The Relational Model
From ER model to Tables

CS1F

IM Lecture 5

Craig Macdonald

Database design lifecycle

2

- Requirements analysis
  - User needs; what must database do?
- Conceptual design
  - High-level description; often using E/R model
- Logical design
  - Translate E/R model into (typically) relational schema

- Schema refinement
  - Check schema for redundancies and anomalies
- Physical design/tuning
  - Consider typical workloads, and further optimise

Overview

3

- The Relational Model
- Understanding Entities & Relationships as 'Tables' in a database
- Converting your diagram into tables
- Introduction to MySQL Workbench

- Thursday
  - Enforcing integrity
  - More on the relational model

Reminder - Data Modelling

4

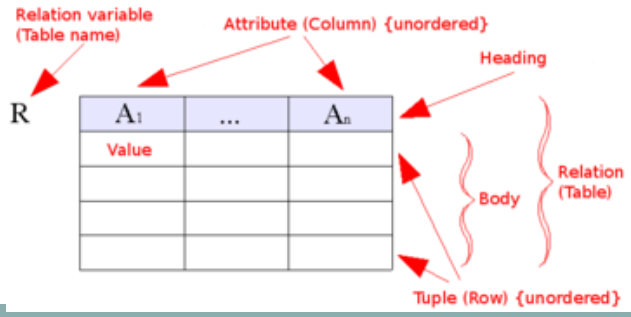
- ER Model allowed us to establish the relationships and dependencies amongst the information
- We now need to arrange the data into a **logical structure** of **relations**
- The logical structure can then be mapped into the storage objects supported by the database – i.e. **tables**

Entities → Tables

5

Roughly, a table (relation) is constructed for each item of interest in a DB

A relation equates (approximately) to an entity type (or some form of relationship) in the ER diagram.



The Heading

6

- All relations must have a heading
  - Name of relation
    - Student
  - Names of columns of relation (the attributes)
    - Name, student ID, exam1, exam2

STUDENT (Name, Student ID, exam1, exam2)

The number of attributes determines the DEGREE of the relation

Relations → Schema

8

- A **relation schema** is a set of attributes
  - written R (A_1, $A_{2,...}A_n$) e.g.
  - Student (name: Text, matric: Number, ex1: Number, ex2: Number)
- Each attribute in a relation schema has a **domain**
- A **relational database schema** is a set of these relation schemas

Domains

9

- Domains are a lot like Data Types in programming
  - Defines the set of values that can be assigned to an attribute
  - Determines the range of allowable operations on each value
    - Add, subtract, concatenate…….

Domains

10

- A **domain** is a set of atomic values that can be assigned to an attribute
- A domain has two aspects:
  - its meaning - e.g. the set of matriculation numbers
  - its format - e.g. a integer in range 0…9999999
- Different DBMS offer different sets of domains:
  - ***MS Access*** offers: Text, Number, Memo, Date/Time, Currency, AutoNumber, Yes/No, etc. - `NOT SQL STANDARD`
  - ***MySQL*** offers **`standard`** SQL types: CHAR (fixed length strings), VARCHAR (variable length strings), INT, FLOAT, DATE...

Quick Reference: (My)SQL Data Types

12

Data Type	Description	Examples
INT	Integer number	1, 5, -100
FLOAT, DOUBLE	Floating point number	-1.1, 5, 6e10
BOOLEAN	Boolean	1, 0
(MySQL doesn't have BOOLEAN) TINYINT(1)	*Integer with only 1 bit*	*1, 0*
CHAR(x)	Fixed length string of length x	'A '
VARCHAR(x)	Variable length string upto length x	'A'
DATE, TIME, TIMESTAMP	Various date/time data types	'2016-01-01 00:00:00.000000'

