DATABASES

GS WEB APPLICATION DEVELOPMENT

Teacher: Vanesa Maldonado Guerrero
Curs 2025/26

Relational Model

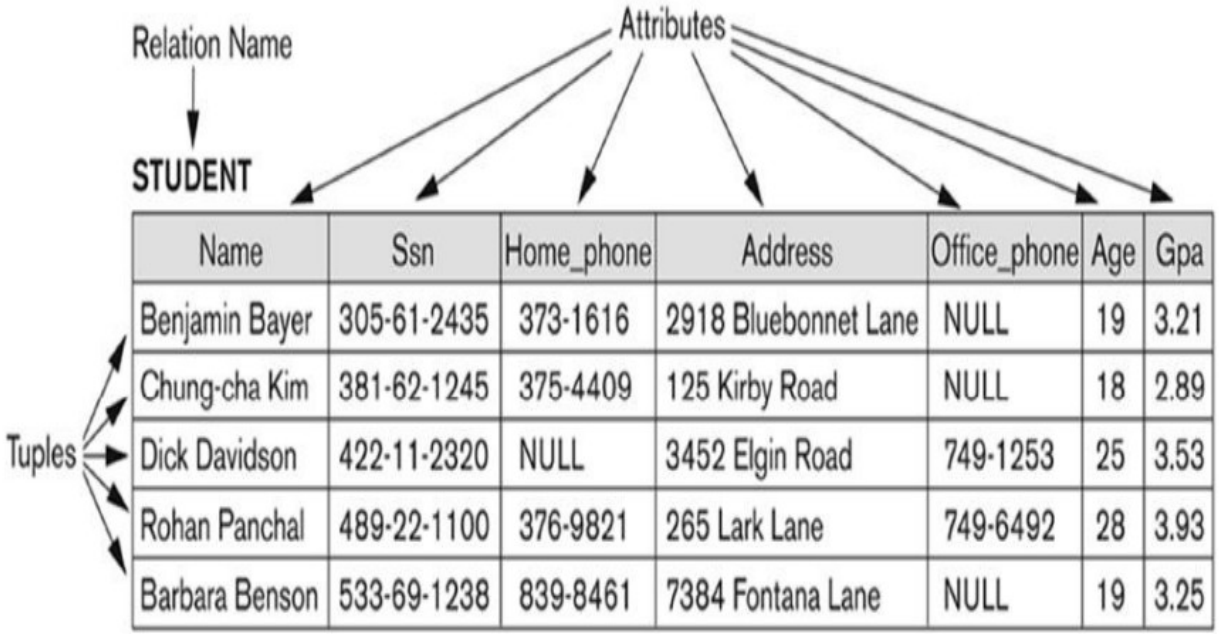
- The Relational Model of Data is based on the concept of a Relation.
- Relational Model was proposed by Edgar Codd in 1969 to model data in the form of relations or tables.
- It represents how data is stored in Relational Databases.

Relational Models Concepts

- **Relation** looks like a **table**.
- **Attribute** are the properties that define a relation. It is the column header that gives an indication of the meaning of the data items in that column.
- **Tuple**: Each row in the relation is known as tuple. The data elements in each row represent certain facts that correspond to a real-world entity or relationship.

Example of a Relation

- A relation **STUDENT**.



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Key Terms in the Relational Model

- **Attribute**: Attributes are the properties that define an entity. For example, ROLL_NO, NAME, ADDRESS etc.
- **Relation Schema**: A relation schema defines the structure of the relation and represents the name of the relation with its attributes. For example, STUDENT (ROLL_NO, NAME, ADDRESS, PHONE and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation it is called Relational Schema.
- **Tuple**: A Tuple represents a row in a relation. Each tuple contains a set of attribute values that describe a particular entity. For example, (1, RAM, DELHI, 9455123451, 18) is a tuple in the STUDENT table.
- **Relation Instance**: The set of tuples of a relation at a particular instance of time is called a relation instance. It can change whenever there is an insertion, deletion or update in the database.
- **Degree**: The number of attributes in the relation is known as the degree of the relation. For example, The STUDENT relation has a degree of 5, as it has 5 attributes.
- **Cardinality**: The number of tuples in a relation is known as cardinality. For example, The STUDENT relation defined above has cardinality 4.
- **NULL Values**: The value which is not known or unavailable is called a NULL value. It is represented by NULL. For example, PHONE of STUDENT having ROLL_NO 4 is NULL.

Types of Keys in the Relational Model

- **Primary Key**: A Primary Key uniquely identifies each tuple in a relation. It must contain unique values and cannot have NULL values.
- **Candidate Key**: A Candidate Key is a set of attributes that can uniquely identify a tuple in a relation.
- **Super Key**: A Super Key is a set of attributes that can uniquely identify a tuple.
- **Foreign Key:** A Foreign Key is an attribute in one relation that refers to the primary key of another relation.
- **Composite Key**: A Composite Key is formed by combining two or more attributes to uniquely identify a tuple.

Types of Keys in the Relational Model

1. Super Key

The set of one or more attributes (columns) that can uniquely identify a tuple (record) is known as Super Key. It may include extra attributes that aren't important for uniqueness but still uniquely identify the row. For Example, STUD_NO, (STUD_NO, STUD_NAME), etc.

- A super key is a group of single or multiple keys that uniquely identifies rows in a table. It supports NULL values in rows.
- A super key can contain extra attributes that aren’t necessary for uniqueness.
- For example, if the "STUD_NO" column can uniquely identify a student, adding "SNAME" to it will still form a valid super key, though it's unnecessary.

Example: Consider the STUDENT table:

A super key could be a combination of STUD_NO and PHONE, as this combination uniquely identifies a student.

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

Types of Keys in the Relational Model

2. Candidate Key

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key. For Example, STUD_NO in STUDENT relation.

- A candidate key is a minimal super key, meaning it can uniquely identify a record but contains no extra attributes.
- It is a super key with no repeated data is called a candidate key.
- The minimal set of attributes that can uniquely identify a record.
- A candidate key must contain unique values, ensuring that no two rows have the same value in the candidate key's columns.
- Every table must have at least a single candidate key.
- A table can have multiple candidate keys but only one primary key.

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

Example: For the STUDENT table below, STUD_NO can be a candidate key, as it uniquely identifies each record.

Table: STUDENT_COURSE

A composite candidate key example: {STUD_NO, COURSE_NO} can be a candidate key for a STUDENT_COURSE table.

