HERIOT WATT UNIVERSITY

F27WD: Web Design & Databases

Databases Lecture 3: Relationships

Fiona McNeill

25th February 2019

LEADERS IN IDEAS AND SOLUTIONS

This week’s link: https://goo.gl/forms/wU5GgUgMqcd1jhn53

HERIOT WATT UNIVERSITY

How was last week?

1. Interesting and clear
2. Interesting but I didn’t understand everything
3. A bit dull but I could understand it
4. I didn’t understand everything and it was dull

Should I be going …?

- More quickly
- More slowly
- At this speed

What did we learn last week?

HERIOT WATT UNIVERSITY

What did we learn last week?

- What data is
- Why we should care about it
- What a database is
- The importance of tables
- How to begin to build and query tables using SQL

HERIOT WATT UNIVERSITY

Relational Databases

LEADERS IN IDEAS AND SOLUTIONS

Why are relations important?

- Sometimes we don’t want to just **state facts**
- We want to talk about how **types** of things and **instances** of things **interact** with each other.

Why are relations important?

- Sometimes we don’t want to just **state facts**
- We want to talk about how **types** of things and **instances** of things **interact** with each other.

This is why we use **relational databases.**

Why are relations important?

- Sometimes we don’t want to just **state facts**
- We want to talk about how **types** of things and **instances** of things **interact** with each other.

This is why we use **relational databases.**

… so we can talk about **relationships** between **data.**

Making an actor table

- What might we want to say about actors?

Making an actor table

- For now, we will say:
    - name
    - date of birth
    - sex
    - fee

Which could be a primary key

1. Name
2. Date of birth
3. Sex
4. Fee
5. More than one of the above
6. None of the above

Which could be a primary key

1. Name
2. Date of birth
3. Sex
4. Fee
5. More than one of the above
6. None of the above

Why is name a primary key?

Normally, a name would **not** be a primary key …

Why is name a primary key?

Normally, a name would **not** be a primary key because different people can have the same name.

Why is name a primary key?

Normally, a name would **not** be a primary key because different people can have the same name.

But actors must all have different names (union rules).

Why is name a primary key?

Normally, a name would **not** be a primary key because different people can have the same name.

But actors must all have different names (union rules).

Michael Keaton's real name is *Michael Douglas*
David Tennant's real name is *David McDonald*

What else might we want to include?

- Actors have agents

What else might we want to include?

- Actors have agents

  Each actor has only one agent. Agents have many actors.

What else might we want to include?

- Actors have agents

Each actor has only one agent. Agents have many actors.

This is a **one-to-many** relationship.

What else might we want to include?

- Agents have names

What else might we want to include?

- Agents have names

Each agent has one name; each name is held by one actor.

What else might we want to include?

- Agents have names

Each agent has one name; each name is held by one actor.

This is a **one-to-one** relationship

What else might we want to include?

- Actors and agents both have bank accounts
- Actors may have specific skills
- Bank accounts have account numbers and balances.

What else might we want to include?

- Actors and agents both have bank accounts

  This is ***one-to-one**** *relationship*
- Actors may have specific skills
- Bank accounts have account numbers and balances.

*This is actually a simplification - here we've decided to model a situation in which people only have one bank account. This makes sense in a database - you probably wouldn't want to record more than one bank account - but if you did you could make it one-to-many relationship

What else might we want to include?

- Actors and agents both have bank accounts

  This is ***one-to-one*** *relationship*

- Actors may have specific skills

  This is a ***many-to-many*** *relationship*

- Bank accounts have account numbers
- and balances.

What else might we want to include?

- Actors and agents both have bank accounts

  *This is **one-to-one** relationship*
- Actors may have specific skills

  *This is a **many-to-many** relationship*
- Bank accounts have account numbers

  *This is a **one-to-one** relationship*
- and balances.

What else might we want to include?

- Actors and agents both have bank accounts

  *This is **one-to-one** relationship*
- Actors may have specific skills

  *This is a **many-to-many** relationship*
- Bank accounts have account numbers

  *This is a **one-to-one** relationship*
- and balances.

  *This is a **one-to-many** relationship*

Actor Table

Name	Sex	DoB	Fee
Carl Pratt	M	1982-03-20	$1.1m
Anna Stone	F	1985-02-17	$1.3m
Rosie Ridley	F	1987-07-14	$1.5m
Jemma Laurence	F	1980-12-01	$1.7m

In a table like this, the relationships between the primary key and the other columns are …

Actor Table

Name	Sex	DoB	Fee
Carl Pratt	M	1982-03-20	$1.1m
Anna Stone	F	1985-02-17	$1.3m
Rosie Ridley	F	1987-07-14	$1.5m
Jemma Laurence	F	1980-12-01	$1.7m

In a table like this, the relationships between the primary key and the other columns are **one-to-one**.

Each primary key links to **only one value** in each other column.

What about these relationships?

- Actors have agents
- Agents have names
- Agents manage several actors
- Actors and agents both have bank accounts
- Actors may have specific skills
- Bank accounts have account numbers and balances.

What about these relationships?

- Actors have agents
- Agents have names
- Agents manage several actors
- Actors and agents both have bank accounts
- Actors may have specific skills
- Bank accounts have account numbers and balances.

These are one-to-one relationships

What about these relationships?

- Actors have agents
- Agents have names
- Agents manage several actors
- Actors and agents both have bank accounts
- Actors may have specific skills
- Bank accounts have account numbers and balances.

These are one-to-many relationships

What about these relationships?

- Actors have agents
- Agents have names
- Agents manage several actors
- Actors and agents both have bank accounts
- Actors may have specific skills
- Bank accounts have account numbers and balances.

This is a many-to-many relationship:

- This requires an extra table

Actor Table

Name	Sex	DoB	Fee	Skill
Carl Pratt	M	1982-03-20	$1.1m	Comic Timing
Anna Stone	F	1985-02-17	$1.3m	Gritty drama
Rosie Ridley	F	1987-07-14	$1.5m	Action
Jemma Laurence	F	1980-12-01	$1.7m	Gritty drama

If name wasn’t the primary key, it would be possible to put multiple rows in for each actor for each skill - but this is very inefficient.

Expressing this in a database

We can’t express this in a single table - it’s far too complex.

Because this is complex, we need to map out how we are going to express this.

We do this using a *relational diagram*

Why aren’t we using E-R diagrams?

- Well … we sort of are.
- Technically, the relational diagrams we are using are E-R diagrams plus additional information that shouldn’t be in an E-R diagram
- In reality, what we are doing is widely referred to as an E-R diagram

What is the difference?

In 2nd year, you will learn about the different levels of database design.

