CS 5530

Database System
Spring 2020

Finish ER Model

Translating ER to Schemas

SQL Tables

Weak Entities

- A weak entity can’t be identified by its own attributes

- It is identified by a combination of:
    - its own attribute(s)
    - and another entity’s key

Weak Entities

- A weak entity can't be identified by its own attributes

- It is identified by a combination of:
  - its own attribute(s)
  - and another entity's key

- i.e., part of its identity is defined by another entity

Weak Entity Sets

- Another way to think about it:
  - “How do I require an entity to copy a key from another entity?”

Example

- Consider the difference between a *course* and a *class*

  - Course strong
    - Subject
    - Number
    - Name
    - Description

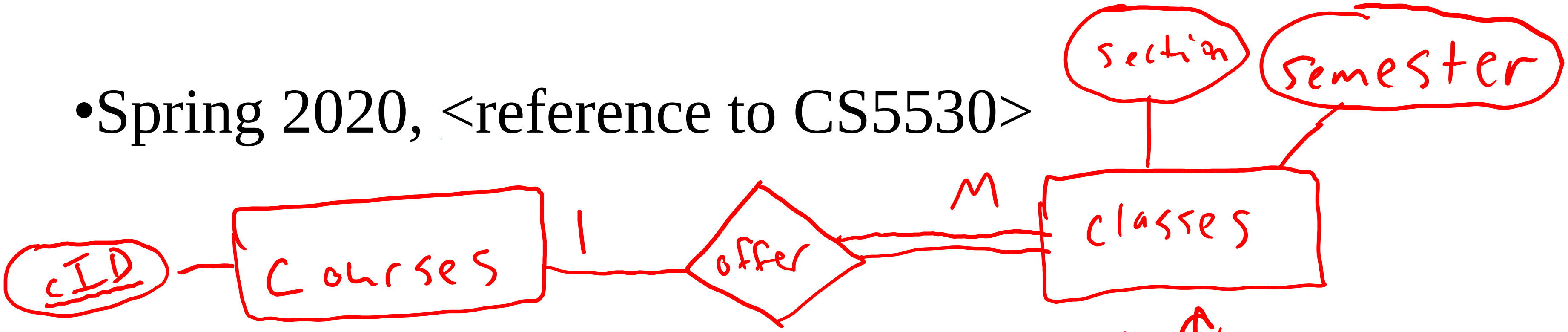
- CS 5530 Database Systems, “in this class we study …”,

Example

- Consider the difference between a *course* and a *class*

  - Class
    - Semester
    - <reference to course>

- Spring 2020, <reference to CS5530>

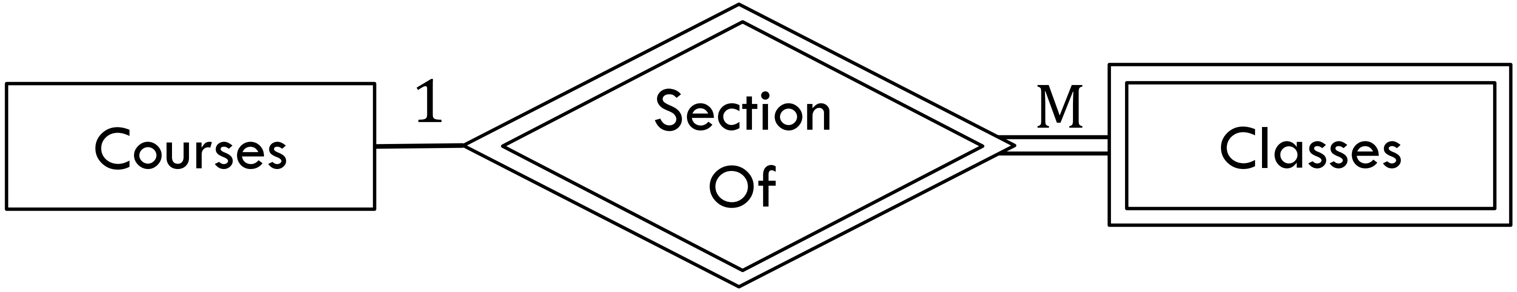


1	S20	CS5530
2	S19	CS5530
~~3~~	~~S20~~	~~CS5530~~

~~weak~~

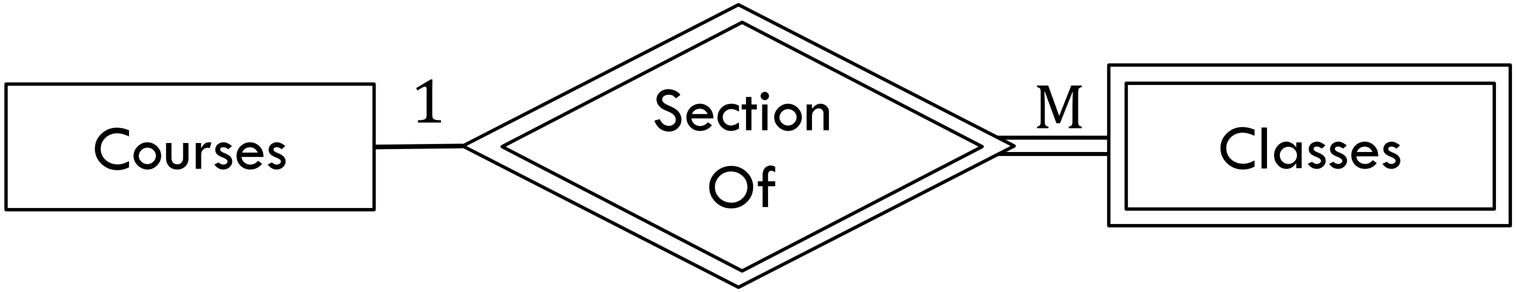
Weak Entity Set

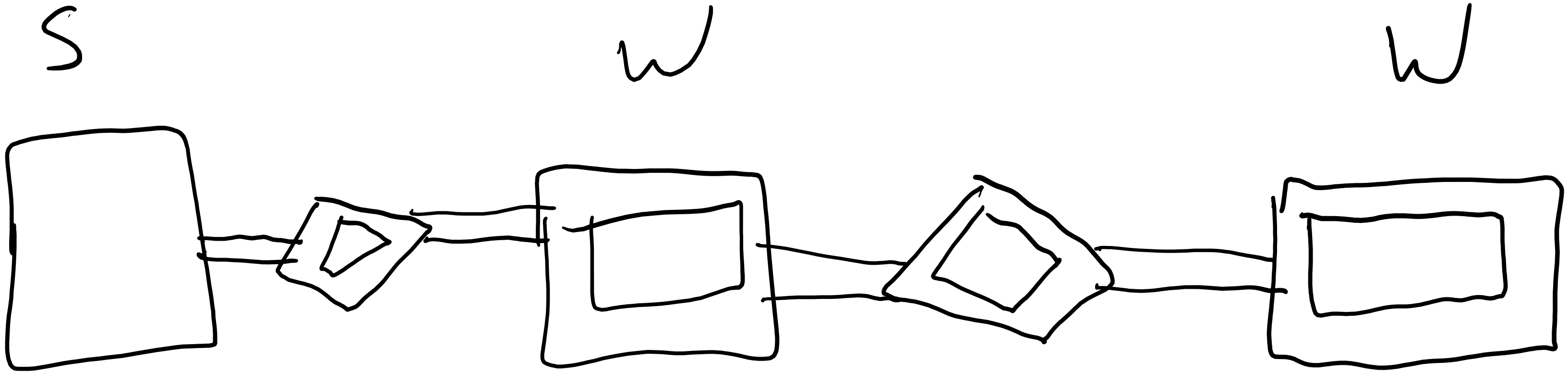
- `Classes` is a weak entity set



Weak Entity Set

- `Classes` is a weak entity set
  - Double rectangle for the weak entity set
  - Double diamond for the supporting relationship set
  - Supporting relationship must be 1-to-M, and weak entity must participate




S
W
W

Weak Entities

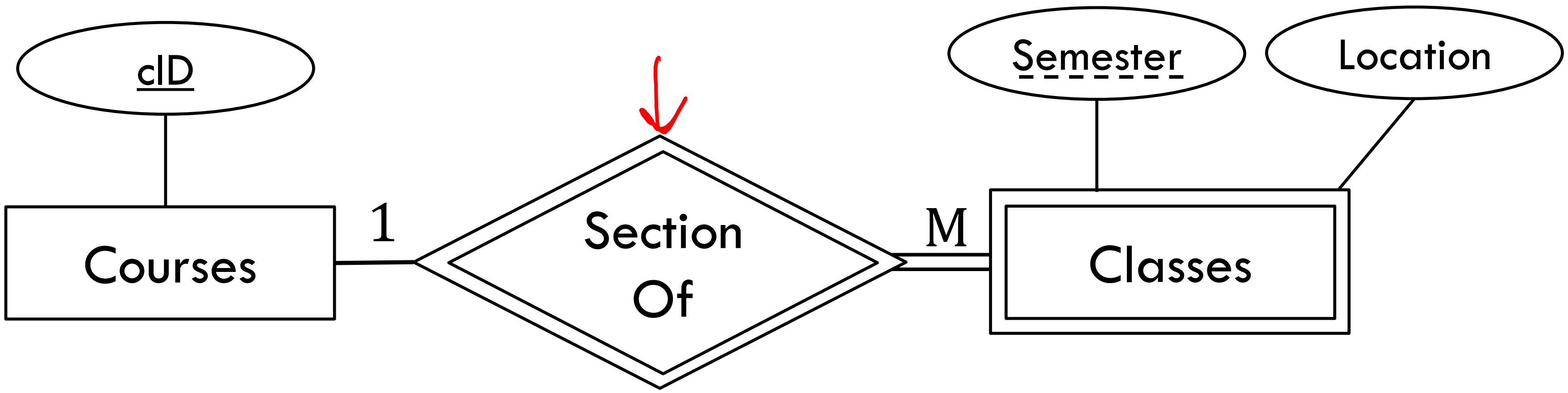
- A weak entity can’t be identified by its own attributes

