# RELATIONAL DATA MODEL

# Relational DBMS

- Edgar F. Codd at IBM invented the relational database in 1970. Called Father of RDBMS.
- The main elements of RDBMS are based on Codd's 13 rules for a relational system.
- Tables (or relations) are related to each other by sharing common characteristics

# RDBMS

A database management system that stores data in the form of **related tables** is called Relational Database Management System.

The goal of RDBMS is to make data easy to store & retrieve

Relational databases help solve problems as they are designed to create tables & then combine the information in interesting ways to create valid information.

# RDBMS

**Typical RDBMS include**

Microsoft Access

Microsoft SQL Server

Sybase (The forerunner of Microsoft SQL Server)

IBM DB2

Oracle

Ingres

MySQL

Postgresql  etc

# RDBMS

**Relation Instance**:- snapshot of DB

Example:

| *branch-name* | *branch-city* | *assets* |
|---|---|---|
| Brighton | Brooklyn | 7100000 |
| Downtown | Brooklyn | 9000000 |
| Mianus | Horseneck | 400000 |
| North Town | Rye | 3700000 |
| Perryridge | Horseneck | 1700000 |
| Pownal | Bennington | 300000 |
| Redwood | Palo Alto | 2100000 |
| Round Hill | Horseneck | 8000000 |

**Schema** :- Logical design of DB.

Example:

Account-schema = (account-number, branch-name, balance)

Branch-schema = (branch-name, branch-city, assets)

Customer-schema = (customer-name, customer-street, customer-city)

# RDBMS

Schema Diagram

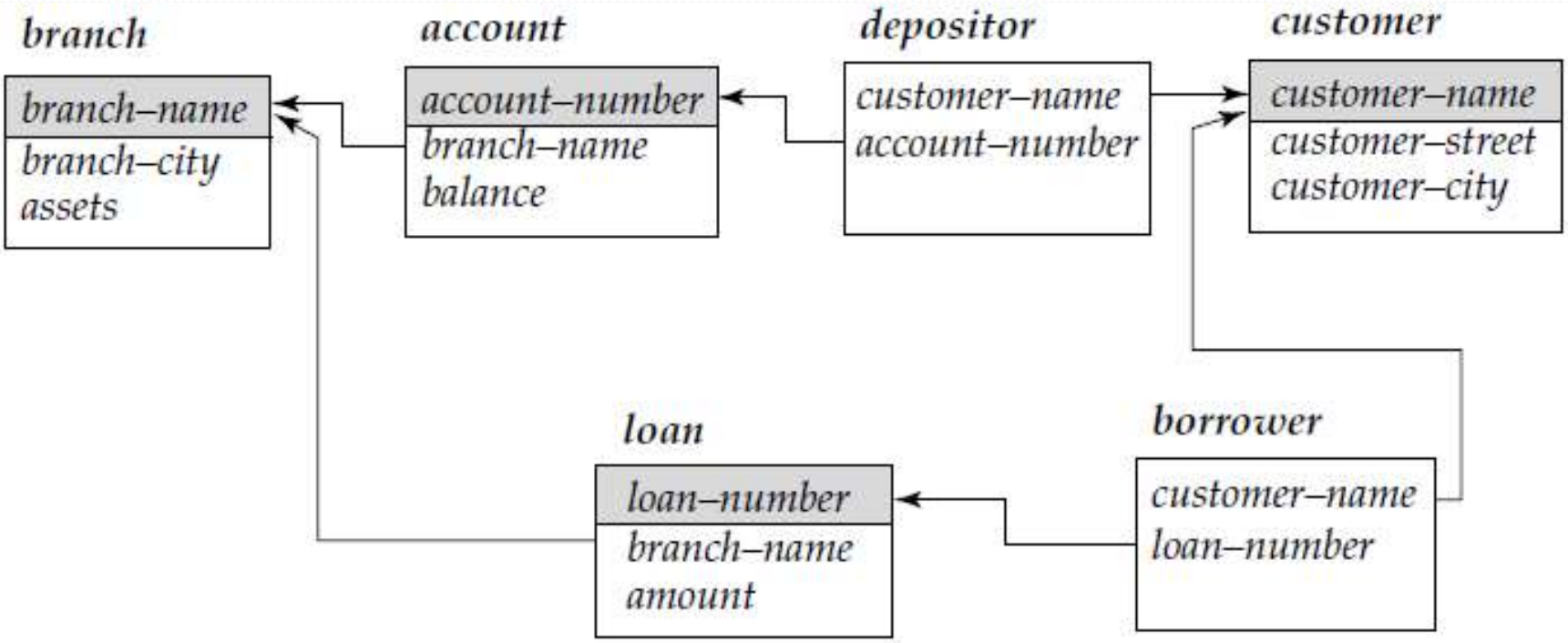
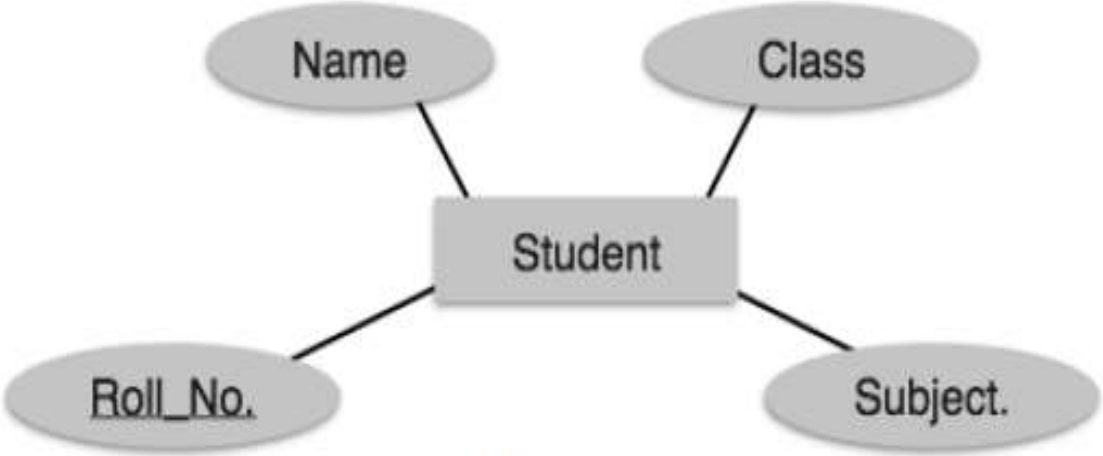



