of the supertype whose values determine the target subtypes.

1. Disjoint: a simple attribute with alternative values to indicate the possible subtypes. (ex: AttributeType =)

2. Overlapping: a composite attribute whose subparts pertain to different subtypes. Each subpart contains a boolean value to indicate whether or not the instance belongs to the associated subtype (ex: AttributeType:) (Make sure that this is written compositely)
Entity Cluster: Set of one or more entity types and associated relationships grouped into a single abstract entity type

Week 4
Definition: create, alter, drop
Manipulation: select, insert, update, delete
Control: grant, deny, revoke
select field_list m
from table_list
group by group_list
having group_condition
order by ordering
- select ssn from Employee
- select ssn, firstName, lastName from Employee
- select ssn, firstName, lastName from Employee Where ssn=101
- select ssn, firstName, lastName from Employee Where firstName='Ali' and lastName='Yılmaz'
- select ssn as Social_Security_Number, firstName + ' ' + lastName from Employee Where gender='F' and salary>=600 and salary<=700
- select ssn as Social_Security_Number, firstName + ' ' + lastName as "Full Name" from Employee Where gender='F' and salary>=600 and salary<=700
- select ssn as Social_Security_Number, firstName + ' ' + lastName as [Full Name] from Employee Where gender='F' and salary>=600 and salary<=700
- select ssn, dno from Employee Where dno=4 or dno=6
- select ssn, dno from Employee Where dno in (4,6)
- select ssn, dname from Department, Employee
- select ssn, managerSsn from Department, Employee Where managerSsn=ssn
You don't have to use select ssn if only managerSsn is being asked.
- select ssn, firstName, dName from Department, Employee Where Employee.dno=Department.dno (This is for blocking cartesian multiplication in queries.)
- select ssn, firstName, dName from Department, Employee Where Employee.dno=Department.dno and dName = 'Marketing'
- select e.ssn, e.firstName, e.dName from Department as d, Employee as e Where e.dno=d.dno and d.dName = 'Marketing'
Using "as" keyword is using aliases.
- select e.ssn, e.firstName, e.dName from Department as d, Employee as e Where e.dno=d.dno
inner join: kesişim
left outer join: A
right outer join: B
full outer join: birleşim
- select e.ssn, e.firstName, e.dName from Employee e left outer join Department e on e.dno=d.dno
- select e.ssn, e.firstName, e.dName, s.ssn, s.firstName, s.dName from Employee e, Employee s where on s.ssn=e.ssn

- select e.ssn, e.firstName, e.dName, s.ssn, s.firstName, s.dName from Employee e left outer join Employee s on on s.ssn=e.ssn
- select * from Employee
- select e.Pno, e.firstName, e.dName, s.ssn, s.firstName, s.dName from Employee e left outer join Employee s on on s.ssn=e.ssn

Nested Query

- select e.firstName, e.firstName from Employee e where e.Dno in (select d.dno from Department d where d.dName in ('Test1', 'Test2'))
- select e.firstName, e.firstName from Employee e, (select d.dno from Department d where d.dName in ('Test1', 'Test2')) d2 where e.Dno=d2.Dno

Distinct Keyword

- select distinct e.firstName from Employee e, (select d.dno from Department d where d.dName in ('Test1', 'Test2')) d2 where e.Dno=d2.Dno

Union

- select p.Pno Project p inner join Department d on p.Dno=d.Dno.... union select w.Pno....
- select * from Employee e where e.Address like '%Kadıköy%'
- select * from Employee e where e.Address like '_a__a'