STUD_NO	TEACHER_NO	COURSE_NO
1	001	C001
2	056	C005

Types of Keys in the Relational Model

3. Primary Key

There can be more than one candidate key in relation out of which one can be chosen as the primary key. For Example, STUD_NO, as well as STUD_PHONE, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).

- A primary key is a unique key, meaning it can uniquely identify each record (tuple) in a table.
- It must have unique values and cannot contain any duplicate values.
- A primary key cannot be NULL, as it needs to provide a valid, unique identifier for every record.
- A primary key does not have to consist of a single column. In some cases, a composite primary key (made of multiple columns) can be used to uniquely identify records in a table.
- Databases typically store rows ordered in memory according to primary key for fast access of records using primary key.

Example: STUDENT table -> Student(STUD_NO, SNAME, ADDRESS, PHONE) , STUD_NO is a primary key

Table: STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

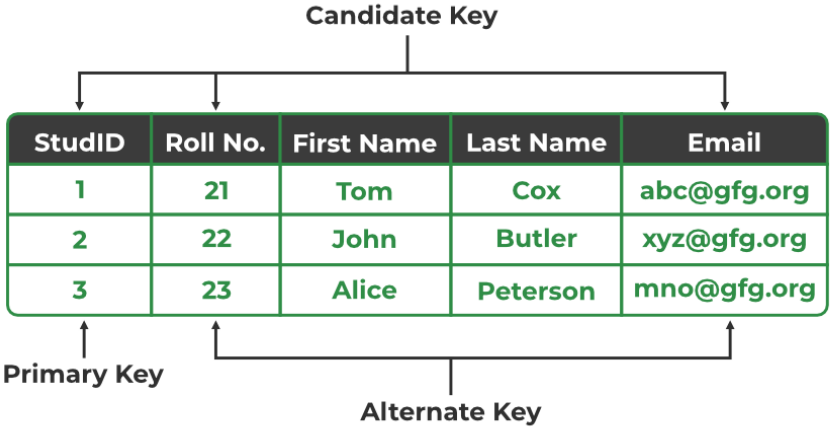
Types of Keys in the Relational Model

4. Alternate Key

An alternate key is any candidate key in a table that is not chosen as the primary key. In other words, all the keys that are not selected as the primary key are considered alternate keys.

- An alternate key is also referred to as a secondary key because it can uniquely identify records in a table, just like the primary key.
- An alternate key can consist of one or more columns (fields) that can uniquely identify a record, but it is not the primary key.

Example: In the STUDENT table, both STUD_NO and PHONE are candidate keys. If STUD_NO is chosen as the primary key, then PHONE would be considered an alternate key.

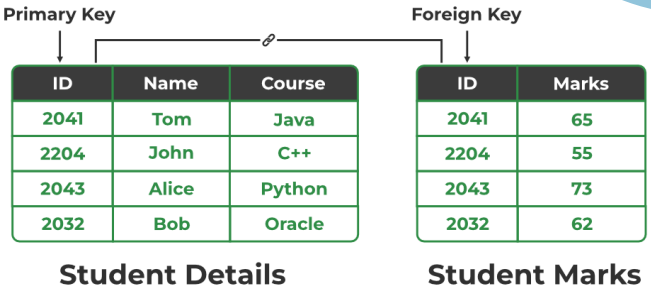


Types of Keys in the Relational Model

5. Foreign Key

A foreign key is an attribute in one table that refers to the primary key in another table. The table that contains the foreign key is called the referencing table and the table that is referenced is called the referenced table.

- A foreign key in one table points to the primary key in another table, establishing a relationship between them.
- It helps connect two or more tables, enabling you to create relationships between them. This is important for maintaining data integrity and preventing data redundancy.
- They act as a cross-reference between the tables.



Example: Consider the STUDENT_COURSE table.

Here, STUD_NO in the STUDENT_COURSE table is a foreign key that references the STUD_NO primary key in the STUDENT table.

Unlike the Primary Key of any given relation, Foreign Key can be NULL as well as may contain duplicate tuples i.e. it need not follow uniqueness constraint. For Example, STUD_NO in the STUDENT_COURSE relation is not unique.

It has been repeated for the first and third tuples. However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique and it cannot be null.

STUD_NO	TEACHER_NO	COURSE_NO
1	005	C001
2	056	C005

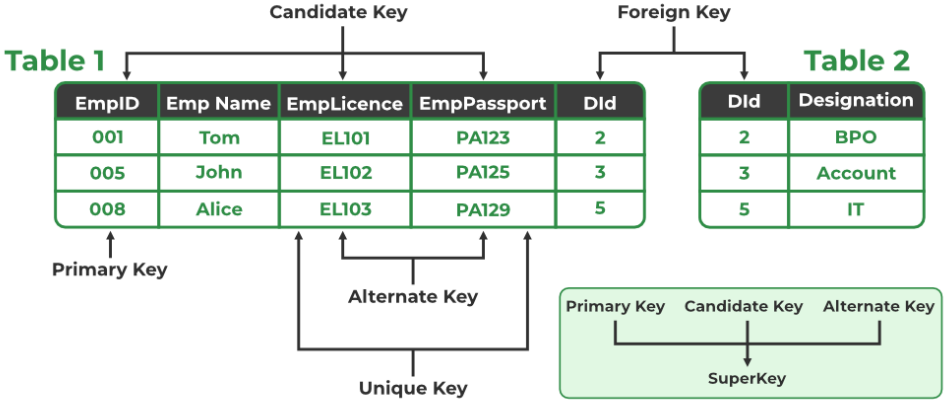
Types of Keys in the Relational Model

6. Composite Key

Sometimes, a table might not have a single column/attribute that uniquely identifies all the records of a table. To uniquely identify rows of a table, a combination of two or more columns/attributes can be used. It still can give duplicate values in rare cases. So, we need to find the optimal set of attributes that can uniquely identify rows in a table.

