



# Introduction to Logical Design

---

## Database Design

Department of Computer Engineering  
Sharif University of Technology

Maryam Ramezani [maryam.ramezani@sharif.edu](mailto:maryam.ramezani@sharif.edu)



- ❑ **Relational database**: a set of **relations**
- ❑ **Relation**: made up of 2 parts:
  - **Schema** : specifies name of relation, plus name and type of each column.
    - e.g., Students(sid: string, name: string, login: string, age: integer, gpa: real).
  - **Instance** : a **table**, with rows and columns.  
**#Rows = cardinality, #fields = degree / arity.**
- ❑ Can think of a relation as a **set** of rows or **tuples** (all rows are distinct).

# Example Instance of Students Relation



| 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 |

❖ Cardinality = 3, degree = 5, all rows distinct



- ❑ A major strength of the relational model: supports simple, powerful *querying* of data.
- ❑ Queries can be written intuitively, and the **DBMS** is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.



- ❑ Developed by IBM (system R) in the 1970s
- ❑ Data Definition Language (DDL)
  - create, modify, delete relations
  - specify constraints
  - administer users, security, etc.
- ❑ Data Manipulation Language (DML)
  - Specify queries to find tuples that satisfy criteria
  - add, modify, remove tuples



- ❑ Creates the Students relation. Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.
- ❑ As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Students  
  (sid CHAR(20),  
   name CHAR(20),  
   login CHAR(10),  
   age INTEGER,  
   gpa REAL)
```

```
CREATE TABLE Enrolled  
  (sid CHAR(20),  
   cid CHAR(20),  
   grade CHAR(2))
```



## ❑ DROP TABLE Students

- Destroys the relation Students. The schema information and the tuples are deleted.

## ❑ ALTER TABLE Students

### ADD COLUMN firstYear: integer

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a null value in the new field.



- ❑ Can insert a single tuple using
  - INSERT INTO Students (sid, name, login, age, gpa)  
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
  
- ❑ Can delete all tuples satisfying some condition
  - DELETE  
FROM Students S  
WHERE S.name = 'Smith'
  
- ❑ Can modify the column values in an existing row
  - UPDATE Students S  
SET S.age = S.age + 1, S.gpa = S.gpa - 1  
WHERE S.sid = 53688



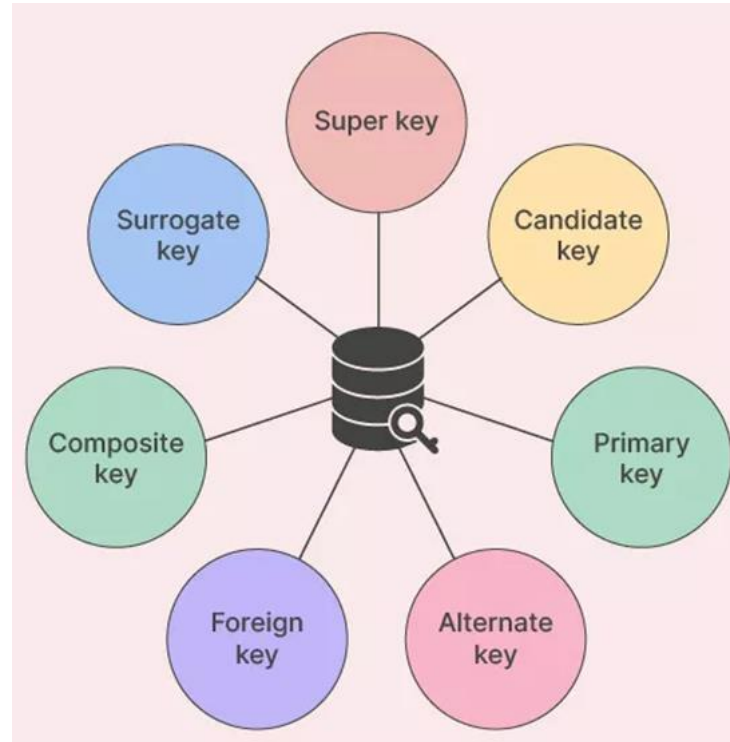


- ❑ **IC:** condition that must be true for *any* instance of the database; e.g., *domain constraints*.
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.
- ❑ A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- ❑ If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!



- ❑ Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys in RDBMS ensure that you can uniquely identify a table record despite these challenges.
- ❑ Allows you to establish a relationship between and identify the relation between tables.
- ❑ Help you to enforce identity and integrity in the relationship.

# Type of Keys





- ❑ A **super key** is a group of single or multiple keys which identifies rows in a table. A super key may have additional attributes that are not needed for unique identification.

| EmpSSN     | EmpNum | Empname |
|------------|--------|---------|
| 9812345098 | AB05   | Shown   |
| 9876512345 | AB06   | Roslyn  |
| 199937890  | AB07   | James   |

In the above-given example, EmpSSN and EmpNum name are superkeys.



- ❑ **Candidate Key**: A super key such that no proper subset is a super key within the relation. A candidate key  $K$  for a relation  $R$  has two properties:
- **Uniqueness**: In each tuple of  $R$ , the values of  $K$  uniquely identify that tuple.
  - **Irreducibility**: No proper subset of  $K$  has the uniqueness property.

