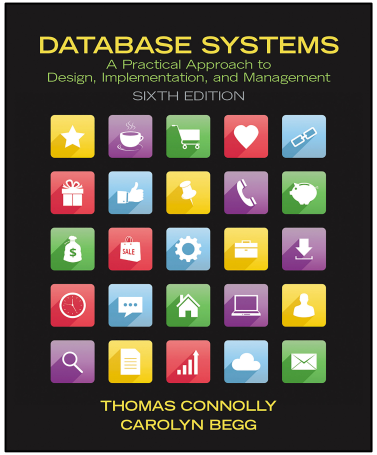
# The Three Phases of Database Design

Topic 2 Lesson 4

# Chapter 16 Connolly and Begg



## Database design

1. Conceptual Database Design
2. Logical Database Design
3. Physical Database Design

## Conceptual Database Design

## Step 1: Conceptual database design

Entity relationship modeling (ER) creates a diagrammatic representation of a conceptual model

Conceptual database design: The process of constructing a model of the data used in an enterprise, independent of **all** physical considerations.

Process must identify the necessary entities, relationships and attributes

## Conceptual database design

1. Identify the entity types
2. Identify the relationship types
   a) Check for fan and chasm traps
3. Identify the attributes
4. Identify the attribute domains (not represented in ERD)
5. Identify the candidate keys and primary key
6. Apply generalization (is-a), aggregation (has-a), composition (part-of)
7. Check model for redundancy

Covered

8. Validate conceptual model against user transactions
9. Review model with user

Not Covered

# Identify entity types

1. Identify the nouns in the user requirement specification
2. Entities should be major objects NOT properties of other objects.
3. Objects that have existence in their own right
4. Look for entity types that may be synonyms of each other
    a. Document the synonyms
5. All entity names should be well descriptive
6. Document the identified entity types in the data dictionary

## Identify relationship types

1. Identify the verbs in the user requirement specification
2. Classify relationships as complex, binary or recursive.
3. Determine the multiplicity of each relationship
4. Check for fan and chasm traps
5. Document and assign meaningful names to the relationships

## Identify entity and relationship attributes

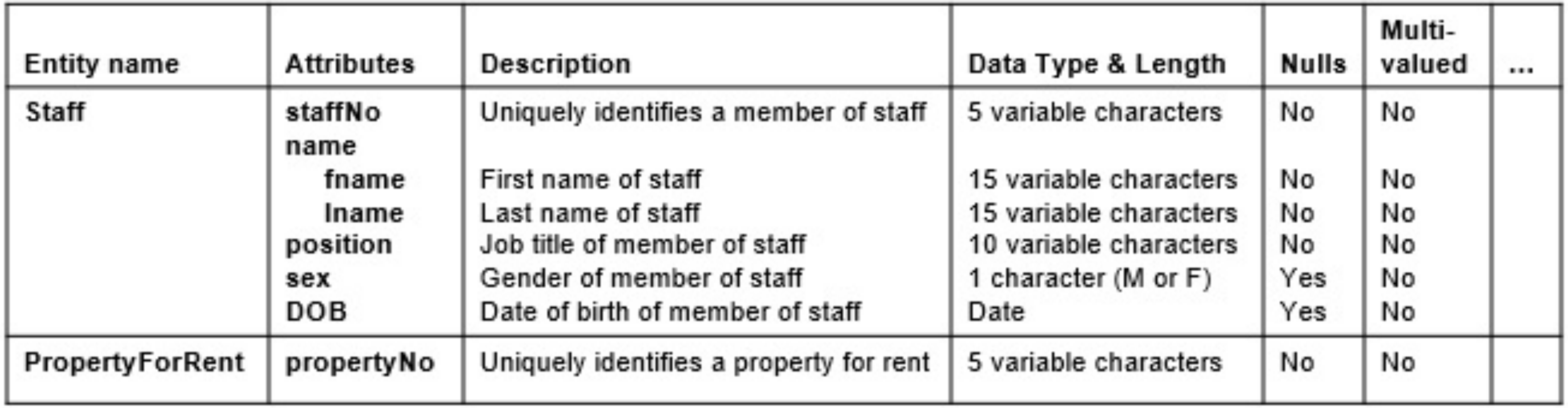
1. Identify the properties or the qualities of the entity types
2. Classify each attribute as:
   a. Simple versus composite attribute
   b. Single versus multi-valued attribute
   c. Derived attribute (ensure attribute can be derived from given attributes)
3. Document the attributes