- It is identified by a combination of:
  - its own attribute(s)
  - and another entity’s key

- i.e., part of its identity is defined by another entity

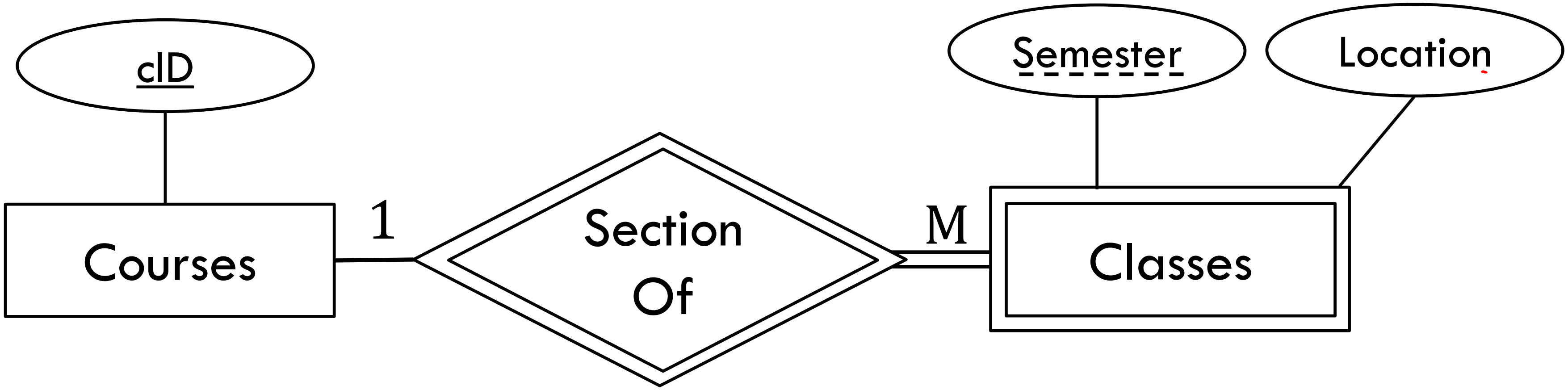
Partial Key

- A weak entity set has a partial key
    - Dashed underline



Partial Key

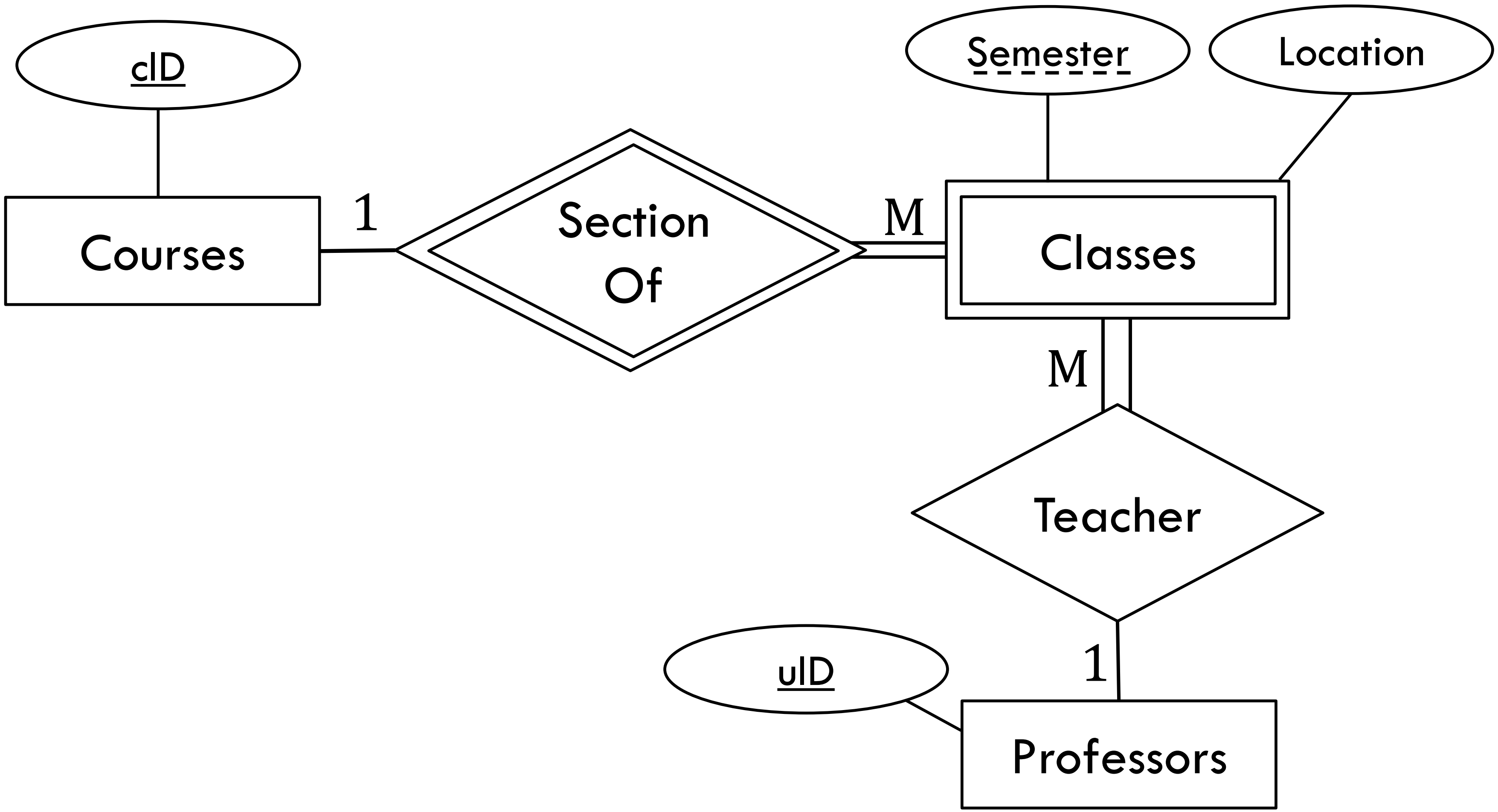
- A weak entity set has a partial key
  - Dashed underline
  - Gets combined with another key
  - `{cID, Semester}` is the key for `Classes`



Weak Entity Sets

- Weak entity sets can have non-supporting relationships

cID, Semester



ER → RM Algorithm

- We’ve been using a “naïve”, but correct relational model
  - Sometimes resulting in unnecessary tables

ER → RM Algorithm

- If we start with a good ER model, the translation to a good RM is mechanical

ER → RM Algorithm

- Basic algorithm:

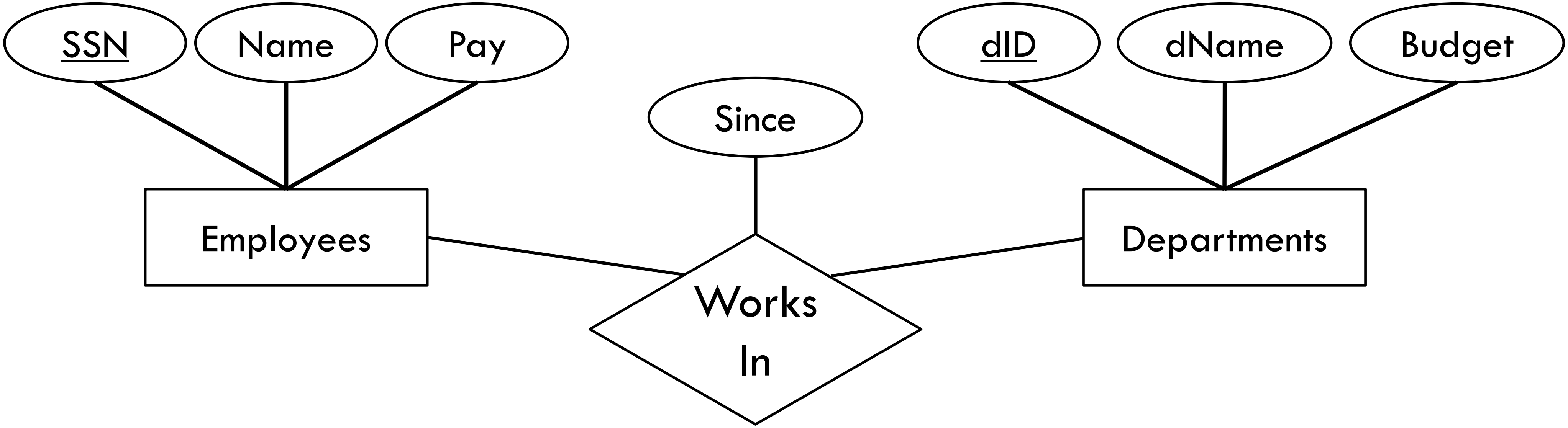
  1. Every entity set becomes a schema

  2. Relationship sets *might* become schemas depending on cardinality, otherwise they are *merged*

Entity Set to Schema

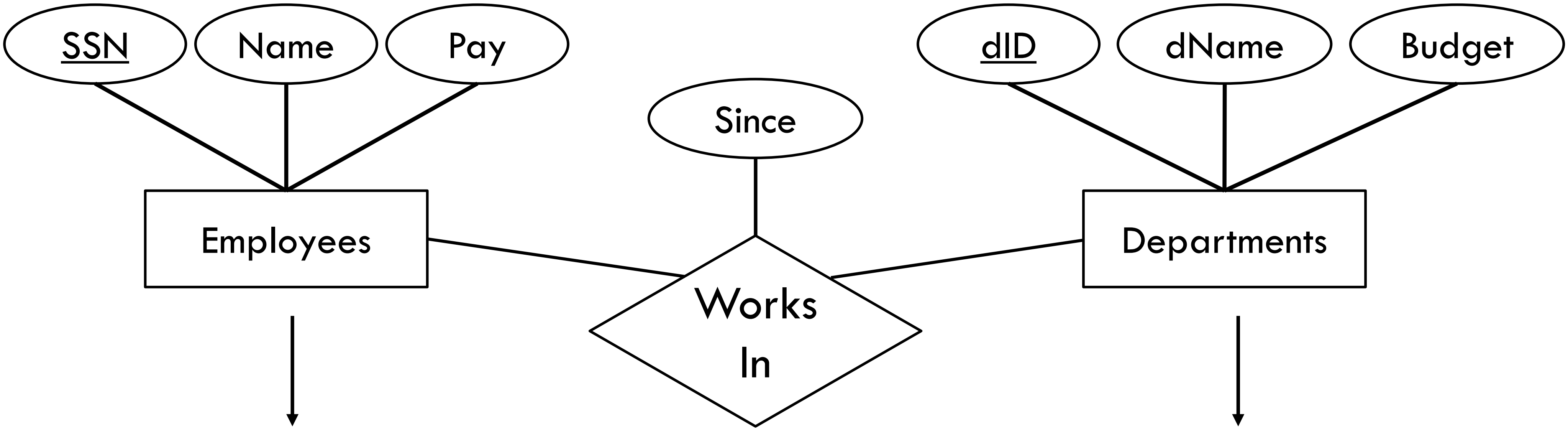
- **Entity Sets**
- Every entity set translates directly to a schema
- Attributes become columns
- Set the key attribute(s) as the primary key