- It acts as a primary key if there is no primary key in a table
- Two or more attributes are used together to make a composite key .
- Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

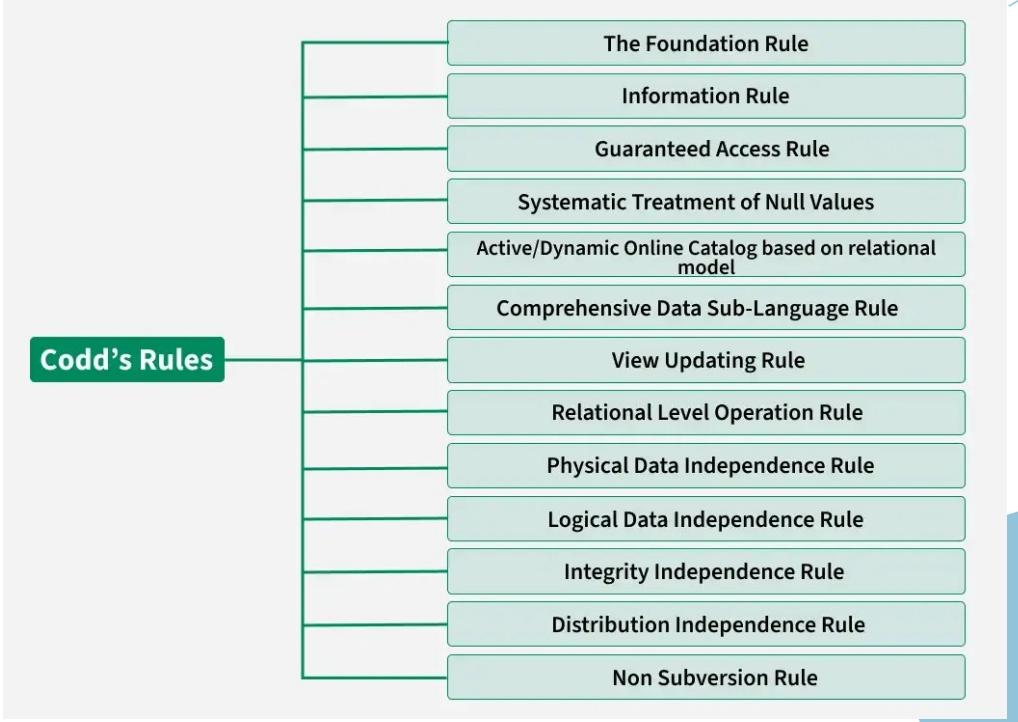
Example: In the STUDENT_COURSE table, {STUD_NO, COURSE_NO} can form a composite key to uniquely identify each record.



CODD's 12 Rules of the Relational Model

Codd’s rules, introduced by E.F. Codd, define the conditions that a database system must meet to qualify as a Relational Database Management System (RDBMS). These rules serve as fundamental guidelines to ensure that data is organized, consistent, and reliably managed.

In practice, however, it is uncommon for any database product to fully comply with all of Codd’s rules. Most systems typically adhere to about eight or nine of them. Although Codd originally proposed 13 rules, they are widely referred to as the 12 rules of Codd.



CODD's 12 Rules of the Relational Model

1. Information rule

All data is organized into logical tables (like spreadsheets), where each row represents a record and each column represents a type of information.

```
CREATE TABLE system_user (ID INT PRIMARY KEY, name VARCHAR(50), age INT, department_ID INT)
--This will create a table with 4 columns, where the user ID is the primary key:

| ID | name | age | department_ID |
| | | | | |

--All tables should represent information logically
```

CODD's 12 Rules of the Relational Model

2. Guaranteed access rule

To find any data, you only need to know the table name, the primary key, and the column name where the data you are looking for is stored.

```
SELECT name
FROM system_user
WHERE ID = 15;

--We can access a piece of data through its primary key
```

CODD's 12 Rules of the Relational Model

3. Rule for the systematic treatment of null values

Null values can have meaning, and the system must be able to handle them, recognizing and treating them differently from other data, such as empty or 0. The logic applied must be correct, and null values are independent of the data in the column, making them essential for accurately representing the absence of information.

```
CREATE TABLE system_user (
    ID INT PRIMARY KEY,
    name VARCHAR(50),
    bornDate DATE -- This column can contain null values
);


INSERT INTO Employees (ID, Name, BirthDate) VALUES
(1, 'John', '1990-05-15'),
(2, 'Mary', NULL), -- Unknown date of birth (null value)
(3, 'Pedro', '1985-09-20');
```

CODD's 12 Rules of the Relational Model

4. Dynamic online catalog based on the relational model

The database description is represented at the logical level in the same way as normal data, so that authorized users can apply the same relational language to their query as they apply to normal data.

```
-- See all tables in the database
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public';

-- See the columns of a table
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'clientes';
```

CODD's 12 Rules of the Relational Model

5. Comprehensive rule of the complete data sub-language

The language you use to work with databases should allow you to do everything you need, such as adding new data, changing existing data, or creating complex queries.

```
SELECT name, department_ID AS Id_Departamento
FROM system_user
WHERE deparment_ID = 5;

--We can perform a complex query that involves several operations
```

CODD's 12 Rules of the Relational Model

6. View update rule

If you can view the data in a special way (such as a view), you should be able to update it as well, and the system will take care of applying those changes correctly.

```
UPDATE departments SET department_ID = 5 WHERE age > 30;

--We can update a view and the changes are reflected correctly
```

CODD's 12 Rules of the Relational Model

7. High level of insertion, updating and cancellation

It focuses on simplifying the way we interact with stored information. Instead of having to program each operation individually as we would in languages like C or Java, the goal is to use a language that is easier for humans to understand. This means we can manipulate data sets at once, making it easier to insert, update, and delete information without having to write complicated code.

```
--Add new users
INSERT INTO System_user (ID, name, department)
VALUES
(101, 'Juan', 'Sales'),
(102, 'María', 'Marketing');

--Update information
UPDATE system_user
SET departament = 'Human Resources'
WHERE ID IN (101, 102);

--Delete user
DELETE FROM system_user
WHERE ID = 103;
```

CODD's 12 Rules of the Relational Model

8. Physical independence of data

The way data is stored shouldn't matter to you when interacting with it. It may be stored in different ways, but that doesn't affect how you use it.

```
ALTER TABLE system_user
ADD COLUMN mail VARCHAR(100);

--We can change the way data is stored without affecting queries
```

CODD's 12 Rules of the Relational Model

9. Logical independence of data

If you change the way you view your data (for example, by reorganizing tables), it shouldn't affect the applications that use that data.

```
ALTER TABLE system_user
ADD COLUMN hire_date DATE;

--We can change the data structure without affecting the applications.
```