- The *conceptual layer* is where you think about what needs to be represented. This level could apply to different implementation - e.g., it doesn't specify that it needs to be relational
- The *logical layer*, where you define how it is actually going to be implemented

What is the difference?

An E-R diagram is technically only for the conceptual layer.

But in reality most database designers create these layers *at the same time* and refer to diagrams that cover both as E-R diagrams.

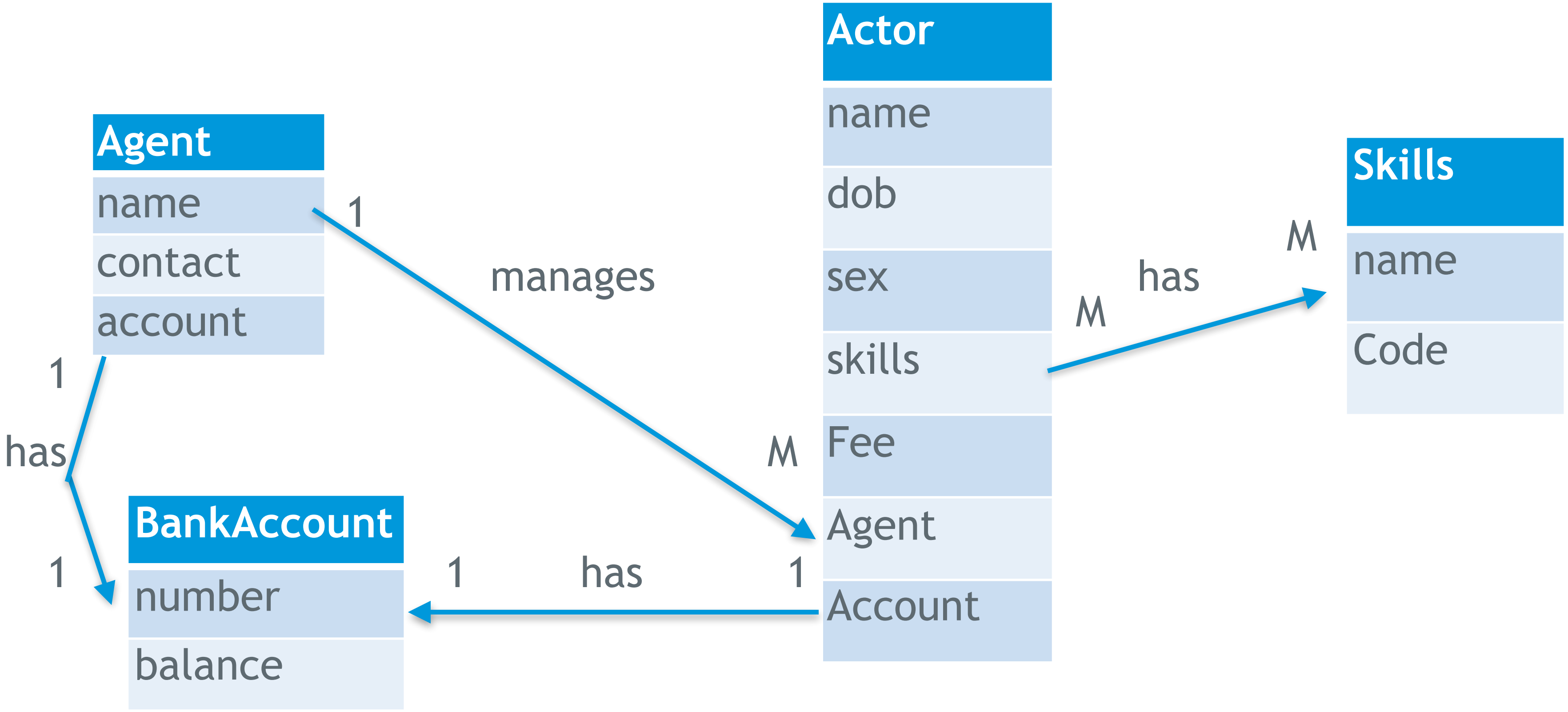
Why am I telling you this?

- At this stage, the distinction between these layers is not important
- It’s fine if you refer to what we are doing as E-R diagrams
- But don’t be surprised/confused when you get into 2nd year and you learn a definition for E-R diagrams that excludes some of what we are doing here.

Why am I telling you this?

- But now that you know this, and you know that you will learn more about this in 2nd year, you can forget about it for now.
- This is not examinable, I just want you to understand why I'm using the term *relational diagram* rather than *E-R diagram,* which you might expect.

