3
Relational Model

- Introduction
- Reducing ER Schema to Tables
- Structure of Relational Model

Introduction

- The relational model is today the primary model for commercial data processing applications.
- It has attained its primary position because of its simplicity as compared to earlier data models such as the network model or the hierarchical model.
- It is a lower level model that uses a collection of **tables** (also called **relations**) to represent both data and the relationship among those data.
- Each table has multiple columns and each column has a unique name.

Reducing ER Schema to Tables

- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.
- Here, we will discuss different techniques to reduce ER schema to tables.
- **Representing Strong Entity Sets:**
  - A strong entity set reduces to a table with the same attributes.

Reducing ER Schema to Tables

- **Representing Composite Attributes:**
  - Composite attributes are flattened out by creating a separate attribute for each component attribute.
- **Representing Multivalued Attributes:**
  - A multivalued attribute M of an entity E is represented by a separate table EM.
  - Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M.
  - Each value of the multivalued attribute maps to a separate row of the table EM.
- **Representing Weak Entity Sets:**
  - A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set and columns for its own attributes.

Reducing ER Schema to Tables

- **Representing Relationship Sets:**
  - A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
  - Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the one side.
  - If participation is *partial* on the many side, replacing a table by an extra attribute in the relation corresponding to the "many" side could result in null values.
  - For one-to-one relationship sets, either side can be chosen to act as the "many" side. That is, extra attribute, containing the primary key, can be added to either of the tables corresponding to the two entity sets.
  - The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.

Reducing ER Schema to Tables

- **Representing Specialization:**
  - **Method 1:**
    - Form a table for the higher level entity set.
    - Form a table for each lower level entity set, include primary key of higher level entity set and local attributes.
    - **Drawback:** getting information requires accessing more than one table.
  - **Method 2:**
    - Form a table for each entity set (higher level and lower level) with all local and inherited attributes.
    - If specialization is total, table for generalized entity is not required to store information.
    - **Drawback:** data may be stored redundantly.

Reducing ER Schema to Tables

- **Representing Aggregation:**
  - To represent aggregation, create a table containing
    - Primary key of aggregated relationship
    - Primary key of the associated entity set
    - Any descriptive attributes of the relationship set
  - For example, consider the example of aggregation shown in the previous chapter. To represent aggregation *manages* between relationship *works-on* and entity set *manager*, create a table *manages*(*employee-id, branch-name, title, manager-name*).
  - Table *works-on* is redundant.

Structure of Relational Model

- Relational model represents the database as a collection of relations, each of which is assigned a unique name.
- Each relation resembles a table of values with rows and columns.
- A relation may be regarded as a set of **tuples**, also called **records**.
- **Basic Structure:**
  - A relational database consists of tables (relations).
  - Each table has column headers. These column headers are called the **attributes**. These attributes are unordered from left to right.
  - For each attribute, there is a set of permitted values, called the **domain** of that attribute.
  - The tuples are unordered from top to bottom.

Structure of Relational Model

- Let D_1 denotes the domain of attribute A_1, D_2 denotes the domain of attribute A_2 and D_n denotes domain of attribute A_n then, any row of the table must consists of n-tuple $(v_1, v_2, \ldots\ldots, v_n)$ where each v_i is in domain D_i. In general, a table of n attributes must be a subset of

$$D_1 \times D_2 \times \ldots.. \times D_{n-1} \times D_n$$

- Since, mathematicians define a relation to be a subset of a Cartesian product of a list of domains, therefore, we use the term **relation** in place of the **table** and **tuple** in place of the **row**. A **tuple variable** is a variable that stands for a tuple.

Structure of Relational Model

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Fig: Example of a relation

Structure of Relational Model

- Let the tuple variable t refers to the first tuple of the relation. We use the notation $t[A_1]$ to denote the value of t on the A_1 attribute. Alternatively, we may write $t[1]$ to denote the value of the tuple t on the first attribute, $t[2]$ to denote the value of the tuple t on the second attribute, and so on. Since a relation is a set of tuples, we use the mathematical notation of $t \in r$ to denote that tuple t is in relation r.
- The order in which tuples appear in a relation is irrelevant, since a relation is set of tuples. We require that, for all relation r, the domain is **atomic**. A domain is atomic if elements of the domain are considered to be indivisible units. It is possible for several attributes to have the same domain. One domain value that a member of any possible domain can have is the **null** value, which signifies that the value is unknown or does not exist.

Structure of Relational Model

- **Database Schema:**
  - The **database schema** is the logical design of the database, and a **database instance** is the collection of data in the database at a given instant in time.
  - The **relation schema** is the logical design of the relation. We use lowercase names for relations and names beginning with an uppercase letter for relation schemas. For example, we use *Account-schema* to denote the relation schema for relation *account*. Thus,

    Account-schema=(account-number, branch-name, balance)
  - In general, a relation schema consists of a list of attributes and their corresponding domains.
  - We denote the fact that account is a relation on Account-schema by

    account (Account-schema)
  - A **relation instance** is collection of information stored in the relation at a particular moment.

