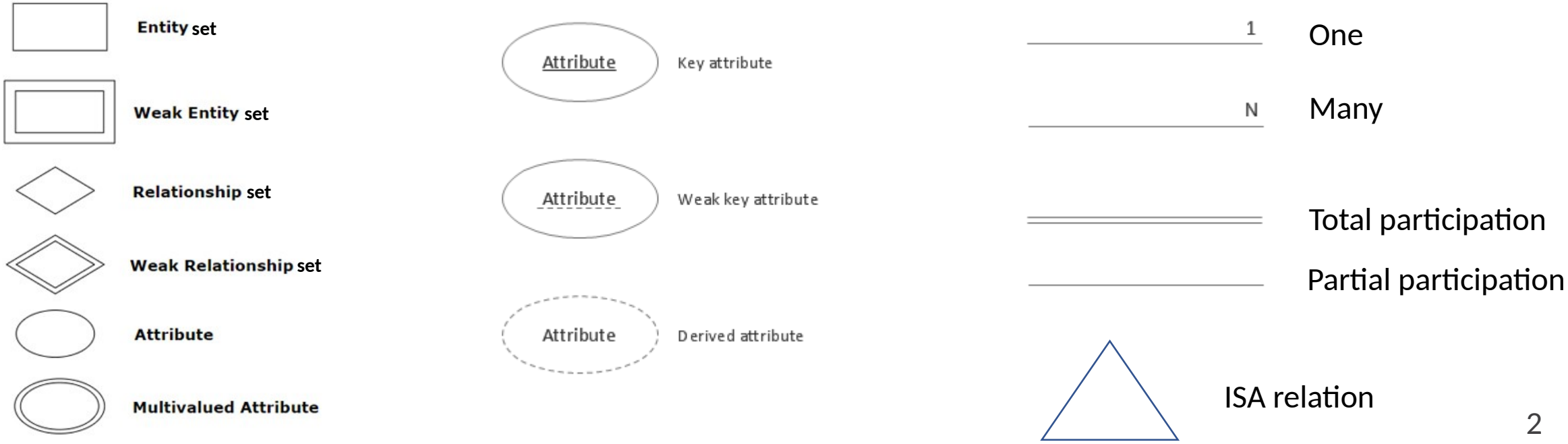
SCC.221
Data Engineering

2024 - Week 2 – Relational Database.

Uraz C Turker

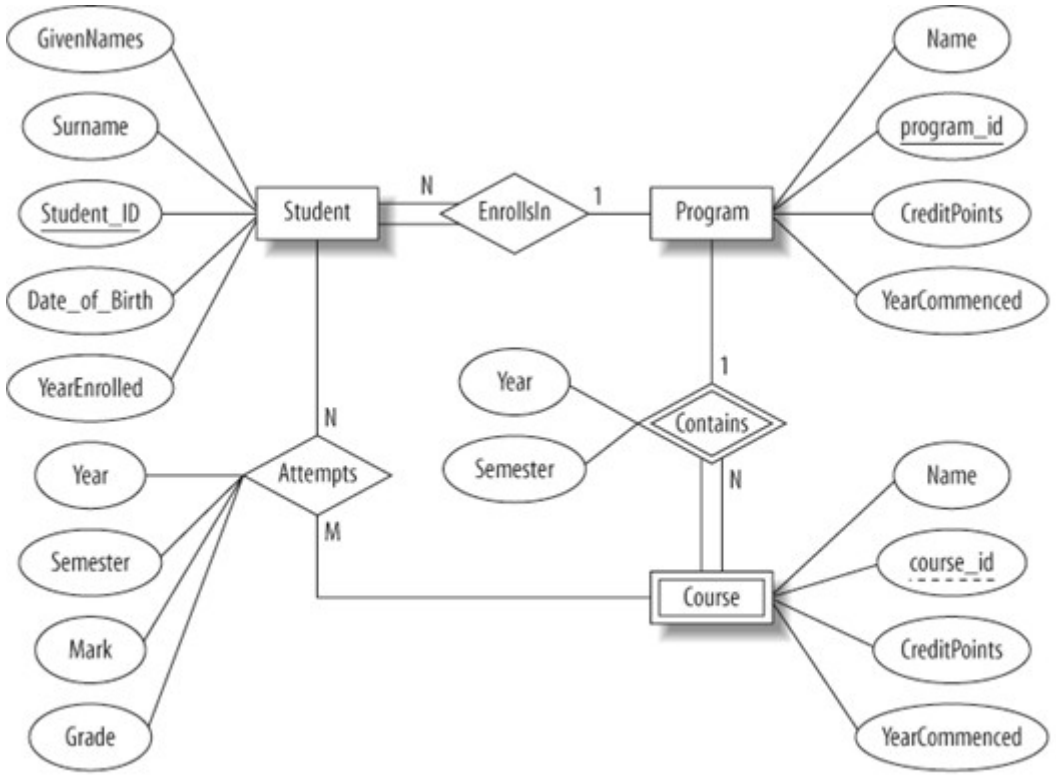
What have we learnt so far?

- We learned key ER concepts which allows us to design relational databases.



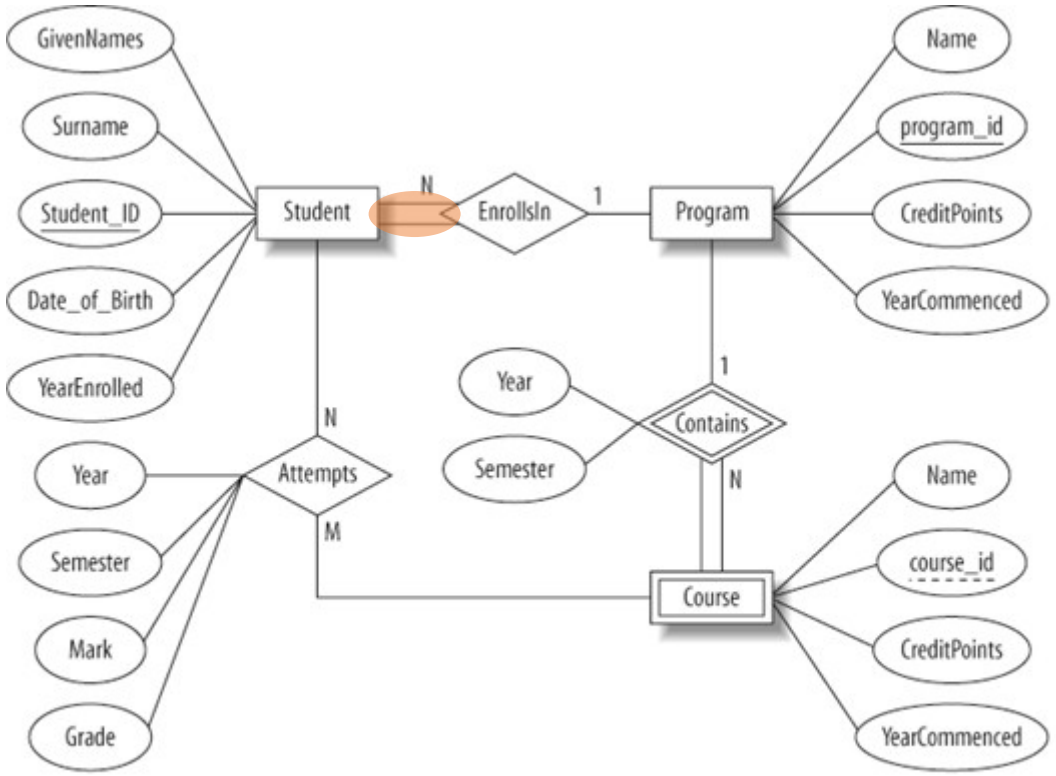
What have we learnt so far?

- A student enrol in program.



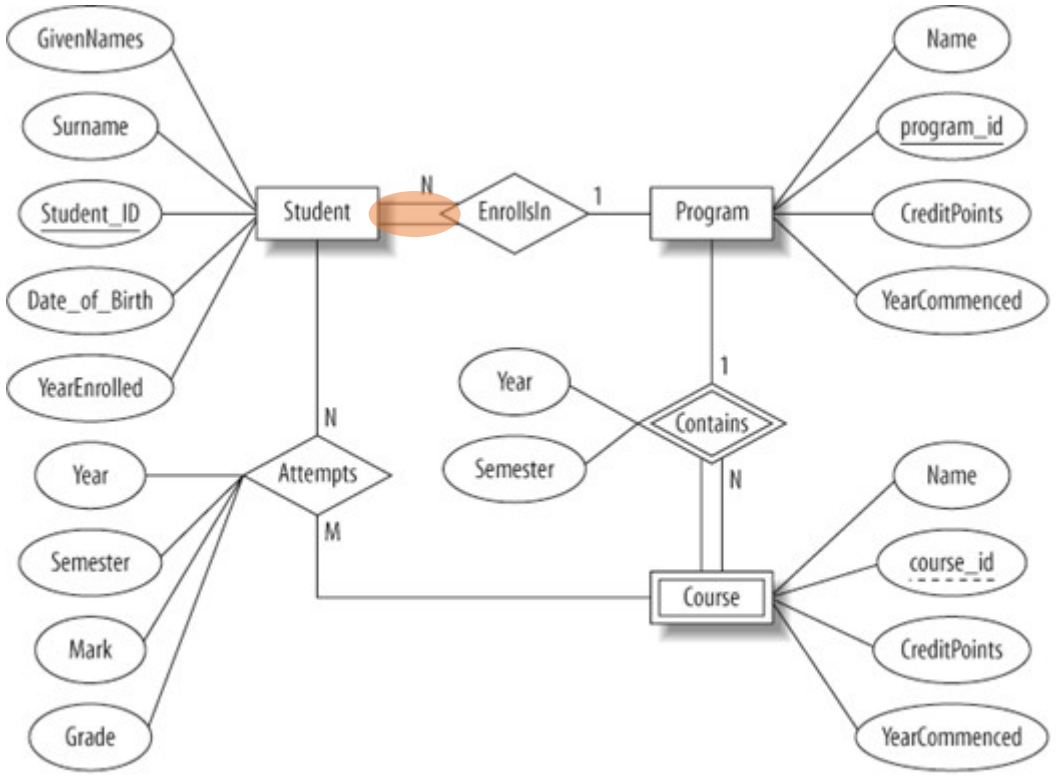
What have we learnt so far?

- A student enrol in program.



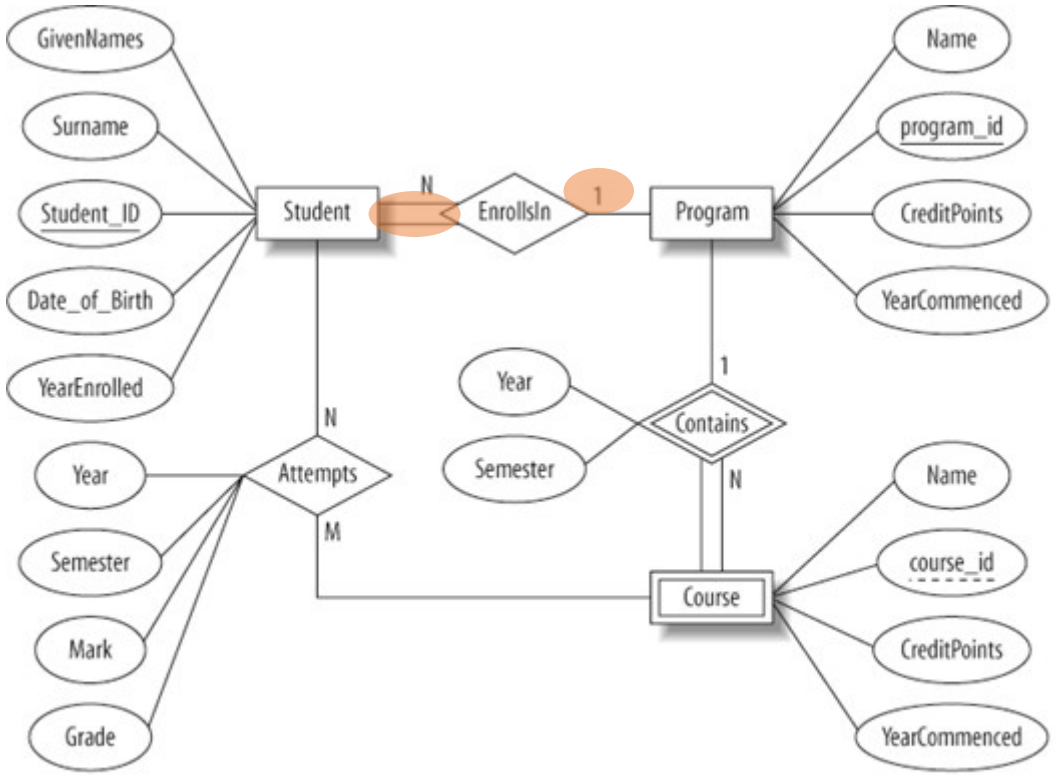
What have we learnt so far?

- A student **must** enrol in program.



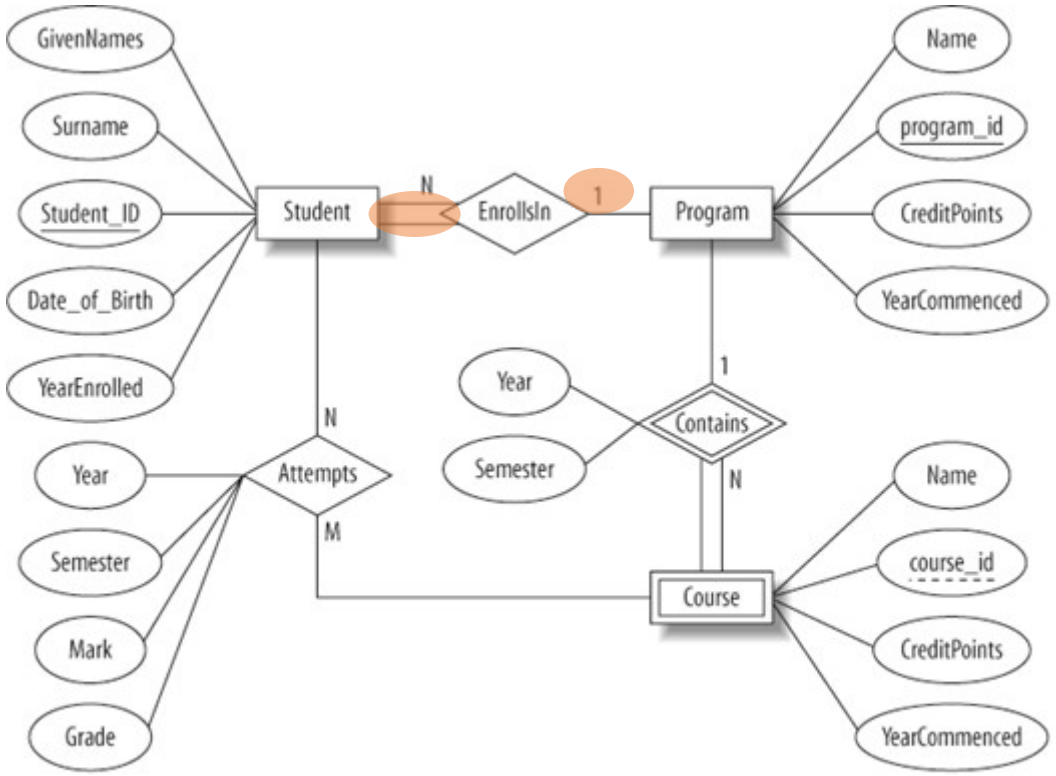
What have we learnt so far?

- A student **must** enrol in program.



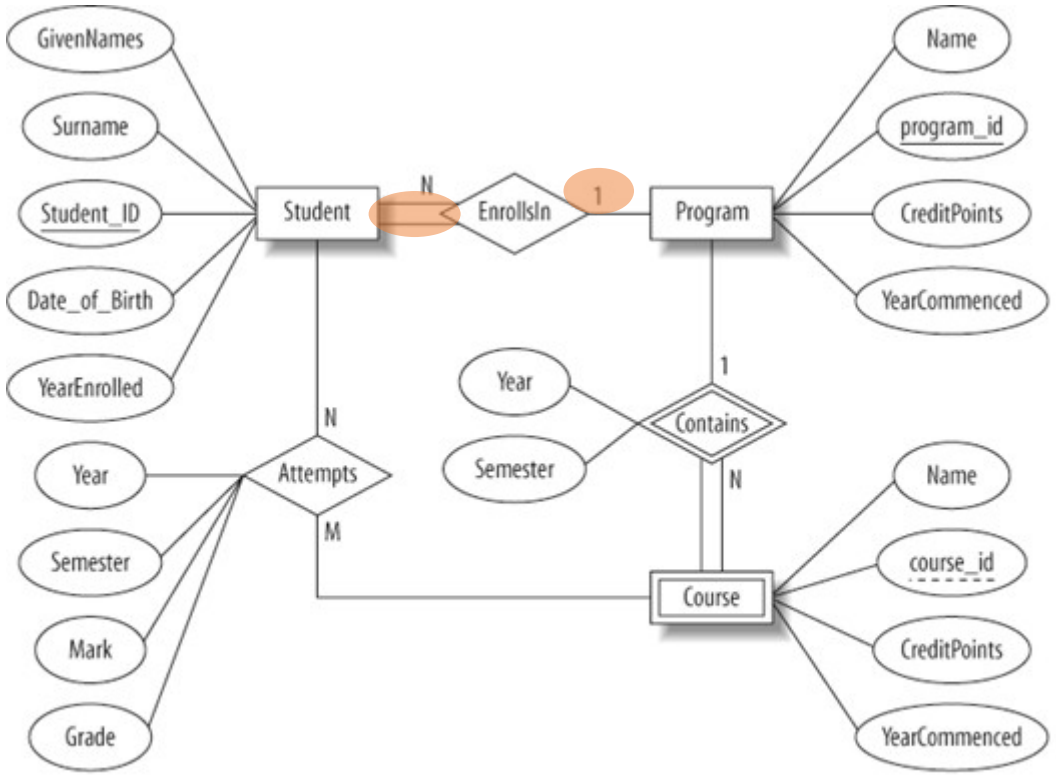
What have we learnt so far?

- A student **must** enrol in **one** program.



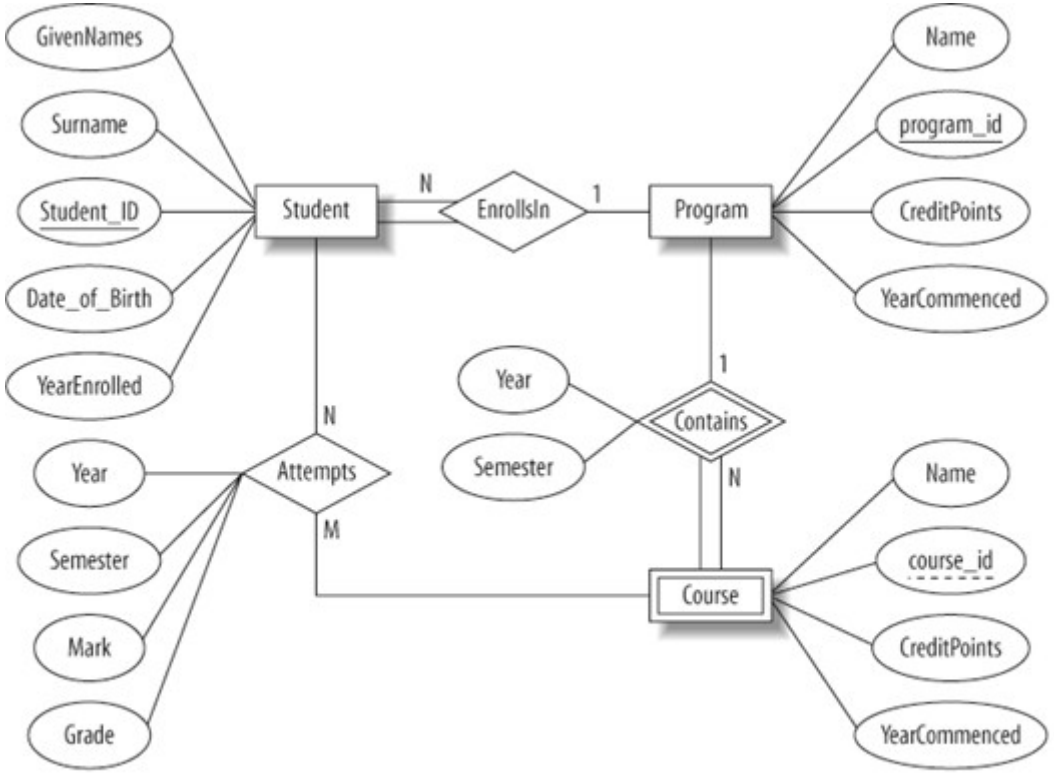
What have we learnt so far?

- A student **must** enrol in **one** program.
  - It has GivenNames, Surname, StudentID, Date_of_Birth, YearEnrolled attributes, where StudentID is a primary key.



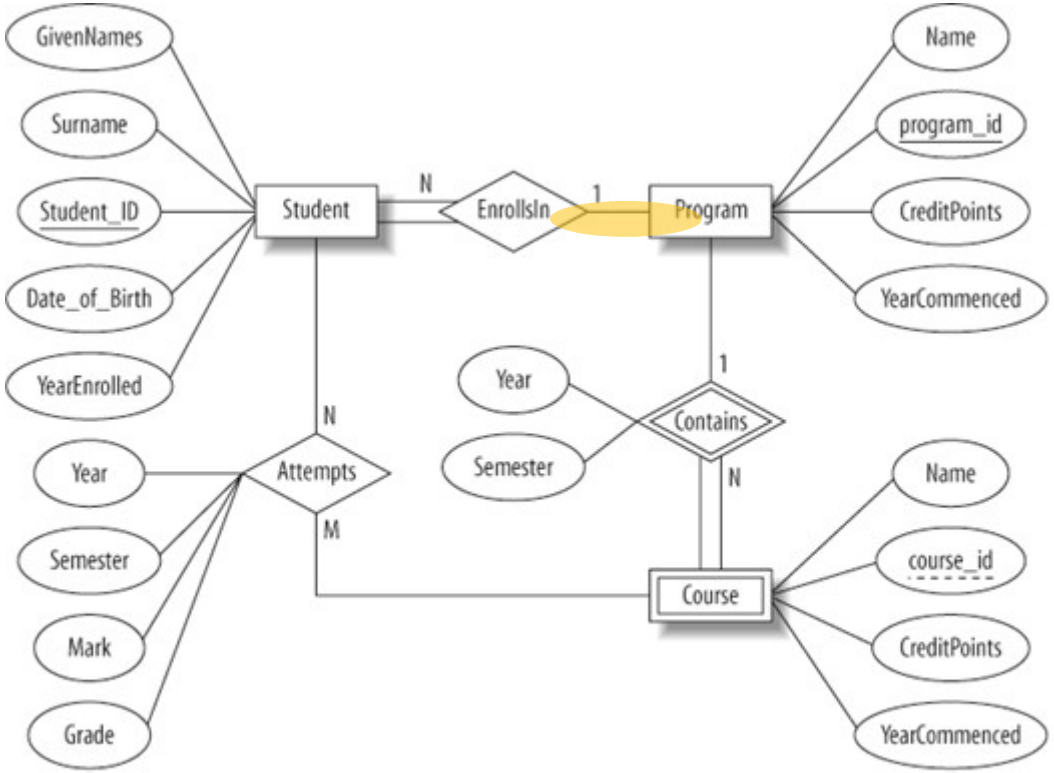
What have we learnt so far?

- Program have students.



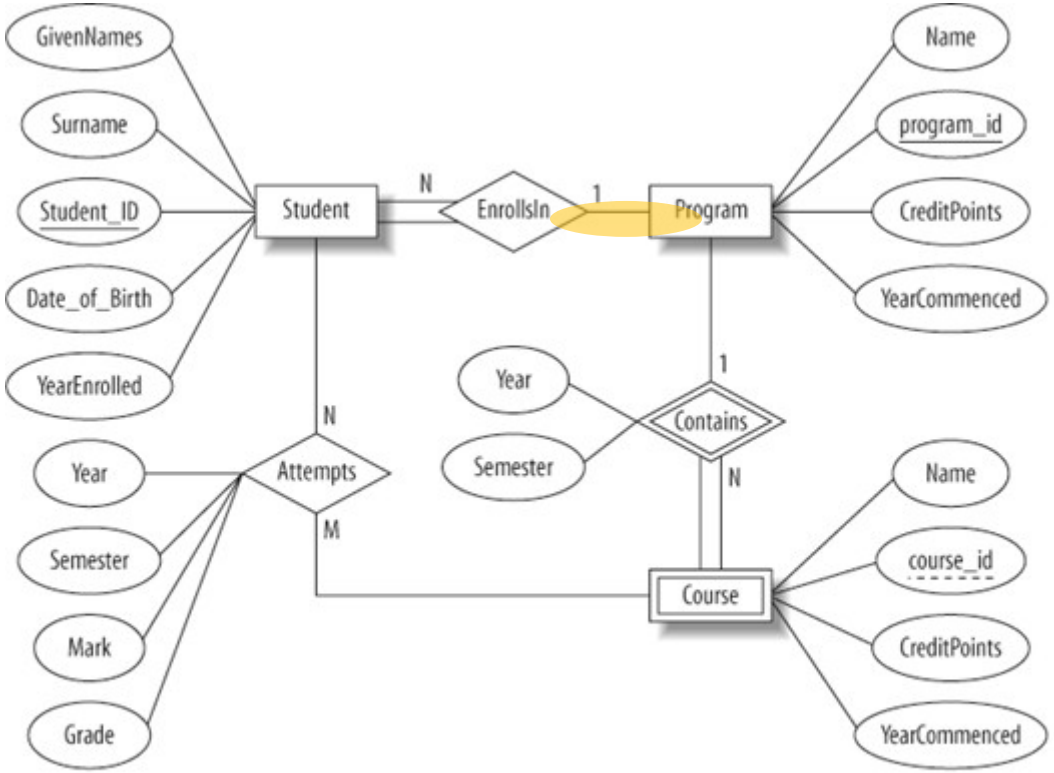
What have we learnt so far?

- Program have students.



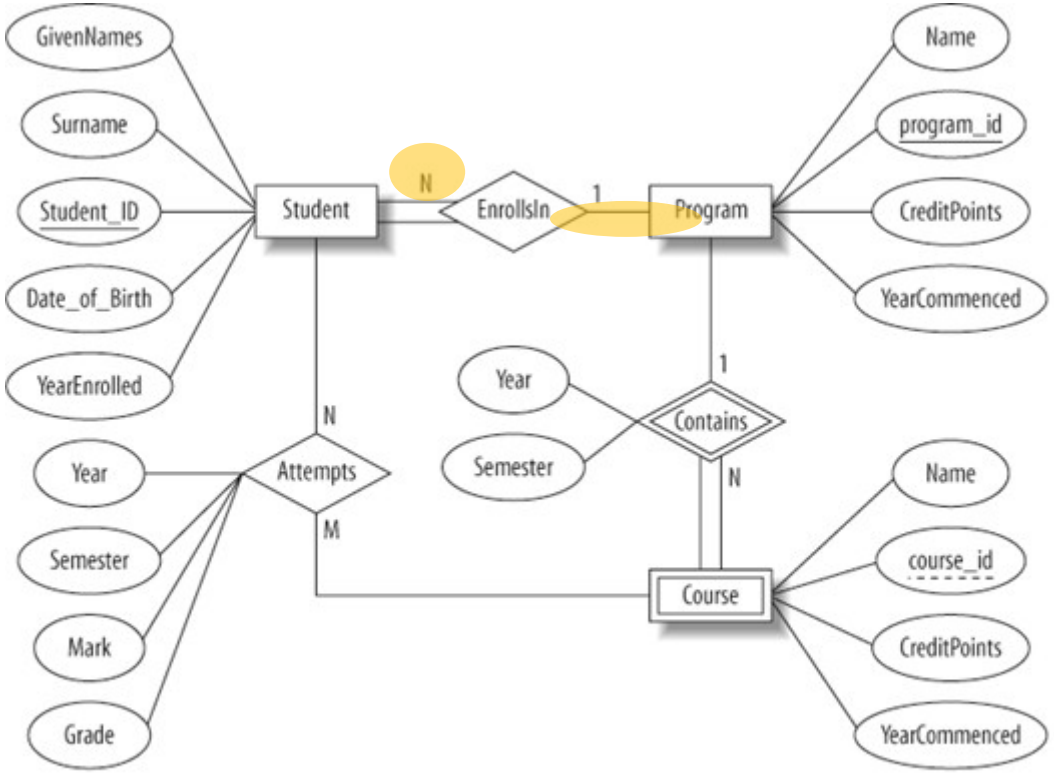
What have we learnt so far?

