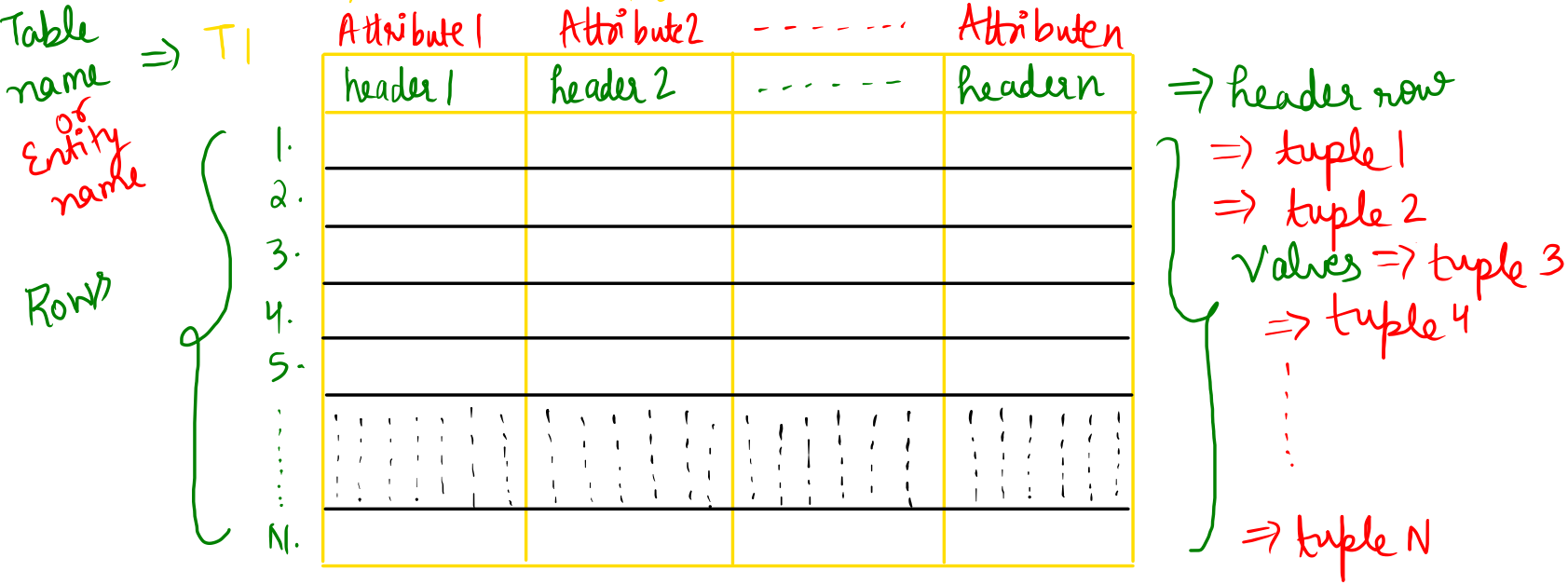
Relational Model

→ Entity type is represented in the form of relations.
→ A relation can be viewed as a table.



⇒ Each row in a table storing values corresponds to a tuple.

Table	=	Relation	= Entity type
Table name	=	Relation name	= Entity name
Header	=	Attribute	= Attribute
Rows	=	Tuples	= NA
Header Row	=	Set of attributes	= Entity Attributes

Domain

A domain defines a set of atomic values permissible corresponding to an attribute.

⇒ can be finite set.
⇒ can be infinite set.

Student

{ name , rollno , age , phone_num , grade }

dom (name) = $[(a-z) \,||\, (A-Z)]^*$ ⇒ Alphabets

dom (rollno) = [0-9] [0-9] [0-9] = 3 digits

dom (age) = { 3+ } = should be greater than 3.

dom (phone_num) = should be exactly 10 digits.

dom(grade) = { A, B, C, D, E, F } = should belong to any one of these values.

