



Let's start



# SCC.201 Database Management Systems

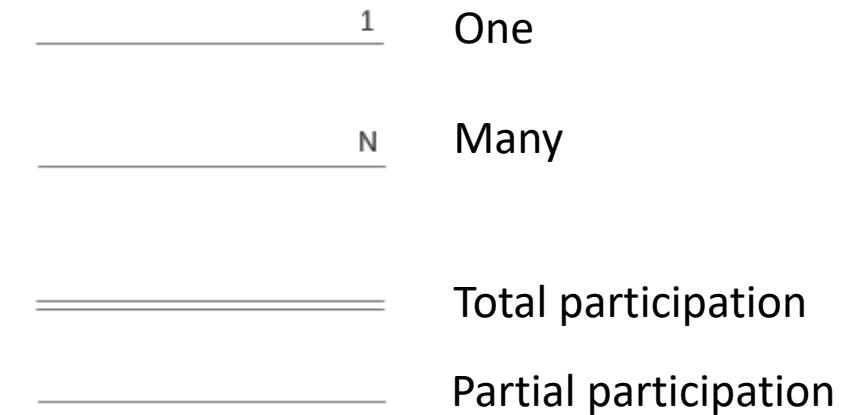
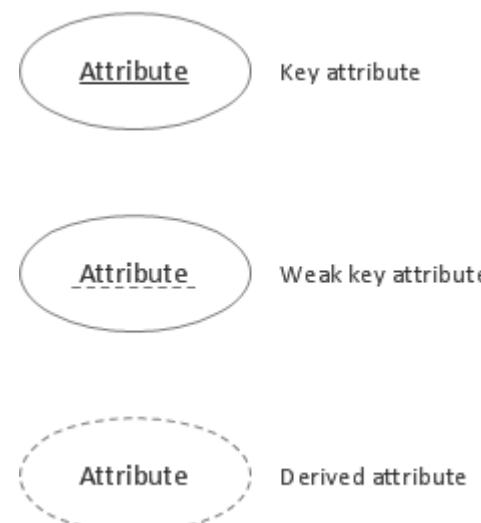
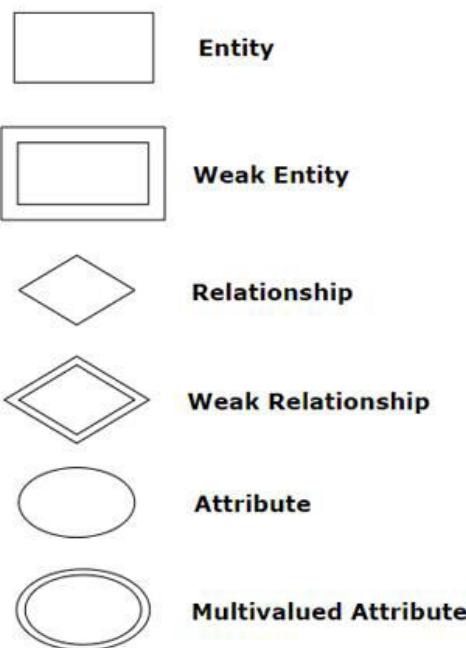
---

2023 - Week 2 – Relational Database.  
Uraz C Turker & Ricki Boswell

# What have we learnt so far?



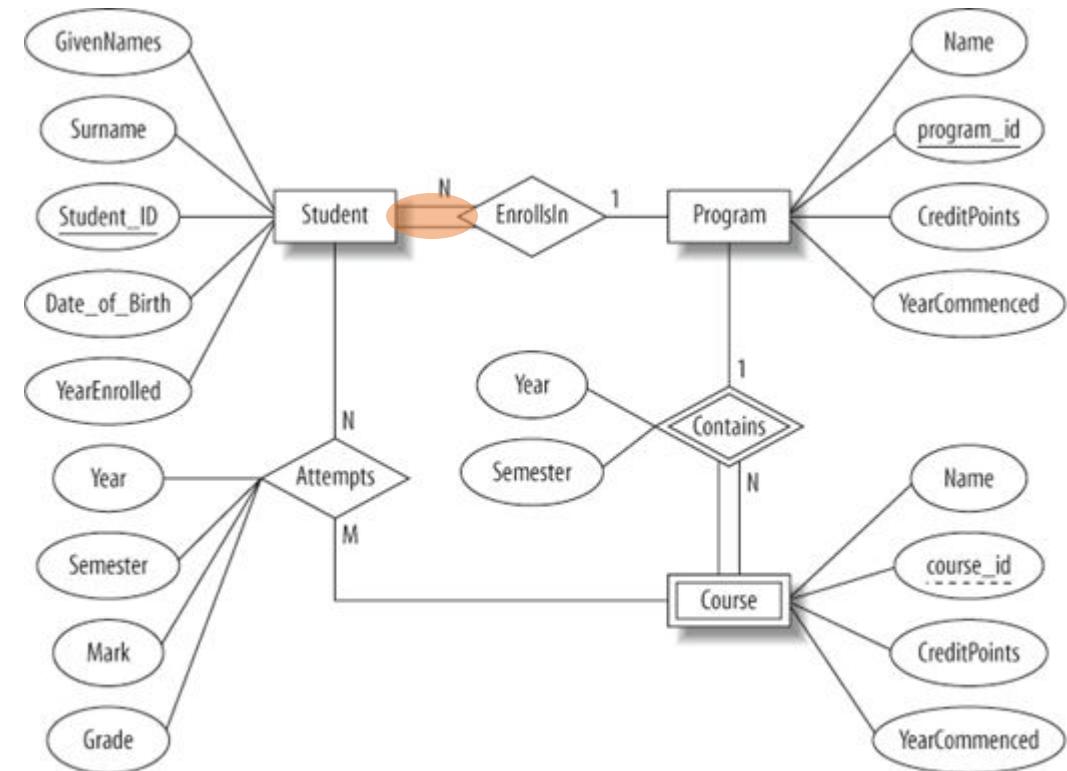
- We learn key ER concepts which allows us to design relational databases.



# What have we learnt so far?



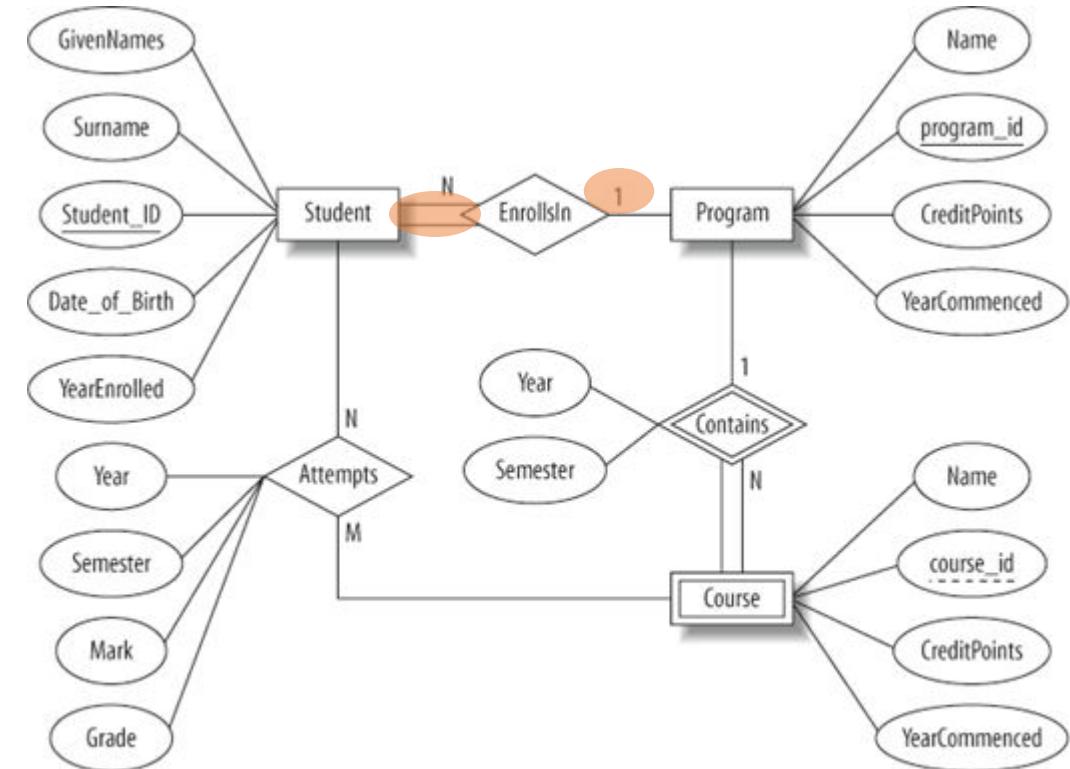
- A student **must** enrol in program.



# What have we learnt so far?



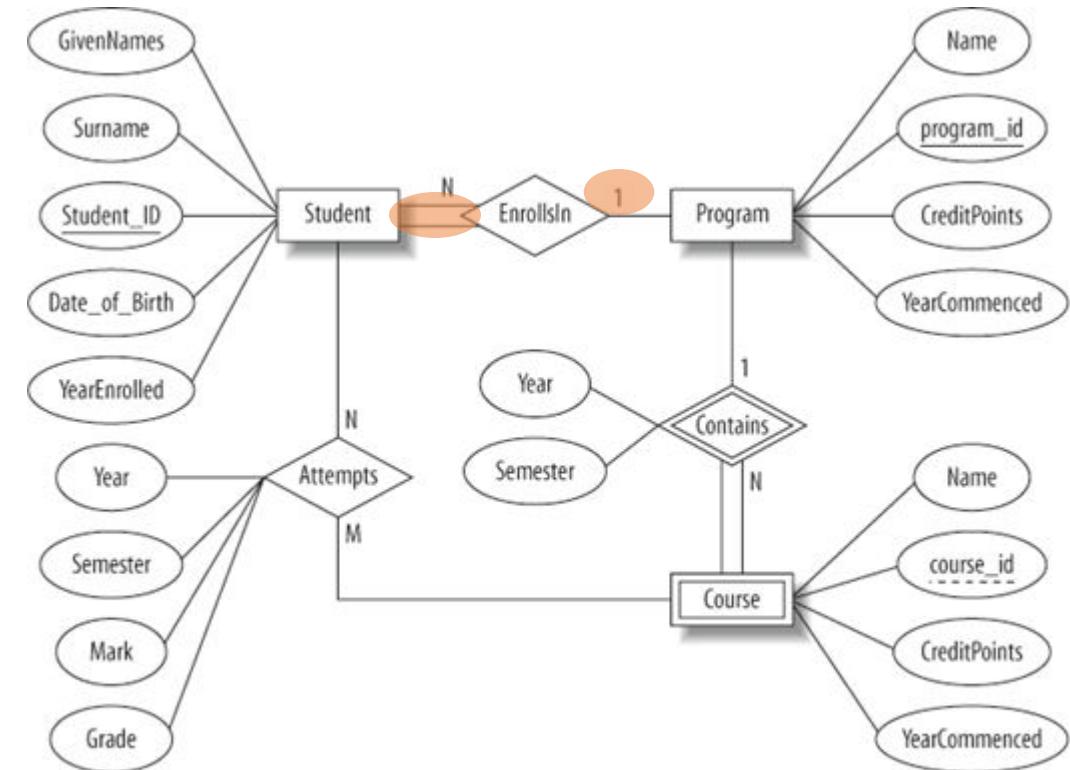
- A student **must** enrol in program.



# What have we learnt so far?



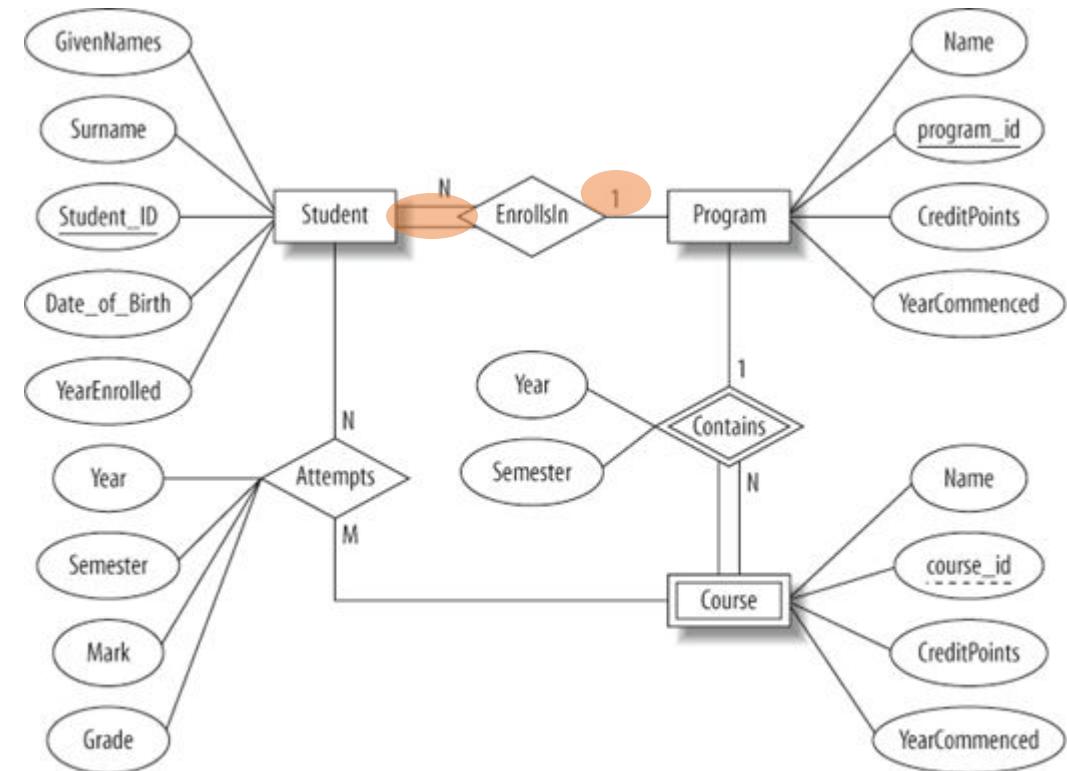
- A student **must** enrol in **one** program.



# What have we learnt so far?



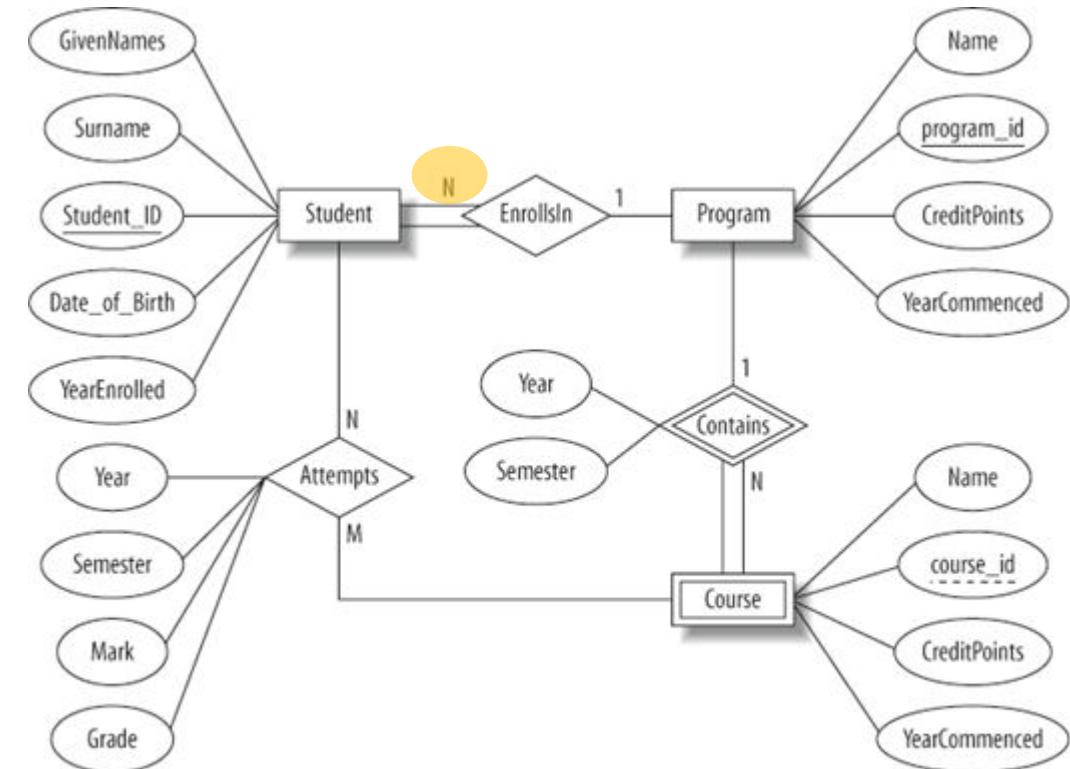
- A student **must** enrol in a program.
  - It has GivenNames, Surname, StudentID, Date\_of\_Birth, YearEnrolled attributes, where StudentID is a primary key.



# What have we learnt so far?



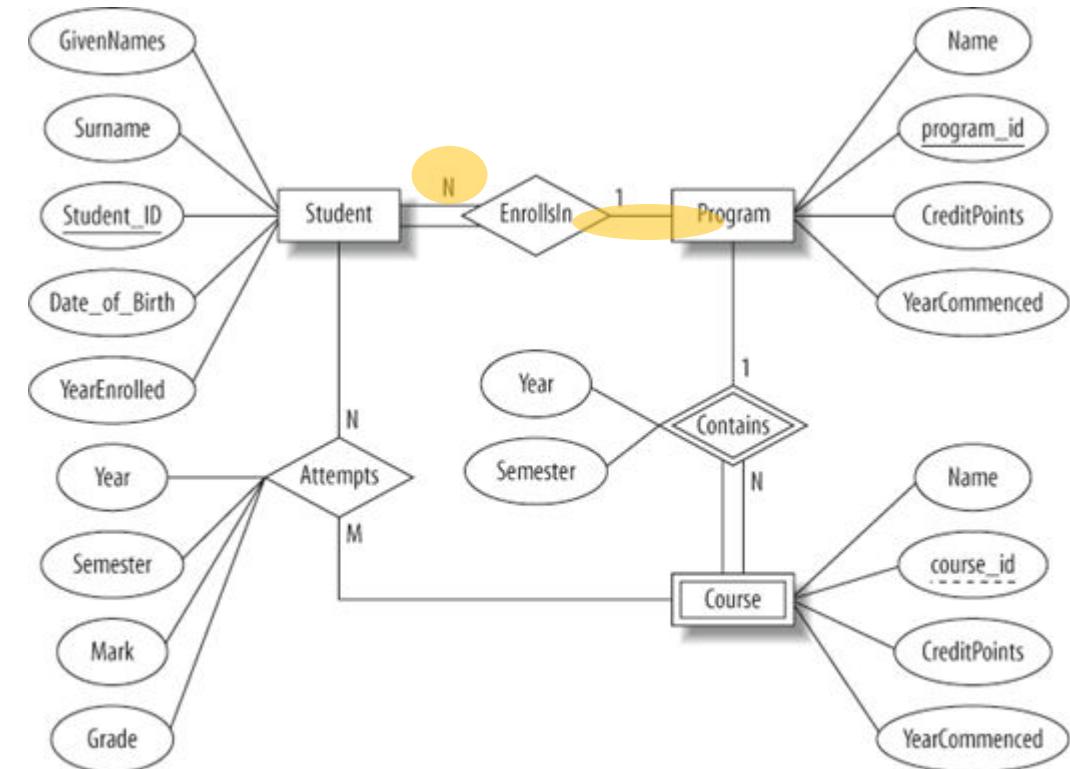
- Program **have many** students.



# What have we learnt so far?



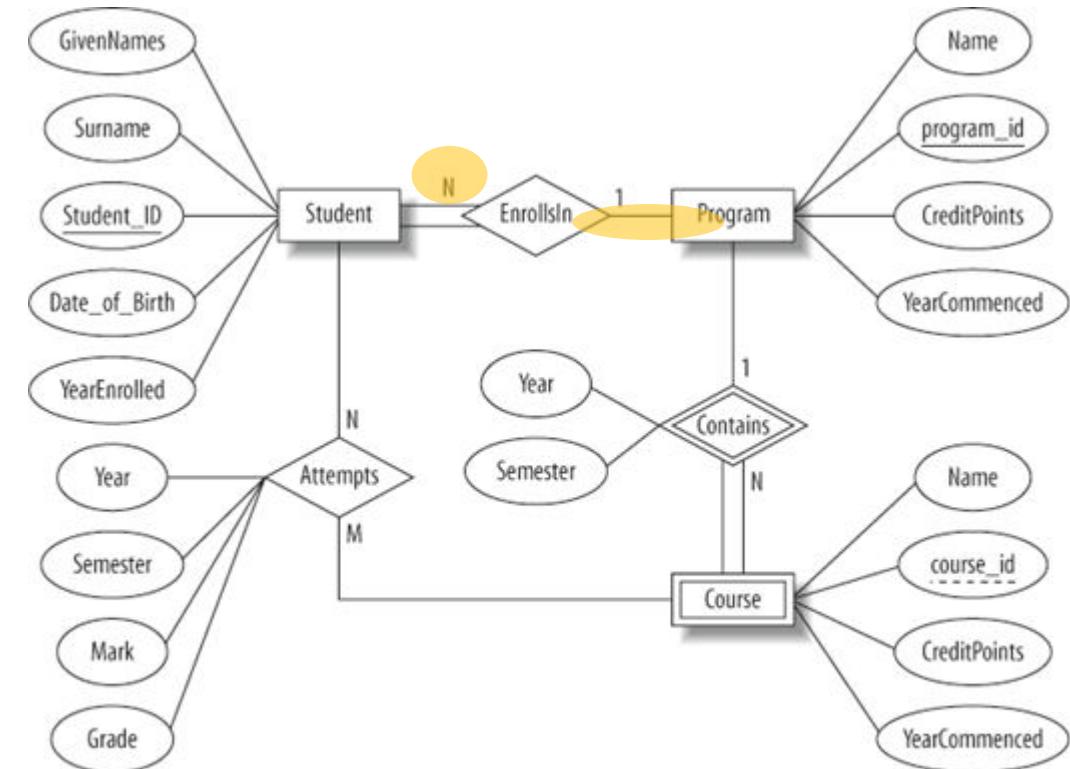
- Program **have many** students.