- Program **may** have students.



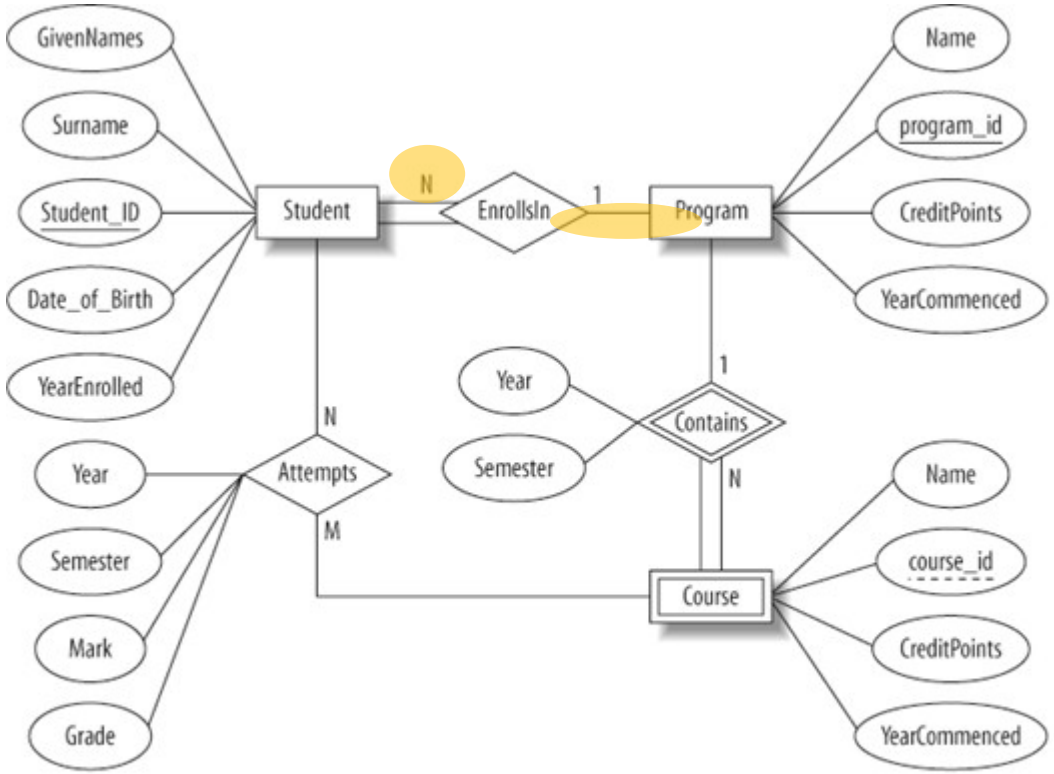
What have we learnt so far?

- Program **may** have students.



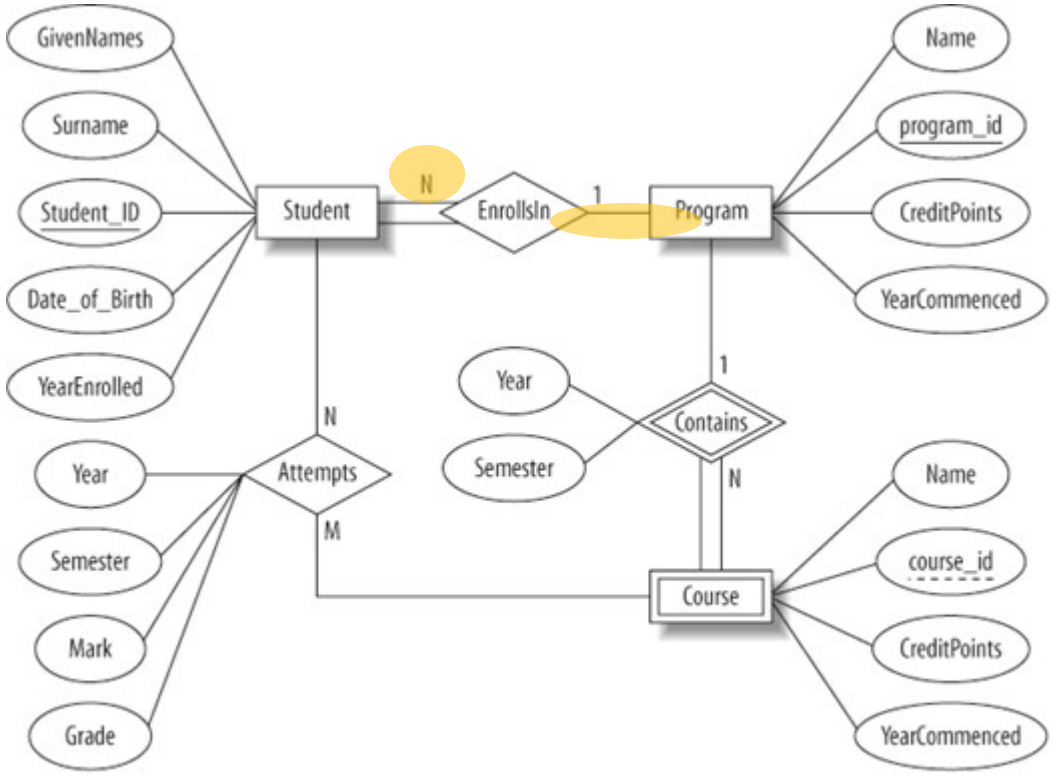
What have we learnt so far?

- Program **may** have **many** students.



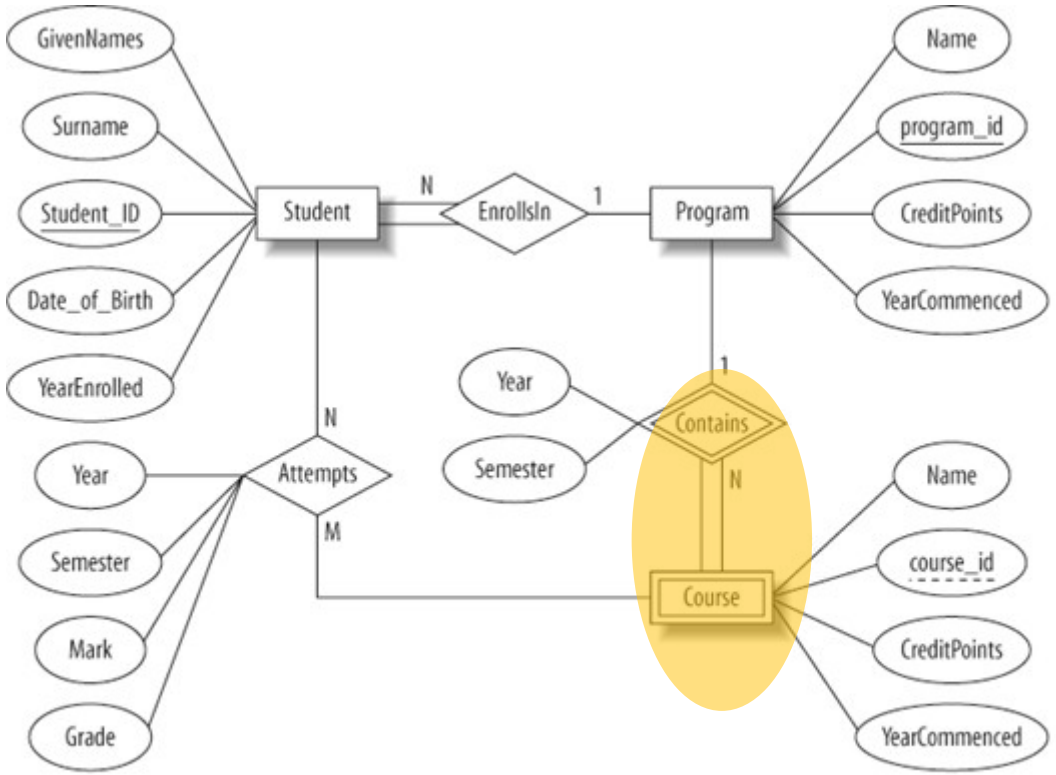
What have we learnt so far?

- Program **may** have **many** students.
  - It has name, program_iD, CreditPoints, YearCommenced where program_iD is a primary key.



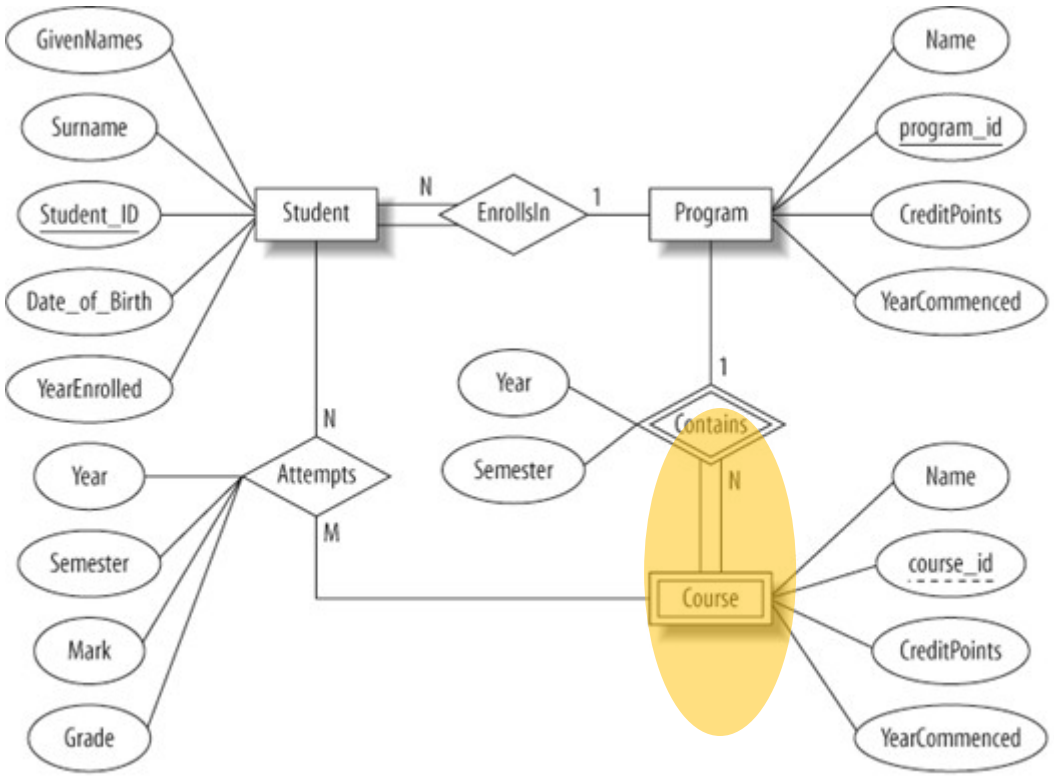
What have we learnt so far?

- When a program is deleted, the DBMS automatically deletes all related courses. (weak entity)



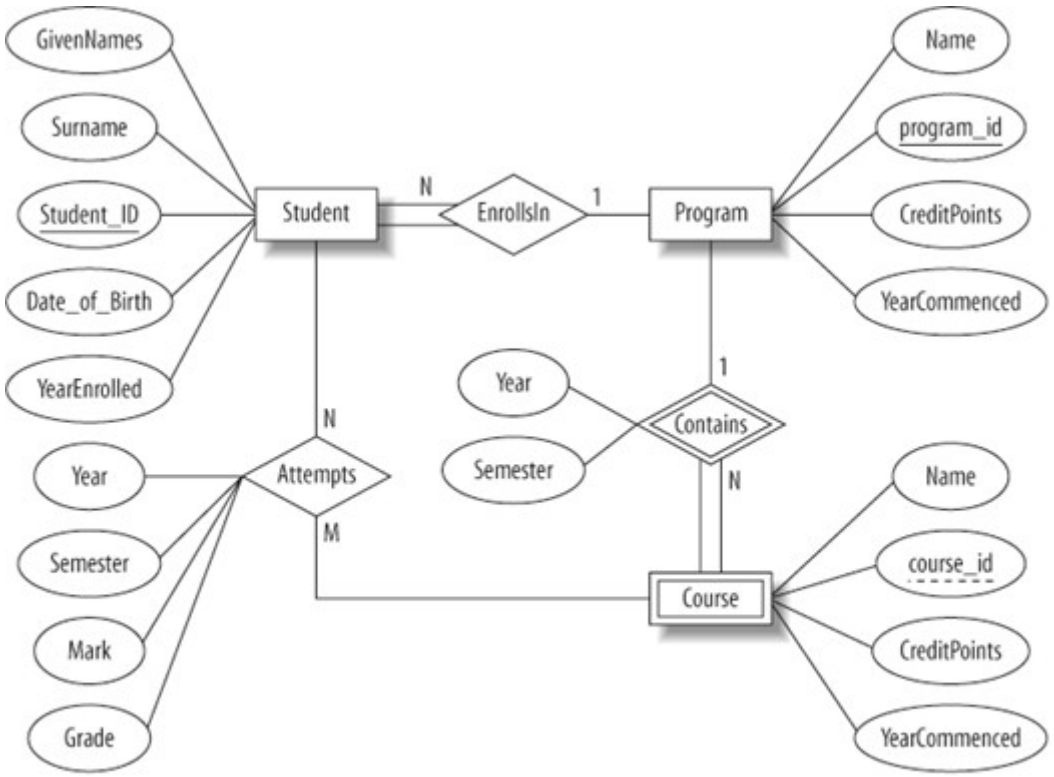
What have we learnt so far?

- When a program is deleted, the DBMS automatically deletes all related courses. (weak entity)
  - It has Name, course_id, CreditPoints, YearCommenced where course_id is the weak key.



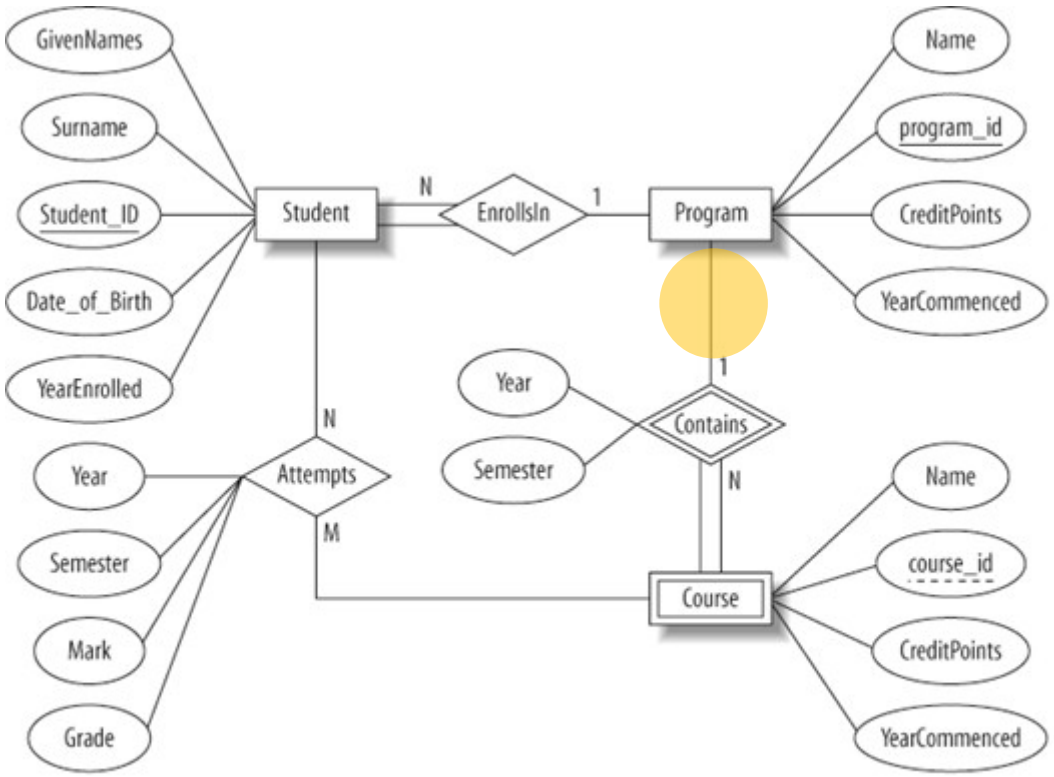
What have we learnt so far?

- Program have courses



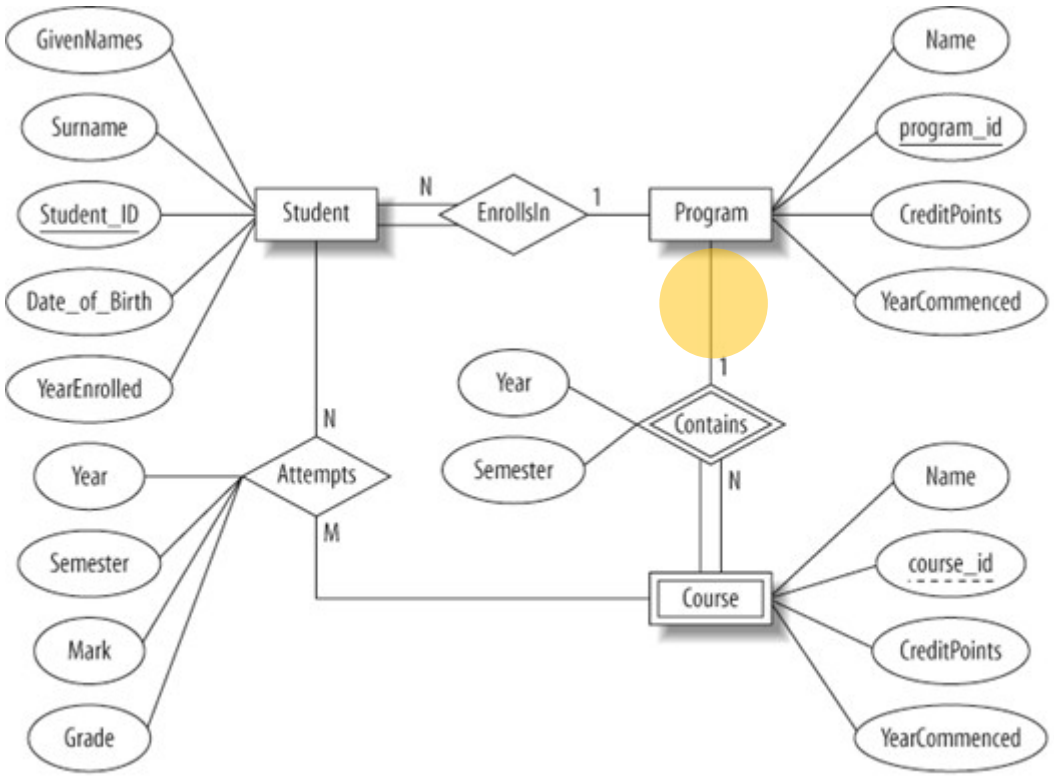
What have we learnt so far?

- Program have courses



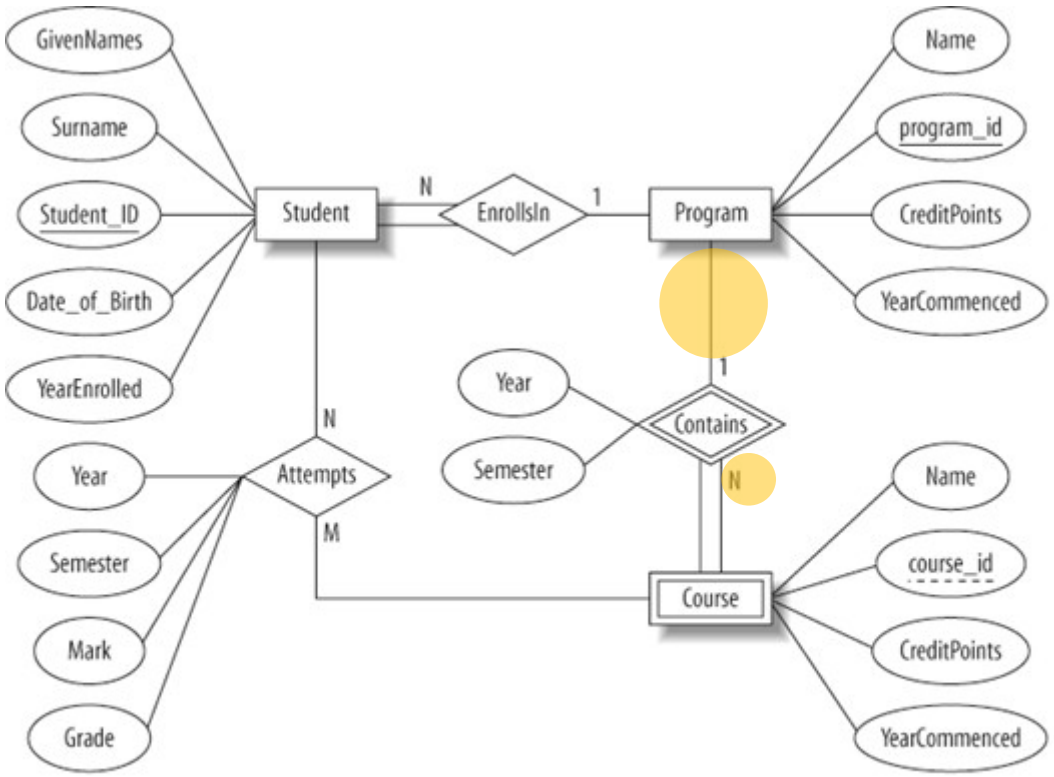
What have we learnt so far?

- Program **may** have courses



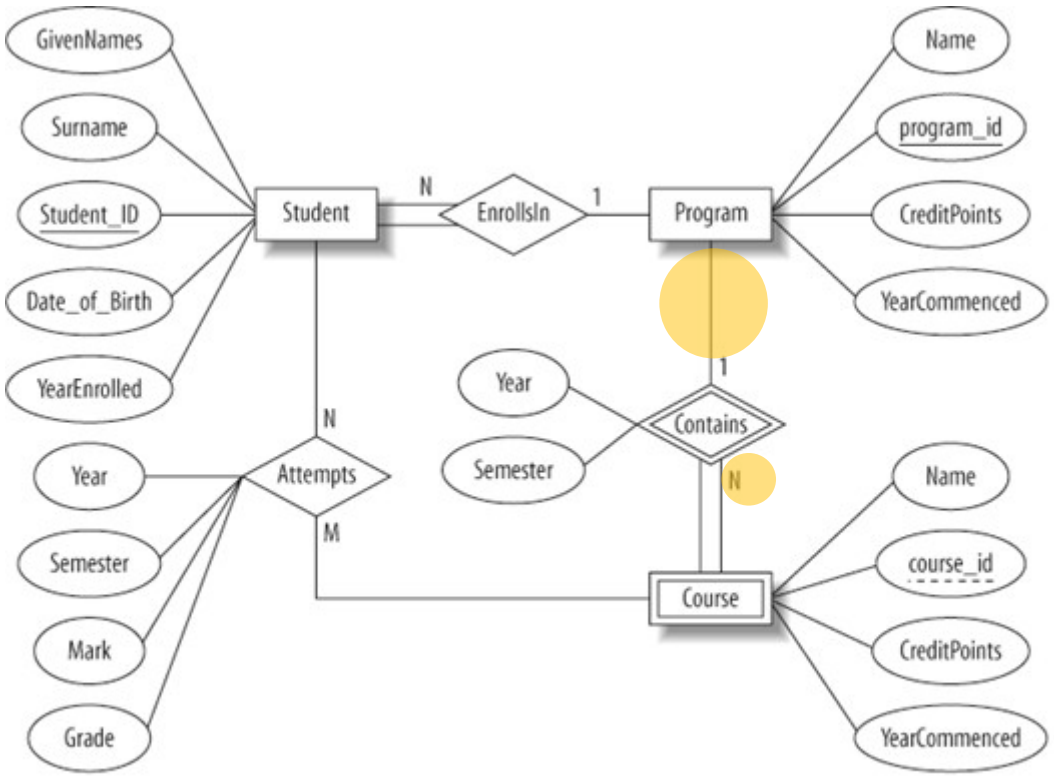
What have we learnt so far?

- Program **may** have courses



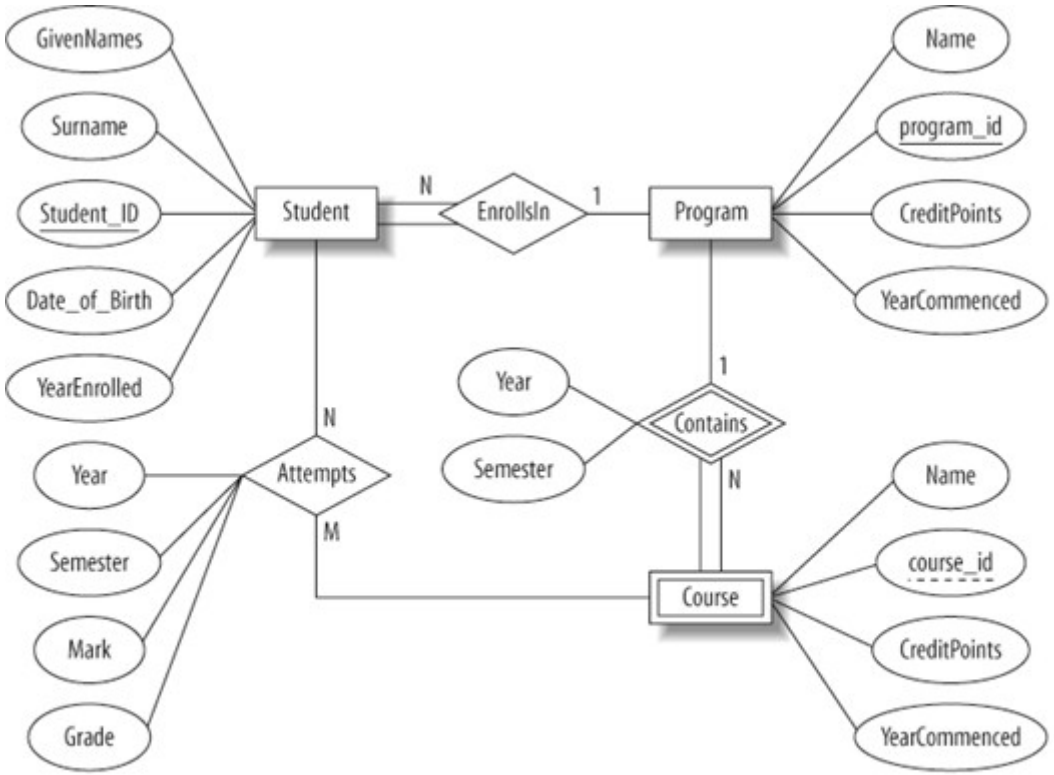
What have we learnt so far?

- Program **may** have **many** courses



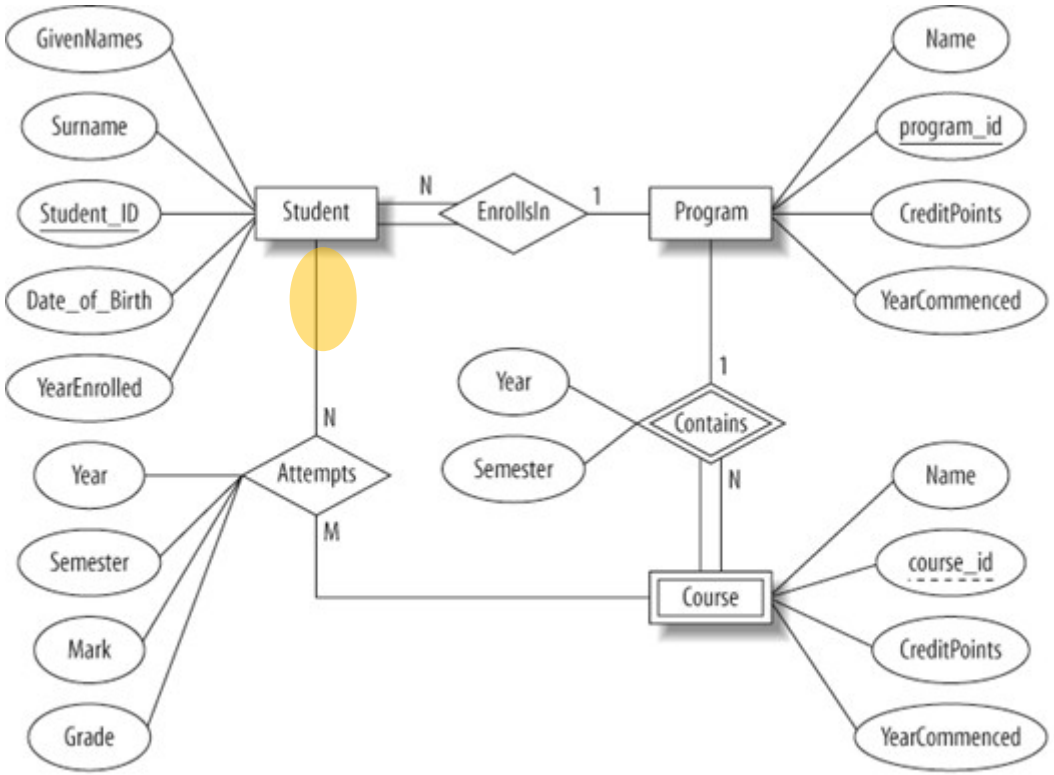
What have we learnt so far?

