The Relational Model
From ER model to Tables

CS-1Q

IM Lecture 5

Craig Macdonald

Database design lifecycle

- 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

- Thursday
  - Converting your diagram into tables
  - Enforcing integrity
  - More on the relational model

Reminder - Data Modelling

- ER Model allowed us to establish the relationships and dependencies amongst the information

- We now need to arrange the data into a **logical structure**

- The logical structure can then be mapped into the storage objects supported by the database - for example **tables**

The Relational Data Model

- Introduced by E.F. Codd in 1970
- Most commonly supported form used in s/w industry
- Simple means of representing & manipulating data
- Has a good theoretical/mathematical grounding
  - More on this later (lecture 7 and 8)

Entities → Tables

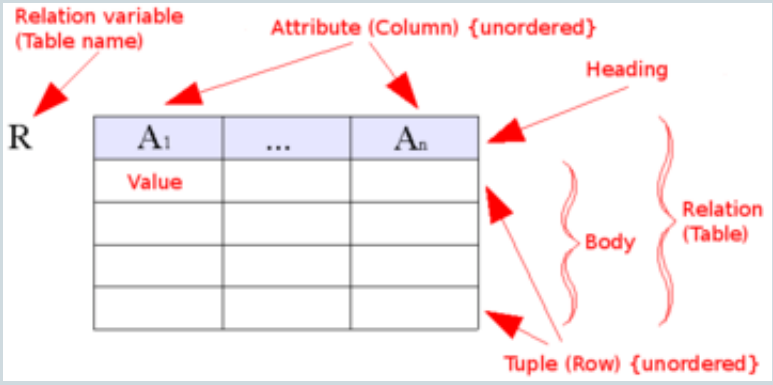
A table (relation) is constructed for each item of interest in a DB

A relation equates (approximately) to an entity type or en in the ER diagram

All relations must have a HEADING and a BODY

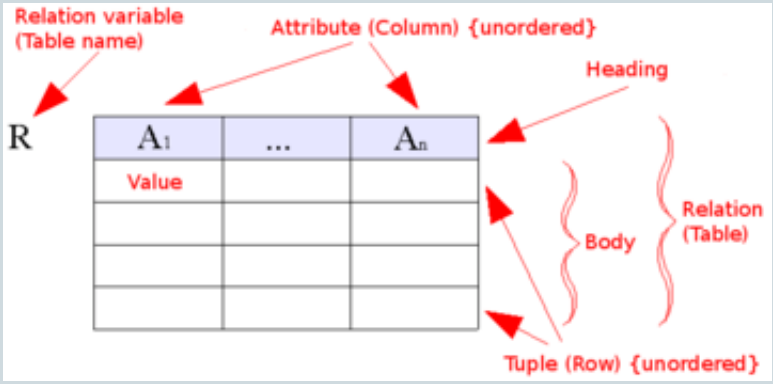
Structure of Data Objects in the Relational Model

- Data is represented in two dimesional TABLES (relations)



Structure of Data Objects in the Relational Model

- Each table has ROWS (tuples) and COLUMNS (attributes)



The Heading

- 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

The Body

- The rows of a relation comprise its body
  - These are referred to as TUPLES
- A tuple is an ordered list of values
- The meaning of each value is determined by it's position in the tuple
- The number of tuples in a relation determines it's CARDINALITY

Degree and Cardinality of a Relation

STUDENT

name	matric	exam1	exam2
Gedge	891023	12	58
Kerr	892361	66	90
Fraser	880123	50	65

- The relation student has:
  - Degree of 4 (number of attributes/columns)
  - Cardinality of 3 (number of rows/tuples)

GOTCHA: Do not confuse this with the cardinality of a relationship type in an E/R diagram

Relations → Schema

- A **tuple** (record) is a row of a relation, i.e. a set of values which are instances of the attributes
  - < 'Fraser', 880123, 66, 90 >

Relations → Schema

- 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

- A **domain** is a set of atomic values that can be assigned to an attribute
- A domain has two parts :
  - its meaning - e.g. the set of matriculation numbers
  - its format - e.g. a six digit integer
- 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, Date, etc.

Domains

- 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…….

