Introduction to Relational Model

Dr. GEETA KASANA

Assistant Professor

Thapar Institute Of Engineering and Technology

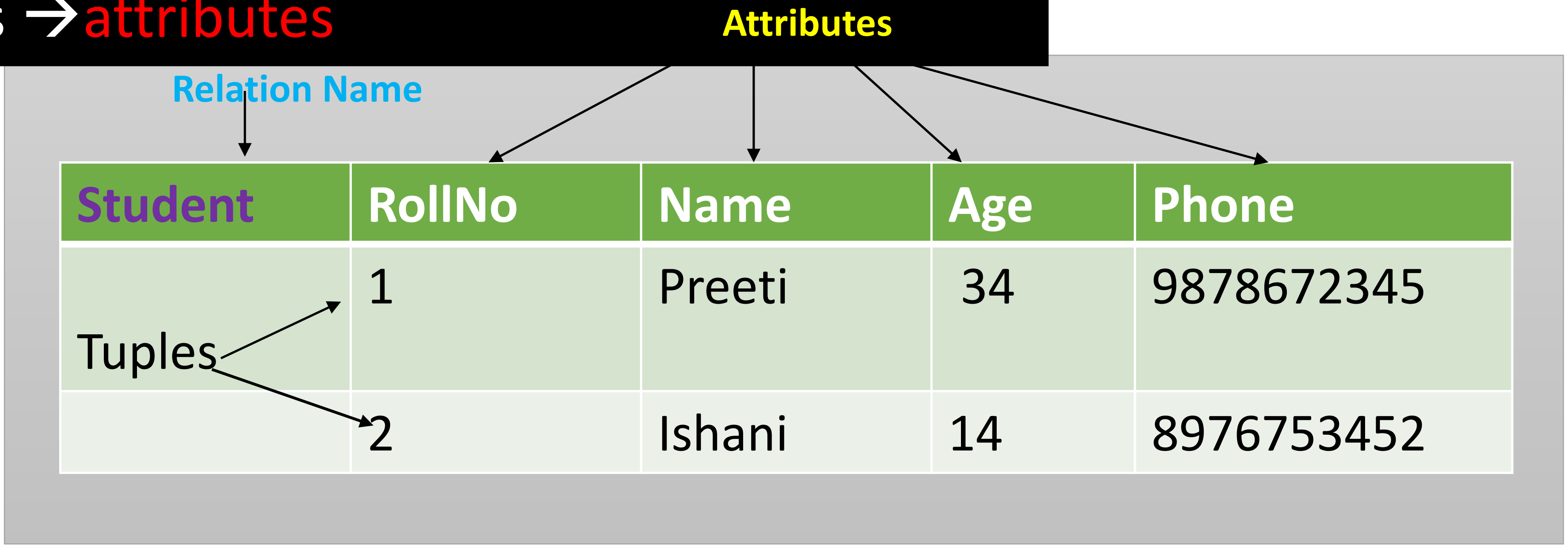
Patiala

History of Relational Data Model

- First introduces by Ted Codd (in 1970)
- Uses Concept of mathematical relation
- First commercial implementations of the relational Model->
- Oracle DBMS,SQL/DS system by IBM
- Current popular RDBMS s->SQL Server & Access (Microsoft), DB2 & Informix(IBM) etc
- Standard for commercial RDBMS –>SQL query Language

❖ Terminologies

- Relational Model represents data as collection of tables.
- A Table is also called a relation
- Each row → tuple
- Column headers →attributes



Attributes

Relation Name

Student	RollNo	Name	Age	Phone
Tuples	1	Preeti	34	9878672345
	2	Ishani	14	8976753452

❖ Terminologies

➢Domain:

- A set of atomic values allowed for an attribute

    - Eg: Employees ages: Possible ages of employees of a company (values between 20 & 70 year old)

❖ Terminologies

➢ Relation Schema:

- Describes a relation
- Made up of a relation name R and a having a list of attributes A1,A2,A3…………….An

STUDENT(Name,Rollno,Age,Address,Phone,Grade)

STUDENT(Name string:,Rollno:integer,Age:integer,Address:string,Phone:integer,Grade: string)

Relational Data Model

- Degree (or arity of a relation):
  - Number of attributes in a relation schema

STUDENT(Name,Rollno,Age,Address,Phone,Grade)

Terminologies

- **Cardinality:**
  - Total number of tuples present in a relation schema

Cardinality =3

STUDENT	ROLL_No	NAME	AGE	PHONE
	1	Preeti	34	9878672345
	2	Ishani	14	8976753452
	3	Sonal	18	7895354615

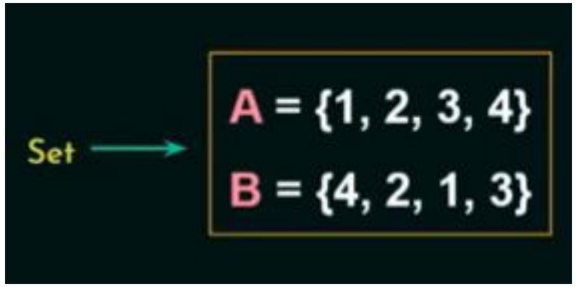
Terminologies

- Relational database Schema:
  - Is a set of relation schemas and a set of integrity constraints.

- Relational State(or Relation Instance:
  - Is a set of tuples at a given moment of time

Characteristics of Relation

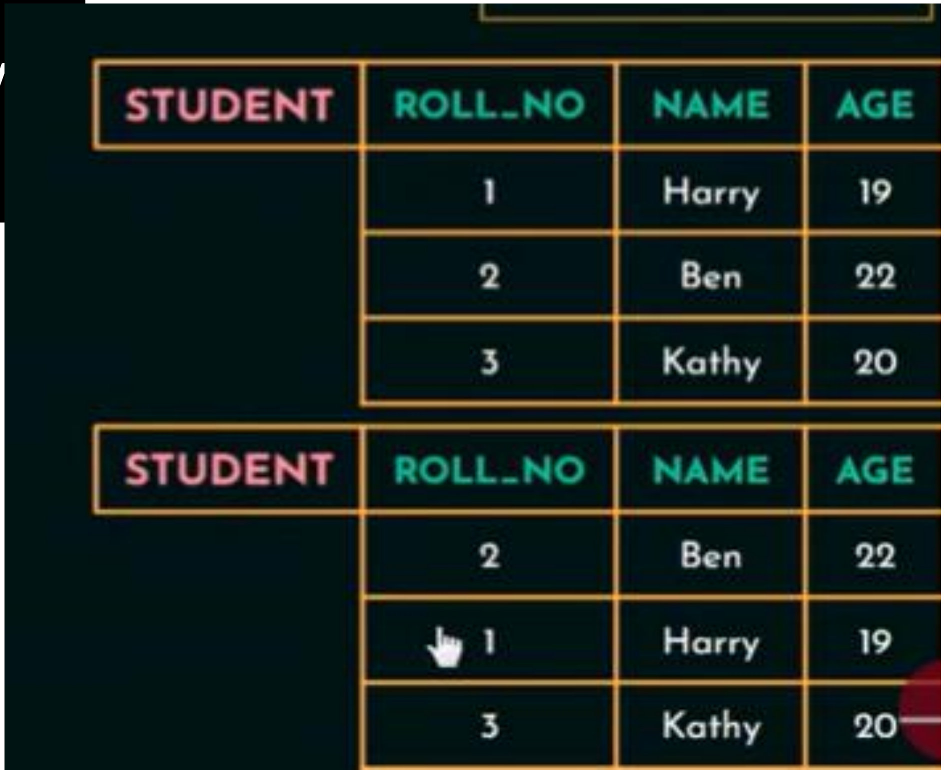
- Ordering of Tuples within a Relation:
  - A relation is a set of tuples .
  - Tuples in a relation need not have any particular order.



