**Chapter 4 The Relational Model**

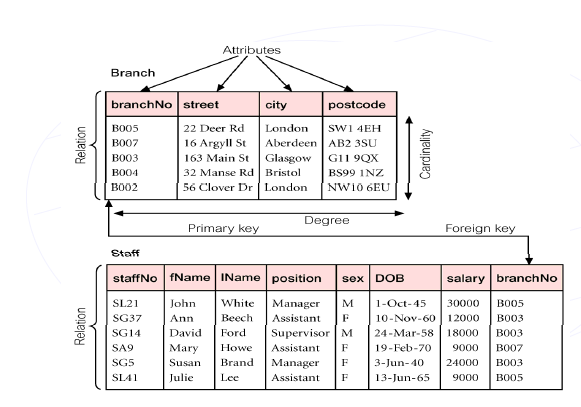
**Objectives:**

* Terminology of relational model.
* How tables are used to represent data.
* Connection between mathematical relations and relations in the relational model.
* Properties of database relations.
* How to identify candidate, primary, and foreign keys.
* Meaning of entity integrity and referential integrity.
* Purpose and advantages of views.

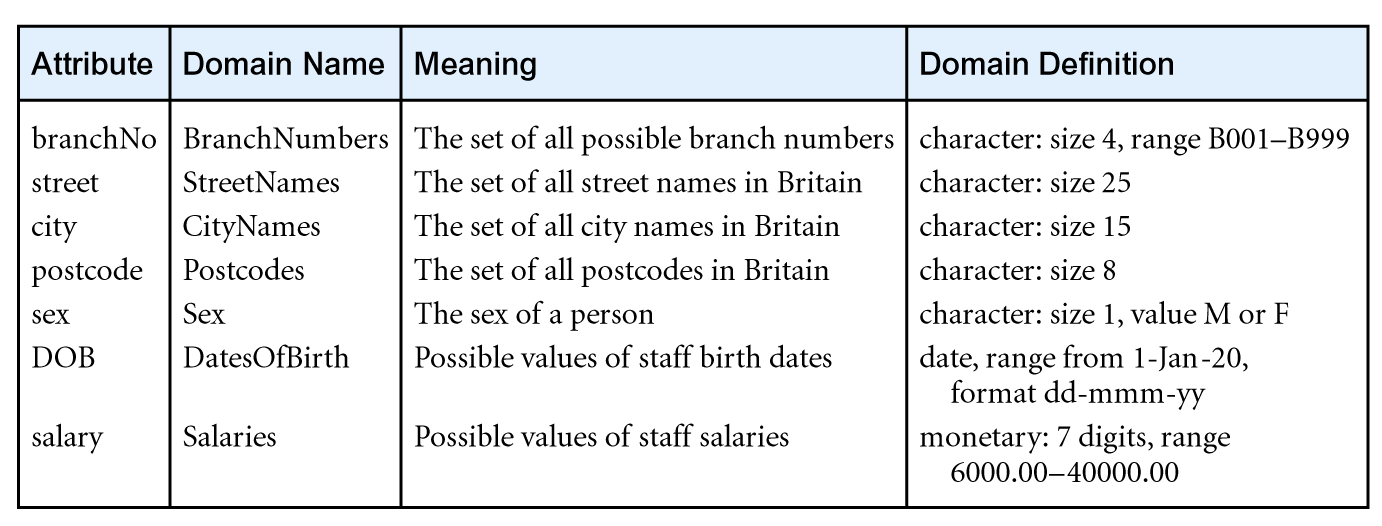
**Relational Model Terminology**

* A relation is a table with columns and rows.
  + Only applies to logical structure of the database, not the physical structure.
* Attribute is a named column of a relation.
* Domain is the set of allowable values for one or more attributes.
* Tuple is a row of a relation.
* Degree is the number of attributes in a relation.
* Cardinality is the number of tuples in a relation.
* Relational Database is a collection of normalized relations with distinct relation names.