See also:
http://dev.mysql.com/doc/refman/5.7/en/data-type-overview.html

CONVERTING YOUR ER DIAGRAM TO TABLES

13

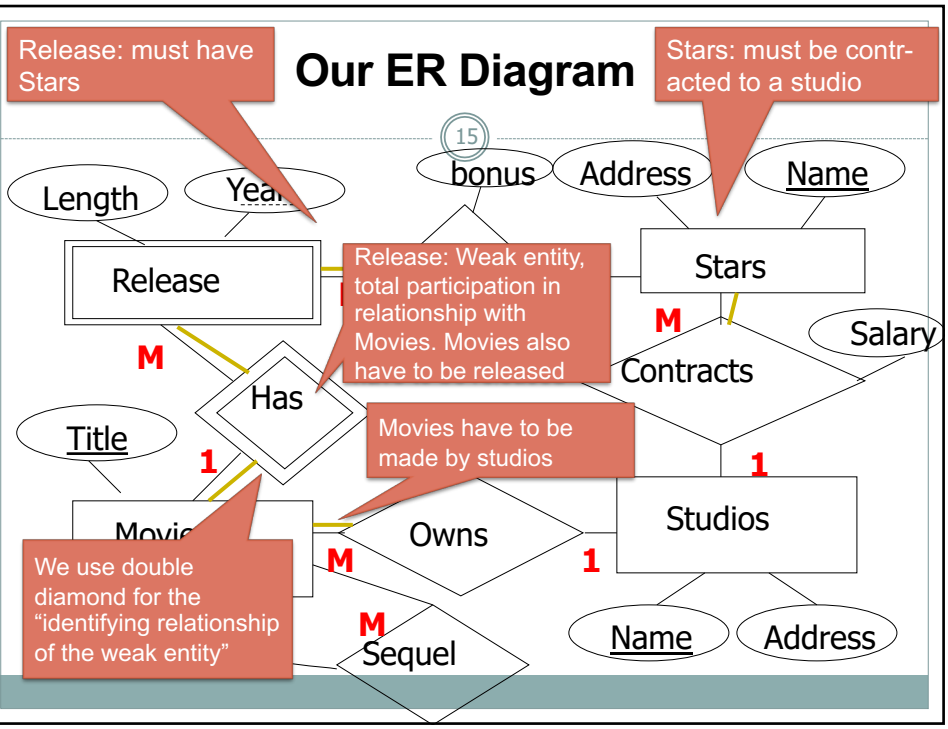
Translating E-R to relational schema

14

1. **Entities and their simple attributes**
2. Weak entities and their simple attributes
3. 1-M relationships (and their attribute)
4. 1-1 relationships (and their attributes)
5. M-N relationships (and their attributes)
6. Composite attributes
7. Multivalued attributes

Our ER Diagram

15

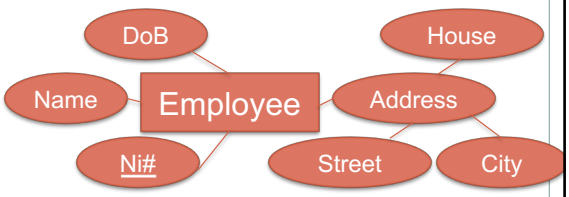


Step 1

ER to Schemas: 1 - Strong Entity Types

16

- For each *strong* entity type, create a relation (table) with:
  - A column for each simple attribute
    - composite attributes - broken into several columns
  - A primary key – one or more of the candidate key attributes



EMPLOYEE

name	NINumber	DoB	House	Street	City

address

Good Practice: Use the name of the composite attribute as a prefix for each of the component names

Step 1

Attributes → Columns

17

- A column of a relation is an **attribute** having:
  - a **name** (indicates the role the column has in this relation)
  - a domain (indicates the set of values it may take)

Student_ID_Number: INT, **address**: VARCHAR(100), **dateOfBirth**: DATE

Step 1

Primary Keys

18

Another Example:

Employee (name: VARCHAR(20) ,**NI_no**: INT)

Project (p_name: VARCHAR(20), **P_ID**: INT)

- a particular staff record can be identified as: *the record in the Employee table where NI_No= 9912345*
- a particular project record can be identified as: *the record in the Project table where P_ID= 125*
- ***all of the record's other data will be accessed via these 'keys'***
  - i.e. given a key value, we can determine everything else about the corresponding record

Step 1

Examples of Primary Keys?

19

- Student Id
- Staff number
- National Insurance Number/Social Security Number
- Course Id

- First Name and Last Name?
- FlightNumber and FlightDate
- Bank Sort Code and Account Number

- Primary keys can consist of a single attribute or **multiple attributes** in combination
  - Called a **composite key**

Step 2

Back to the ER Diagram: Relations - Entities

20

- Movie(Title, Type)
- Stars(Name, Address)
- Studio(Name, Address)

What about Release?

Step 2

The Relational Model

21

- A Movie has a Release
- In the relational database
  - how does each member of the Release entity set know which Movie they are related to?