Relational diagram



Agent

name

contact

account

Actor

name

dob

sex

skills

Fee

Agent

Account

Skills

name

Code

BankAccount

number

balance

1 manages M

M has M

1 has 1

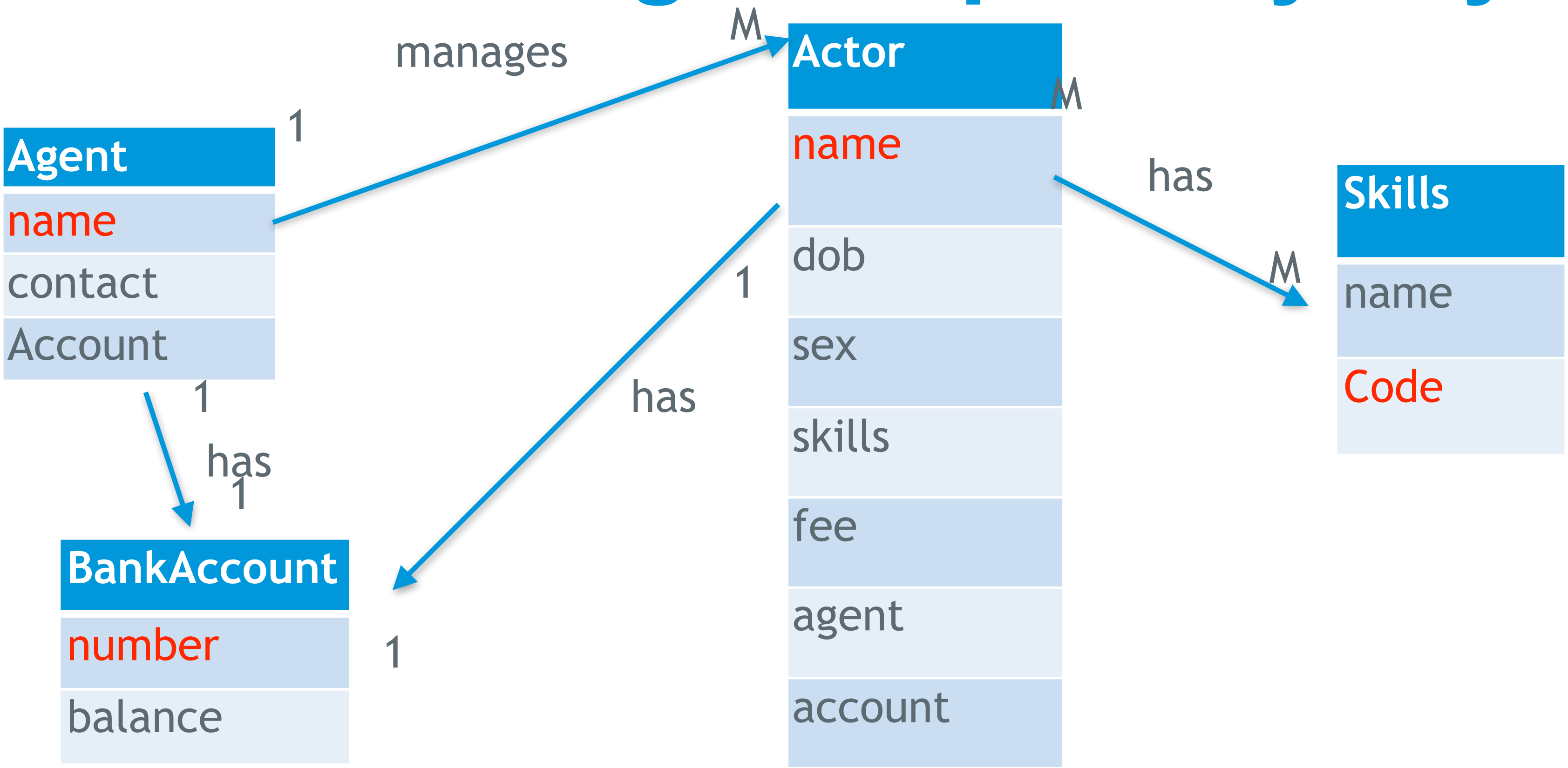
1 has 1

Relational diagrams contain tables and **relationships between tables.**

Relational diagrams - primary keys

Each table in the Relational must have a **primary key**

Relational diagram - primary key



Agent
name
contact
Account

Actor
name
dob
sex
skills
fee
agent
account

Skills
name
Code

BankAccount
number
balance

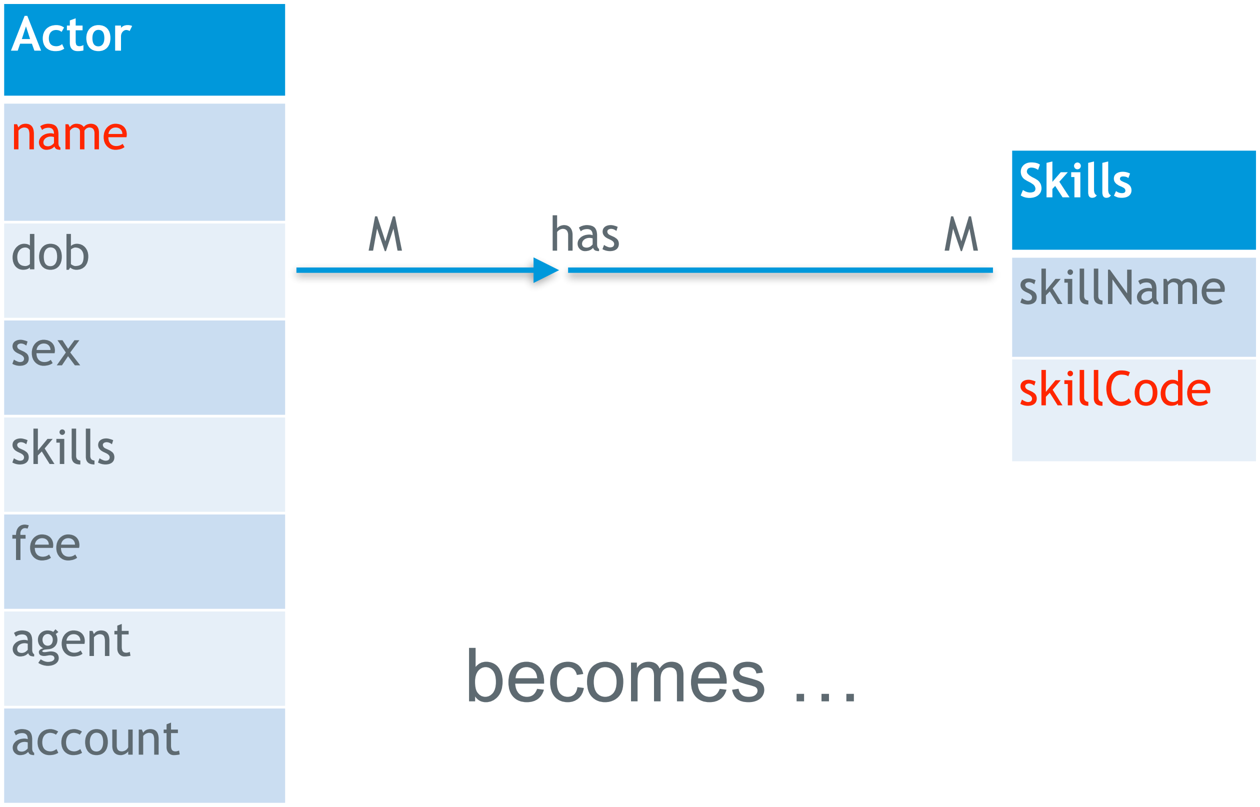
manages
has
has
has

Expressing this in a database

We can describe **one-to-one** relationships and **one-to-many** relationships in this way.