Summary of a Table

- The STUDENT relation may be thought of as a 2-D table

STUDENT

name	Student ID	exam1	exam2
Gedge	891023	12	58
Kerr	892361	66	90
Fraser	880123	50	65

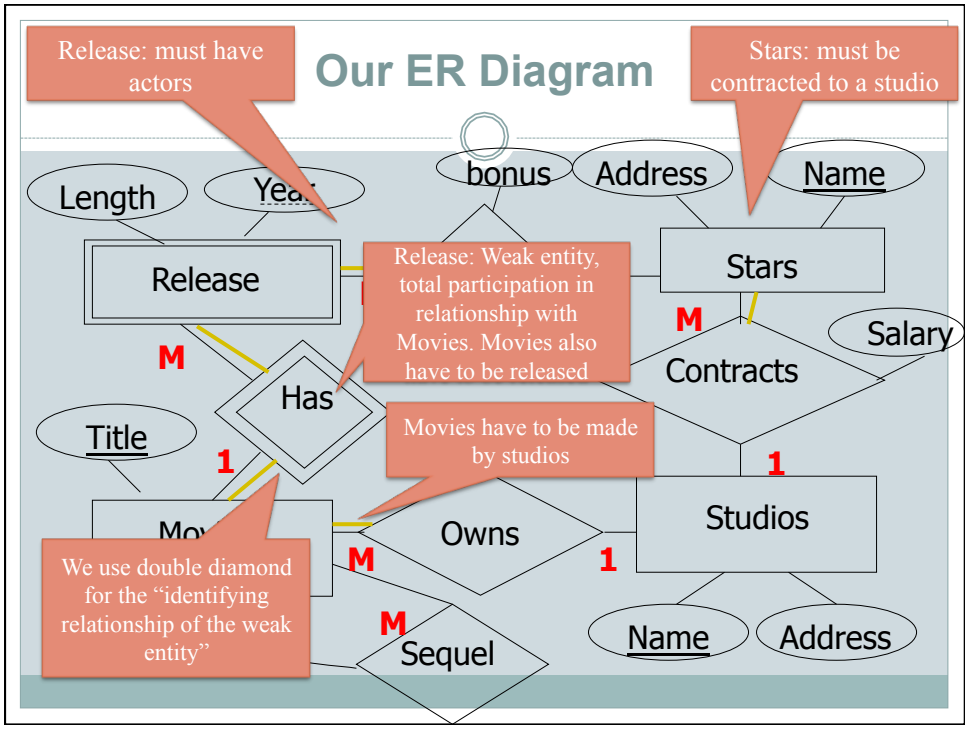
- A relation has
  - a **name - STUDENT**
  - an unchanging set of **columns** which are *named* and *typed (domain)*
  - a time varying set of **rows**, which are the current set of **records** for the relation

Converting your ER Diagram to Tables

Translating E-R to relational schema

18

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



ER to Schemas: 1 - Strong Entity Types

- 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 of the candidate key attributes

EMPLOYEE

name	NINumber	DoB	House	Street	City

address

Attributes → Columns

- 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: **integer**, **address**: varchar(100), **dateOfBirth**: date

Back to the ER Diagram: Relations - Entities

22

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

Primary Keys

Another Example –

Employee (name: Text,**NI_no**: Number)

Project (p_name: Text, **P_ID**: Number)

- 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'***

Back to the ER Diagram: Relations - Entities

24

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

What about Release?

The Relational Model

- 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

Weak Entities Mapping

26

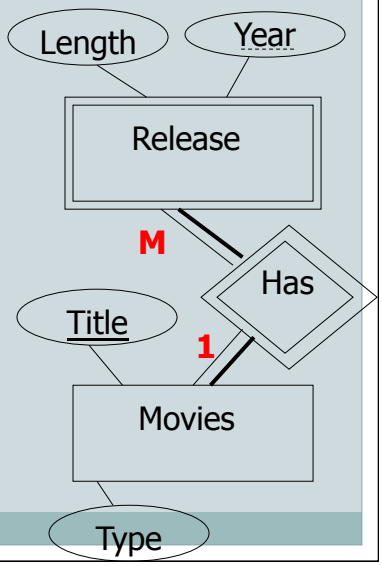
- Create keys for weak entities from
  - foreign keys of identifying relationship types
  - partial keys of the weak entity
- ***Rule:*** *For each weak entity*
  - *create a new relation schema*
  - *for each identifying relationship: add the key attributes of the related entity to the new schema as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

Relations, Relationships

27

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

Foreign key



Foreign Keys

- 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
    - "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*