➢ This is done via **KEYS**

- **Primary**
- **Foreign**: references to the primary key of another table

Step 2

Weak Entities Mapping

22

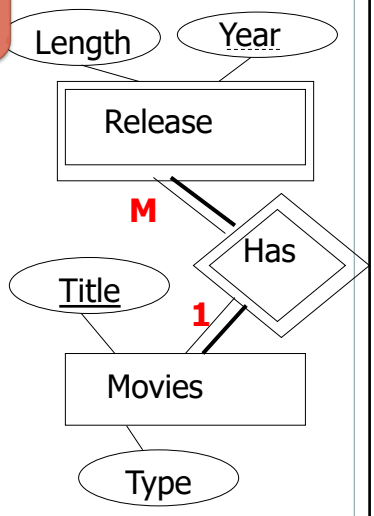
- Create primary key for a weak entity type from
  - primary key attributes of the identifying relationship types
  - partial keys of the weak entity
- ***Rule:*** *For each weak entity*
  - *create a new relation schema*
  - *for each identifying relationship: add the key attributes of the related entity to the new schema as foreign key attributes*
  - *declare the primary key of the schema*
  - *add the simple attributes*

Step 2

Relations, Relationships

- Movie(Title, Type)
- **Release(Title, Year, Length)**
- Stars(Name, Address)
- Studio(Name, Address)

Title is a **Foreign key**



Step 2

Foreign Keys: Definition

24

- The foreign key is used to *cross-reference* between tables
- Definition: A **foreign key** is an attribute (or set of attributes) that *exist in one or more tables* and which is the *primary key* for *one* of those tables.
- The foreign key restricts the domain of the attribute
  - A value in a foreign key MUST exist in the referenced primary key attribute

Movies

Title	Type
42nd Street	Musical
Miracle on 34th Street	Chrismas

Release

Title	Year	Length
Snake Pit	1948	108

Violation of Foreign Key constraint

Step 2

Using Foreign Keys

25

- A foreign key is a *referential constraint* between two tables, i.e. a value in a foreign key MUST exist in the referenced primary key
- A table may have multiple foreign keys and each foreign key can reference a different table
- Foreign keys need *not* have the same attribute name across tables
  - Could have been Release(MovieTitle, Year, Length)
  - Names should differ to help readability!
- The MUST have the same data type
- Foreign keys are important in modelling **weak entities** and **relationships**.

Step 2

Foreign Keys

26

"John Cooper advises Jane Jones"

- There are only two ways of connecting two related pieces of data in a relational database
  - 1. They are in the same tuple (row) of the same table, i.e. with the same primary key value
    - "Jane" and "Jones" are connected since they are in the same record
  - 2. They are in tuples which are connected by a foreign key or a chain of foreign keys
    - "Jones" and "Cooper" are connected by the foreign key, *adviser*

Step 2

Back to the ER Diagram: Relations - Entities

27

What do we have thus far?

- Movie(Title : VARCHAR(50), Type : VARCHAR(20))
- Release(Title : VARCHAR(50), Year : INT, Length : INT)
- Stars(Name : VARCHAR(50), Address : VARCHAR(50))
- Studio(Name : VARCHAR(50), Address : VARCHAR(50))

Notes:

- each table has primary key attribute(s)
- Release's Title is a foreign key reference to Movie's Title

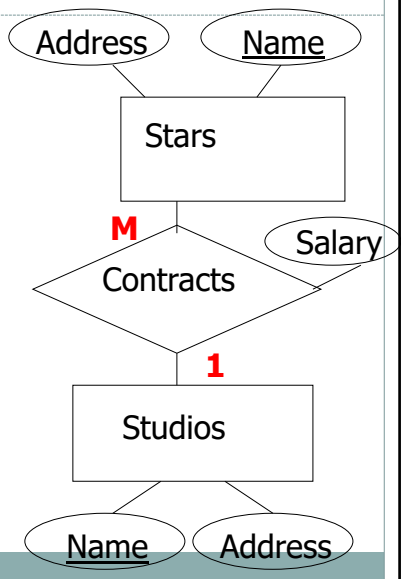
Step 3

Relationships: 1-to-many

28

- Movie(Title, Type)
- Release(Title, Year, Length)
- Stars(Name, Address)
- Studio(Name, Address)

1-to-many rule:
the primary key on the 'one side' of the relationship is added to the 'many side' as a foreign key.