CODD's 12 Rules of the Relational Model

10. Independence of integrity

The rule that ensures data is correct and complete must be separated from the applications that use it.

```
ALTER TABLE system_user
ADD CONSTRAINT ck_age_legal
CHECK (age >= 18);

--We can set an integrity rule without affecting applications
```

CODD's 12 Rules of the Relational Model

11. Independence of distribution

If the data is in different physical locations, the system must be able to handle that seamlessly, as if it were all in one place.

```
SELECT * FROM name@RemoteServer;

--We can manage a distributed database without affecting queries
```

CODD's 12 Rules of the Relational Model

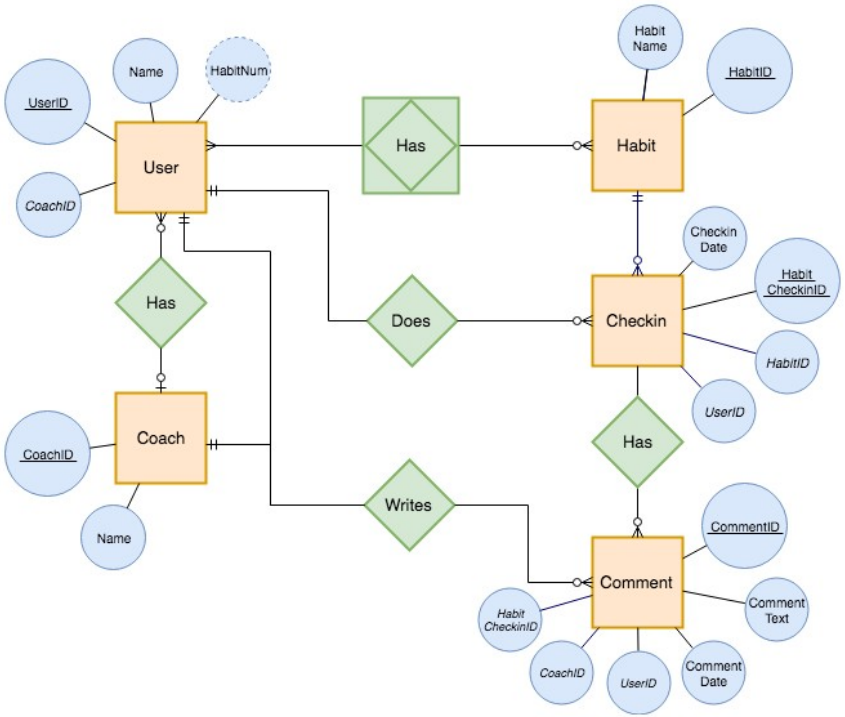
12. Rule of non-subversion

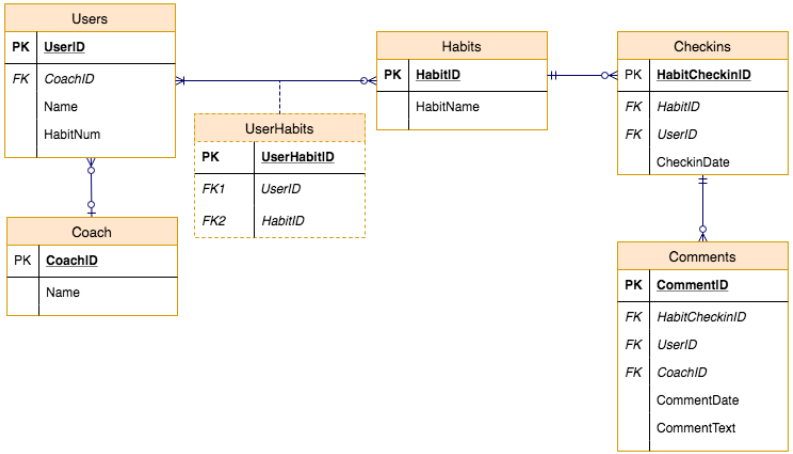
Even if you can access the database in more advanced ways, you shouldn't be able to bypass the rules and restrictions you've set to ensure data integrity.

```
DELETE FROM system_user WHERE age < 25;

--Even with advanced access, restrictions are still respected
```

How to Convert ER Diagrams to Tables in DBMS?





How to Convert ER Diagrams to Tables in DBMS?

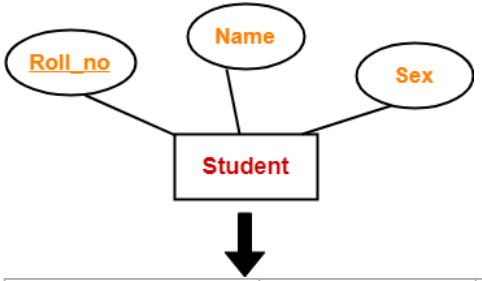
RULE-01: FOR STRONG ENTITY SET WITH ONLY SIMPLE ATTRIBUTES

A **strong entity set with only simple attributes** will require **only 1 table** in relational model.

Attributes of the table will be the attributes of the entity set.

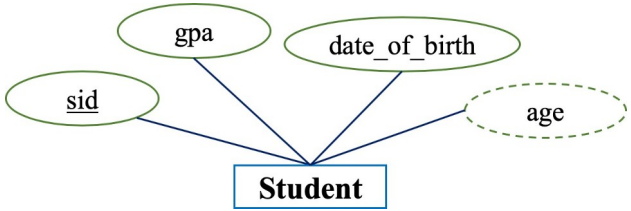
Derived attributes removed from the table.

The primary key of the table will be the key attribute of the entity set.



Roll_no	Name	Sex

Schema: Student(Roll_no, Name, Sex)



sid	gpa	date_of_birth

Schema: Student(sid, gpa, date_of_birth)

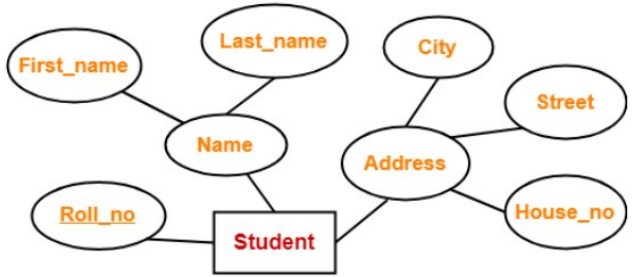
How to Convert ER Diagrams to Tables in DBMS?

RULE-02: FOR STRONG ENTITY SET WITH COMPOSITE ATTRIBUTES

A **strong entity set with any number of composite attributes** will require **only 1 table** in relational model.