- A student **… attempt …** courses.
- A course **.. be attempted by ….** students.



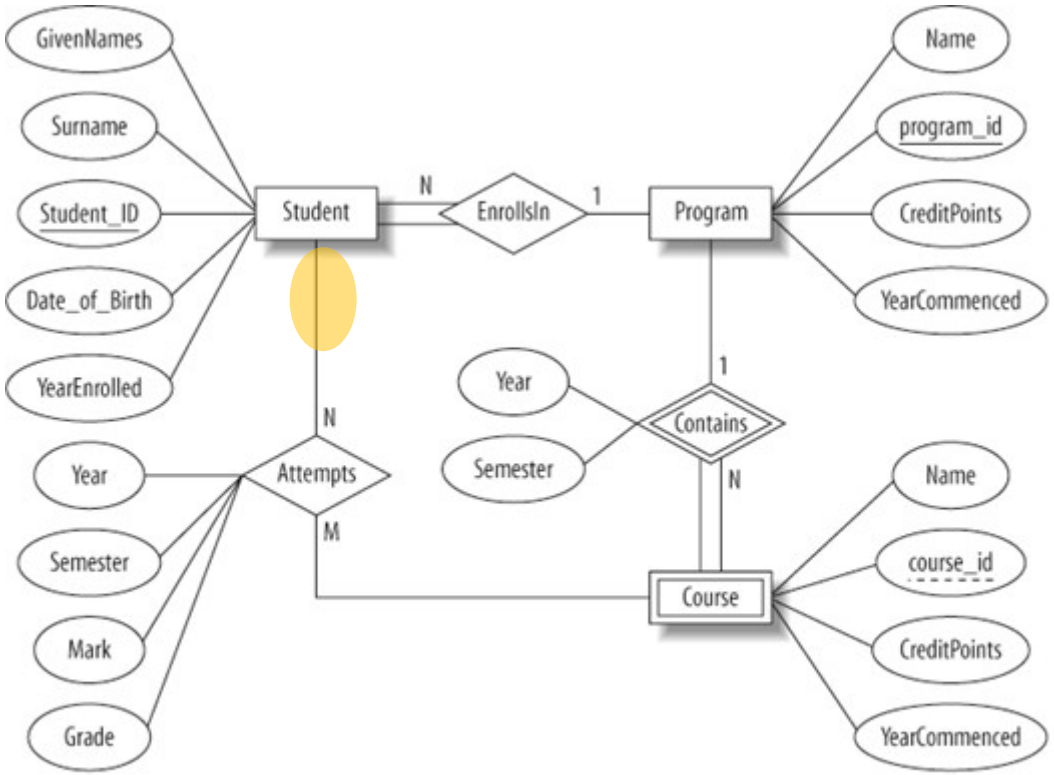
What have we learnt so far?

- A student **… attempt …** courses.
- A course **.. be attempted by ….** students.



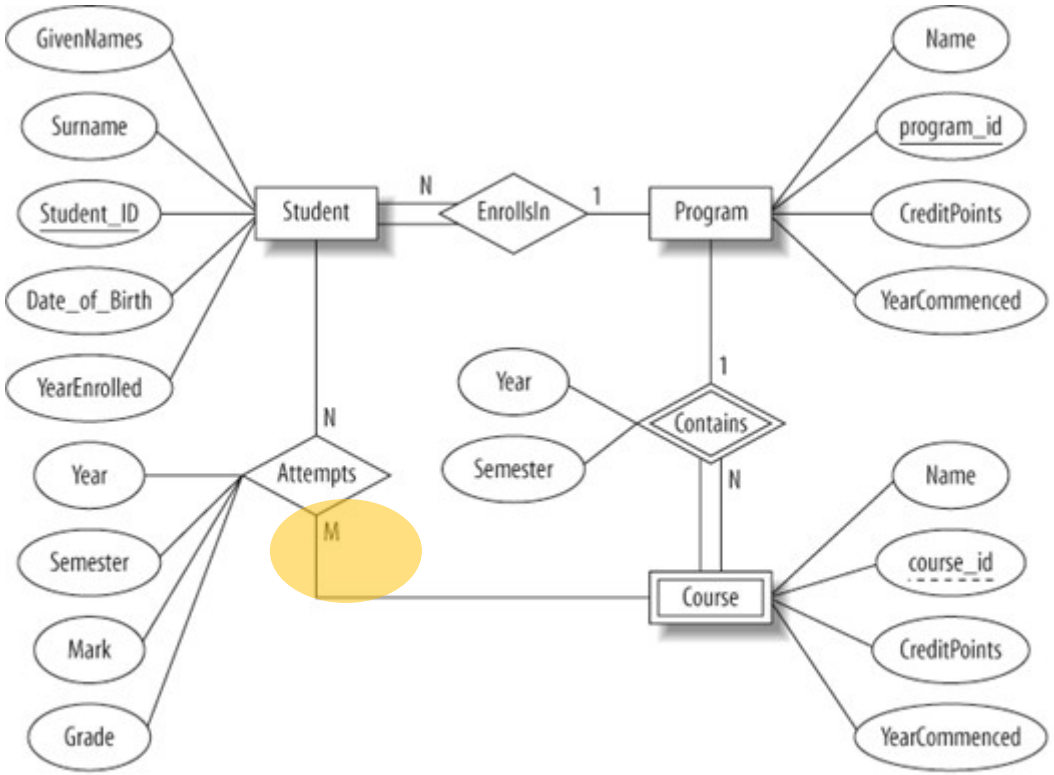
What have we learnt so far?

- A student **may attempt …** courses.
- A course **.. be attempted by ….** students.



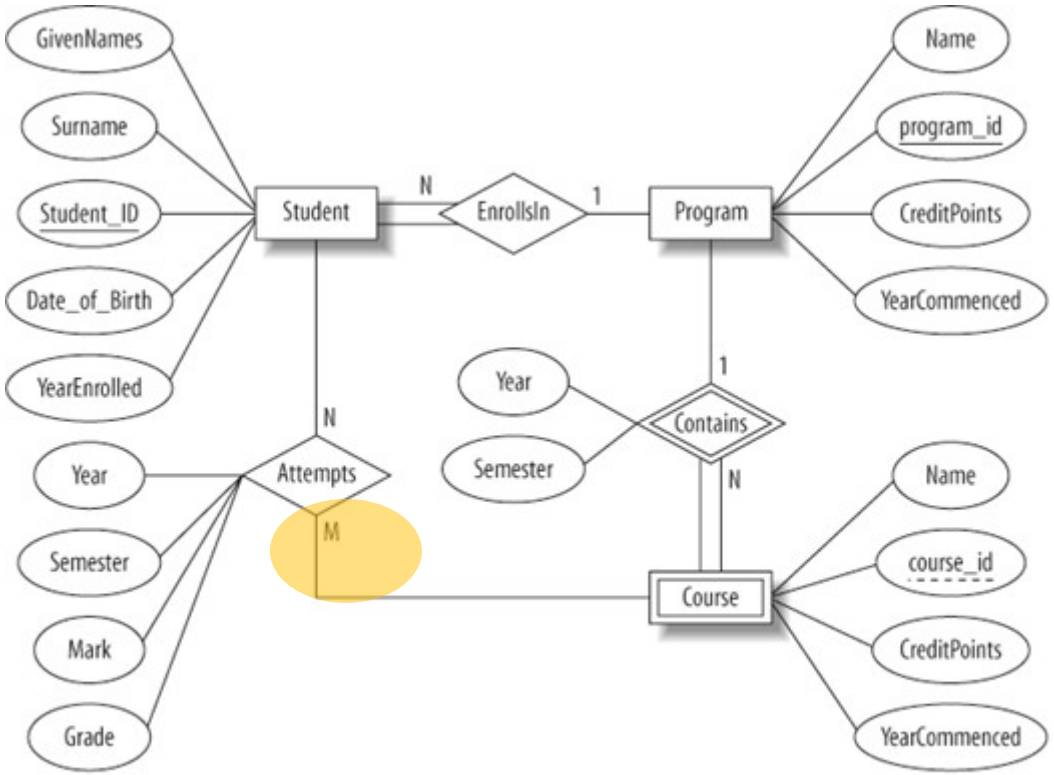
What have we learnt so far?

- A student **may attempt …** courses.
- A course **… be attempted by …** students.



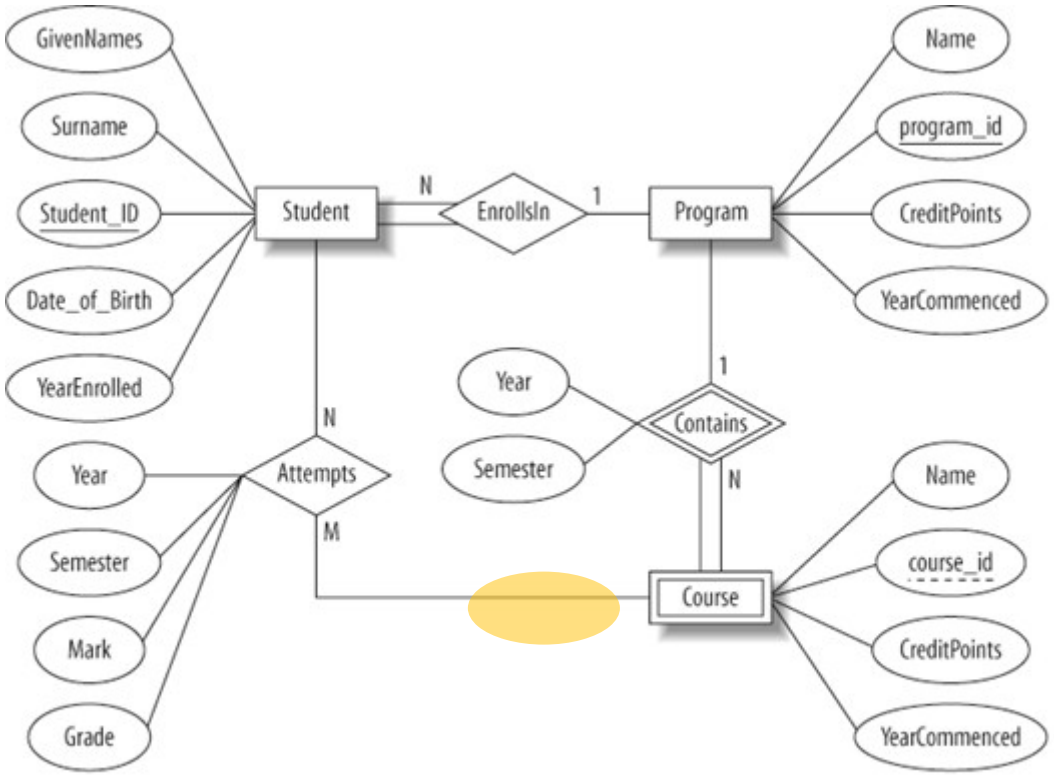
What have we learnt so far?

- A student **may attempt many** courses.
- A course **… be attempted by …** students.



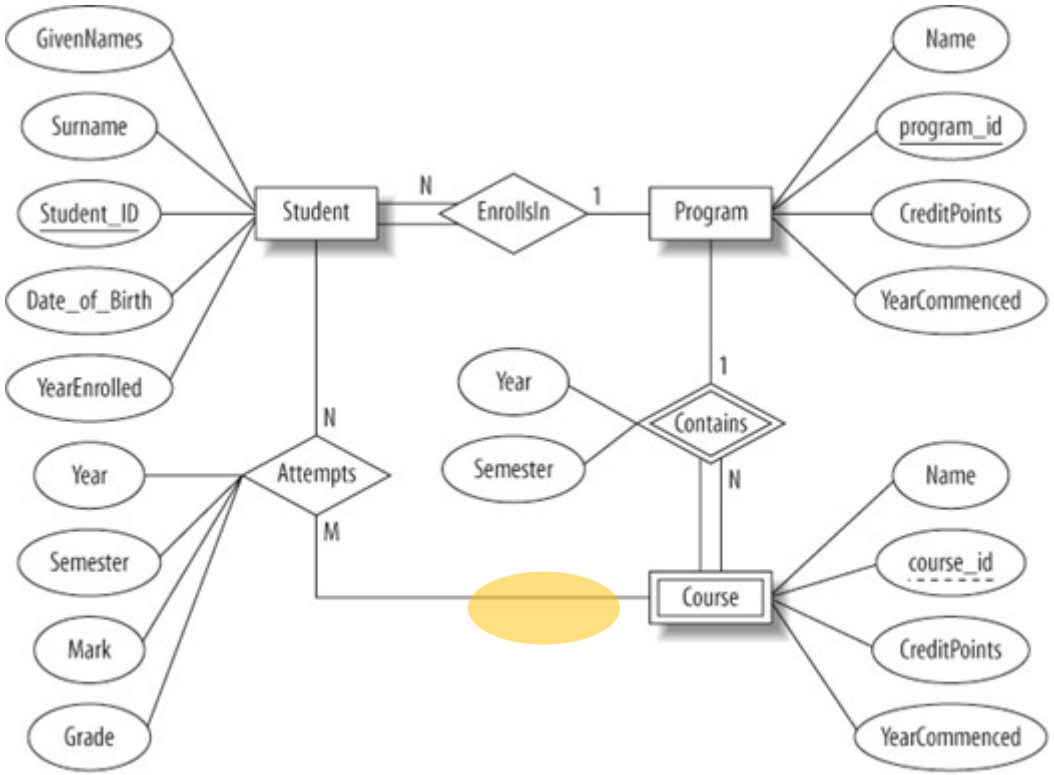
What have we learnt so far?

- A student **may attempt many** courses.
- A course **…. be attempted by ….** students.



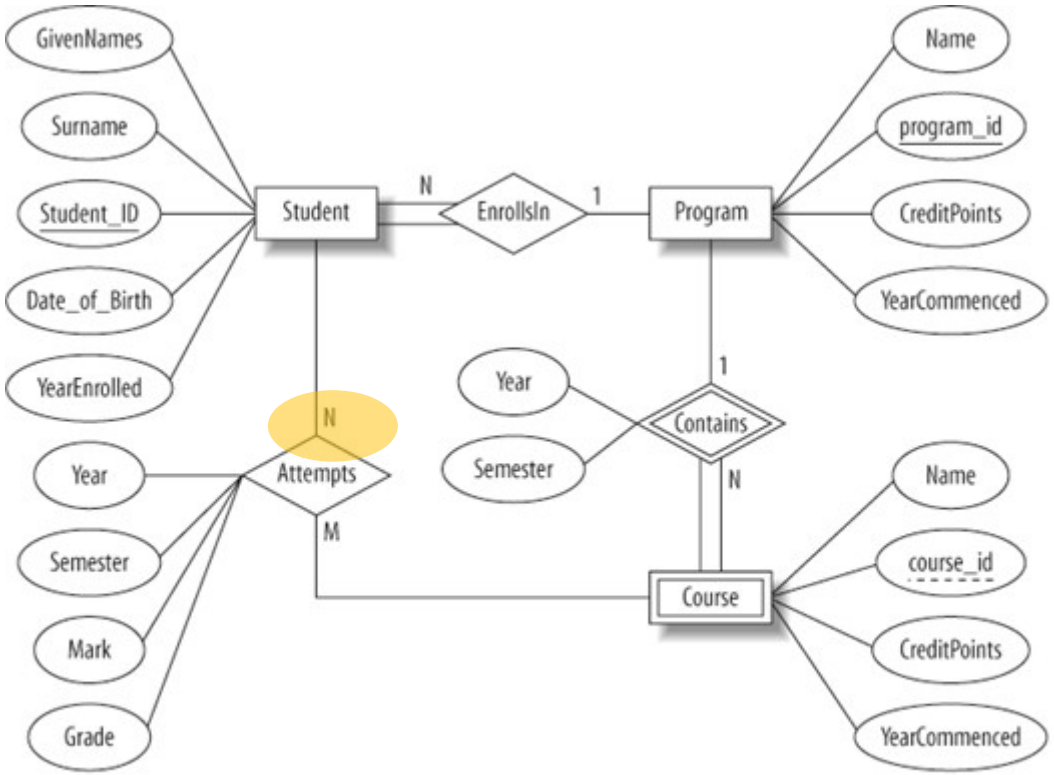
What have we learnt so far?

- A student **may attempt many** courses.
- A course **may be attempted by ….** students.



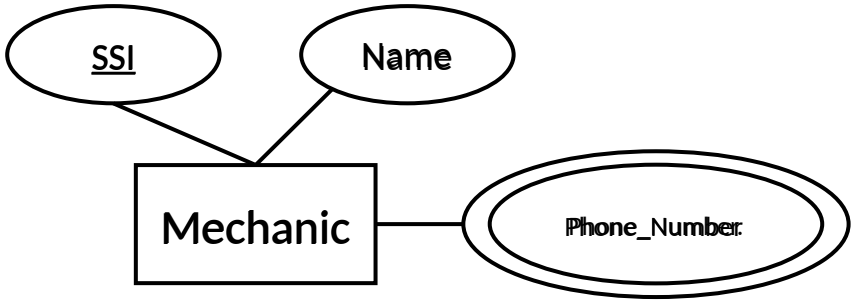
What have we learnt so far?

- A student **may attempt many** courses.
- A course **may be attempted by many** students.



Let us see the correspondence between ERD and Relational database.

- ER diagram



- Relation.

SSI	Name	Phone_Number
87542702	Tom	75315567, 75315264
68201937	Uraz	75335521, 75334567
23139827	Nick	75315544, 75315237

Relational Database: Definitions

- Instead of Entity Sets and Entities
- We use **relation** as a *set* of rows or *tuples* (i.e., all rows are distinct) with description.

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2
53650	Shero	shero@math	19	3.8

Rows/

Relational Database: Definitions

- *Relation is a mathematical statement*
- *Relation:* made up of 2 parts:

Instance: a *table* with rows and columns.

#Rows = *cardinality*, #fields/attributes = *degree / arity.*

Relation Schema (RS): specifies the relation's name and **type (domain)** of each **column**.

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2
53650	Shero	shero@math	19	3.8

<u>Write the Relation Schema for the given Relation Instance.</u>

Students(*sid*: INT, *name*: TEXT, *login*: TEXT, *age*: INT, *gpa*: DOUBLE).

Name of the attribute: Domain of the attribute

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

https://quake.fandom.com/wiki/Weapon_(Q1)

Relational Database: Definitions

Weapon	Damage	Ammo Type	Maximum Ammo	Fire Mode	Rate of Fire (rpm)	Muzzle Velocity (m/s)	Maximum Range (m)
Axe	20	None	∞	Single	120	-	-
Shotgun	24	Shells	100	Pump-action	120	380	78
Super Shotgun	56	Shells	100	Pump-action	85	380	78
Nailgun	9	Nails	200	Automatic	600	110	229
Super Nailgun	18	Nails	200	Automatic	600	158	229
Grenade Launcher	120	Rockets	100	Semi-auto	100	22.8	-

- *Relation:* made up of 2 parts:

  Instance: a *table* with rows and columns.

  #Rows = *cardinality*, #fields/attributes = *degree / arity*.

  Relation Schema (RS): specifies the relation's name and **type (domain)** of each **column**.

  Write the Relation Schema for the given Relation Instance.

Weapons(*Weapon*: TEXT, *Damage*: INT, *AmmoType*: TEXT,
MaximumAmmo: INT, *FireMode*: TEXT,
RateofFire: INT, *MuzzleVelocity*:DOUBLE, *MaximumRange*: INT).

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Relational Database: Definitions

- *Relation:* made up of 2 parts:

  Instance: a *table* with rows and columns.

Team_ID	Wins	Loses	Name	Leader
223	1	43	Les Miserables	Uraz
14	16	0	Thanos	Thanos
1	23	3	Avengers	Iron Man

#Rows = *cardinality*, #fields/attributes = *degree / arity.*

Relation Schema (RS): specifies the relation's name and **type (domain)** of each **column**.

<u>Write the Relation Schema for the given Relation Instance.</u>

Teams(*Team_ID*: INT, *Wins*: INT, *Loses*: INT, *Name*: TEXT, *Leader*: TEXT).

MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

Exercise…

Relational Database: Definitions

- *Relation:* made up of 2 parts:

  Instance: a *table* with rows and columns.

Team_ID	Wins	Loses	Name	Leader
223	1	43	Les Miserables	Uraz
14	16	0	Thanos	Thanos
1	23	3	Avengers	Iron Man

#Rows = *cardinality*, #fields/attributes = *degree / arity.*

Relation Schema (RS): specifies the relation's name and **type (domain)** of each **column**.

Write the Relation Schema for the given Relation Instance.

Teams(*Team_ID*: INT, *Wins*: INT, *Loses*: INT, *Name*: TEXT, *Leader*: TEXT).

What would happen if I made a mistake and entered the following row of data?

<Uraz, 34, 55, A poor lecturer, Uraz>

Relational model (Syntax vs Semantics)

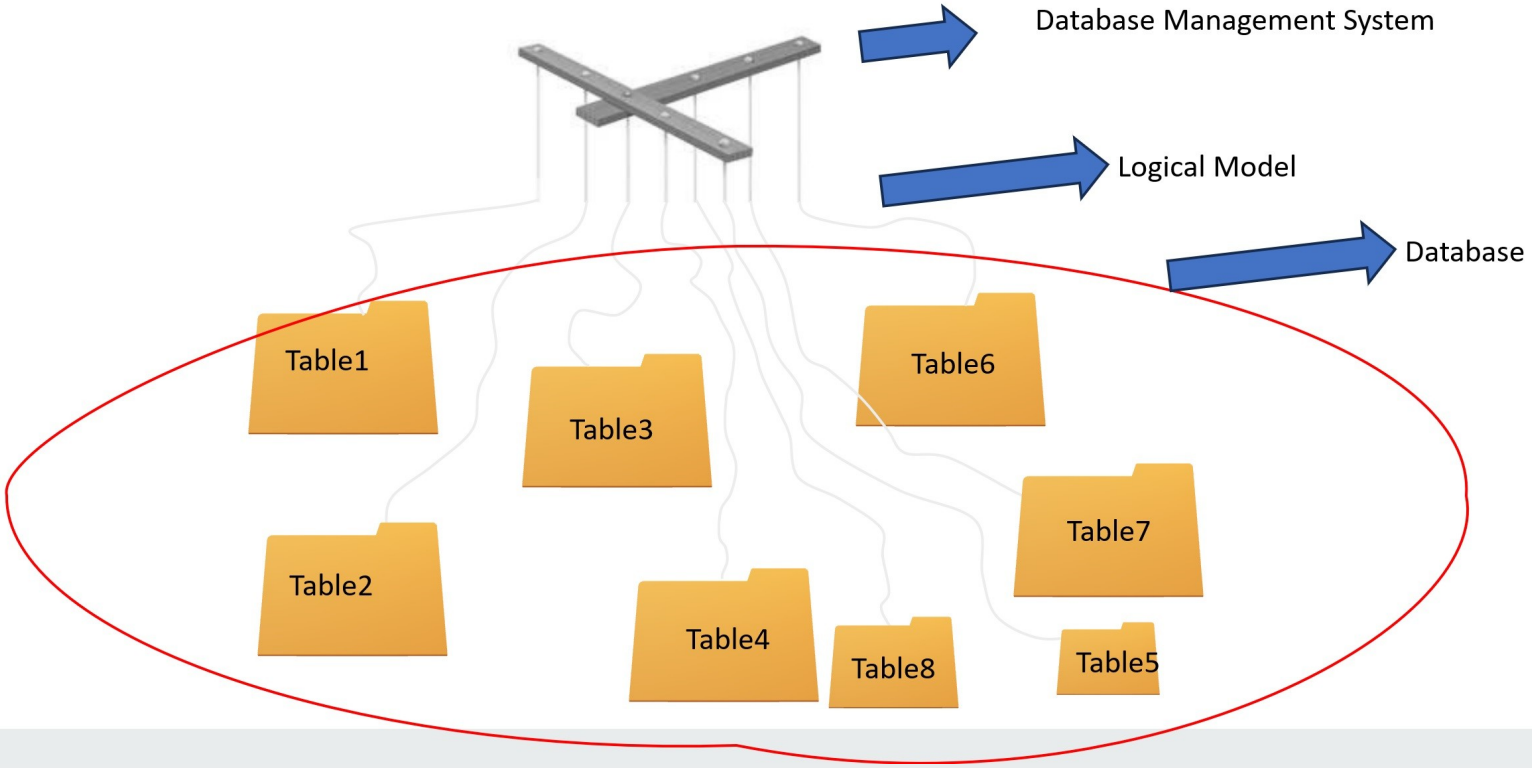
Model	Weight	ChassisN	Max_Speed
	1400	12h37	200
Toyota_Corolla	1300	84t34	200
Hyundai E.GLS	1400	43j5h2	210

What else can go wrong?

- What would happen if I erase a key value?
- Data kept by a DBMS **must obey some rules**!
- To prevent these, relational databases are built upon logical rules (constraints) set by
  - Real-world rules (Context), designers' choices, functions, etc., using DDL.
- These rules establish what we call the **integrity** of the data
- Integrity of the data is protected by DBMS if INTEGRITY CONSTRAINTS ARE GIVEN.

Relational Database: Definitions

Integrity Constraints are a part of the Logical Model and are provided to the DBMS <u>while tables are created</u>.



Definitions : Integrity Constraints (ICs)

- Integrity constraint (IC): a condition that must be true for *any* database instance.
- A *legal* instance of a relation satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!
- 1) Domain constraint: In any database instance, a value of the attribute must be an element of the attribute's domain (*gpa* is a DOUBLE-valued attribute, so we can only store DOUBLE values in related cells) as set in RS.

Students(*sid*: INT, *name*: TEXT, *login*: TEXT, *age*: INT, *gpa*: DOUBLE).

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Shero	shero@cs	18	3.2
53650	Shero	shero@math	19	3.8
"1177"	Uraz	u.turker@cs	39	[illegible]

Domain constraint breached (gpa is a DOUBLE value, Na is a TEXT)

Definitions : Integrity Constraints (ICs)

Map	Vehicle	Occupants	Type
	Buggy	2	Land
Miramar	PG-117	5	Water
Vikendi	C-130	100	AIR

2) Entity Integrity constraint: A key value cannot be duplicated or left empty in any instance.

- Once a DB Admin sets a key, the *DBMS must inspect every modification* on the data w.r.t the key value.
- DB Admin's task (using DDL) is to set the primary key for the data.
  - DBMS has built-in functions to protect key constraints; to activate those functions, DBA must identify them using DDL (week 4!).
- Write the Relational Schema for this table

PubG(*Map*: TEXT, *Vehicle*: Text, *Occupants*: INT, *Type*: Text, *PRIMARY KEY*: Map).

Relational Database: Definitions

- *Relation:* made up of 2 parts:

  Instance: a *table* with rows and columns.

Weight	cost	owner	explosive	Type
1kg	441	X82jxm	1	1
2kg	123	Fr29x9a	0	3
1kg	223	7ndj2qs	0	12

#Rows = *cardinality*, #fields/attributes = *degree / arity.*

Relation Schema (RS): specifies the relation's name and **type (domain)** of each **column**.

Write the Relation Schema for the given Relation Instance.

Ammunition(*Weight*: TEXT, *cost*: INT, *owner*: TEXT, *explosive*: BOOLEAN, *Type*: INT, *PRIMARY KEY*: owner).

Relational Database: Definitions

- *Relation:* made up of 2 parts:

  Instance: a *table* with rows and columns.