Entity Set to Schema



Entity Set to Schema

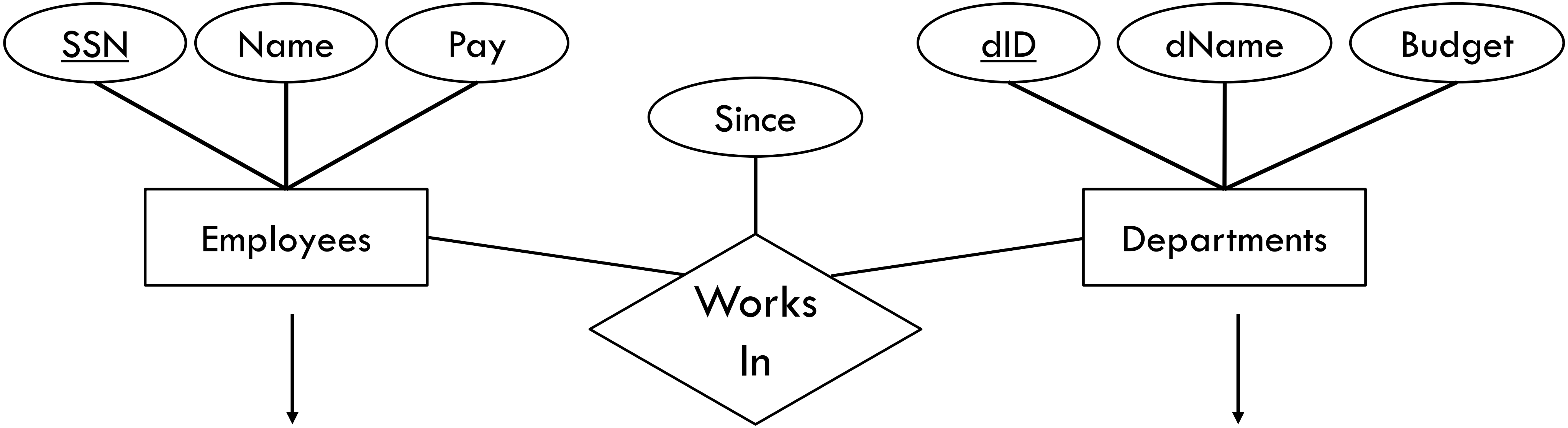
NN = not null



SSN	Name	Pay
…	…	…
NN	NN	NN
PK		

dID	dName	Budget
…	…	…
NN	NN	NN
PK		

Entity Set to Schema



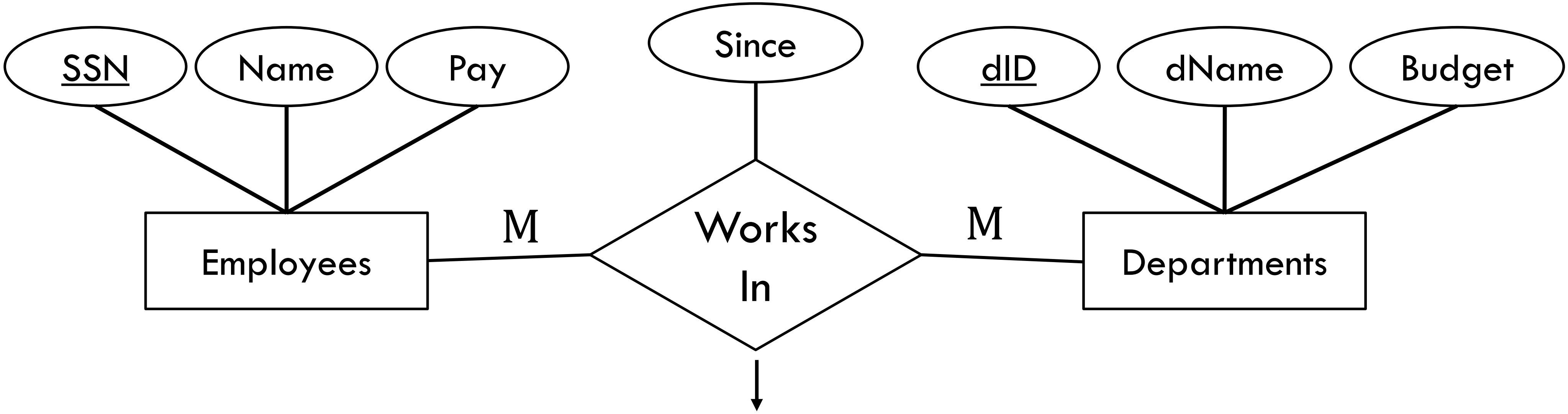
SSN	Name	Pay
…	…	…
NN	NN	NN
PK		

?

dID	dName	Budget
…	…	…
NN	NN	NN
PK		

Relationship Set to Schema

- **Many-to-Many**
  - New schema for the relationship



SSN

Name

Pay

Since

dID

dName

Budget

Employees

M

Works In

M

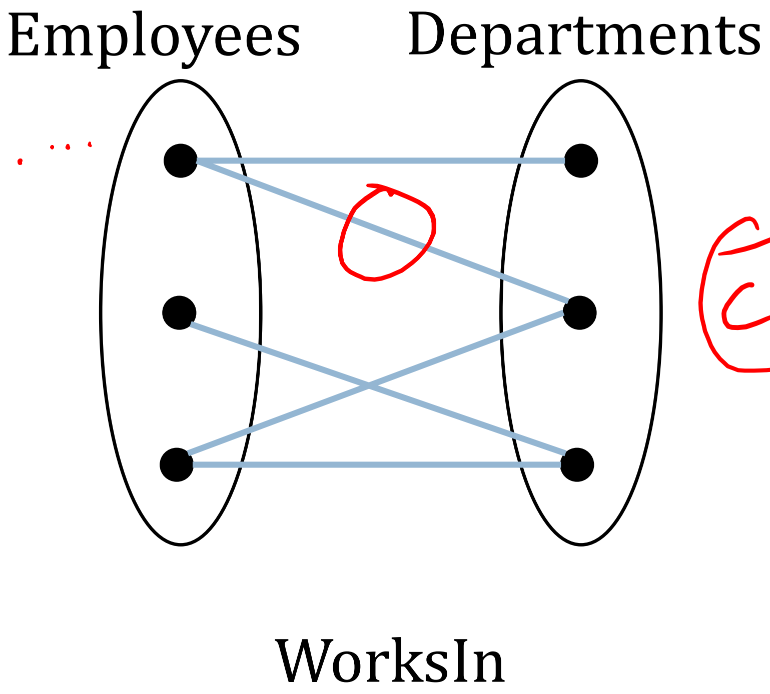
Departments

?	Since	?
…	…	…

Relationship Set to Schema

- What makes a reference between employee ←→department?

{1, CS}

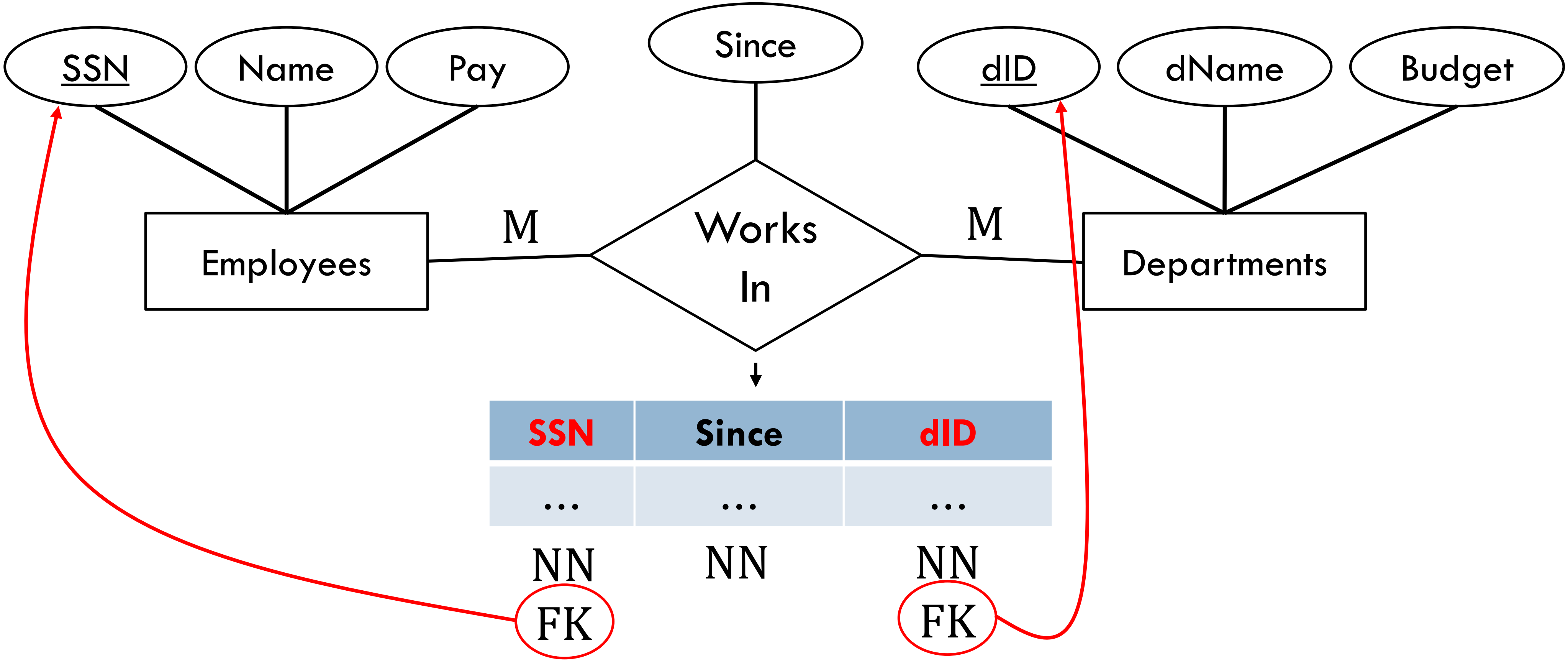


1 steve, ...