Cardinality

→ Cardinality of domain ⇒ Number of values present in a finite domain set.

→ Cardinality of relation ⇒ Number of tuples stored in a relation.

denoted by → |X|

|dom| ⇒ cardinality of domain
|relation| ⇒ cardinality of relation

A relation $r(R)$ is a mathematical relation of degree n on the domains $dom(A1)$, $dom(A2)$, ..., $dom(An)$, which is a subset of the Cartesian product (denoted by x) of the domains that define R:

$$r(R) \subseteq (dom(A1) \times dom(A2) \times \ldots \times (dom(An))$$

→ relation R is a subset of Cartesian product of domains set of attributes.

If we denote the total number of values, or cardinality, in a domain D by $|D|$ (assuming that all domains are finite), the total number of tuples in the Cartesian product is,

$$|dom(A1)| \times |dom(A2)| \times \ldots \times |dom(An)|$$

Characteristics of a Relation

→ Ordering of tuples in a relation

relation1

	A1	A2	A3	A4	A5	A6
R1	a_1	a_2	a_3	a_4	a_5	a_6
R2	b_1	b_2	b_3	b_4	b_5	b_6
R3	c_1	c_2	c_3	c_4	c_5	c_6
R4	d_1	d_2	d_3	d_4	d_5	d_6
R5	e_1	e_2	e_3	e_4	e_5	e_6
R6	f_1	f_2	f_3	f_4	f_5	f_6

----- There can be various permutations & combinations that are possible, however all of them are exactly same.

relation2

	A1	A2	A3	A4	A5	A6
R2	b_1	b_2	b_3	b_4	b_5	b_6
R3	c_1	c_2	c_3	c_4	c_5	c_6
R5	e_1	e_2	e_3	e_4	e_5	e_6
R1	a_1	a_2	a_3	a_4	a_5	a_6
R6	f_1	f_2	f_3	f_4	f_5	f_6
R4	d_1	d_2	d_3	d_4	d_5	d_6

----- relation 1 = relation 2 = ...

→ Ordering of values within a tuple.

relation1

	A1	A2	A3	A4	A5	A6
R1	a_1	a_2	a_3	a_4	a_5	a_6
R2	b_1	b_2	b_3	b_4	b_5	b_6
R3	c_1	c_2	c_3	c_4	c_5	c_6
R4	d_1	d_2	d_3	d_4	d_5	d_6
R5	e_1	e_2	e_3	e_4	e_5	e_6
R6	f_1	f_2	f_3	f_4	f_5	f_6

relation2

	A2	A5	A1	A3	A6	A4
R1	a_2	a_5	a_1	a_3	a_6	a_4
R2	b_2	b_5	b_1	b_3	b_6	b_4
R3	c_2	c_5	c_1	c_3	c_6	c_4
R4	d_2	d_5	d_1	d_3	d_6	d_4
R5	e_2	e_5	e_1	e_3	e_6	e_4
R6	f_2	f_5	f_1	f_3	f_6	f_4

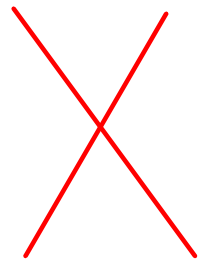
..... There can be various permutations & combinations that are possible, however all of them are exactly same.

relation1 = relation2 = ...

→ Values & Nulls in a tuple

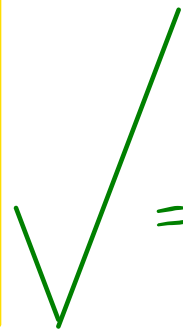
Each value in a tuple is an atomic value. A special value called NULL, is used when value unknown, value exists but is not available, or attribute does not apply to this tuple (also known as value undefined).

A1	A2	A3	A4
a_1	a_2	a_3	$\{a_{41}, a_{42}\}$
b_1	b_2	b_3	b_{41} b_{42}



⇒ This is not allowed as per relational model.