While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself

Roll numbers are unique identification numbers that can be assigned to students at the time of admission or after admission. To avoid instances of students of different batches (party) having the same roll numbers, you can set a common prefix to all student roll numbers of each batch (party).



Roll_no	First_name	Last_name	House_no	Street	City

Schema: Student(Roll_no, First_name, Last_name, House_no, Street, City)

How to Convert ER Diagrams to Tables in DBMS?

RULE-03: FOR STRONG ENTITY SET WITH MULTIVALUED ATTRIBUTES

A **strong entity set with any number of multivalued attributes** will require **2 tables** in relational model.

One table will contain all the simple attributes with the primary key.

Other table will contain the primary key and all the multivalued attributes.

Why do we need to create separate schema for multivalued attributes?

For answering this question, consider the STUDENT relation given in Table 1 with Phone as multivalued attribute. In Phone attribute, some records have more than one phone numbers and some without phone numbers. The phone numbers are separated by comma for the records with more than one phone numbers. If we store phone numbers like Table 1, the main problem is some queries cannot be directly answered.

RegNo	*SName*	*Gen*	*PR*	*Phone*
R1	Sundar	M	BTech	9898786756, 9897786776
R3	Karthik	M	MCA	
R5	Priya	F	MS	9809780967, 7898886756
R6	Ram	M	MTech	9876887909

How to Convert ER Diagrams to Tables in DBMS?

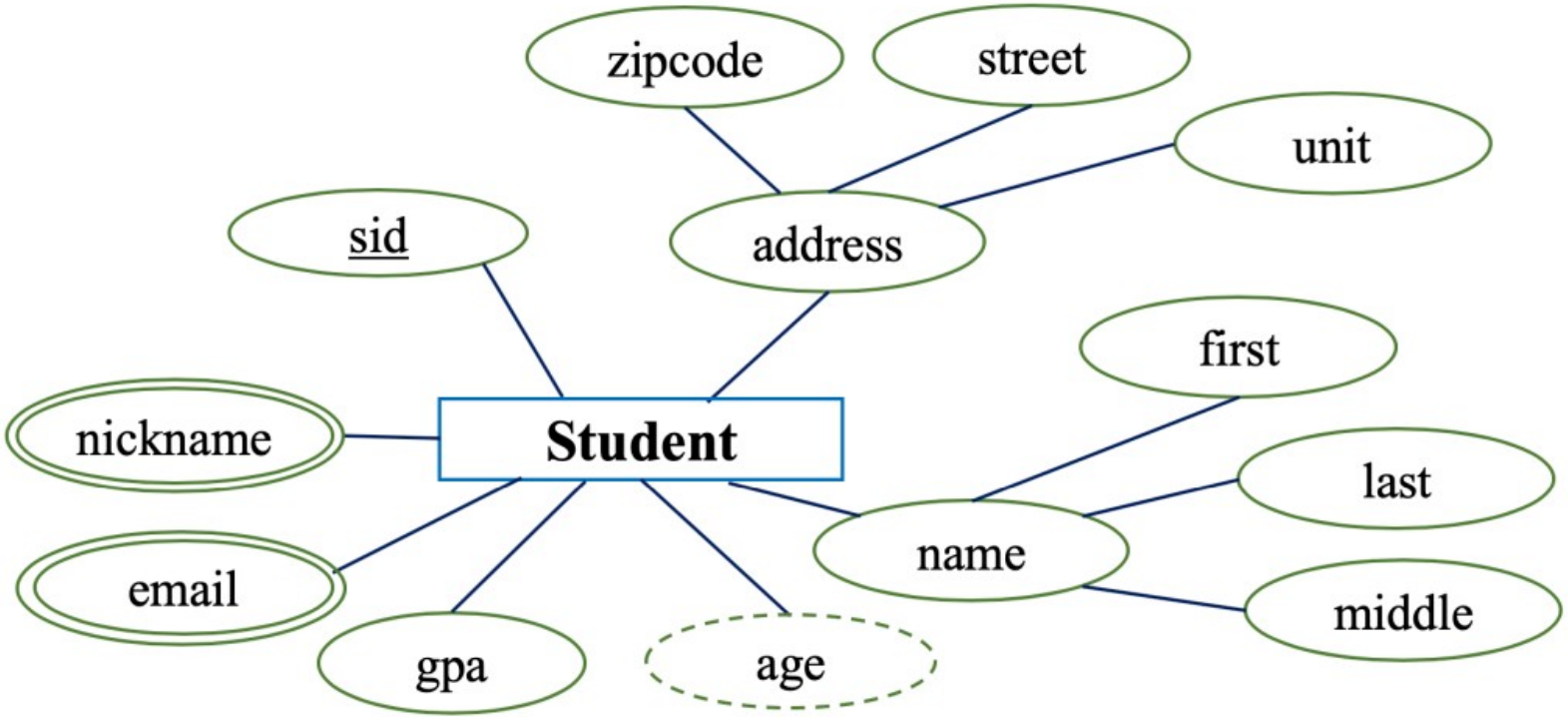
RULE-03: FOR STRONG ENTITY SET WITH MULTIVALUED ATTRIBUTES

To avoid such a problem, we need to store this Phone column into separate table along with the RegNo attribute as given below in table 2. As a result, we will get two tables, STUDENT and STU_PHONE.

RegNo	*SName*	*Gen*	*PR*
R1	Sundar	M	BTech
R2	Ram	M	MS
R3	Karthik	M	MCA
R4	John	M	BSc

RegNo	*Phone*
R1	9898786756
R1	9897786776
R5	9809780967
R5	7898886756
R6	9876887909

TASK. BUILD A RELATIONAL SCHEMA FROM THE FOLLOWING ER DIAGRAM



TASK. BUILD A RELATIONAL SCHEMA FROM THE FOLLOWING ER DIAGRAM

1. Identify attribute types

- Primary key (PK): sid
- Simple attribute: gpa
- Composite attributes:
  - name → (first, middle, last)
  - address → (street, unit, zipcode)
- Multivalued attributes:
  - nickname
  - email
- Derived attribute (removed): age