But **many-to-many** relationships need to be split up into simpler relationships using a **join table.**

Relational diagram - join tables



Actor

name

dob

sex

skills

fee

agent

account

M

has

M

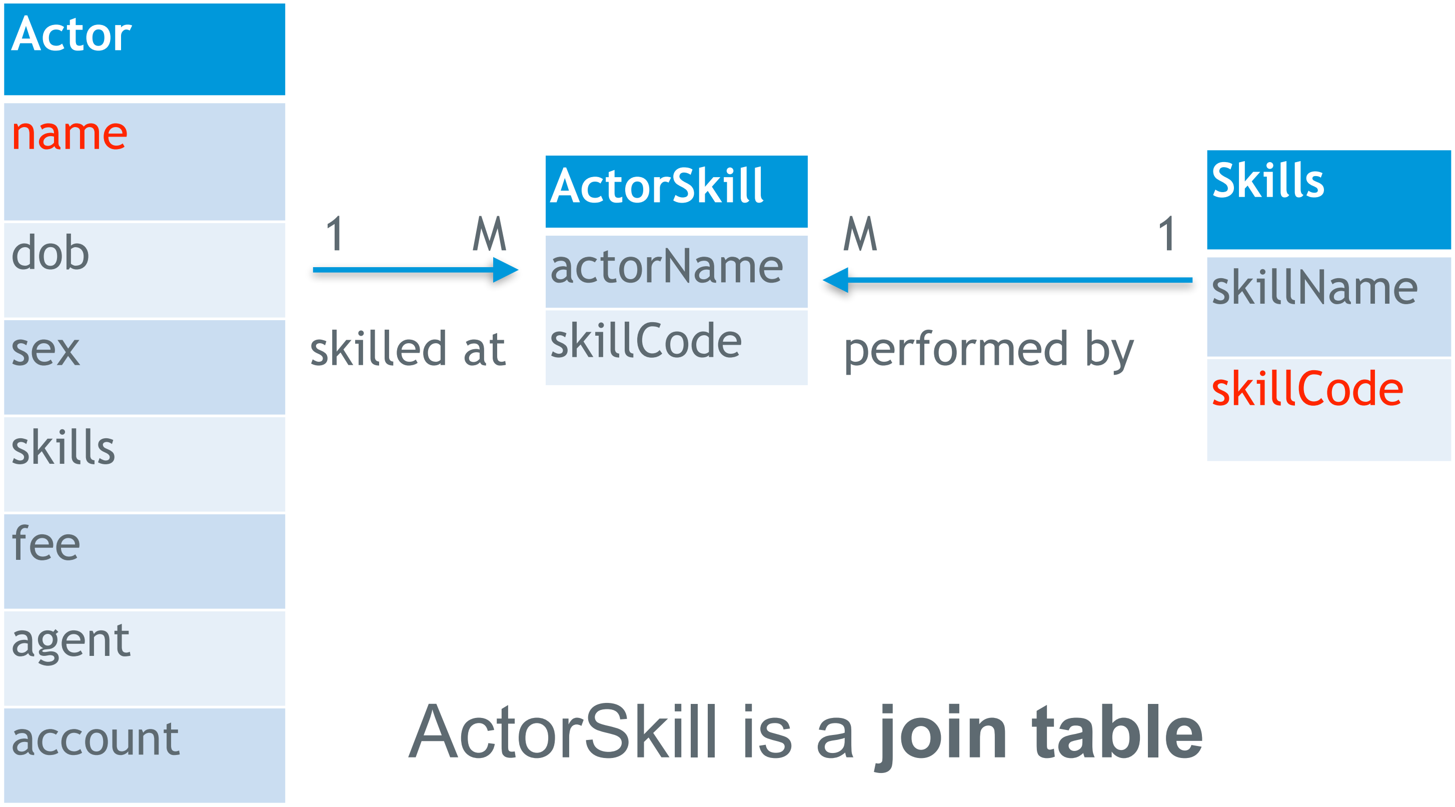
Skills

skillName

skillCode

becomes …

Relational diagram - join tables



Actor

name

dob

sex

skills

fee

agent

account

1 M

skilled at

ActorSkill

actorName

skillCode

M 1

performed by

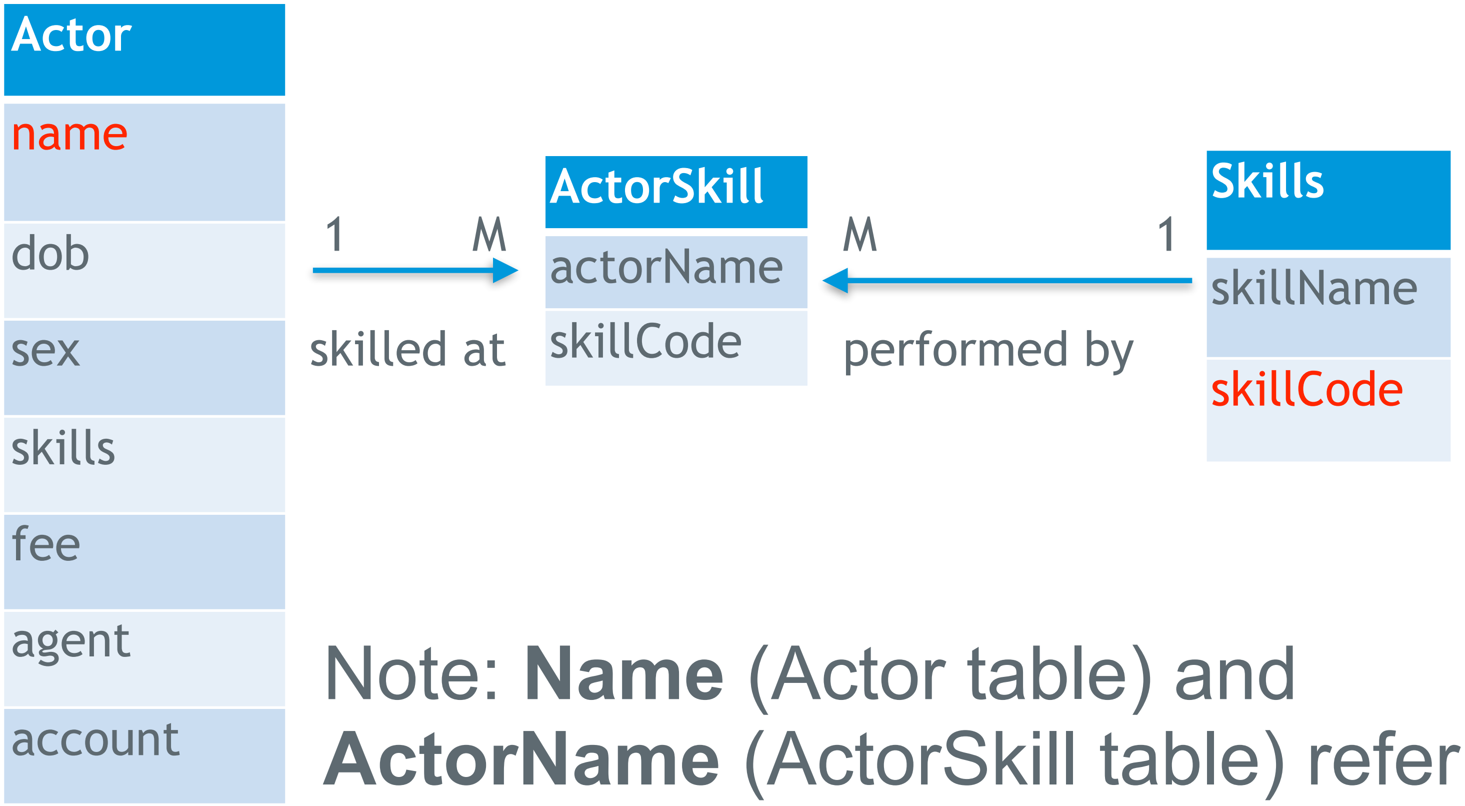
Skills

skillName

skillCode

ActorSkill is a **join table**

Relational diagram - join tables



Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
skillName
skillCode

Note: **Name** (Actor table) and **ActorName** (ActorSkill table) refer to the *same thing*

Relational diagram - join tables

ActorSkill
actorName
skillCode

What is the primary key in this table?