```
Student{ID, Aadhar_ID, F_name, M_name, L_name, Age}
```

Here we can see the two candidate keys ID and Aadhar\_ID. So here, there are present more than one candidate keys, which can uniquely identify a tuple in a relation.

Identifying a candidate key requires that we know the “real-world” meaning of the attribute(s) involved so that we can decide whether duplicates are possible.



- ❑ **Primary Key** is a set of attributes (or attribute) which uniquely identify the tuples in relation or table. **The candidate key that is selected to identify tuples uniquely within the relation.** **The primary key is a minimal super key, so there is one and only one primary key in any relationship.** For example,

```
Student{ID, Aadhar_ID, F_name, M_name, L_name, Age}
```

Here only ID or Aadhar\_ID can be primary key because the name, age can be same, but ID or Aadhar\_ID can't be same.

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.



- ❑ A set of fields is a (candidate) key for a relation if :
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.
    - Part 2 false? A **superkey**.
    - If there's  $>1$  key for a relation, one of the keys is chosen (by DBA) to be the **primary key**.
- ❑ E.g., sid is a key for Students. (What about name?) The set {sid, gpa} is a superkey.

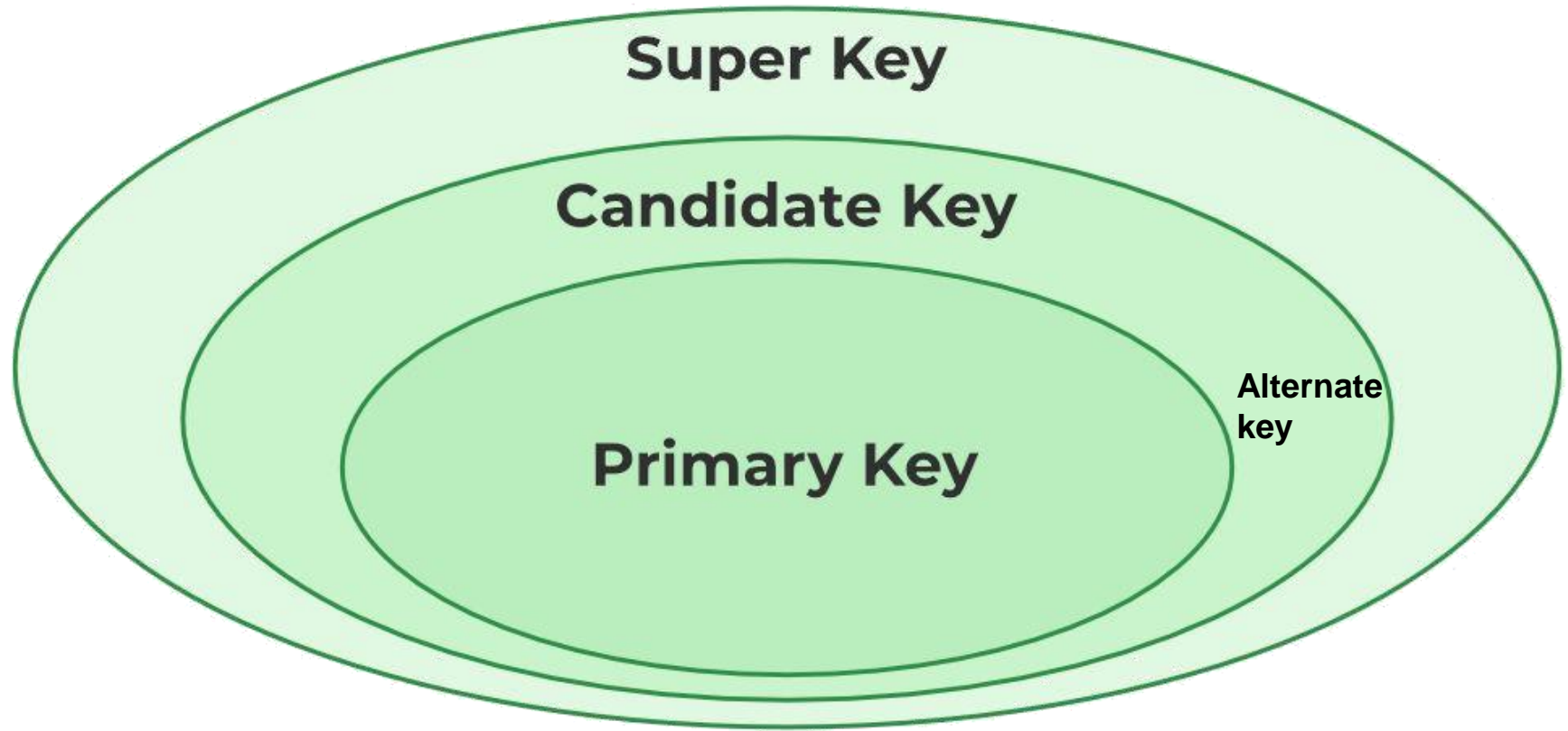


- ❑ **Alternate Key:** The candidate keys that are not selected to be the primary key are called **alternate keys**. This is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.
- ❑ Alternate key = candidate key – primary key

In this table, StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