# What have we learnt so far?



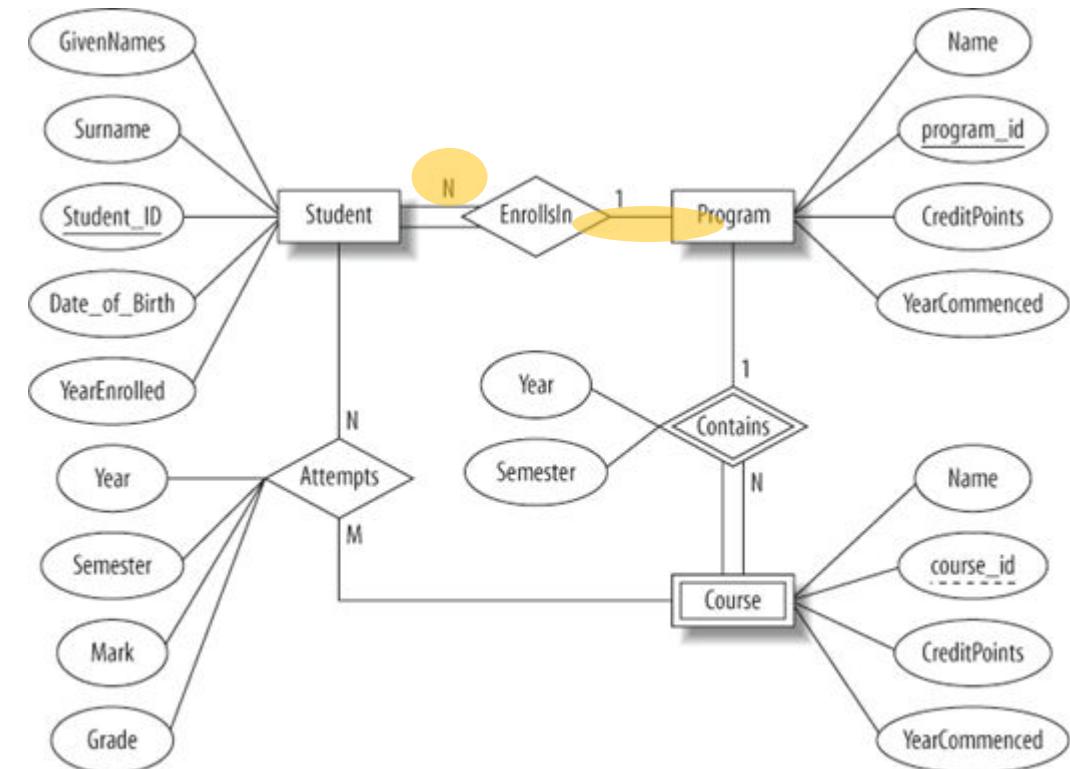
- Program **may have many** students.



# What have we learnt so far?



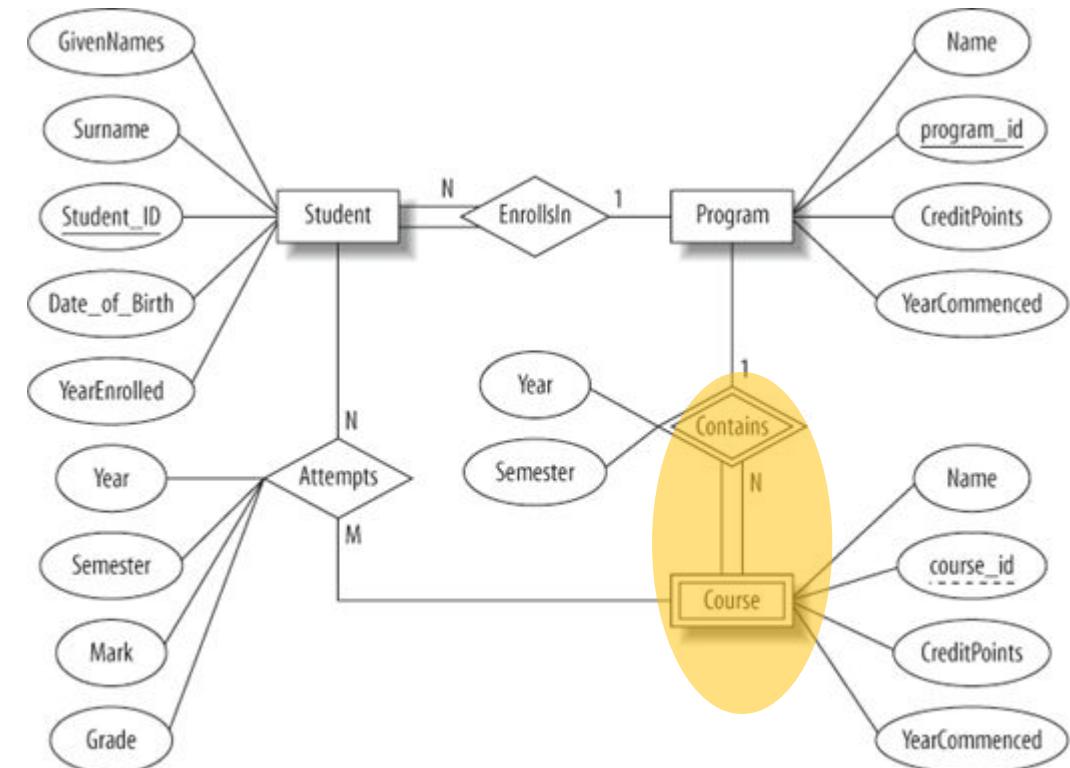
- Program **has many** students.
  - It has name, program\_id, CreditPoints, YearCommenced where program\_id is a primary key.



# What have we learnt so far?



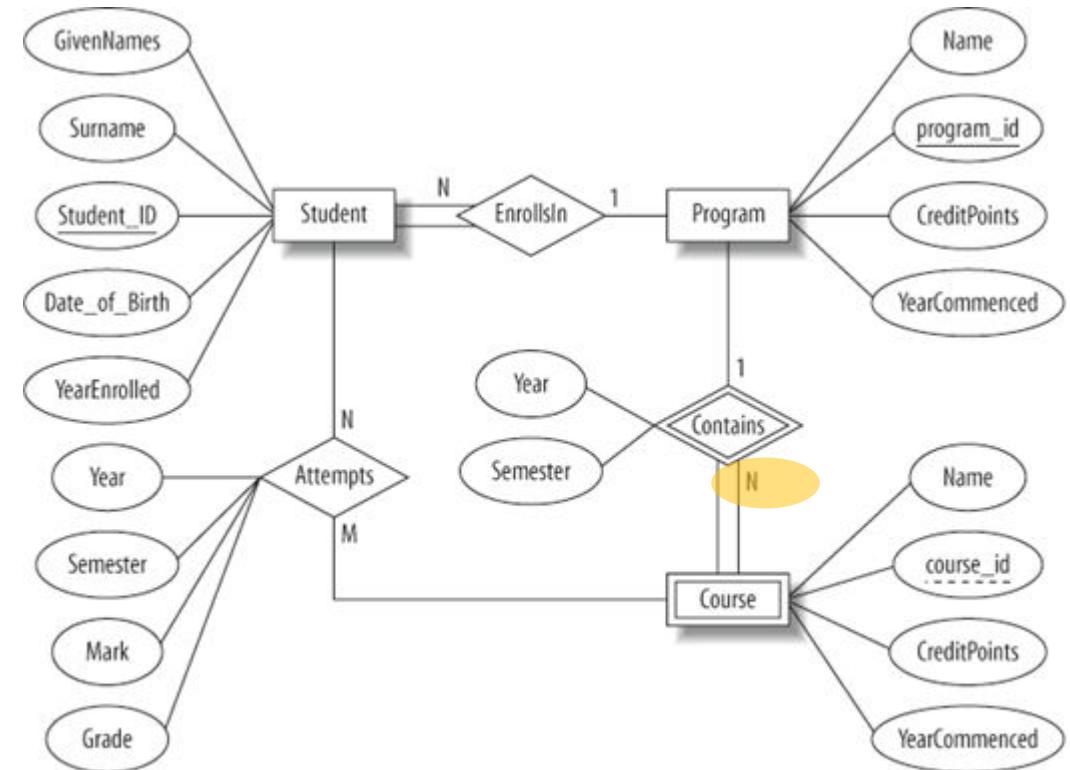
- A **course** without a program has **no definition/meaning**. (weak entity)



# What have we learnt so far?



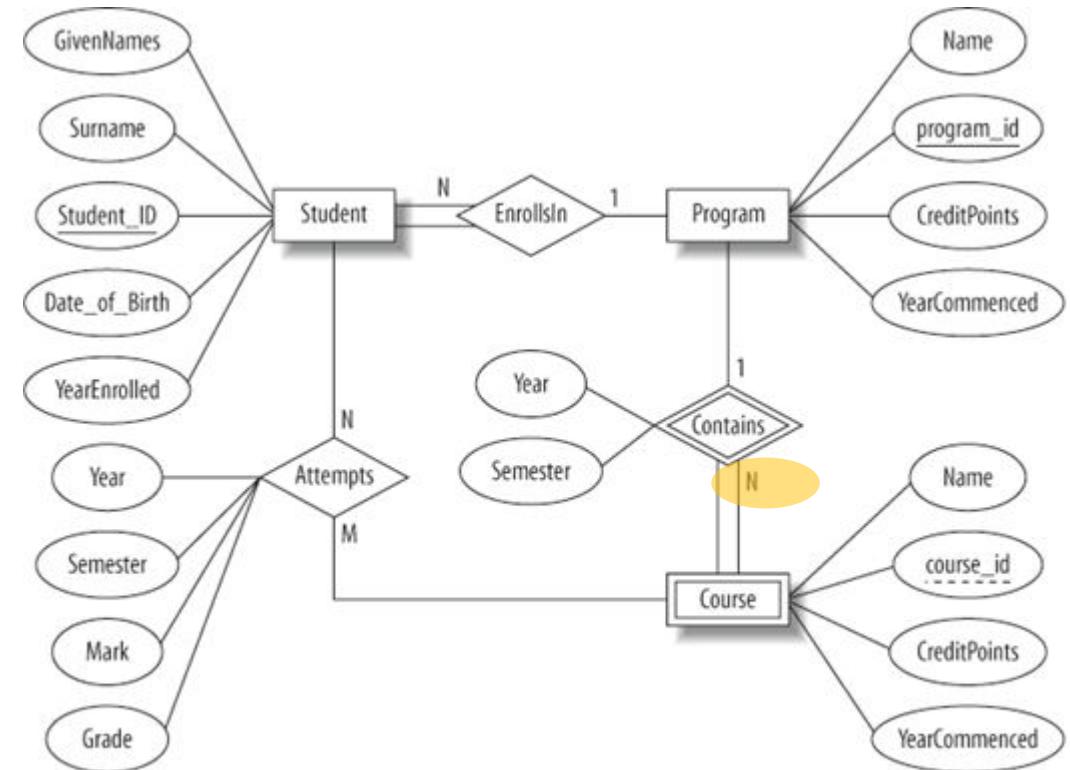
- Program have courses



# What have we learnt so far?



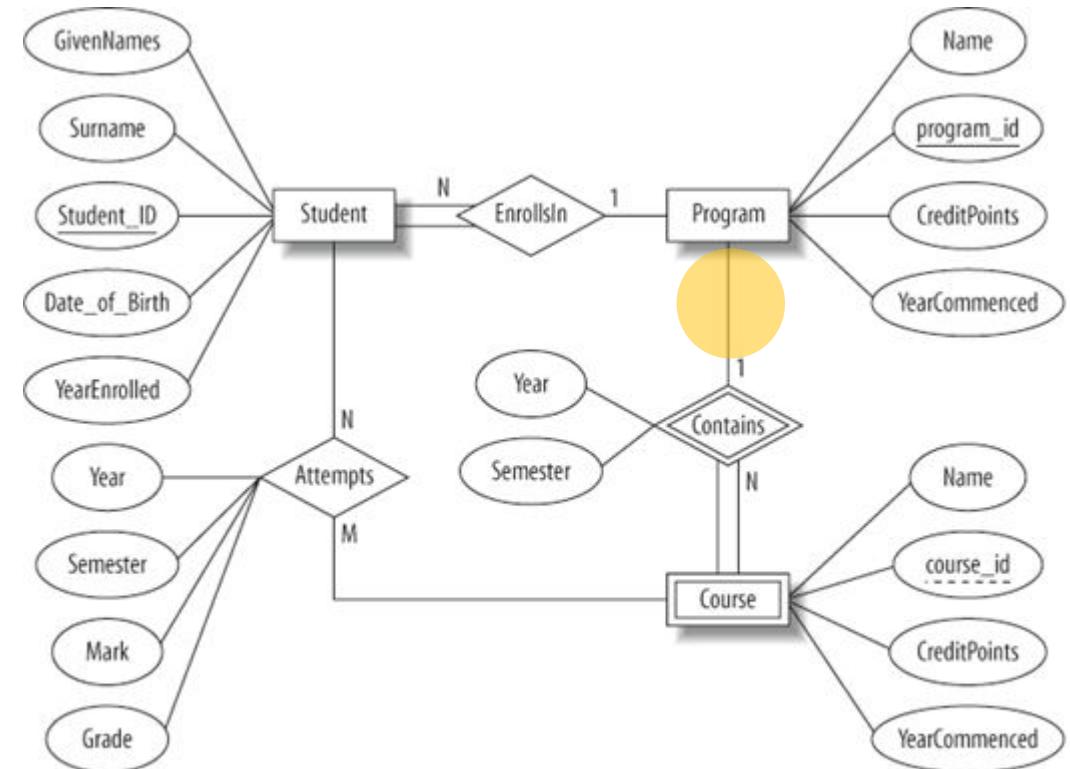
- Program courses have many



# What have we learnt so far?



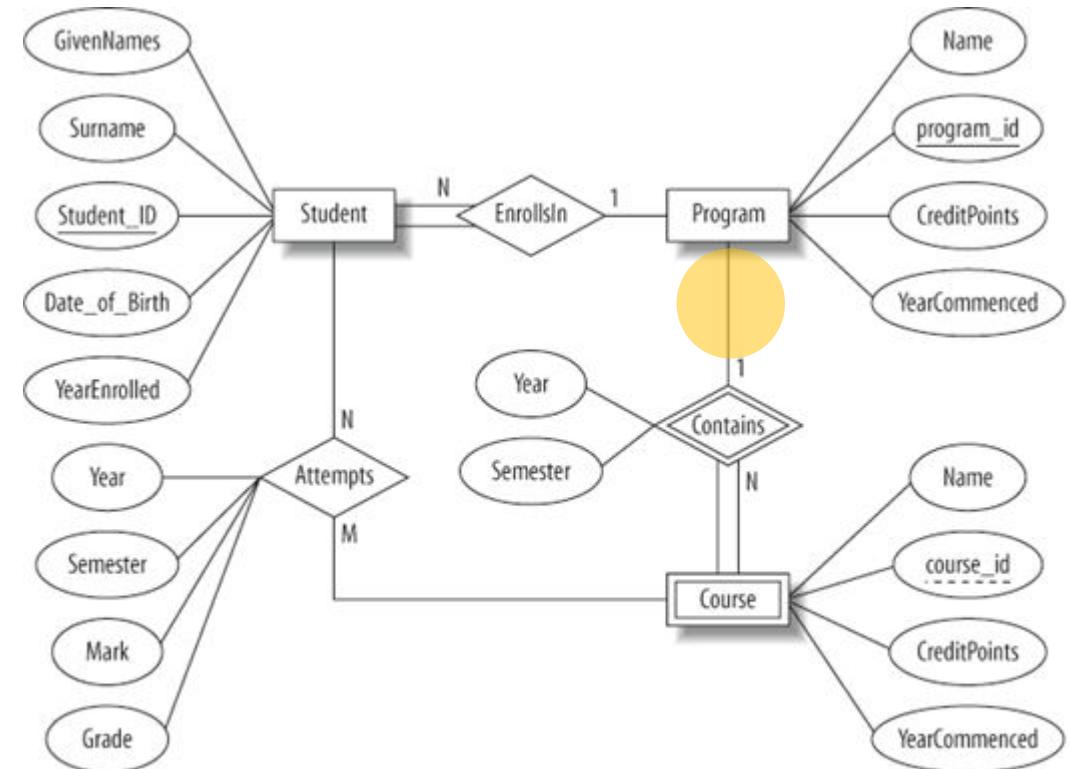
- Program courses have many



# What have we learnt so far?



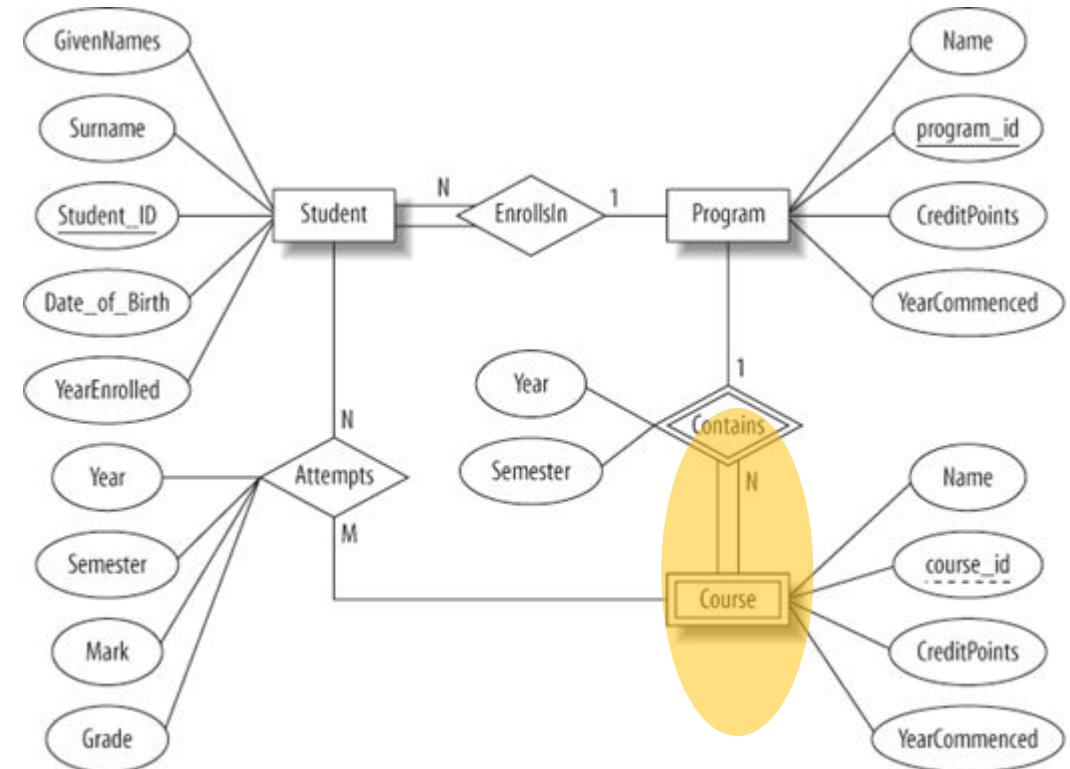
- Program **may** have **many** courses



# What have we learnt so far?

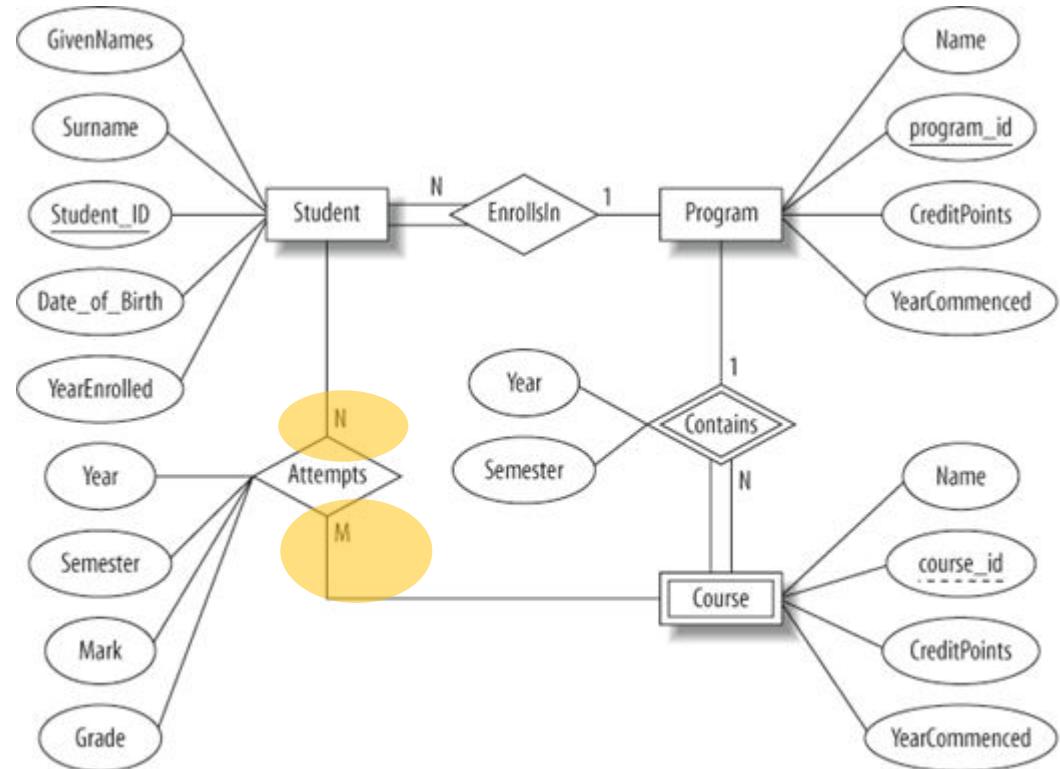
---

- A **course** without a program has **no definition/meaning**. (weak entity)
  - It has Name, course\_id, CreditPoints, YearCommenced where course\_id is the weak key.



# What have we learnt so far?

- A student **may attempt many** courses.
- A course **may have many** students.



# This lecture

---

- We will start learning the Relational Model.
  - Schema and definition.
  - Integrity constraints.
  - Key constraints
  - Referential Integrity



## Curriculum Design: Outline Syllabus

This module builds upon knowledge gained in Part I by providing a theoretical background to the design, implementation and use of database management systems, both for data designers and application developers. It takes into account all relevant aspects related to information security in the design, development and use of database systems. The course consists of a number of related sections, which range from single lectures to multi-lecture streams, depending on the required depth of coverage. The sections are as follows.

**Introduction**: we begin with a brief history of how the need for database management systems (DBMS) grew over time and how they are applied in day to day scenarios.

**Database Design**: before making use of a DBMS, we must capture our requirements : what data do we actually wish to model? We make use of the Extended Entity-Relationship (EER) model which is both a technique and a notation for designing the data in a DBMS independent way.

**The Relational Model**: now the de-facto standard for DBMS, this was a revolutionary step taken in 1970. We extensively examine the Model, looking at relational database systems, the model itself and the normalisation process; the relational algebra (the mathematical theory that underpins the model), the three schema architecture and schema definition in SQL. Finally, we look at how we can map the EER model into an equivalent Relational Model. The resultant database is then examined in terms of access rights and privileges.

**A (re)introduction to SQL**: SQL is the de-facto standard for DBMS query languages. We look at both the DDL (data definition language) and DML (data manipulation language). We introduce the use of views, a powerful mechanism for providing privacy and security. We look at the Discretionary Access Control (DAC) features that allow the granting and withholding of access rights and privileges.

**Accessing relational DBMS via Java**: we explore the facilities of the JDBC and show how we can write applications in Java which connect with a relational DBMS (in practice, MySQL).