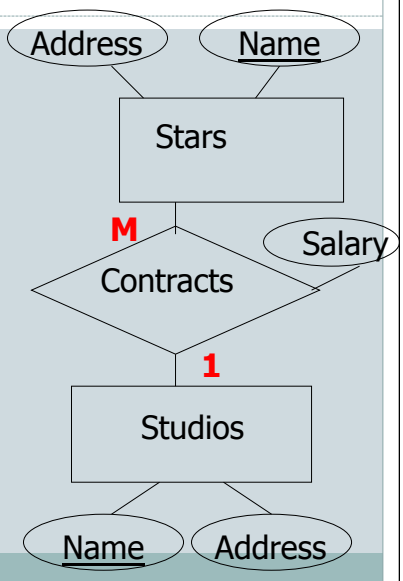
Foreign Keys

- A **foreign key** is an attribute (or set of attributes) that *exist in more than one table* and which is the *primary key* for *one* of those tables
- The foreign key is used to *cross-reference* tables
- A foreign key is a '*referential constraint*' between two tables
- A table may have multiple foreign keys
- Each foreign key can have a different referenced table
- Foreign keys do *not* have to have the same label across tables
  - Could have be Release(MovieTitle,Year, Length)

Relations, Relationships

- 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.

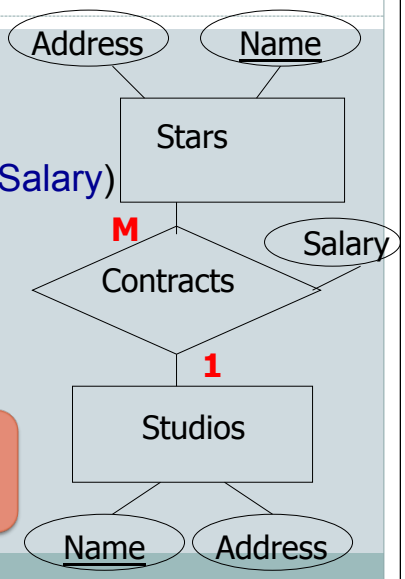


Relations, Relationships

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

1-to-many rule:
the primary key on the '
side' of the re
added to the
a foreign key.

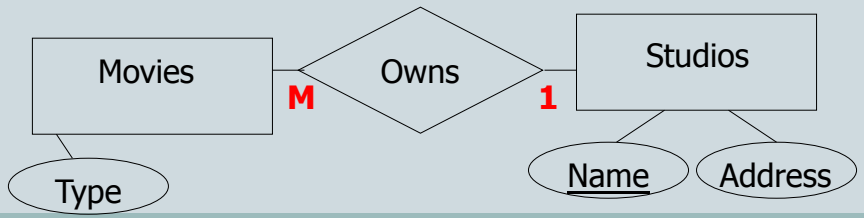
Foreign key



Relations, Relationships

32

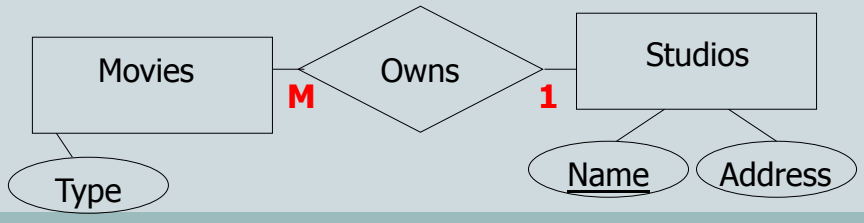
- Movie(Title, Type)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)



Relations, Relationships

33

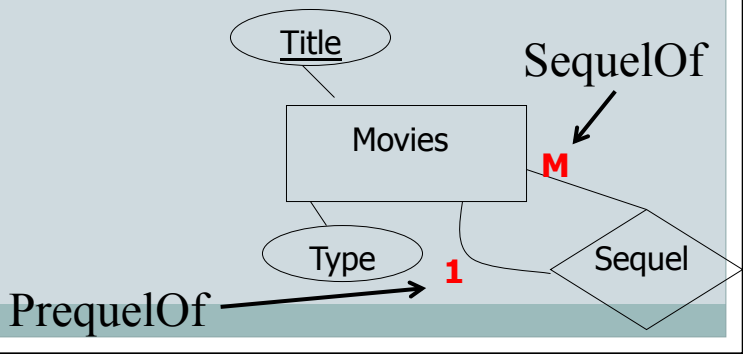
- Movie(Title, Type, Studio)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio, Salary)
- Studio(Name, Address)