CS, CompSci, ...

Relationship Set to Schema

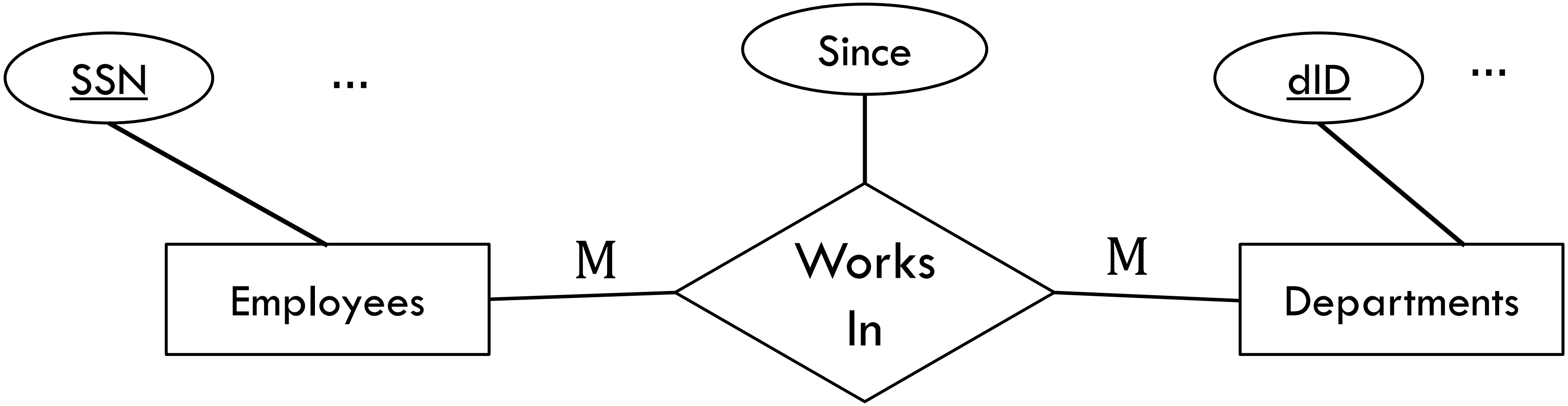
- **Many-to-Many**
  - Primary keys of relating entities as foreign keys



SSN	Since	dID
…	…	…

Relationship Set to Schema

- **Many-to-Many**
  - What is the key of the `WorksIn` table?

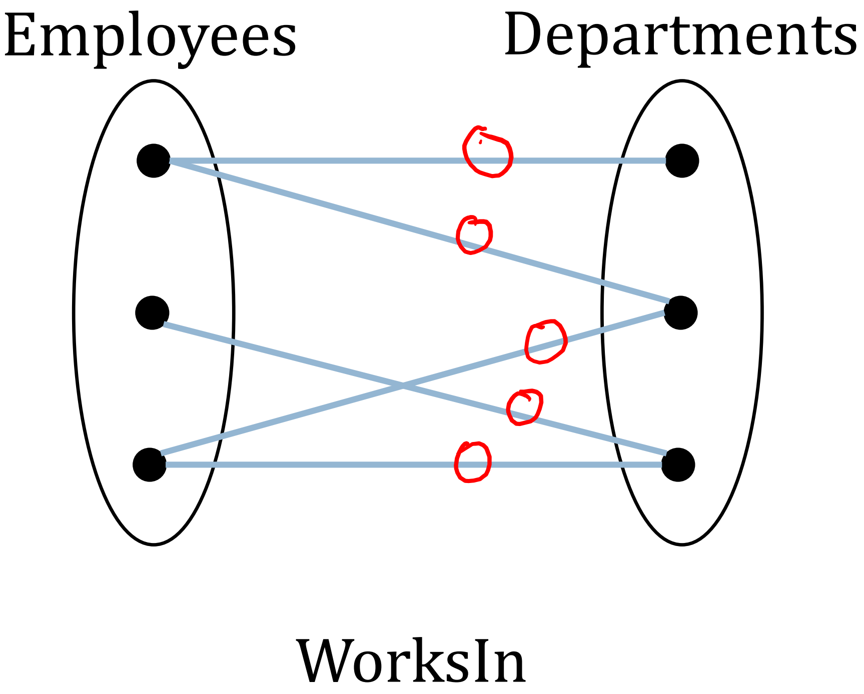


SSN	Since	dID
…	…	…
NN FK	NN	NN FK

PK?

Relationship Set to Schema

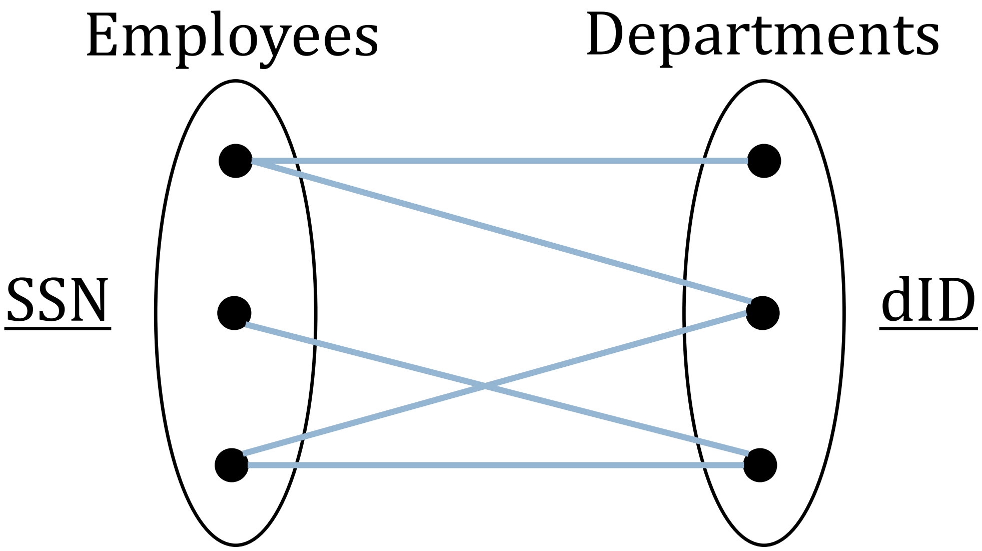
- How do we uniquely identify a line (many-to-many)?



{ssN, dID}

Relationship Set to Schema

- How do we uniquely identify a line (many-to-many)?



Employees

Departments

SSN

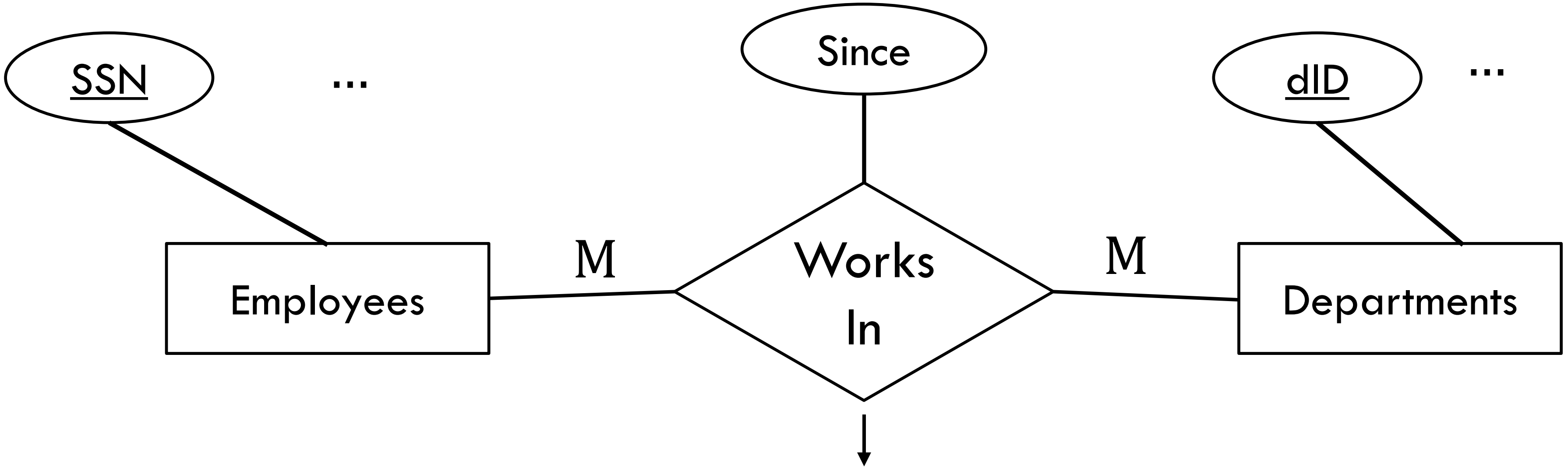
dID

WorksIn

(SSN, dID)