Set →

A = {1, 2, 3, 4}

B = {4, 2, 1, 3}

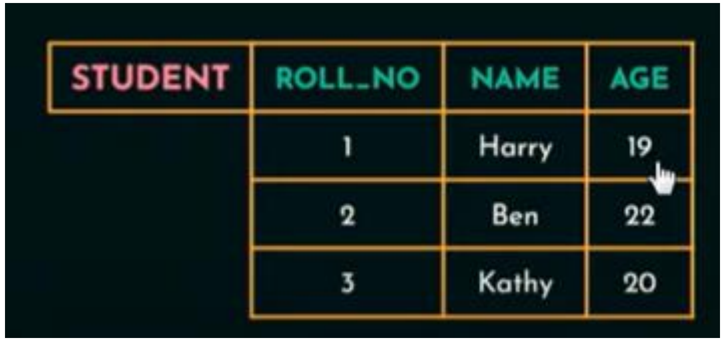


STUDENT	ROLL_NO	NAME	AGE
	1	Harry	19
	2	Ben	22
	3	Kathy	20

STUDENT	ROLL_NO	NAME	AGE
	2	Ben	22
	1	Harry	19
	3	Kathy	20

Characteristics of Relation

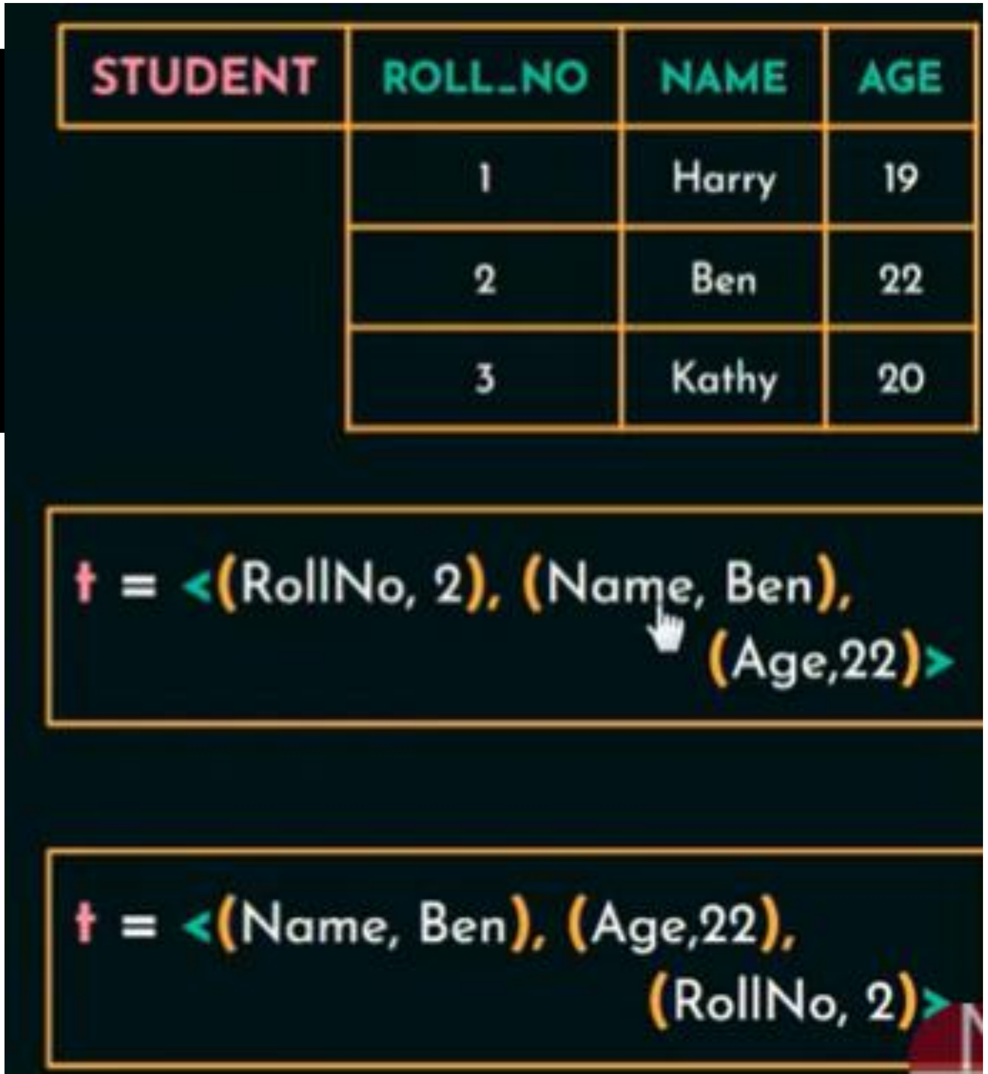
- Ordering of Values within a Tuple:
  - An n-tuple -> ordered list of n values , so ordering of values in a tuple is important.
  - With an alternative definition of relation , ordering of values in a tuple is unnecessary



STUDENT	ROLL_NO	NAME	AGE
	1	Harry	19
	2	Ben	22
	3	Kathy	20

Characteristics of Relation

- Ordering of Values within a Tuple:
  - A tuple -> set of (<attribute>, <value>) pair , then ordering of attribute is not important



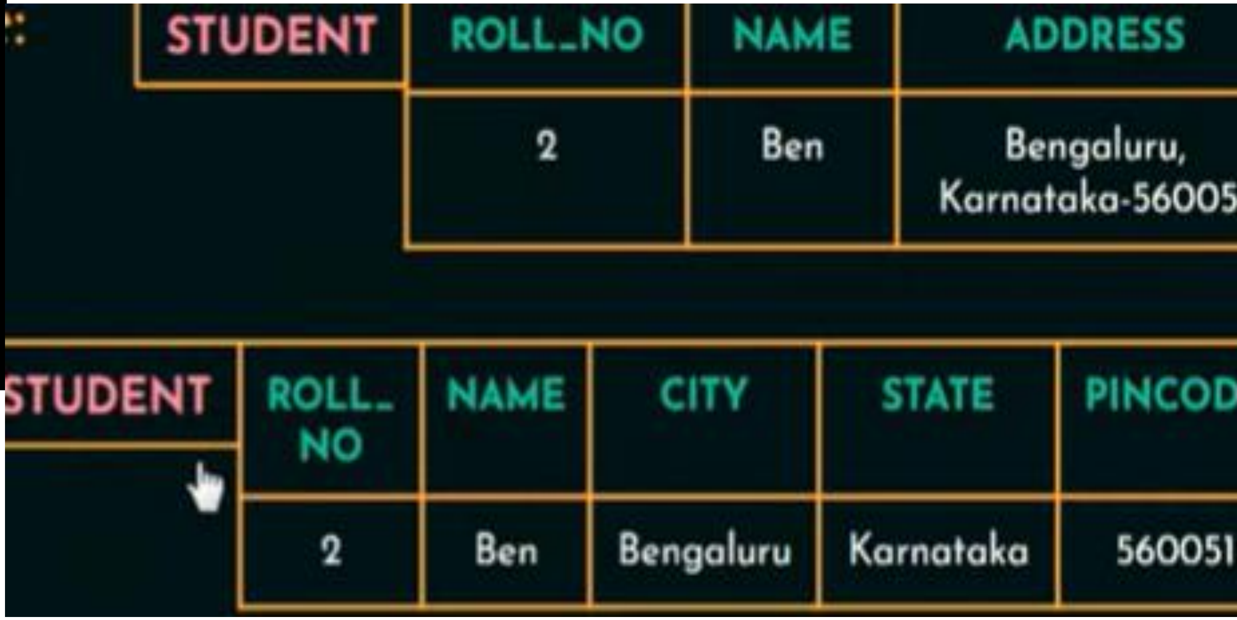
STUDENT	ROLL_NO	NAME	AGE
	1	Harry	19
	2	Ben	22
	3	Kathy	20