Relations, Relationships

34

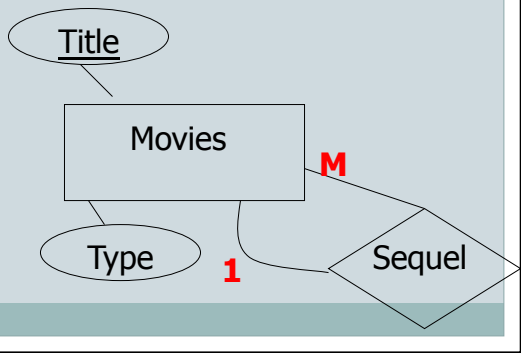
- Movie(Title, Type, Studio)
- Release(Title, Year, Length)
- Stars(Name, Address, Studio , Salary)
- Studio(Name, Address)



Relations, Relationships

35

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



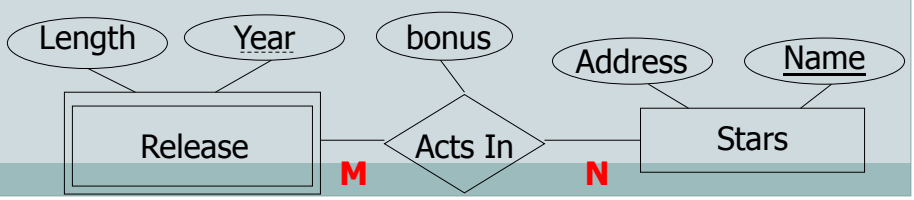
Relations, Relationships

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



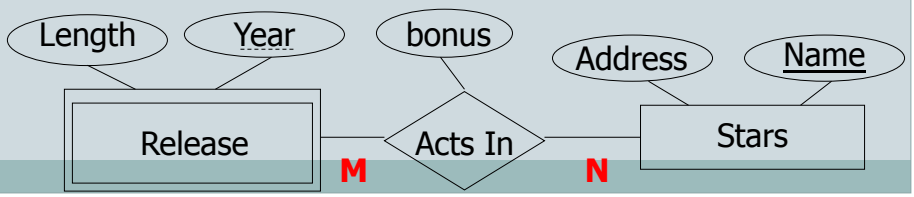
Relations, Relationships

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

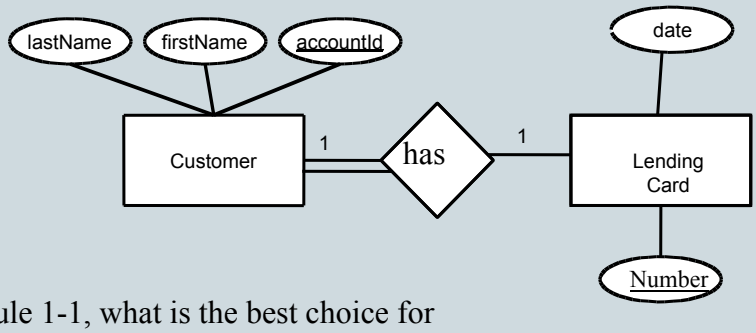


One-to-one relationships (and their attributes)

38

- 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*

38



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

- Customer(accountId, firstName,lastName, cardNumber)
- LendingCard(Number, date)

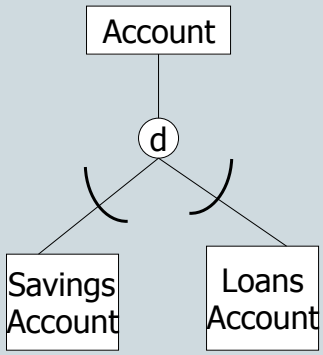
Take the primary key from the `mandatory end' and add it to the `optional end' as a foreign key.

39

Subtypes & Supertype

40

- 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.