Relationship Set to Schema

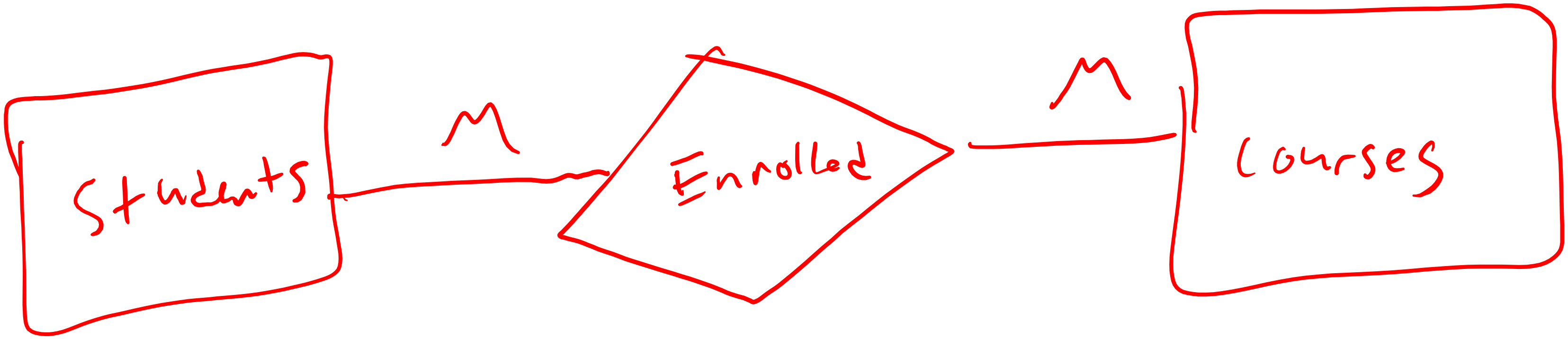
- **Many-to-Many**
  - Key is the combination of the foreign keys



SSN	**dID**	**Since**
…	…	…
NN	NN	NN
FK	FK	

-------PK-------

M-M Example



Students

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

Enrolled

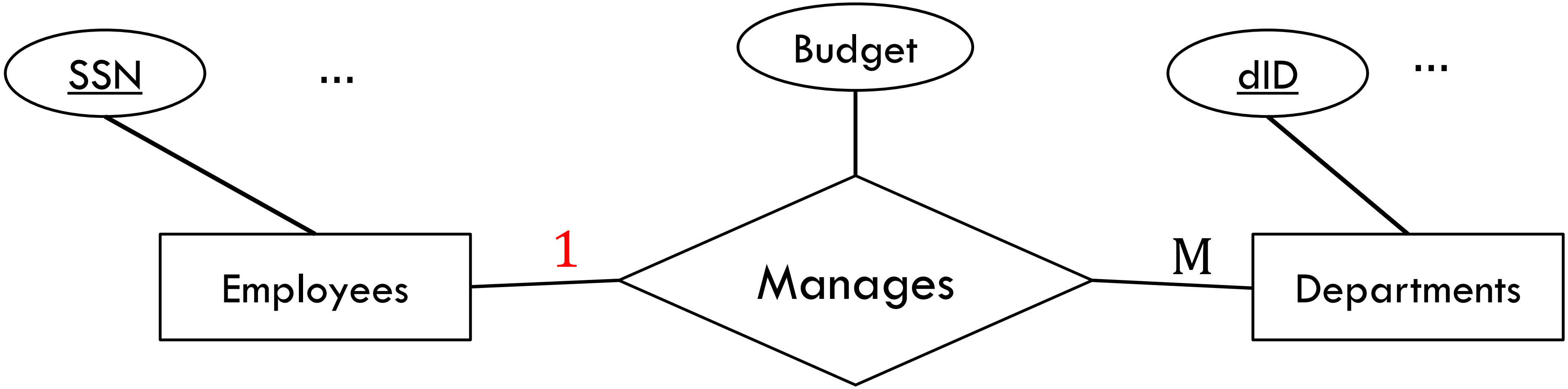
sID	cID	Grd
1	3500	A
1	3810	A-
1	5530	A
2	3810	A
2	5530	B

Courses

cID	Name
3500	SW Practice
3810	Architecture
5530	Databases

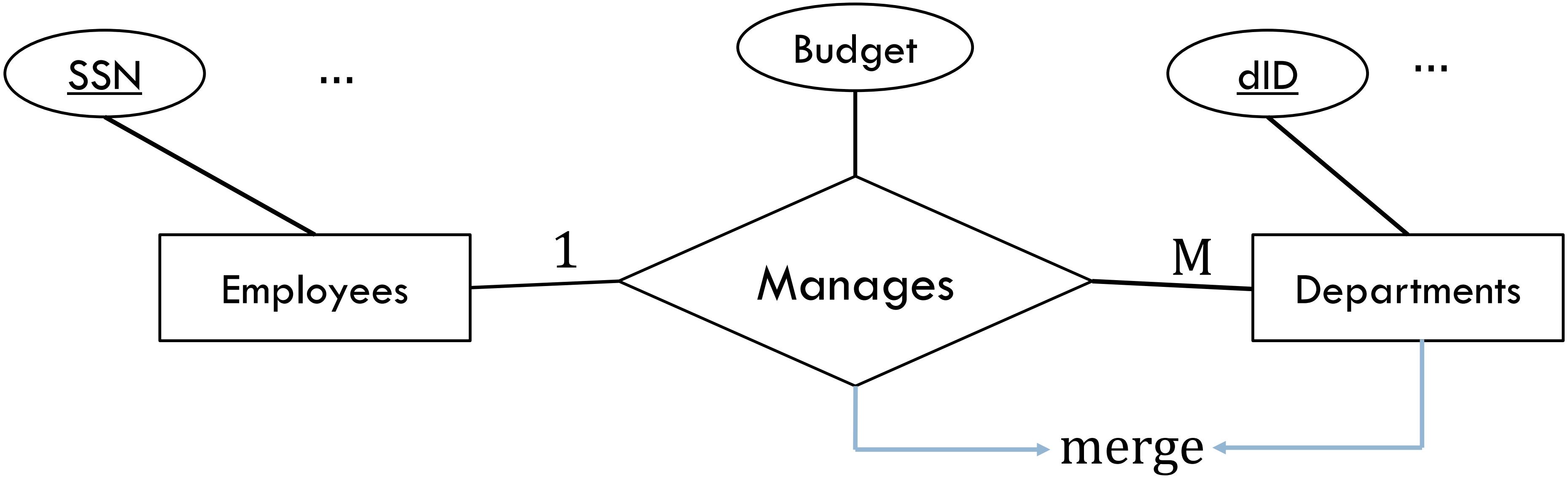
Relationship Set to Schema

- **1-to-Many**
  - Department can only have 1 manager



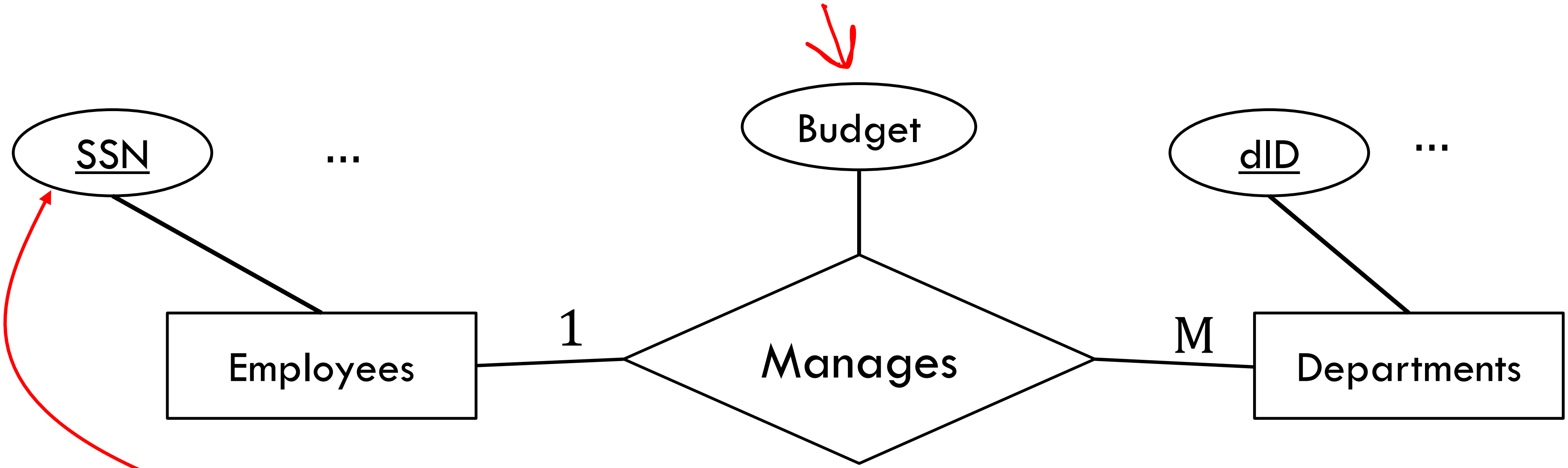
Relationship Set to Schema

- **1-to-Many**
  - Department can only have 1 manager
  - Merge the relationship into the entity