t = <(RollNo, 2), (Name, Ben), (Age,22)>

t = <(Name, Ben), (Age,22), (RollNo, 2)>

Characteristics of Relation

- Values & Nulls in a Tuple:
  - Each value in a tuple is an atomic value i,e it does not have composite values . We should have FLAT
  - Nulls : unknown or not applicable

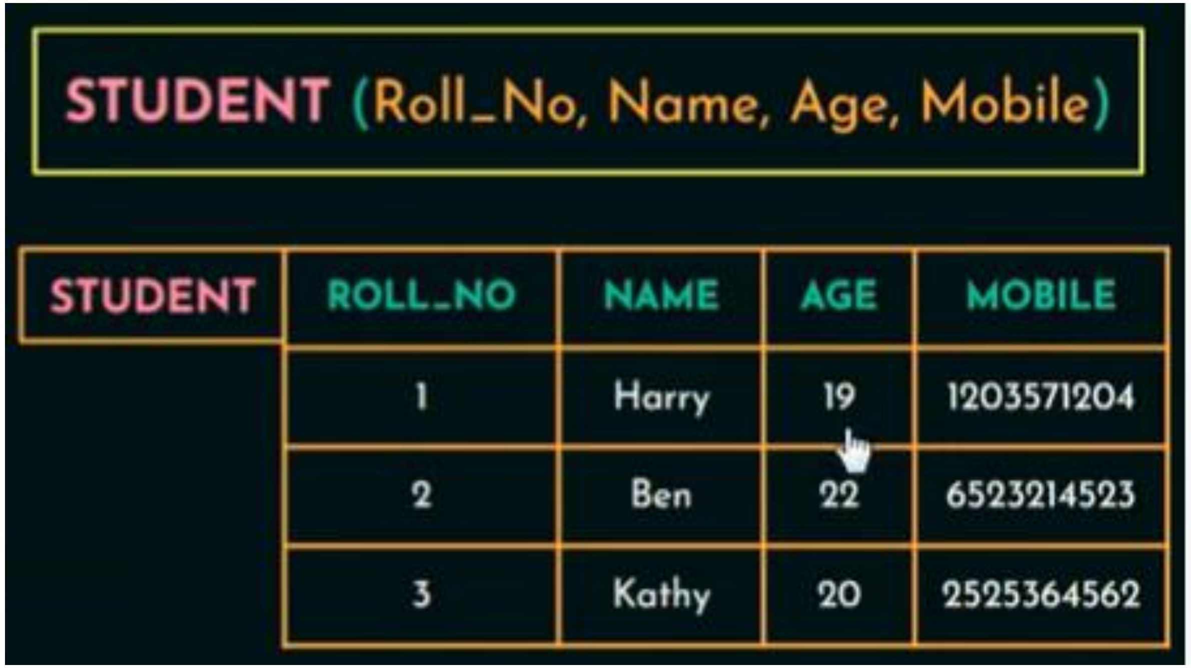


STUDENT	ROLL_NO	NAME	ADDRESS
	2	Ben	Bengaluru, Karnataka-56005

STUDENT	ROLL_ NO	NAME	CITY	STATE	PINCOD
	2	Ben	Bengaluru	Karnataka	560051

Characteristics of Relation

- **Interpretation of a relation:**
  - The relation schema can be represented as a declaration or assertion
  - Each tuple can be interpreted as fact



STUDENT (Roll_No, Name, Age, Mobile)

STUDENT	ROLL_NO	NAME	AGE	MOBILE
	1	Harry	19	1203571204
	2	Ben	22	6523214523
	3	Kathy	20	2525364562

KEYS of Relation

- PRIMARY KEY
- CANDIDATE KEY
- ALTERNATE KEY
- COMPOSITE KEY
- FOREGIN KEY
- SUPER KEY

Relational Model Constraints

- Constraints on databases:
  - Inherent Model - Based: Inherent in the model(Already existing)
  - Eg: duplicate records are not allowed
- Schema based:
  - Defined directly in the schemas of the data model
    - Eg: Age of an employee should be between 25-65
- Application based:
  - Must be expressed and enforced by the application programs.

CONSTRAINTS

The Constraints can be placed at the column level or table level.

Column Level Constraint: These constraints are defined along with the column definition . These constraints can be applied to any one column at a time. If the constraints spans across multiple columns , then the table level constraints are used.

Table Level Constraint: If the data constraint attached to a specific cell in a table references the content of another cell in the table then the table level constraint is used.

Relational Model Constraints: Schema Based Constraints

- Domain Constraints
- Key Constraints
- Constraints on NULL
- Entity Integrity Constraint
- Referential Integrity Constraint

Schema Based Constraints

Domain Constraints

- It specifies that within each tuple or within each row the value of an attribute has to be atomic or individual.
- Performs the datatype check of each attribute.

STUDENT	ROLL_No	NAME	AGE
	1	Preeti	34
	2	Ishani	14
	3	Sonal	A

Violates Domain Constraint

Schema Based Constraints

- Key Constraints
  - An attribute that can uniquely identify each tuple in a relation is called a Key