Step 3

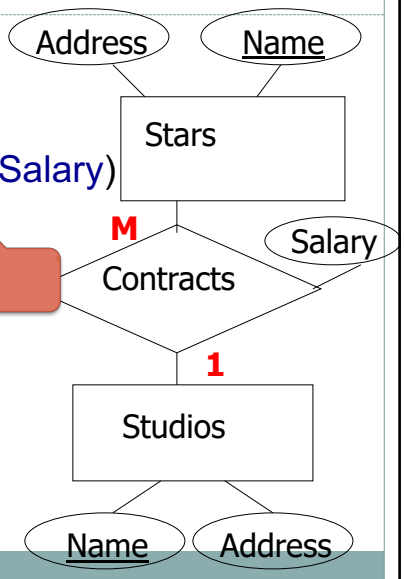
Relationships: 1-to-many

29

- Movie(Title, Type)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)

Foreign key

1-to-many rule:
the primary key on the 'one side' of the relationship is added to the 'many side' as a foreign key.

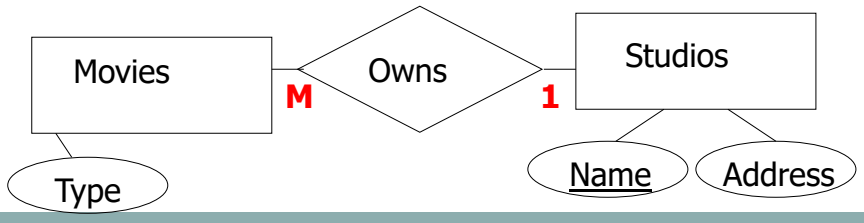


Step 3

Relationships: 1-to-many

30

- Movie(Title, Type)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)

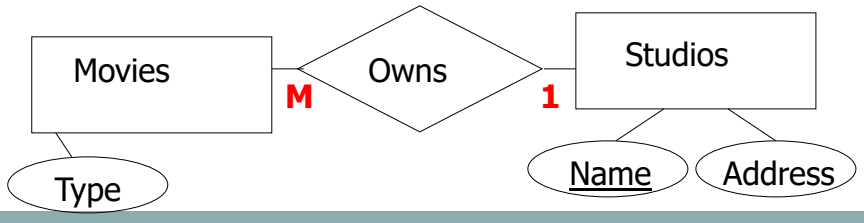


Step 3

Relationships: 1-to-many

31

- Movie(Title, Type, Studio)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)

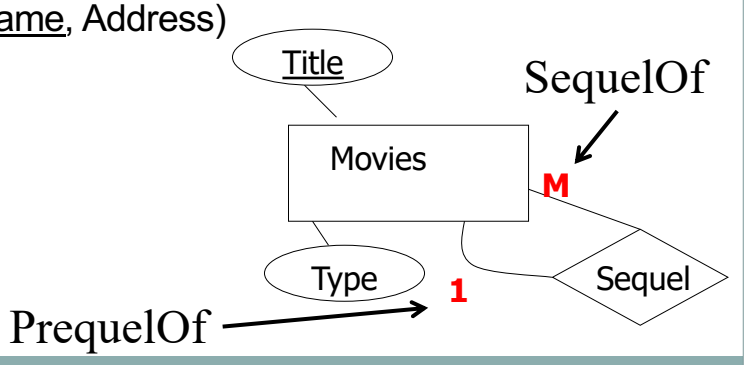


Step 3

Relationships: 1-to-many

32

- Movie(Title, Type, Studio)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio , Salary)
- Studio(Name, Address)

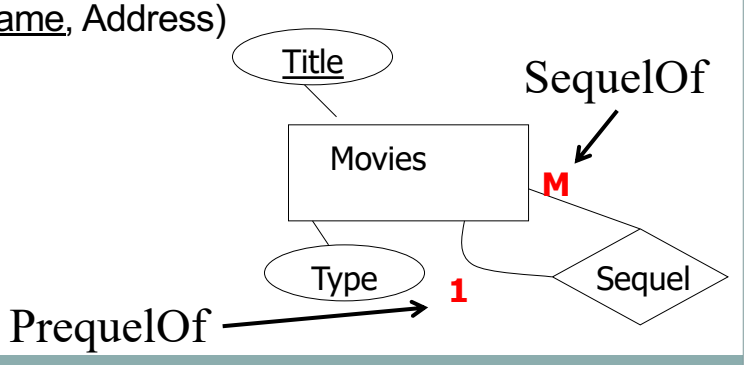


Step 3

Relationships: 1-to-many

33

- Movie(Title, Type, Studio, SequelOf)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio , Salary)
- Studio(Name, Address)