Relationship Set to Schema

- **1-to-Many**
  - Department can only have 1 manager

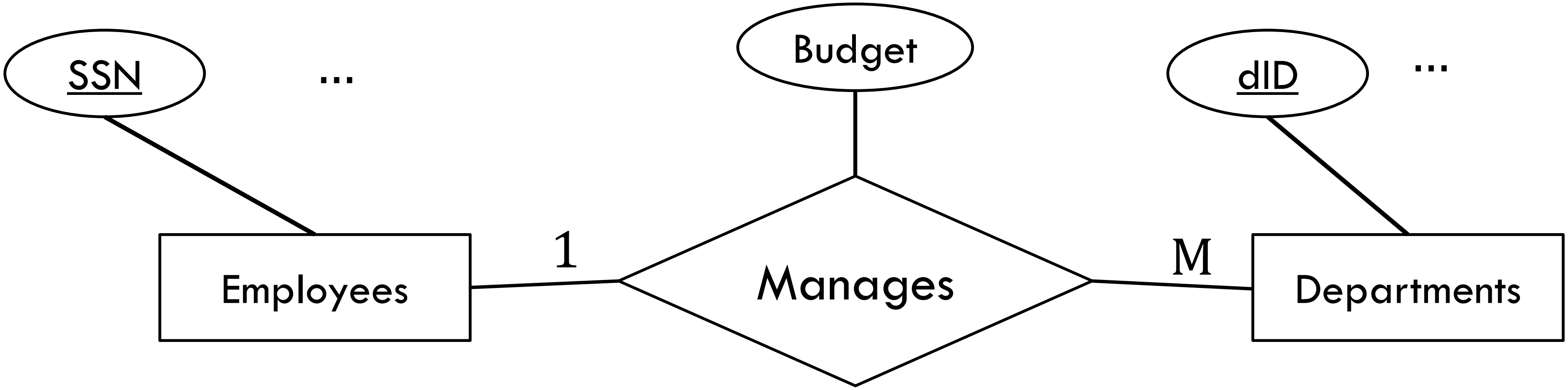


SSN	…
…	…
NN	NN
PK	

dID	…	Mnger	Budget
…	…		
NN	NN	FK	
PK			

Relationship Set to Schema

- **1-to-Many**
  - Can `Mnger` and `Budget` be NULL?

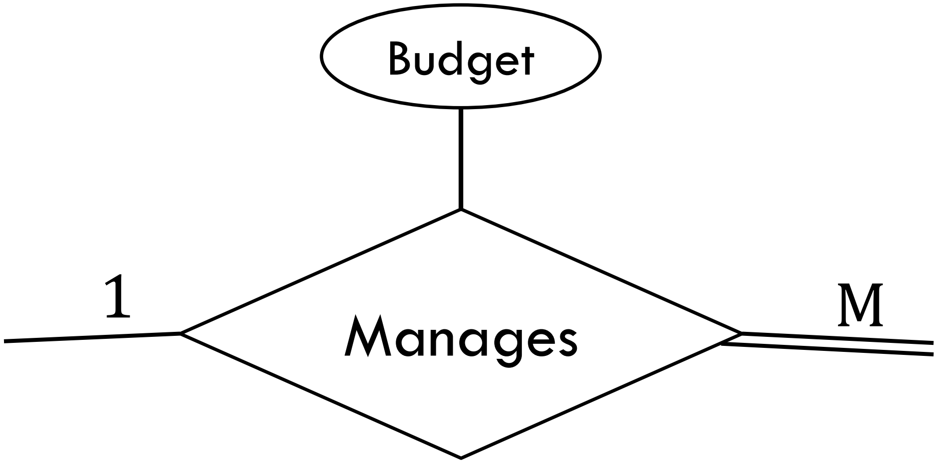


SSN	…
…	…
NN PK	NN

dID	…	Mnger	Budget
…	…		
NN PK	NN	FK NN?	NN?

Participation Constraints

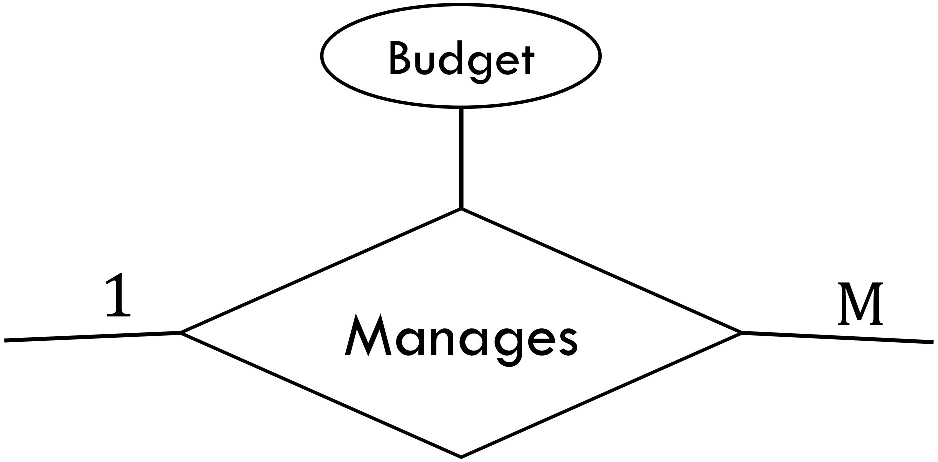
- If participation is required, set `NOT NULL`
- Else, `NULL` is allowed



…	Mnger	Budget
…		

FK NN

NN



…	Mnger	Budget
…		

FK

Foreign Key + NULL

- Referential integrity only enforced when inserting non-null data

- Not all departments have a manager

Employees

SSN	Name	Pay
1	Joe	…
2	Steve	…
3	Meg	…

Departments

dID	dName	Budget	Mnger
a	Chem	…	1
b	Phys	NULL	NULL
c	CS	…	3

FK

Naïve Library

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379
1005	978-0394823379
1006	978-0062278791

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
4	888-8888
4	999-9999

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Reduced

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN	CheckedOutBy
1001	978-0590353427	1
1002	978-0590353427	NULL
1003	978-0679732242	NULL
1004	978-0394823379	1
1005	978-0394823379	4
1006	978-0062278791	4

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
4	888-8888
4	999-9999

Titles

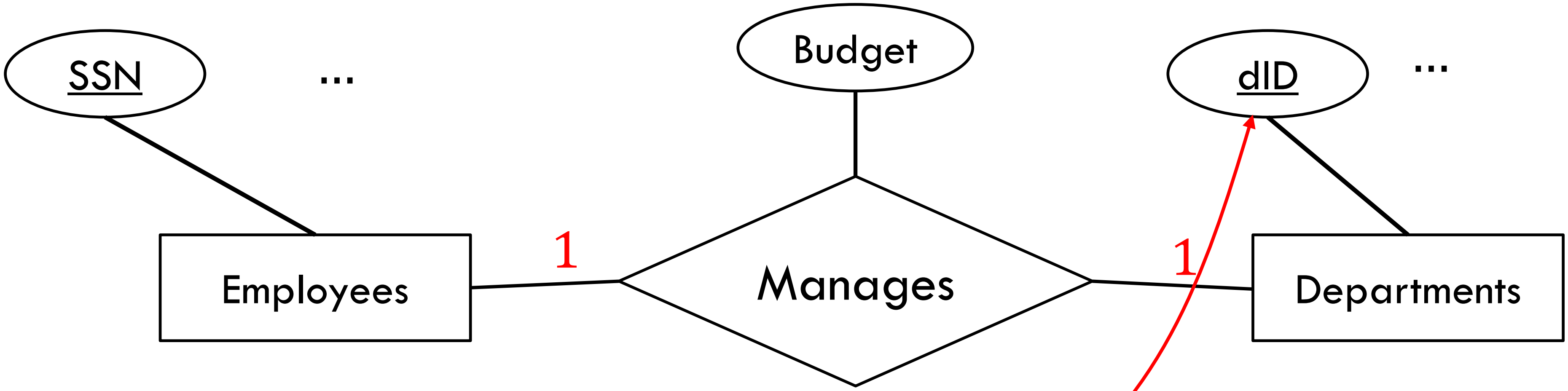
ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Performance

- In general:
  - Fewer joins = better performance
  - Fewer tables = fewer joins
- 1-to-M don’t need their own tables

Relationship Set to Schema

- **1-to-1**
  - Treat as 1-to-M
  - Merge relationship into one of the other tables