## Identify domains for the attributes

The conceptual designer should gather as much detail on the types of values expected for each attribute. This detail is stored in a data dictionary file to be shared with the users of the data. The data type constraints as well as the size of data can be stored in the data dictionary.

However, remember actual data types cannot yet be determined and should not be represented in the conceptual design.

# Data dictionary example



| Entity name | Attributes | Description | Data Type & Length | Nulls | Multi-valued | ... |
|---|---|---|---|---|---|---|
| Staff | staffNo | Uniquely identifies a member of staff | 5 variable characters | No | No | |
| | name | | | | | |
| | fname | First name of staff | 15 variable characters | No | No | |
| | lname | Last name of staff | 15 variable characters | No | No | |
| | position | Job title of member of staff | 10 variable characters | No | No | |
| | sex | Gender of member of staff | 1 character (M or F) | Yes | No | |
| | DOB | Date of birth of member of staff | Date | Yes | No | |
| PropertyForRent | propertyNo | Uniquely identifies a property for rent | 5 variable characters | No | No | |

## Determine candidate, primary keys

1. Identify the candidate keys
2. Choose the primary key from the candidate keys that are:
   a. Candidate key with the minimal set of attributes
   b. Candidate key that is least likely to be updated
   c. Candidate key with the fewest number of bytes
   d. Candidate key with the lowest maximum value
   e. Candidate key that is easiest to manipulate for a user.
3. All other candidate keys are designated as alternate keys
4. Be willing to add new attributes that provide uniqueness if the current candidate keys are composite
5. Make sure that keys are properly identified for weak entity

# EER to represent hierarchical relationships

1. Generalization (IS-A) allows us to represent super and subclasses for an entity type.
   a. Participation - all members of the superclass must fall into a subclass {Mandatory | Optional}
   b. Disjoint - subclasses do not share members {And | Or}
2. Composition (Part-of) allows us to represent an entity type that composes another entity type (strong ownership).
3. Aggregation (Has-a) allows us to represent  an entity type that has a collection of another entity type

## Check model for redundancy

1. Review 1-1 relationships to ensure the entity types are really different entity types and not synonyms.
2. Remove redundant relationships: relationships that provide the same information as another relationship.
   a. Multiple paths between entity types are a potential source for redundancy
3. Consider time and its effect on each relationship
   a. Some relationships may seem redundant but really are necessary due to changes in relationships due to time

## Validate conceptual model with transactions

1. The conceptual data model must provide a response for all user defined transactions.
2. If the model cannot provide an answer, the conceptual model is  not complete.
3. Two methods that use two different representations of data model:
   a. Textual description of the user transaction
   b. Transaction pathway through the conceptual model to retrieve response for the transaction
4. Check for fan and chasm trap

## Review conceptual model with user

1. Must get sign-off from the user that the model capture all necessary data.
2. Implies user has verified all transactions can be answered

## Summary (Conceptual Design)

Creating a conceptual design is an iterative process, where your goal is to produce an unambiguous representation of the data domain and its processes.

The goal is communication between a technical team and a nontechnical teal to ensure the proposed technical solution fulfills the needs of the enterprise.

# Practice problem: musicians

Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database.

Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians do not have cell phones, often share the same address, and no address has more than one landline phone. Given their limited use, cell phones are not tracked.

Each instrument used in a song recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).

Each album recorded on the Notown label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.

Each song recorded at Notown has a title and an author. The author of a song is a musician. There is 1 and only 1 author per song.