Relational diagram - join tables

ActorSkill
actorName
skillCode

There is no **individual** primary key.

Instead, we have a **combined** primary key.

The primary key is (ActorName,SkillCode).

This is generally the case with join tables.

Foreign Keys

A **foreign key** is a field in relational table that matches a **primary key** in another table.

These establish the **links** and express the **relationships** between the tables.

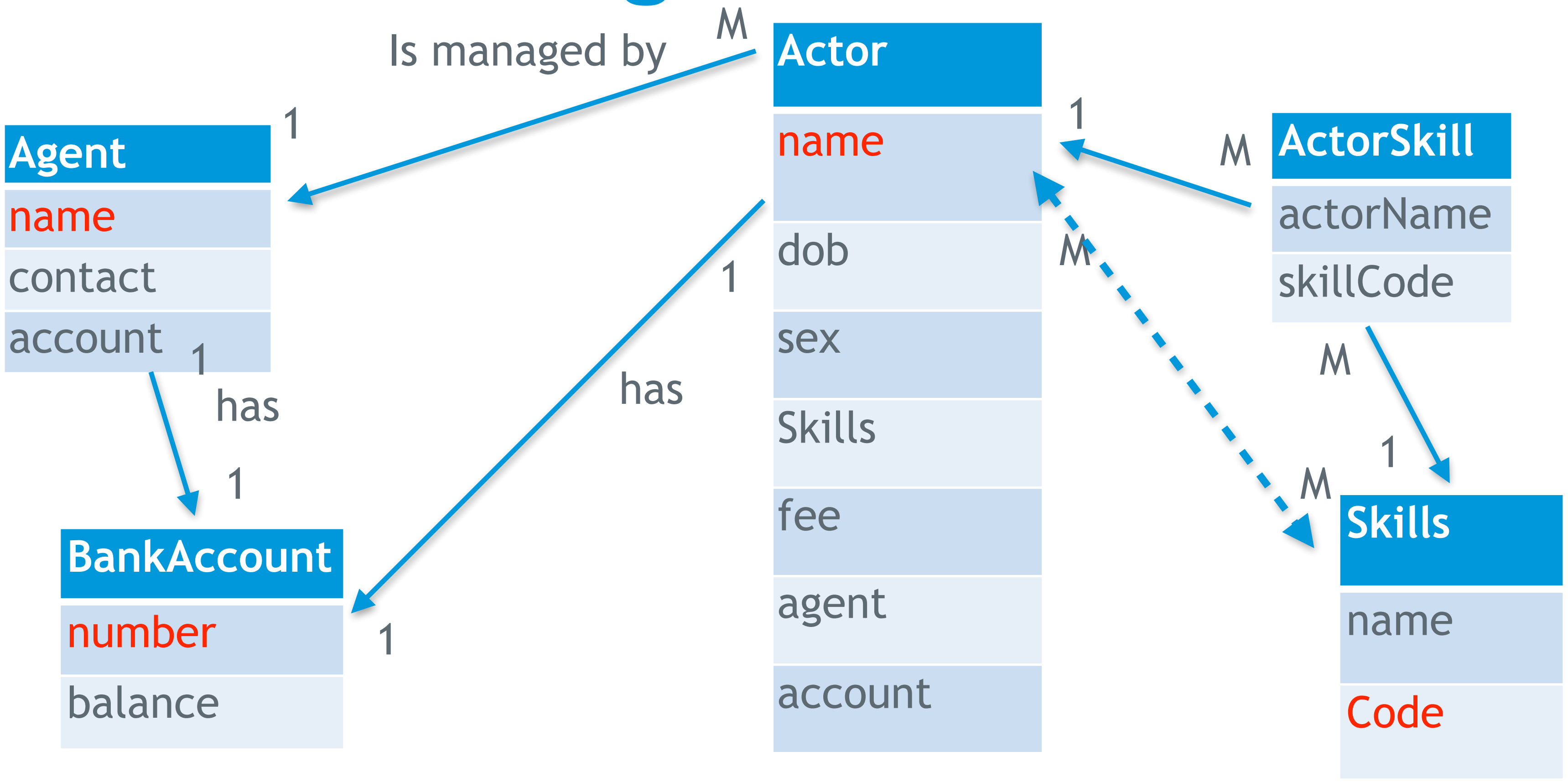
Foreign Keys

A **foreign key** is a field in relational table that matches a **primary key** in another table.