SSN	…	Manages	Budget
…	…	…	…

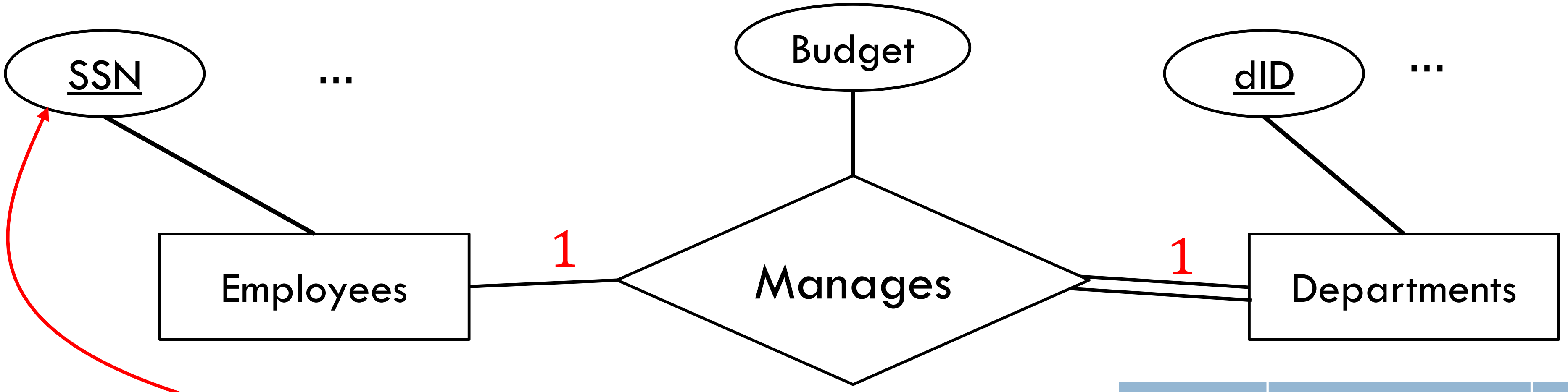
PK　　FK

NN　　UNIQUE

dID	…
…	…

Relationship Set to Schema

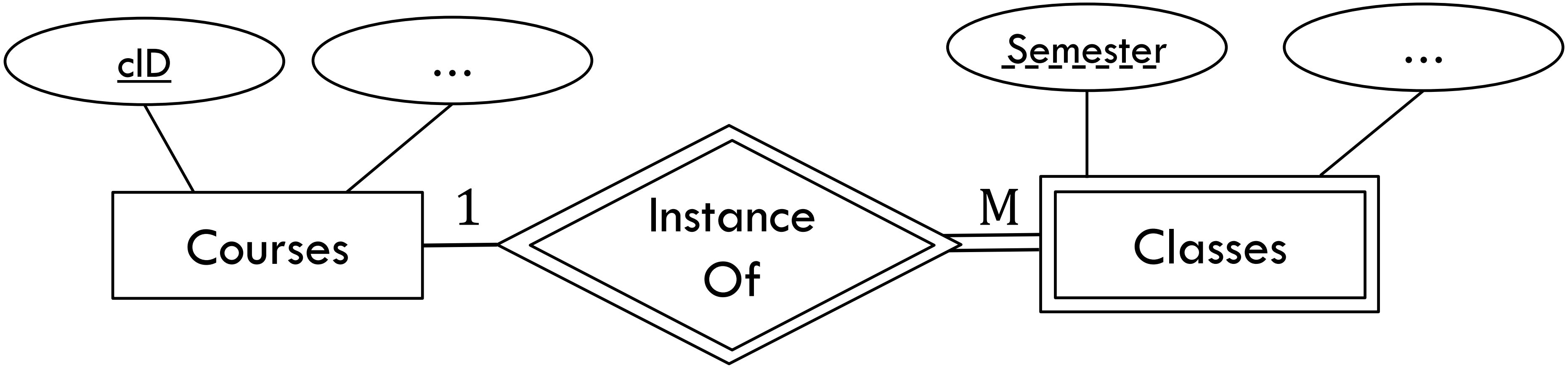
- **1-to-1**
  - If participation required on one side, use that side
  - Add `NOT NULL`



dID	ManagedBy	Budget
…	…	…
PK NN	FK UNIQUE NN	NN

Relationship Set to Schema

- **Supporting Relationship**
  - A supporting relationship is 1-to-M
  - Same procedure, except supporting key is combined with partial key



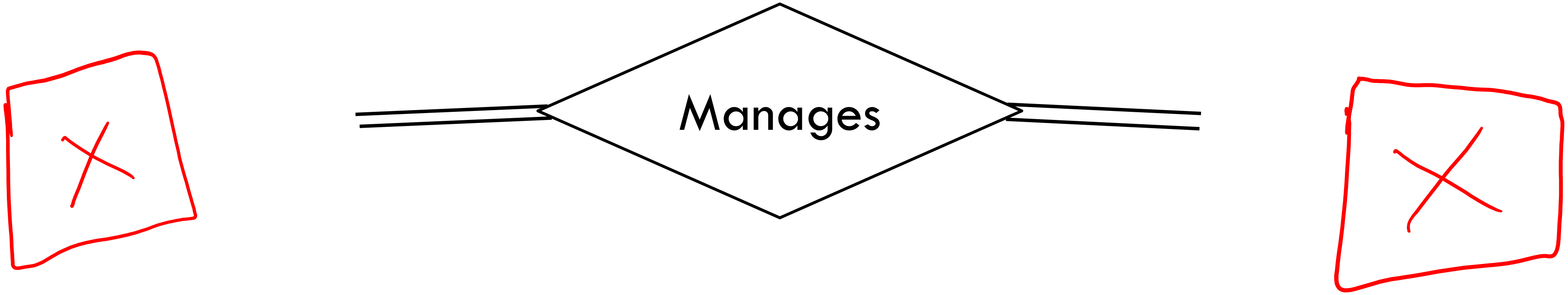
cID	...
...	...
NN	NN
PK	

cID	Semester	...
...	...	...
NN	NN	
FK		

-------PK-------

Double Participation

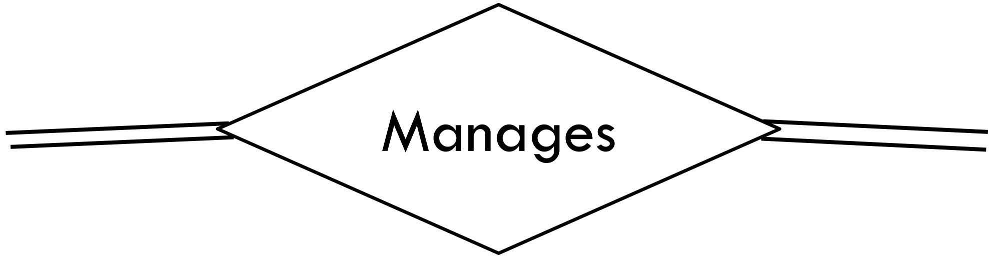
- If participation required on **both** sides…
  - Regardless of cardinality



→ 1 , ?

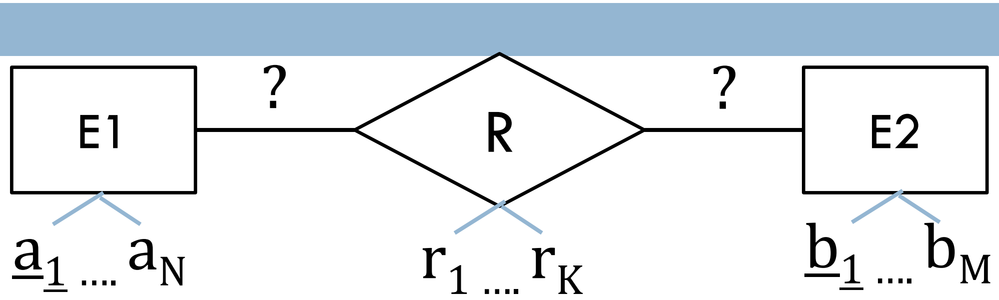
Double Participation

- If participation required on **both** sides…
  - Regardless of cardinality
  - Chicken/egg problem



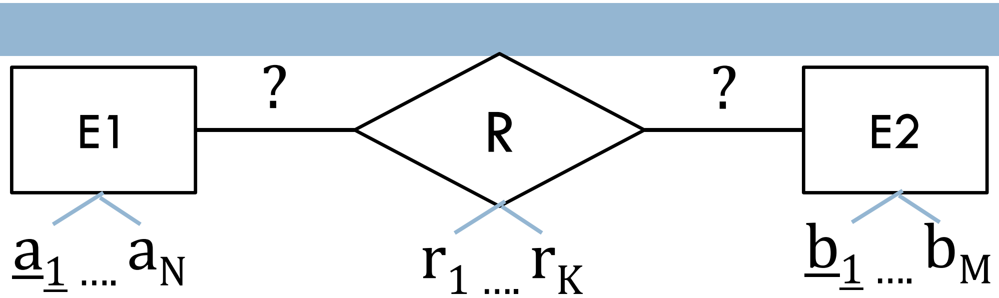
- Difficult to capture with schema design
  - Instead, enforce with SQL commands

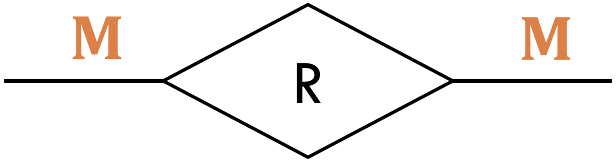
Algorithm Summary



Algorithm Summary

🟥 = FK



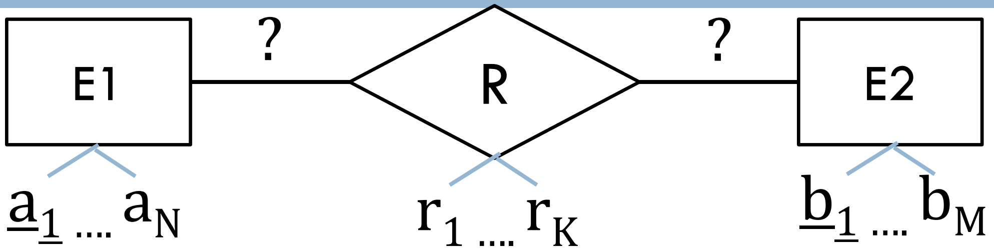


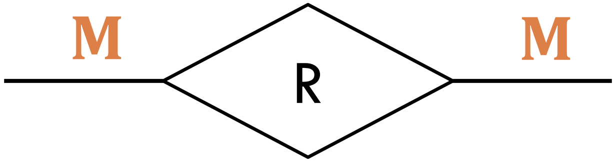
$E1 = \{\underline{a_1}, \ldots, a_N\}$ $E2 = \{\underline{b_1}, \ldots, b_N\}$