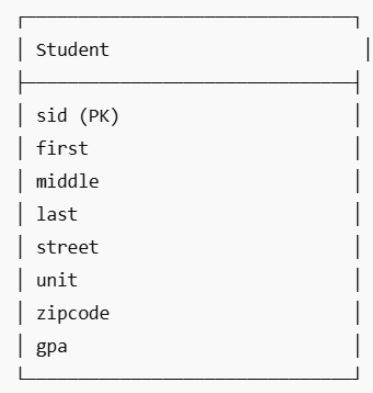
2. Apply transformation rules

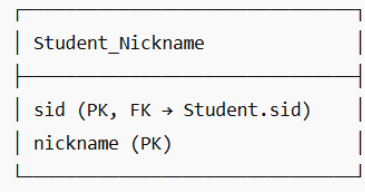
Main table (simple + decomposed composite attributes)

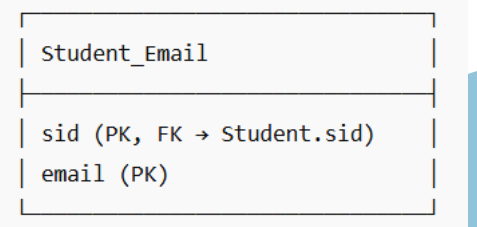
Table for the multivalued attribute nickname

Table for the multivalued attribute email

```
Student(sid, first, middle, last, street, unit, zipcode, gpa)
Student_Nickname(sid, nickname)
Student_Email(sid, email)
```





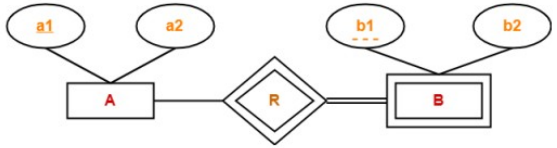


How to Convert ER Diagrams to Tables in DBMS?

RULE-04: FOR BINARY RELATIONSHIP WITH WEAK ENTITY SET

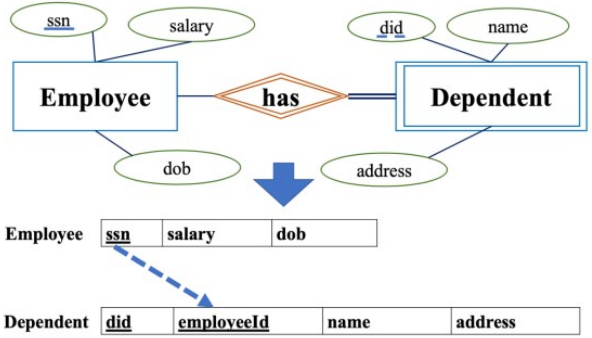
Weak entity set always appears in association with identifying relationship with total participation constraint.

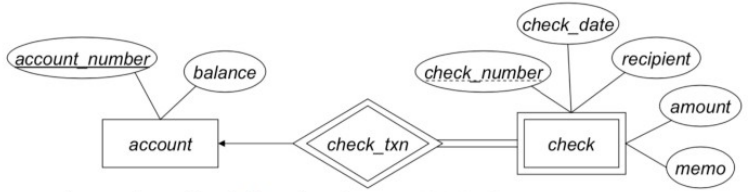
Examples



Here, **2 tables** will be required

1. A(a1, a2)
2. BR(a1, b1, b2)





account(*account_number*, *balance*)
check schema:
Discriminator is *check_number*
Primary key for *check* is: (*account_number*, *check_number*)
check(*account_number*, *check_number*, *check_date*, *recipient*, *amount*, *memo*)

How to Convert ER Diagrams to Tables in DBMS?

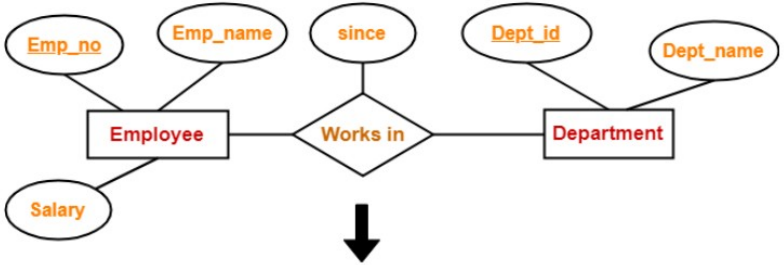
RULE-05: TRANSLATING RELATIONSHIP SET WITHOUT CARDINALITY RATIOS DESCRIPTION INTO A TABLE

A relationship between 2 relations will require **3 tables** in the relational model.

Attributes of the table are

- Primary key attributes of the participating entity sets
- Its own descriptive attributes if any.

Set of non-descriptive attributes will be the primary key.



Emp_no	Dept_id	since

Schema: WorksIn(Emp_no, Dept_id, since)

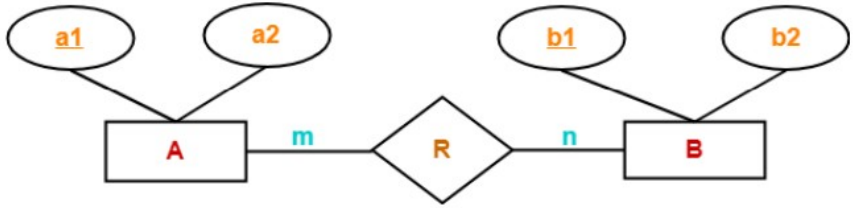
Note.

If we consider the overall ER diagram, 3 tables will be required

- One table for the entity set “Employee”
- One table for the entity set “Department”
- One table for the relationship set “WorksIn”

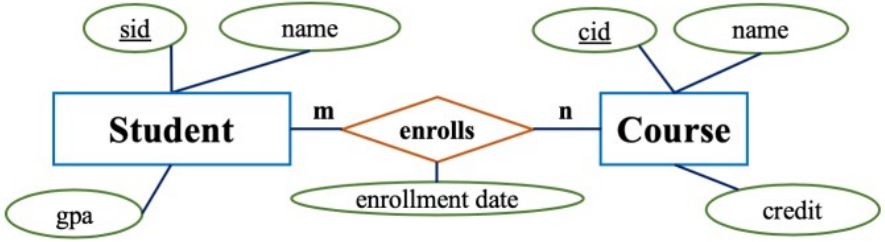
How to Convert ER Diagrams to Tables in DBMS?