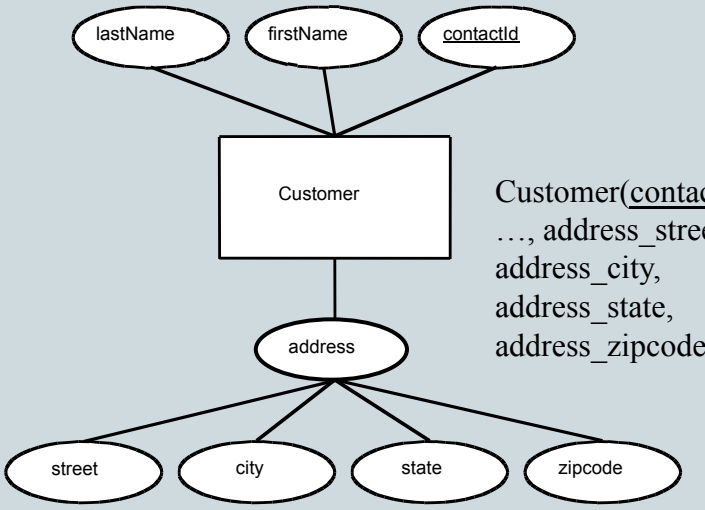
- Account(ID, Customer, Balance)
- Savings(AccountID, MinimumBalance)
- Loans(AccountID, Limit)

Translating E-R to relational schema

41

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

Composite attributes

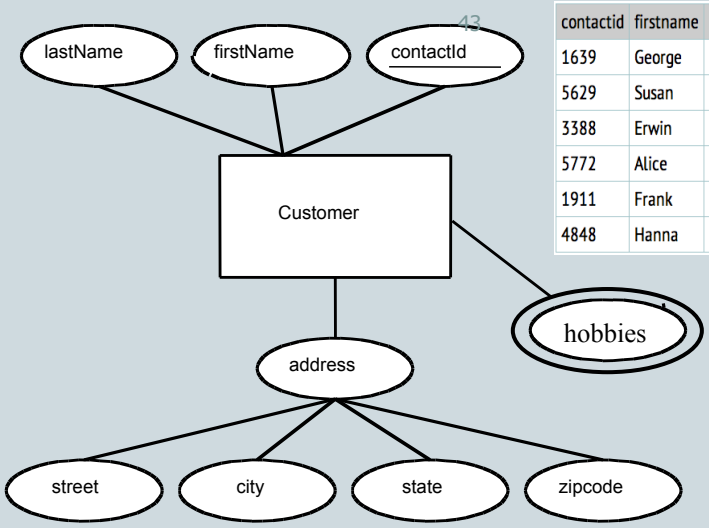


Customer(<u>contactId</u>, …, address_street, address_city, address_state, address_zipcode)

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

42

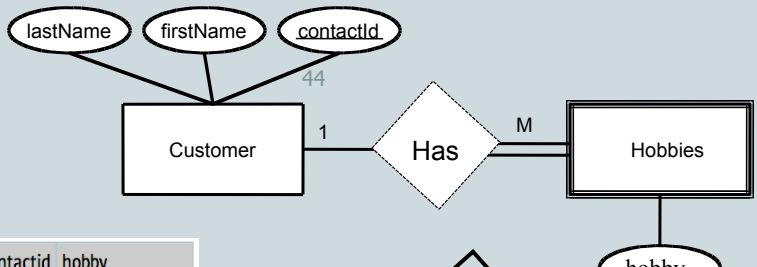
Multi-valued attributes



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



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

Constraints on relational databases

45

- **Inherent integrity constraints:**
  - must hold for all relational databases
  - typically enforced by DBMS
- **Enterprise constraints:**
  - specific to a particular application

Integrity constraints

46

- Primary key values must be unique
- Primary key values cannot be NULL
- Foreign key values:
  - must exist in the primary key of the referenced relations
  - may be NULL (if it is not a mandatory participation)

Enterprise constraints

47

- Application dependent
- Examples:
  - specified non-key attributes must not be NULL
    eg: all students must have a name, even if the primary key is the student number
  - values of one attribute must be less than values in another attribute
    eg: age of parent must be greater than age of child

Enforcing constraints

48

On update to the database that violates the constraints, the DBMS can:

- allow the update anyway (ignore constraints)
- refuse to perform the update
- compensate
  - cascade: make change, and check everything that refers to it
  - restricting: only change tuples that don't violate constraints
  - set foreign keys to null if referential integrity violated