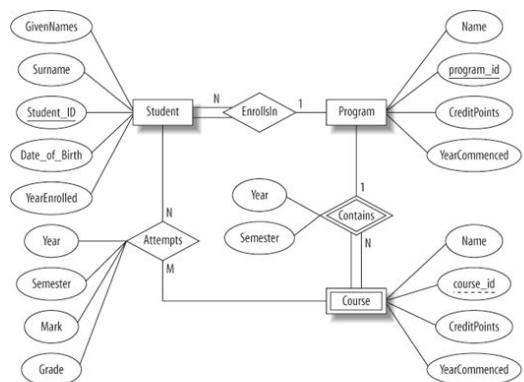
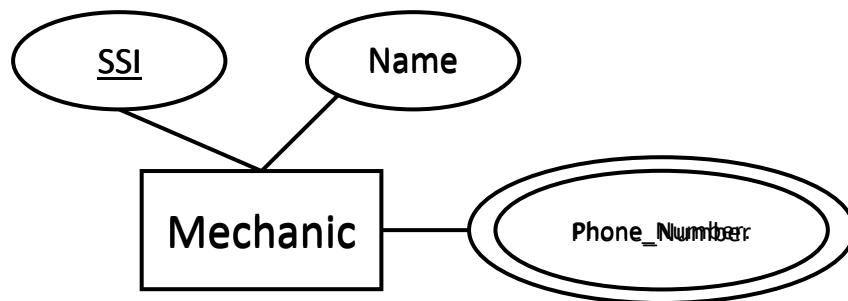
**The Physical Model**: as Computer Scientists, our students need an awareness of the techniques that allow rapid access to stored data. In this section, we examine the physical data organisation and associated access methods. We show under what circumstances the organisations can be applied, and we look at how queries can be optimised.

**Transaction processing and concurrency control**: a huge part of DBMS in practice is the need to support transactions and concurrency, allowing huge numbers of users to access the DBMS at any one time while still ensuring the consistency of the data. This stream examines the problems and solutions in depth.

That is, we will start learning correspondence between ERD and Relational database.



- ER diagram
- Relation.



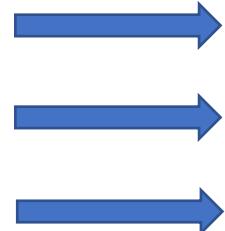
SSI	Name	Phone_Number
87542702	Tom	75315567, 75315264
68201937	Uraz	75335521, 75334567
23139827	Nick	75315544, 75315237



# Relational Database: Definitions

---

- You can think of a **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/Tuples

# Relational Database: Definitions

---

- *Relation*: made up of 2 parts:
  1. *Instance*: a *table* with rows and columns.  
*#Rows = cardinality, #fields/attributes = degree / arity.*
  2. *Relation Schema*: specifies the relation's name and **type (domain)** of each **column**.
    - E.G. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).

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

# Relational Database: Definitions

After this lecture, we will not use derived, multivalued attributes in a relation.

- We can represent a relation schema using mathematical notation

- $S(A_1, A_2, A_3, \dots, A_n)$  Where  $S$  is the schema's name,  $A$  is an attribute.
  - E.g.  $\text{Students}(sid:\text{string}, name:\text{string}, login:\text{string}, age:\text{integer}, gpa:\text{real})$ .
- Further, we can use  $\text{dom}(A_i)$  to denote the domain/type of an attribute.
  - E.g.  $\text{dom}(\text{Students}(name)) = \text{string}$ .
- Formally: a relation (its state, instantaneous state, etc.) of a relation schema  $S(A_1, A_2, A_3, \dots, A_n)$  is a set of  $m$ -tuples  $r(S) = \{t_1, t_2, t_3, \dots, t_m\}$  where each tuple  $t_i = (v_1, v_2, v_3, \dots, v_n)$  has  $n$  values where
  - $n$  is the arity.
- We also use  $t_i[A_j]$  to denote the value of the  $j$ th attribute of tuple  $i$ .
- $t_2[\text{name}] = t_3[\text{name}] = \text{Shero}$

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/Tuples

	sid	name	login	age	gpa
$t_1$	V1	V2	V3	V4	V5
$t_2$	V1	V2	V3	V4	V5
$t_3$	V1	V2	V3	V4	V5

# Relational model (Syntax vs Semantics)

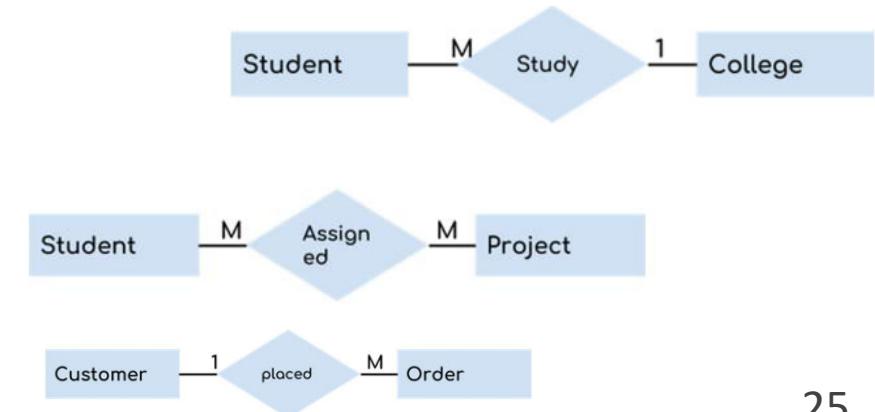


- 
- What would happen if I erase a key value?
  - What would happen if I entered a row
    - <Mercedes, London,3341E42,300>
  - Data within a relation must obey some rules!
  - Relational databases are built upon logical rules (constraints) set by
    - Real-world rules, designers' choices, functions, etc.
  - These rules establish what we call the **integrity** of the data.

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

# Integrity Constraints for relational databases.

- **Integrity of data:** it is the state of data in which data obeys the constraints set by DBA.
- Domain constraints.
  - Values in tuples should obey types of attributes.
- Key constraints.
  - Keys of a relation must be unique, non-redundant, and not Null (entity integrity constraint).
  - Referential integrity.
    - If two relational schemas are related to each other,
      - The relation must be made by using keys,
      - The DBMS must preserve this relation.
- User-defined constraints.



# Definitions : Integrity Constraints (ICs)



- **Integrity constraint (IC):** a condition that must be true for *any* database instance.
  - ICs are **specified by DB designer/Admin** (using DDL) when a schema is defined.
  - ICs are **checked (by DBMS automatically)** when relations are modified.
- 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!
- **Domain constraint:** In any database instance, a value of the attribute must be an element of the attribute's domain.

string	string	string	integer	real
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	X

Domain constraint breached  
(gpa is a real value, Na is a string)

# Key constraints: lets recall keys.

-----

- A *key attribute* can **distinguish** a row/tuple (entity).
- A *candidate key* is a **set of key attributes**.
- A *composite key* is a set of attributes that can **distinguish** a row/tuple (entity).

Model	Weight	ChassisN	Max_Speed
BMW 3.21	1400	12h37	200
Toyota_Corolla	1300	84t34	200
Hyundai E.GLS	1400	43j5h2	210

{Model, ChassisN} are candidate keys

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

SSI is a key.

Name	Surname	Team	Age
Ben	Pogba	Juv.	20
Ben	White	Ars.	20
Diogo	White	MU	21

Name, Surname is a composite key.

# Key constraints: another type of key.



- A set of fields/attributes is a key (from now on, we may omit the term composite) for a relation if :
  - 1. No two distinct tuples/rows can have the same values in all key fields, and
  - 2. This is not true for any subset of the key.
  - What would happen part 2 is false?

What does this mean?

Name	Surname	Team	Age
Ben	Pogba	Juv.	20
Ben	White	Ars.	20
Diogo	White	MU	21

{Name,Surname} is a primary key as neither Name, nor Surname is a key.

# Key constraints: another type of key.



- A set of fields/attributes is a *key* (from now on, we may omit the term composite) for a relation if :
  - 1. No two distinct tuples/rows can have the same values in all key fields, and
  - 2. This is not true for any subset of the key.
  - Part 2 false? A *superkey*.
- E.g., *sid* is a key for Students.
- Find a superkey.
- The set  $\{sid, gpa\}$  is a superkey.
- What else?
- Actually, all the sets of attributes that contain a key is a superkey.
- $\{sid, age\}$ ,  $\{sid, age, gpa\}$ ,  $\{sid, name, age, login, gpa\}$ ,...
- Every relation has at least one superkey (all attributes!)

Don't just look at the data, use knowledge of the real world – What is going to state unique!

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

A composite key (or a key) is also a superkey! (minimal superkey).

# Enforcing key constraints.

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

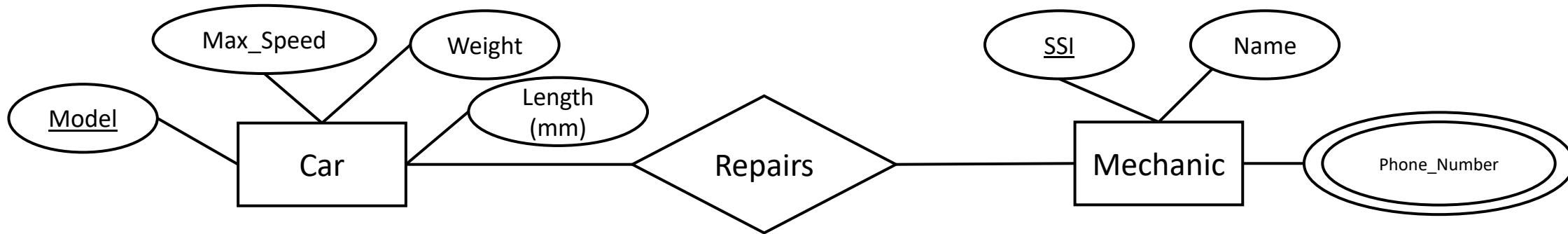
- Once a key is set by DB Admin, the DBMS must inspect every modification on the data w.r.t key value.
- Entity integrity constraint:** no key value can be NULL.
- So it is DB Admin's task (using DDL) to set the primary key for the data.
  - DBMS has built-in functions to protect key constraints, to activate those functions DBA must identify them using DDL (next week!).

# Referential integrity: How about relationships?

---



- How do we set the integrity of relationships?



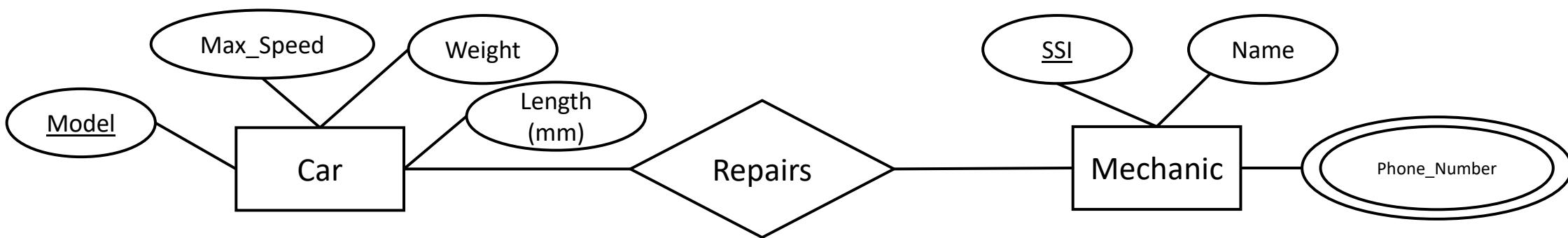
# Referential integrity: Another key! Foreign keys

DBMS: What is this “Model” all about?

- Foreign keys:

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

Model	<u>SSI</u>	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: Foreign keys

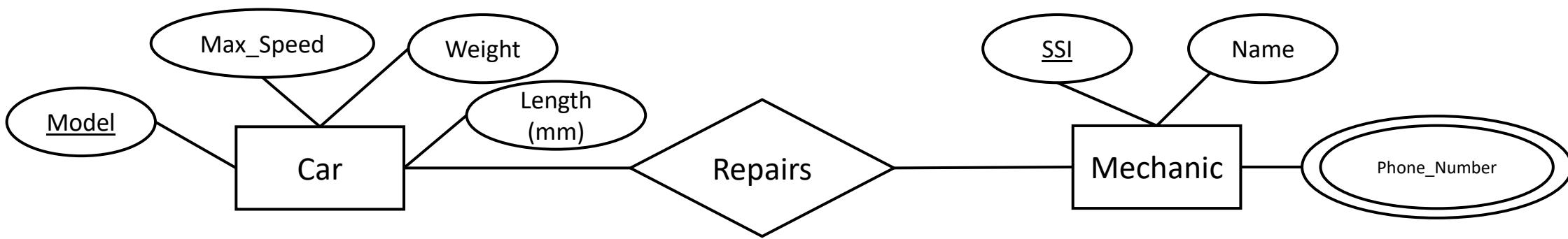
**DBMS:** What is this “Model” all about?

**DB-Admin:** It is the primary key of a related relation.

- Foreign keys:

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

<u>Model</u>	<u>SSI</u>	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: Foreign keys

**DBMS:** What is this “Model” all about?

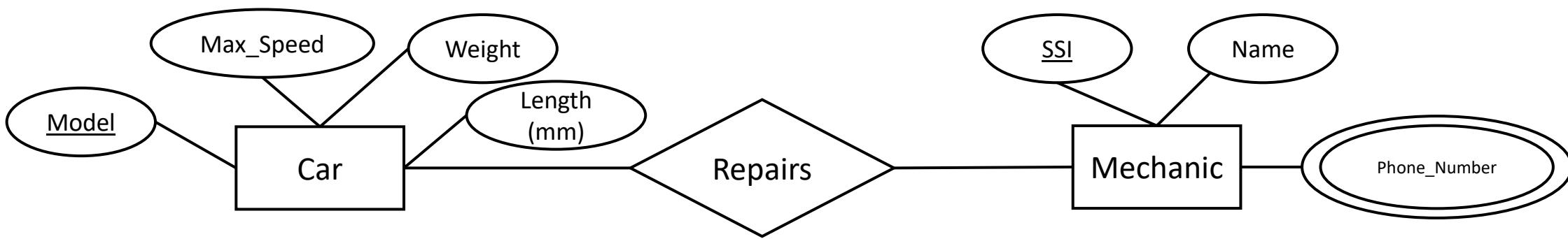
**DB-Admin:** It is the primary key of a related relation.

**DBMS:** Ahaa!! It is FOREIGN KEY.

- Foreign keys:

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

Model	<u>SSI</u>	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: Foreign keys

- Foreign keys:

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

**DBMS:** What is this “Model” all about?

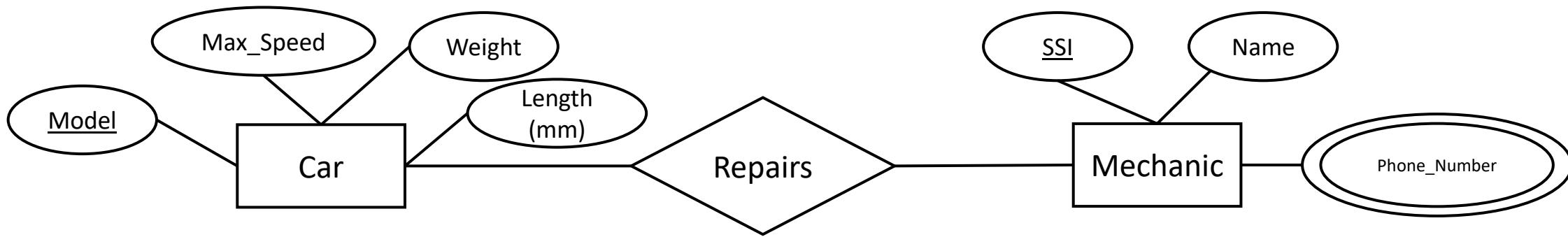
**DB-Admin:** It is the primary key of a related relation.

**DBMS:** Ahaa!! It is a FOREIGN KEY.

**DMBS:** But what is the referencing relation?

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

## REFERENCING RELATION



# Referential integrity: Foreign keys

- Foreign keys:

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

**DBMS:** What is this “Model” all about?

**DB-Admin:** It is the primary key of a related relation.

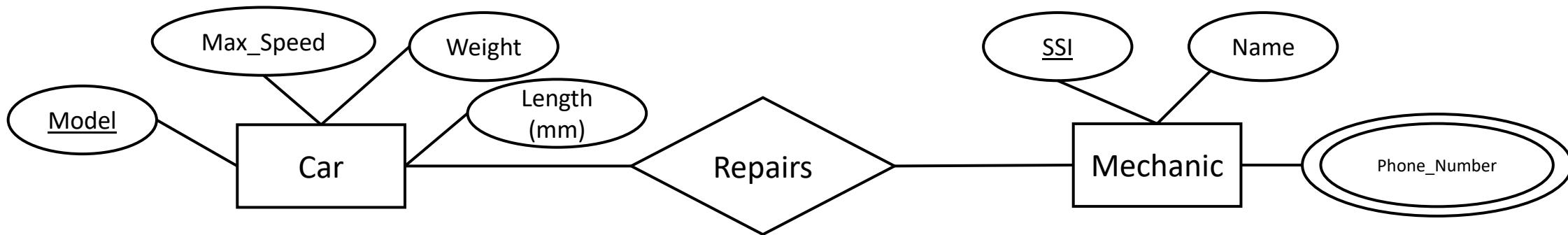
**DBMS:** Ahaa!! It is FOREIGN KEY.

**DMBS:** But what is the referencing relation?

**DB-Admin:** It is from the “Car” relation (using DDL) .

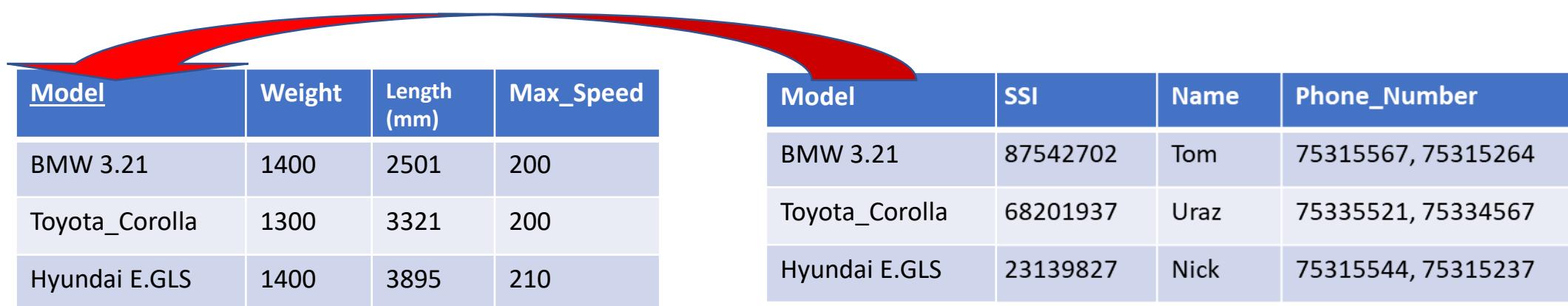
<u>Model</u>	<u>SSI</u>	Name	Phone_Number
BMW 3.21	87542702	Tom	75315567, 75315264
Toyota_Corolla	68201937	Uraz	75335521, 75334567
Hyundai E.GLS	23139827	Nick	75315544, 75315237

## REFERENCING RELATION



# Referential integrity: Foreign keys

- **Foreign key:** Set of fields in one relation that is used to ‘refer’ to a tuple/row in another relation. (**Must correspond to the primary key of the referring relation.**) Like a ‘logical pointer’.

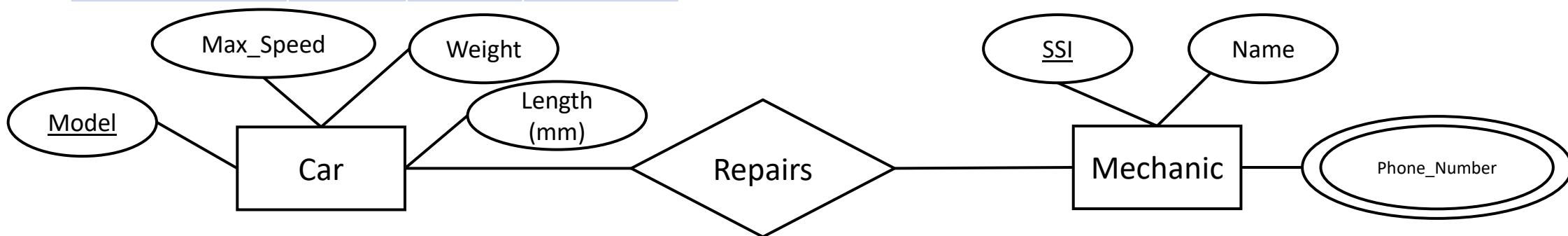


The diagram illustrates two tables and their relationship. A red curved arrow points from the 'Model' column in the first table to the 'Model' column in the second table, indicating a foreign key relationship.

