Models to Schema & Normal Form Refinement

Kathleen Durant PhD

CS 3200

ER Process

Where to start

- To make an ER model from a verbal description you need to identify
  - Entities
  - Attributes
  - Relationships
  - Multiplicity

General guidelines

- Since entities are things or objects they are often nouns in the description
- Attributes are facts or properties, and so are often nouns also
- Verbs often describe relationships between entities

Conceptual model (ER) steps

- **Identify entity types**
- **Identify relationship types**
- **Identify and associate attributes with entity or relationship types**
- **Determine attribute domains**
- **Determine candidate, primary, and alternate key attributes**
- **Consider use of enhanced modeling concepts (optional step)**
- **Validate conceptual model against user transactions**
- **Review conceptual data model with user**

Logical database design for the relational model

- **Derive relations for logical data model**
- Validate relations using normalization
- Validate relations against user transactions
- Define integrity constraints

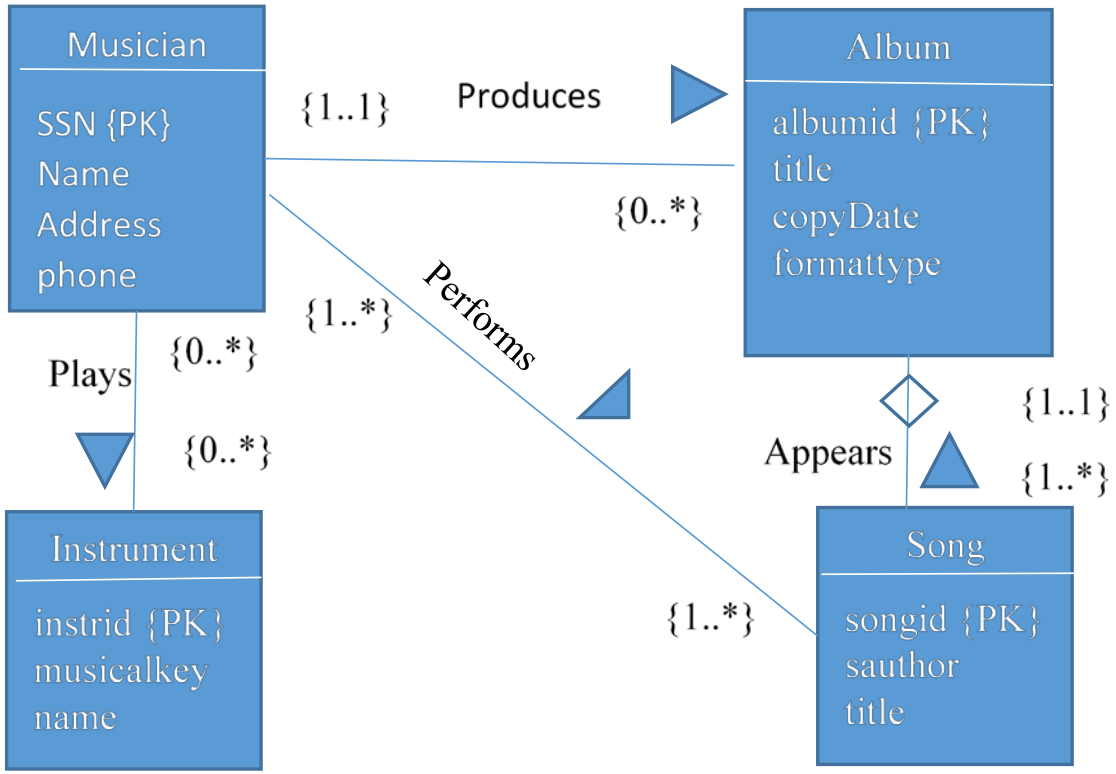
UML Model to Relational Schema

Entity/Relationship	Mapping
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1:* binary relationship	Post primary key of entity on 'one' side to act as foreign key in relation representing entity on 'many' side. Any attributes of relationship are also posted to 'many' side.
1:1 binary relationship:	
(a) Mandatory participation on both sides	Combine entities into one relation.
(b) Mandatory participation on one side	Post primary key of entity on 'optional' side to act as foreign key in relation representing entity on 'mandatory' side.
(c) Optional participation on both sides	Arbitrary without further information.
Superclass/subclass relationship	See Table 16.1.
: binary relationship, complex relationship	Create a relation to represent the relationship and include any attributes of the relationship. Post a copy of the primary keys from each of the owner entities into the new relation to act as foreign keys.
Multi-valued attribute	Create a relation to represent the multi-valued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.