Structure of Relational Model

- The same attribute may appear in many schemas. Using common attributes is way of relating tuples of distinct relations.
- We also need a relation to describe the association between relations. For example, to describe the association between *customers* and *accounts*, the relation schema to describe this association is

  Depositor-schema (customer-id, account-number)
- And the corresponding relation is

  depositor (Depositor-schema)

- **Keys:**
  - The notion of **superkey**, **candidate key**, and **primary key**, as discussed in E-R model are also applicable to the relational model.
  - Let **R** be a relation schema. If we say that a subset **K** of **R** is a *superkey* for **R**, then values for **K** are sufficient to identify a unique tuple of each possible relation **r(R)**. That is, if *t1* and *t2* are in *r* and *t1≠t2*, then *t1[K] ≠t2[K]*.

Structure of Relational Model

- **K** is a *candidate key* if **K** is minimal and we consider one candidate key as a *primary key*.
- If a relation schema is based on tables derived from an E-R schema, it is possible to determine the primary key for a relation schema from the primary keys of the entity sets or relationship sets from which the schema is derived.
- **Strong Entity Set:**
  - The primary key of the entity set becomes the primary key of the relation.
- **Weak Entity Set:**
  - The primary key of the relation consists of the union of the primary key of the strong entity set and discriminator of the weak entity set.
- **Relationship Set:**
  - The union of the primary keys of the related entity sets becomes a superkey of the relation.

Structure of Relational Model

- If the relationship is many-to-many, this superkey is also the primary key.
- If the relationship is many-to-one or one-to-many, the primary key of the many entity set becomes the relation's primary key.
- If the relationship is one-to-one, the relation's primary key can be that of either entity set.

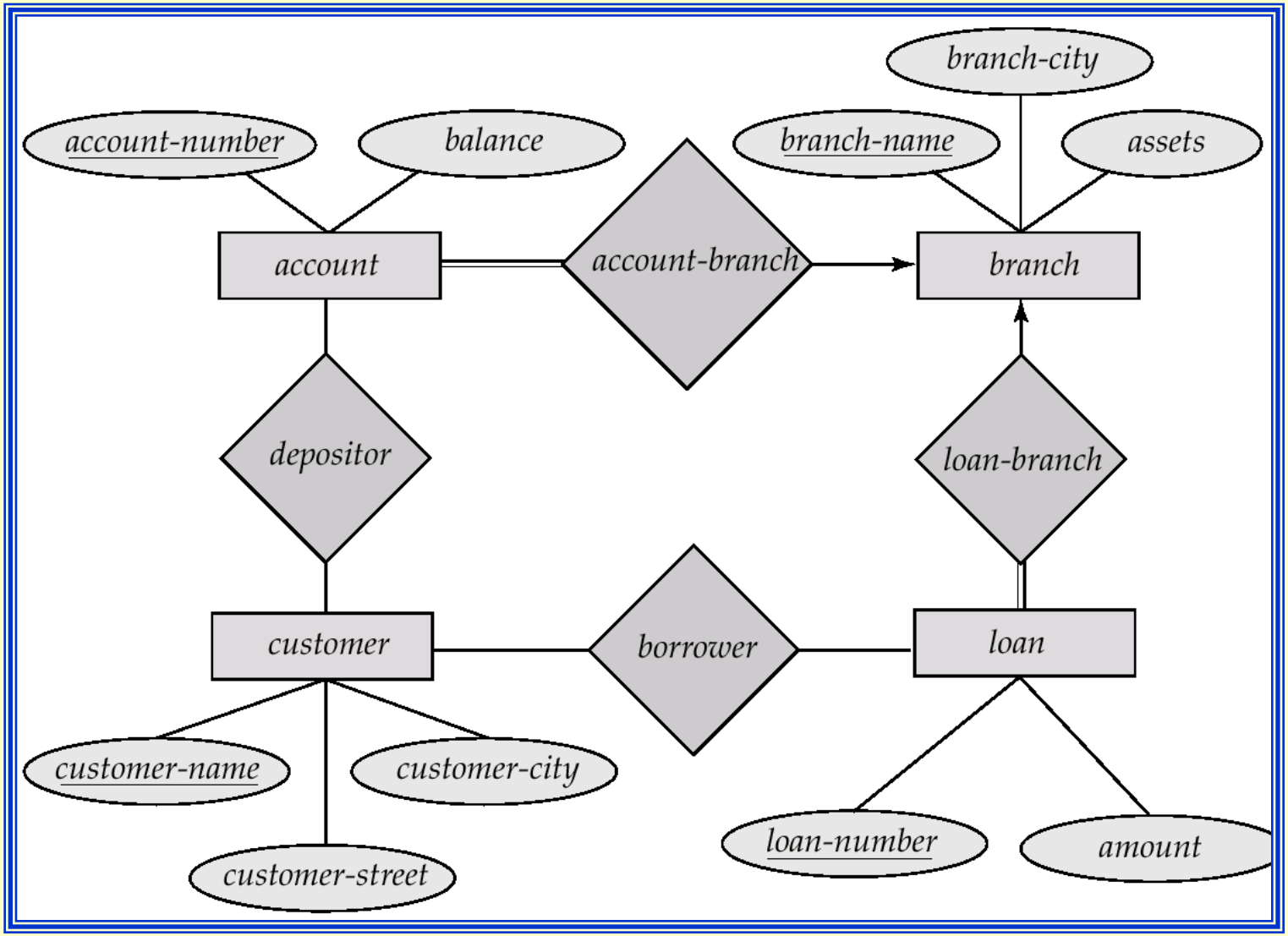
- **Combined Tables:**
  - A many-to-one or one-to-many relationship set can be represented by a relation consisting of the attributes of many entity set and attributes (if any exists) of the relationship set. The primary key of the relation is the primary key of the many entity set.
  - For one-to-one relationship set, the relation is constructed like that for may-to-one relationship set. However, we can choose either entity set's primary key as the primary key of the relation.