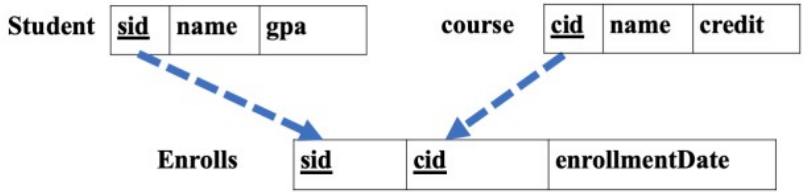
RULE-06: FOR BINARY RELATIONSHIP WITH CARDINALITY RATIO M:N



Here, **3 tables** will be required

1. A(a1, a2)
2. R(a1, b1)
3. B(b1, b2)

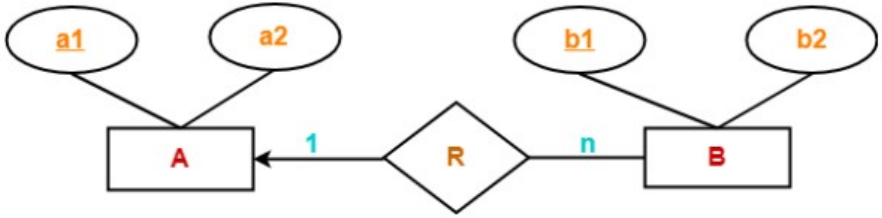


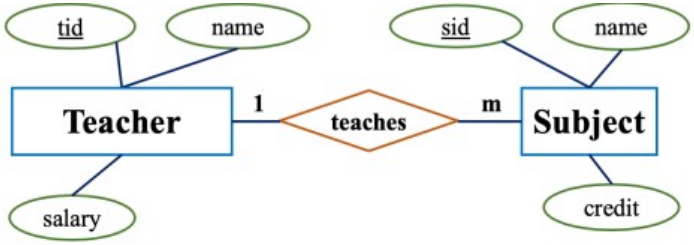


How to Convert ER Diagrams to Tables in DBMS?

RULE-07: FOR BINARY RELATIONSHIPS WITH CARDINALITY RATIO 1:1,N OR 1,M:1

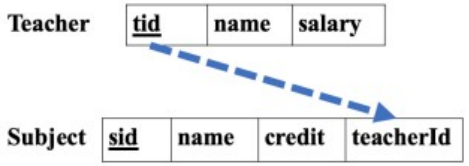
CASE-01: FOR BINARY RELATIONSHIP WITH CARDINALITY RATIO 1:1,N





Here, 2 tables will be required

1. A(a1, a2)
2. BR(b1, b2, a1)

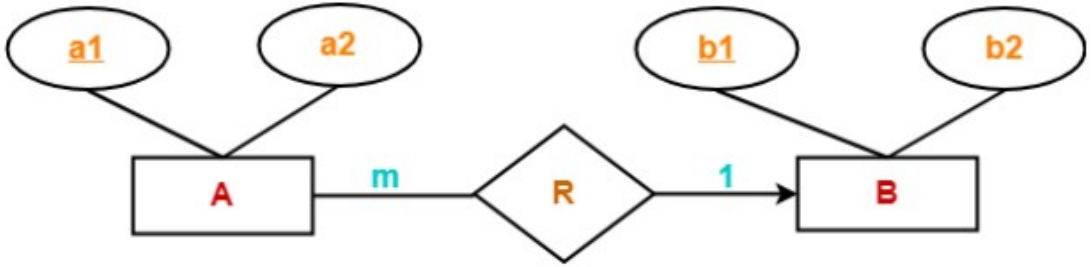


NOTE. Here, combined table will be drawn for the entity set B and relationship set R.

How to Convert ER Diagrams to Tables in DBMS?

RULE-07: FOR BINARY RELATIONSHIPS WITH CARDINALITY RATIO 1:1,N OR 1,M:1

CASE-02: FOR BINARY RELATIONSHIP WITH CARDINALITY RATIO 1,M:1



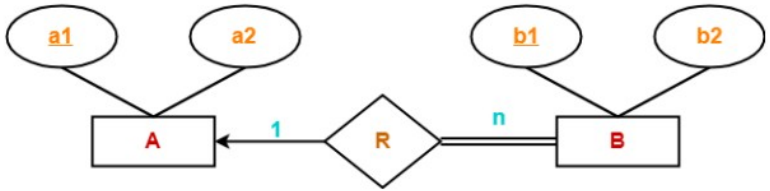
Here, 2 tables will be required

1. AR(a1, a2, b1)
2. B(b1, b2)

NOTE. Here, combined table will be drawn for the entity set A and relationship set R.

How to Convert ER Diagrams to Tables in DBMS?

RULE-08: FOR BINARY RELATIONSHIP WITH CARDINALITY RATIO 1:0,N



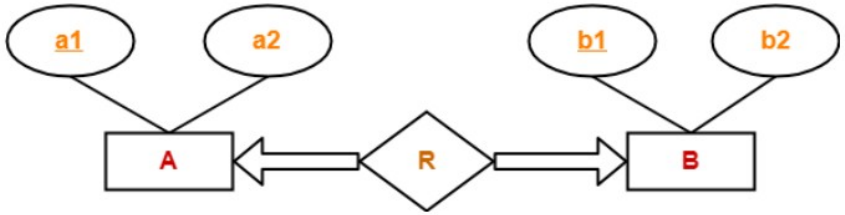
Then, **3 tables** will be required

1. A(a1, a2)
2. R(a1, b1)
3. B(b1, b2)

How to Convert ER Diagrams to Tables in DBMS?

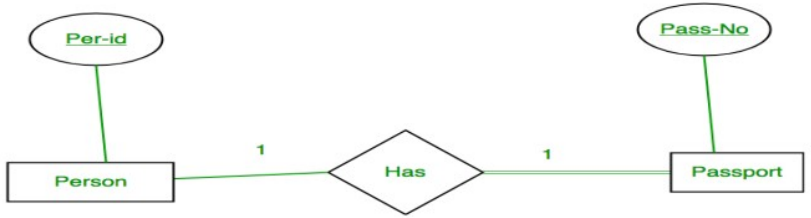
RULE-09: FOR BINARY RELATIONSHIP WITH CARDINALITY 1,1;1,1.

If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table



Here, Only one table is required

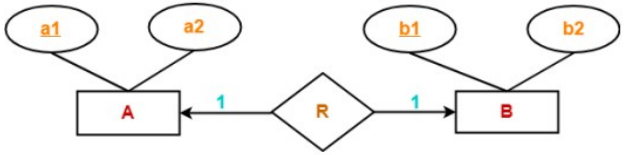
- ARB(a1, a2, b1, b2)

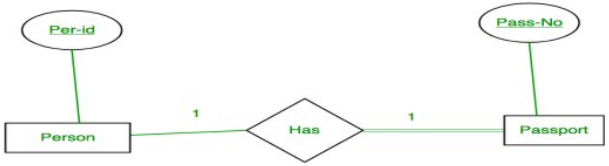


How to Convert ER Diagrams to Tables in DBMS?