Representation of superclass / subclass relationship

Participation constraint	Disjoint constraint	Relations required
Mandatory	Nondisjoint {And}	Single relation (with one or more discriminators to distinguish the type of each tuple)
Optional	Nondisjoint {And}	Two relations: one relation for superclass and one relation for all subclasses (with one or more discriminators to distinguish the type of each tuple)
Mandatory	Disjoint {Or}	Many relations: one relation for each combined superclass/subclass
Optional	Disjoint {Or}	Many relations: one relation for superclass and one for each subclass

Classwork: Convert UML to a relational schema



The six basic steps: data model to relational model

Step 1: Identify the data elements (UML model)

Step 2: Subdivide each element into its smallest useful components

Step 3: Identify the tables and assign columns

Step 4: Identify the primary and foreign keys

Step 5: Review whether the data structure is normalized

Step 6: Identify the indexes

SCHEMA REFINEMENT

What's wrong with this table?

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Data redundancy of the the branch data

Data Redundancy and Update Anomalies

- **Relations that contain redundant information may potentially suffer from update anomalies.**

- **Types of update anomalies include**
  - **Insertion**
  - **Deletion**
  - **Modification**

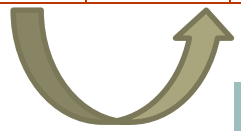
Data Redundancy

- *Redundancy* is at the root of several problems associated with relational schemas:
  - Redundant storage
  - Insert/delete/update anomalies
- Integrity constraints, in particular *functional dependencies*, can be used to identify schemas with such problems and to suggest *schema refinements*.
- Role of functional dependencies (FD)s in detecting redundancy:
  - Consider a relation R with 3 attributes, ABC.
    - No FDs hold: There is no redundancy.
    - Given A → B: Several tuples can have the same A value, and if so, they'll all have the same B value (**Redundancy**)
- Schema refinement technique: *decomposition* (replace relation ABC with, say, AB and BC, or AC and AB).

Example: R determines W in table {S,N,L,R,W,H}

- Social Security #, Name, Lot, Rating, Wage, Hours per week

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40



dependency

- FDS S → {S,N,L,R,W,H} AND R → W
- Problems due to R → W :
  - *Update anomaly*: Can we change W in just the 1st tuple of SNLRWH?
  - *Insertion anomaly*: What if we want to insert an employee and don't know the hourly wage for his rating?
  - *Deletion anomaly*: If we delete all employees with rating 5, we lose the information about the wage for rating 5

Example Solution

2 smaller tables removes update anomolies

REPLACE:

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

WITH:

Hourly_Emps2

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Wages

R	W
8	10
5	7

Normal Form: Codd's Objectives

- Free the collection of relations from undesirable insertion, update and deletion **dependencies**
  - If schema has duplicated data in multiple rows
    - Forced to update/delete all copies of a piece of data
    - How do you know you got all copies of it?
- Reduce the need for restructuring the collection of relations
  - Build an extensible design now as oppose to later
- Make the relational model more informative to users
  - Cleaner model should be easier to understand
- Make the collection of relations neutral to the query statistics
  - Designed for general purpose querying

First Normal Form

- Tuples in a relation must contain the same number of fields
- The domain of each attribute is atomic
- The value of each attribute contains only a single value
- No attributes are sets
  - No repeating groups

Levels of Normal Form

- Level 1: **No repeating entities or group of elements**
  - Do not have multiple columns representing the same type of entity
  - Primary key that represents the entity
- Example: Table mother (MotherName varchar(40), child1 varchar(20), child2(varchar(20)…child8 varchar(20))
  - CREATE 3 tables: Mother, Children and Offspring
    - Offspring links Mother and Children together
  - OR CREATE 2 tables where the Child relation has a foreign key to the Mother relation

2 table solution

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Mother Id	Mother Name	Child1	Child2	Child3	Child4
1	Elsa	Mary	Alice	NULL	NULL
2	Golda	George	Fred	NULL	NULL
3	Viola	Ava	NULL	NULL	NULL
4	Iris	Kayla	NULL	NULL	NULL
5	Daisy	Harry	NULL	NU	

- Create Table Mother, and a Table Child
- Create a foreign key that links Child to Mother