A1	A2	A3	A4	A5
a_1	a_2	a_3	a_{41}	NULL
a_1	a_2	a_3	a_{42}	NULL
b_1	b_2	b_3	b_{41}	b_{42}



⇒ This is correct way of representing attribute values.

Relational Model Constraints

→ Inherent model constraints or implicit constraints.
→ Schema-based constraints or explicit constraints.
→ Application-based or semantic constraints or business rules.

1. Employee's salary should not exceed his/her supervisor.
2. Total no. of hrs that an employee can work on all projects should not exceed 50 hrs per week.

Schema-based Constraints

- **Domain constraints** → Each attribute's value should be atomic & belong to that attribute's domain set.
  a) Data type b) Range c) List.

- **NULL constraints** → It specifies whether null values are permitted or not.

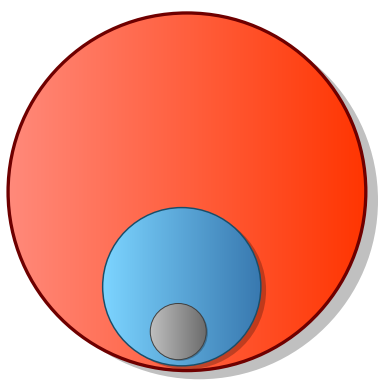
- **Key constraints** → • Uniqueness constraint

Student

name	age	gender	roll no	grades	phone_num

Set of all attributes = {name, age, gender, roll no, grades, phone_num}

→ Every tuple will be unique & can be used to identify a student.
→ There can be multiple subsets of attributes, which can be used to uniquely identify a tuple.
→ Each of those subsets is known as a superkey.
→ After removing redundant attributes → **minimal superkey = key.**



Red circle => set of superkeys

Blue circle => set of keys or set of candidate keys.

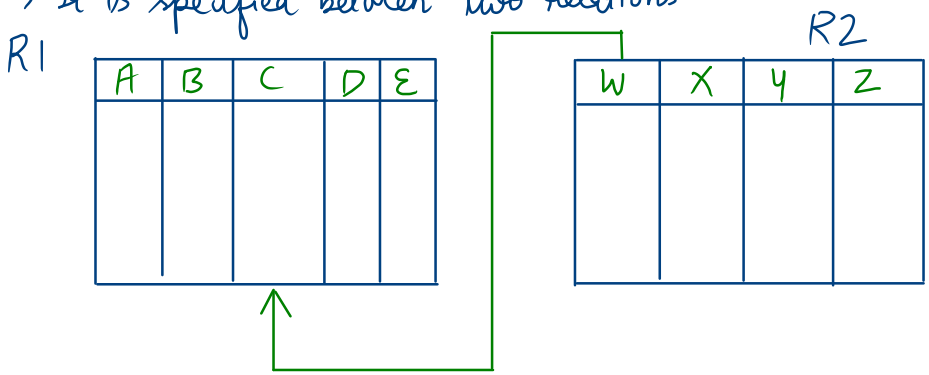
Grey circle => set of primary key having exactly one value.

Entity Integrity Constraints =>

→ It states that no primary key value can be NULL. * even if primary key is composite, no atomic attribute of P.K. can be NULL.

Referential Integrity Constraint =>

→ It is specified between two relations.



⇒ Attribute W of R2 refers or takes reference from attribute C of R1.

Conditions of foreign key constraint

→ Attribute C should be a primary key of relation R1 & attribute W will be referred to as the foreign key in relation R2.

→ Data types of both C & W should be same.

→ Values in W can be NULL and duplicate, however this is not true for values in C.

R1 = Referenced relation.
R2 = Referencing relation.

Constraint Violations

Whenever any operation on database is performed which changes its state, Integrity constraints should not be violated.

- Insert
- update
- delete

} only 3 operations can change any database's state.

Constraint violations due to Insert operation

→ domain constraint violation.

It can happen if any attribute value is given and that does not appear in the corresponding domain or is not of the appropriate data type.