$R = \{\underline{a_1}, \underline{b_1}, r_1, \ldots r_k\}$

Algorithm Summary

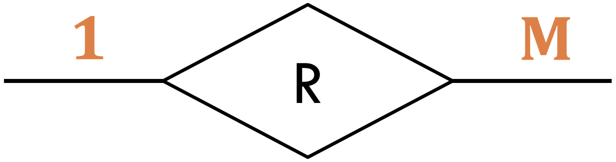
🟥 = FK





E1 = {$\underline{a_1}$, …, a_N}   E2 = {$\underline{b_1}$, …, b_N}

R = {$\underline{a_1}$, $\underline{b_1}$, r_1, … r_k}

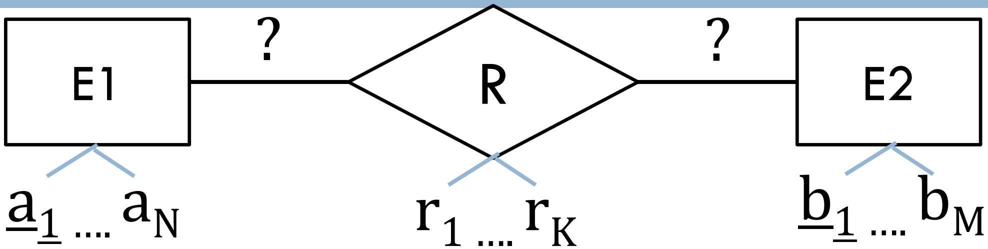


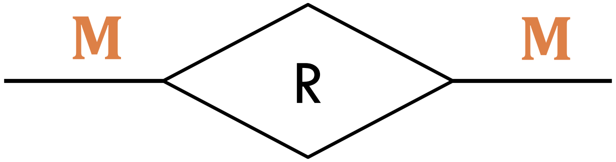
E1 = {$\underline{a_1}$, …, a_N}

E2 = {$\underline{b_1}$, …, b_N, a_1, r_1, … r_k}

Algorithm Summary

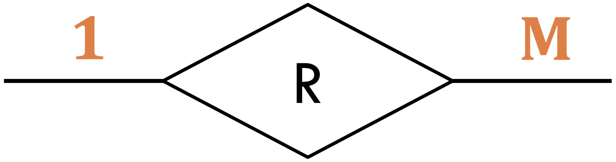
🟥 = FK





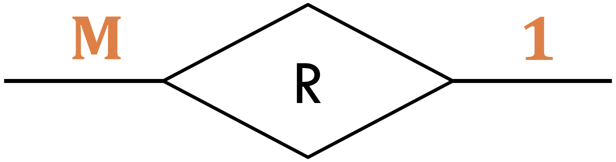
$E1 = \{\underline{a_1}, …, a_N\}$ $E2 = \{\underline{b_1}, …, b_N\}$

$R = \{\underline{a_1}, \underline{b_1}, r_1, … r_k\}$



$E1 = \{\underline{a_1}, …, a_N\}$

$E2 = \{\underline{b_1}, …, b_N, a_1, r_1, … r_k\}$

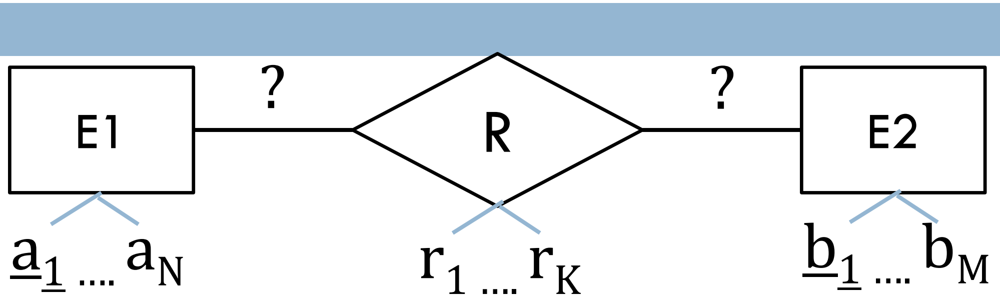


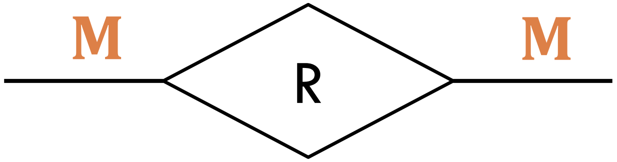
$E1 = \{\underline{a_1}, …, a_N, b_1, r_1, … r_k\}$

$E2 = \{\underline{b_1}, …, b_N\}$

Algorithm Summary

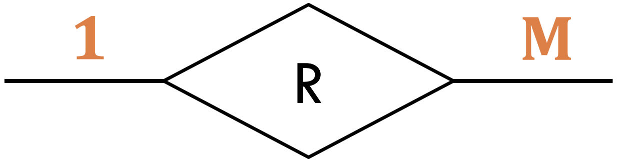
🟥 = FK





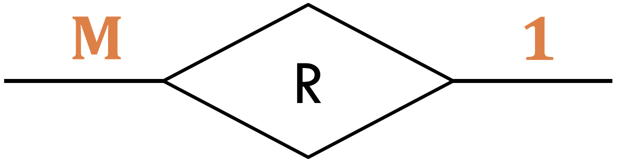
$\mathtt{E1} = \{\underline{a}_1, \ldots, a_N\}$ $\mathtt{E2} = \{\underline{b}_1, \ldots, b_N\}$

$\mathtt{R} = \{\textcolor{red}{\underline{a}_1}, \textcolor{red}{\underline{b}_1}, r_1, \ldots r_k\}$



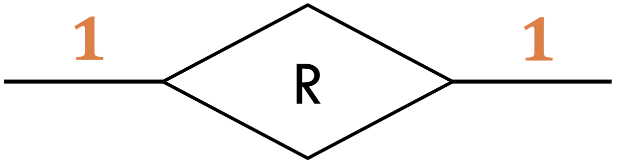
$\mathtt{E1} = \{\underline{a}_1, \ldots, a_N\}$

$\mathtt{E2} = \{\underline{b}_1, \ldots, b_N, \textcolor{red}{a_1}, r_1, \ldots r_k\}$



$\mathtt{E1} = \{\underline{a}_1, \ldots, a_N, \textcolor{red}{b_1}, r_1, \ldots r_k\}$

$\mathtt{E2} = \{\underline{b}_1, \ldots, b_N\}$



Treat as 1:M or M:1
Mark foreign key as unique

Algorithm Summary

- `NOT NULL` determined by participation constraints
- Total participation on both sides – enforced in software, not in schema

Creating Tables in SQL

```
create table <name> (
  <column1Name> <type> <properties>,
  <column2Name> <type> <properties>,
  …,
  <table properties>
);
```

- Properties are optional

MySQL Data Types

- Numeric
  - Integers – int, <tiny, small, medium, big>int, <unsigned>
  - Reals – float, double, decimal

MySQL Data Types

- Numeric
  - Integers – int, <tiny, small, medium, big>int, <unsigned>
  - Reals – float, double, decimal

- Dates (very common in DBs)
  - Date, datetime, time stamp, time, year

MySQL Data Types

- Numeric
  - Integers – int, <tiny, small, medium, big>int, <unsigned>
  - Reals – float, double, decimal

- Dates (very common in DBs)
  - Date, datetime, time stamp, time, year

- Strings
  - char(m), varchar(m)

MySQL Data Types

- Numeric
  - Integers – int, <tiny, small, medium, big>int, <unsigned>
  - Reals – float, double, decimal

- Dates (very common in DBs)
  - Date, datetime, time stamp, time, year

- Strings
  - char(m), varchar(m)

- Blobs – Binary Large Objects

- Enums

Strings

- char(N) - exactly N characters
- varchar(N) - up to N characters

Quiz

- Best type for ‘CardNum’?

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Quiz

- Best type for ‘CardNum’?

  int unsigned

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Quiz

- Best type for ‘Author’?

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Quiz

- Best type for ‘Author’?

  varchar(…)

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Quiz

- Best type for ‘ISBN’?

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Quiz

- Best type for ‘ISBN’?

  char(14)

  unless we get rid of the dashes

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Strings

- CHAR(N) - exactly N characters
  - ISBN
  - Phone number

- VARCHAR(N) - up to N characters
  - Title
  - Author
  - Name

Strings

- How to pick (N) for VARCHAR(N)?
- Requires N bytes for storage plus 1 or 2 bytes for size
  - 1 size byte if N <= 255
  - 2 size bytes if N > 255

Strings

- How to pick (N) for VARCHAR(N)?
- Requires N bytes for storage plus 1 or 2 bytes for size
  - 1 size byte if N <= 255
  - 2 size bytes if N > 255
- Pick the smallest value that is always large enough

Properties

- Column properties:
  - `NOT NULL`
  - `DEFAULT ‘hello’`
  - `AUTO_INCREMENT`

- Table properties:
  - `PRIMARY KEY (column1, …)`
  - `UNIQUE (column1, …)`
  - `FOREIGN KEY …`
  - `INDEX (column1, …)`

Not Null

- Improves index optimizations
- …it’s not just for table design
- Specifying `NOT NULL` is not required on key columns
  - But SQL will automatically set it

Quiz

- Appropriate properties for ‘CardNum’?

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Quiz

- Appropriate properties for ‘CardNum’?

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

- `not null`
- `auto_increment`
- `primay key`

Quiz

- Appropriate properties for ‘CardNum’?

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Quiz

- Appropriate properties for ‘CardNum’?

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

- `not null`

Creating Tables

- Let’s create this table (without the contents)

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Creating Tables

```
create table Titles (
  ISBN char(14) not null,
  Title varchar(255) not null,
  Author varchar(255) not null,
  primary key (ISBN)
);
```

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert

Exercise

- Command to create this table? (without the contents)

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427

```
create table <name> (
  <column1Name> <type> <properties>,
  <column2Name> <type> <properties>,
  <table properties>
);
```

Foreign Keys

```
FOREIGN KEY (<column>) REFERENCES
<table>(<table's key>)
ON DELETE <action>
ON UPDATE <action>
```

Foreign Keys

```
FOREIGN KEY (<column>) REFERENCES
<table>(<table’s key>)
ON DELETE <action>
ON UPDATE <action>
```

- `<action>` can be:
  - `RESTRICT` (default): disallow the change
  - `CASCADE`: also delete/update in child table
  - `SET NULL`: nullify key in child table
  - `SET DEFAULT`: set to column’s default value

Foreign Key Example

```
CREATE TABLE Phones (
  ...
  FOREIGN KEY (CardNum)
  REFERENCES Patrons (CardNum)
  ON DELETE CASCADE
)
```

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
4	888-8888
4	999-9999

Library

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Inventory

Serial	ISBN
1001	978-0590353427
1002	978-0590353427
1003	978-0679732242
1004	978-0394823379
1005	978-0394823379
1006	978-0062278791

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Phones

CardNum	Phone
1	555-5555
2	666-6666
3	777-7777
4	888-8888
4	999-9999

Titles

ISBN	Title	Author
978-0590353427	Harry Potter	Rowling
978-0679732242	The Sound and the Fury	Faulkner
978-0394823379	The Lorax	Seuss
978-0062278791	Profiles in Courage	Kennedy
978-0441172719	Dune	Herbert