Mother Id	Mother Name
1	Elsa
2	Golda
3	Viola
4	Iris
5	Daisy

Child Id	Name	Mother
11	Mary	1
12	Alice	1
13	George	2
14	Fred	2
15	Ava	3
16	Kayla	4
17	Harry	5

3 table solution

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Mother Id	Mother Name	Child1	Child2	Child3	Child4
1	Elsa	Mary	Alice	NULL	NULL
2	Golda	George	Fred	NULL	NULL
3	Viola	Ava	NULL	NULL	NULL
4	Iris	Kayla	NULL	NULL	NULL
5	Daisy	Harry	NULL	NULL	NULL

- Create Table Mother, Table Offspring and a Table Children
- Link them together via a unique representation (social security number)

Parent Id	Offspring Id
1	11
1	12
2	13
2	14
3	15
4	16
5	17

Mother Id	Mother Name
1	Elsa
2	Golda
3	Viola
4	Iris
5	Daisy

Offspring Id	Offspring Name
11	Mary
12	Alice
13	George
14	Fred
15	Ava
16	Kayla
17	Harry

Benefits of 1NF

- Minimize duplicated data
- Beneficial when you want to extend your database by adding more concepts
- Example: Say you now want to model the father relationship ?
- With the unnormalized NF solution you are forced to duplicate all of the offspring data in the father relation

Adding the Father Relation

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Mother Id	Mother Name	Child1	Child2	Child3	Child4
1	Elsa	Mary	Alice	NULL	NULL
2	Golda	George	Fred	NULL	NULL
3	Viola	Ava	NULL	NULL	NULL
4	Iris	Kayla	NULL	NULL	NULL
5	Daisy	Harry	NULL	NULL	NULL

NOT FIRST NORMAL FORM (1NF) – DUPLICATES ENTITIES					
Father Id	Father Name	Child1	Child2	Child3	Child4
21	Sam	Mary	Alice	Fred	NULL
22	Sal	George	NULL	NULL	NULL
23	Hal	Ava	NULL	NULL	NULL
24	Ed	Kayla	NULL	NULL	NULL
25	George	Harry	NULL	NULL	NULL

- Forced to duplicate child data in both mother and father relationship
- Leads to errors in child data during updates and deletions
- Hard to query child data
- Limits schema
  - 5 children?

1NF with Father Relation

Father Id	Father Name
21	Sam
22	Sal
23	Hal
24	Ed
25	George

Mother Id	Mother Name
1	Elsa
2	Golda
3	Viola
4	Iris
5	Daisy

Offsring Id	Name	Father	Mother
11	Mary	1	21
12	Alice	1	21
13	George	2	22
14	Fred	2	21
15	Ava	3	23
16	Kayla	4	24
17	Harry	5	25

Family Table contains Mapping between mother and offspring and father and offspring

Second normal form

- Schema must be in first normal form
  - You have eliminated group sets
  - Every tuple has a unique key
- **Each field not in the primary key provides a fact about the entity represented via the (entire) primary key**
  - The primary key must be minimal – no extra fields
  - No partial dependency on part of the primary key
    - Only applies to composite primary key
- Helps you identify a relation that may represent more than one entity
- **All fields must be functionally dependent on the complete primary key**

Example 2NF vs. Not 2NF

1st Normal Form but NOT 2ndNORMAL FORM				
Mother Id	First Name	Last Name	Hospital	Hospital Address
1	Elsa	General	BIDMC	Boston
2	Golda	Major	MGH	Boston
3	Viola	Funt	TMC	Cambridge
4	Iris	Batter	BIDMC	Brighton
5	Daisy	Mae	Mayo	Allston

2nd NORMAL FORM			
Mother Id	First Name	Last Name	Hospital Id
1	Elsa	General	1
2	Golda	Major	2
3	Viola	Funt	3
4	Iris	Batter	1
5	Daisy	Mae	4

	2nd NORMAL FORM	
Hospital ID	Hospital	Hospital Address
1	BIDMC	Boston
2	MGH	Boston
3	TMC	Cambridge
4	Mayo	Allston

3rd Normal Form

- **No dependencies between 2 non-key attributes**
- Typically the form most database developers strive to be at

- Bill Kent quote:

  Every non-key attribute must provide a fact about the key, the whole key and nothing but the key