Schemas with Domains

49

- 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

50

- 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

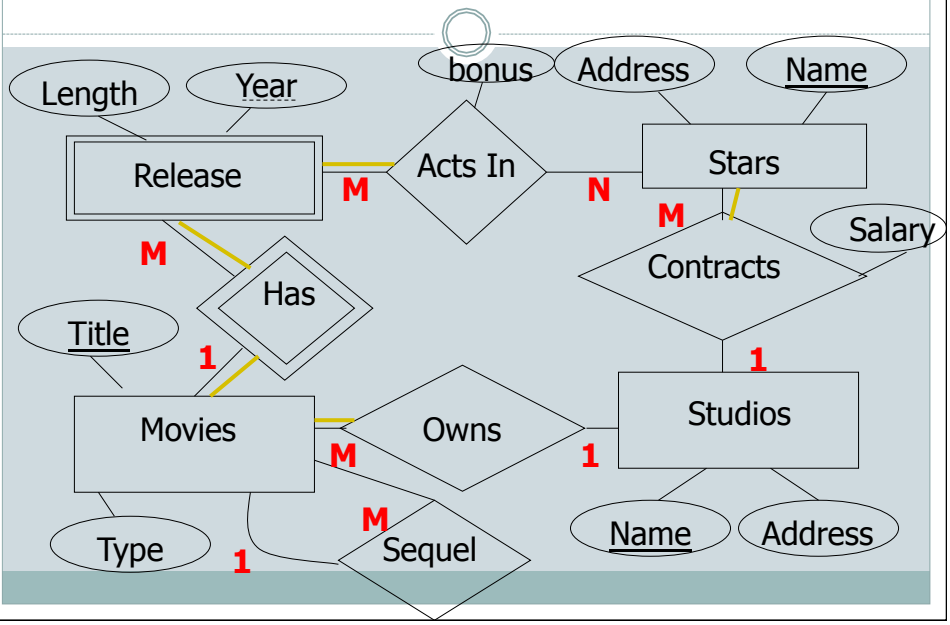
Some Tips!

- 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

Reminder

- Strong entities
  - build a table with columns for each attribute
- Weak entities
  - build a table with columns for each attribute
  - Add the PK of the owner entity
- Relationships
  - 1-N – N side
  - N-M – new table
  - 1-1– any side
- Sub-types
  1. Collapse to large supertype relation, OR
  2. Compose as 1-to-1 relationships

Our ER Diagram



Practically…

55

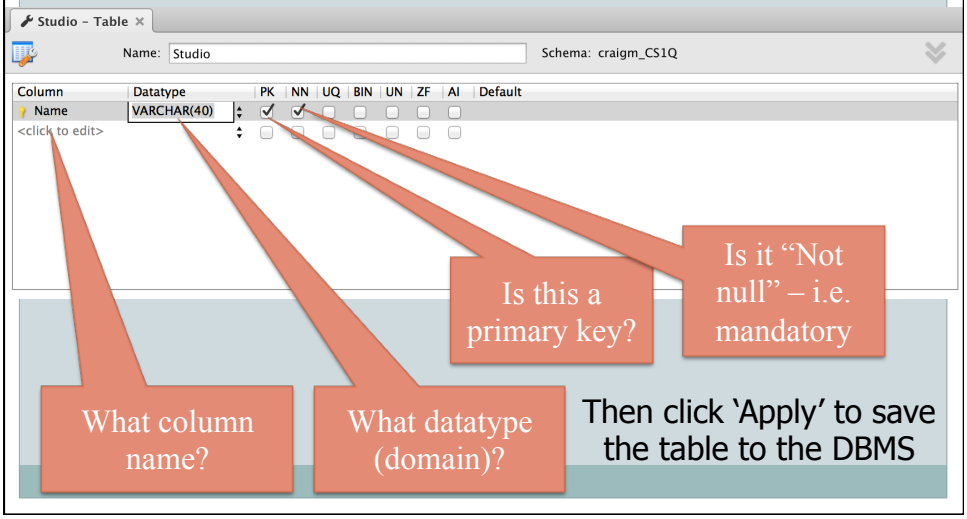
- We use MySQLWorkBench to create tables once we have our relational database schema

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
BIT

Table Editor (1): Create Attributes

56



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

Reminder

- Strong entities
  - build a table with columns for each attribute
- Weak entities
  - build a table with columns for each attribute
  - Add the PK of the owner entity
- Relationships
  - 1-1– any side
  - 1-N – N side
  - N-M – new table
- Sub-types
  1. Collapse to large supertype relation, OR
  2. Compose as 1-to-1 relationships

Essential Reading

After this lecture (Rolland)

- Chap 1, section 1.5.3
- Chap 3, sections 3.1 & 3.2
- Notes from lecture 4 and lecture 5

- Before next lecture (Rolland)
  - Chap 4, sections 4.1 & 4.3