STUDENT	ROLL_No	NAME	AGE
	1	Preeti	34
	2	Ishani	14
	3	Ishani	14

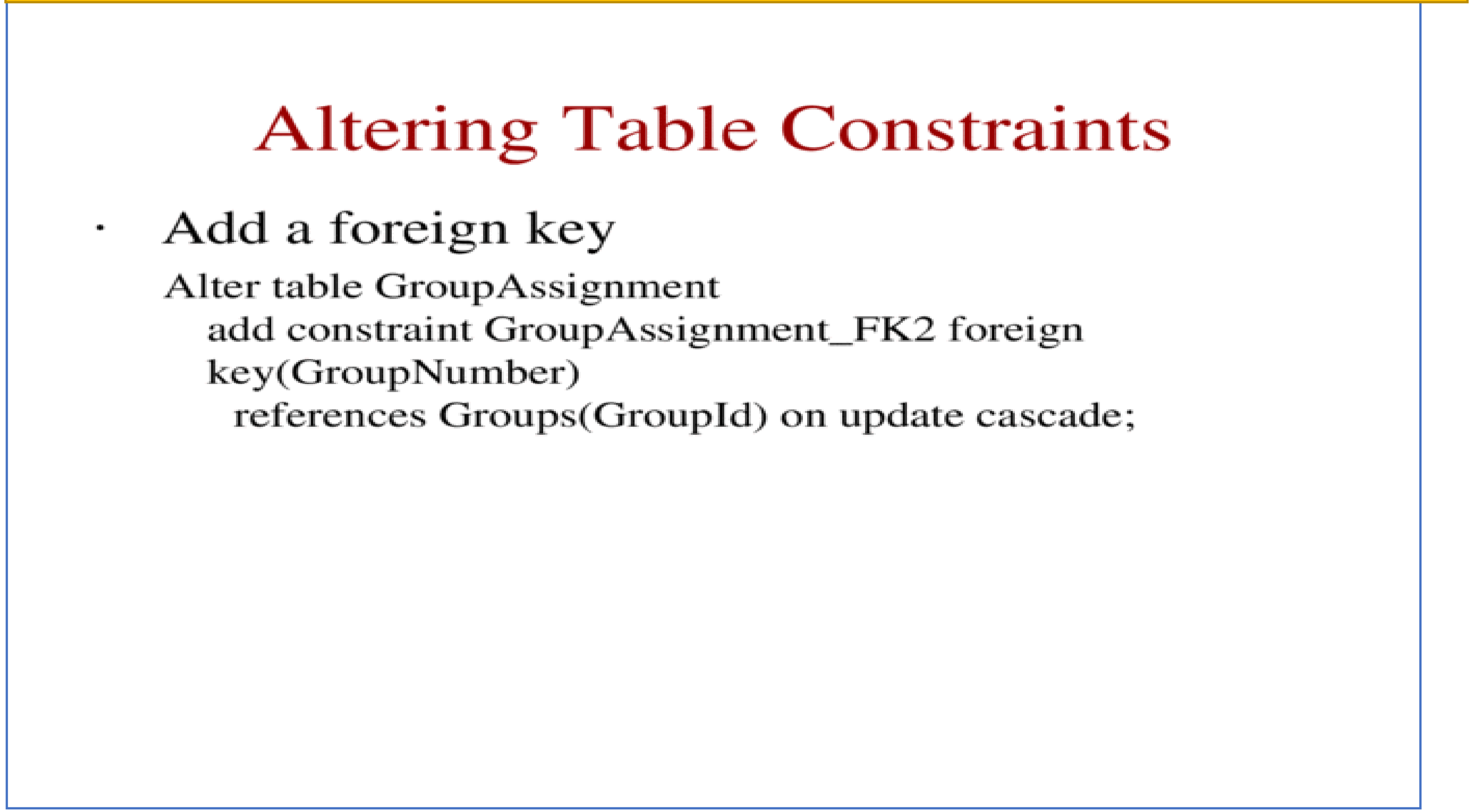
Key Constraints

There are a number of key constraints in SQL that ensure that an entity or record is uniquely or differently identified in the database. There can be more than one key in the table but it can have only one primary key.

Some of the key constraints in SQL are :

- Primary Key Constraint
- Foreign Key Constraint
- Unique Key Constraint

CONSTRAINTS



Altering Table Constraints

- Add a foreign key

```
Alter table GroupAssignment
   add constraint GroupAssignment_FK2 foreign
   key(GroupNumber)
     references Groups(GroupId) on update cascade;
```

Add/Remove Constraints

- After you create a table, you can use the Alter Table statement to
  - add or remove a primary key, unique, foreign key, or check constraint

- To drop a table's primary key constraint, just specify the Primary Key keywords:

```
Alter Table Sale
    Drop Primary Key
```

- To drop a unique, foreign key, or check constraint, you must specify the constraint name:

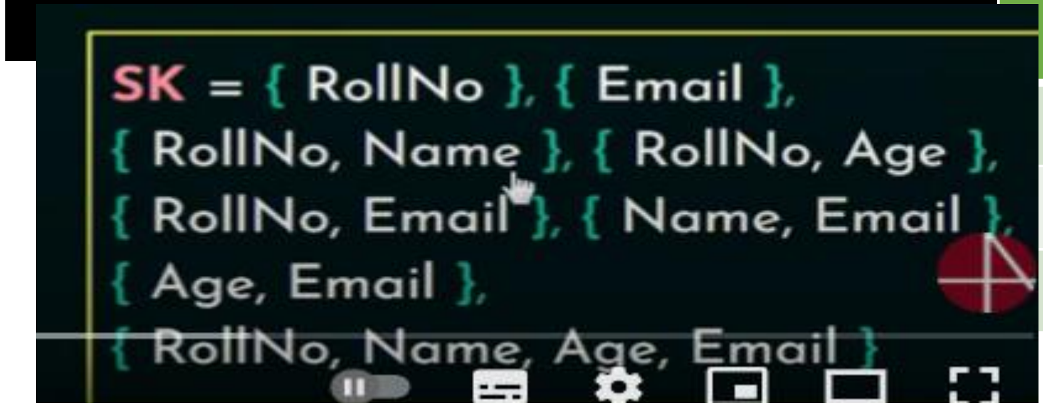
```
Alter Table Sale
    Drop Constraint SaleCustomerFK
```

- To add a new constraint, use the same constraint syntax as in a Create Table statement:

```
Alter Table Sale
    Add Constraint SaleSaleTotChk Check( SaleTot >= 0 )
```

Schema Based Constraints

- Super Key Constraints
  - A super key specifies that no two tuples can have the same value
  - Every relation has at least one superkey – set of all attributes.