Example 3NF vs. Not 3NF (but 2NF)

2nd NORMAL FORM				
Mother Id	First Name	Last Name	Hospital Id	Room Number
1	Elsa	General	1	36
2	Golda	Major	2	48
3	Viola	Funt	3	36
4	Iris	Batter	1	41
5	Daisy	Mae	4	32

2nd or 3rd NORMAL FORM		
Hospital ID	Hospital	Hospital Address
1	BIDMC	Boston
2	MGH	Boston
3	TMC	Cambridge
4	Mayo	Allston

3rd NORMAL FORM			
Mother Id	First Name	Last Name	Registration Id
1	Elsa	General	1
2	Golda	Major	2
3	Viola	Funt	3
4	Iris	Batter	4
5	Daisy	Mae	5

3rd Normal Form		
Registration Id	Hospital Id	Room Id
1	1	36
2	2	48
3	3	36
4	1	41
5	4	32

Normal Form Tips

- Review your attributes in your tables and ensure that they are facts about the complete key and only the complete key
- No duplicating groups in a table
- Split many to many relationships up into 2 many to 1 relationships by identifying the relation that maps them together

Normal Form Summary

Normal form	Description
First (1NF)	The value stored at the intersection of each row and column must be a scalar value, and a table must not contain any repeating columns.
Second (2NF)	Every non-key column must depend on the entire primary key.
Third (3NF)	Every non-key column must depend only on the primary key.

Note

- Most designers stop at the third normal form.

The benefits of normalization

- More tables, and each table has an index on its primary key. That makes data retrieval more efficient.
- Each table contains information about a single entity. That makes data retrieval and insert, update, and delete operations more efficient.
- Each table has fewer indexes, which makes insert, update, and delete operations more efficient.
- Data redundancy is minimized, which simplifies maintenance and reduces storage.

MODEL TO SQL SCHEMA

Example to model

A university consists of a number of departments. Each department offers several majors. A number of courses make up each major. Students declare a particular major and take courses towards the completion of that major. Each course is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students

Example: Entities

- A **university** consists of a number of **departments**. Each department offers several **majors**. A number of **courses** make up each **major. Students** declare a particular major and take courses towards the completion of that major. Each course is taught by a **lecturer** from the appropriate department, and each lecturer tutors a group of students