RULE-10: FOR BINARY RELATIONSHIPS WITH CARDINALITY RATIO 1,1:0,1 OR 0,1:1,1

If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table





Here, **2 tables** will be required. Either combine 'R' with 'A' or 'B'

Way-01: for 1,1;0,1

1. AR(a1, a2, b1)
2. B(b1, b2)

Way-02: for 0,1;1,1

1. A(a1, a2)
2. BR(a1, b1, b2)

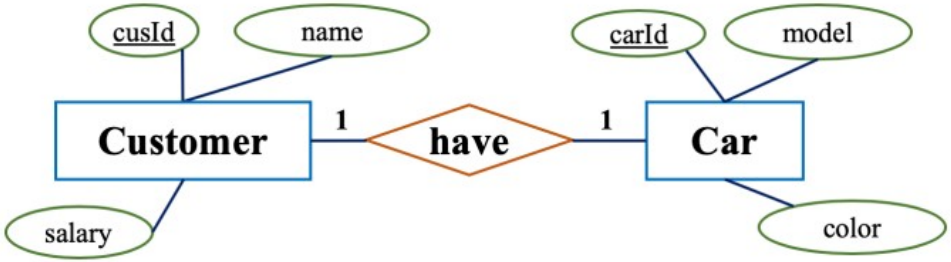
A person has 0 or 1 passport number and Passport is always owned by 1 person. So it is 1:1 cardinality with full participation constraint from Passport (1,1:0,1).

How to Convert ER Diagrams to Tables in DBMS?

RULE-11: FOR BINARY RELATIONSHIPS WITH CARDINALITY RATIO 0,1:0,1

Then, **3 tables** will be required

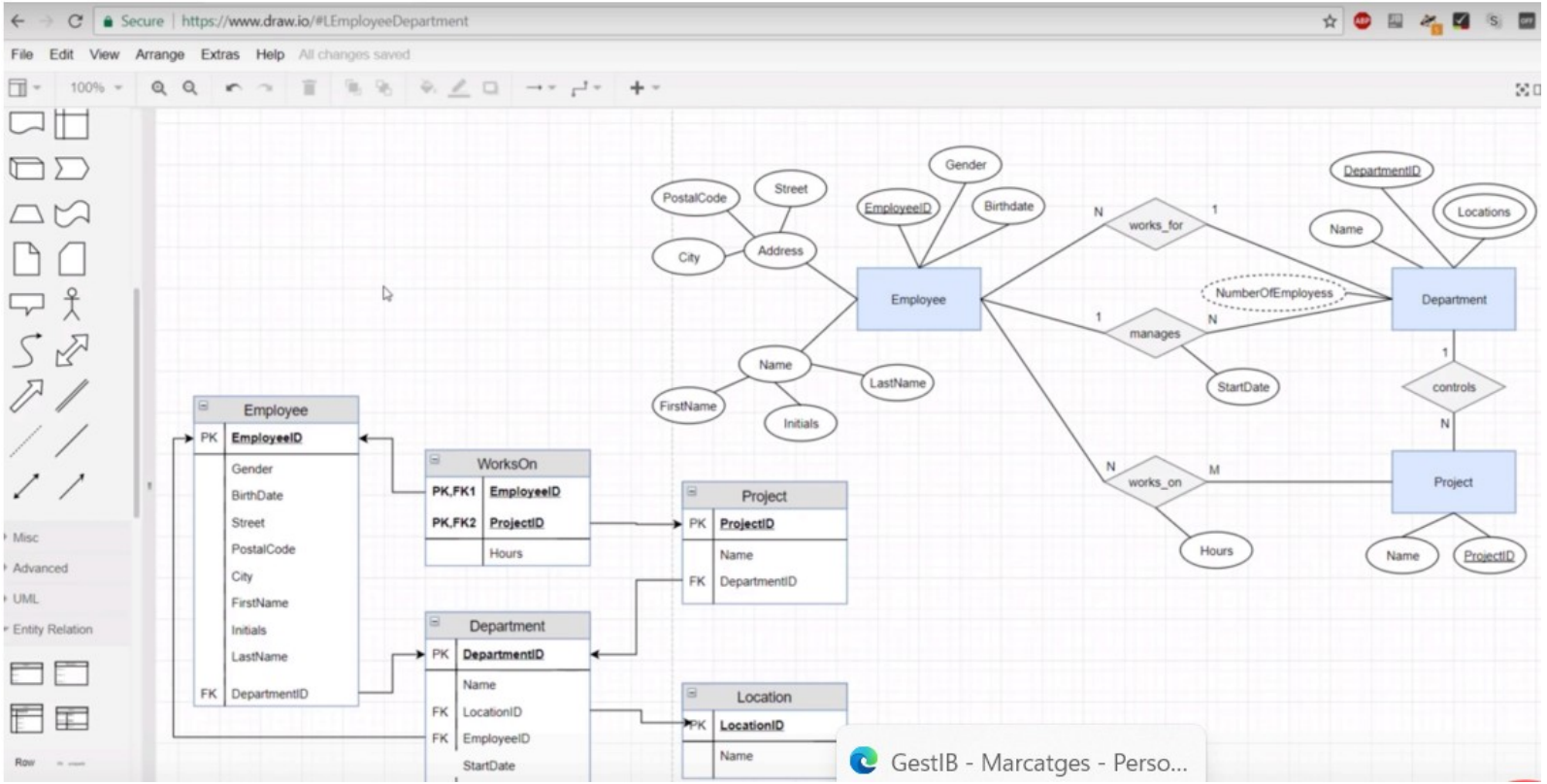
1. A(a1, a2)
2. R(a1, b1)
3. B(b1, b2)



Then, **3 tables** will be required

1. Customer(cusId, name, salary)
2. CusCar(cusId, carId)
3. Car(carId, model, color)

EXAMPLE OF CONVERT AN ER DIAGRAM TO THE RELATIONAL DATA MODEL



PRACTICE PROBLEM BASED ON CONVERTING ER DIAGRAM TO TABLES