**Instances of Branch and Staff (part) Relations**

****

**Examples of Attribute Domains**



**Mathematical Definition of Relation**

* Consider two sets, D1 & D2, where D1 = {2, 4} and D2 = {1, 3, 5}
* Cartesian product, D1 × D2, is set of all ordered pairs, where first element is member of D1 and second element is member of D2
* D1 × D2 = {(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)}
* Alternative way is to find all combinations of elements with first from D1 and second from D2
* Any subset of Cartesian product is a relation; e.g.

R = {(2, 1), (4, 1)}

* May specify which pairs are in relation using some condition for selection; e.g
  + second element is 1:

R = {(x, y) | x ε D1, y ε D2, and y = 1}

* + first element is always twice the second:

S = {(x, y) | x ε D1, y ε D2, and x = 2y}

* Consider three sets D1, D2, D3 with Cartesian Product D1 ×D2 ×D3; e.g.

D1 = {1, 3} D2 = {2, 4} D3 = {5, 6}

D1 × D2 ×D3 = {(1,2,5), (1,2,6), (1,4,5), (1,4,6), (3,2,5), (3,2,6), (3,4,5), (3,4,6)}

* Any subset of these ordered triples is a relation.
* Cartesian product of n sets (D1, D2, . . ., Dn) is:

D1 ×D2 × . . . × Dn = {(d1, d2, . . . , dn) | d1 ε D1, d2 ε D2, . . . , dnε Dn}

usually written as:

***n***

**X *Di***

***i* = 1**

* Any set of *n*-tuples from this Cartesian product is a relation on the *n* sets

**Relation schema**

Named relation defined by a set of attribute and domain name pairs

**Relational database schema**

Set of relation schemas, each with a distinct name

**Properties of Relations**

* Relation name is distinct from all other relation names in relational schema
* Each cell of relation contains exactly one atomic (single) value
* Each attribute has a distinct name
* Values of an attribute are all from the same domain
* Each tuple is distinct; there are no duplicate tuples
* Order of attributes has no significance
* Order of tuples has no significance, theoretically

**Relational Keys**

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

Branch (branchNo, street, city, postcode)

Viewing (clientNo, propertyNo, viewDate, comment )

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

* **Candidate Key**
* Superkey (K) such that no proper subset is a superkey within the relation
* In each tuple of R, values of K uniquely identify that tuple (uniqueness)
* No proper subset of K has the uniqueness property (irreducibility)

Branch (branchNo, street, city, postcode)

Viewing (clientNo, propertyNo, viewDate, comment )

* **Primary Key**
* Candidate key selected to identify tuples uniquely within relation
* **Alternate Keys**
* Candidate keys that are not selected to be primary key
* **Foreign Key**
* Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation

Branch (branchNo, street, city, postcode)

Viewing (clientNo, propertyNo, viewDate, comment )

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

**Integrity Constraints**

* **Null**
* Represents value for an attribute that is currently unknown or not applicable for tuple
* Deals with incomplete or exceptional data
* Represents the absence of a value and is not the same as zero or spaces, which are values
* **Entity Integrity**
* In a base relation, no attribute of a primary key can be null
* **Referential Integrity**
* If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null
* **General Constraints**
* Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise

**Base Relation**

Named relation corresponding to an entity in conceptual schema, whose tuples are physically stored in database.

**View**

* Dynamic result of one or more relational operations operating on base relations to produce another relation
* A virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.
* Contents of a view are defined as a query on one or more base relations.
* Views are dynamic, meaning that changes made to base relations that affect view attributes are immediately reflected in the view.

**Purpose of Views**

* Provides powerful and flexible security mechanism by hiding parts of database from certain users.
* Permits users to access data in a customized way, so that same data can be seen by different users in different ways, at same time.
* Can simplify complex operations on base relations.

**Updating Views**

* All updates to a base relation should be immediately reflected in all views that reference that base relation.
* If view is updated, underlying base relation should reflect change.
* There are restrictions on types of modifications that can be made through views:
  + Updates are allowed if query involves a single base relation and contains a candidate key of base relation.
  + Updates are not allowed involving multiple base relations.
  + Updates are not allowed involving aggregation or grouping operations.