Team_ID	Wins	Loses	Name	Leader
223	1	43	Les Miserables	Uraz
14	16	0	Thanos	Thanos
1	23	3	Avengers	Iron Man

#Rows = *cardinality*, #fields/attributes = *degree / arity.*

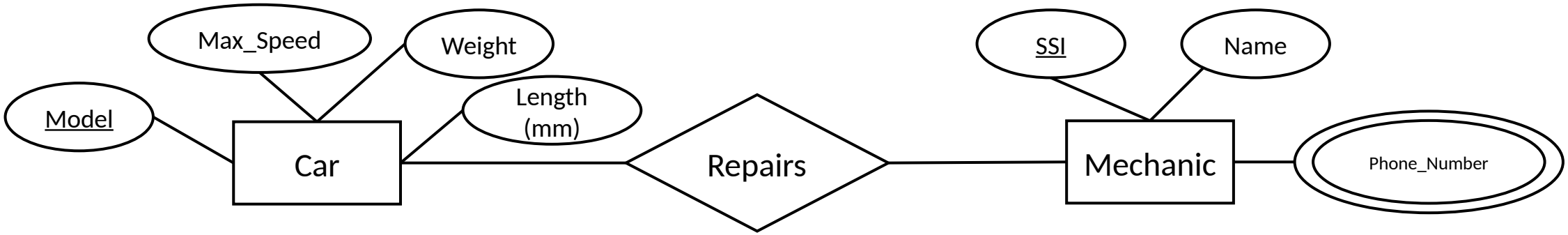
Relation Schema (RS): specifies the relation's name and **type (domain)** of each **column**.

Write the Relation Schema for the given Relation Instance.

Teams(*Team_ID*: INT, *Wins*: INT, *Loses*: INT, *Name*: TEXT, *Leader*: TEXT, *PRIMARY KEY:* Team_ID).

Referential integrity: How do we set Participation and Multiplicity constraints?

- How do we set the integrity of relationships?



Relationship sets

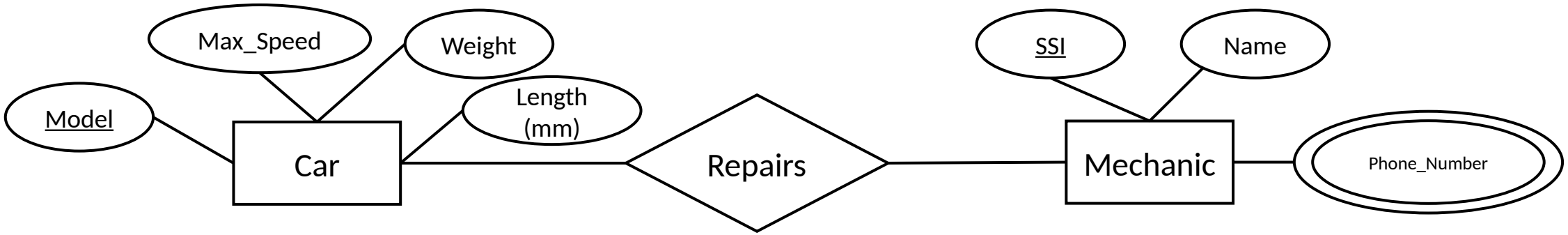
- We inform DBMS regarding the **referential integrity** for relationship sets while creating them using Foreign Keys (in DDL).

Referential integrity: Another key! Foreign keys

- **Foreign keys: a peripheral attribute that establishes referential integrity between entity sets.**

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

SSI	Name	Phone_Number
87542702	Tom	75315567, 75315264
68201937	Uraz	75335521, 75334567
23139827	Nick	75315544, 75315237

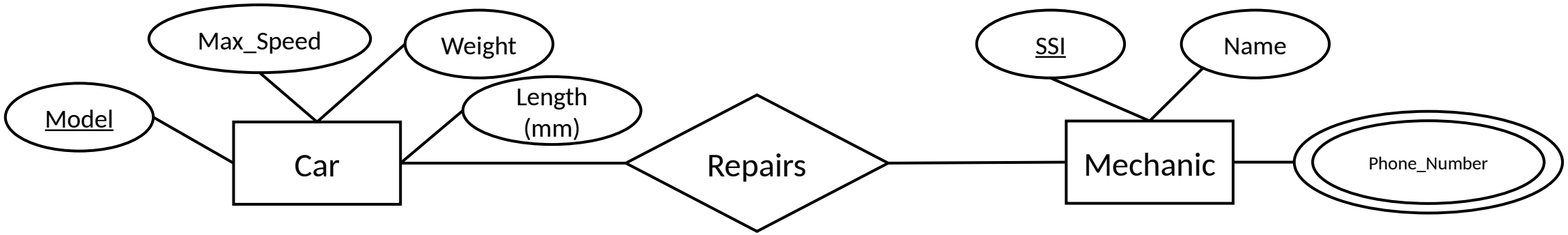


Referential integrity: Another key! Foreign keys

- **Foreign keys: a peripheral attribute that establishes referential integrity between entity sets.**
- It requires either i) **importing <u>the Primary Key</u> attribute of one table to the other table**

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

SSI	Name	Phone_Number
87542702	Tom	75315567, 75315264
68201937	Uraz	75335521, 75334567
23139827	Nick	75315544, 75315237

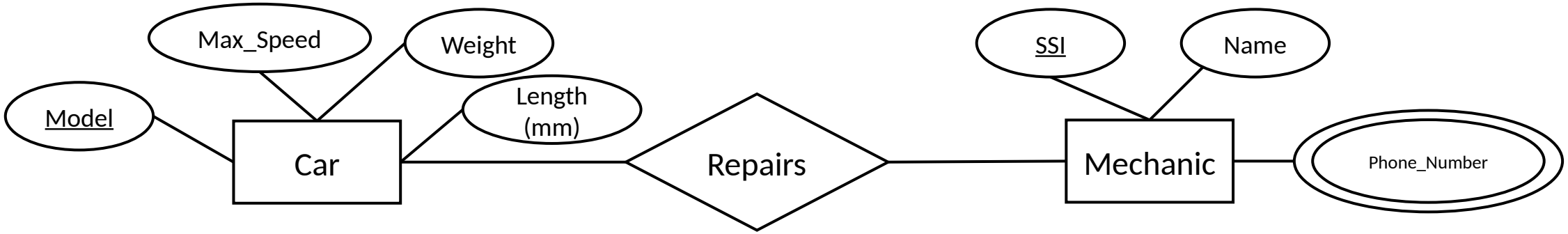


Referential integrity: Another key! Foreign keys

- **Foreign keys: a peripheral attribute that establishes referential integrity between entity sets.**
- It requires either i) **importing <u>the Primary Key</u> attribute of one table to the other table**

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

SSI	Name	Phone_Number
87542702	Tom	75315567, 75315264
68201937	Uraz	75335521, 75334567
23139827	Nick	75315544, 75315237

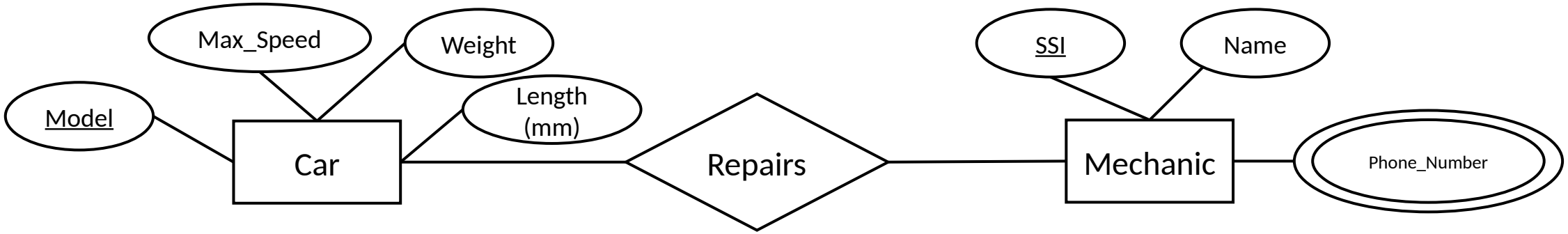


Referential integrity: Another key! Foreign keys

- **Foreign keys: a peripheral attribute that establishes referential integrity between entity sets.**
- It requires either i) **importing <u>the Primary Key</u> attribute of one table to the other table**

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

Model	SSI	Name	Phone_Number
BMW 3.21	87542702	Tom	75315567, 75315264
Toyota_Corolla	68201937	Uraz	75335521, 75334567
Hyundai E.GLS	23139827	Nick	75315544, 75315237

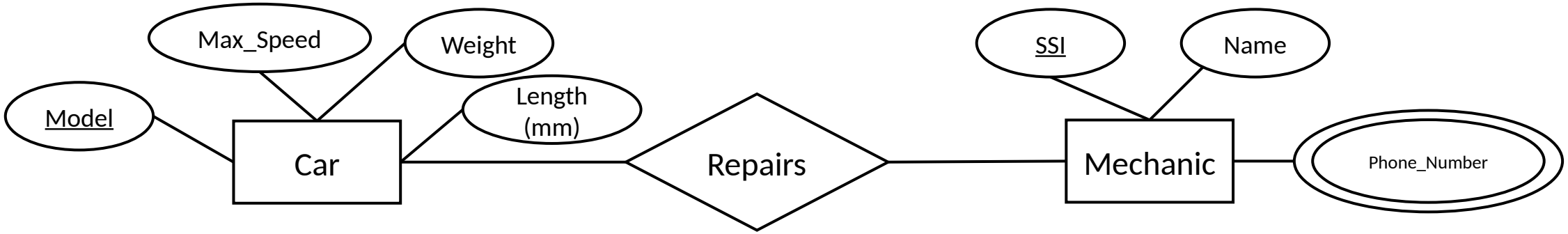


Referential integrity: Another key! Foreign keys

- **Values may be in different order!**

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

Model	SSI	Name	Phone_Number
Toyota_Corolla	87542702	Tom	75315567, 75315264
Hyundai E.GLS	68201937	Uraz	75335521, 75334567
BMW 3.21	23139827	Nick	75315544, 75315237



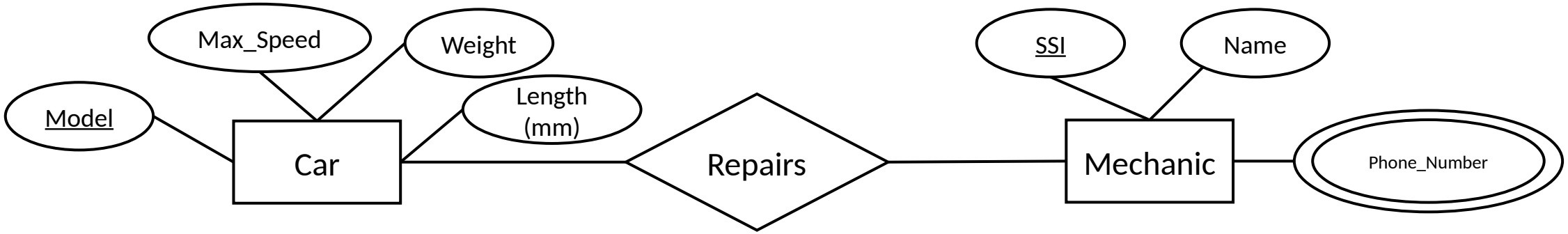
Referential integrity: Another key! Foreign keys

- **Foreign keys: a peripheral attribute that establishes referential integrity between entity sets.**
- It requires either i) **importing <u>the Primary Key</u> attribute of one table to the other table** or ii) **creating a new table that holds the primary keys of the tables in relation**.

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

Model	SIS
Toyota_Corolla	87542..
Hyundai E.GLS	68201..
BMW 3.21	2313..

SSI	Name	Phone_Number
87542702	Tom	75315567, 75315264
68201937	Uraz	75335521, 75334567
23139827	Nick	75315544, 75315237

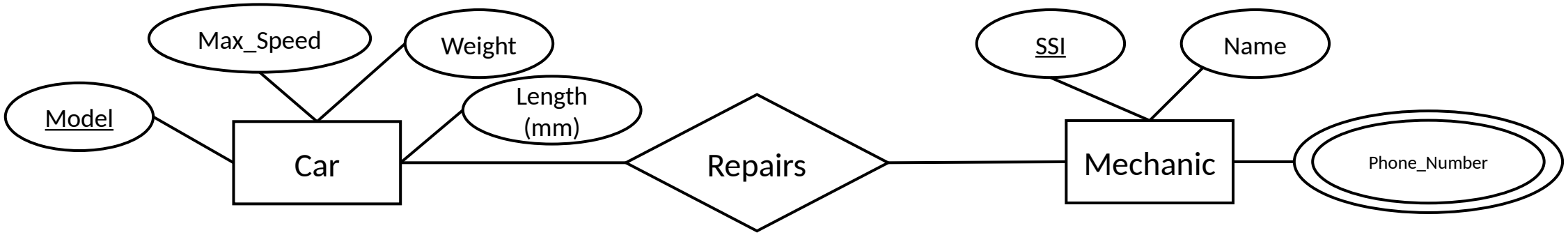


Referential integrity: Foreign keys

- ***Foreign key***: `logical **pointer**'.

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

Model	SSI	Name	Phone_Number
Toyota_Corolla	87542702	Tom	75315567, 75315264
Hyundai E.GLS	68201937	Uraz	75335521, 75334567
BMW 3.21	23139827	Nick	75315544, 75315237



Properties of foreign keys

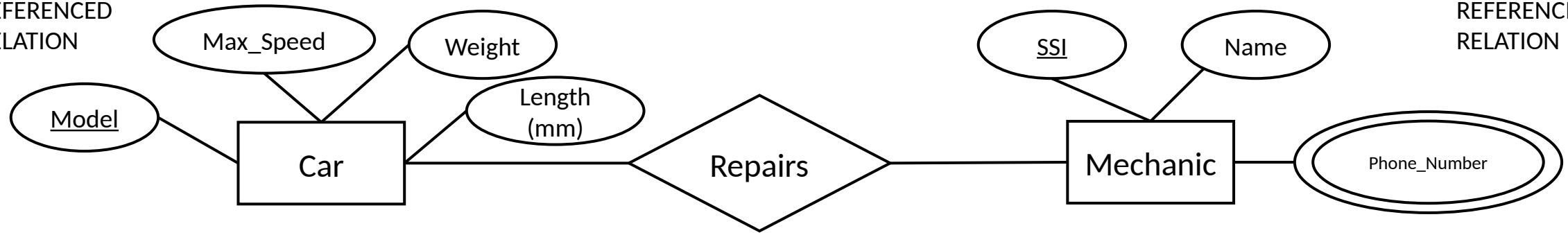
- Foreign key must:
  - Have the same name and domain/type as the referencing relation.
  - Related entities **must** have the same values.

Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

Model	SSI	Name	Phone_Number
Toyota_Corolla	87542702	Tom	75315567, 75315264
Hyundai E.GLS	68201937	Uraz	75335521, 75334567
BMW 3.21	23139827	Nick	75315544, 75315237

REFERENCED RELATION

REFERENCING RELATION



Properties of foreign keys

- Foreign key must:
  - Have the same name and domain/type as the referencing relation.
  - Related entities **must** have the same values.

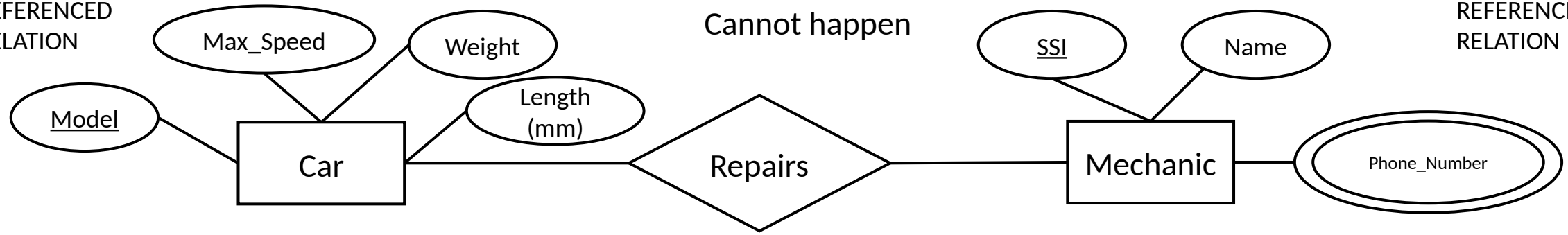
Model	Weight	Length (mm)	Max_Speed
BMW 3.21	1400	2501	200
Toyota_Corolla	1300	3321	200
Hyundai E.GLS	1400	3895	210

Model	SSI	Name	Phone_Number
Toyota	87542702	Tom	75315567, 75315264
Hyundai E.GLS	68201937	Uraz	75335521, 75334567
BMW 3.21	23139827	Nick	75315544, 75315237

REFERENCED RELATION

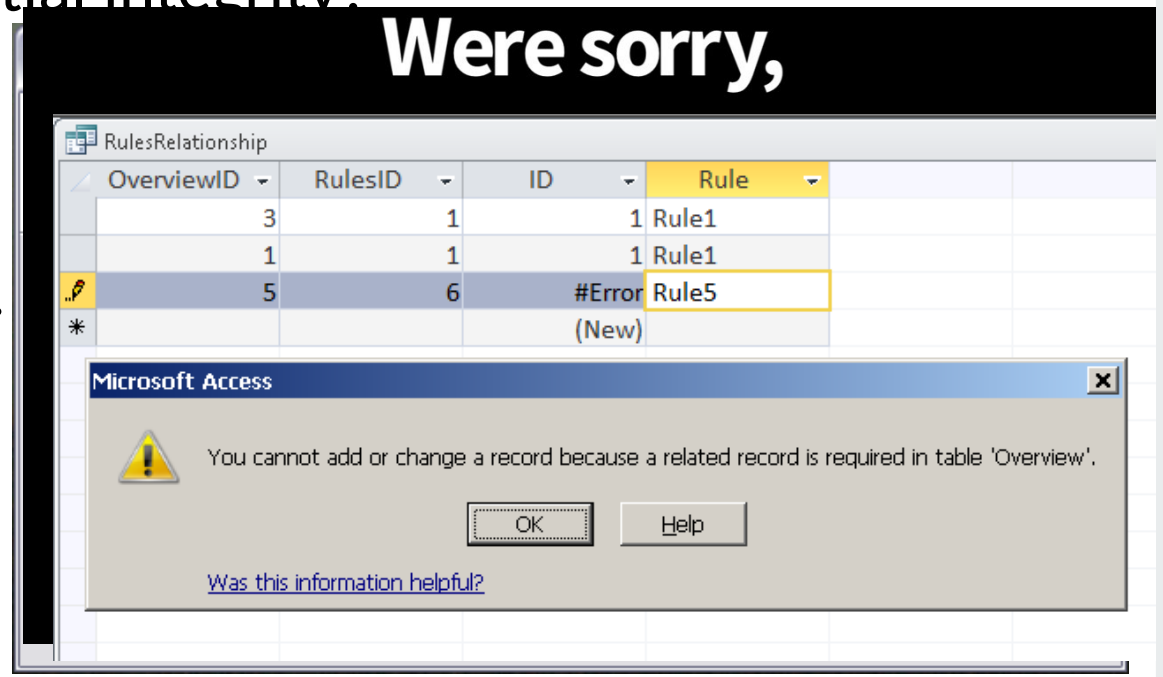
Cannot happen

REFERENCING RELATION



Referential integrity: Foreign keys

- If all foreign key constraints are enforced, **referential integrity** is achieved, i.e., no dangling references, dissimilar values, etc.
- Can you name a data model w/o referential integrity?
- Links in HTML.
- Pointers in C++.
- Phone numbers recorded in your phone.
- In Database.



Foreign Key in action.

Find the total quantity and items for orders of customers living in "Pennsylvania".

Customers	Customer#	Name	Street	City	Country
	AT01	Alan Turing	Maida Vale	London	UK
	JB01	Jean Bartik	Woodland Walk	Pennsylvania	USA
	MH01	Margaret Hamilton	300 E Street	Pennsylvania	USA
	AL01	Ada Lovelace	Hucknall Road	Nottingham	UK
	EC01	Edgar F. Codd	15 Parks Road	Oxford	UK

Items	Item#	Description	Category
	0001	Hard Disk Drive	Internal Hardware
	0002	16GB RAM	Internal Hardware
	0003	Mechanical Keyboard	Peripherals
	0004	LCD 32" HD Monitor	Display
	0005	2200 RTX GPU 11GB	Internal Hardware

Process "Customers": Find rows with "Pennsylvania" and Get the Key Values.

Process "Orders": Find Item# and Quantity using keys.

Process "Items": Find Descriptions using Item#.

Orders	Order#	Item#	Customer#	Delivery_date	Quantity
	Or0022	0002	MH01	2020-02-10	2
	Or0023	0004	AL01	2020-01-30	1
	Or0024	0001	AT01	2020-02-05	1
	Or0025	0005	JB01	2020-02-06	1
	Or0026	0003	EC01	2020-02-01	3
	Or0027	0004	JB01	2020-02-03	6

->9, LCD32" Monitor, 2200 RTX GPU 11GB, 16GBram.

Can you draw the ER diagram for these relations? (May be next week!)

Strategies to enforce Referential Integrity

- Consider Students and Enrolled; *sid* in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted?
- *Reject it!*

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B
1177	SCC201	A

Strategies to enforce Referential Integrity

- What should be done if a Students tuple (say 53650) is deleted?

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53650	Shero	shero@eecs	18	3.2
53689	Smith	smith@math	19	3.8

- Also delete all Enrolled tuples that refer to it.
- Disallow deletion of a Students tuple that is referred to.
- Set sid in Enrolled tuples that refer to it to a *default sid*.
- (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value *null*, denoting *`unknown'* or *`inapplicable'*.)

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

- What happens if the primary key of the Students tuple is updated (53650 to 00001)?

Referential Integrity in SQL/92

- SQL/92 supports all 4 options on deletes and updates.
  - Default is NO ACTION (*delete/update is rejected*)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

Lets speculate about ICs. for the following tables.

Customers	Customer#	Name	Street	City	Country
	AT01	Alan Turing	Maida Vale	London	UK
	JB01	Jean Bartik	Woodland Walk	Pennsylvania	USA
	MH01	Margaret Hamilton	300 E Street	Pennsylvania	USA
	AL01	Ada Lovelace	Hucknall Road	Nottingham	UK
	EC01	Edgar F. Codd	15 Parks Road	Oxford	UK

Assume these tables.

Do tables obey key constraints?

Do tables obey domain constraints?

What are the foreign keys?

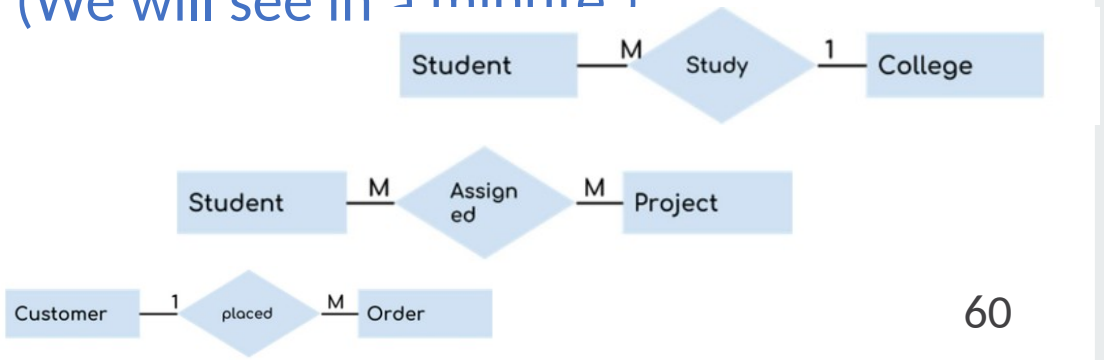
"Print the orders of Customers whose Customer number starts with A?"

Items	Item#	Description	Category
	0001	Hard Disk Drive	Internal Hardware
	0002	16GB RAM	Internal Hardware
	0003	Mechanical Keyboard	Peripherals
	0004	LCD 32" HD Monitor	Display
	0005	2200 RTX GPU 11GB	Internal Hardware

Orders	Order#	Item#	Customer#	Delivery_date	Quantity
	Or0022	0002	MH01	2020-02-10	2
	Or0023	0004	AL01	2020-01-30	1
	Or0024	0001	AT01	2020-02-05	1
	Or0025	0005	JB01	2020-02-06	1
	Or0026	0003	EC01	2020-02-01	3
	Or0027	0004	JB01	2020-02-03	6

Integrity Constraints for relational databases.

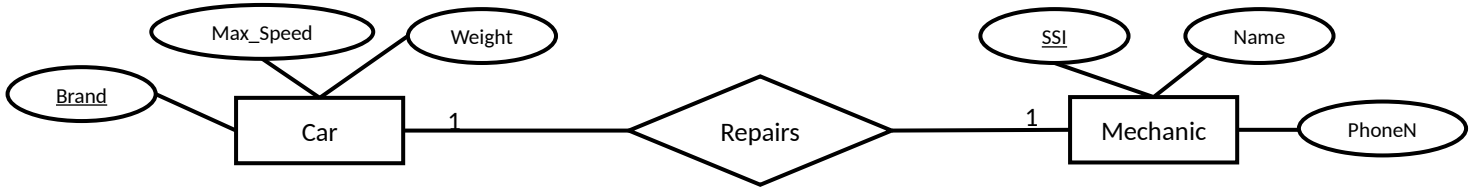
- Integrity of data: it is the state of data in which data obeys the constraints set by DBA.
- 1) Domain constraints.
  - Values in tuples should obey types of attributes. (You should not provide text to INT field)
- 2) Entity Integrity constraints.
  - Keys of a relation must be unique, non-redundant, and not Null (entity integrity constraint)
- 3) Referential integrity.
  - How are two relations related to each other? (We will see in a minute.)
    - The relation must be made by using keys,
    - The DBMS must preserve this relation.



- EXTREMELY IMPORTANT CONTENT A HEAD!!!

How do we derive Foreign Keys and ICs for different relationship types?

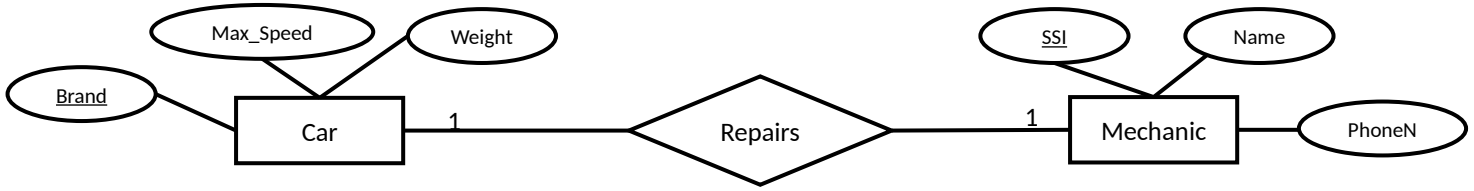
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
-
-
-
  -
  -
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

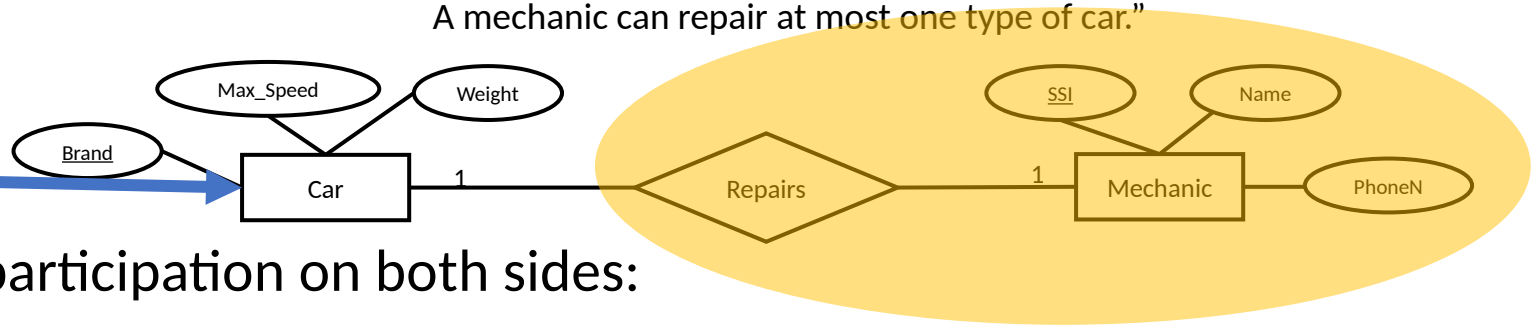
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
-
-
  -
  -
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."