Step 4

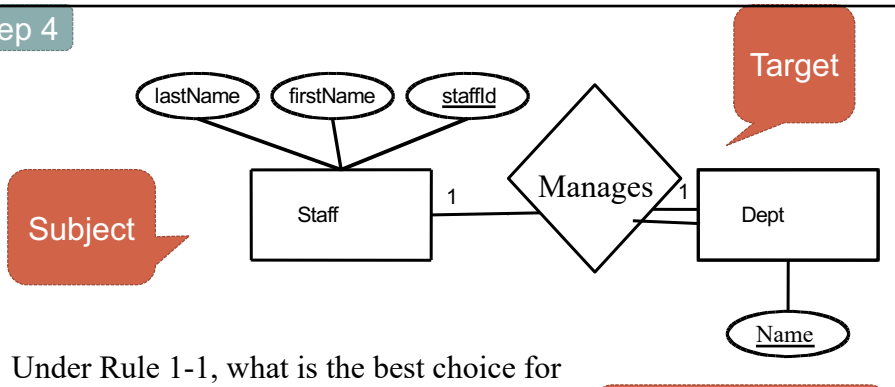
One-to-one relationships (and their attributes)

34

- The foreign key attributes may be added to either schema
- ***Rule 1-1:*** *For each one-to-one relationship type between two entity types, choose one entity type to be the subject and one to be the target type*
  - *add the key attributes of the subject class to the target schema as foreign key attributes*
  - *add the attributes of the relationship to the target schema*

34

Step 4



Under Rule 1-1, what is the best choice for "subject" and "target"?

Set this to be "NOT NULL"

- Staff(staffId, firstName,lastName)
- School(Name, headOfSchoolStaffId)

"If the relationship is mandatory for one entity but not the other, the put foreign key into the table for which participation is mandatory"

35

Step 5

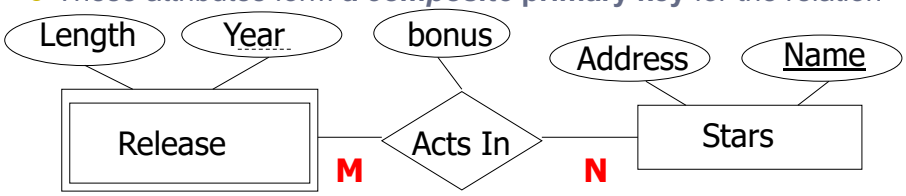
Relationships: many-to-many

36

- Movie(Title, Type, Studio, SequelOf)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio , Salary)
- Studio(Name, Address)

Many-to-Many Rule:

- A **new relation** is produced which contains the primary keys from both sides of the relationship *as foreign keys*
- These attributes form **a *composite* primary key** for the relation



Step 5

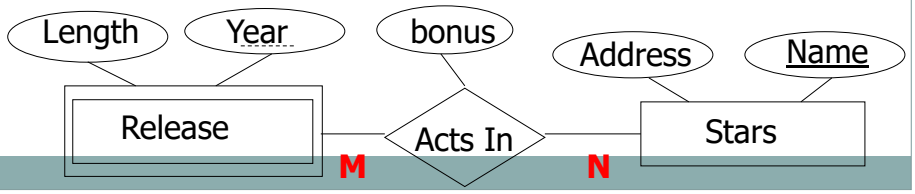
Relationships: many-to-many

37

- Movie(Title, Type, Studio, SequelOf)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio , Salary)
- Studio(Name, Address)
- ActsIn(Title,Year,StarName, bonus)

Many-to-Many Rule:

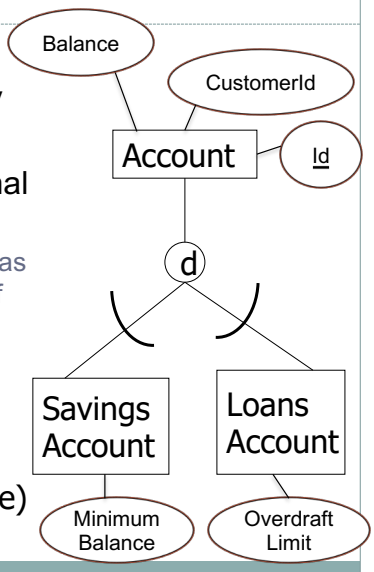
- A **new relation** is produced which contains the primary keys from both sides of the relationship *as foreign keys*
- These attributes form **a composite primary key** for the relation