These establish the links between the tables

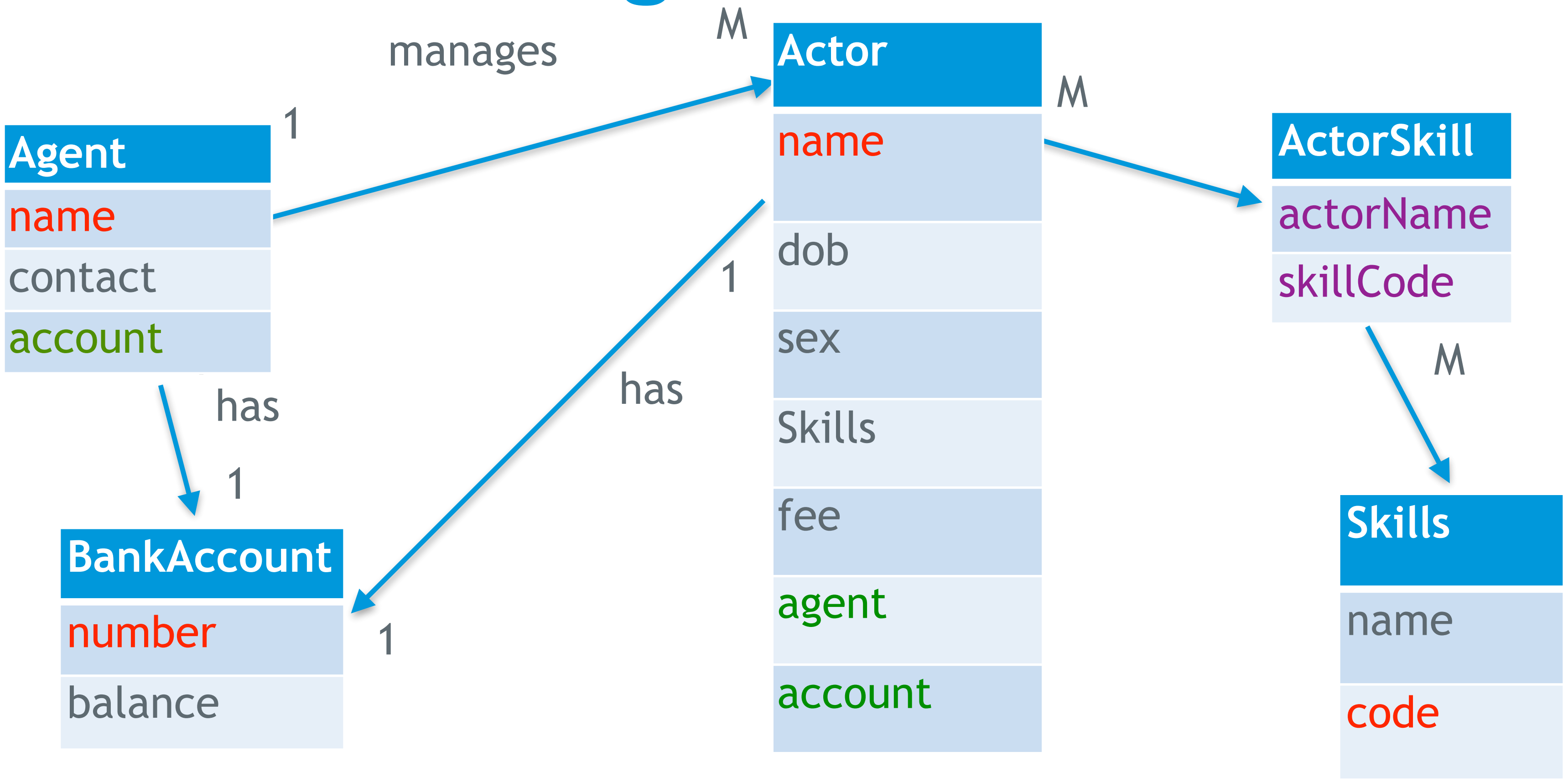
Remember *you can use different names in different tables to refer to the same thing.*

Relational diagram



Primary key
Foreign key
Jointly primary and individually foreign keys

Relational diagram



Agent
- name
- contact
- account

BankAccount
- number
- balance

Actor
- name
- dob
- sex
- Skills
- fee
- agent
- account

ActorSkill
- actorName
- skillCode

Skills
- name
- code

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Agent.name = Actor.agent

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Agent.name = Actor.agent
Agent.account =

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Agent.name = Actor.agent
Agent.account = BankAccount.number
Actor.account =

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

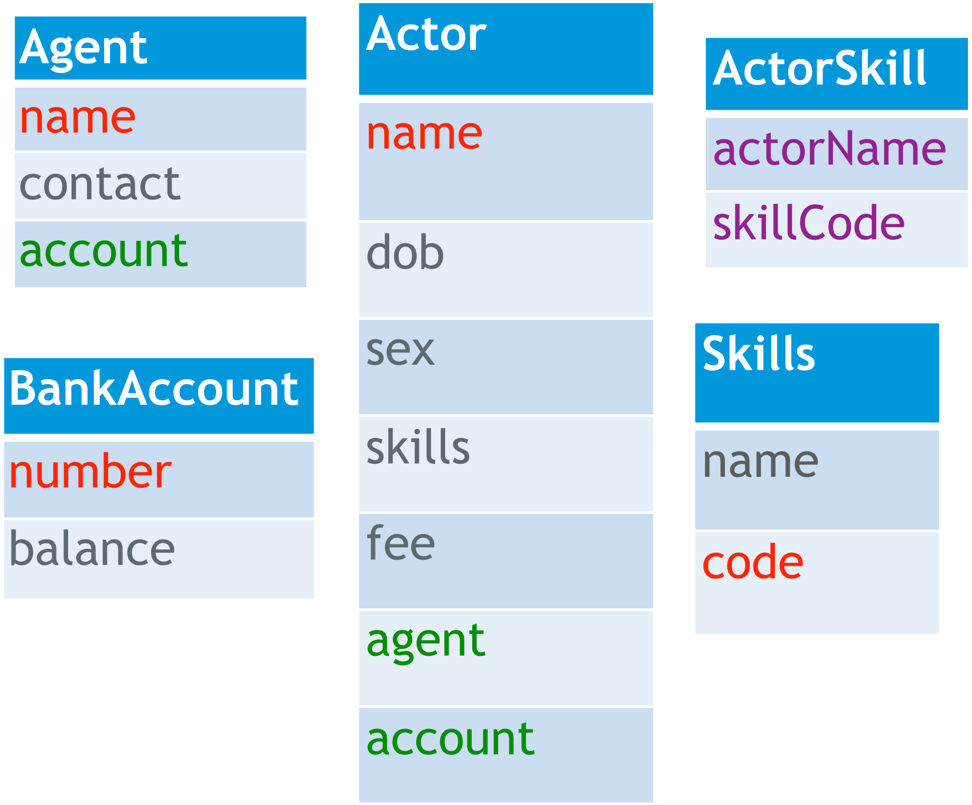
ActorSkill
actorName
skillCode

Skills
name
code

Agent.name = Actor.agent
Agent.account = BankAccount.number
Actor.account = BankAccount.number
Actor.name =

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences



Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Agent.name = Actor.agent
Agent.account = BankAccount.number
Actor.account = BankAccount.number
Actor.name = ActorSkill.actorName
ActorSkill.skillCode =

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Agent.name = Actor.agent
Agent.account = BankAccount.number
Actor.account = BankAccount.number
Actor.name = ActorSkill.actorName
ActorSkill.skillCode = skills.Code

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Every table should have **at least one** foreign key **or** a foreign key that points to it- otherwise it's not linked to other things.