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



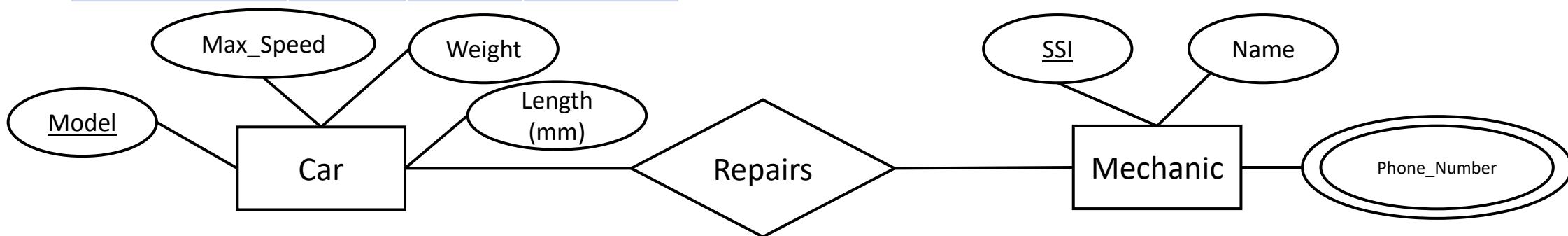
# Referential integrity: Foreign keys

- Foreign keys: Set of fields in one relation that is used to ‘refer’ to a tuple/row in another relation. (Must correspond to the primary key of the referring relation.) Like a ‘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
BMW 3.21	87542702	Tom	75315567, 75315264
Toyota_Corolla	68201937	Uraz	75335521, 75334567
Hyundai E.GLS	23139827	Nick	75315544, 75315237

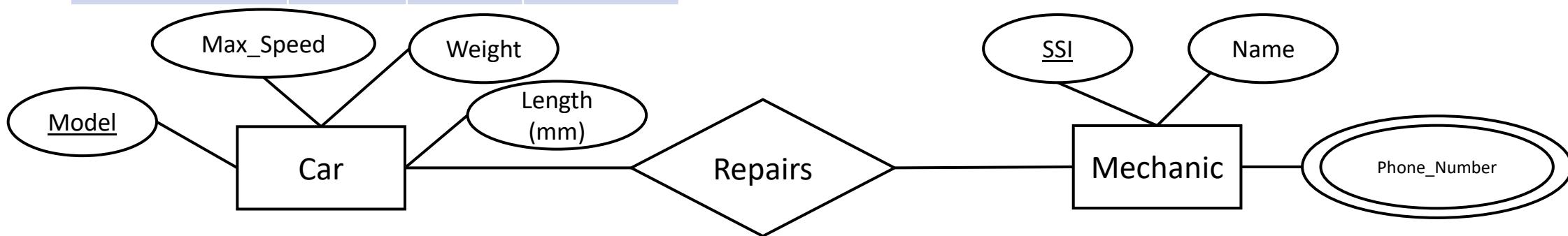


# Referential integrity: Foreign keys

- Foreign keys: Set of fields in one relation that is used to ‘refer’ to a tuple/row in another relation. (Must correspond to the primary key of the referring relation.) Like a ‘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
BMW 3.21	87542702	Tom	75315567, 75315264
Toyota_Corolla	68201937	Uraz	75335521, 75334567
Hyundai E.GLS	23139827	Nick	75315544, 75315237



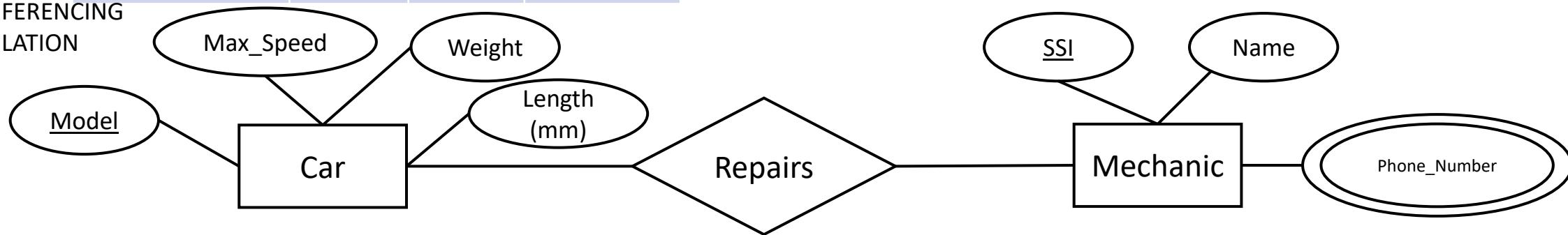
# Referential integrity: Foreign keys

- Foreign key must:
  - Have the same name and domain/type as the referencing relation.
  - Having 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

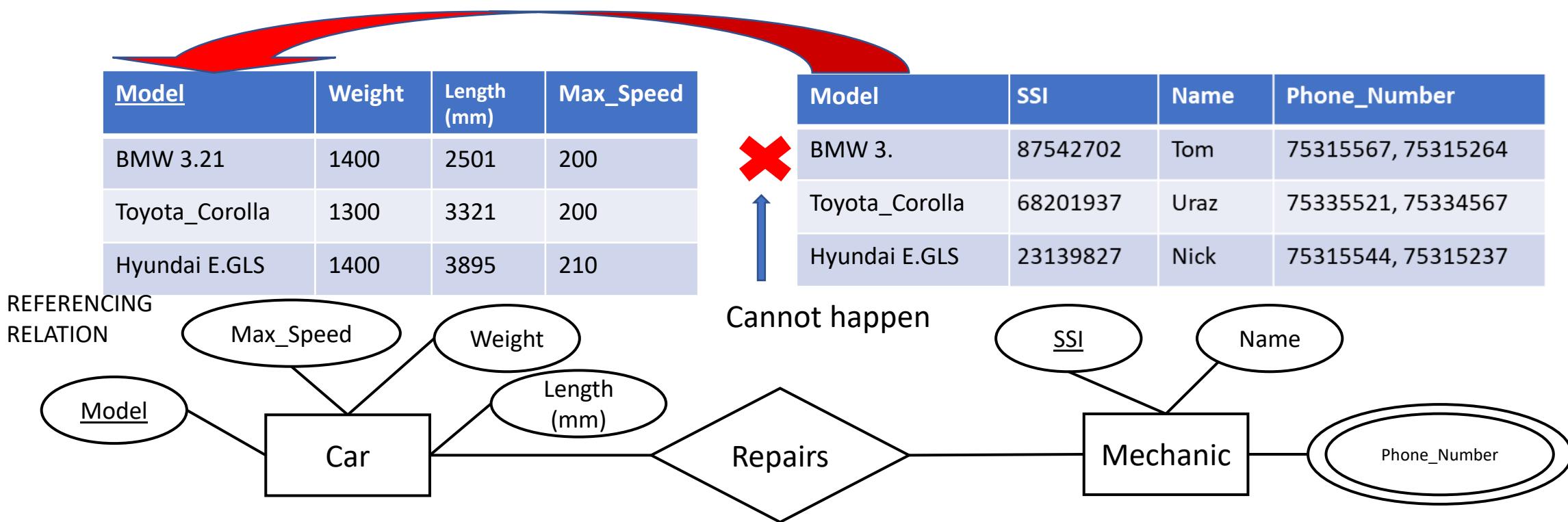
Model	SSI	Name	Phone_Number
BMW 3.21	87542702	Tom	75315567, 75315264
Toyota_Corolla	68201937	Uraz	75335521, 75334567
Hyundai E.GLS	23139827	Nick	75315544, 75315237

REFERENCING  
RELATION



# Referential integrity: Foreign keys

- Foreign key must:
  - Have the same name and domain/type as the referencing relation.
  - Having the same values.



# 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 examples?
- Links in HTML
- Pointers in C++
- Phone numbers

OverviewID	RulesID	ID	Rule
3	1	1	Rule1
1	1	1	Rule1
5	6	#Error	Rule5
*			(New)

You cannot add or change a record because a related record is required in table 'Overview'.

OK

Help

[Was this information helpful?](#)

e sorry,  
you have dialed  
sconnected or  
ger in service.  
ck the number  
and dial again

# Foreign Key in action.

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

Find primary keys and foreign keys  
and also mark referencing relations.

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

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

Process "Customers"  
Then "Orders" and "Items" using Foreign  
Keys.  
->3, LCD32" Monitor, RTX GPU, 16GBram.

Can you draw the ER diagram for these relations? (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!***

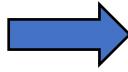
Students				
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

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 are the all tuples.

Do tables obey key constraints?

Do tables obey domain constraints?

Do tables obey referential integrity?

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

---

Please read 5.1-5.3 from the textbook “Fundamentals of database systems.”

**See you all in the following lecture!**

ERD to Relational Schema

# Recall

---



- Relation.

# Recall

---

- Relation.
  - An instance containing m-tuples where each tuple has n values.
    - n is the arity/degree of relation.

# Recall

---

- Relation.
  - An instance containing m-tuples where each tuple has n values.
    - n is the arity/degree of relation.
  - Relation Schema: The name of the relation + names and types (domains) of attributes.

# Recall

---

- Relation.
  - An instance containing m-tuples where each tuple has n values.
    - n is the arity/degree of relation.
  - Relation Schema: The name of the relation + names and types (domains) of attributes.

Fictional\_Characters(CharacterID:*integer*, CharacterName:*string*)

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by

• • • • • • • • • •

# Integrity/Integrity constraint

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - ...
  - ...
  - ...
  - ...
  - ...
  - ...

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- 
- 
- 
- 
- 
- 
-

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
  - 
  - 
  - 
  - 
  -

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
    - The  $i^{\text{th}}$  value of tuple  $j$  must be in set  $\text{dom}(t_i[A_j])$
    - 
    - 
    - 
    -

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
    - The  $i^{\text{th}}$  value of tuple  $j$  must be in set  $\text{dom}(t_i[A_j])$
  - **Key constraints**
    - 
    - 
    -

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
    - The  $i^{\text{th}}$  value of tuple  $j$  must be in set  $\text{dom}(t_i[A_j])$
  - **Key constraints**
    - A key value cannot be null (**Entity integrity constraint**)
    - 
    -

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
    - The  $i^{\text{th}}$  value of tuple  $j$  must be in set  $\text{dom}(t_i[A_j])$
  - **Key constraints**
    - A key value cannot be null (**Entity integrity constraint**)
    - **Foreign key**: It is the primary key to the foreign relation. Included in the current relation to establish a logical link between them.
    -

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
    - The  $i^{\text{th}}$  value of tuple  $j$  must be in set  $\text{dom}(t_i[A_j])$
  - **Key constraints**
    - A key value cannot be null (**Entity integrity constraint**)
    - **Foreign key**: It is the primary key to the foreign relation. Included in the current relation to establish a logical link between them.
    - **Referential Integrity**: Foreign keys must have the same type and values in the current relation
    -

# Integrity/Integrity constraint

---

- A relation contains data which must obey some rules given by
  - DB Designer,
  - Functions,
  - Real-World rules.
- These rules are referred to as **Constraints**:
  - **Domain constraints**
    - The  $i^{\text{th}}$  value of tuple  $j$  must be in set  $\text{dom}(t_i[A_j])$
  - **Key constraints**
    - A key value cannot be null (**Entity integrity constraint**)
    - **Foreign key**: It is the primary key to the foreign relation. Included in the current relation to establish a logical link between them.
    - **Referential Integrity**: Foreign keys must have the same type and values in the current relation.
    - **Superkey**: A set of attributes such that its subset is a key for the relation.



- We will see some simple rules that regulate ERD->Relational Model conversion.





- We will see some simple rules that regulate ERD->Relational Model conversion.
  - This is important because
    - ERD->SQL 
    - 
    - 
    - 
    -



- We will see some simple rules that regulate ERD->Relational Model conversion.
  - This is important because
    - ERD->SQL 
    - Relational model -> SQL 
  - 
  - 
  -

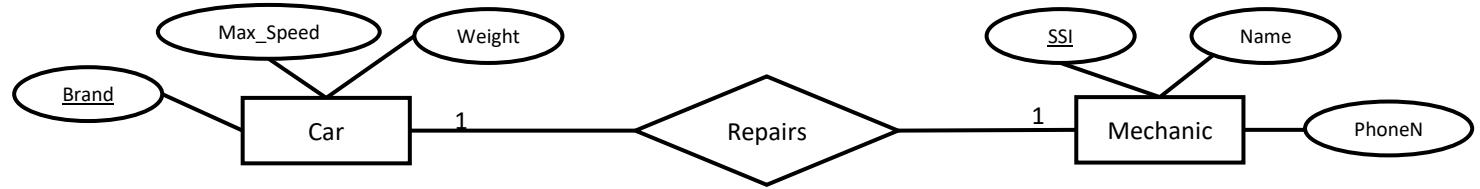


- We will see some simple rules that regulate ERD->Relational Model conversion.
  - This is important because
    - ERD->SQL
    - Relational model -> SQL
  - Jumping directly to SQL declarations will never be an option because we will optimise relations via
    - Relational Algebra (next week)
    - Normalisation (the week after next week).

# Relationship types

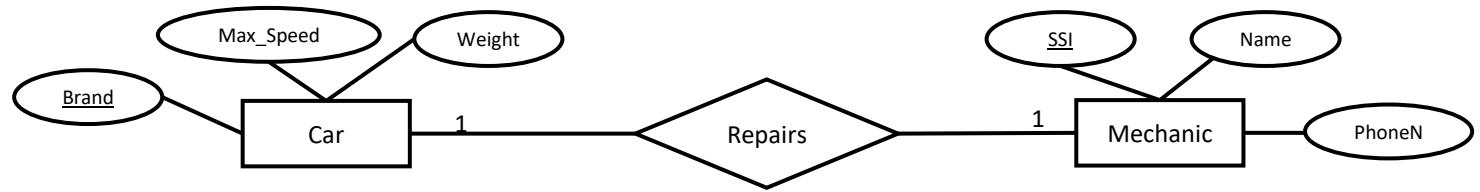
---

- 1 to 1 relations



# Relationship types

- 1 to 1 relations

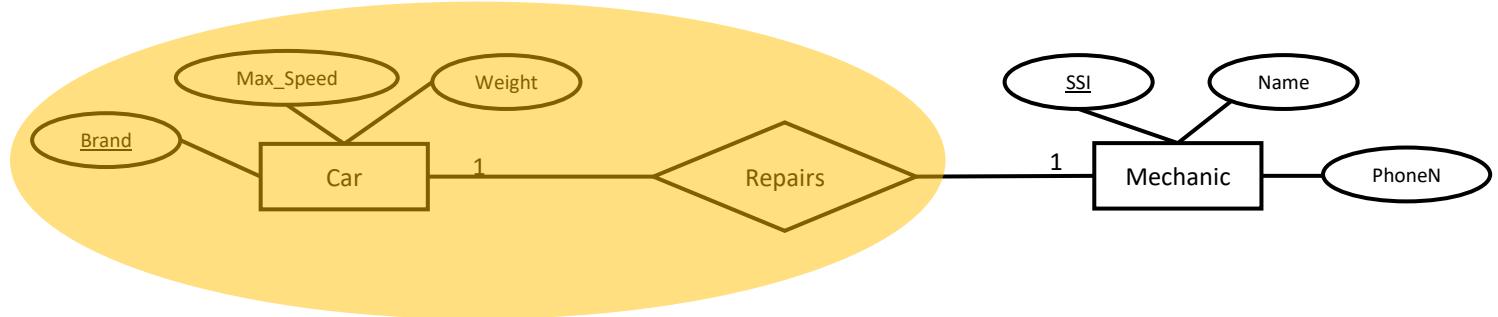


Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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

# Relationship types

- 1 to 1 relations

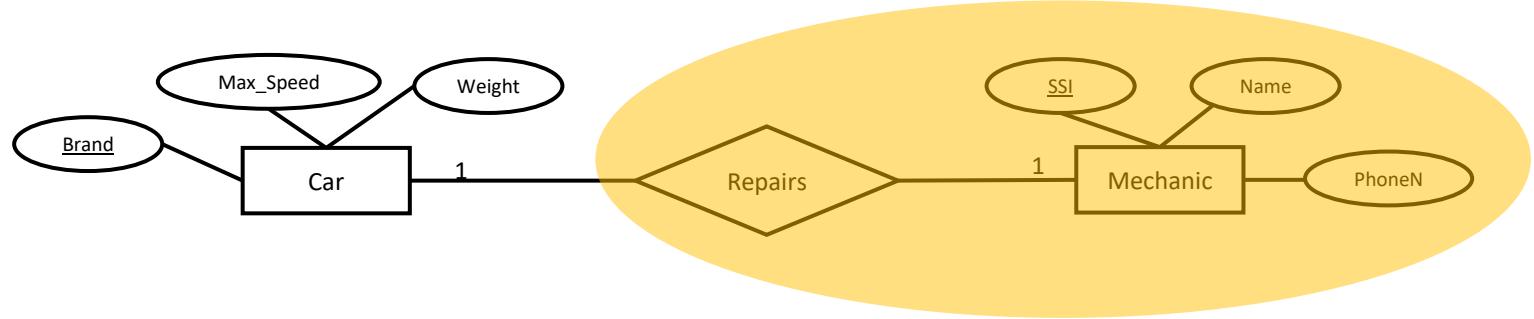


Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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

# Relationship types

- 1 to 1 relations

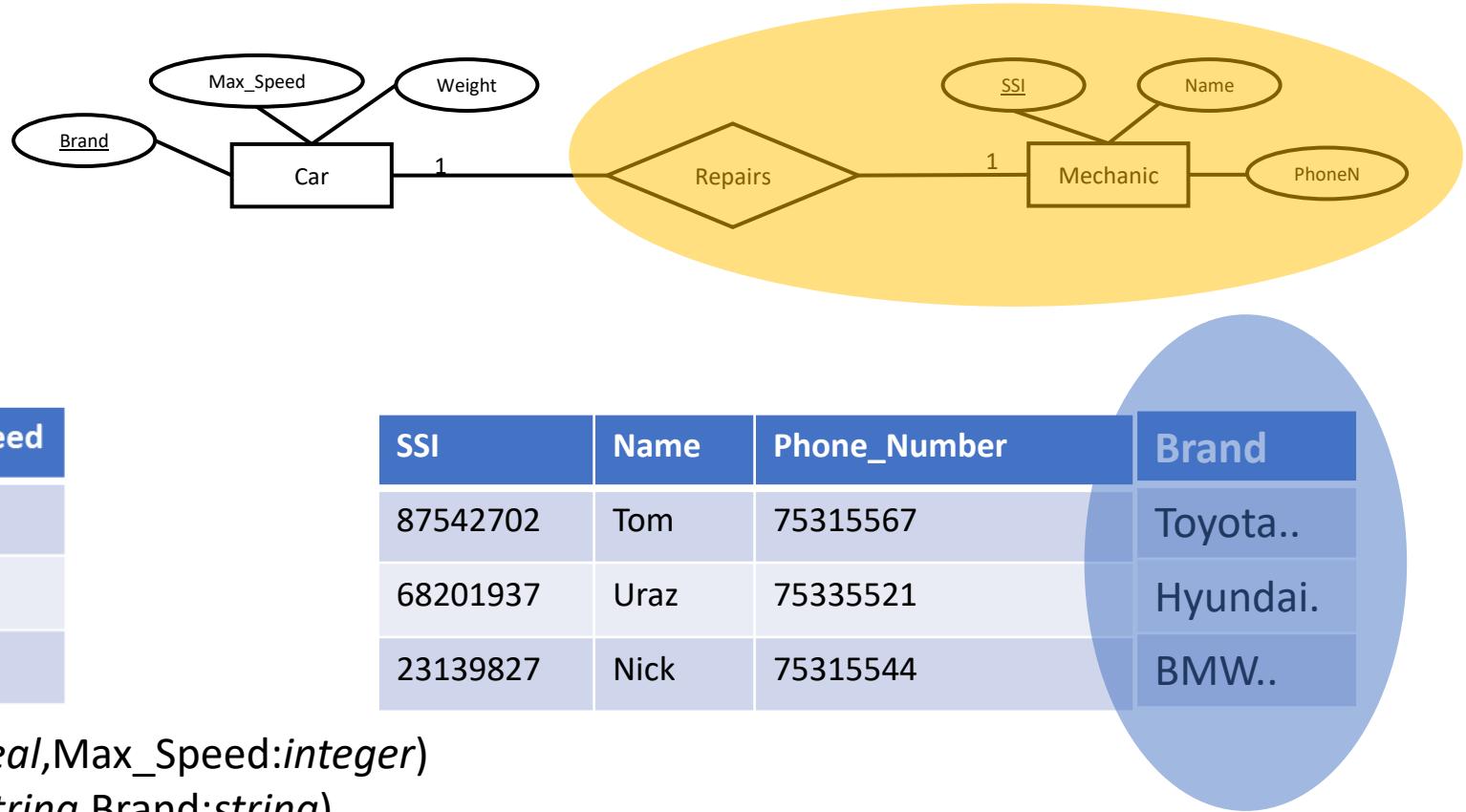


Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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

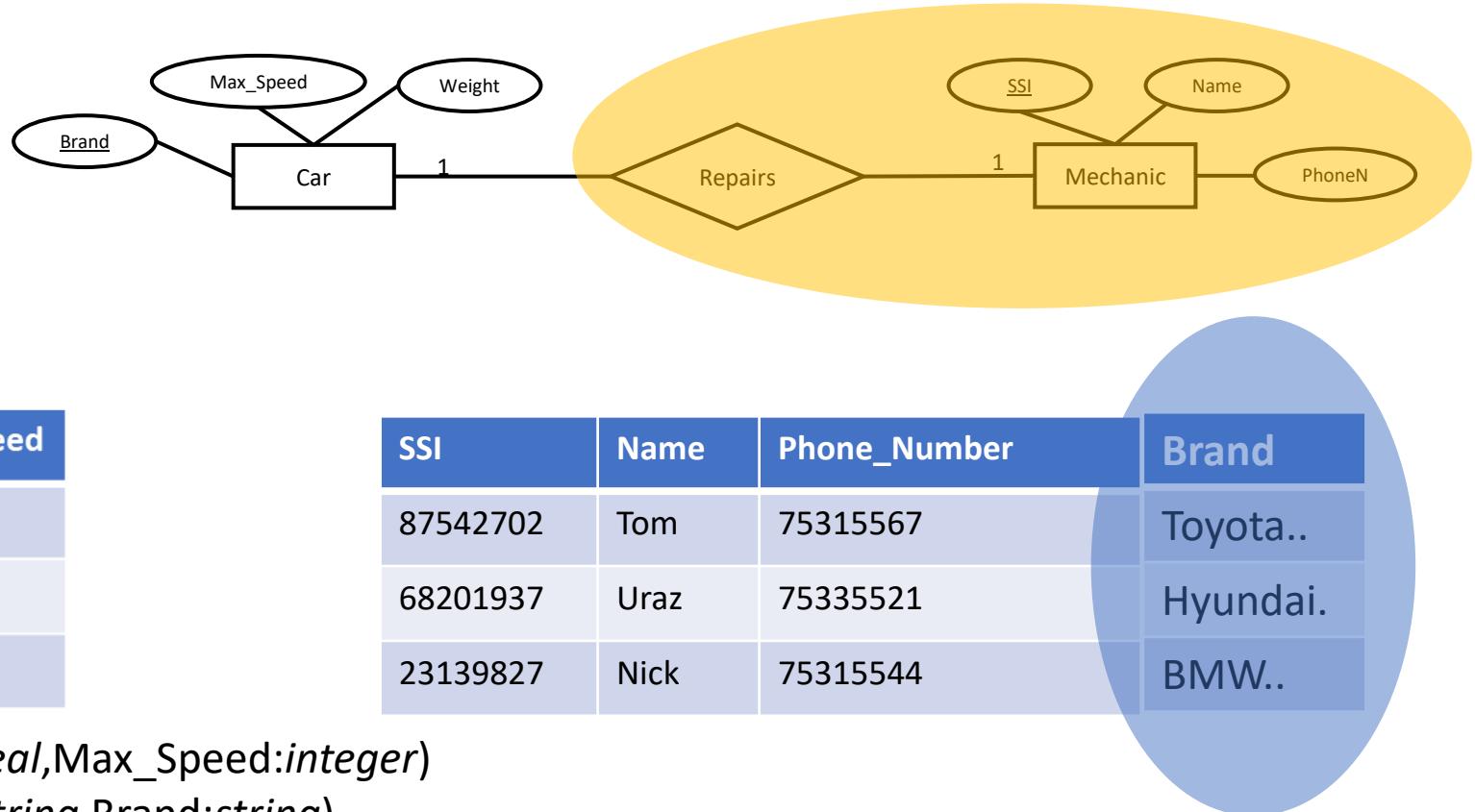
# Relationship types

- 1 to 1 relations



# Relationship types

- 1 to 1 relations

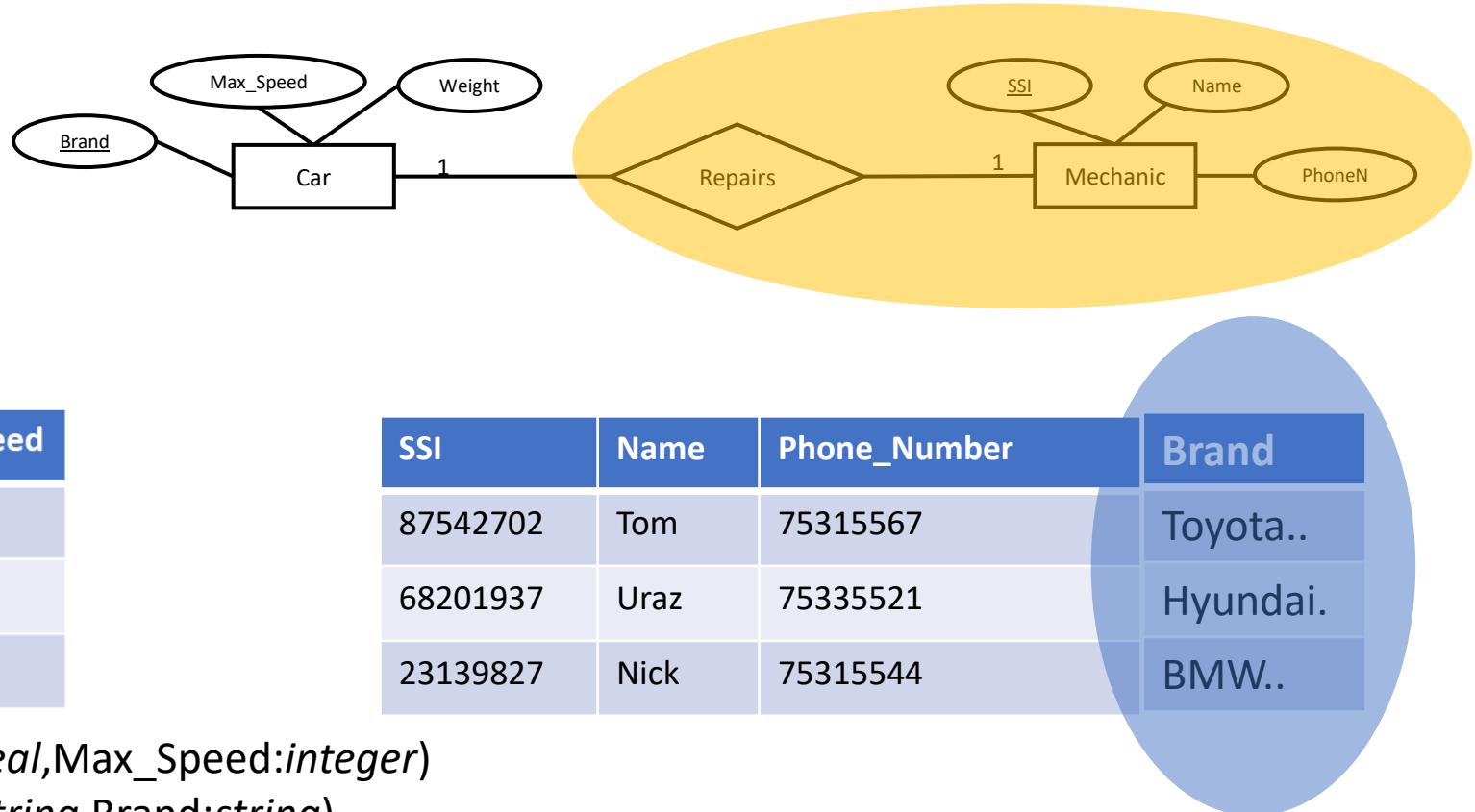


Car(*Brand:string,Weight:integer,Length:real,Max\_Speed:integer*)  
Mec\_Rep(*SSI:string,Name:string,Phone:string,Brand:string*)

This is not enough.. Why?