Subtypes & Supertype

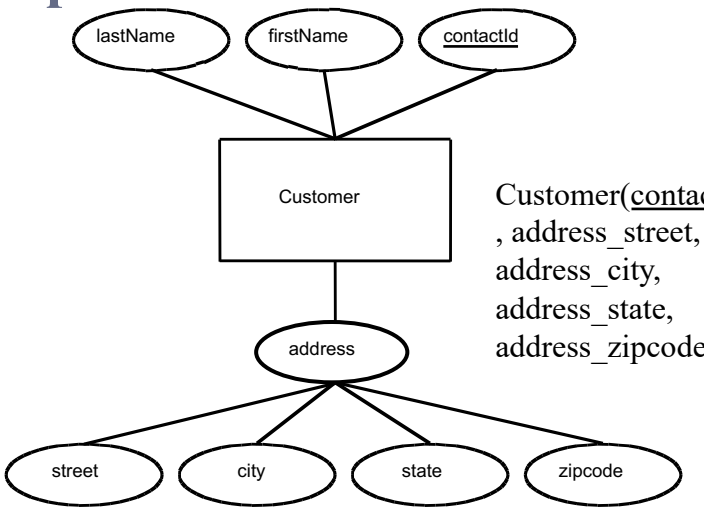
38

- Not really part of the main relational model, and not directly supported by most standard relational DBMS
- You cannot directly map to a relational schema. Two choices:
  - Model the supertype and all its subtypes as a **single table** (and leave attributes *null* if they don't apply), OR
  - Turn each subtype into its **own table** and set up 1:1 relationships between them super-entity types.
- Account(<u>ID</u>, Customer, Balance)
- Savings(<u>AccountID</u>, MinimumBalance)
- Loans(<u>AccountID</u>, OverDraftLimit)



Step 6

Composite attributes



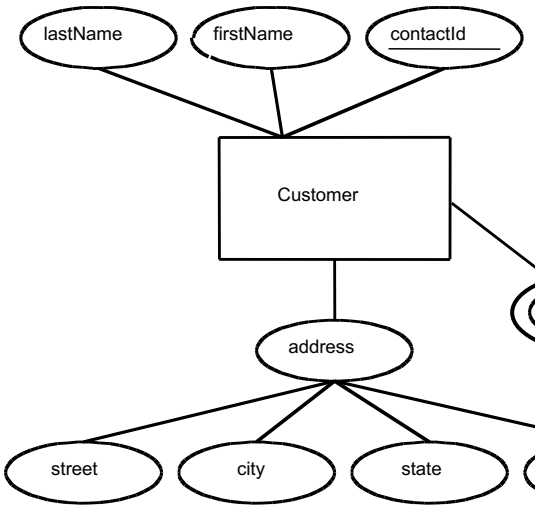
Customer(<u>contactId</u>,… , address_street, address_city, address_state, address_zipcode)

Create an attribute in the relation schema for each component attribute
Use the name of the composite attribute as a prefix for each of the component names

39

Step 7

Multi-valued attributes



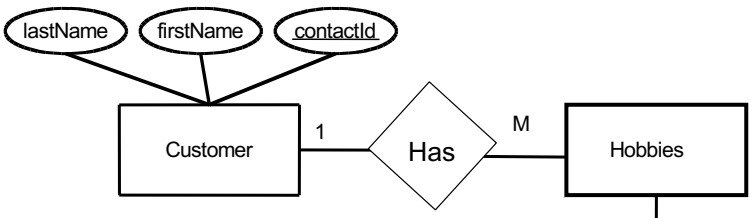
contactid	firstname	lastname	hobbies
1639	George	Barnes	reading
5629	Susan	Noble	hiking, movies
3388	Erwin	Star	hockey, skiing
5772	Alice	Buck	
1911	Frank	Borders	photography, travel, art
4848	Hanna	Diedrich	gourmet cooking

Difficult to search for a particular hobby

Represent each multi-valued attribute as if it were a weak entity class

40

Step 7



contactid	hobby
1639	reading
5629	hiking
5629	movies
3388	hockey
3388	skiing
1911	photography
1911	travel
1911	art
4848	gourmet cooking

41

Summary: ER -> Relations

- Strong entities
  - build a table with columns for each attribute

[Step 1]

- Weak entities
  - build a table with columns for each attribute
  - Add the PK of the owner entity

[Step 2]

- Relationships
  - 1-N: N side
  - N-M: new relation
  - 1-1: any side
- Sub-types
  1. Collapse to large supertype relation, OR
  2. Compose as 1-to-1 relationships

[Steps 3-5]

Schemas with Domains

43

- Movie(Title, Type, Studio, SequelOf)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)
- ActsIn(Title, Year, StarName, bonus)

VARCHAR(xx)
INT
BIT

Schemas with Domains

44

- Movie(Title, Type, Studio, SequelOf)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)
- ActsIn(Title, Year, StarName, bonus)