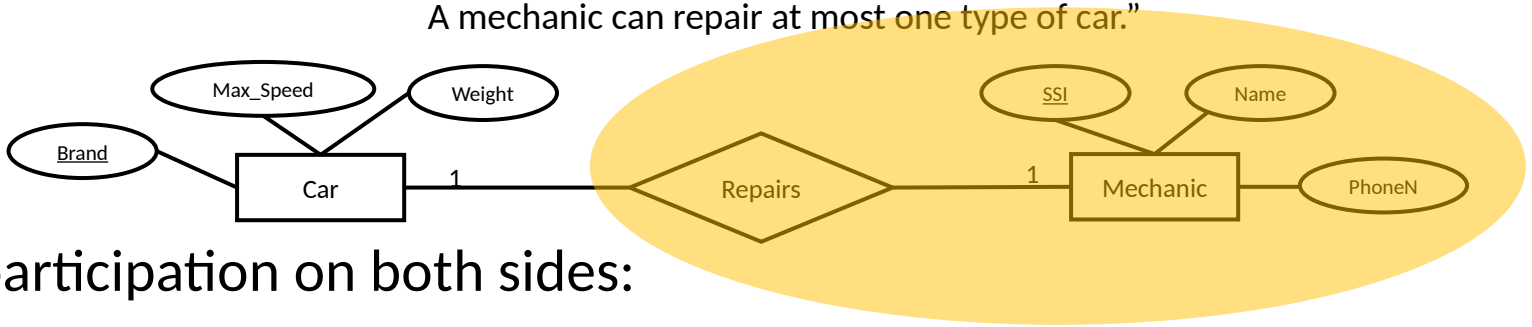


Referenced Table

- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
-
-
  -
  -
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

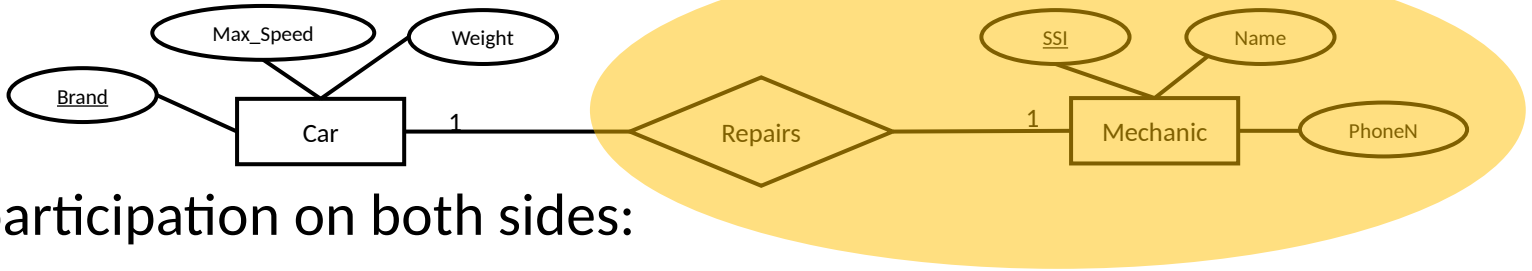
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
- Import the primary key of the referenced table to the referencing one.
-
  -
  -
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

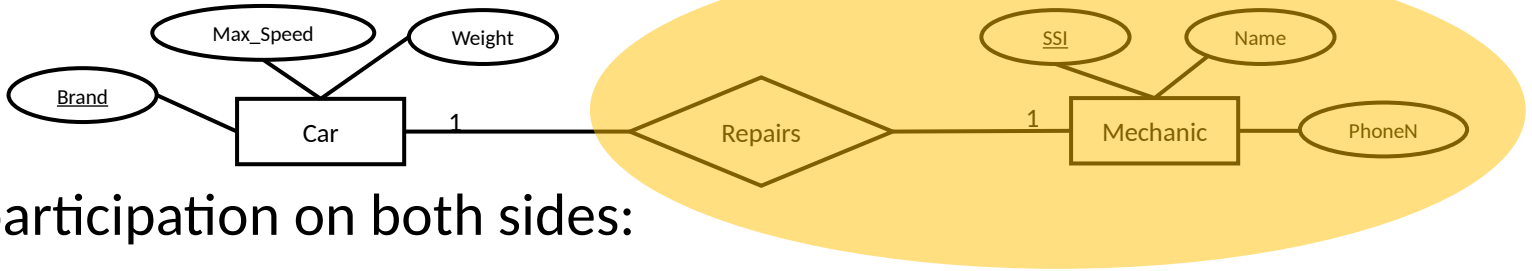
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
- Import the primary key of the referenced table to the referencing one.
- This key will be the foreign key and declare it its foreign key:
  -
  -
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

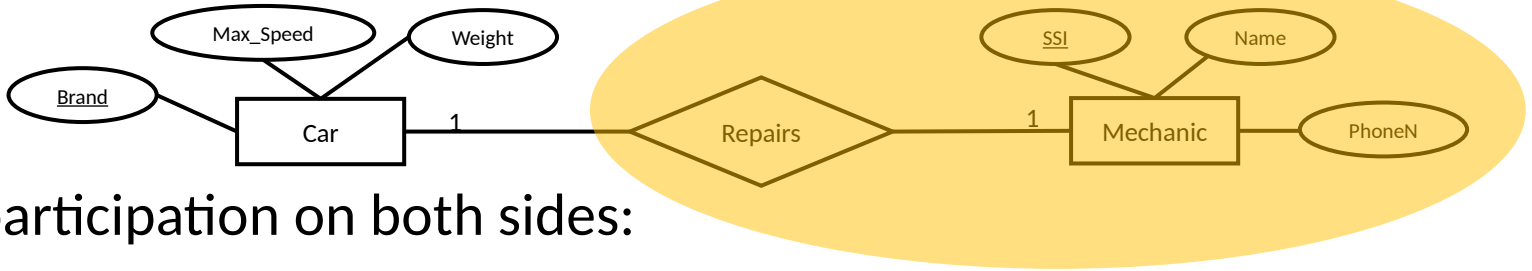
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
- Import the primary key of the referenced table to the referencing one.
- This key will be the foreign key and declare it its foreign key:
  - Car(*Brand*:TEXT,*Weight*:INT,*Length*:DOUBLE,*Max_Speed*:INT, *PRIMARY KEY*:BRAND)
  - Mec_Rep(SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*,Brand:*TEXT*, *PRIMARY KEY*:SSI, *Foreign Key: Brand REFERENCING:CAR*)
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

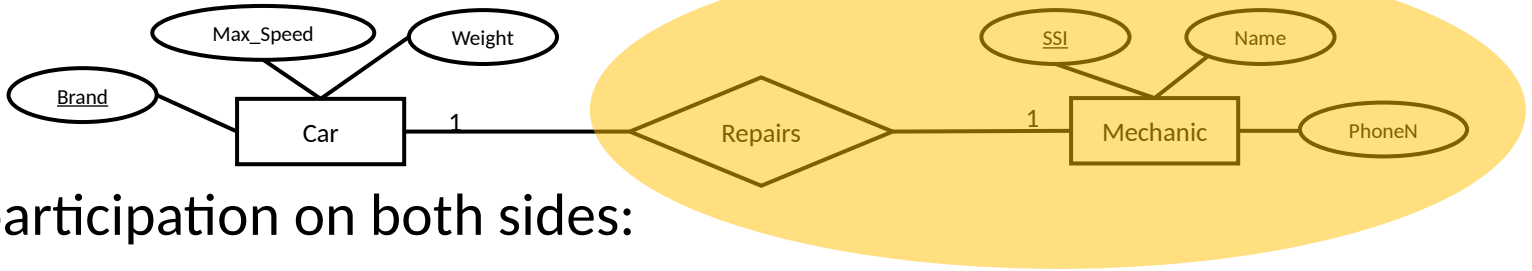
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
- Import the primary key of the referenced table to the referencing one.
- This key will be the foreign key and declare it its foreign key:
  - Car(*Brand*:TEXT,*Weight*:INT,*Length*:DOUBLE,*Max_Speed*:INT, *PRIMARY KEY*:BRAND)
  - Mec_Rep(SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*,Brand:*TEXT*, *PRIMARY KEY*:SSI, *Foreign Key: Brand REFERENCING*:CAR)
-
  -

How do we derive Foreign Keys and ICs for different relationship types?

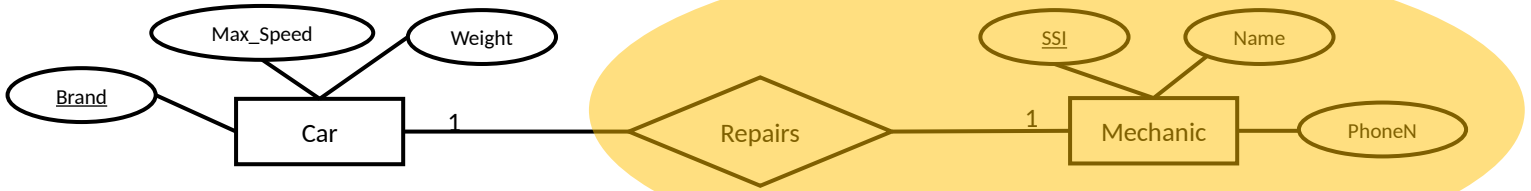
"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relation having partial participation on both sides:
- Select one table (randomly) as the referenced table and the other as the referencing table.
- Import the primary key of the referenced table to the referencing one.
- This key will be the foreign key and declare it its foreign key:
  - Car(*Brand*:TEXT,*Weight*:INT,*Length*:DOUBLE,*Max_Speed*:INT, *PRIMARY KEY*:BRAND)
  - Mec_Rep(SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*,Brand:*TEXT*, *PRIMARY KEY*:SSI, *Foreign Key: Brand REFERENCING*:CAR)
- Do you think that this is enough?
  -

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



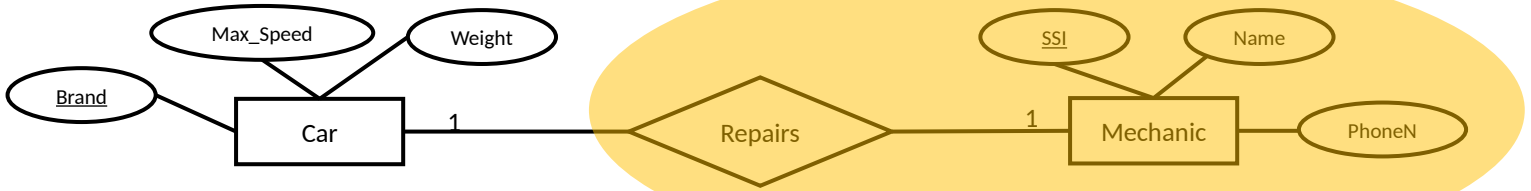
- 1 to 1 relation having partial participation on both sides:
  - Car(*Brand*:TEXT,*Weight*:INT,*Length*:DOUBLE,*Max_Speed*:INT, *PRIMARY KEY*:BRAND)
  - Mec_Rep(SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*,Brand:*TEXT*, *PRIMARY KEY:SSI*, *Foreign Key: Brand REFERENCING:CAR*)

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



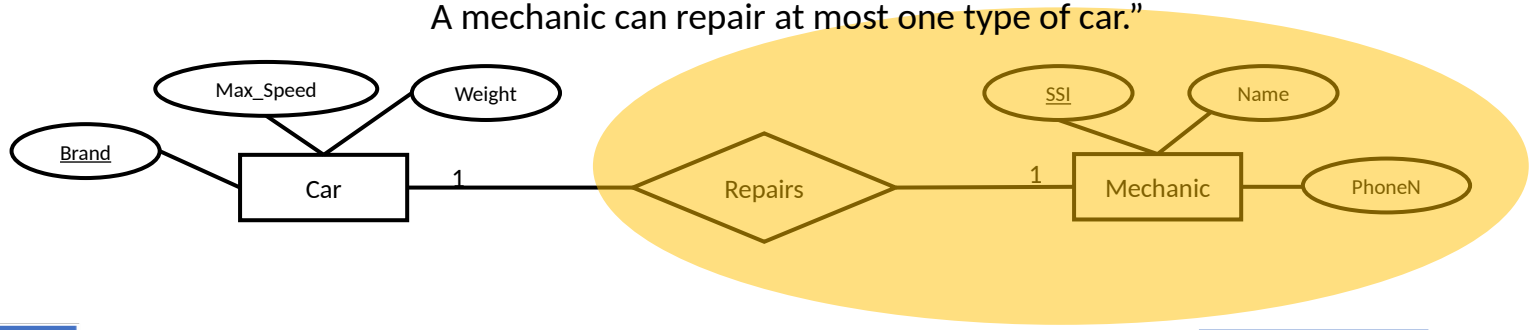
- 1 to 1 relation having partial participation on both sides:
  - Car(*Brand*:TEXT,*Weight*:INT,*Length*:DOUBLE,*Max_Speed*:INT, *PRIMARY KEY*:BRAND)
  - Mec_Rep(SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*,Brand:*TEXT*, *PRIMARY KEY:SSI*, *Foreign Key: Brand REFERENCING:CAR*)

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21
43279823	[illegible]	72352362	BMW 3.21

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relations

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

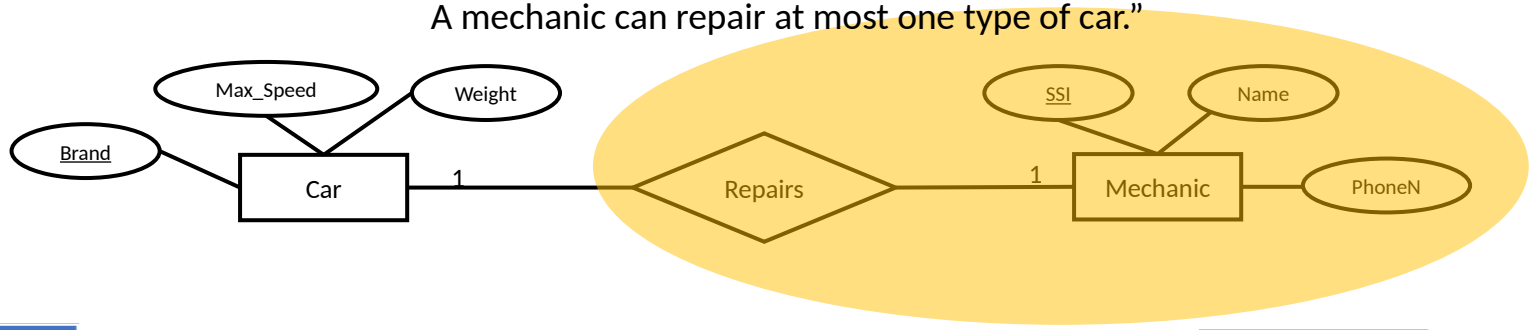
Car(Brand:TEXT,Weight:INT,Length:DOUBLE, Max_Speed:INT, PRIMARY KEY:BRAND)

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR)

The repairs relation is one-to-one. Therefore, for every SSI, there must exist one Brand. Moreover, as Brand cannot repeat, we use the **UNIQUE** keyword.

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relations

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

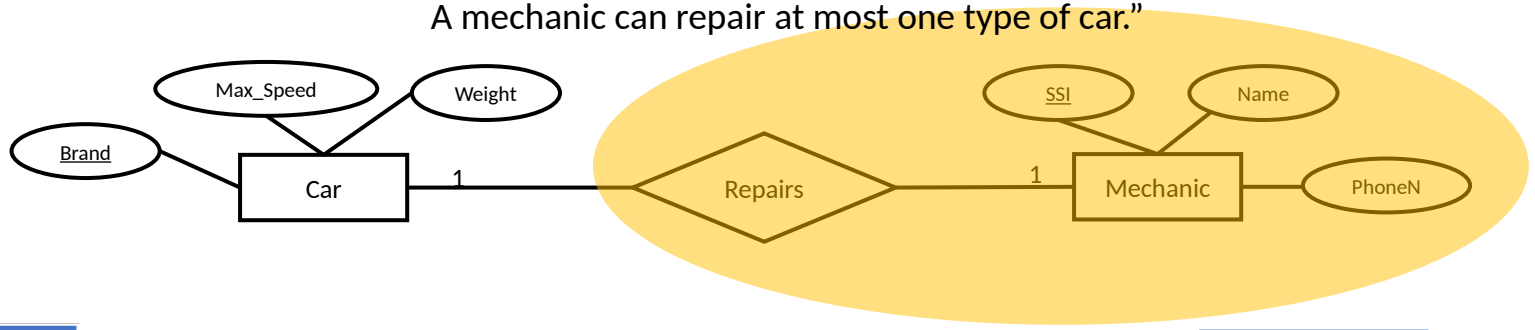
Car(Brand:TEXT,Weight:INT,Length:DOUBLE, Max_Speed:INT, PRIMARY KEY:BRAND)

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE)

The repairs relation is one-to-one. Therefore, for every SSI, there must exist one Brand. Moreover, as Brand cannot repeat, we use the **UNIQUE** keyword.

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relations

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

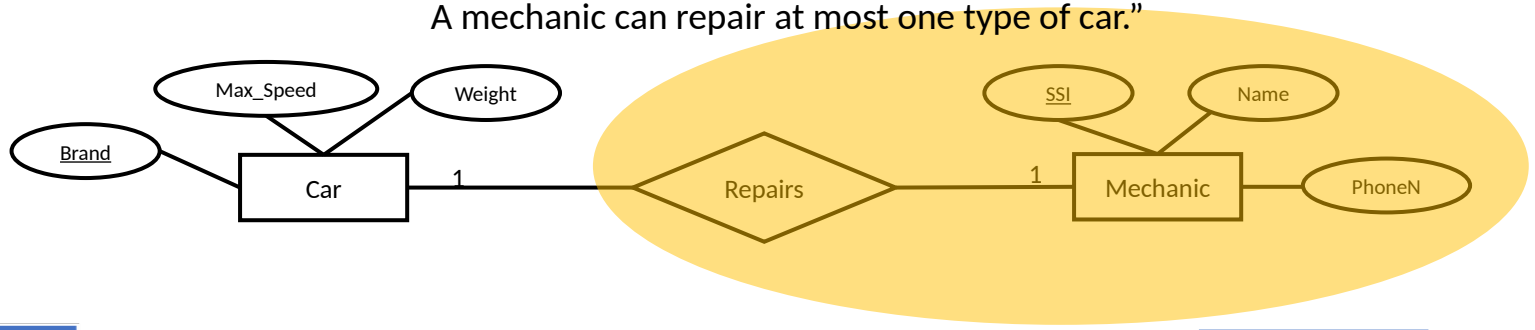
Car(Brand:TEXT,Weight:INT,Length:DOUBLE, Max_Speed:INT, PRIMARY KEY:BRAND)

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE)

Assume I delete the tuple "BMW 3.21, 1400, 3.21, 200" from the CAR table. What value should DBMS set for the Mechanic that can repair BMW 3.21?

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relations

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

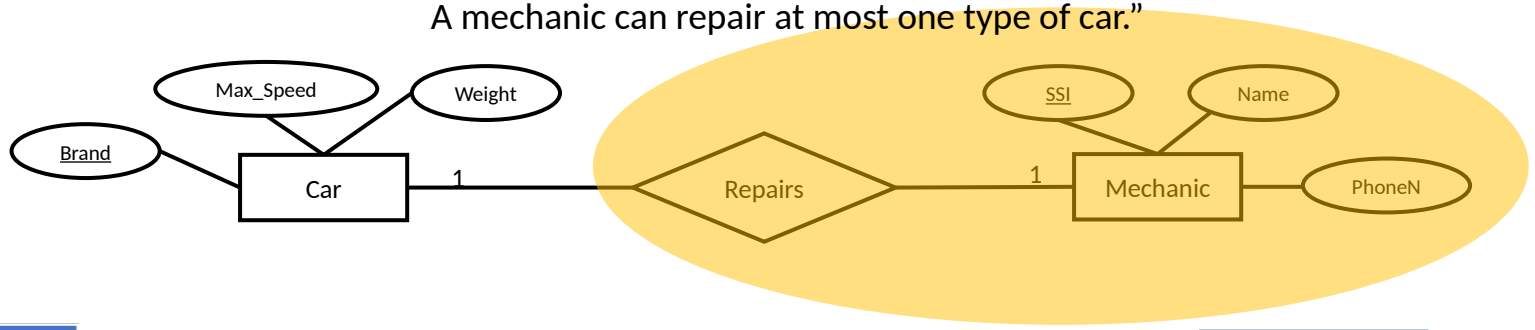
Car(Brand:TEXT,Weight:INT,Length:DOUBLE, Max_Speed:INT, PRIMARY KEY:BRAND)

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE)

Assume I delete the tuple "BMW 3.21, 1400, 3.21, 200" from the CAR table. What value should DBMS set for the Mechanic that can repair BMW 3.21?
Since the Repairs Relation **partially participates** in both ends, I can select **SET NULL** or **SET DEFAULT.**

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relations

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

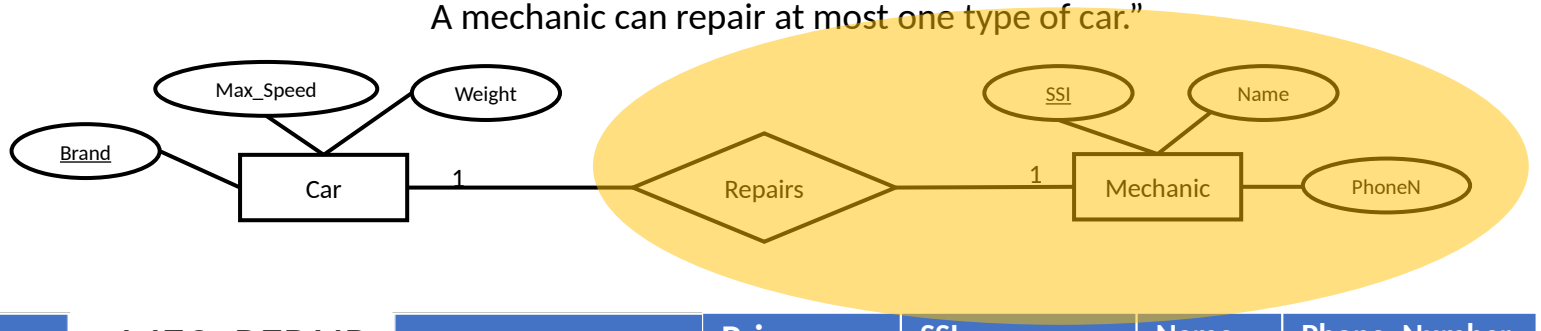
Car(Brand:TEXT,Weight:INT,Length:DOUBLE, Max_Speed:INT, PRIMARY KEY:BRAND)

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE, on Delete SET NULL/DEFAULT)

Assume I delete the tuple "BMW 3.21, 1400, 3.21, 200" from the CAR table. What value should DBMS set for the Mechanic that can repair BMW 3.21?
Since the Repairs Relation **partially participates** in both ends, I can select **SET NULL** or **SET DEFAULT.**

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

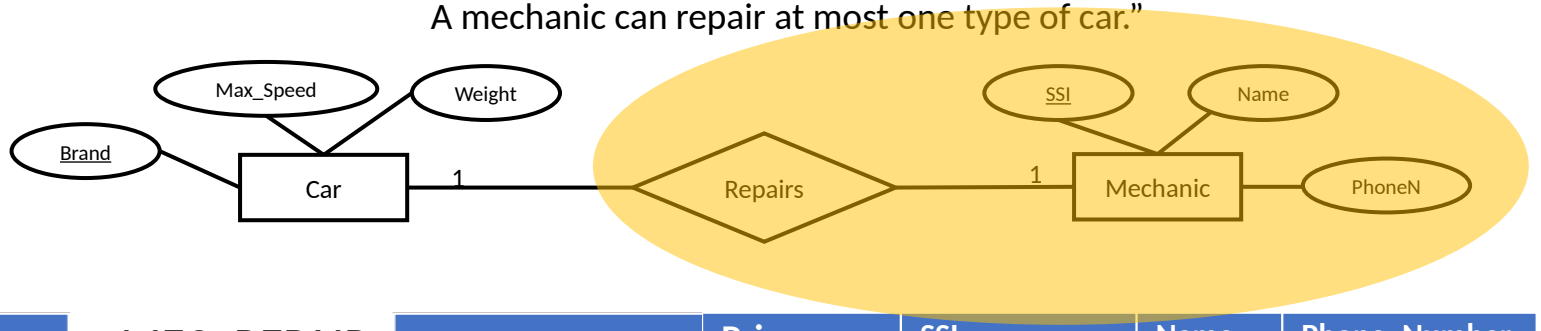
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
-
-
-

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

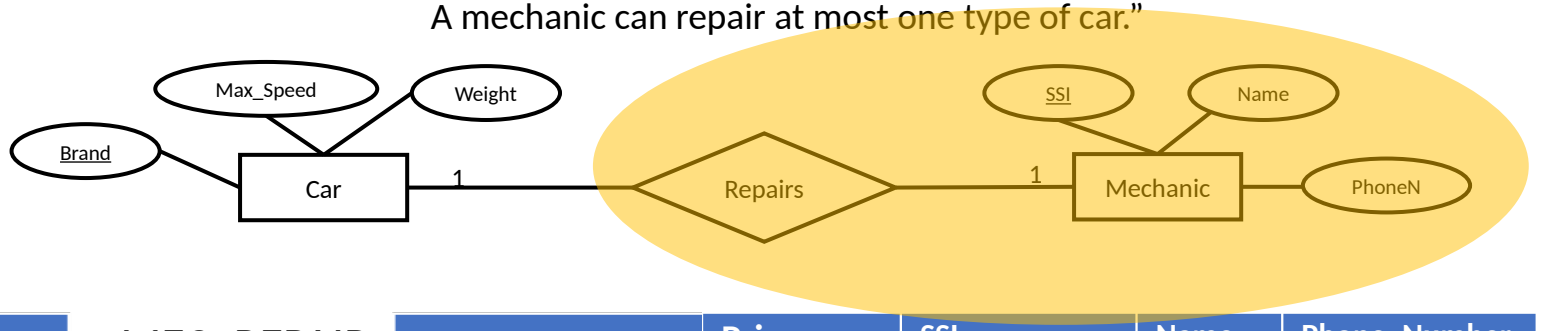
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
-
  -

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

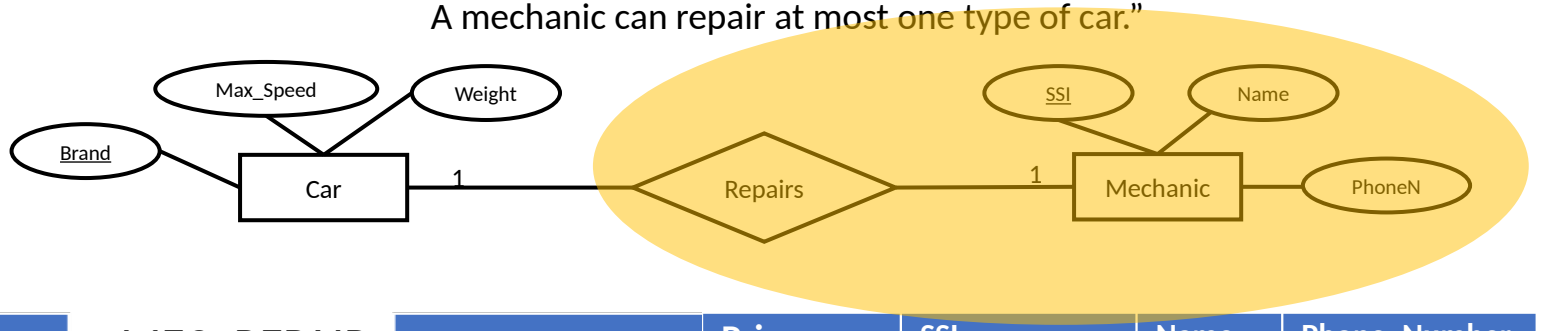
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
-

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

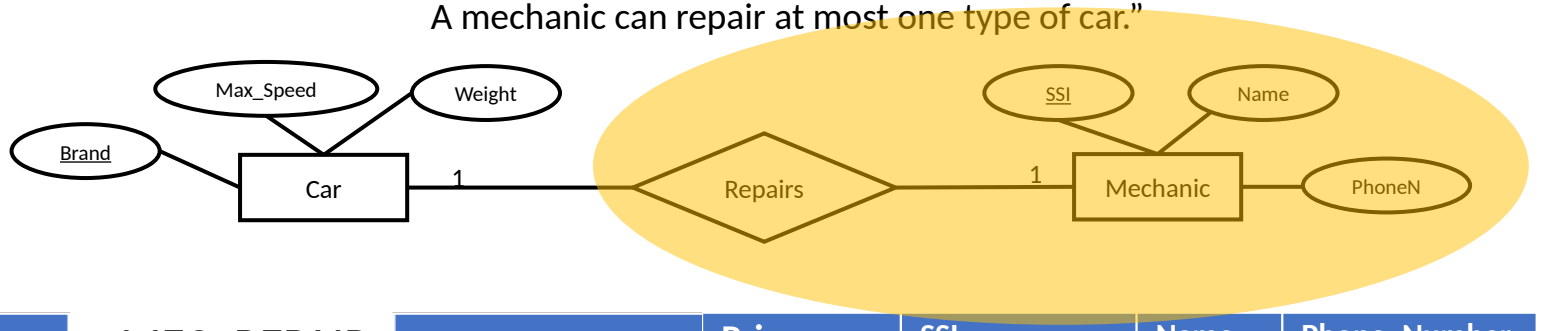
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
DEFAULT	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If SET DEFAULT -> Select all such tuples in the referencing table (MEC_REPAIR) and set the foreign key value to a default value (you have to specify this) of these tuples in the referencing table (MEC_REPAIR).