# Relationship types

- 1 to 1 relations

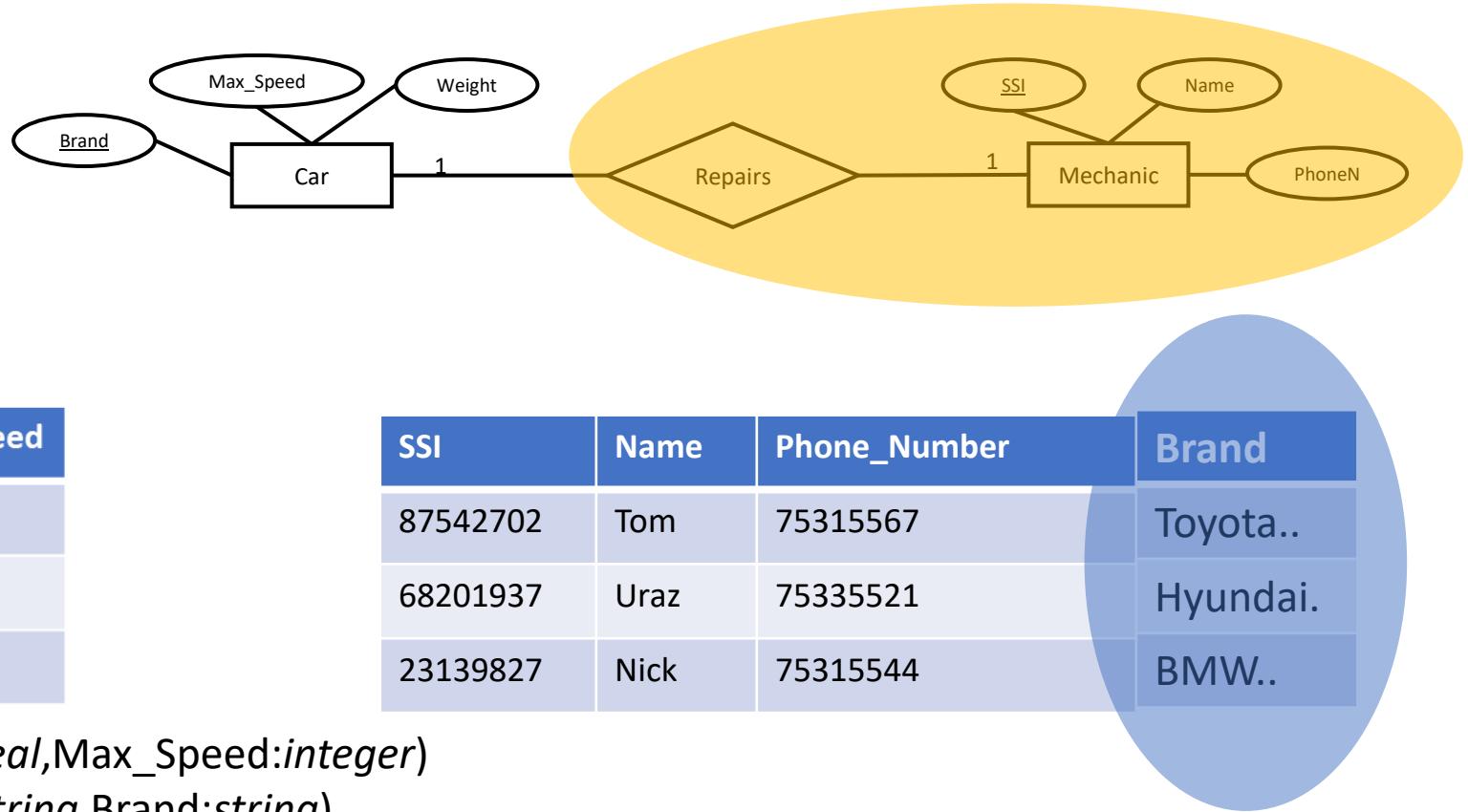


Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)  
Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

IC's? Foreign key Brand referencing CAR. On delete \_\_\_\_?

# Relationship types

- 1 to 1 relations

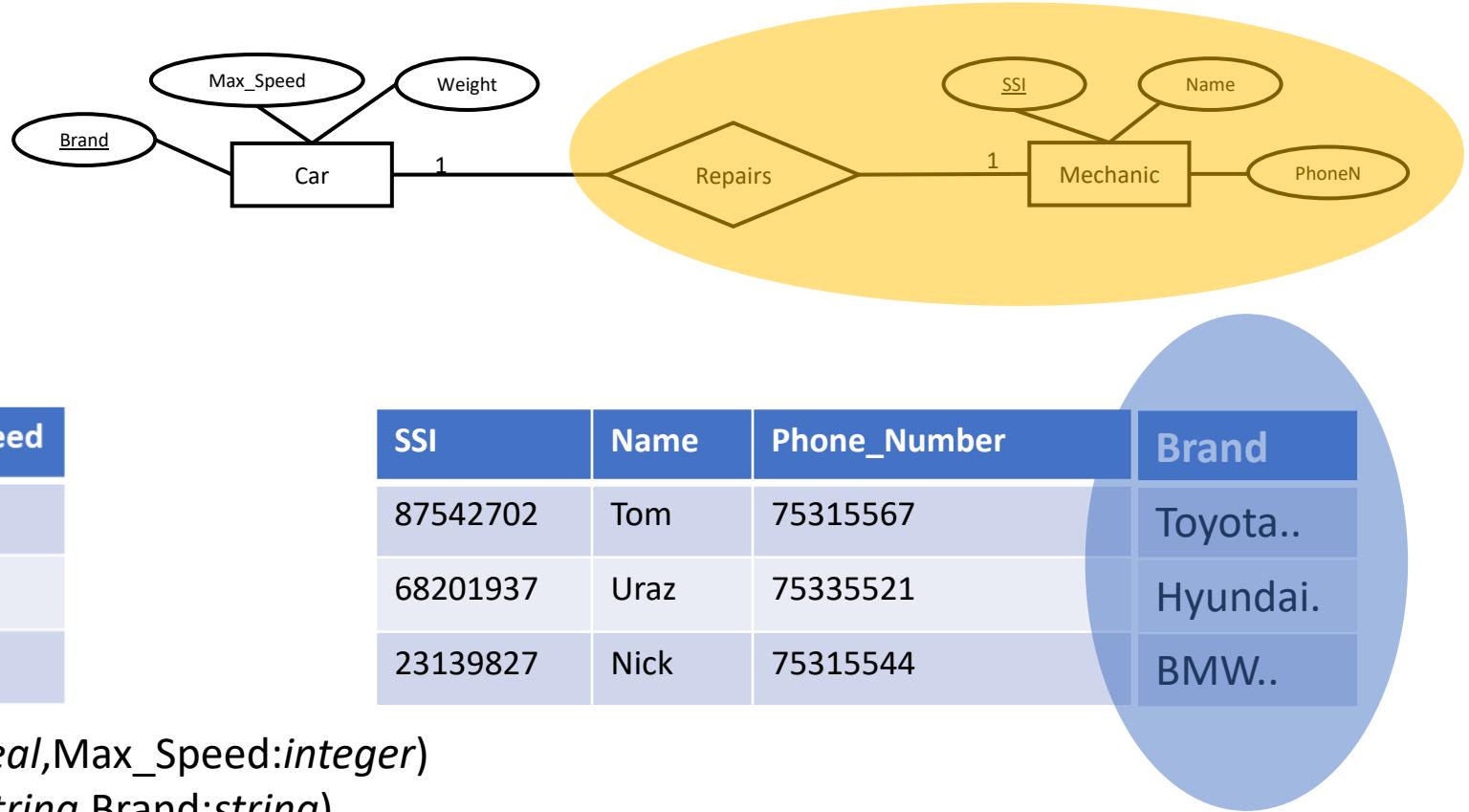


Car(*Brand:string,Weight:integer,Length:real,Max\_Speed:integer*)  
Mec\_Rep(*SSI:string,Name:string,Phone:string,Brand:string*)

Foreign key Brand referencing CAR. On delete reject?

# Relationship types

- 1 to 1 relations

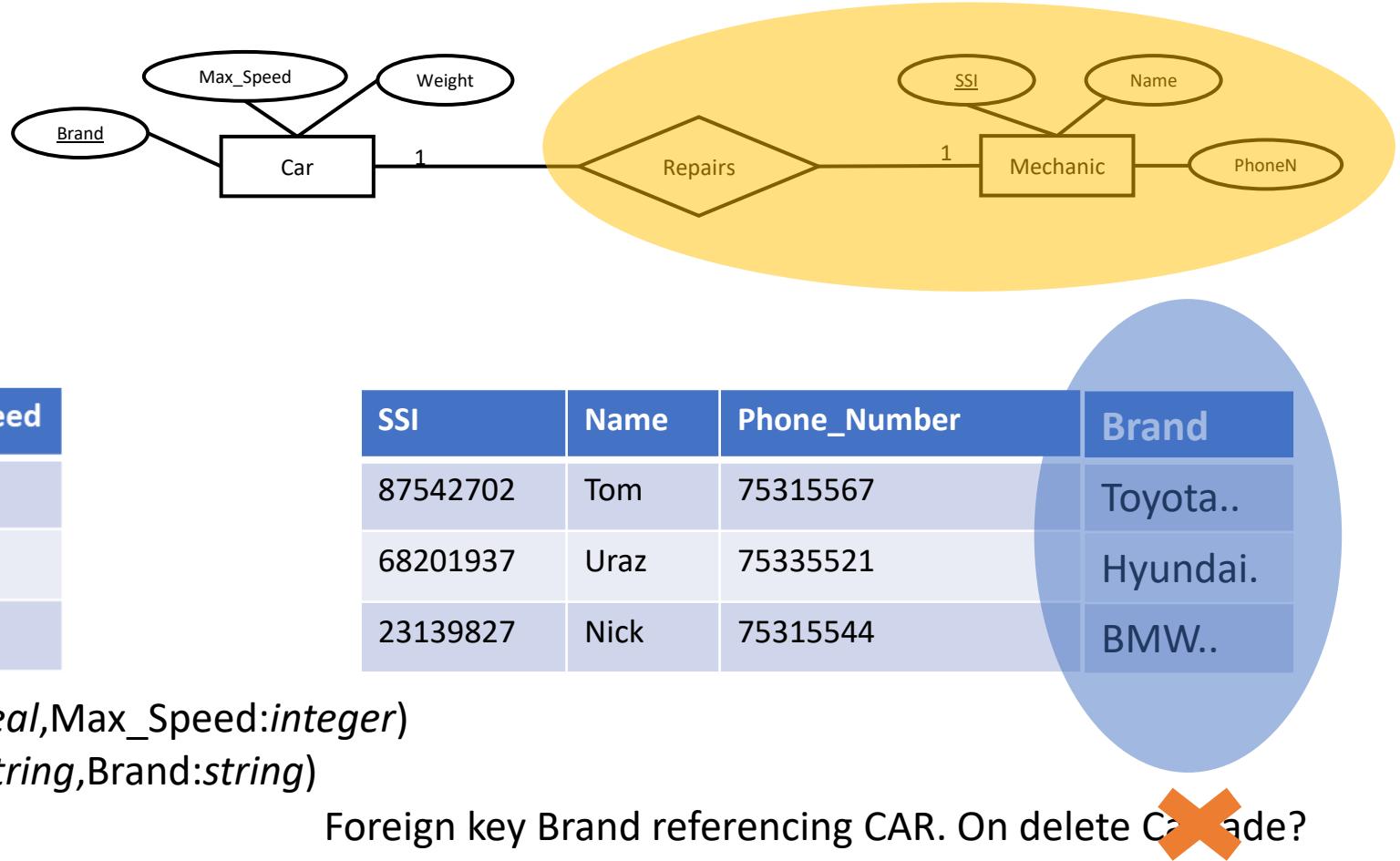


`Car(Brand:string,Weight:integer,Length:real,Max_Speed:integer)`  
`Mec_Rep(SSI:string,Name:string,Phone:string,Brand:string)`

Foreign key Brand referencing CAR. On delete Cascade?

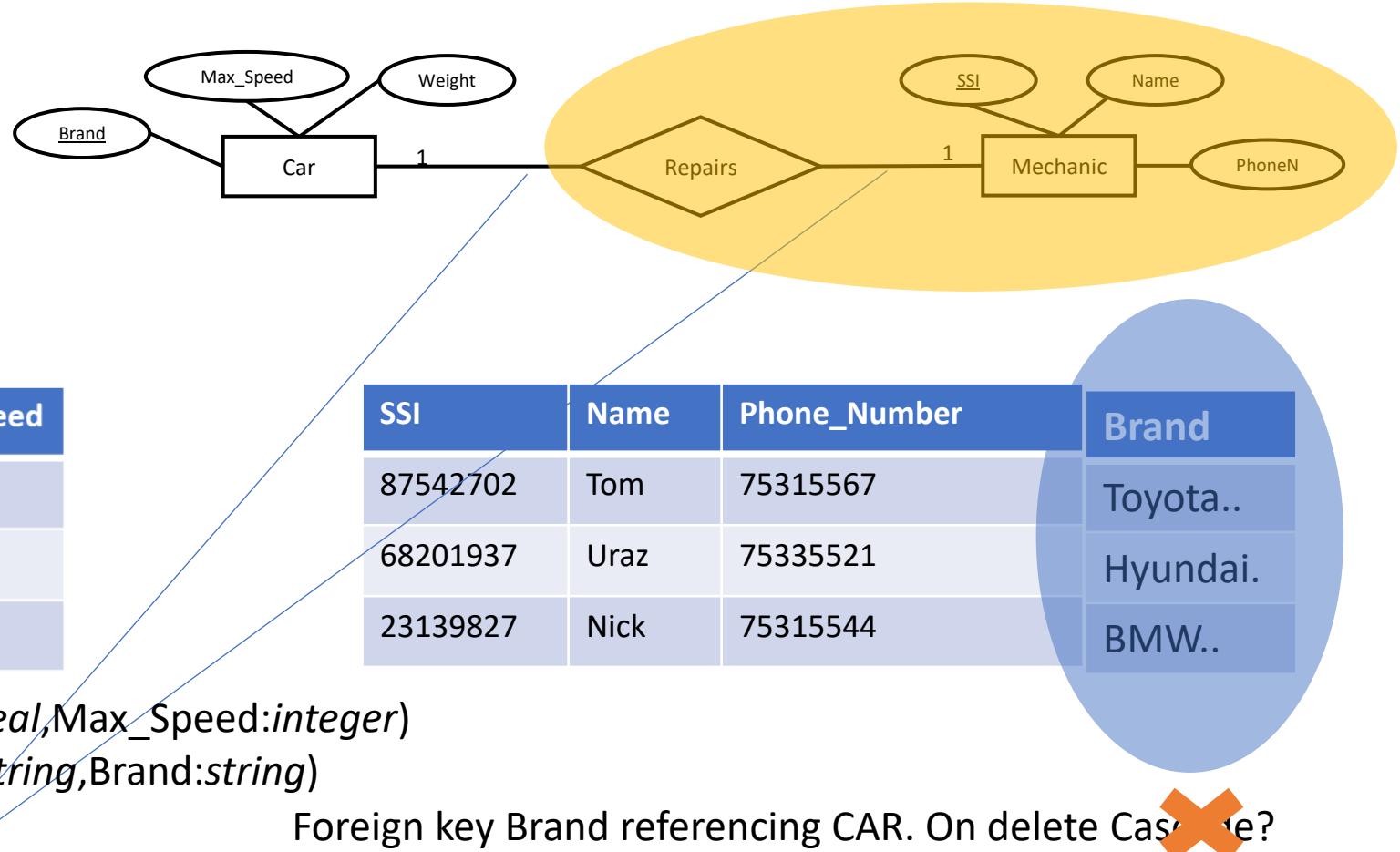
# Relationship types

- 1 to 1 relations



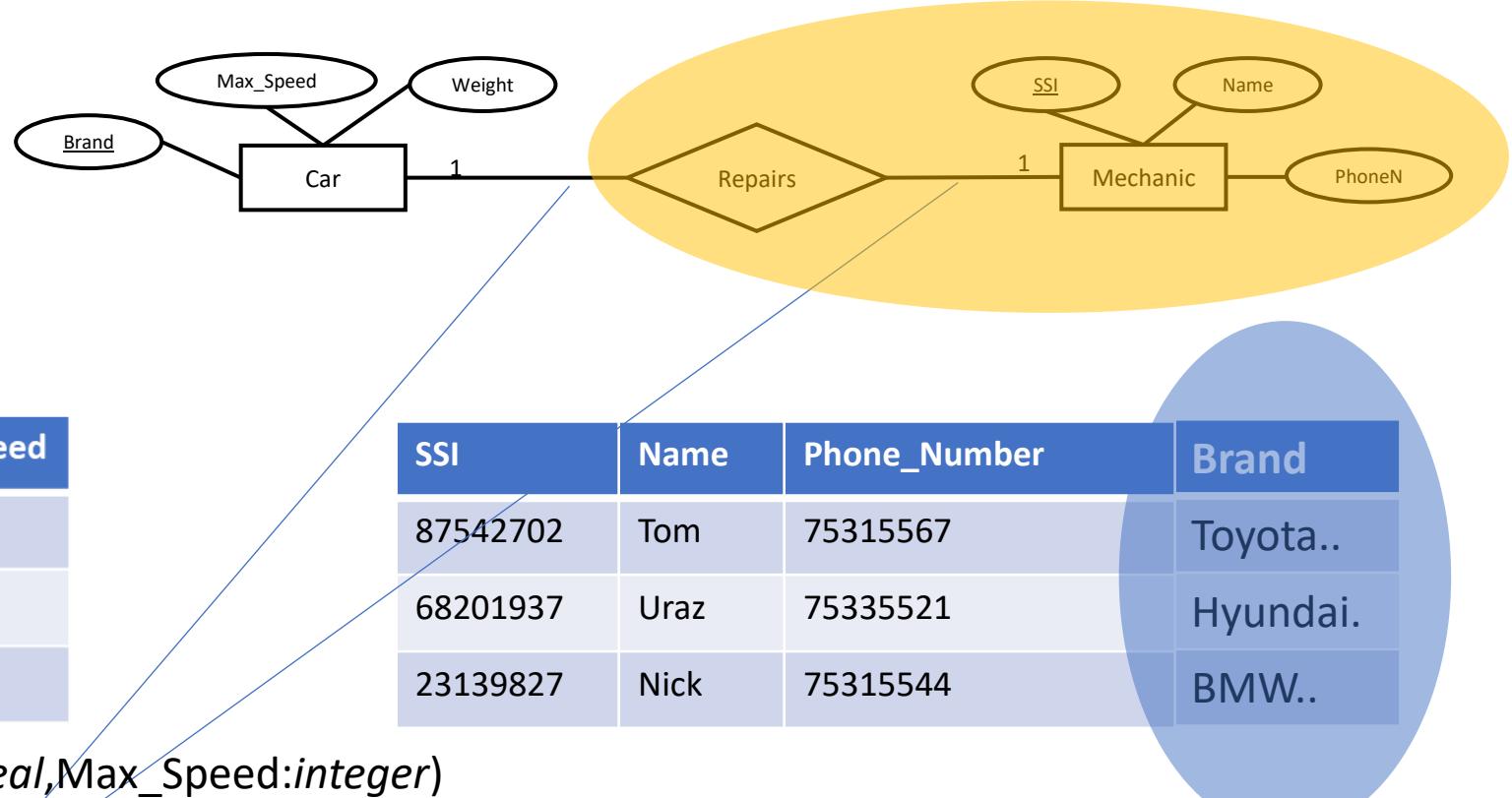
# Relationship types

- 1 to 1 relations



# Relationship types

- 1 to 1 relations



Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

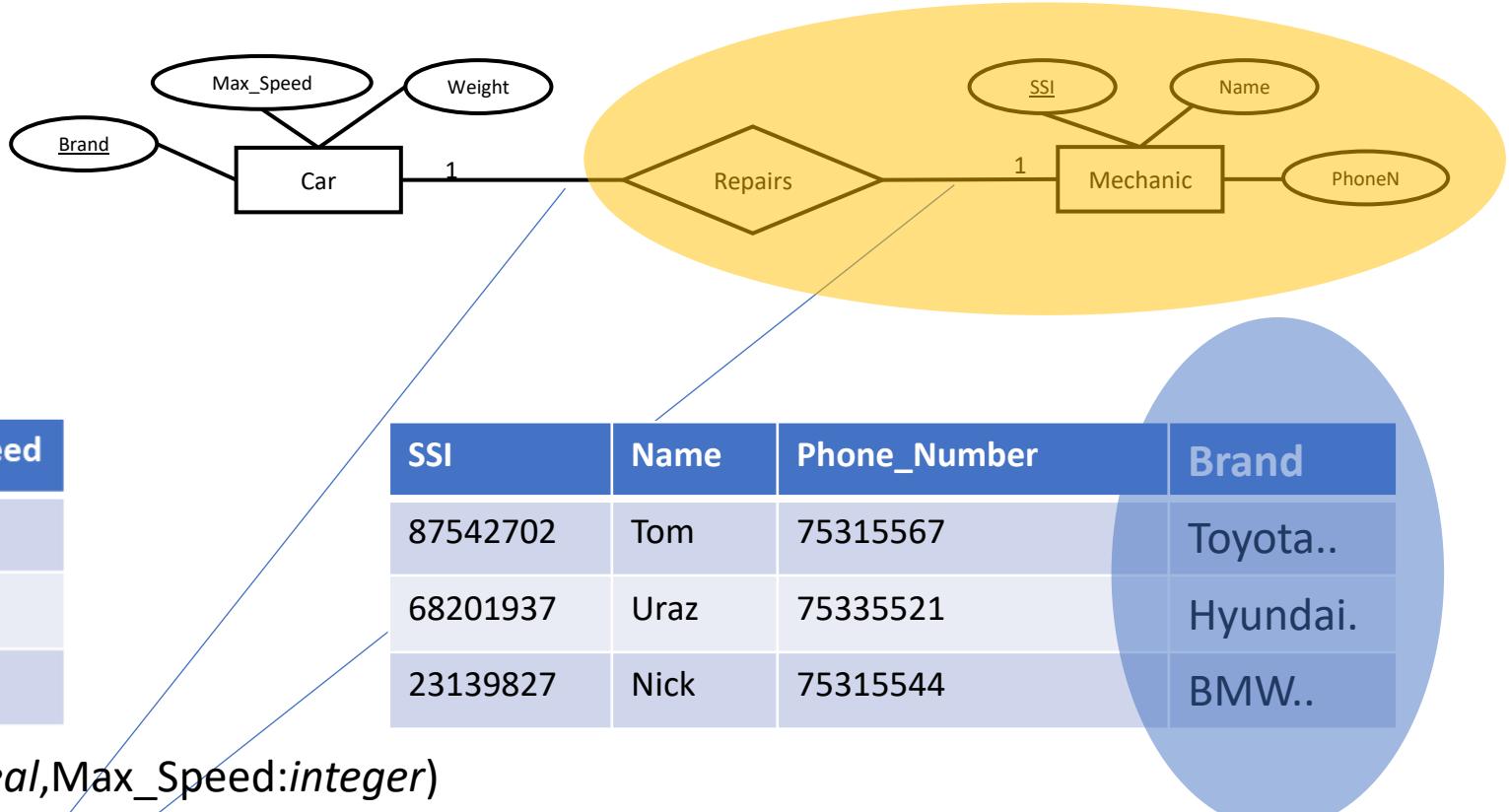
Partial participation!