| StudID | Roll No | First Name | LastName | Email         |
|--------|---------|------------|----------|---------------|
| 1      | 11      | Tom        | Price    | abc@gmail.com |
| 2      | 12      | Nick       | Wright   | xyz@gmail.com |
| 3      | 13      | Dana       | Natan    | mno@yahoo.com |



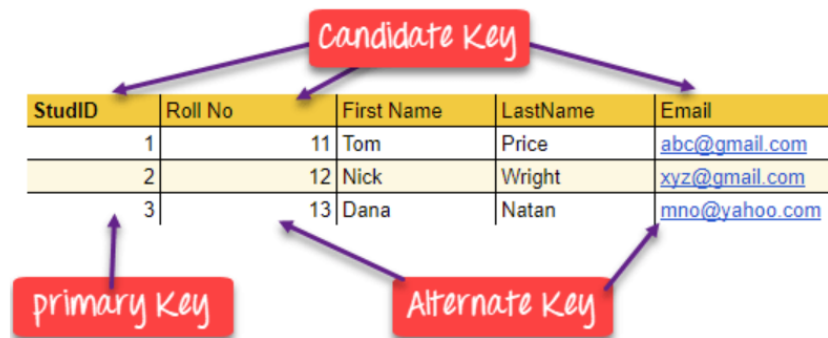


# Example



Candidate key Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

| StudID | Roll No | First Name | LastName | Email         |
|--------|---------|------------|----------|---------------|
| 1      | 11      | Tom        | Price    | abc@gmail.com |
| 2      | 12      | Nick       | Wright   | xyz@gmail.com |
| 3      | 13      | Dana       | Natan    | mno@yahoo.com |



Candidate Key in DBMS



- ❑ **Foreign Key:** is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. An attribute, or set of attributes, within one relation that matches the candidate key of some (possibly the same) relation.

# Foreign Key



| DeptCode | DeptName |
|----------|----------|
| 001      | Science  |
| 002      | English  |
| 005      | Computer |

| Teacher ID | Fname | Lname   |
|------------|-------|---------|
| B002       | David | Warner  |
| B017       | Sara  | Joseph  |
| B009       | Mike  | Brunton |

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

| Teacher ID | DeptCode | Fname | Lname   |
|------------|----------|-------|---------|
| B002       | 002      | David | Warner  |
| B017       | 002      | Sara  | Joseph  |
| B009       | 001      | Mike  | Brunton |