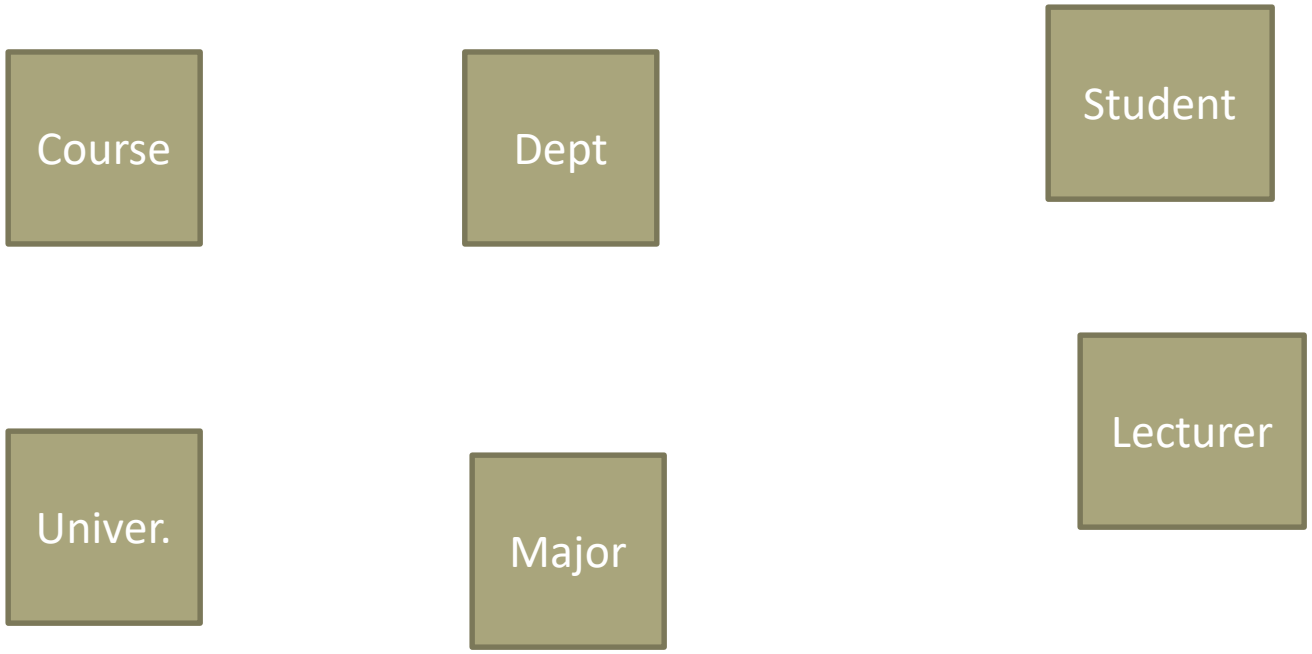
Example: Relationships

- A **university** consists of a number of departments. Each department **offers** several **majors**. A number of **courses** **make up** each major. **Students** **declare** a particular major and **take** courses towards the completion of that major. Each course is **taught** by a **lecturer** from the appropriate department, and each lecturer **tutors** a group of students

Entities

How do we add:

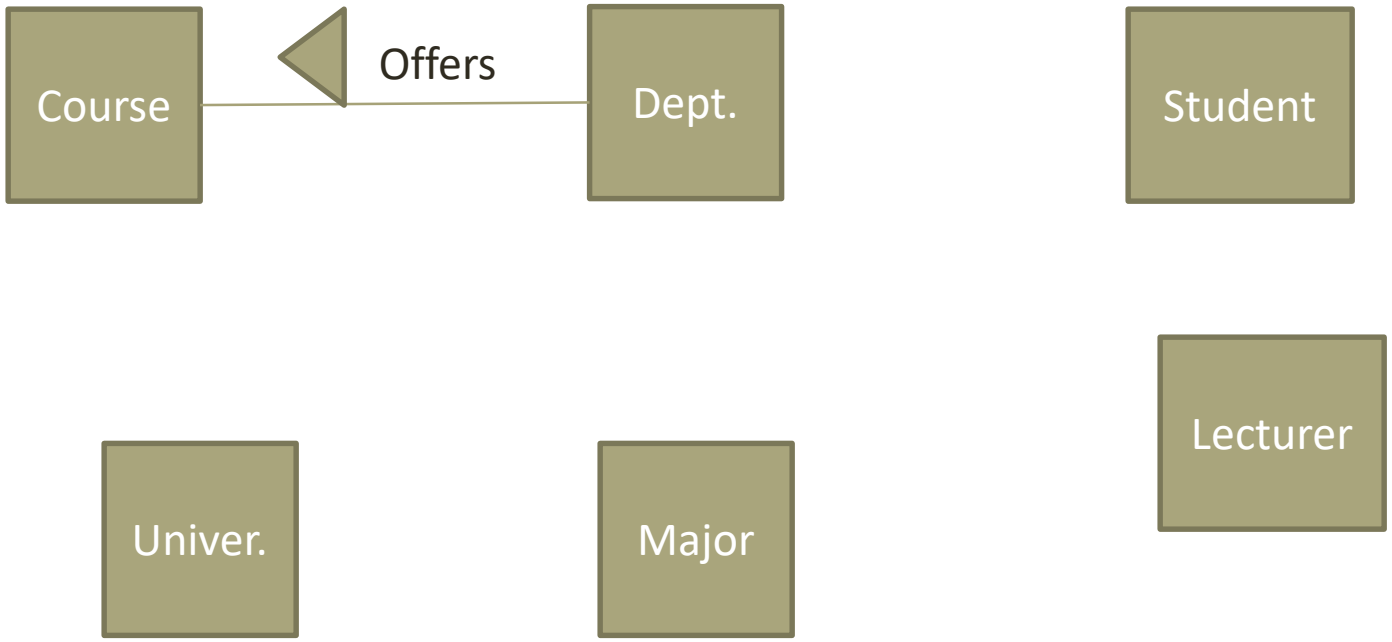
Department offers several majors



Relationships

How do we add relationships:

Department offers several majors



Model to SQL Tables?

- Identify the tables for the entities
- Identify the tables for the relationships
- Identify the table name, field names and data types
- Identify the primary keys
- Identify the foreign keys
  - Determine behavior for DELETE/UPATE operations
- Represent other column and table constraints
  - NULL allowed for field?
  - Default value for a field?