Foreign key Brand referencing CAR. On delete Ca~~X~~nde?

# Relationship types

- 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



`Car(Brand:string,Weight:integer,Length:real,Max_Speed:integer)`

`Mec_Rep(SSI:string,Name:string,Phone:string,Brand:string)`

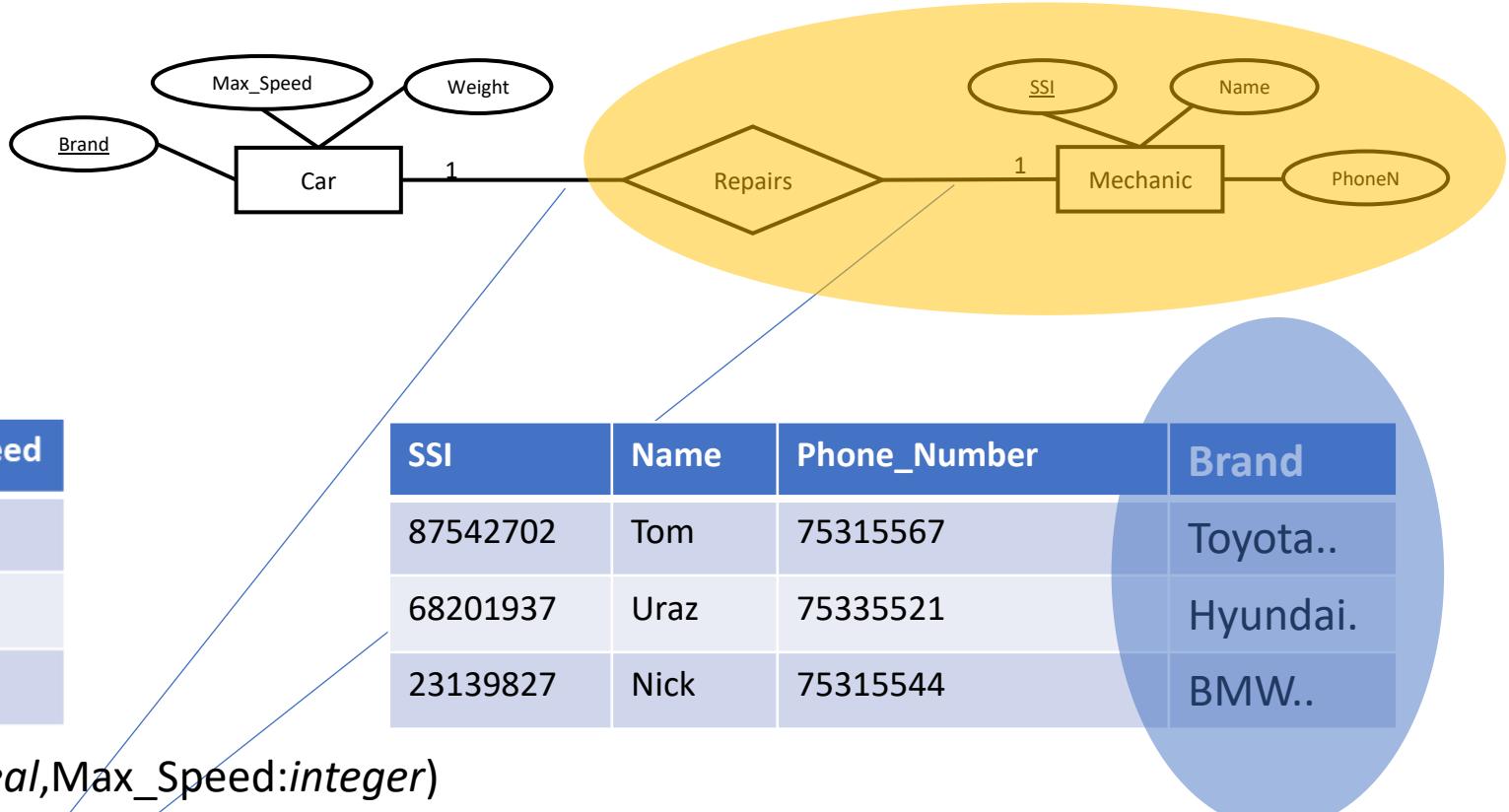
Partial participation!

Foreign key Brand referencing CAR. On delete **SET NULL**

# Relationship types

- 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



`Car(Brand:string,Weight:integer,Length:real,Max_Speed:integer)`

`Mec_Rep(SSI:string,Name:string,Phone:string,Brand:string)`

Partial participation!

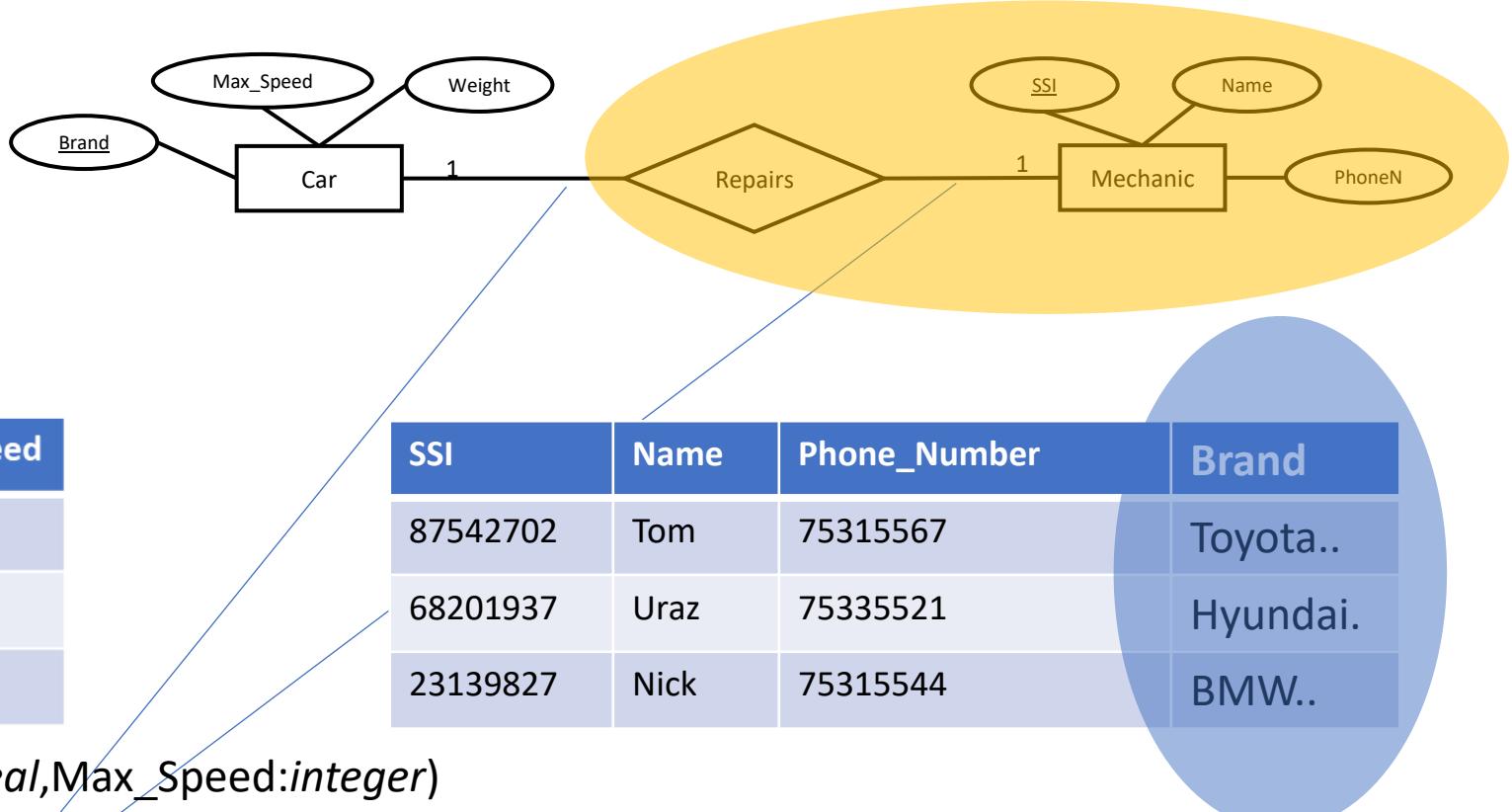
This is not enough.. Why?

Foreign key Brand referencing CAR. On delete **SET NULL**

# Relationship types

- 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



`Car(Brand:string,Weight:integer,Length:real,Max_Speed:integer)`

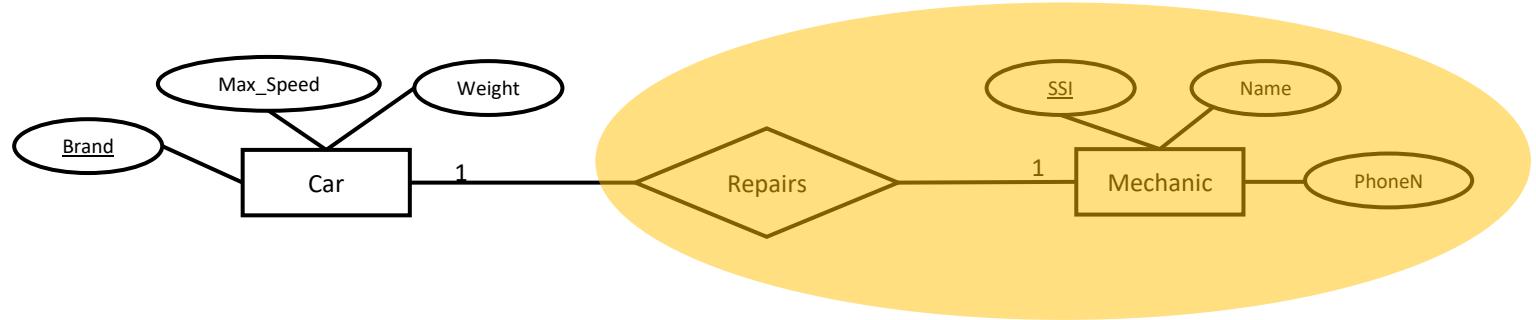
`Mec_Rep(SSI:string,Name:string,Phone:string,Brand:string)`

Partial participation!

Foreign key Brand referencing CAR. On delete **SET NULL**,  
**Brand is UNIQUE**

# Relationship types

- 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..
68201937	Uraz	75335521	Hyundai..
23139827	Nick	75315544	BMW..

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

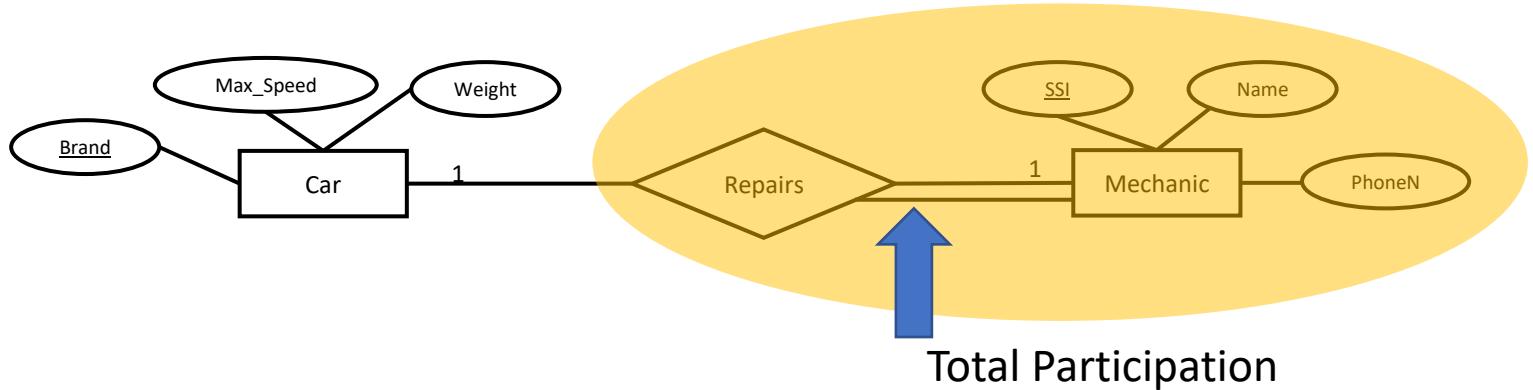
IC for CAR: Primary key Brand

IC's for Mec\_Rep : Primary key SSI, Foreign key Brand referencing CAR.

On deleting car tuple **SET NULL/DEFAULT**, Brand is **UNIQUE**.

# Relationship types

- 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..
68201937	Uraz	75335521	Hyundai..
23139827	Nick	75315544	BMW..

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

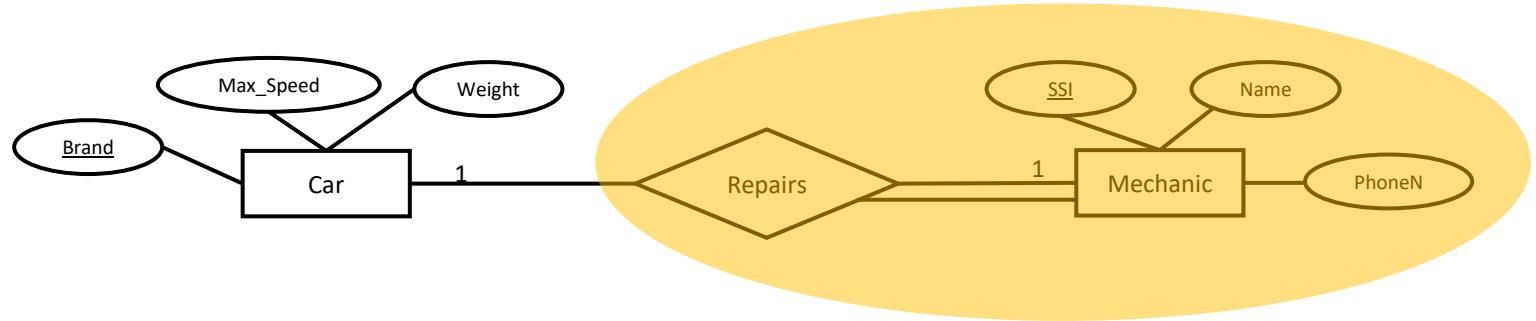
Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

IC for CAR: Primary key Brand

IC's for MEC\_Rep: Primary key SSI, Foreign key Brand referencing CAR. On delete ??

# Relationship types

- 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..
68201937	Uraz	75335521	Hyundai..
23139827	Nick	75315544	BMW..

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

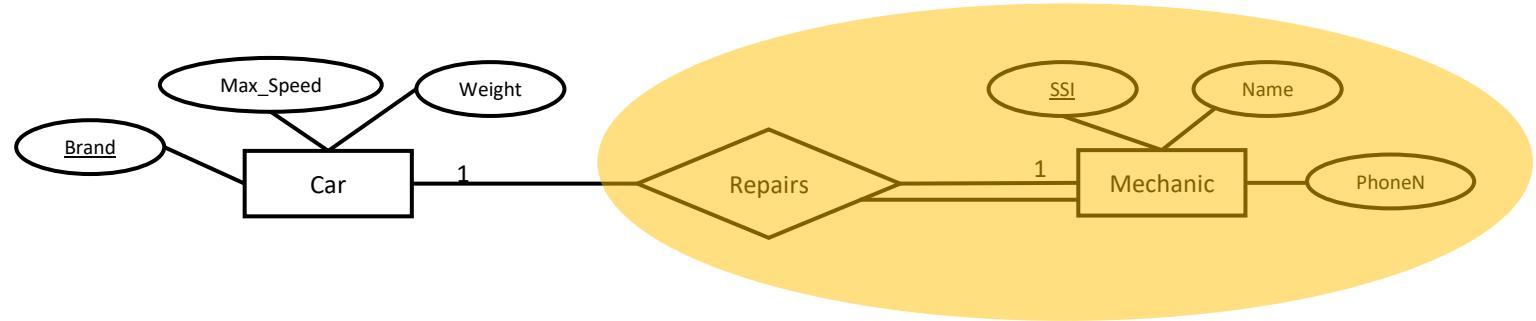
IC for CAR: Primary key Brand

IC's for Mec\_Rep: Primary key SSI, Foreign key Brand referencing CAR.

On delete **CASCADE/REJECT**

# Relationship types

- 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..
68201937	Uraz	75335521	Hyundai..
23139827	Nick	75315544	BMW..

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

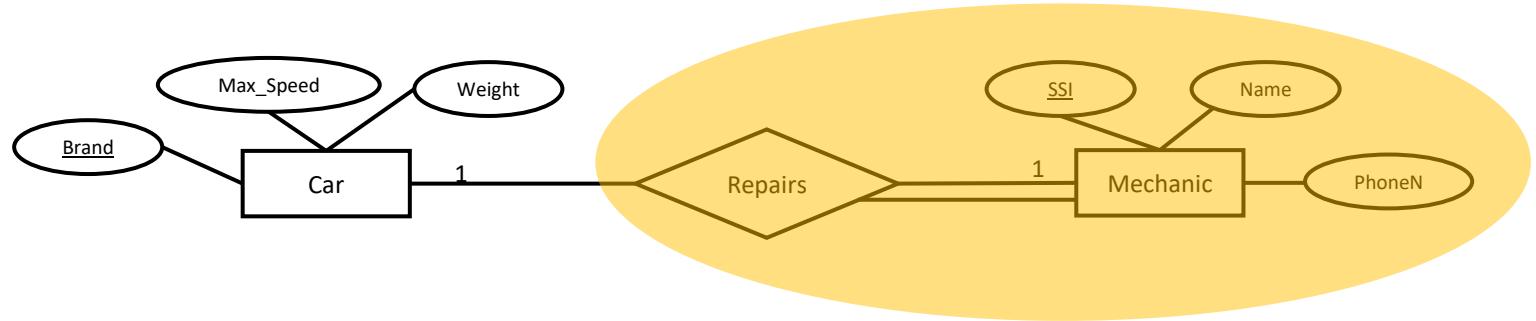
IC for CAR: Primary key Brand

IC's for Mec\_Rep: Primary key SSI, Foreign key Brand referencing CAR.

On delete **CASCADE/REJECT, BRAND CANNOT BE NULL**

# Relationship types

- 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..
68201937	Uraz	75335521	Hyundai..
23139827	Nick	75315544	BMW..

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

Mec\_Rep(SSI:string,Name:string,Phone:string,Brand:string)

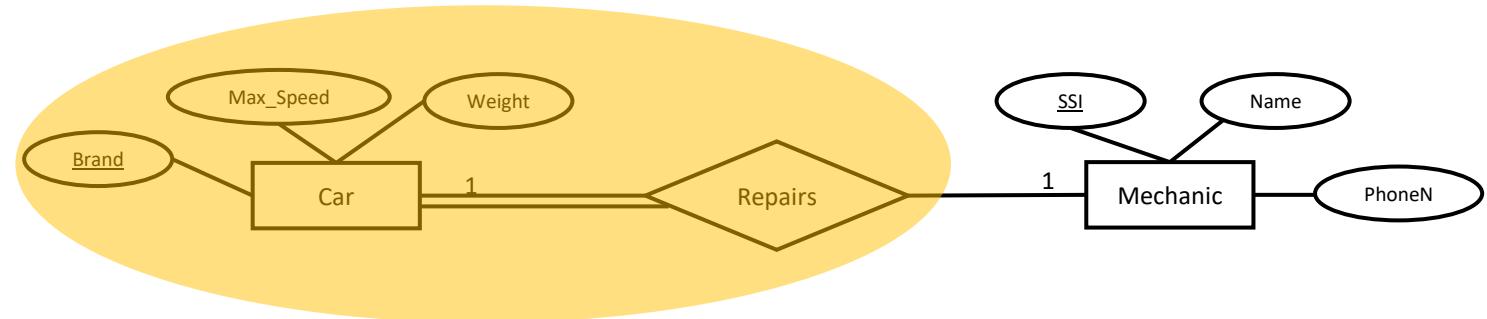
IC for CAR: Primary key Brand

IC's for Mec\_Rep: Primary key SSI, Foreign key Brand referencing CAR.