Structure of Relational Model

- **Multivalued Attributes:**
  - Multivalued attribute **M** is represented by a table consisting of the primary key of the entity set or relationship set of which **M** is an attribute plus a column **C** holding an individual value of **M**. The primary key of the entity set or relationship set, together with the attribute **C** becomes the primary key for the relation.
- A relation schema, say r_1, derived from an E-R schema may include among its attributes the primary key of another relation schema, say r_2. This attribute is called a **foreign key** from r_1, referencing r_2. The relation r1 is also called the referencing relation of the foreign key dependency, and r2 is called the referenced relation of the foreign key.

- **Schema Diagram:**
  - A database schema, along with primary key and foreign key dependencies can be depicted pictorially by schema diagrams. See next two slides to convert ER diagram to database schema.

Structure of Relational Model



Structure of Relational Model

- Each relation appears as a box with the attributes listed inside and the relation name above it. If there are primary key attributes, a horizontal line crosses the box with the primary key attribute above the line. Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.

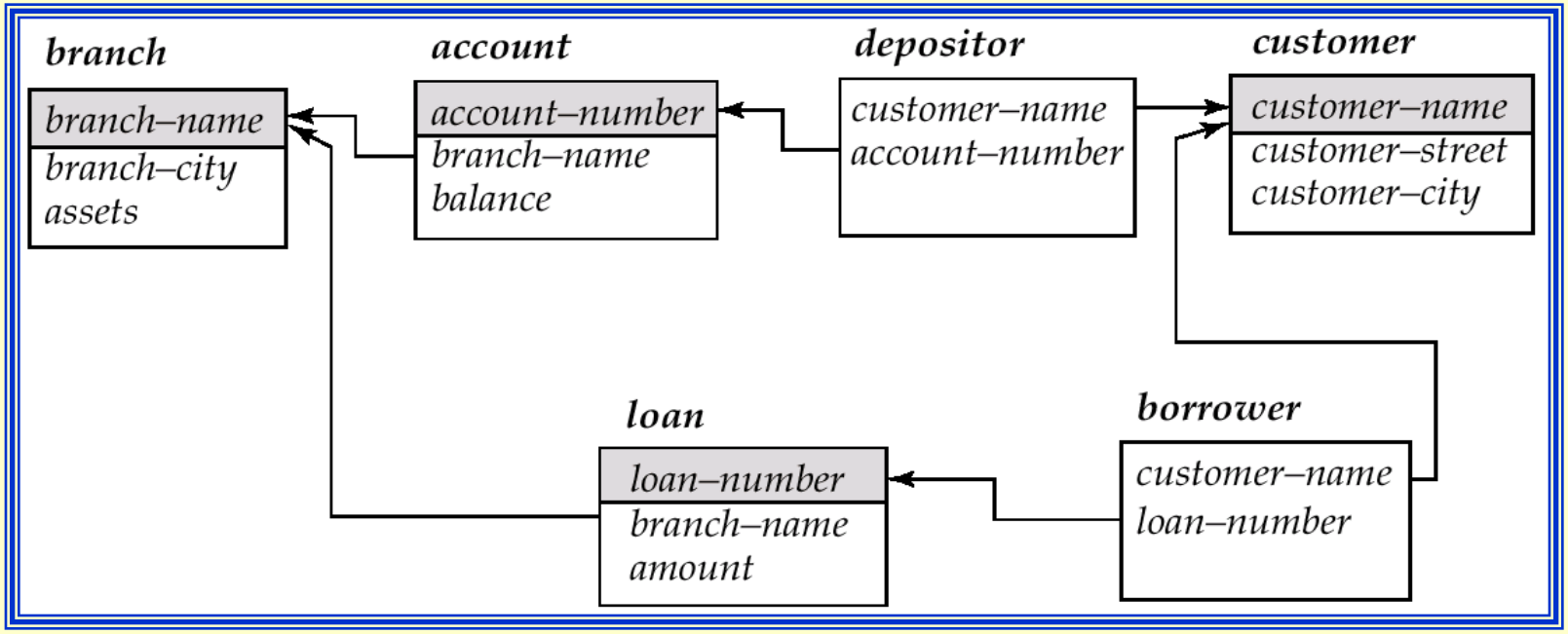


Fig: Schema diagram