Example → Inserting 'Z' in grade attribute of student relation.
or
Inserting alphabets in phone_num attribute of student relation.

→ Key constraint violation

If a key value in the new tuple already exists in the relation.

Example → Inserting student with roll no. 5, however another student with same roll no. already exists in the student relation.

→ Entity Integrity violation

Inserting NULL in any part of primary key attributes.

Example → Let's suppose RollNo & phone number together forms the primary key in student relation. So whenever we try to put NULL in any of the attributes RollNo. or phone number, it will result in entity integrity violation.

→ Referential Integrity violation

It can be violated if the value of any foreign key refers to a tuple that does not exist in the referenced relation.

Employee

name	SSN	.....	DNUM
amit	1234	..	2
arun	5678	..	2
Sonu	9918	..	1
Riya	1203	..	4

Department

DNO	DNAME	DLOCATIONS
1.	HR	Noida
2.	Tech	NCR
3.	Finance	Mumbai
4.	Sales	Pune

Trying to insert a new tuple with values

{Rohan, 6859, ..., 5} ⇒ It will violate referential integrity.

Dealing with Insertion constraint violation

1. Rejection with no reason.
2. Correct the reason for rejecting the insertion.
3. DBMS can ask the user to change the value which could lead to cascade back.

Constraint violation due to delete operation

→ Referential Integrity Violation

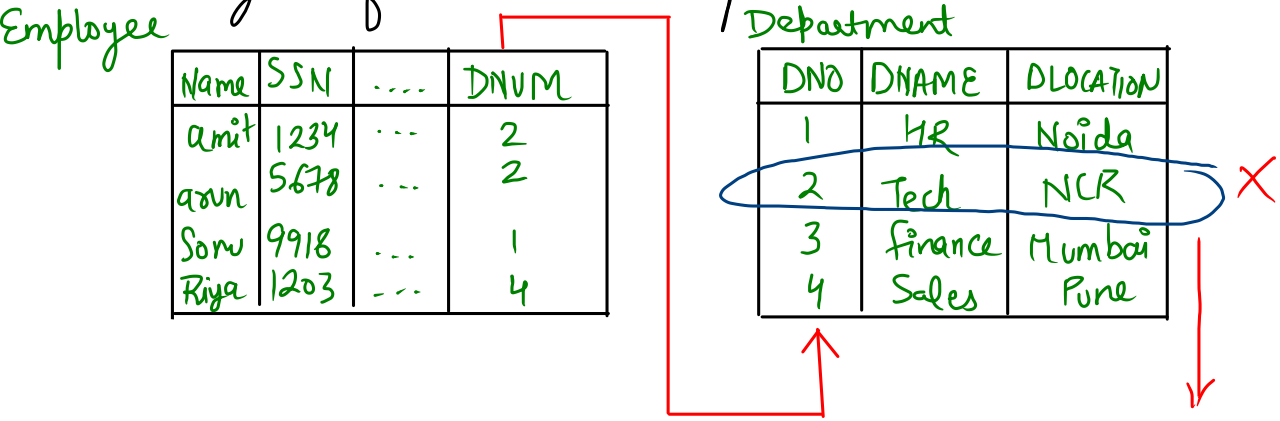
If the tuple being deleted is referenced by foreign keys from other tuples in the database.

Employee

Name	SSN	....	DNUM
amit	1234	...	2
arun	5678	...	2
Sonu	9918	...	1
Riya	1203	...	4

Department

DNO	DNAME	DLOCATION
1	HR	Noida
2	Tech	NCR
3	finance	Mumbai
4	Sales	Pune



If we try to delete this tuple then it will result in R.E. constraint violation.

Dealing with deletion constraint violation

1. Set NULL or Set default.
2. Restrict.
3. Cascade delete.

Constraint violation due to update operation

→ Update operation can be viewed as deletion followed by insertion.

→ Hence, it will be applicable as we read in previous slides.