- ❑ **Composite Key:** a candidate key that consists of two or more attributes, (table columns) that together uniquely identify an entity occurrence (table row). **Composite key cannot be null and should be unique.**
- The main difference between the primary key and the composite key is the **primary key is derived from a unique column. Composite is derived by a combination of two or columns. Individually they are not unique but when combined they provide uniqueness.**
  - After getting the composite key, we mention it as the primary key Which will be used for the identification of the rows from the table.
  - If a table already consists of a primary key having a single attribute as an entity for uniquely identifying the records in a table, **then the composite key is of no use as we always select the primary key having minimum attributes (or columns).**

| Customer_ID | Product_ID | Order_Quantity |
|-------------|------------|----------------|
| 1011        | 9023       | 10             |
| 1122        | 9023       | 15             |
| 1099        | 9031       | 20             |
| 1177        | 9031       | 18             |
| 1011        | 9111       | 50             |



- ❑ **Compound Key:** is a composite key for which each attribute that **makes up the key is a foreign key in its own right.**
  - A compound key is where two or more attributes are used to uniquely identify each record in a table.
  - **Each attribute from the compound key is a primary key from a different table.**
  - **A compound key is used when no single attribute in a table can be used as a primary key.**

# Compound Key



Two-Wheel-Rental Inc hires bikes to customers for one day. They have a relational database with three tables as shown below:

## Example

| Members         | Bikes         | Hire             |
|-----------------|---------------|------------------|
| <u>MemberID</u> | <u>BikeID</u> | <u>MemberID*</u> |
| Name            | Make          | <u>BikeID*</u>   |
| Address         | Colour        | <u>HireDate</u>  |
| Telephone       | MT338X440     |                  |

The Hire table needs a compound key of MemberID, BikeID and HireDate.

- MemberID alone can't be used – a member can make more than one hire
- BikeID alone can't be used – a bike can be hired by more than one member
- MemberID and BikeID can't be used – a member could hire the same bike on a different day

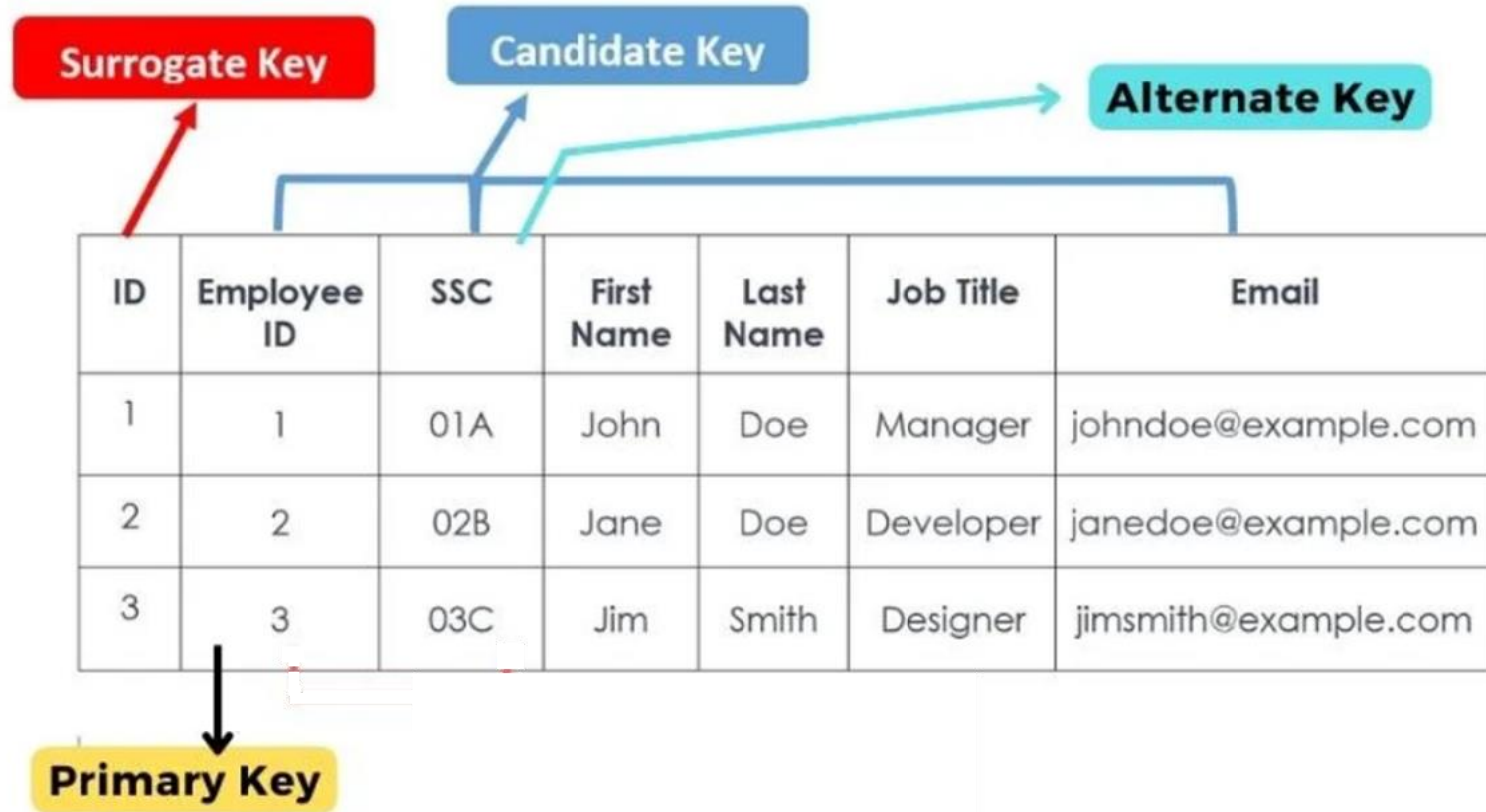
| Members         | Bikes         | Hire             |
|-----------------|---------------|------------------|
| <u>MemberID</u> | <u>BikeID</u> | <u>MemberID*</u> |
| Name            | Make          | <u>BikeID*</u>   |
| Address         | Colour        | <u>HireDate</u>  |
| Telephone       | MT338X440     |                  |



- ❑ **Surrogate Key:** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.
  - A common example of surrogate key is auto-incrementing integer as primary key in a table.



# Type of Keys



# Primary and Candidate Keys in SQL



❑ Possibly many candidate keys (specified using UNIQUE), one of which is chosen as the primary key.

- ❖ “For a given student and course, there is a single grade.”
- ❖ “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”
- ❖ Used carelessly, an IC can prevent the storage of database instances that arise in practice!