VARCHAR(xx)

INT

BIT

Schemas with Domains

45

- Movie(Title : VARCHAR(50), Type : VARCHAR(50), Studio : VARCHAR(50), SequelOf : VARCHAR(50))
- Release(Title : VARCHAR(50), Year : Int, Length: Int)
- Stars(Name : VARCHAR(50), Address : VARCHAR(100), Studio VARCHAR(50), Salary : Int)
- Studio(Name : VARCHAR(50), Address : VARCHAR(100))
- ActsIn(Title : VARCHAR(50), Year : Int, StarName : VARCHAR(50), bonus : Int)

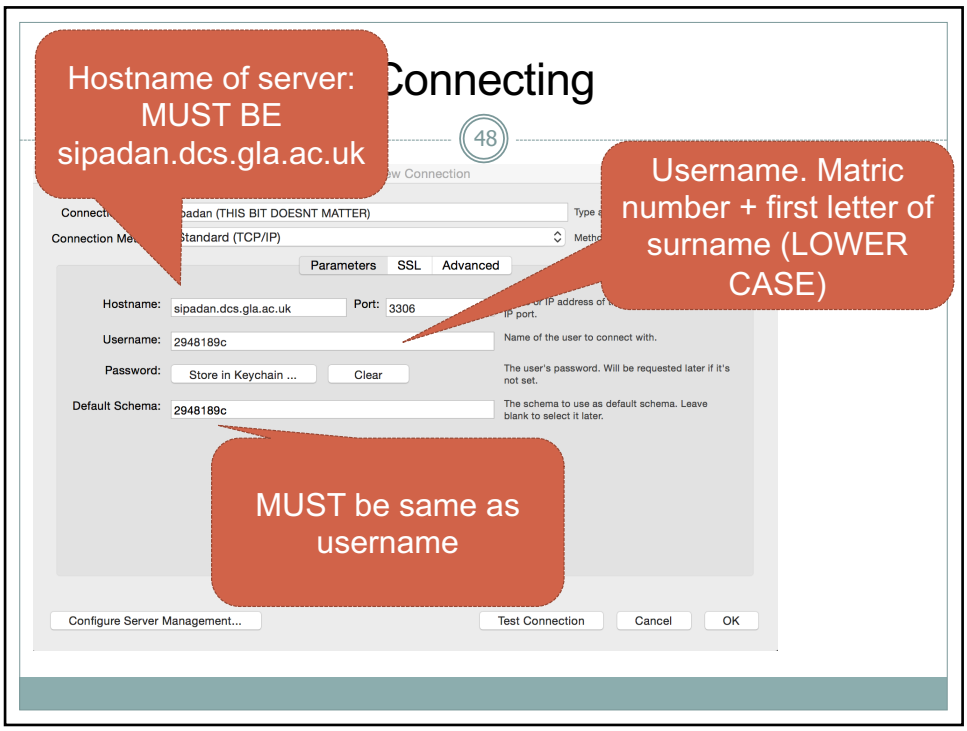
Some Tips!

- Follow the stepwise guide – it works!

- Write a schema first – then go to the DBMS to build the tables
- Add the entities OWN attributes – then decide what FKs to add
- Be careful to select good data types – they must match when you go to connect PKs and FKs

USING MYSQLWORKBENCH

47



Practically…

49

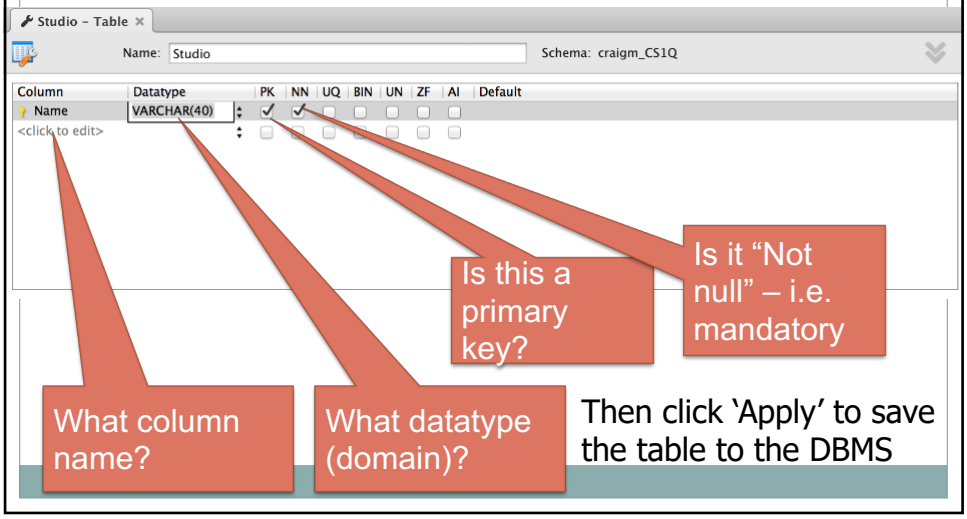
- We use MySQLWorkBench to create tables once we have our relational database schema