**Figure 3.9** Schema diagram for the banking enterprise.

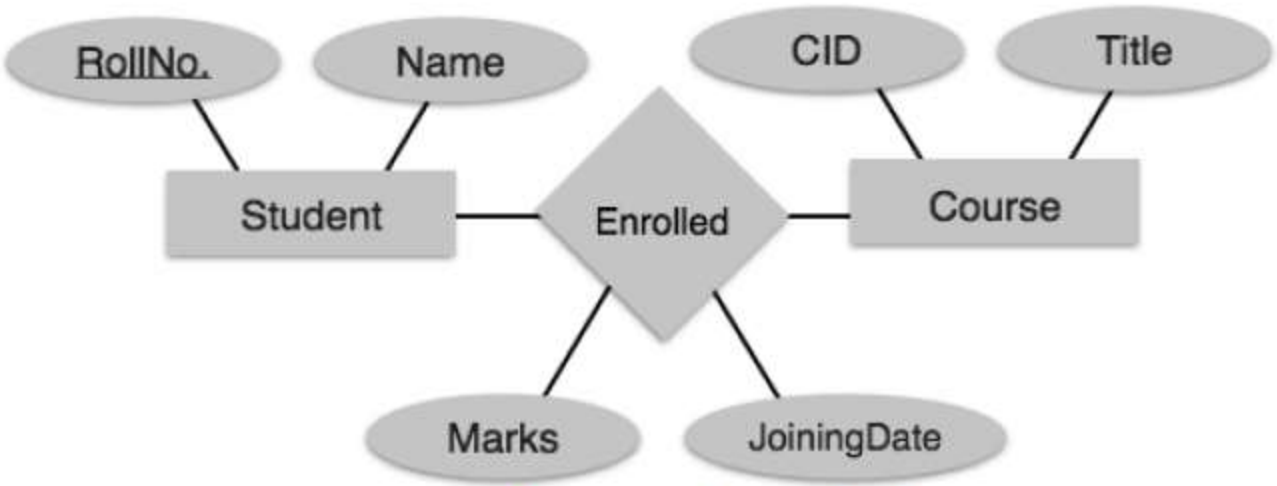
# Translation ER Model to Relational Model




[Image: Mapping Entity]

- Create table for each entity
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key

# Translation ER Model to Relational Model




[Image: Mapping relationship]

- Create table for a relationship
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

# Integrity Constraints

# Integrity Constraint

An **integrity constraint** (**IC**) is a condition specified on a database schema and restricts the data that can be stored in an instance of the database.

If a database instance satisfies all the integrity constraints specifies on the database schema, it is a **legal** instance.

A DBMS permits only legal instances to be stored in the database.

# Integrity Constraint

**Domain Constraints:**

A relation schema specifies the domain of each field in the relation instance. These domain constraints in the schema specify the condition that each instance of the relation has to satisfy.

Example:

create domain Dollars numeric(12,2)

create domain Pounds numeric(12,2)

define the domains Dollars and Pounds to be decimal numbers with a total of 12 digits, two of which are placed after the decimal point.