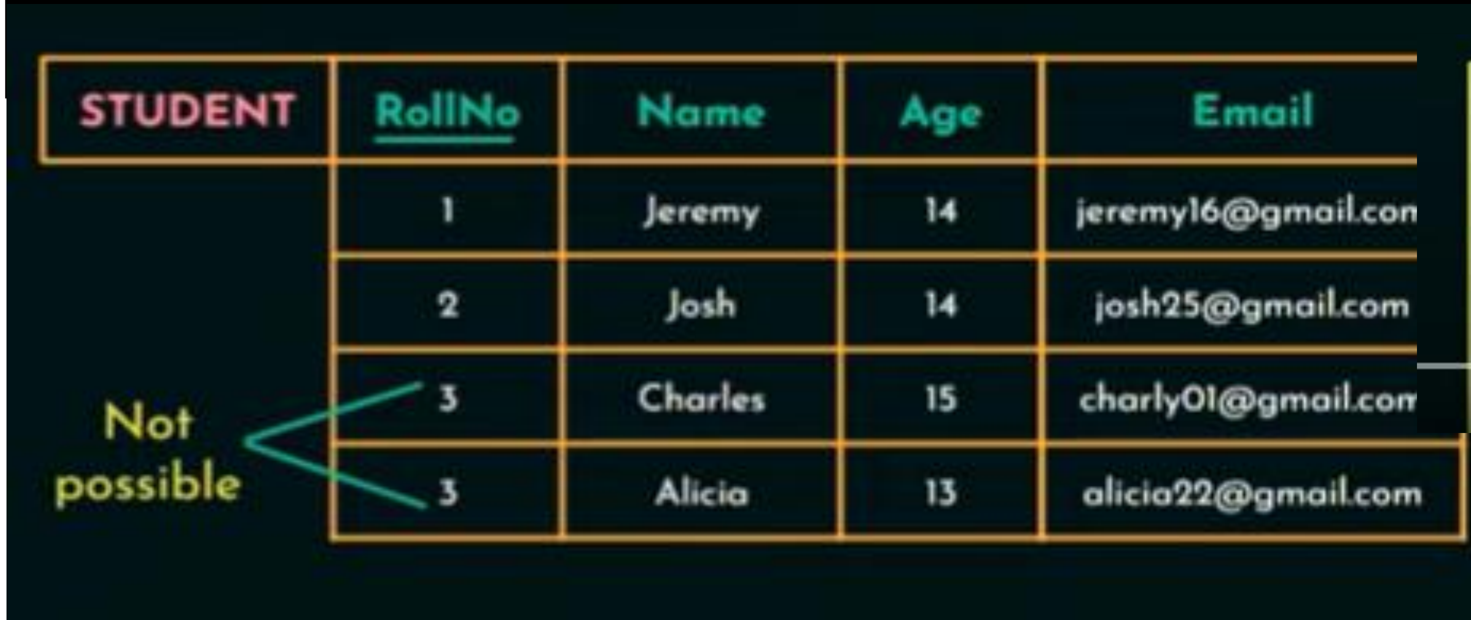


SK = { RollNo }, { Email },
{ RollNo, Name }, { RollNo, Age },
{ RollNo, Email }, { Name, Email },
{ Age, Email },
{ RollNo, Name, Age, Email }

STUDENT	ROLL_No	NAME	AGE	Mail_Id
	1	Preeti	34	preeti@gmail.com
	2	Ishani	14	ishani@gmail.com
	3	Sonal	14	sonal@gmail.com

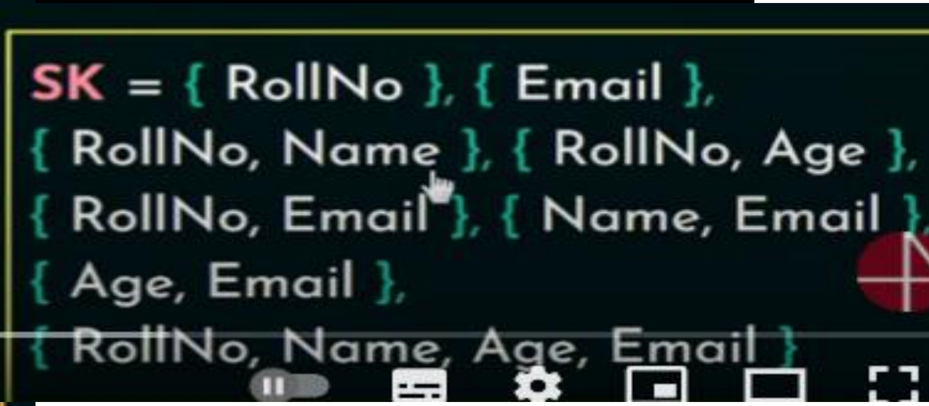
Schema Based Constraints

- A Key Constraints satisfies 2 constraints:
  - Two tuples cannot have identical values for all the attributes in the key.
  - It is minimal superkey



STUDENT	RollNo	Name	Age	Email
	1	Jeremy	14	jeremy16@gmail.con
	2	Josh	14	josh25@gmail.com
	3	Charles	15	charly01@gmail.con
	3	Alicia	13	alicia22@gmail.com

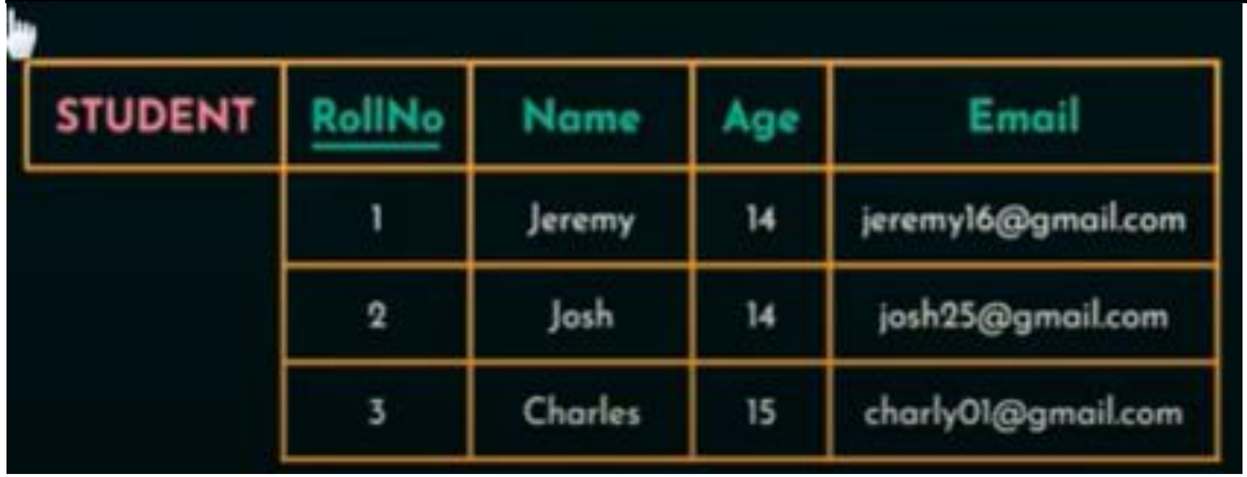
Not possible



SK = { RollNo }, { Email }, { RollNo, Name }, { RollNo, Age }, { RollNo, Email }, { Name, Email }, { Age, Email }, { RollNo, Name, Age, Email }

Schema Based Constraints

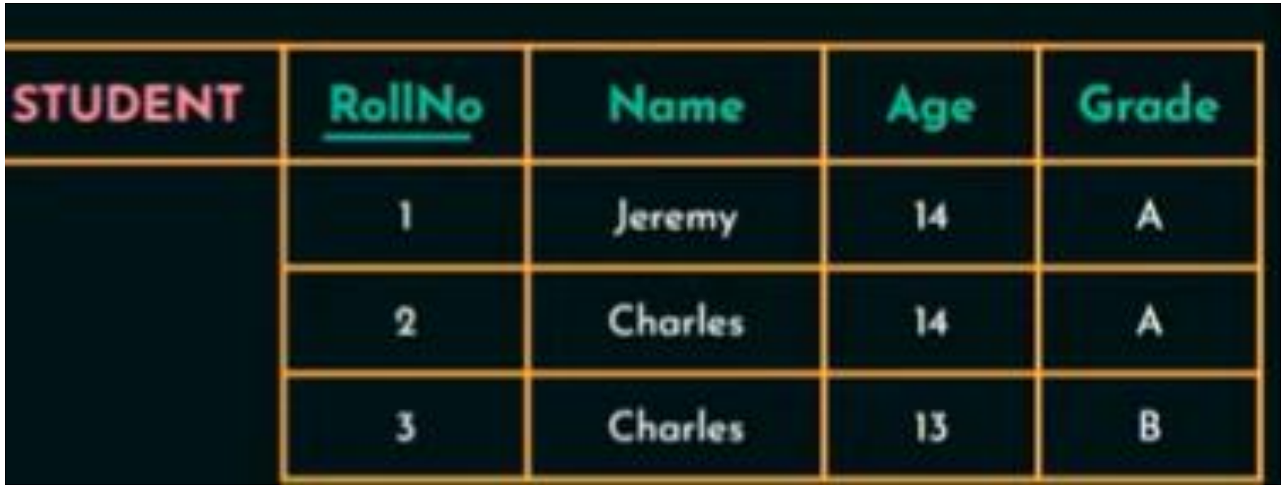
- Candidate Keys:
  - Set of attributes that uniquely identify the tuples in a relation.