Movie(Title, Type, Studio, SequelOf)
Release(Title, Year, Length)
Stars(Name, Address, Studio, Salary)
Studio(Name, Address)
ActsIn(Title, Year, StarName, bonus)

VARCHAR(xx)
INT
TINYINT(1)

Table Editor (1): Create Attributes

50



Then click 'Apply' to save the table to the DBMS

Doing it manually

51

- We can make a schema manually using a CREATE TABLE SQL statement

```
CREATE TABLE Movies (
        Title varchar(50) NOT NULL,
        Type varchar(45) DEFAULT NULL,
        Studio varchar(45) DEFAULT NULL,
        SequelOf varchar(50) DEFAULT NULL,
        PRIMARY KEY (Title)
)
```

1. The command
Column name
Data type & size
Required?
Primary key attribute(s)

- Try it out yourself for a single table!
- References: http://www.mysqltutorial.org/mysql-create-table/ https://dev.mysql.com/doc/refman/8.0/en/creating-tables.html

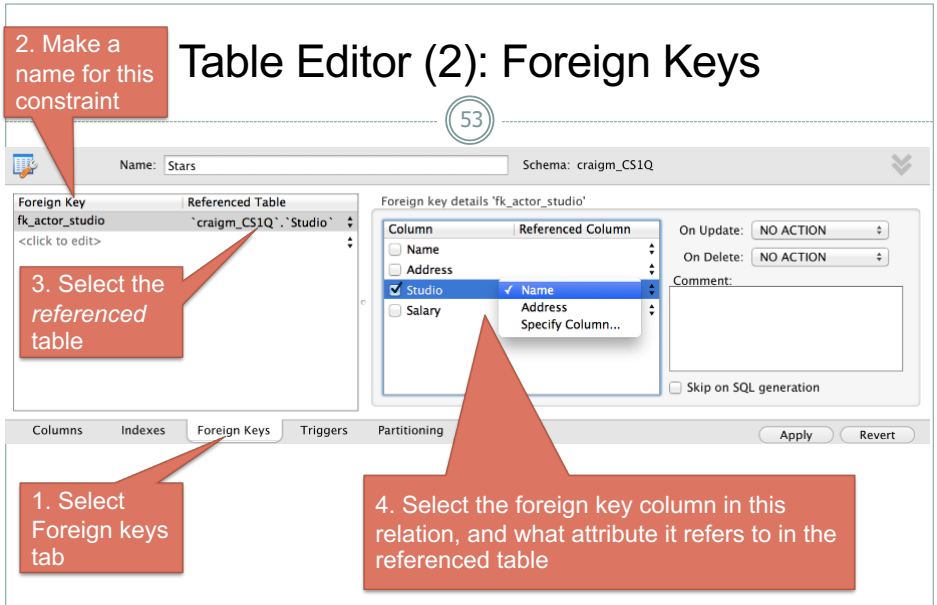
Practical Tips

52

- Create the tables in the same order as the **Translating E-R Diagram to Relational Schema** rules
  - Create Strong entities first
  - Weak entities next
  - When creating a foreign key, the referenced table & column **must already** exist!
- Which order to create these tables?
  - Movie(Title, Type, Studio, SequelOf) — 3
  - Release(Title, Year, Length) — 4
  - Stars(Name, Address, Studio, Salary) — 2
  - Studio(Name, Address) — 1

Table Editor (2): Foreign Keys

53



Then click 'Apply' to save the table to the DBMS

What to do now

- This week's lab is on converting your ER diagram into tables in MySQL
  - Follow the labsheet instructions to connect to the MySQL database server
  - Use the step-wise guide included in this lecture to design your relational schema, then create your tables
- Once you have created the tables you will have to create the relationships between the tables
  - This will be easier if you spend time getting the tables correct
  - Get your tutor to keep checking your tables are ok

Reading

55

- Garcia Molina
  - Chapter 7, Section 1-1.2 (pages 303-307)

OR

https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/