```
CREATE TABLE Enrolled  
(sid CHAR(20)  
  cid CHAR(20),  
  grade CHAR(2),  
  PRIMARY KEY (sid,cid))
```

```
CREATE TABLE Enrolled  
(sid CHAR(20)  
  cid CHAR(20),  
  grade CHAR(2),  
  PRIMARY KEY (sid),  
  UNIQUE (cid, grade))
```



- ❑ Foreign key : Set of fields in one relation that is used to 'refer' to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a 'logical pointer'.
- ❑ E.g. **sid** is a foreign key referring to **Students**:
  - Enrolled(**sid**: string, cid: string, grade: string)
  - If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references.
- ❑ **Foreign key attributes in R1 referring to R2 have the following rules:**
  - The FK attributes in R1 have the same domain(s) as the primary key attributes of R2
  - The value of FK in tuple t1 in R1 must reference an existing PK value in tuple t2 of R2

# Foreign Keys in SQL



- ❑ Only students listed in the Students relation should be allowed to enroll for courses.

```
CREATE TABLE Enrolled
(sid CHAR(20), cid CHAR(20), grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid) REFERENCES Students )
```

Enrolled

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

FOREIGN Key

Students

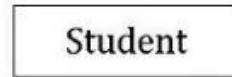
| sid   | name  | login      | age | gpa |
|-------|-------|------------|-----|-----|
| 53666 | Jones | jones@cs   | 18  | 3.4 |
| 53688 | Smith | smith@eecs | 18  | 3.2 |
| 53650 | Smith | smith@math | 19  | 3.8 |

PRIMARY Key

# Mapping Entities and Attributes

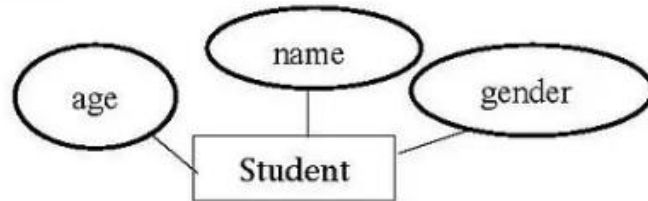


## 1. Entity



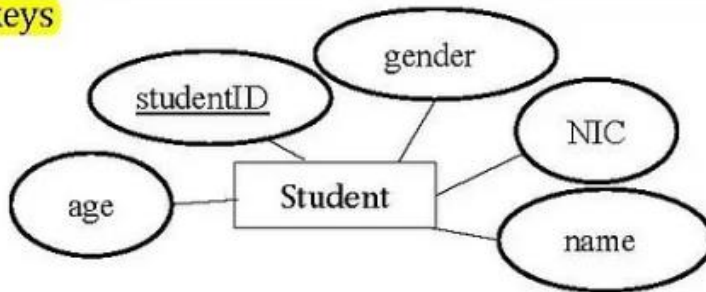
Student table

## 2. simple attributes



Student (name,age, gender)

## 3. keys

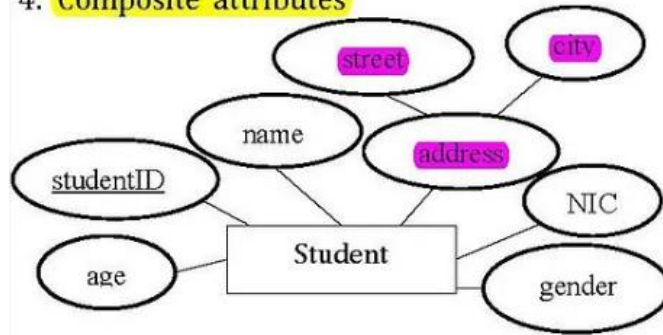


Student (studentID, NIC, name, age, gender)

# Mapping Entities and Attributes



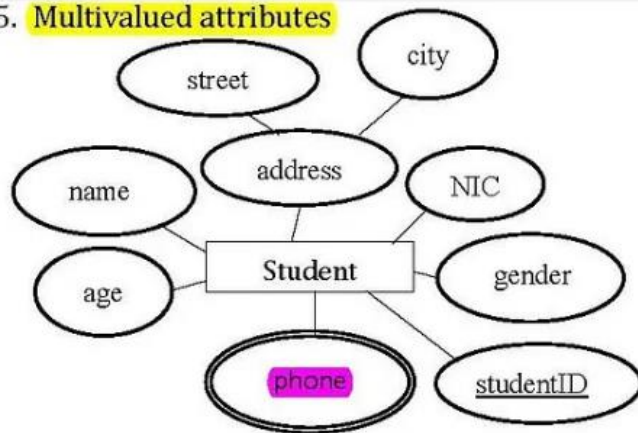
## 4. Composite attributes



Student (studentID, NIC, name, age, gender, street,city)

A strong entity set with any number of composite attributes will require only one table in relational model.

## 5. Multivalued attributes



Student (studentID, NIC, name, age, gender, street,city)

Student\_phone (studentID, phone)

f.k. p.k

- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.



## Note on multivalued attribute:

- ❑ Create separate table for each multivalued attribute **requires join** on two tables which is time consuming.
- ❑ If maximum number of multivalued attributes is known as #a, so the entity table should have #a columns.
  - Student(studentID,NIC,name,age,gender,street.city,phone1,phone2,phone3)
  - There will be no joining
  - There are many null values in the data



## Derived Attribute

- ☐ We do not include derived attributes in logical design.
- ☐ We can use views, we will see this more later.





- ❑ Chapter 9 of FUNDAMENTALS OF Database Systems, SEVENTH EDITION
- ❑ Chapter 4 of Database Systems A Practical Approach to Design, Implementation, and Management, SIXth edition