STUDENT	RollNo	Name	Age	Email
	1	Jeremy	14	jeremy16@gmail.com
	2	Josh	14	josh25@gmail.com
	3	Charles	15	charly01@gmail.com

Schema Based Constraints

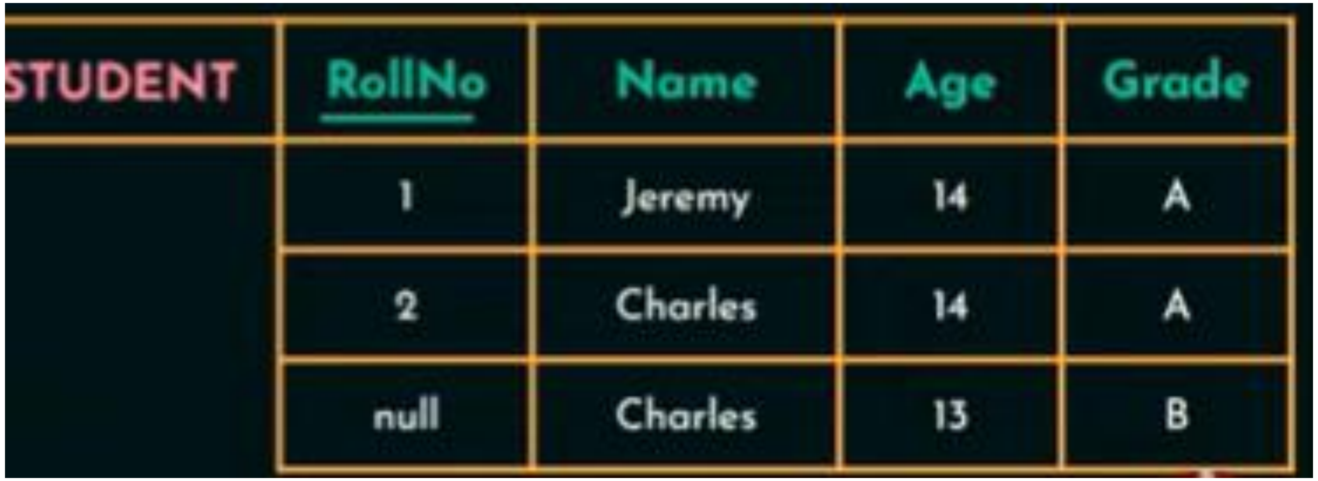
- Constraints on NULL Values
    - Specifies whether null values are permitted or not(NOT NULL)



STUDENT	RollNo	Name	Age	Grade
	1	Jeremy	14	A
	2	Charles	14	A
	3	Charles	13	B

Schema Based Constraints

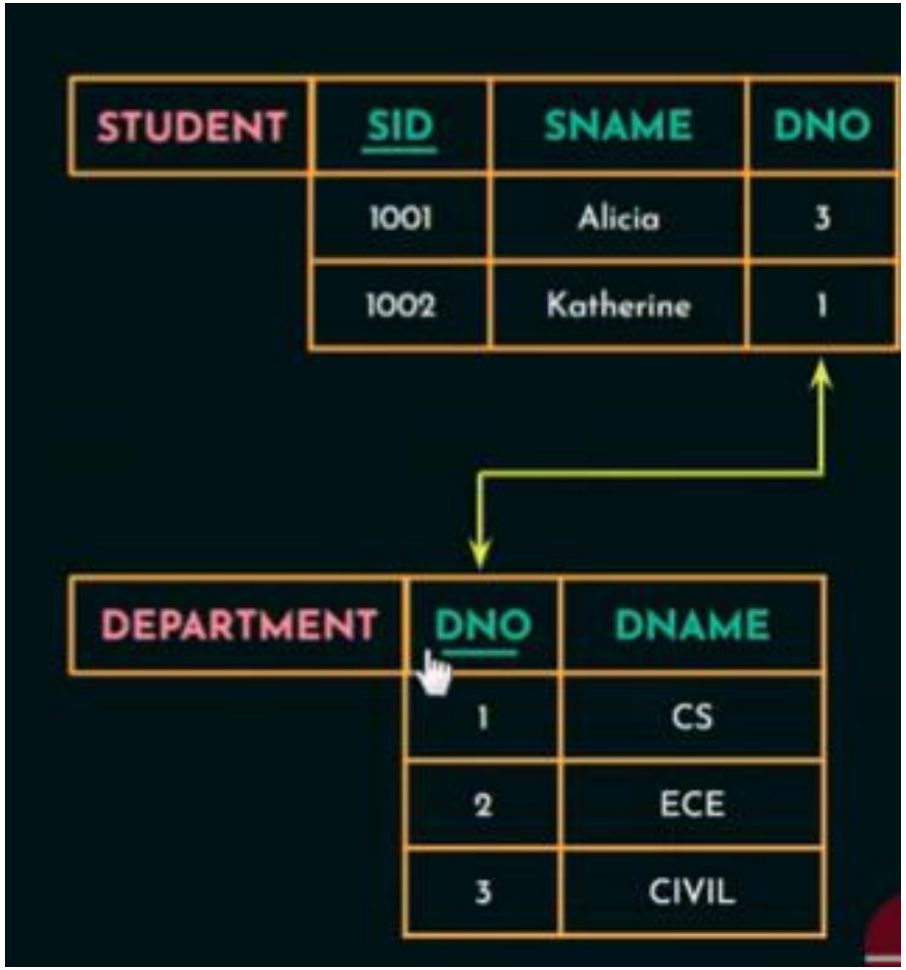
- Entity Integrity Constraint:
  - States that no primary key value can be null



STUDENT	RollNo	Name	Age	Grade
	1	Jeremy	14	A
	2	Charles	14	A
	null	Charles	13	B

Schema Based Constraints

- Referential Integrity Constraint:
  - Specified between two relations
  - States that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