Primary key
Foreign key
Primary and Foreign key

Foreign key equivalences

Agent
name
contact
account

BankAccount
number
balance

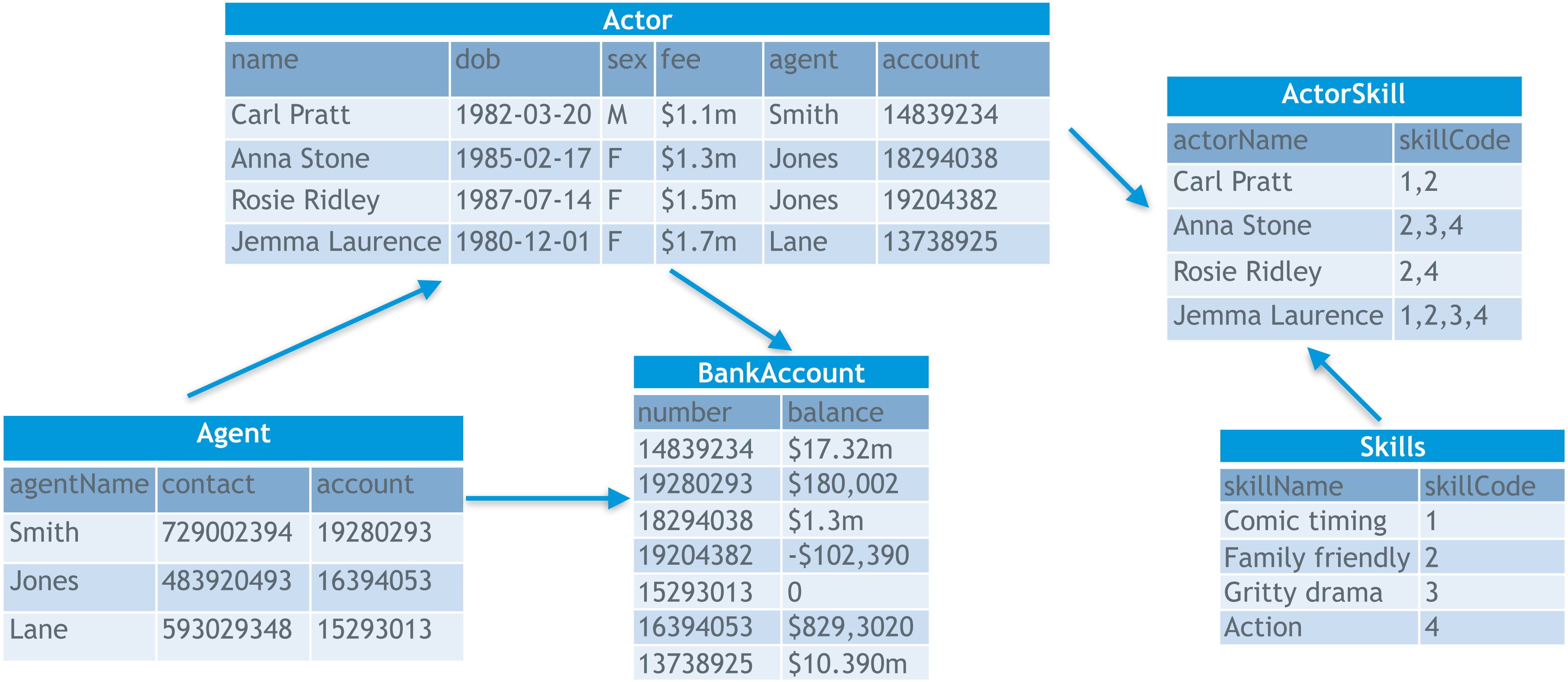
Actor
name
dob
sex
skills
fee
agent
account

ActorSkill
actorName
skillCode

Skills
name
code

Join tables consist **only** of foreign keys - because their purpose is to link tables.

Instantiated Relational diagram



Actor

name	dob	sex	fee	agent	account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

ActorSkill

actorName	skillCode
Carl Pratt	1,2
Anna Stone	2,3,4
Rosie Ridley	2,4
Jemma Laurence	1,2,3,4

Agent

agentName	contact	account
Smith	729002394	19280293
Jones	483920493	16394053
Lane	593029348	15293013

BankAccount

number	balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

Skills

skillName	skillCode
Comic timing	1
Family friendly	2
Gritty drama	3
Action	4

Instantiated Relational dia

ActorSkill

actorName	skillCode
Carl Pratt	1
Carl Pratt	2
Anna Stone	2
Anna Stone	3
Anna Stone	4
Rosie Ridley	2
Rosie Ridley	4
Jemma Laurence	1
Jemma Laurence	2
Jemma Laurence	3
Jemma Laurence	4

Actor

name	dob	sex	fee	agent	account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

BankAccount

number	balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

Agent

agentName	contact	account
Smith	729002394	19280293
Jones	483920493	16394053
Lane	593029348	15293013

Skills

skillName	skillCode
Comic timing	1
Family friendly	2
Gritty drama	3
Action	4

Creating the tables

```
CREATE TABLE Actor (
name VARCHAR(50) PRIMARY KEY,
DoB DATE,
sex CHAR(1),
fee DECIMAL,
agent VARCHAR(50),
account INT,
FOREIGN KEY (agent) REFERENCES Agent(name),
FOREIGN KEY (account) REFERENCES BankAccount(number)
) ENGINE=INNODB;
```

This will create the ActorSkill table
But if we put this straight into MySQL, it won’t work.

Creating the tables

```
CREATE TABLE Actor (
name VARCHAR(50) PRIMARY KEY,
DoB DATE,
sex CHAR(1),
fee DECIMAL,
agent VARCHAR(50),
account INT,
FOREIGN KEY (agent) REFERENCES Agent(name),
FOREIGN KEY (account) REFERENCES BankAccount(number)
) ENGINE=INNODB;
```

This will create the ActorSkill table
But if we put this straight into MySQL, it won’t work.

Because it is referring to **foreign keys** that **don’t exist.**
We need to create the tables that the foreign keys are referring to first.

Creating the tables

So first of all we add the tables it references:

```
CREATE TABLE BankAccount (
number INT PRIMARY KEY,
balance VARCHAR(10)
) ENGINE=INNODB;

CREATE TABLE Agent (
name VARCHAR(50) PRIMARY KEY,
contact INT,
account INT,
FOREIGN KEY (account) REFERENCES BankAccount(number)
) ENGINE=INNODB;
```

Creating the tables

- We’ll add the other tables and all of the instances into SQL
- Code can be found in lecture3_SQL.rtf

Creating the tables - adding primary keys to join tables

```
CREATE TABLE ActorSkill (
actorName VARCHAR(50) PRIMARY KEY,
skillCode INT PRIMARY KEY,
FOREIGN KEY (skillCode) REFERENCES Skill(code),
FOREIGN KEY (actorName) REFERENCES Actor(name)
) ENGINE=INNODB;
```

This will cause errors

Creating the tables - adding primary keys to join tables

```
CREATE TABLE ActorSkill (
actorName VARCHAR(50),
skillCode INT ,
PRIMARY KEY (actorName, skillCode),
FOREIGN KEY (skillCode) REFERENCES Skill(code),
FOREIGN KEY (actorName) REFERENCES Actor(name)
) ENGINE=INNODB;
```

Querying the table

Let’s ask a query that needs to get information from two tables

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

For example, let’s find out the bank balance of all actors. This is a one-to-one relationship

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

What columns do we want?