Each musician may play several instruments, and a given instrument may be played by several musicians.

Each album has a number of songs on it, but no song may appear on more than one album.

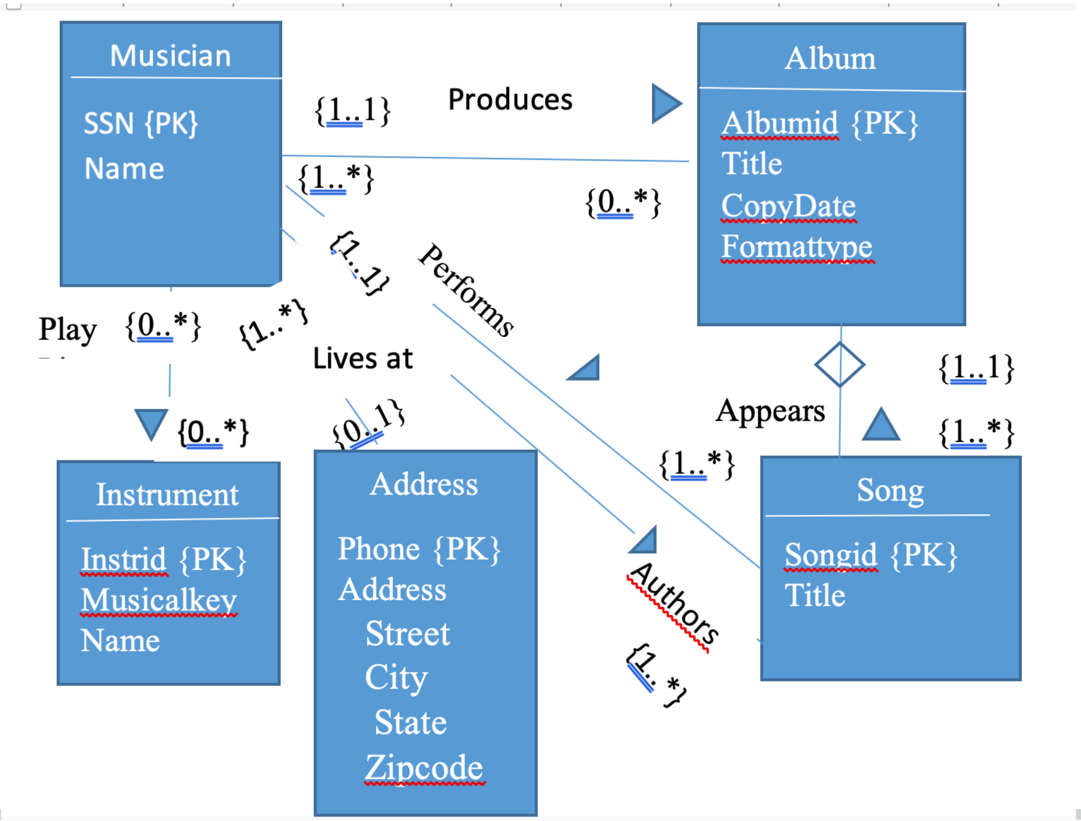
Each song is performed by one or more musicians, and a musician may perform a number of songs.

Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

Design a conceptual schema for Notown and draw an UML diagram for your schema. Be sure to indicate all key and multiplicity constraints and any assumptions you make.

# Conceptual design

- We need both authors and performs between musician and song to capture both relationships
- Alternatively authors and producers could have been a subclass of musician
- We chose to make address a separate entity since many musicians may live at the same address


Musician
SSN {PK}
Name
Album
Albumid {PK}
Title
CopyDate
Formattype
Produces
{1..1}
{0..*}
{1..*}
{1..1}
Performs
Play
{0..*}
{1..*}
Lives at
{0..*}
{0..1}
Appears
{1..1}
{1..*}
{1..*}
Instrument
Instrid {PK}
Musicalkey
Name
Address
Phone {PK}
Address
Street
City
State
Zipcode
Song
Songid {PK}
Title
Authors
{1..*}