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Hyundai E.GLS	1400	3.16	210

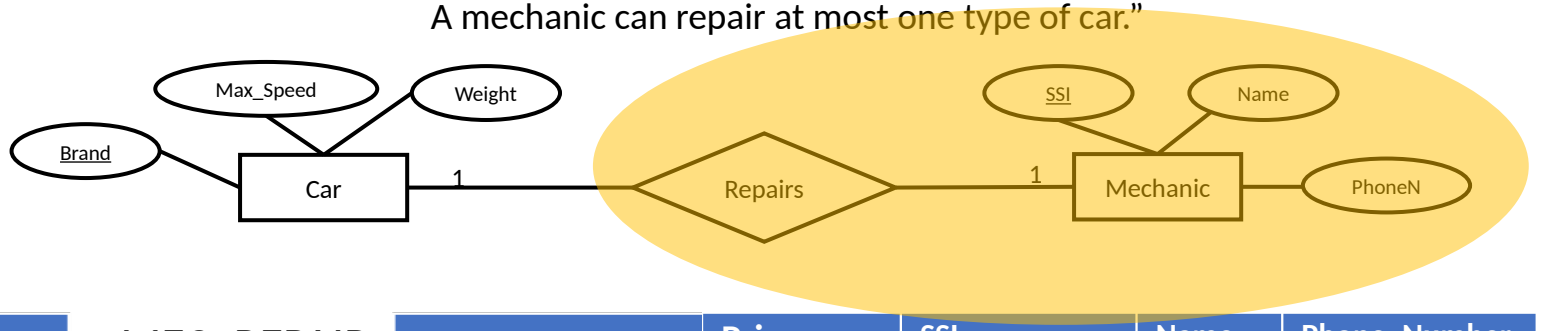
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
DEFAULT	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If SET DEFAULT -> Select all such tuples in the referencing table (MEC_REPAIR) and set the foreign key value to a default value (you have to specify this) of these tuples in the referencing table (MEC_REPAIR). And delete tuples in CAR

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

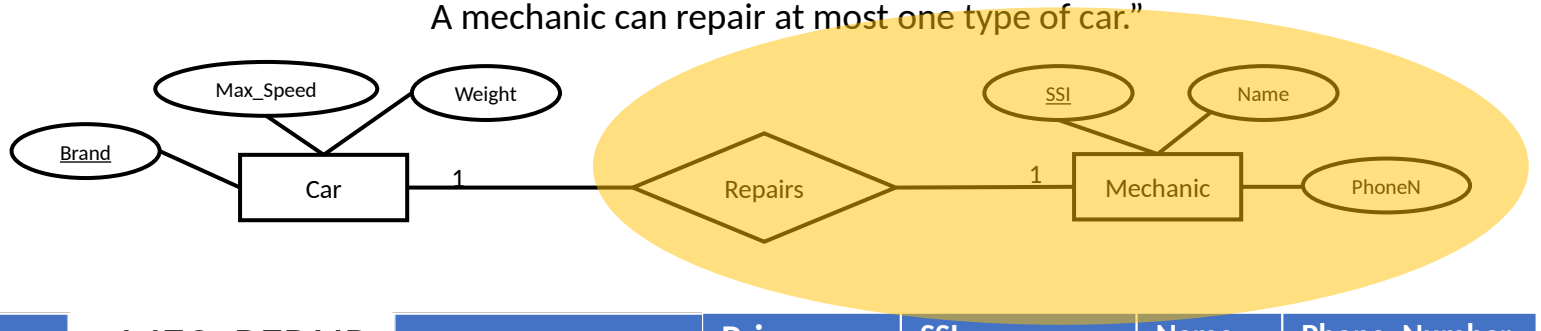
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If SET NULL -> Select all these tuples in the referencing table (MEC_REPAIR)

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

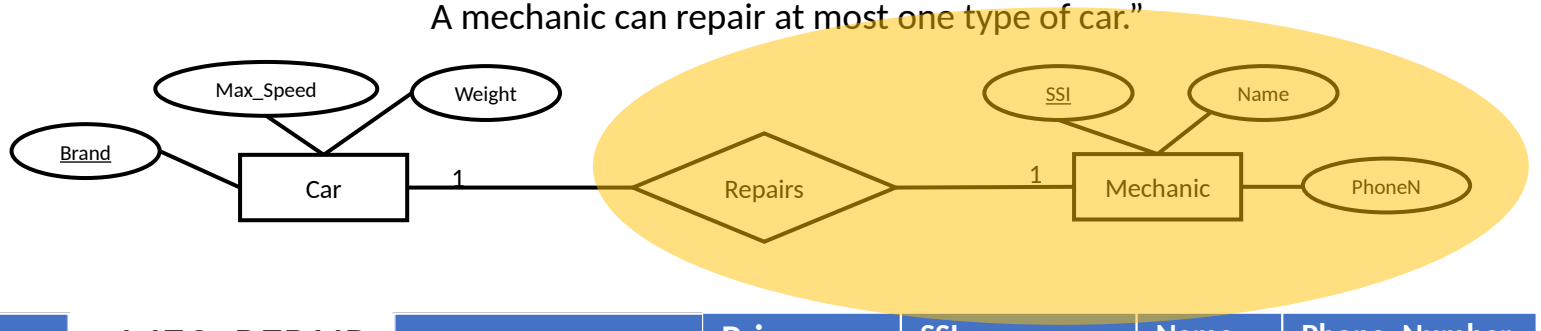
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
NULL	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If SET NULL -> Select all these tuples in the referencing table (MEC_REPAIR) and set the foreign key value to a NULL value of these tuples in the referencing table (MEC_REPAIR).

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic can repair at most one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Hyundai E.GLS	1400	3.16	210

MEC_REPAIR

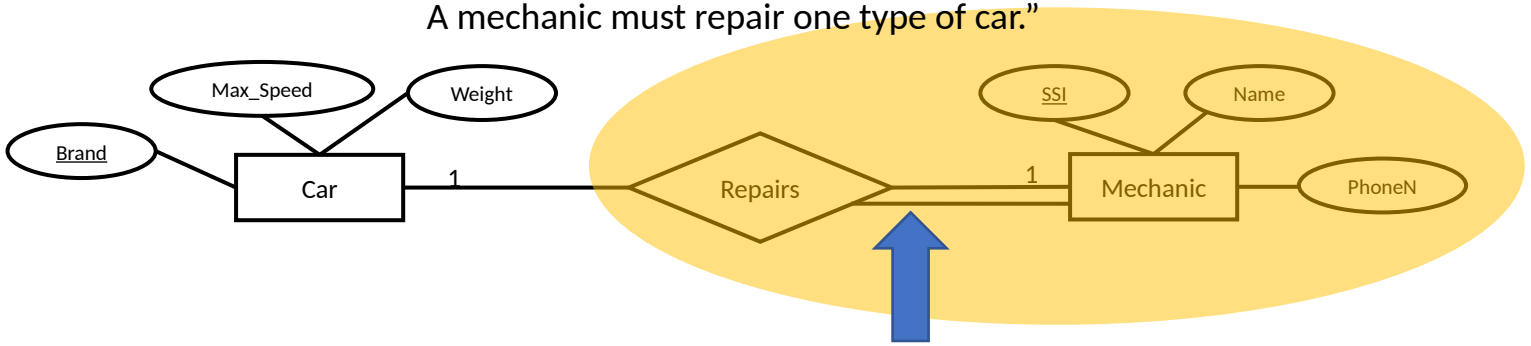
Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
NULL	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If SET NULL -> Select all these tuples in the referencing table (MEC_REPAIR) and set the foreign key value to a NULL value of these tuples in the referencing table (MEC_REPAIR).
  - And delete the tuples in the referenced table (CAR).

How do we derive Foreign Keys and ICs for different relationship types?

- 1 to 1 relations

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



Total Participation: Mechanic Must Repair One Type of Car

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

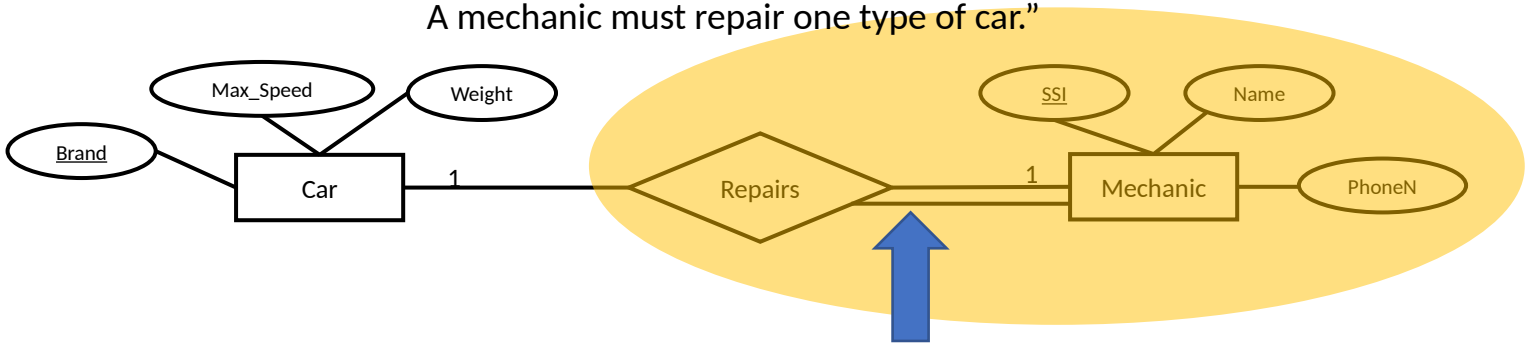
Car(Brand:TEXT,Weight:INT,Length:DOUBLE,Max_Speed:INT, PRIMARY KEY:BRAND)

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR,

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



- 1 to 1 relations

Total Participation: Mechanic Must Repair One Type of Car

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Car(Brand:TEXT,Weight:INT,Length:DOUBLE,Max_Speed:INT, PRIMARY KEY:BRAND)

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

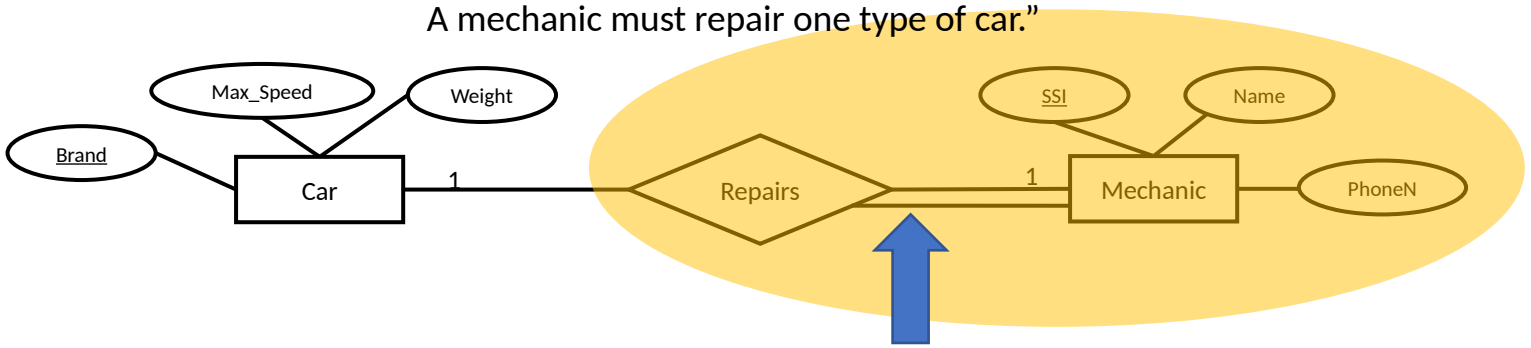
Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE

Since the Repairs relation is one-to-one, the Brand must be unique.

How do we derive Foreign Keys and ICs for different relationship types?

- 1 to 1 relations

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



Total Participation: Mechanic Must Repair One Type of Car

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Car(Brand:TEXT,Weight:INT,Length:DOUBLE,Max_Speed:INT, PRIMARY KEY:BRAND)

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

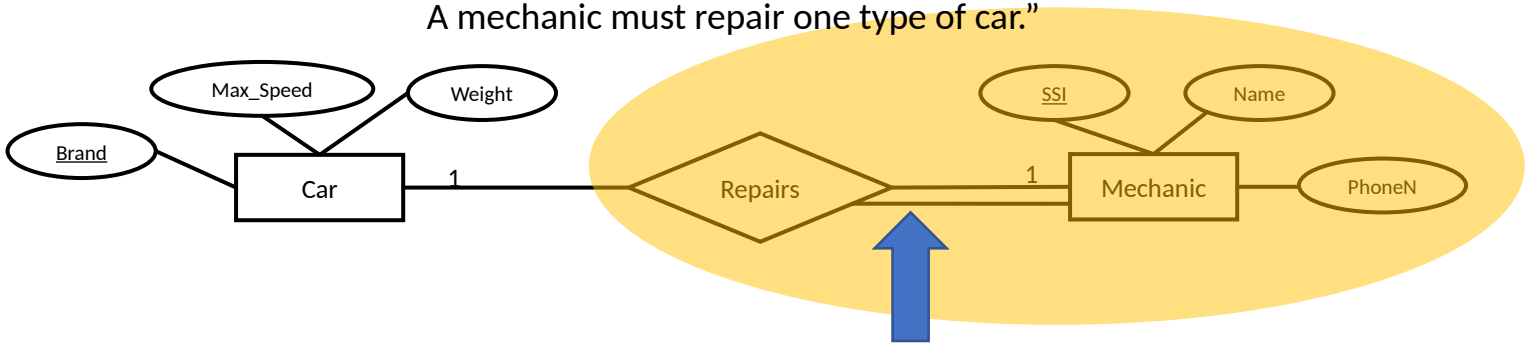
Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE, Brand NOT NULL)

Since the Repairs relation is one-to-one, the Brand must be unique.
Since for every SSI There must exist Brand (total participation), Brand cannot be NULL

How do we derive Foreign Keys and ICs for different relationship types?

- 1 to 1 relations

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



Total Participation: Mechanic Must Repair One Type of Car

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Car(Brand:TEXT,Weight:INT,Length:DOUBLE,Max_Speed:INT, PRIMARY KEY:BRAND)

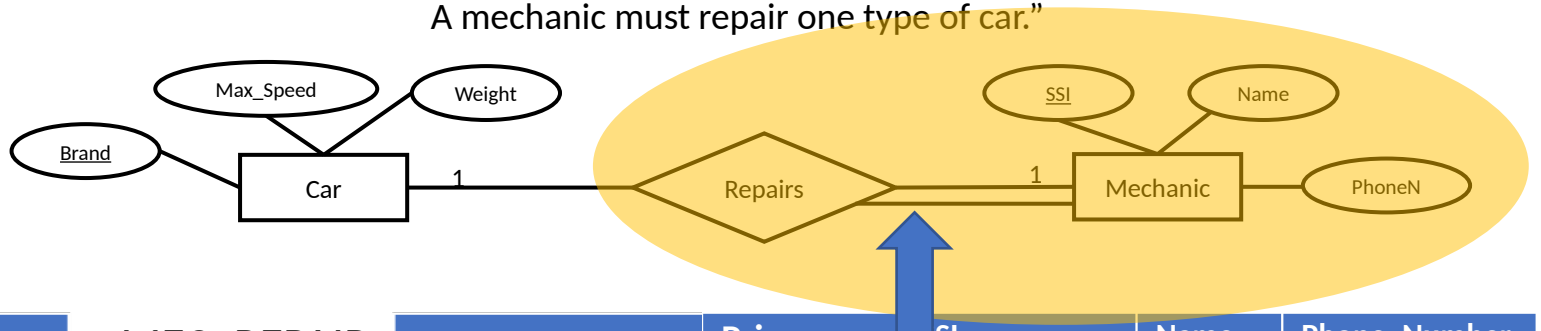
SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

Mec_Rep(SSI:TEXT,Name:TEXT,Phone:TEXT,Brand:TEXT, PRIMARY KEY:SSI, Foreign Key: Brand REFERENCING:CAR, Brand is UNIQUE, Brand NOT NULL, on Delete CASCADE/REJECT)

Since the Repairs relation is one-to-one, the Brand must be unique.
Since for every SSI There must exist Brand (total participation), Brand cannot be NULL, and when a tuple from the referenced table (Car) is removed, DBMS either REJECT this deletion or CASCADE this deletion.

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

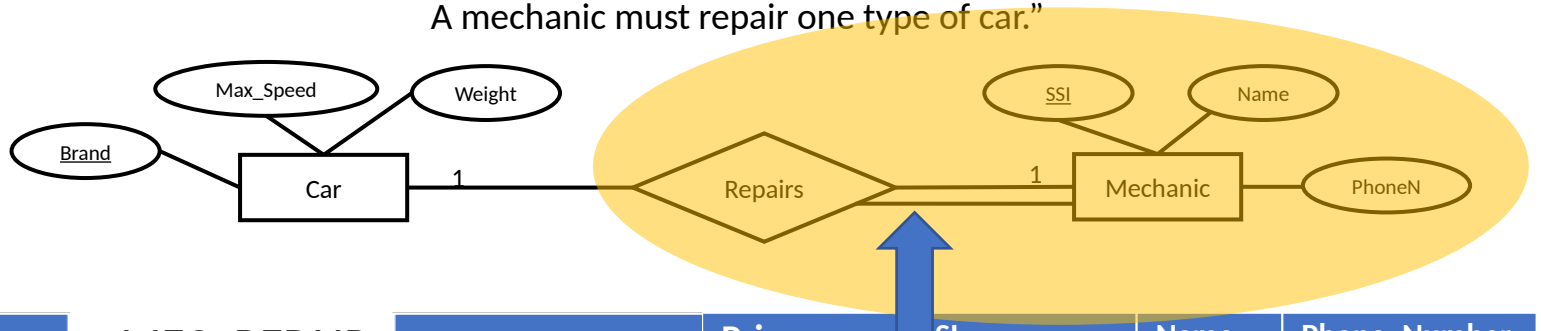
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
-
-
  -
  -

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

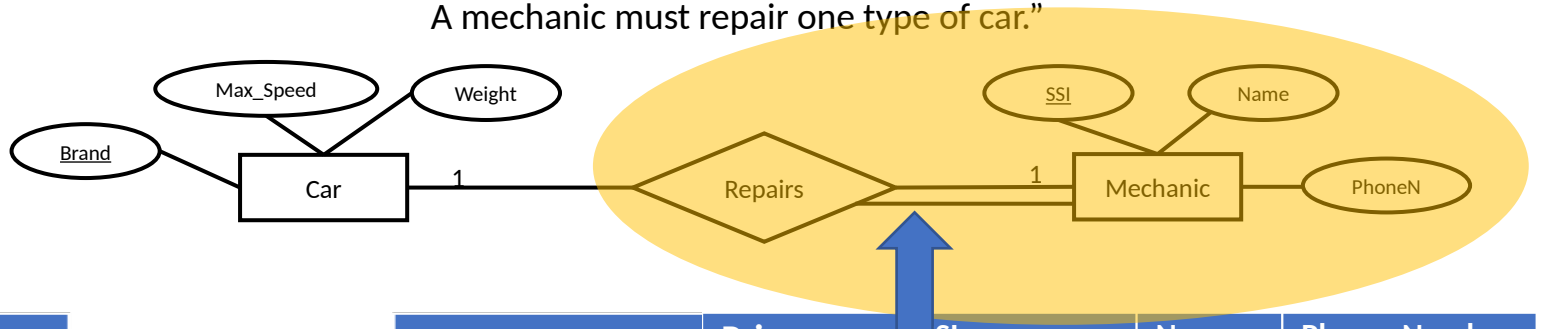
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
-
  -
  -

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

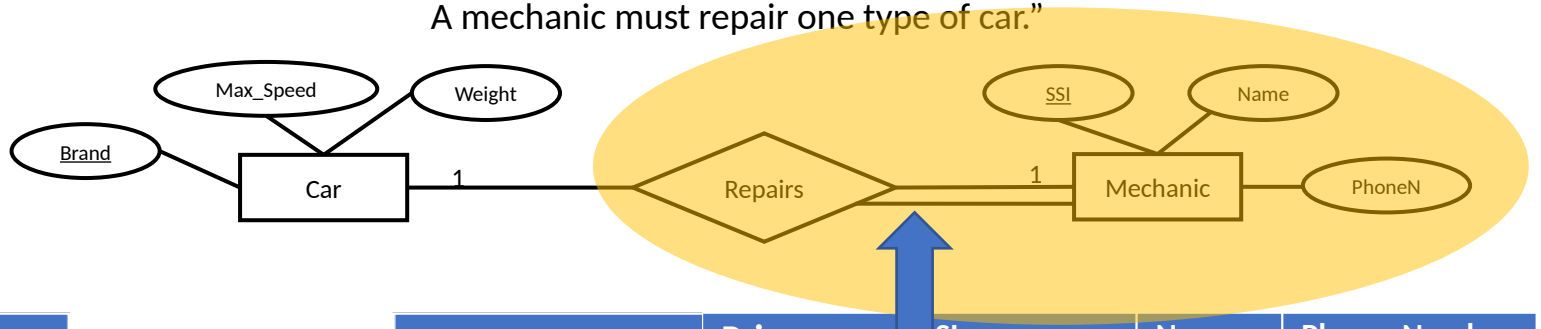
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  -
  -

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

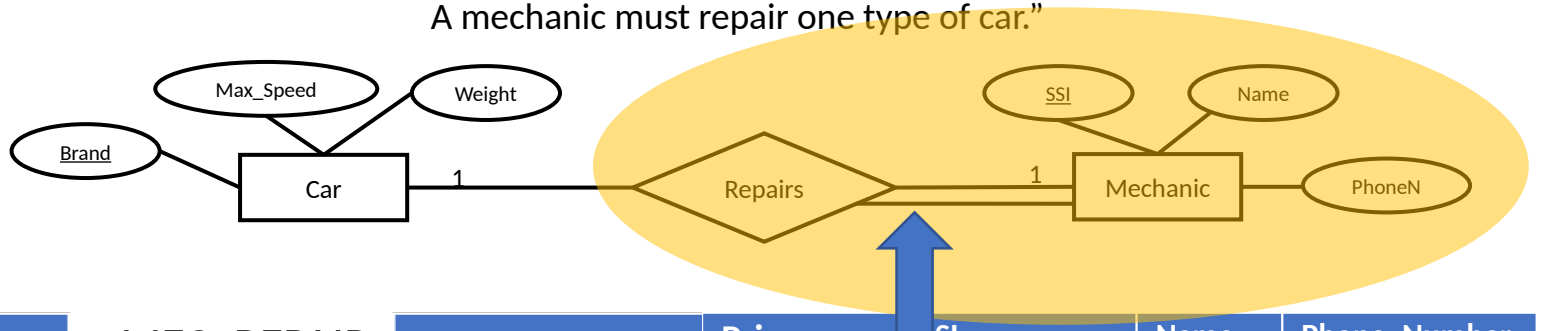
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
[illegible]				
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If CASCADE -> Delete all these tuples in the referencing table (MEC_REPAIR)

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Hyundai E.GLS	1400	3.16	210

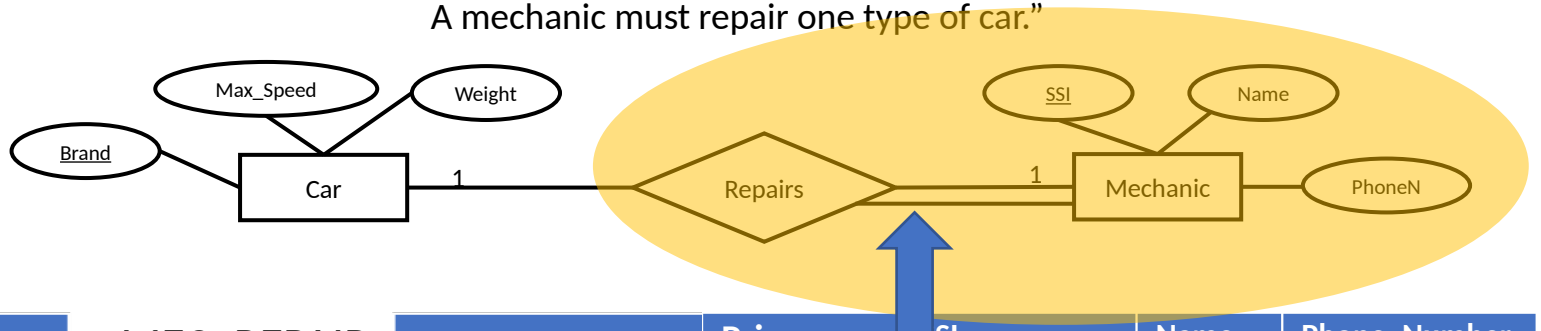
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If CASCADE -> Delete all these tuples in the referencing table (MEC_REPAIR) and delete the tuple in the referenced table (CAR) OR
  -

Referential Integrity

"A car can be repaired by at most one mechanic.
A mechanic must repair one type of car."



CAR

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

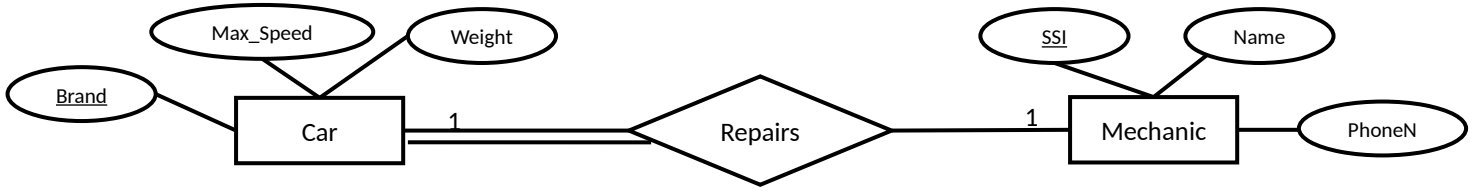
MEC_REPAIR

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

- If a tuple (say 2nd tuple) is to be deleted from referenced table (CAR)
- Get the primary key value of the tuple (Toyota_Corolla).
- Find all the tuples with values (Toyota_Corolla) in the referencing table (MEC_REPAIR)
  - If CASCADE -> Delete all these tuples in the referencing table (MEC_REPAIR) and delete the tuple in the referenced table (CAR) OR
  - If REJECT-> Do NOT allow deletion of the tuple in the referencing and in the referenced table (CAR)

How do we derive Foreign Keys and ICs for different relationship types?

"A car must be repaired by one mechanic.
A mechanic can repair at most one type of car."



- 1 to 1 relations

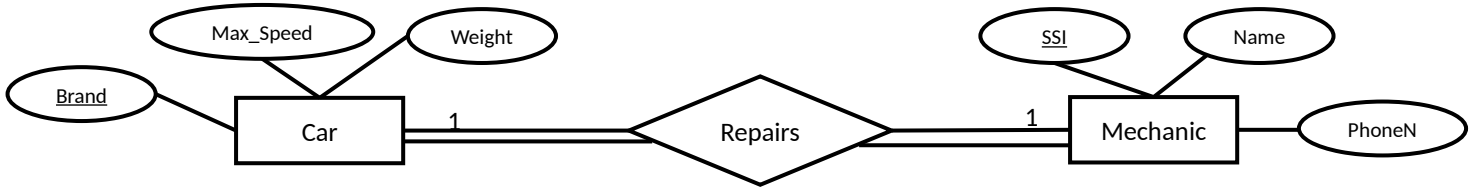
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number	Brand
87542702	Tom	75315567	Toyota_Corolla
68201937	Uraz	75335521	Hyundai E.GLS
23139827	Nick	75315544	BMW 3.21

Study this one…

How do we derive Foreign Keys and ICs for different relationship types?