On delete **CASCADE/REJECT**, BRAND CANNOT BE NULL, Brand is UNIQUE

# Relationship types

- 1 to 1 relations



Brand	Weight	Length	Max_Speed	SSI
BMW 3.21	1400	3.21	200	87542702
Toyota_Corolla	1300	3.18	200	68201937
Hyundai E.GLS	1400	3.16	210	23139827

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

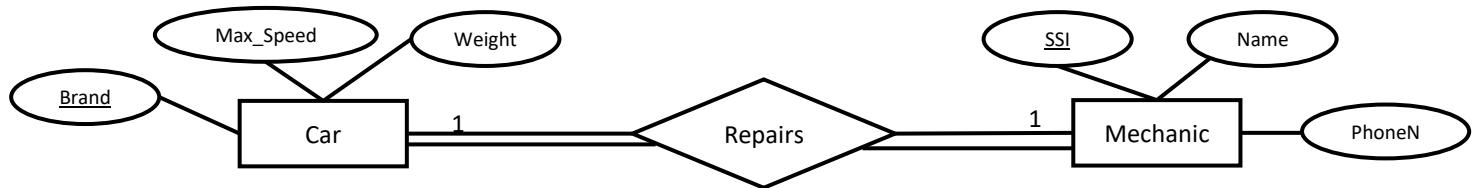
Car\_Rep(*Brand:string,Weight:integer,Length:real,Max\_Speed:integer,SSI: string*)

IC's for Car\_Rep: Primary key Brand, Foreign key SSI referencing Mec. On delete CASCADE/REJECT, SSI CANNOT BE NULL, SSI is UNIQUE

Mec (*SSI:string,Name:string,Phone:string*) IC for Mec: Primary key SSI

# Relationship types

- 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

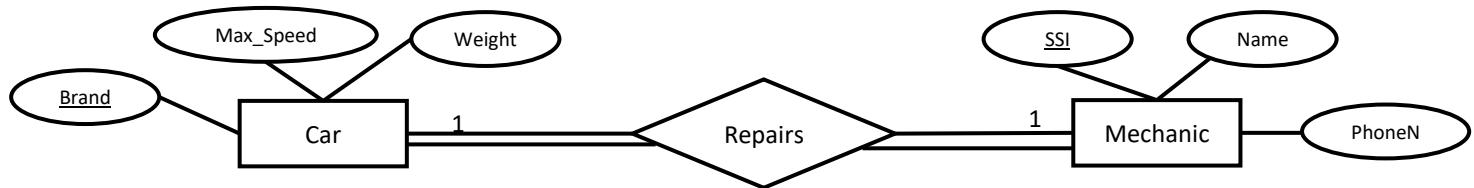
SSI	Name	Phone_Number
87542702	Tom	75315567
68201937	Uraz	75335521
23139827	Nick	75315544

# Relationship types

- 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

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)



SSI	Brand
87542702	Toyota..
68201937	Hyundai..
23139827	BMW..

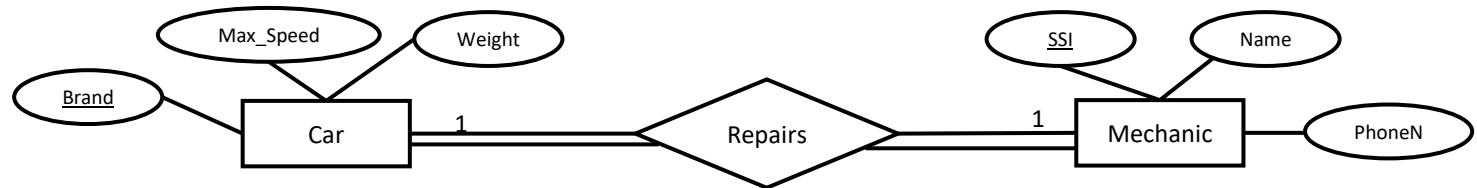
Car\_Rep\_Mec(Brand:string, SSI: string)

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

Mec (SSI:string,Name:string,Phone:string)

# Relationship types

- 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

Car(Brand:string,Weight:integer,Length:real,Max\_Speed:integer)

SSI	Brand
87542702	Toyota..
68201937	Hyundai..
23139827	BMW..

Car\_Rep\_Mec(Brand:string, SSI: string)

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

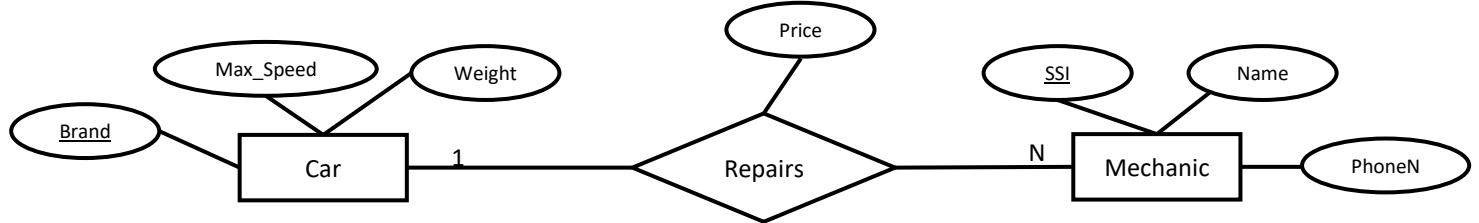
Mec (SSI:string,Name:string,Phone:string)

IC's for Car\_Rep\_Mec: Primary key SSI, Foreign Key SSI referencing Mec, Foreign Key Brand referencing Car, **On delete CASCADE/Reject, BRAND CANNOT BE NULL, Brand is UNIQUE**

# Relationship types

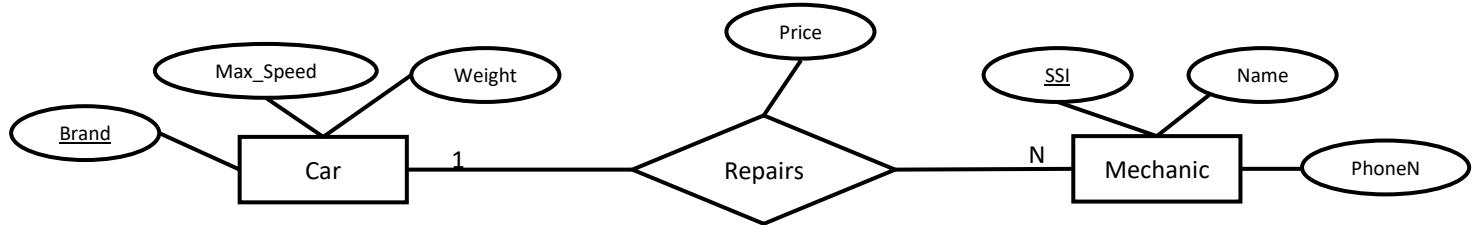
---

- 1 to Many relations  
(Same for Many to 1 relations)



# Relationship types

- 1 to Many relations  
(Same for Many to 1 relations)



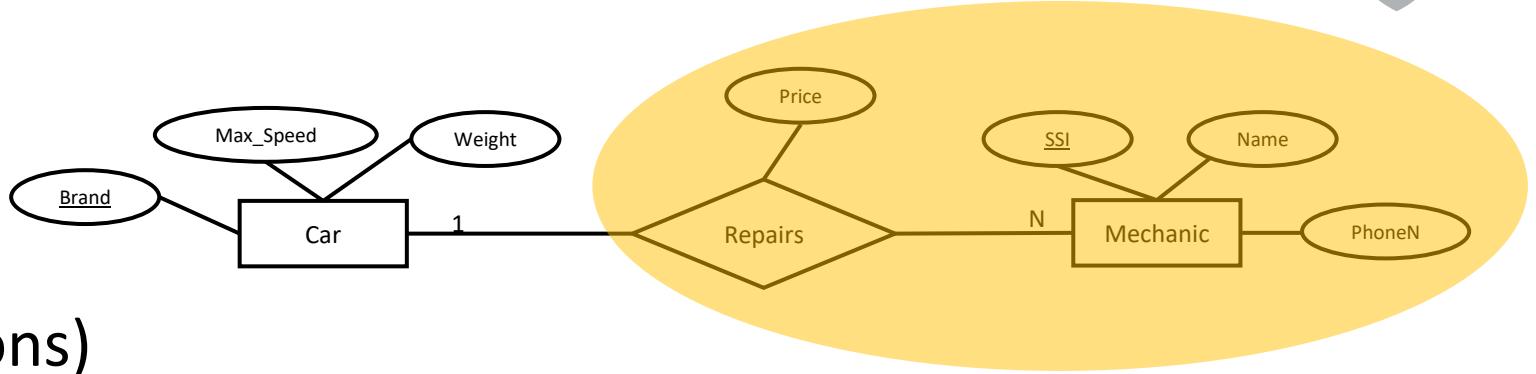
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Price
10
23
12

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

# Relationship types

- 1 to Many relations  
(Same for Many to 1 relations)

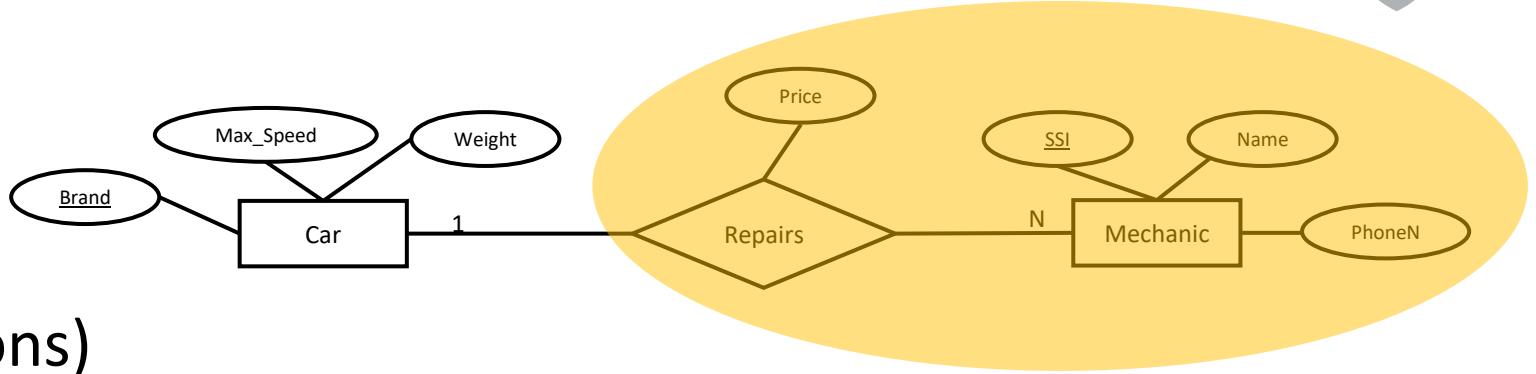


Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

Price	SSI	Name	Phone_Number
10	87542702	Tom	75315567
23	68201937	Uraz	75335521
12	23139827	Nick	75315544

# Relationship types

- 1 to Many relations  
(Same for Many to 1 relations)

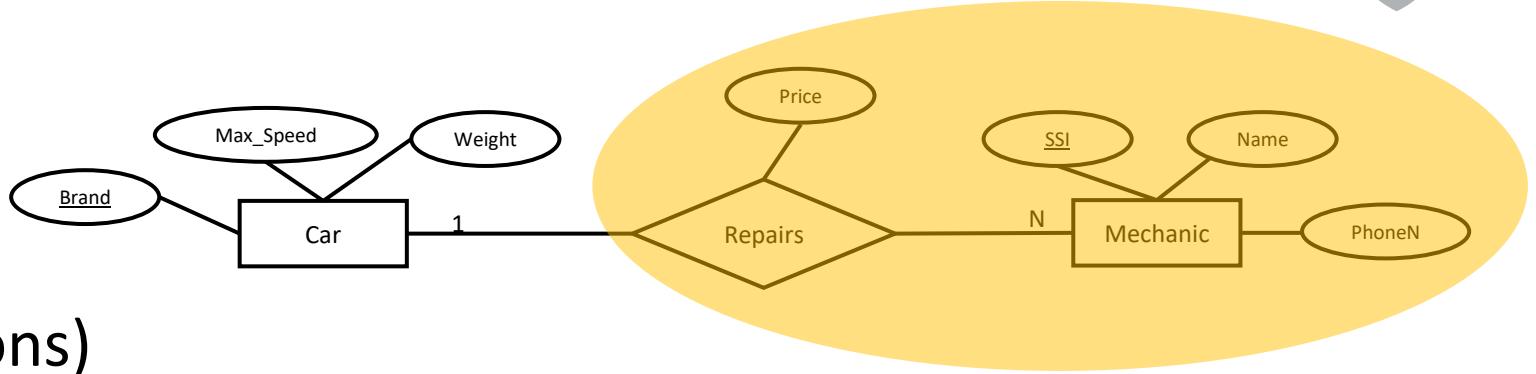


Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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

# Relationship types

- 1 to Many relations  
(Same for Many to 1 relations)



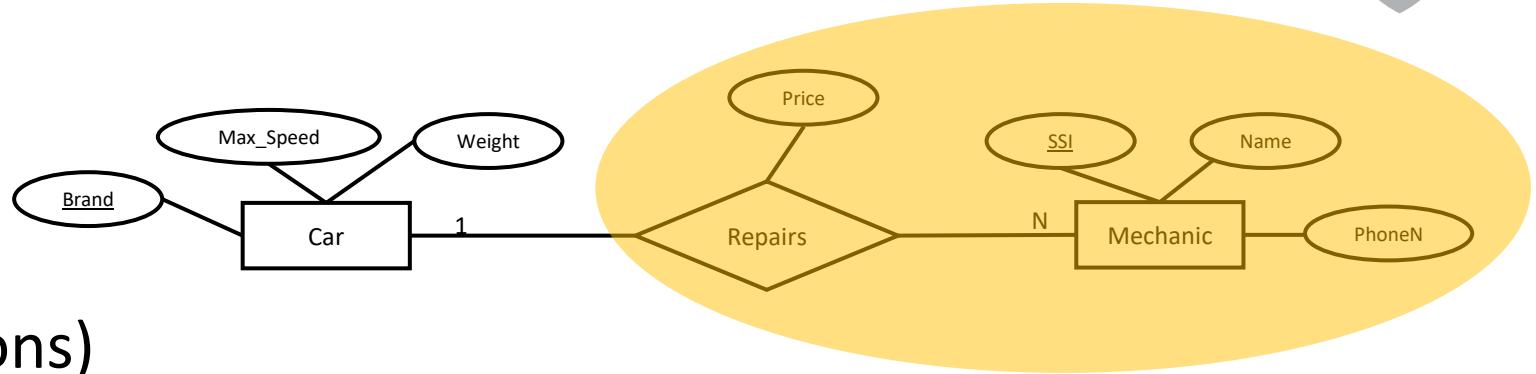
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

# Relationship types

- 1 to Many relations  
(Same for Many to 1 relations)



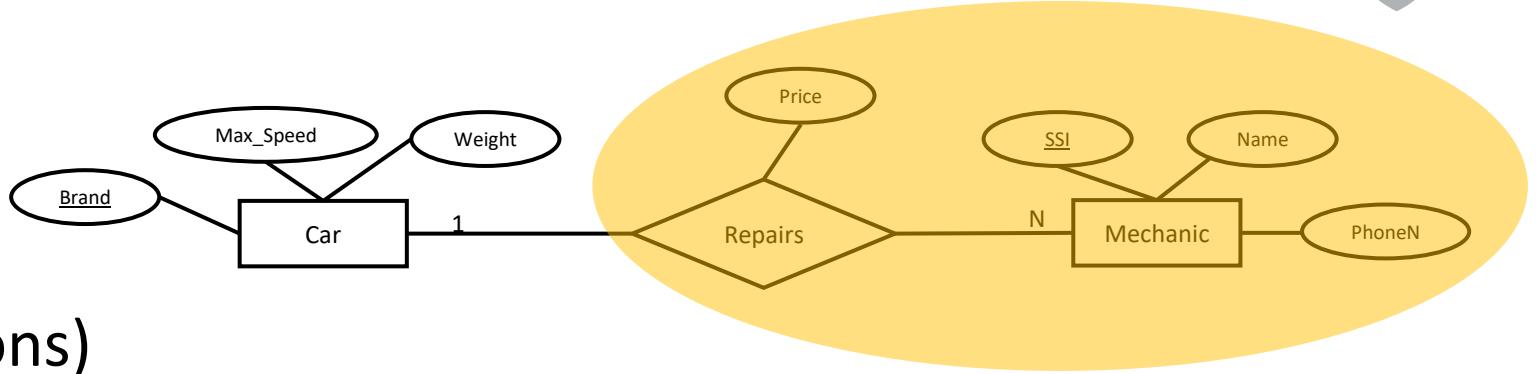
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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:string, Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.  
Mec\_R (Brand:string, Price: integer, SSI:integer, Name:string, Phone\_Number:string )

# Relationship types

- 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

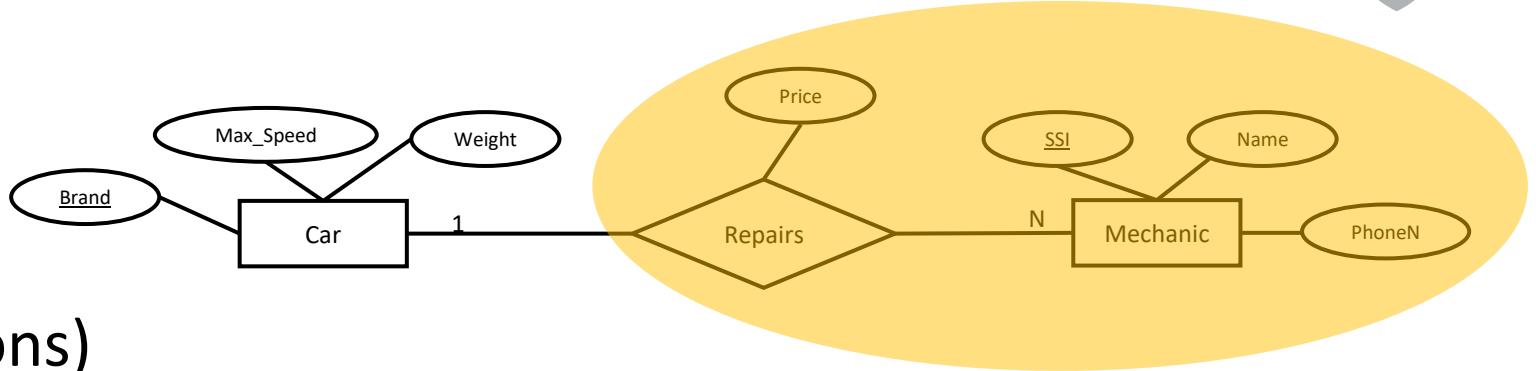
Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544

Car (Brand:string, Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

Mec\_R (Brand:string, Price: integer, SSI:integer, Name:string, Phone\_Number:string ) IC's: Primary key SSI, Foreign Key Brand referencing Car.

# Relationship types

- 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

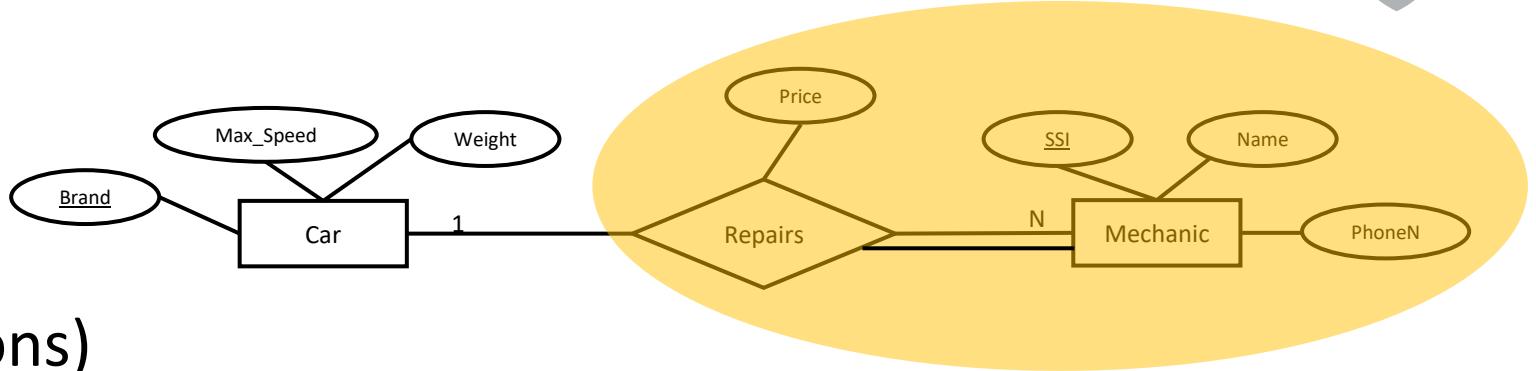
Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544
BMW 3.21	11	23761281	Alex	73828732

Car (Brand:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

Mec\_R (Brand:string,Price: integer, SSI:integer, Name:string, Phone\_Number:string ) IC's: Primary key SSI, Foreign Key Brand referencing Car.

# Relationship types

- 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

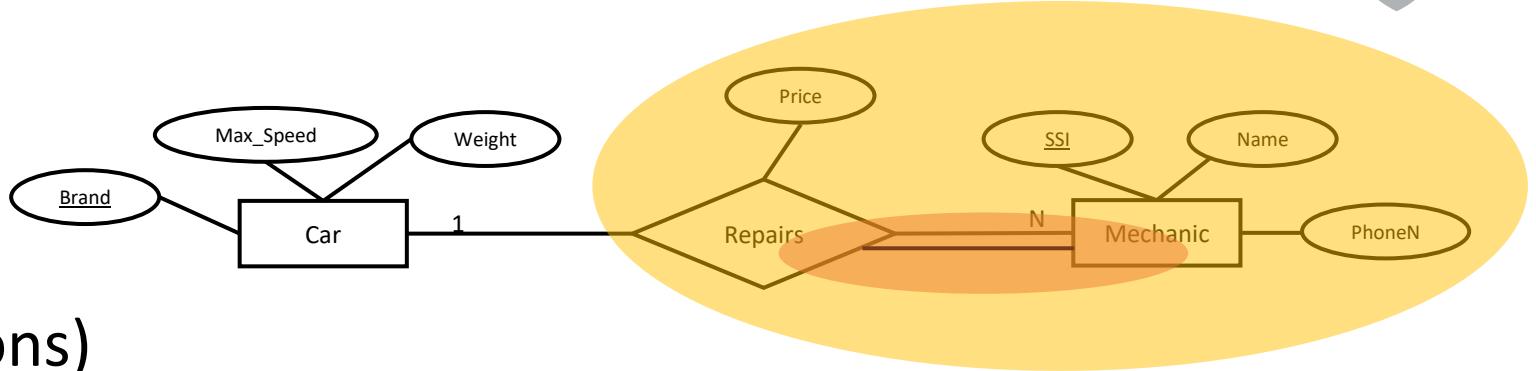
Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544
BMW 3.21	11	23761281	Alex	73828732

Car (Brand:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

Mec\_R (Brand:string,Price: integer, SSI:integer, Name:string, Phone\_Number:string ) IC's: Primary key SSI, Foreign Key Brand referencing Car.