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

What columns do we want?
Name and *Balance*

What tables do these come from?

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

What columns do we want?
Name and *Balance*

What tables do these come from?
Actor and *BankAccount*

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

What columns do we want?
Name and *Balance*

What tables do these come from?
Actor and *BankAccount*

So we have:

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

```
SELECT name, balance FROM Actor, BankAccount
```

Will this work?

Actor					
Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

BankAccount	
Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

What columns do we want?
Name and *Balance*

What tables do these come from?
Actor and *BankAccount*

So we have:

```
SELECT name, balance FROM Actor, BankAccount
```

Will this work? Not really - why not?

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

What columns do we want?
Name and *Balance*

What tables do these come from?
Actor and *BankAccount*

So we have:

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

SELECT name, balance FROM Actor, BankAccount

Will this work? Not really - why not?
Because it returns all combinations of names and balances in these tables, even if they are not connected.
We need to restrict it.

Actor

Name	DoB	Sex	Fee	Agent	Account
Carl Pratt	1982-03-20	M	$1.1m	Smith	14839234
Anna Stone	1985-02-17	F	$1.3m	Jones	18294038
Rosie Ridley	1987-07-14	F	$1.5m	Jones	19204382
Jemma Laurence	1980-12-01	F	$1.7m	Lane	13738925

What columns do we want?
Name and *Balance*

What tables do these come from?
Actor and *BankAccount*

So we have:

BankAccount

Number	Balance
14839234	$17.32m
19280293	$180,002
18294038	$1.3m
19204382	-$102,390
15293013	0
16394053	$829,3020
13738925	$10.390m

```
SELECT name, balance FROM Actor, BankAccount
```

Will this work? Not really - why not?
Because it returns all combinations of names and balances in these tables, even if they are not connected.
We need to restrict it.

```
SELECT name, balance FROM Actor, BankAccount WHERE Actor.account =
BankAccount.number;
```

What about a table giving the agent of each actor.

What will:

```
SELECT name, name FROM Actor, Agent;
```

Give us?

What about a table giving the agent of each actor.

What will:

```
      SELECT name, name FROM Actor, Agent;
```

Give us?

Nothing! It won’t run because it is ambiguous. We need to be more specific.

```
      SELECT Actor.name, Agent.name FROM Actor, Agent;
```

What about a table giving the agent of each actor.

What will:

```
SELECT name, name FROM Actor, Agent;
```

Give us?

Nothing! It won’t run because it is ambiguous. We need to be more specific.

```
SELECT Actor.name, Agent.name FROM Actor, Agent;
```

But we also need to specify the link:

```
SELECT Actor.name, Agent.name FROM Actor, Agent WHERE
Actor.agent = Agent.name;
```

What about a table giving the agent of each actor.

What will:

```
SELECT name, name FROM Actor, Agent;
```

Give us?

Nothing! It won't run because it is ambiguous. We need to be more specific.

```
SELECT Actor.name, Agent.name FROM Actor, Agent;
```

But we also need to specify the link:

```
SELECT Actor.name, Agent.name FROM Actor, Agent WHERE
Actor.agent = Agent.name;
```

This will work, but it's a bit confusing because the columns have the same name. We can tell SQL to rename the columns:

What about a table giving the agent of each actor.

What will:

```
SELECT name, name FROM Actor, Agent;
```

Give us?

Nothing! It won't run because it is ambiguous. We need to be more specific.

```
SELECT Actor.name, Agent.name FROM Actor, Agent;
```

But we also need to specify the link:

```
SELECT Actor.name, Agent.name FROM Actor, Agent WHERE
Actor.agent = Agent.name;
```

This will work, but it's a bit confusing because the columns have the same name. We can tell SQL to rename the columns:

```
SELECT Actor.name AS actorName, Agent.name AS agentName FROM Actor, Agent
WHERE Actor.agent = Agent.name;
```

Joins

Joins are an important part of relational databases. We have met **join tables** already. How do we use joins for selecting?

```
SELECT name, balance FROM Actor JOIN BankAccount ON
account = number;
```

Joins

Joins are an important part of relational databases. We have met **join tables** already. How do we use joins for selecting?

```
SELECT name, balance FROM Actor JOIN BankAccount ON
account = number;
```

This SELECT statement:

- States the columns it is interested in (name, balance)
- From the tables it wants to join (Actor, BankAccount)
- And on what the tables are joining (account=number)

Joins

Joins are an important part of relational databases. We have met **join tables** already. How do we use joins for selecting?

SELECT name, balance FROM Actor JOIN BankAccount ON account = number;

This SELECT statement:

- States the columns it is interested in (name, balance)
- From the tables it wants to join (Actor, BankAccount)
- And on what the tables are joining (account=number)

Joins

Joins are an important part of relational databases. We have met **join tables** already. How do we use joins for selecting?

SELECT name, balance FROM Actor JOIN BankAccount ON account = number;

This SELECT statement:

- States the columns it is interested in (name, balance)
- From the tables it wants to join (Actor, BankAccount)
- And on what the tables are joining (account=number)

Joins

SELECT name, balance FROM Actor, BankAccount WHERE Actor.account = BankAccount.number;

and

SELECT name, balance FROM Actor JOIN BankAccount ON account = number;

Are **functionally equivalent.**

Join statements are generally preferred because they are clearer and more explicit.

You can use either in assessment for this course, unless you are specifically told to use one/both.

Further queries to try

```
SELECT Actor.name AS actorName, Agent.name AS agentName FROM
Actor, Agent WHERE Actor.agent = Agent.name;
```

Further queries to try

```
SELECT Actor.name AS actorName, Agent.name AS agentName FROM
Actor, Agent WHERE Actor.agent = Agent.name;
```

and

```
SELECT Actor.name as actorName, Agent.name AS agentName FROM
Actor JOIN Agent ON Actor.agent = Agent.name;
```

So what part of what we have done is not E-R?

- Foreign keys and join tables
  - They are about the logical links between the conceptualisation of the database, rather than the data itself, so if we are being technically precise they do not belong in an E-R diagram
- This distinction will not be used in this course, and we will refer to them all as relational diagrams.

What next?

- You will practice using these in the next lab.
- You will be given some queries in English to try to create SQL queries from.