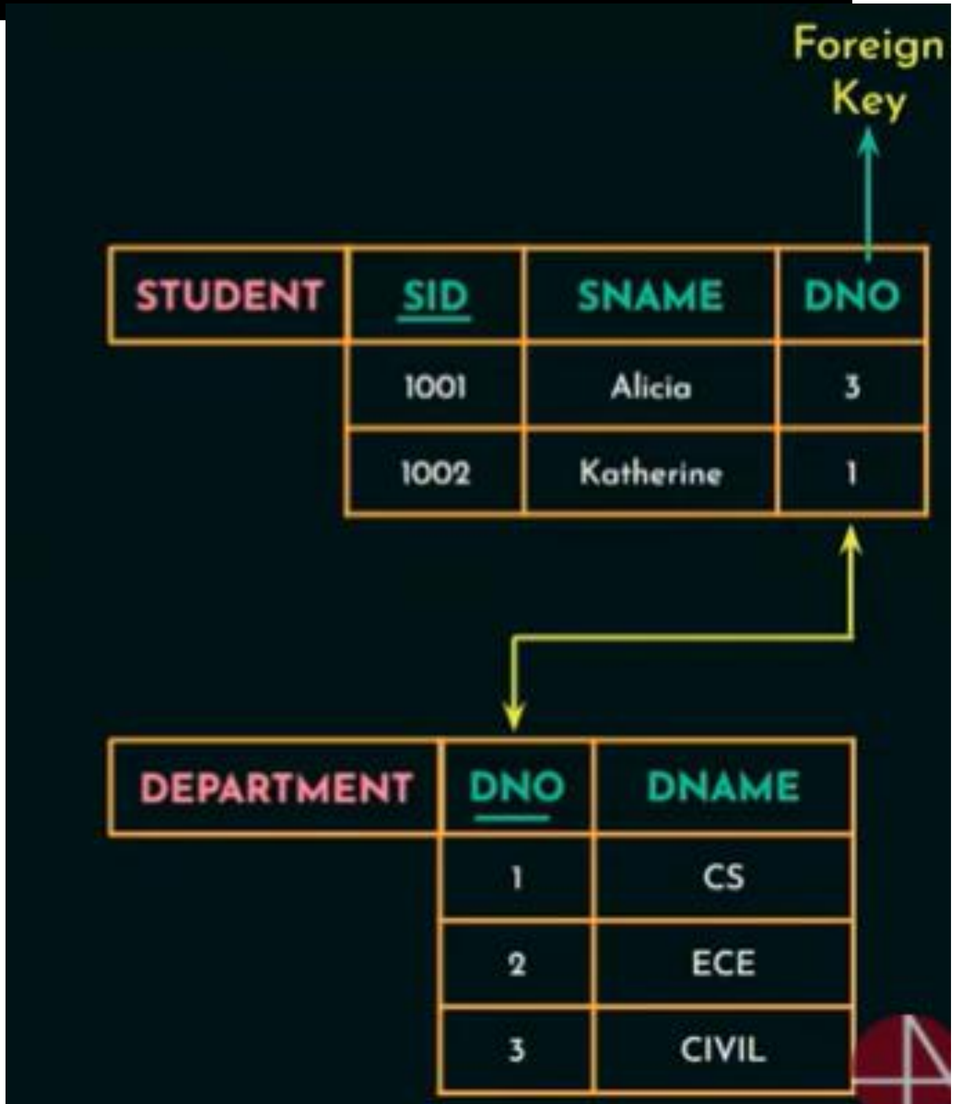


STUDENT	SID	SNAME	DNO
	1001	Alicia	3
	1002	Katherine	1

DEPARTMENT	DNO	DNAME
	1	CS
	2	ECE
	3	CIVIL

Schema Based Constraints

- Foreign Key must satisfy the following:
  - Same domain
  - Value of FK in a tuple either occurs as a value of PK i.e t1[FK]=t2[PK] or isnull



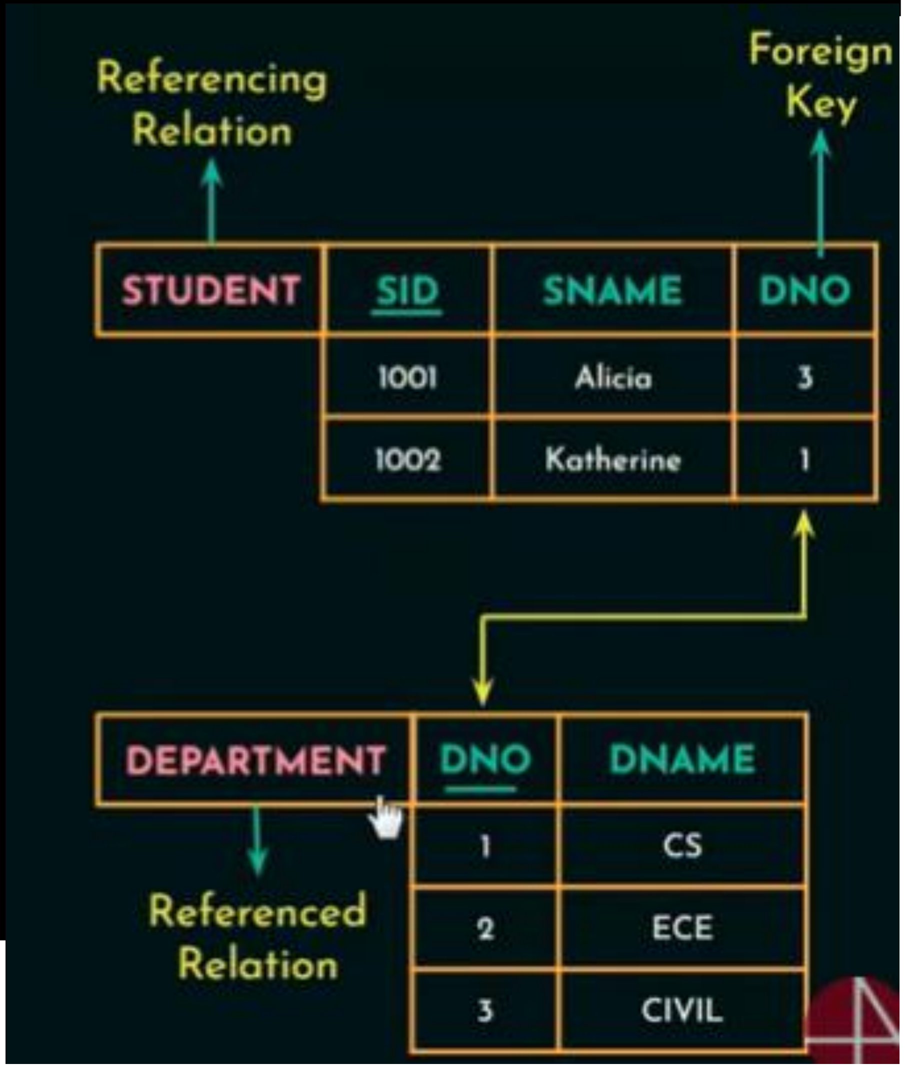
Foreign Key

STUDENT	SID	SNAME	DNO
	1001	Alicia	3
	1002	Katherine	1

DEPARTMENT	DNO	DNAME
	1	CS
	2	ECE
	3	CIVIL

Schema Based Constraints

- Foreign Key must satisfy the following:
  - Same domain
  - Value of FK in a tuple either occurs as a value of PK i.e t1[FK]=t2[PK] or isnull



STUDENT	SID	SNAME	DNO
	1001	Alicia	3
	1002	Katherine	1

DEPARTMENT	DNO	DNAME
	1	CS
	2	ECE
	3	CIVIL

CHECK CONSTRAINT

Business Rule validations can be applied to a table column by using CHECK constraint . CHECK constraint must be specified as a logical expression that evaluates either to TRUE or FALSE.

It takes substantially longer to execute as compared to NOT NULL, PRIMARY KEY, FOREIGN KEY or UNIQUE. Thus check constraints must be avoided if the constraint can be defined using the Not Null, Primary key or Foreign key constraint.

CHECK CONSTRAINT

AT Column level

Synatx: <column name><datatype>(<size>) CHECK (<Logical Expression>)

At TABLE LEVEL

CHECK(Logical Expression>

A CHECK integrity constraint requires that a condition be TRUE OR UN KNOWN for the row to be processed.