# Relationship types

- 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

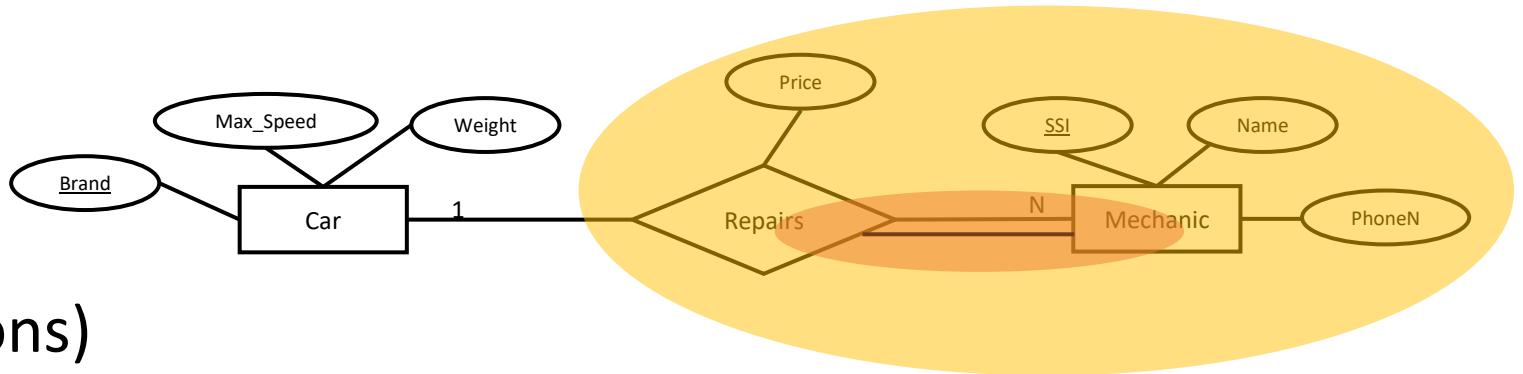
Brand	Price	SSI	Name	Phone_Number
BMW 3.21	10	87542702	Tom	75315567
Toyota_Corolla	23	68201937	Uraz	75335521
Hyundai E.GLS	12	23139827	Nick	75315544
BMW 3.21	11	23761281	Alex	73828732

Car (Brand:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

Mec\_R (Brand:string,Price: integer, SSI:integer, Name:string, Phone\_Number:string ) IC's: Primary key SSI, Foreign Key Brand referencing Car.

# Relationship types

- 1 to Many relations  
(Same for Many to 1 relations)



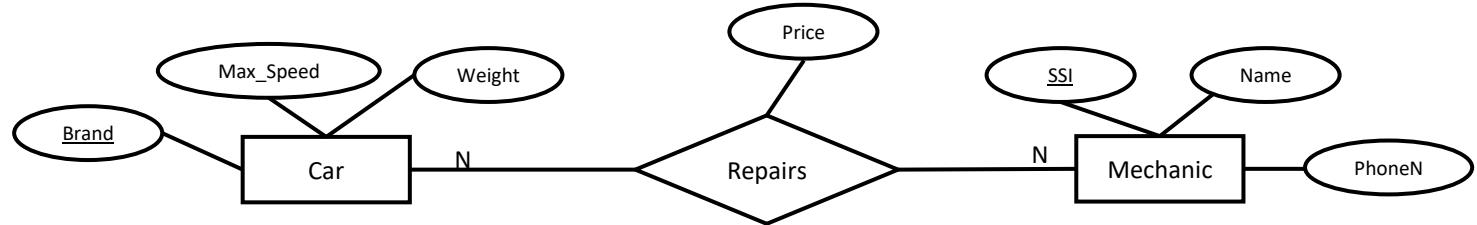
Brand	Weight	Length	Max_Speed
BMW 3.21	1400	3.21	200
Toyota_Corolla	1300	3.18	200
Hyundai E.GLS	1400	3.16	210

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:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

Mec\_R (Brand:string,Price: integer, SSI:integer, Name:string, Phone\_Number:string ) IC's: Primary key SSI, Foreign Key Brand referencing Car, On delete **CASCADE/REJECT, BRAND CANNOT BE NULL**

# Relationship types



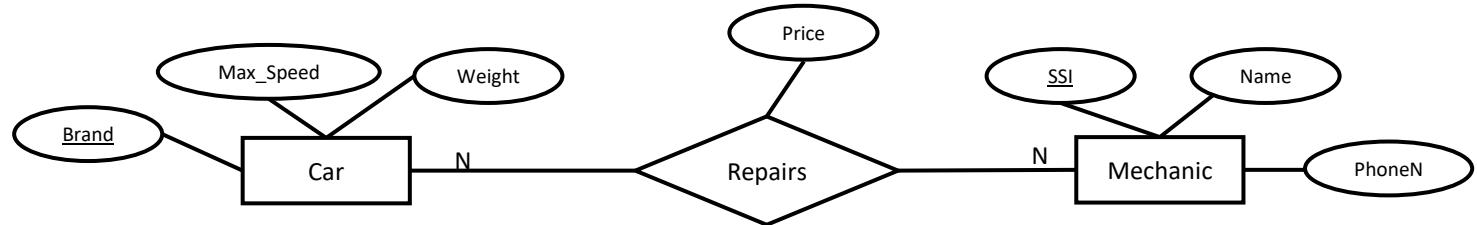
- Many to Many (N-N, N-M, X-Y,....)

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

Price
10
23
12

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

# Relationship types



- 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

Price
10
23
12

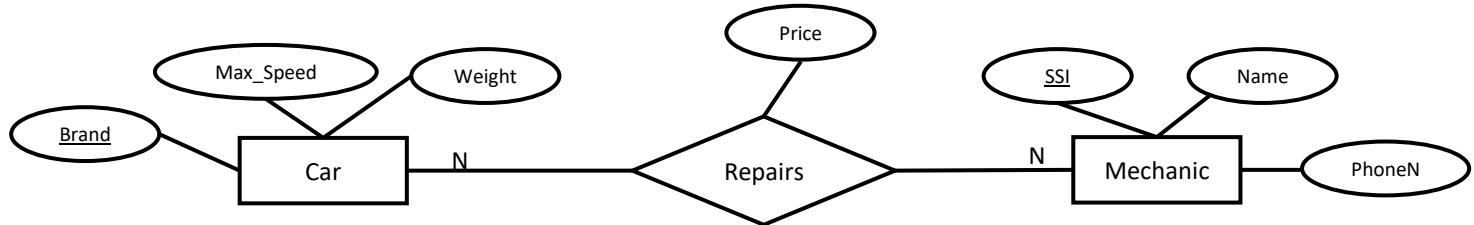
SSI	Name	Phone_Number
87542702	Tom	75315567
68201937	Uraz	75335521
23139827	Nick	75315544

Car (Brand:*string*, Weight: *integer*, Length: *real*, Max\_Speed: *integer*) IC: Primary key Brand.

Rep (Price: *Integer*).

Mec (SSI: *string*, Name: *string*, Phone\_Number: *string*,) IC: Primary key SSI.

# Relationship types



- 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:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

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

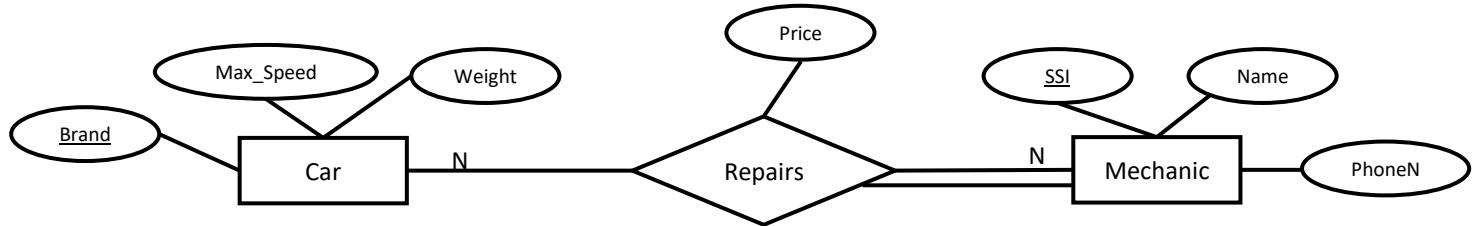
Rep(Price:Integer, SSI:integer, Brand:string).  
IC's: Primary Key {SSI,BRAND}  
Foreign Key SSI referencing Mec,  
Foreign Key Brand referencing Car.

Allows combinations!

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

Mec (SSI:string, Name:string, Phone\_Number:string,) IC: Primary key SSI.

# Relationship types



- 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:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

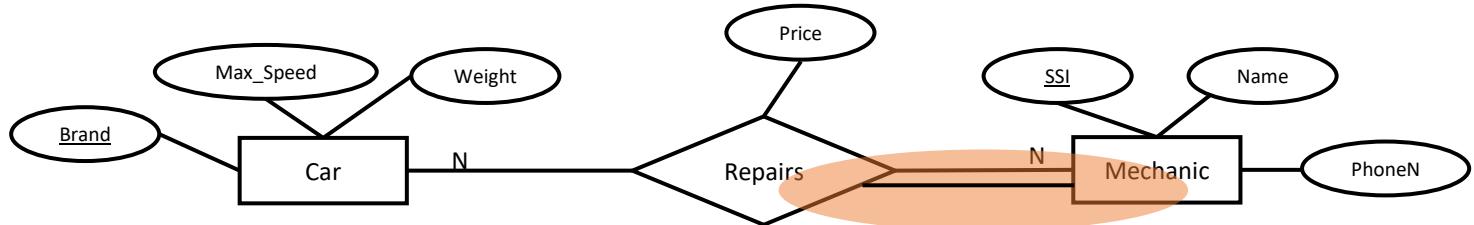
Price	SSI	Brand
10	87542702	BMW 3.21
23	68201937	Toyota_Corolla
12	23139827	Hyundai E.GLS

Rep(Price:Integer,SSI:integer,Brand:string).  
IC's: Primary Key {SSI,BRAND}  
Foreign Key SSI referencing Mec,  
Foreign Key Brand referencing Car.

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

Mec (SSI:string, Name:string, Phone\_Number:string,) IC: Primary key SSI.

# Relationship types



- 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:string,Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

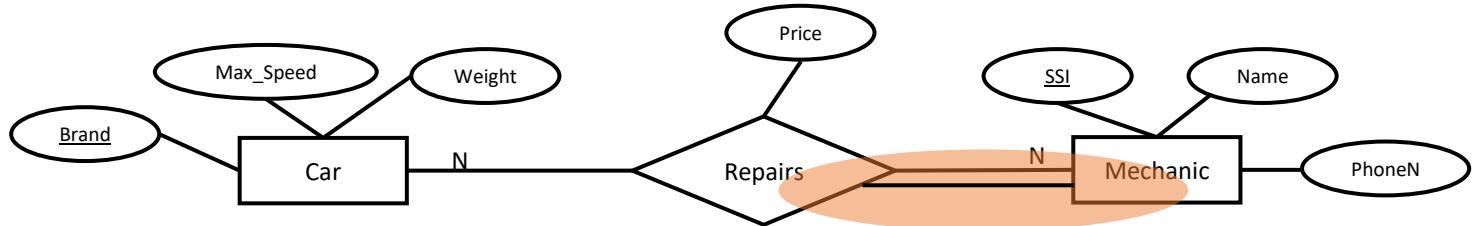
Price	SSI	Brand
10	87542702	BMW 3.21
23	68201937	Toyota_Corolla
12	23139827	Hyundai E.GLS

Rep(Price:Integer,SSI:integer,Brand:string).  
IC's: Primary Key {SSI,BRAND}  
Foreign Key SSI referencing Mec,  
Foreign Key Brand referencing Car

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

Mec (SSI:string, Name:string, Phone\_Number:string,) IC: Primary key SSI.

# Relationship types



- 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:string, Weight: integer, Length:real, Max\_Speed:integer) IC: Primary key Brand.

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

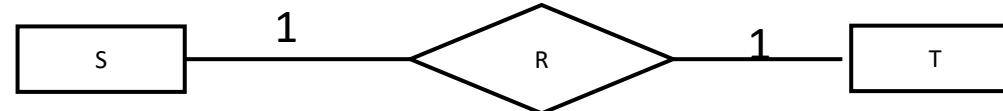
Rep(Price:Integer, SSI:integer, Brand:string).

IC's: Primary Key {SSI,BRAND}  
 Foreign Key SSI referencing Mec,  
 Foreign Key Brand referencing Car,  
**BRAND CANNOT NULL, ON DELETE CASCADE.**

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

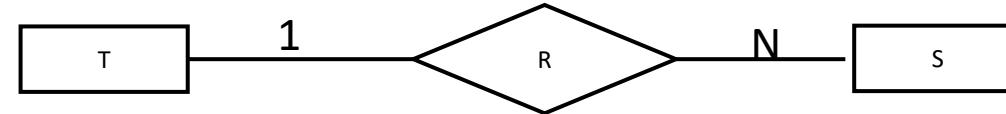
Mec (SSI:string, Name:string, Phone\_Number:string,) IC: Primary key SSI.

# ER-to-Relational Mapping (Rules)



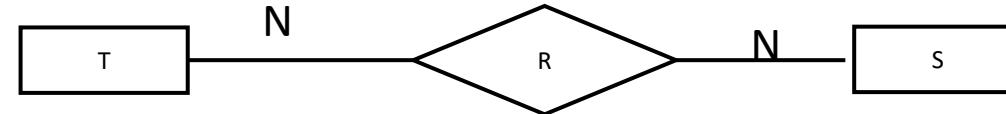
- *1:1 Relationships*
  - For each **1:1** binary relationship in ER schema, identify relations *S* and *T* (in relational schema) that correspond to entity types participating in relationship
  - Choose one of relations, for example, *S* and include primary key of *T* as foreign key in *S*

# ER-to-Relational Mapping (Rules)



- *1:N Relationships*
  - For each **1:N** binary relationship in ER schema, identify relation *S* (in relational schema) that represents participating entity at *N*-side of relationship
  - Include primary key of relation *T* as foreign key in *S* (where *T* represents entity at 1-side of relationship)
    - Each entity instance on *N*-side is related to at most one entity instance on 1-side (primary key constraint)

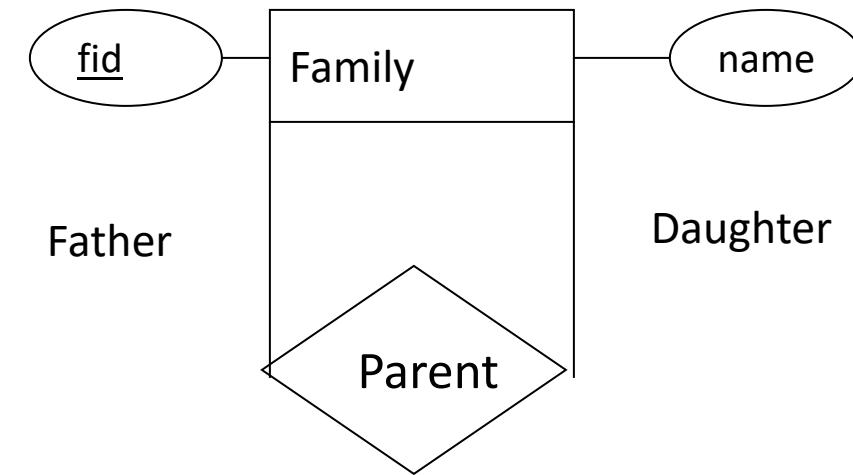
# ER-to-Relational Mapping (Rules)



- *M:N relationships*
  - For each **M:N** binary relationship in ER schema, create new relation  $R$  to represent many-to-many relationship
  - Include primary keys of relations that represent participating entity types as foreign key attributes in  $R$ 
    - The combination of primary keys of the two relations representing participating entities will form primary key of  $R$

# Unary relations

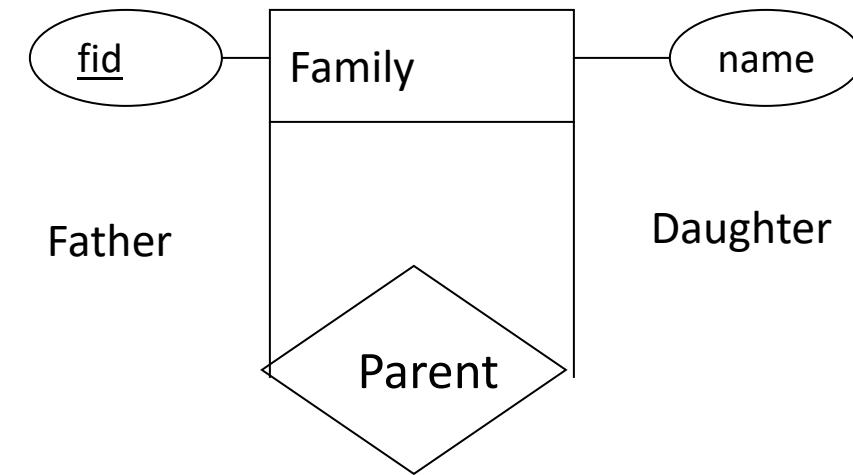
---



# Unary relations

---

Family(fid:*integer*, Name:*string*, fatherOf:*integer*, daughterOf:*integer*)

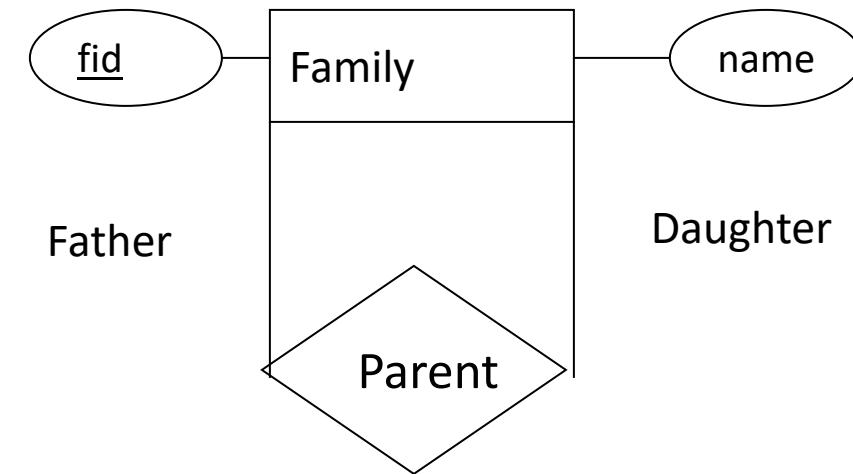


# Unary relations

---

Family(fid:*integer*, Name:*string*, fatherOf:*integer*, daughterOf:*integer*)

IC's: Primary key fid foreign key, fatherOf referencing Family. Foreign key daughterOf referencing Family.

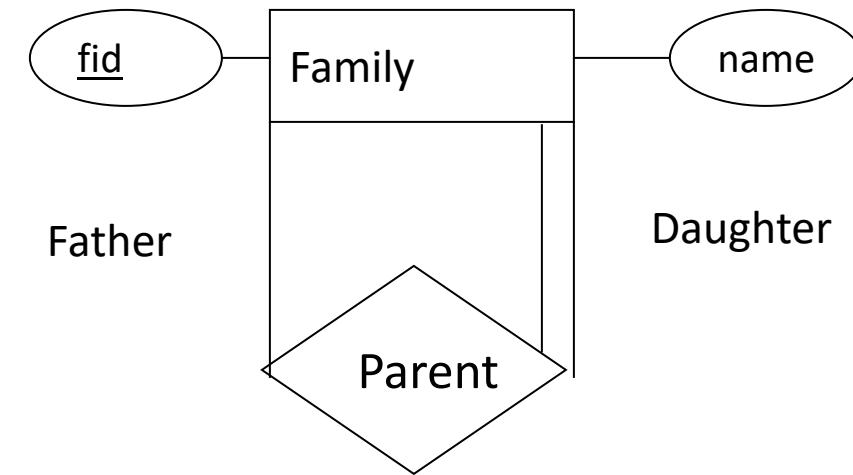


# Unary relations

---

Family(fid:*integer*, Name:*string*, fatherOf:*integer*, daughterOf:*integer*)

IC's: Primary key fid foreign key, fatherOf referencing Family. Foreign key daughterOf referencing Family.

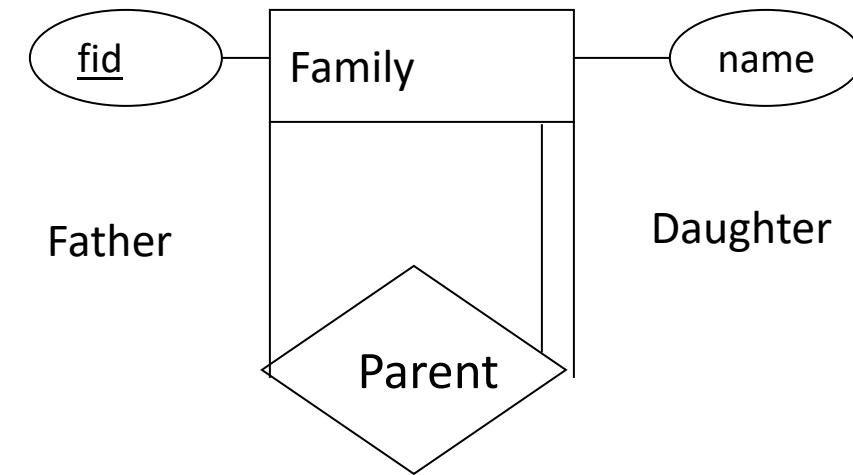


# Unary relations

---

Family(fid:*integer*, Name:*string*, fatherOf:*integer*, daughterOf:*integer*)

IC's: Primary key fid foreign key, fatherOf referencing Family. Foreign key daughterOf referencing Family, **daughterOf CANNOT Null**, **daughterOf is UNIQUE**.

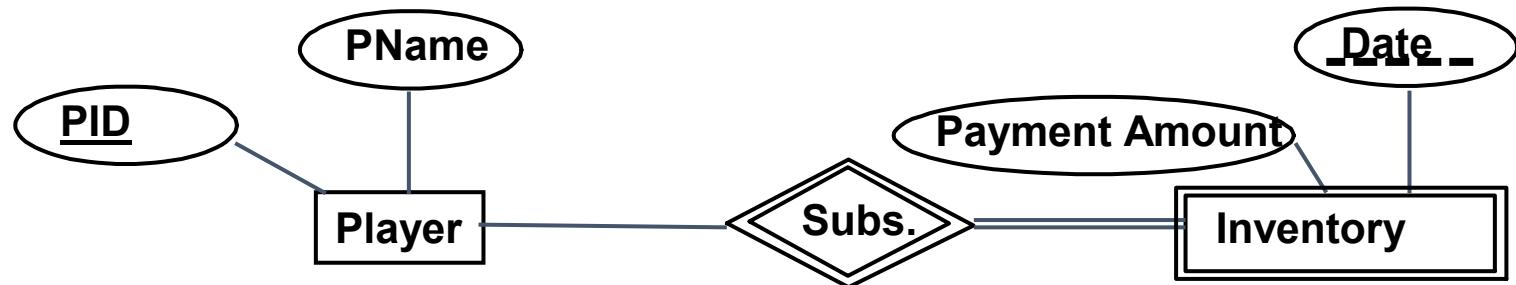


# Weak entity sets

---

Player(PID:integer,PName:string)

- 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:integer,PaymentAmount:string,Date:date)**

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

# Exercise

---