"A car must be repaired by one mechanic.
A mechanic must repair one type of car."



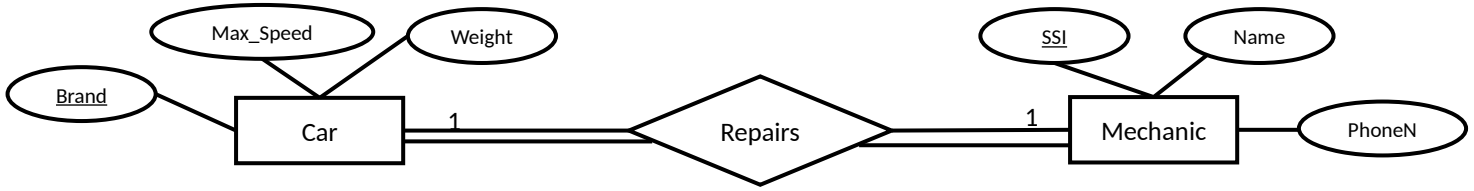
- 1 to 1 relations

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

SSI	Name	Phone_Number
87542702	Tom	75315567
68201937	Uraz	75335521
23139827	Nick	75315544

How do we derive Foreign Keys and ICs for different relationship types?

"A car must be repaired by one mechanic.
A mechanic must repair one type of car."



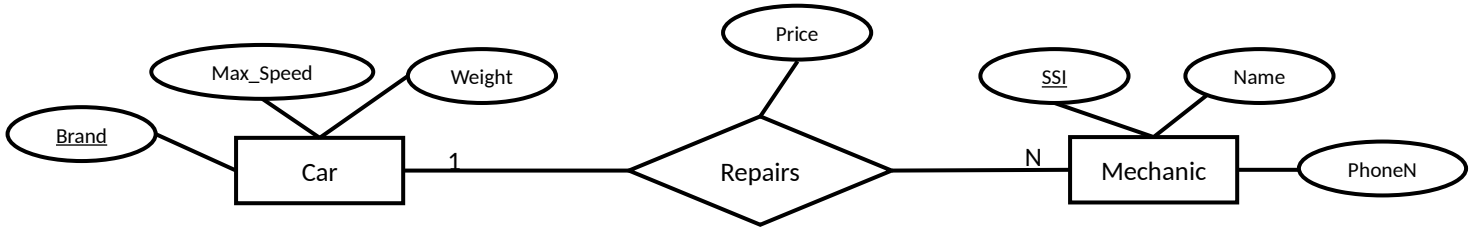
- 1 to 1 relations

Brand	Weight	Length	Max_Speed	SSI	Name	Phone_Number
BMW 3.21	1400	3.21	200	87542702	Tom	75315567
Toyota_Corolla	1300	3.18	200	68201937	Uraz	75335521
Hyundai E.GLS	1400	3.16	210	23139827	Nick	75315544

Car_Rep_Brand(Brand:TEXT,Weight:INT,Length:DOUBLE,Max_Speed:INT, Primary Key:Brand, SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*, *SSI CANNOT NULL*, *SSI IS UNIQUE*)
OR
Car_Rep_Brand(Brand:TEXT,Weight:INT,Length:DOUBLE,Max_Speed:INT, Primary Key:SSI, SSI:*TEXT*,Name:*TEXT*,Phone:*TEXT*, *BRAND CANNOT NULL*, *BRAND IS UNIQUE*)

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."

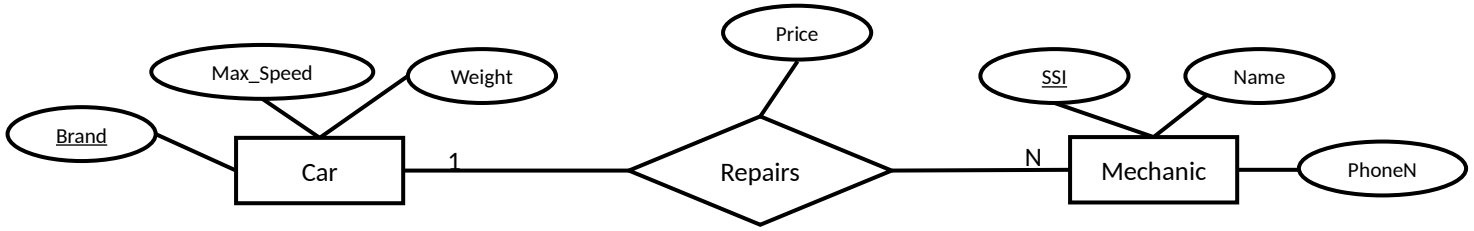


- 1 to Many relations

(Same for Many to 1 relations)

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

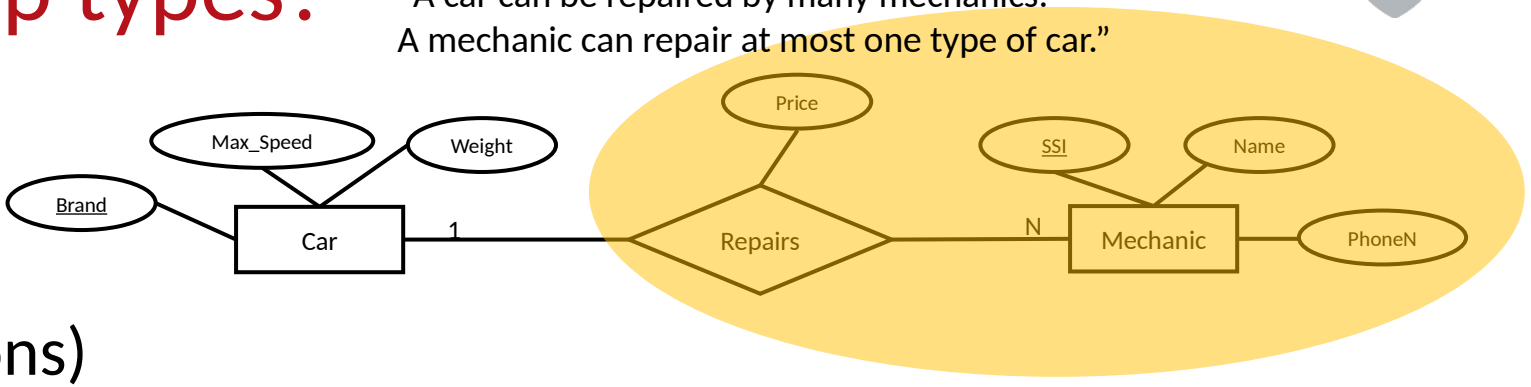
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Price
10
23
12

SSI	Name	Phone_Number
87542702	Tom	75315567
68201937	Uraz	75335521
23139827	Nick	75315544

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."



- 1 to Many relations

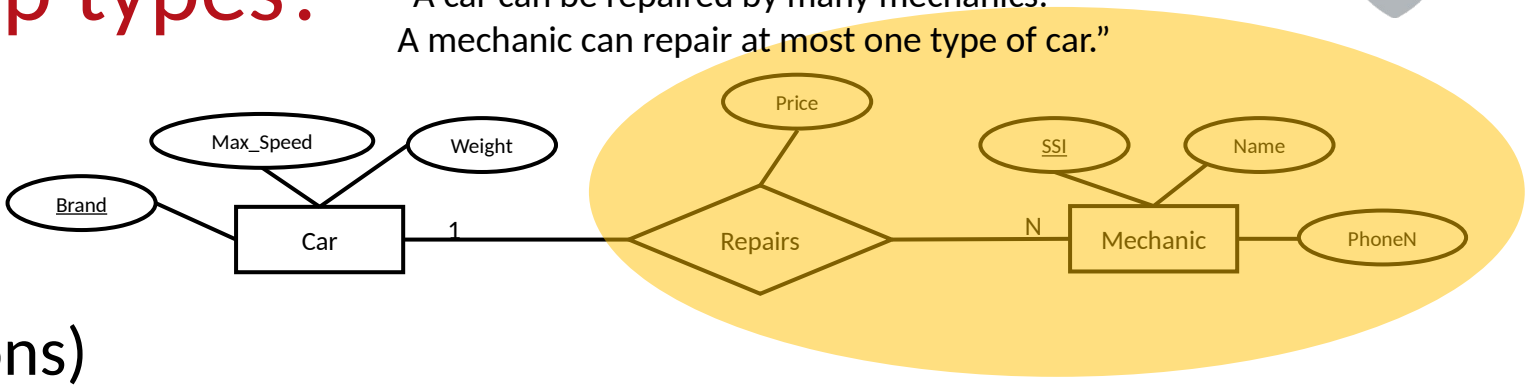
(Same for Many to 1 relations)

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

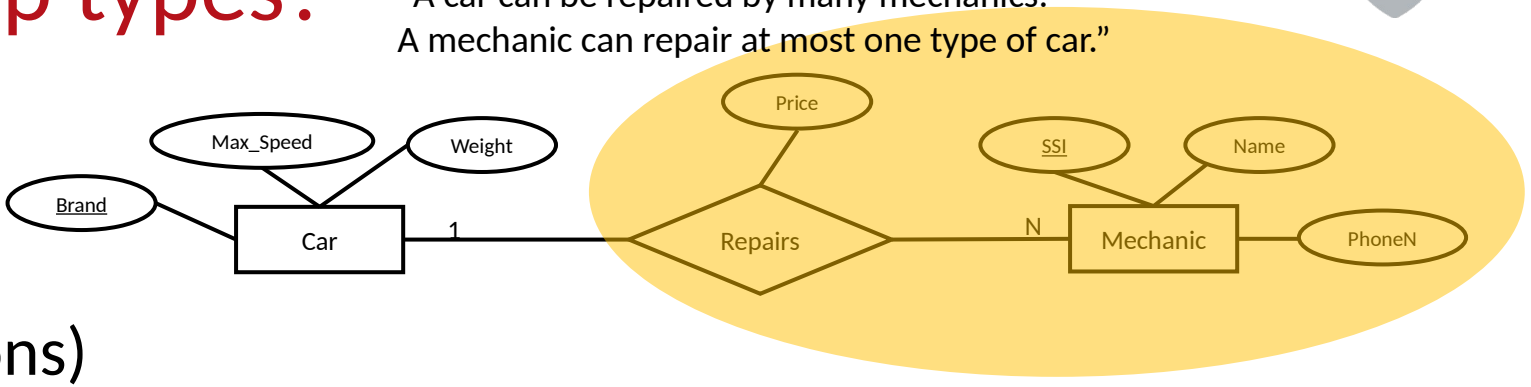
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand)

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

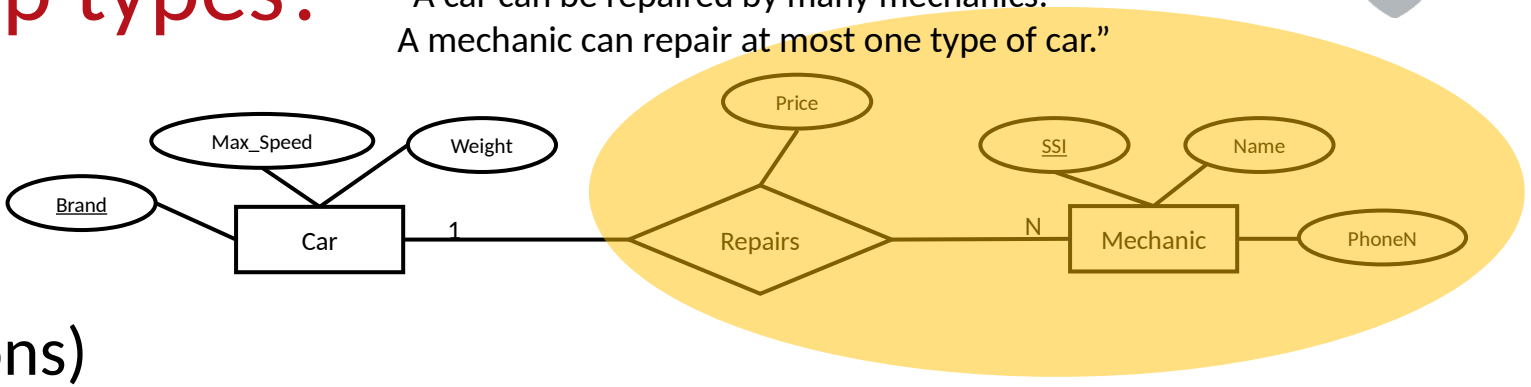
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand)
Mec_R (Brand:*TEXT*,Price: *INT*, SSI:*INT*, Name:*TEXT*, Phone_Number:*TEXT* ,Primary key: SSI, Foreign Key Brand referencing Car.

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

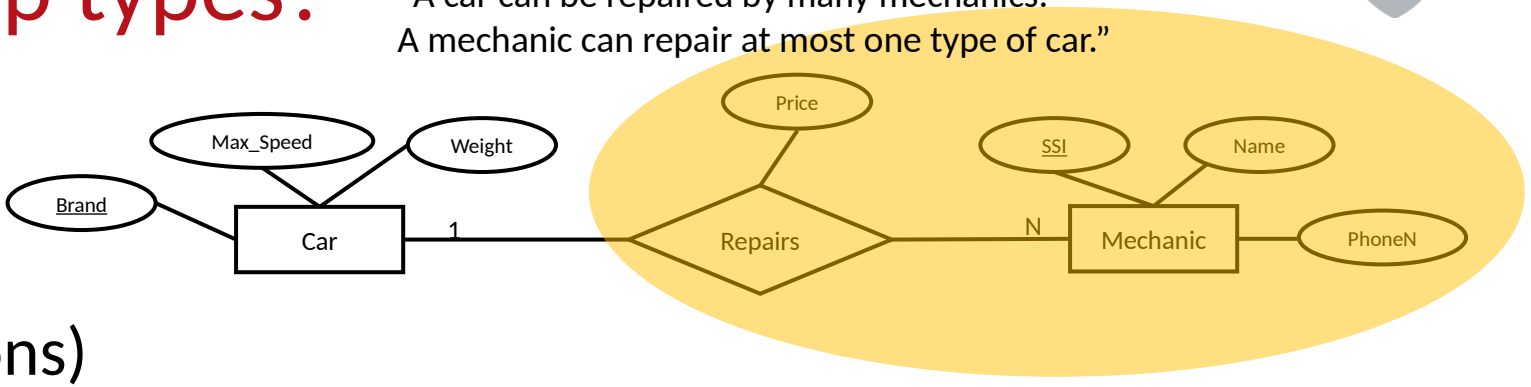
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand)
Mec_R (Brand:*TEXT*,Price: *INT*, SSI:*INT*, Name:*TEXT*, Phone_Number:*TEXT* ,Primary key: SSI, Foreign Key Brand referencing Car. ON DELETE SET NULL/DEFAULT).

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair at most one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

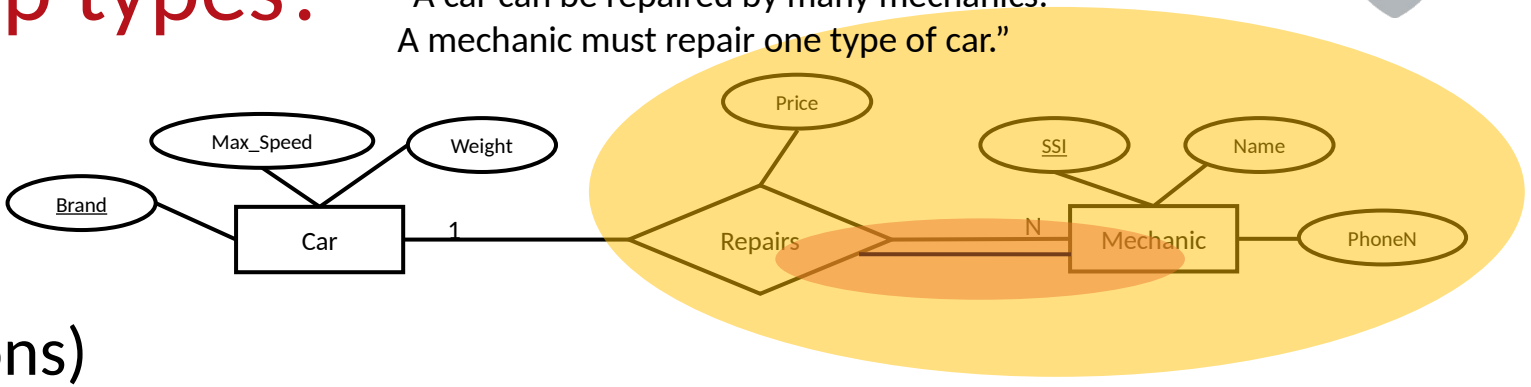
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544
BMW 3.21	11	23761281	Alex	73828732

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand)
Mec_R (Brand:*TEXT*,Price: *INT*, SSI:*INT*, Name:*TEXT*, Phone_Number:*TEXT* ,Primary key: SSI, Foreign Key Brand referencing Car. ON DELETE NULL/DEFAULT). (Should I need to say SSI UNIQUE?)

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic must repair one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

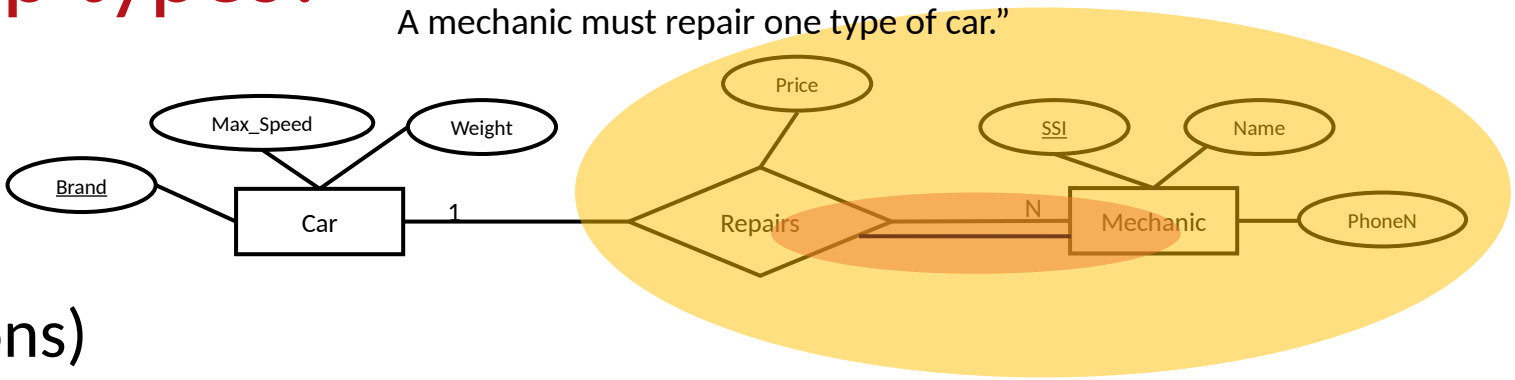
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544
BMW 3.21	11	23761281	Alex	73828732

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand)
Mec_R (Brand:*TEXT*,Price: *INT*, SSI:*INT*, Name:*TEXT*, Phone_Number:*TEXT* ,Primary key: SSI, Foreign Key Brand referencing Car.

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic must repair one type of car."



- 1 to Many relations

(Same for Many to 1 relations)

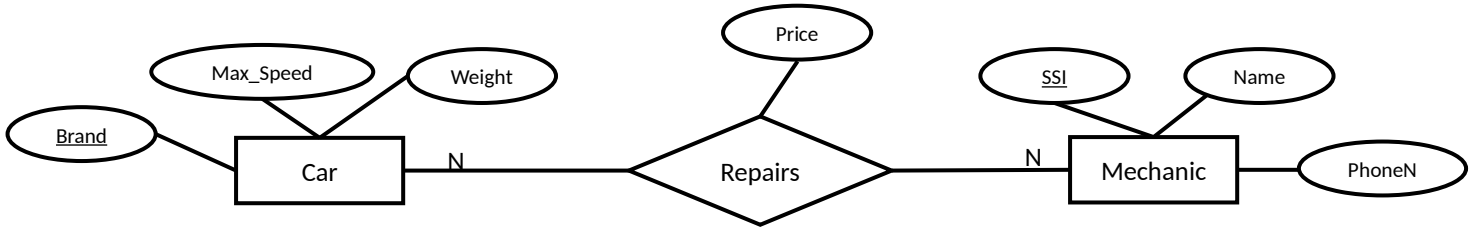
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544
BMW 3.21	11	23761281	Alex	73828732

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand)
Mec_R (Brand:*TEXT*,Price: *INT*, SSI:*INT*, Name:*TEXT*, Phone_Number:*TEXT* ,Primary key: SSI, Foreign Key Brand referencing Car. On Delete CASCADE/REJECT, BRAND CANNOT BE NULL)

How do we derive Foreign Keys and ICs for different relationship types?

"A car can be repaired by many mechanics.
A mechanic can repair many types of car."



- Many to Many (N-N, N-M, X-Y,....)

Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Car (Brand:*TEXT*,Weight: *INT*, Length:*DOUBLE*, Max_Speed:*INT*, Primary key: Brand.

Price	SSI	Brand
10	87542702	BMW 3.21
23	68201937	Toyota_Corolla
12	23139827	Hyundai E.GLS

Rep(SSI:*TEXT*,*Brand*: *TEXT*, Primary Key {Brand, SSI}, Price: *INT*)

Allows combinations!

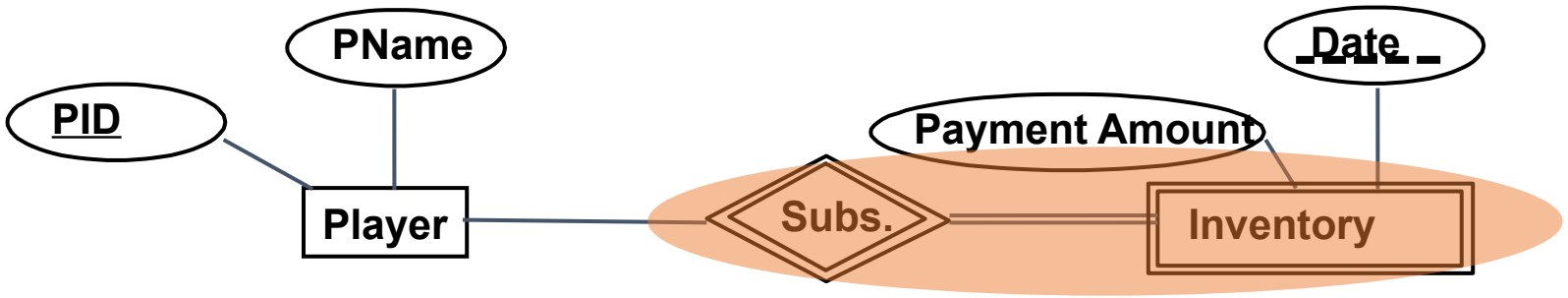
SSI	Name	Phone_Number
87542702	Tom	75315567
68201937	Uraz	75335521
23139827	Nick	75315544

Mec(SSI:*TEXT*, Primary Key {SSI}, Name:*TEXT*, Phone_Number:*TEXT*)

Weak entity sets

Player(PID:INT,PName:TEXT)

- In a weak-entity set, the existence of an entity depends on the existence of an entity in the entity set in relation!

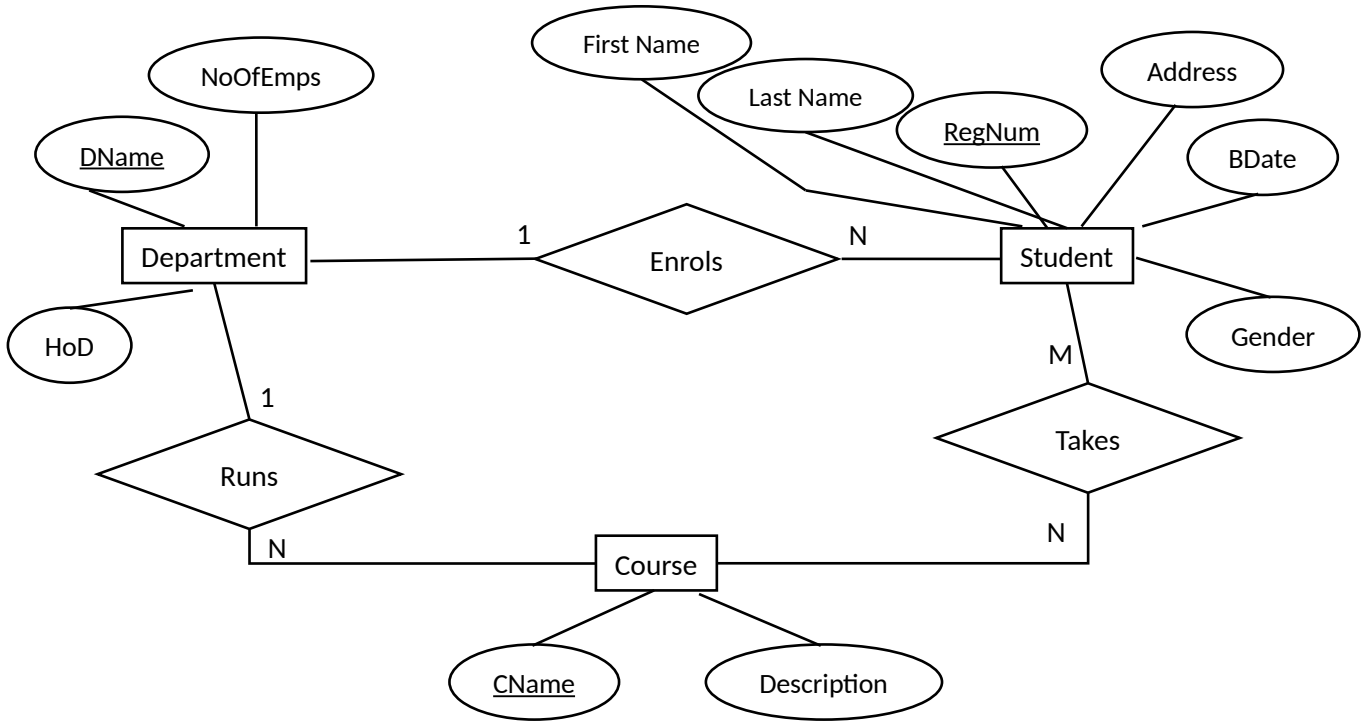


- If a player is deleted from the game server, the inventory information must be deleted.