ACHECK constraint has the following limitations:

- The condition must be a Boolean expression that can be evaluated using the values in the row being inserted or updated.
- The condition cannot include the SYSDATE, UID ,USER or USERENV SQL functions.
- A check constraint can NOT be defined on a SQL View.
- The check constraint defined on a table must refer to only

CHECK CONSTRAINT

Data values being inserted into the column CUST_NO must start with the capital letter C

Data values being inserted into the column FNAME, MNAME and LNAME should be in UPPERCASE

```
CREATE TABLE CUST_MSTR(CUST_NO VARCHAR2(10), FNAME VARCHAR2(25)
CHECK (CUST_NO LIKE ’ C%’), CHECK(FNAME=UPPER(FNAME))
```

CHECK CONSTRAINT

```
CREATE TABLE employees ( employee_id INT NOT NULL,
last_name VARCHAR(50) NOT NULL, first_name VARCHAR(50),
salary MONEY, CONSTRAINT check_employee_id CHECK
(employee_id BETWEEN 1 and 10000) );
```

```
CREATE TABLE employees ( employee_id INT NOT NULL, last_name
VARCHAR(50) NOT NULL, first_name VARCHAR(50), salary MONEY, CONSTRAINT
check_salary CHECK (salary > 0) );
```

CHECK CONSTRAINT

```
CREATE TABLE suppliers
(
  supplier_id numeric(4),
  supplier_name varchar2(50),

  CONSTRAINT  check_supplier_id
  CHECK (supplier_id BETWEEN 100 and 999)
);
```

```
--insert into suppliers values(101,'XYZ')
--insert into suppliers values(999,'abc')
--insert into suppliers values(100,'XYZ')
insert into suppliers values(11,'XYZ')
```

```
ORA-02290: check constraint
(SQL_OQWECISXRJQQVHFYUMWHY
ESUK.CHECK_SUPPLIER_ID) violated
ORA-06512: at "SYS.DBMS_SQL",
line 1721
```

CHECK CONSTRAINT

Using an ALTER TABLE statement

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name CHECK
(column_name condition);
```

```
ALTER TABLE employees ADD CONSTRAINT check_last_name
CHECK (last_name IN ('Smith', 'Anderson', 'Jones'));
```

CHECK CONSTRAINT

Using an ALTER TABLE statement

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

Enable a Check Constraint

```
ALTER TABLE table_name WITH CHECK CHECK CONSTRAINT constraint_name;
```

```
ALTER TABLE employees WITH CHECK CHECK CONSTRAINT check_salary;
```

Disable a Check Constraint

```
ALTER TABLE table_name NOCHECK CONSTRAINT constraint_name;
```

```
ALTER TABLE employees NOCHECK CONSTRAINT check_salary;
```

CHECK CONSTRAINT

- Any predicate can go into a CHECK() constraint

– CHECK (sex IN (0,1,2,9))

– CHECK (birthdate < CURRENT_TIMESTAMP)

– CHECK (IQ >= 0)

– CHECK (birthdate < hire_date)

– CHECK (partcode LIKE ‘P-%’)

– CHECK (x BETWEEN 0 and 42)

- Remember that a smart optimizer will put all these constraints into the execution plan

- TRIGGERs and STORED PROCEDUREs tell the optimizer nothing

NULL Constraint

A NULL value is different from a blank or a zero. A NULL value cn be inserted into columns of any datatype.

Principle of NULL Values

- Setting a NULL value is appropriate when the actual value is unknown, or when a value would not be meaningful.
- A NULL value is not equivalent to a value of zero if the data type is number and is not equivalent to spaces if the data type is **character**
- A NULL value will evaluate to NULL in any expression(NULL*10 is NULL)
- NULL value can be inserted into columns of any datatype.
- If the column has a NULL value, Oracle ignores any UNIQUE, FOREIGN KEY, CEHCK constraints that may be attached to the column

NOT NULL Constraint

- NOT NULL constraint can be applied at Column level.
- NOT NULL column constraint ensures that a table column can not be left empty. It becomes mandatory

Synatx: <Column Name><DataType>(<Size> NOT NULL

Foreign Key with ON DELETE CASCADE

Use the ON DELETE CASCADE option to specify whether you want rows deleted in a child table when corresponding rows are deleted in the parent table. If you do not specify cascading deletes, the default behavior of the database server prevents you from deleting data in a table if other tables reference it.

Foreign Key

```
CREATE TABLE student1
  (student_num number(4) PRIMARY KEY,
   student_name VARCHAR2(25))
```

```
 CREATE TABLE hostel_info2(s_num number(4)
REFERENCES  student1(student_num),roomno
number(4))
```

```
INSERT INTO STUDENT VALUES(1, 'ISHANI')
  INSERT INTO STUDENT VALUES(2, 'ATHARV')
  INSERT INTO STUDENT VALUES(3, 'SIMRAN')
```

```
 INSERT INTO HOSTEL_INFO VALUES(1, 101)
 INSERT INTO HOSTEL_INFO VALUES(2, 102)
 INSERT INTO HOSTEL_INFO VALUES(3, 103)
```

```
DELETE FROM STUDENT WHERE
STUDENT_NUM=2
select * from hostel_info
```

```
ORA-02292: integrity constraint
(SQL_OQWECISXRJQQVHFYUMWHYESUK.SYS_C0078555716) violated - child
record found ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Foreign Key with ON DELETE CASCADE

```
CREATE TABLE student
   (student_num number(4) PRIMARY KEY,
    student_name VARCHAR2(25))
```

```
CREATE TABLE hostel_info
  (student_num number(4),
    roomno number(4),
    FOREIGN KEY (student_num) REFERENCES student
    ON DELETE CASCADE);
```

```
INSERT INTO STUDENT VALUES(1, ‘RAM')
   INSERT INTO STUDENT VALUES(2, ‘IRA')
   INSERT INTO STUDENT VALUES(3, ‘RANU')
```

```
INSERT INTO HOSTEL_INFO VALUES(1, 101)
INSERT INTO HOSTEL_INFO VALUES(2, 102)
INSERT INTO HOSTEL_INFO VALUES(3, 103)
```

```
DELETE FROM STUDENT WHERE
STUDENT_NUM=2
Successfully done
select * from hostel_info
```

Foreign Keys with "set null on delete

A foreign key with "set null on delete" means that if a record in the parent table is deleted, then the corresponding records in the child table will have the foreign key fields set to null. The records in the child table will **not** be deleted.

Foreign Keys with "set null on delete

```
CREATE TABLE student4(student_num number(4) PRIMARY
KEY,student_name VARCHAR2(25))
```

```
CREATE TABLE hostel_info4(s_num number(4) REFERENCES
student4(student_num) on delete set null,roomno number(4));
```

```
delete from student4 where student_num=2
Select * from hostel_info4
Select * from student4
```

```
INSERT INTO STUDENT4 VALUES(1, 'RAM')
  --INSERT INTO STUDENT4 VALUES(2, 'GAURAV')
 --INSERT INTO STUDENT4 VALUES(3, 'SIMRAN')
```

```
INSERT INTO HOSTEL_INFO4 VALUES(1, 101)
--INSERT INTO HOSTEL_INFO4 VALUES(2, 102)
--INSERT INTO HOSTEL_INFO4 VALUES(3, 103)
```

STUDENT_NUM	STUDENT_NAME
1	RAM
3	SIMRAN

S_NUM	ROOMNO
1	101
-	102
3	103

Thank You