# Integrity Constraint

**Referential Integrity:**

ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.

Example: SQL

branch-name char(15) references branch

# Key Constraints

**Super Key:**

An attribute, or set of attributes, that **uniquely** identifies a tuple within a relation.

However, a super key may contain additional attributes that are not necessary for a unique identification.

**Candidate Key:**

A super key such that no proper subset is a super key within the relation. There are two parts of the candidate key definition:

i. Two distinct tuples in a legal instance cannot have identical values in all the fields of a key.

ii. No subset of the set of fields in a candidate key is a unique identifier for a tuple.

# Key Constraints

**Primary Key:**

The candidate key that is selected to identify tuples uniquely within the relation.

The candidate keys that are not selected as the primary key are called as **alternate keys**.

Features of the primary key:

1. Primary key will not allow duplicate values.

2.Primary key will not allow null values.

3. Only one primary key is allowed per table.

# Key Constraints

**Foreign Key:**

Foreign keys represent the relationships between tables. A foreign key is a column (or a group of columns) whose values are derived from the primary key of some other table.

Features of foreign key:

1. Records cannot be inserted into a detail table if corresponding records in the master table do not exist.
2. Records of the master table cannot be deleted or updated if corresponding records in the detail table actually exist.

# Integrity Constraint

Assertions:

An assertion is a predicate expressing a condition that we wish the database always to satisfy.

create assertion <assertion-name> check <predicate>

Example :

create assertion *sum-constraint* check

(not exists (select * from *branch*

where (select sum(*amount) from loan*

where *loan.branch-name = branch.branch-name)*

>= *(select sum(balance) from account*

where *account.branch-name = branch.branch-name)))*

# Integrity Constraint

Triggers:

A trigger is a statement that the system executes automatically as a **side effect** of a modification to the database.

To design a trigger mechanism, we must meet two requirements:

1. Specify when a trigger is to be executed. This is broken up into an **event** that causes the trigger to be checked and a **condition** that must be satisfied for trigger execution to proceed.
2. Specify the **actions** to be taken when the trigger executes.

# Integrity Constraint

Triggers:

Example: define a trigger that replaces the blank value in a phone number field of an inserted tuple by the null value.

create trigger *setnull-trigger before update on r* referencing new row as *nrow* for each row when *nrow.phone-number* = ' ' set *nrow.phone-number* = *null*

# Relational Algebra

# Relational Algebra

**Fundamental Operations**

Unary Operation

- Select
- Project
- Rename

Binary Operations

- Union
- Set different
- Cartesian product

# Relational Algebra

**Select (σ)** :

(a) Example: select those tuples of the *loan relation where the branch is "Perryridge,"*

$\sigma_{\text{branch-name} = \text{"Perryridge"}}$ (loan)

(b) Example: find those tuples pertaining to loans of more than $1200 made by the Perryridge branch

$\sigma_{\text{branch-name} = \text{"Perryridge"} \wedge \text{amount} > 1200}$ *(loan)*

# Relational Algebra

**Project(Π)**

Example: list all loan numbers and the amount of the loan as

$$\Pi_{\text{loan-number, amount}}(\text{loan})$$

**Union(∪)**

Example: find the names of all bank customers who hav either an account or a loan or both.

$\Pi_{\text{customer-name}}$ (borrower ) ∪ $\Pi_{\text{customer-name}}$ (depositor)

borrower : Customers with a loan in the bank

# Relational Algebra

**Set Difference** (-)

Example: find all customers of the bank who have an account but not a loan

$\Pi_{\text{customer-name}}$ (depositor) – $\Pi_{\text{customer-name}}$ (borrower )

**Cartesian-Product (X)**

Example: find the names of all customers who have a loan at the Perryridge branch.

3. $\Pi_{customer\text{-}name}(\sigma_{borrower.loan\text{-}number = loan.loan\text{-}number}(\sigma_{branch\text{-}name = \text{“Perryridge”}}(borrower \times loan)))$

# Relational Algebra

**Rename (ρ)**

relational-algebra expression *E*,

ρ x (E)

returns the result of expression *E under the name x*.

*Additional operations are:*

- *Set intersection*
- *Assignment*
- *Natural join*

# Relational Calculus

**Tuple Relational Calculus**

A query in the tuple relational calculus is expressed as

$$\{t \mid P(t)\}$$

the set of all tuples *t such that predicate P is true for t.*

**Domain Relational Calculus**

An expression in the domain relational calculus is of the form

$$\{< x1, x2, \ldots, xn > \mid P(x1, x2, \ldots, xn)\}$$

where x1, x2, . . . , xn represent domain variables. P represents a formula composed of atoms, as was the case in the tuple relational calculus.