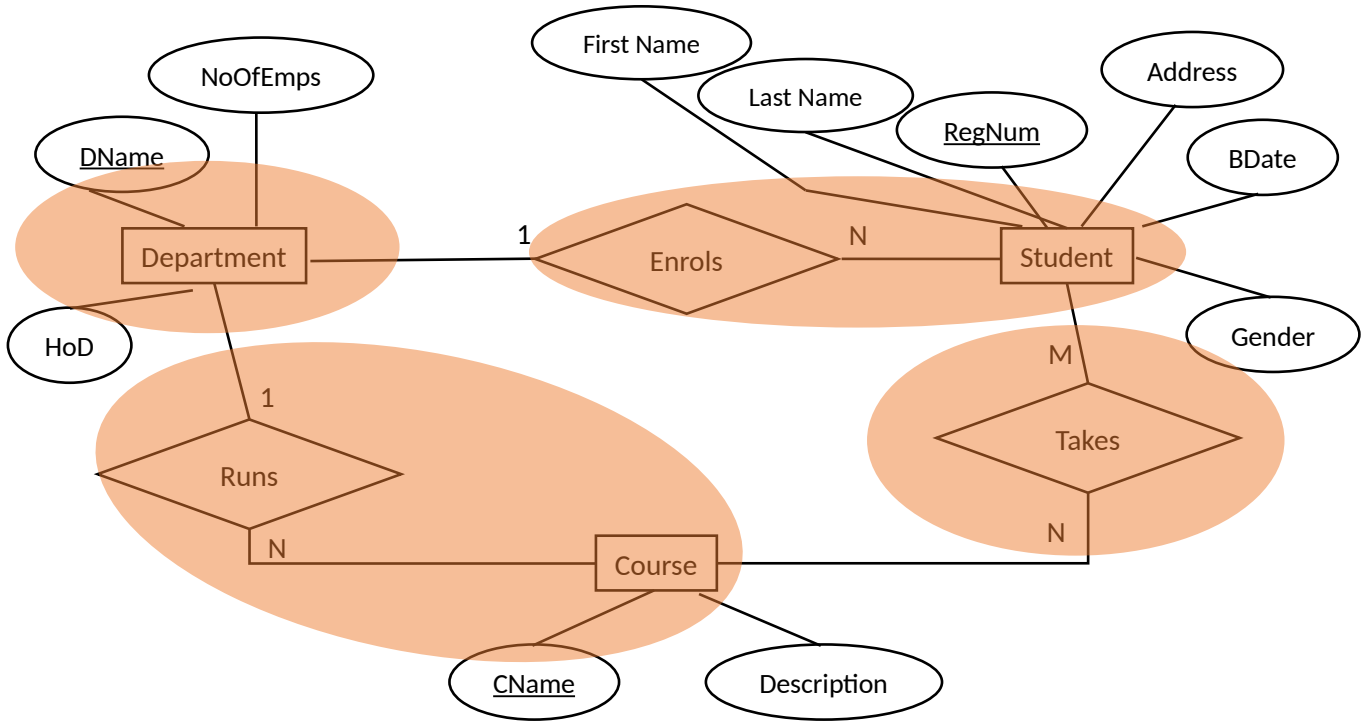
Inv_Subs(PID:INT,PaymentAmound:TEXT,Date:date

,Primary key :{PID,DATE}, foreign key PID, referencing Player, on delete: cascade, PID cannot be null)

Exercise

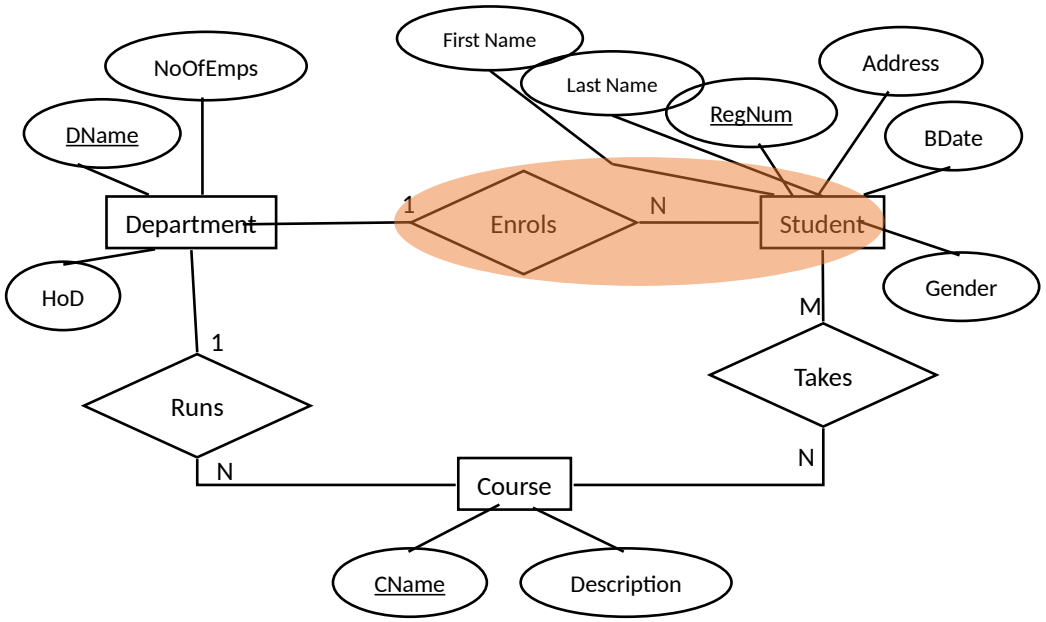

First Name
NoOfEmps
Last Name
Address
RegNum
DName
BDate
1
N
Department
Enrols
Student
HoD
Gender
M
1
Takes
Runs
N
N
Course
CName
Description

Recall


First Name
NoOfEmps
Last Name
Address
RegNum
DName
BDate
1
N
Department
Enrols
Student
HoD
Gender
M
1
Takes
Runs
N
N
Course
CName
Description

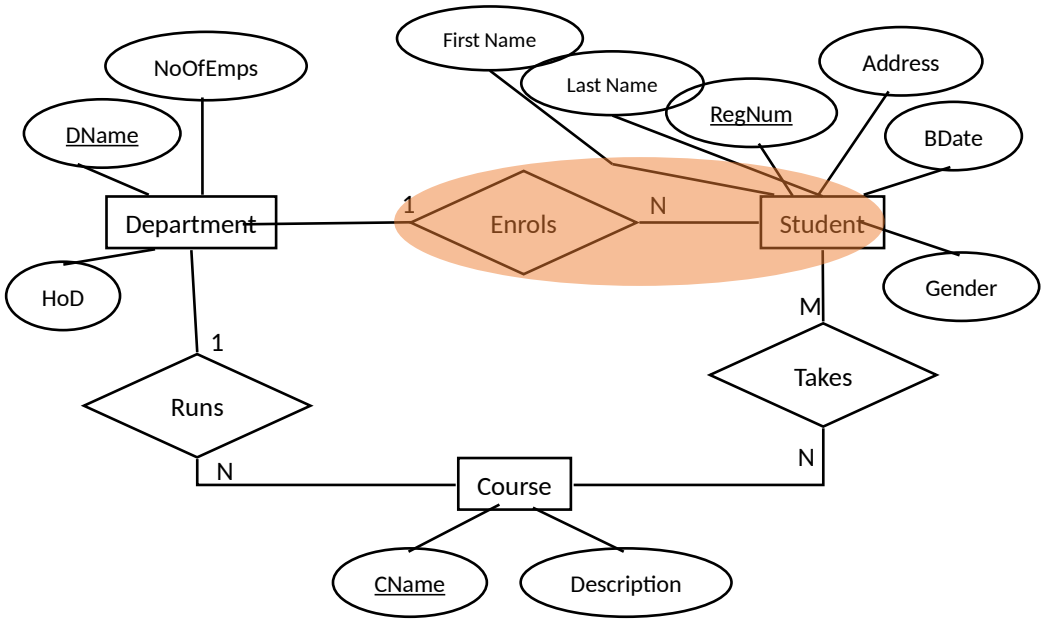
Exercise solutions

- Department(DName TEXT NOT NULL, HoD TEXT, NoOfEmp INTEGER, PRIMARY KEY(DName))
-
-
-



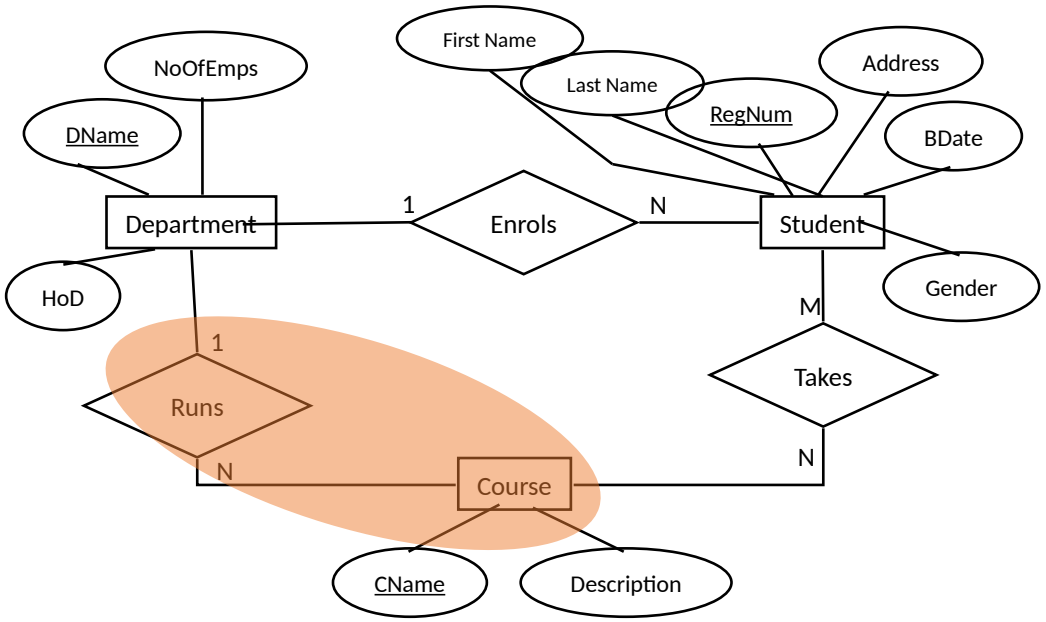
Exercise solutions

- Department(DName TEXT NOT NULL, HoD TEXT, NoOfEmp INTEGER, PRIMARY KEY(DName))
- StudentsEnrol(firstName TEXT, lastName TEXT, RegNumber INTEGER NOT NULL, Address TEXT, BDate TEXT, Gender TEXT, DepName TEXT, PRIMARY KEY (RegNumber), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
-
-



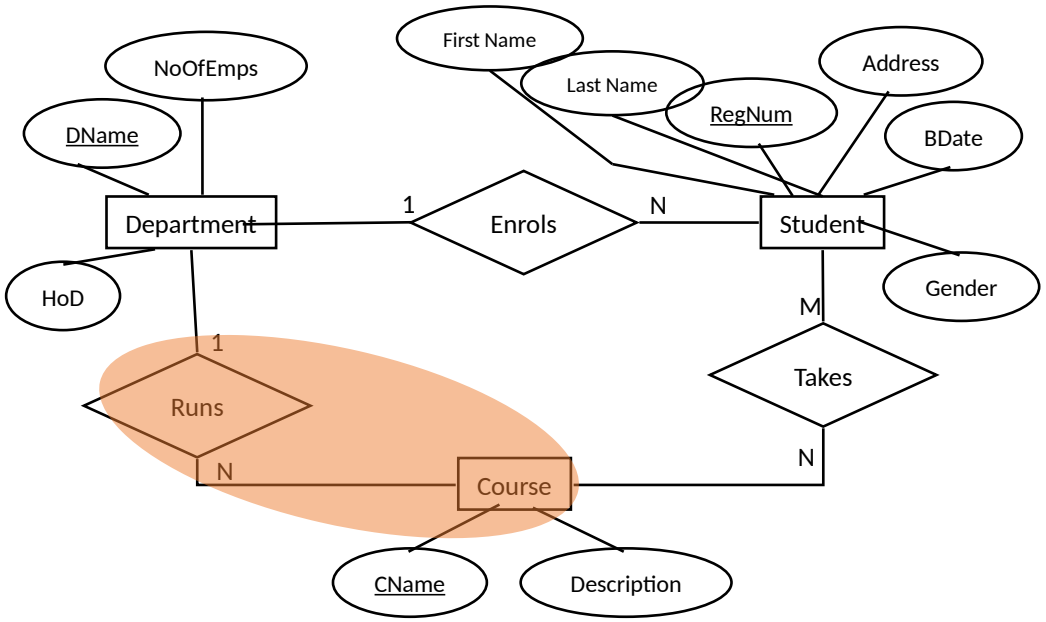
Exercise solutions

- Department(DName TEXT NOT NULL, HoD TEXT, NoOfEmp INTEGER, PRIMARY KEY(DName))
- StudentsEnrol(firstName TEXT, lastName TEXT, RegNumber INTEGER NOT NULL, Address TEXT, BDate TEXT, Gender TEXT, DepName TEXT, PRIMARY KEY (RegNumber), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
-
-



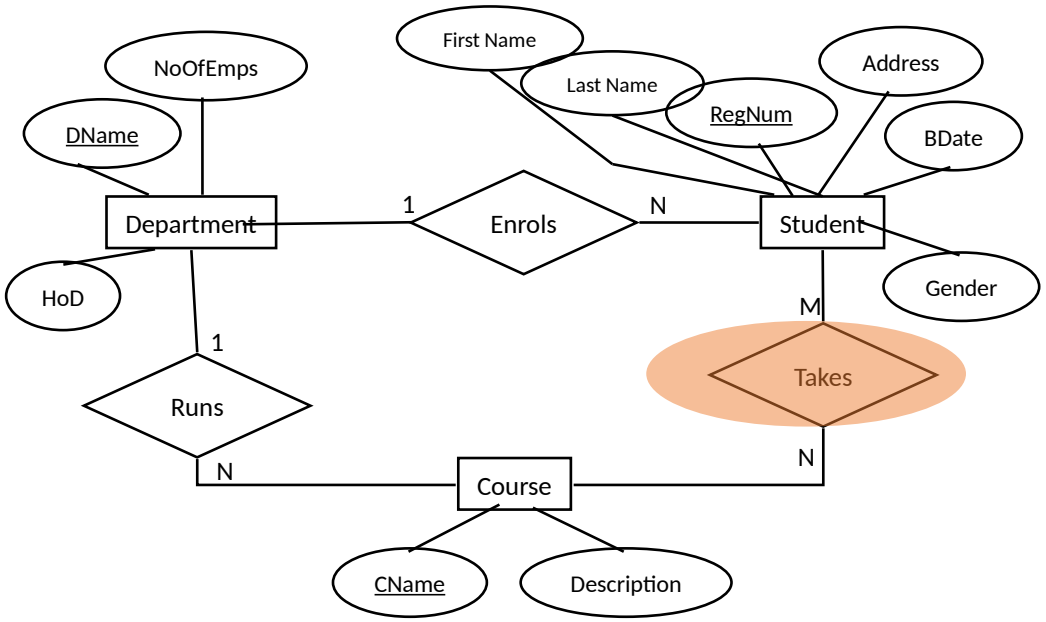
Exercise solutions

- Department(DName TEXT NOT NULL, HoD TEXT, NoOfEmp INTEGER, PRIMARY KEY(DName))
- StudentsEnrol(firstName TEXT, lastName TEXT, RegNumber INTEGER NOT NULL, Address TEXT, BDate TEXT, Gender TEXT, DepName TEXT, PRIMARY KEY (RegNumber), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
- CourseRuns(CName TEXT NOT NULL, Desc TEXT, DepName TEXT, PRIMARY KEY (CName), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
-



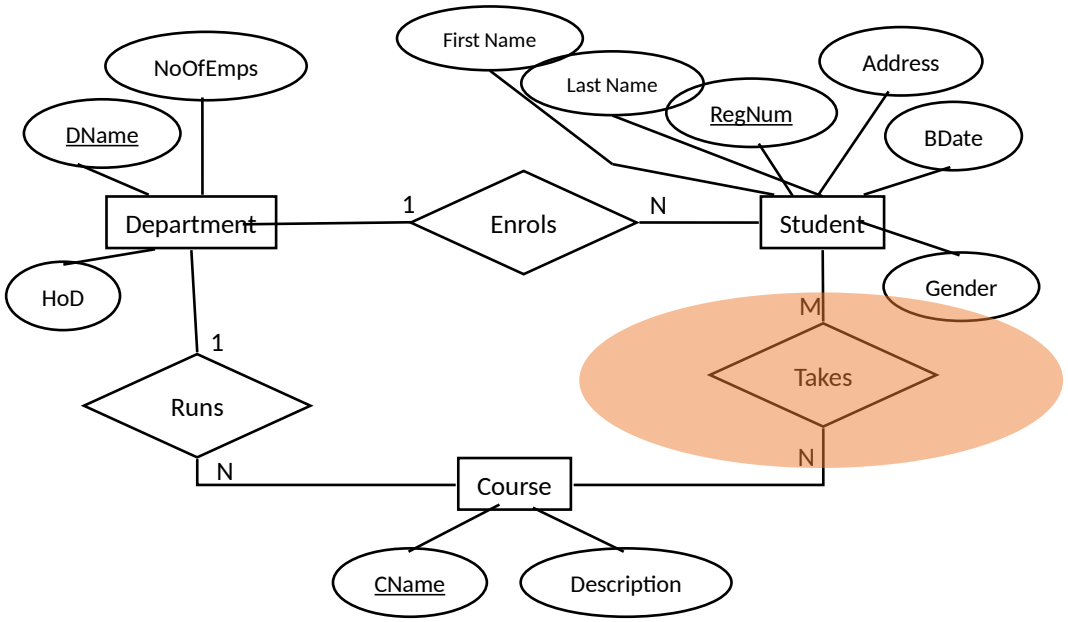
Exercise solutions

- Department(DName TEXT NOT NULL, HoD TEXT, NoOfEmp INTEGER, PRIMARY KEY(DName))
- StudentsEnrol(firstName TEXT, lastName TEXT, RegNumber INTEGER NOT NULL, Address TEXT, BDate TEXT, Gender TEXT, DepName TEXT, PRIMARY KEY (RegNumber), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
- CourseRuns(CName TEXT NOT NULL, Desc TEXT, DepName TEXT, PRIMARY KEY (CName), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
-



Exercise solutions

- Department(DName TEXT NOT NULL, HoD TEXT, NoOfEmp INTEGER, PRIMARY KEY(DName))
- StudentsEnrol(firstName TEXT, lastName TEXT, RegNumber INTEGER NOT NULL, Address TEXT, BDate TEXT, Gender TEXT, DepName TEXT, PRIMARY KEY (RegNumber), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
- CourseRuns(CName TEXT NOT NULL, Desc TEXT, DepName TEXT, PRIMARY KEY (CName), FOREIGN KEY(DepName) REFERENCES Department(DName) ON DELETE SET NULL)
- StTakesCourse(CName TEXT NOT NULL, RegNumber INTEGER NOT NULL, PRIMARY KEY(CNAME,RegNumber), FOREIGN KEY (CName) REFERENCES CourseRuns(CName), ON DELETE SET NULL, FOREIGN KEY (RegNumber) REFERENCES StudentsEnrol(RegNumber), ON DELETE SET NULL)



Summary

- Relational Model is where we translate ER or EER to a mathematical representation called Relations.
-
-
-
-

Summary

- Relational Model is where we translate ER or EER to a mathematical representation called Relations.
- A Relation has two components: An Instance and A Relational Schema (RS).
-
-
-

Summary

- Relational Model is where we translate ER or EER to a mathematical representation called Relations.
- A Relation has two components: An Instance and A Relational Schema (RS).
- While writing the RS, the domain of attributes (to enforce Domain Integrity), the primary key (to enforce Entity Integrity), foreign keys (to enforce Referential Integrity) and Multiplicity and Participation constraints must be provided.
-
-

Summary

- Relational Model is where we translate ER or EER to a mathematical representation called Relations.
- A Relation has two components: An Instance and A Relational Schema (RS).
- While writing the RS, the domain of attributes (to enforce Domain Integrity), the primary key (to enforce Entity Integrity), foreign keys (to enforce Referential Integrity) and Multiplicity and Participation constraints must be provided.
- We saw **UNIQUE**, **CANNOT BE NULL**, **SET NULL/DEFAULT**, **ON DELETE REJECT/CASCADE** to enforce multiplicity and participation constraints.
-

Summary

- Relational Model is where we translate ER or EER to a mathematical representation called Relations.
- A Relation has two components: An Instance and A Relational Schema (RS).
- While writing the RS, the domain of attributes (to enforce Domain Integrity), the primary key (to enforce Entity Integrity), foreign keys (to enforce Referential Integrity) and Multiplicity and Participation constraints must be provided.
- We saw **UNIQUE**, **CANNOT BE NULL**, **SET NULL/DEFAULT**, **ON DELETE REJECT/CASCADE** to enforce multiplicity and participation constraints.
- For a given ERD there can be more than one RS.

Related exam question.

Question 1

Entity relationship diagrams

1.a Please draw the ER diagram for the given description.

> InfoLab21 corp is planning to introduce a new chapter for the Zork Nemesis series. As in the case of *Zork Nemesis 2: The revenge of the fallen*, in the new game *Zork Nemesis 3: Who is Uraz?* a player has a unique playerID, with a Name, and an email address. A player must have at least one character and a character may have at most one associated player. A character has a unique characterName, with Power, Rating, Money, and ExperienceScore. A character may own several inventory items. An inventory item has a unique Item_type, with a Price, and a Wearable attribute. An inventory item belongs to one character; when a character is deleted, the inventory information must be removed from the database.

[5 marks]

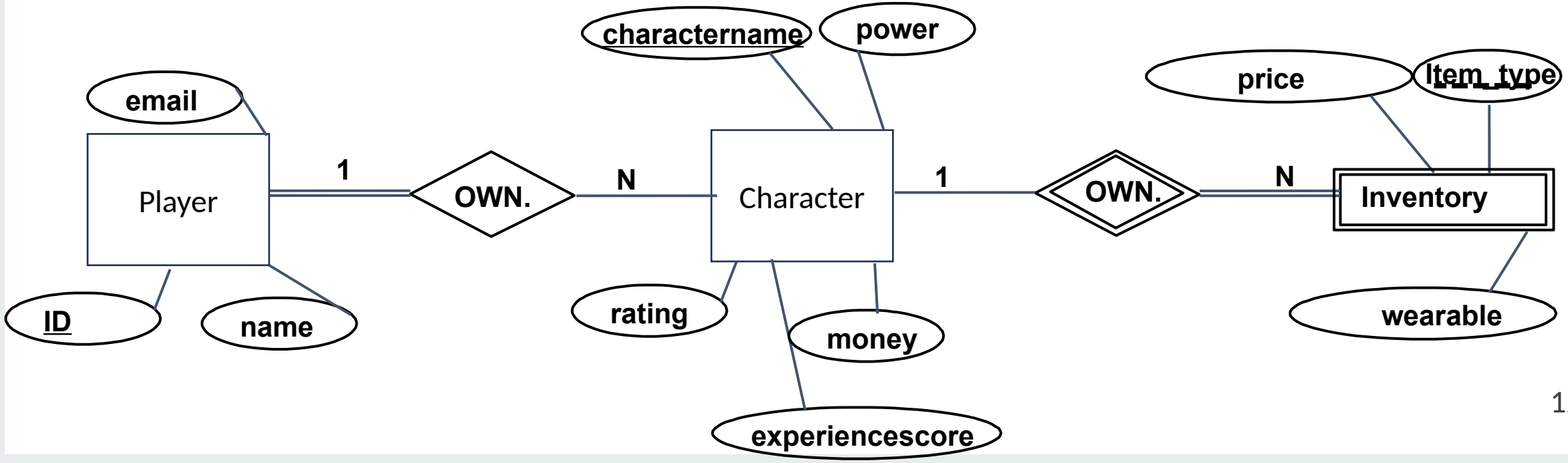
1.b Please provide the Relational Schema and Integrity Constraints for the relations derived from your ER diagram.

[5 marks]

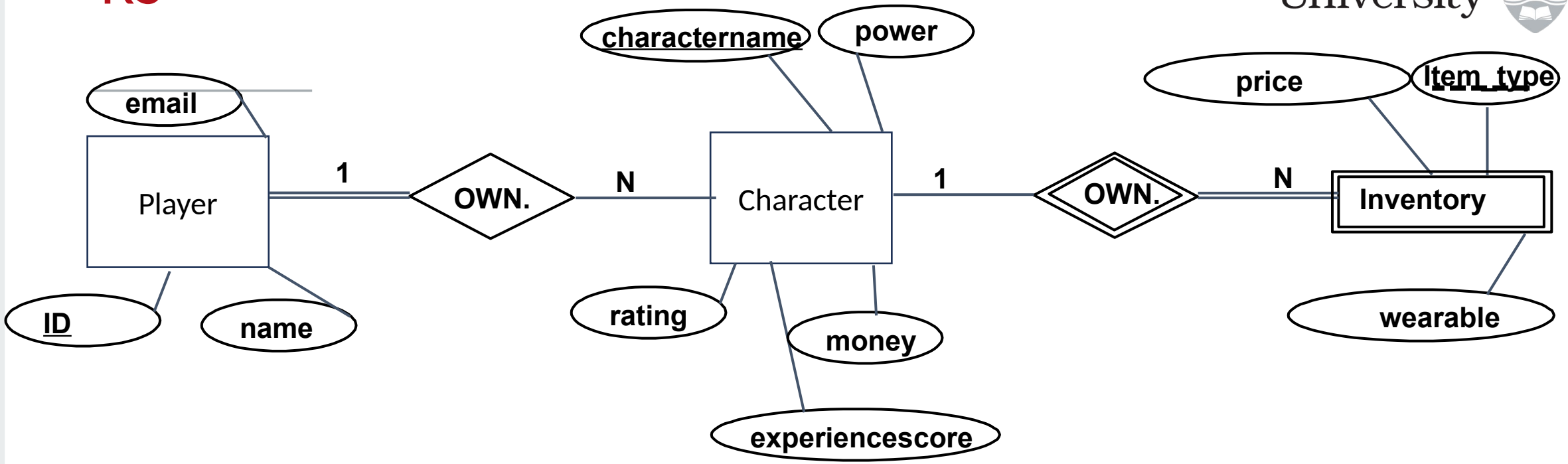
ERD

1.a Please draw the ER diagram for the given description.

InfoLab21 corp is planning to introduce a new chapter for the Zork Nemesis series. As in the case of *Zork Nemesis 2: The revenge of the fallen*, in the new game *Zork Nemesis 3: Who is Uraz?* a player has a unique playerID, with a Name, and an email address. A player must have at least one character and a character may have at most one associated player. A character has a unique characterName, with Power, Rating, Money, and ExperienceScore. A character may own several inventory items. An inventory item has a unique Item_type, with a Price, and a Wearable attribute. An inventory item belongs to one character; when a character is deleted, the inventory information must be removed from the database.

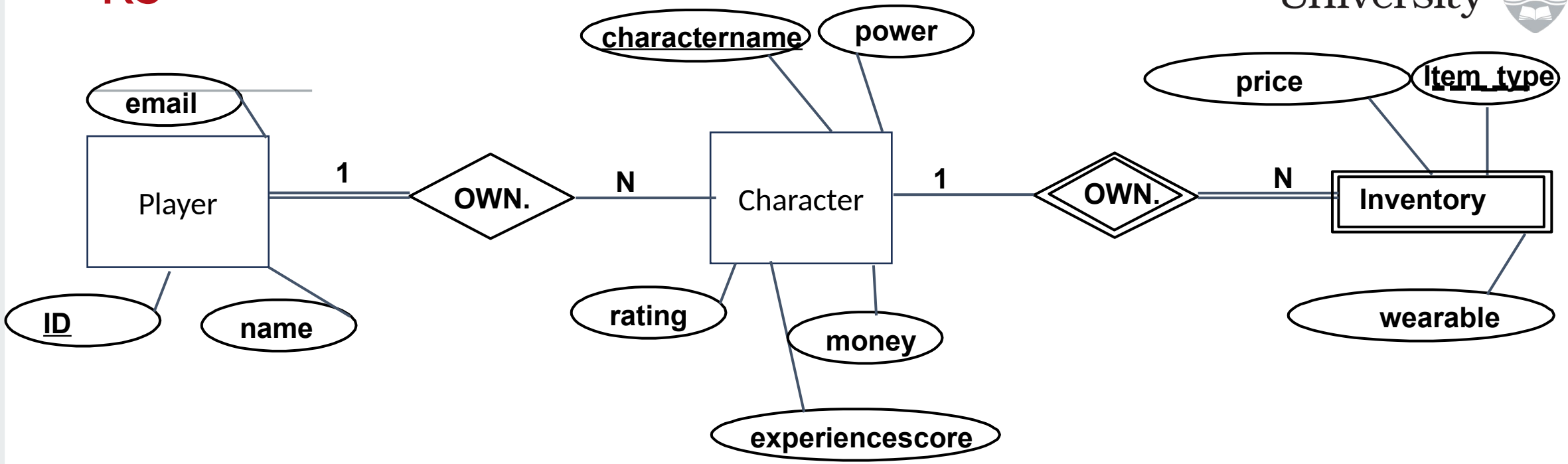


RS



Player_OWN_ Character (name:TEXT, email:TEXT, ID:INT, power:INT, money:INT, rating:INT, experiencescore:INT, charactername:TEXT, PRIMARK KEY:charactername)

RS



Player_OWN_ Character (name:TEXT, email:TEXT, ID:INT, power:INT, money:INT, rating:INT, experiencescore:INT, charactername:TEXT, PRIMARK KEY:charactername)

Inventory_OWN(item_type:INT, price:INT, wearable:BOOLEAN, charactername:TEXT, FOREIGN KEY: charactername REFERENCING Player_OWN_ Character, ON DELETE CASCADE, item_type is UNIQUE, PRIMARY KEY{item_type,charactername})