The Relational Model

Topics List

- Relational Model Terminology
- Properties of Relations
- Relational Keys
- Integrity Constraints
- 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.

Relational Model Terminology

- **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 normalised relations with distinct relation names**.**

Instances of Branch and Staff Relations

Attributes

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Relation

Cardinality

Degree

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relation

Examples of Attribute Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Alternative Terminology for Relational Model

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Topics List

- Relational Model Terminology
- Properties of Relations
- Relational Keys
- Integrity Constraints
- Views

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.

Properties of Relations

- Each tuple is distinct; there are no duplicate tuples.
- Order of attributes has no significance.
- Order of tuples has no significance, theoretically.

Topics List

- Relational Model Terminology
- Properties of Relations
- Relational Keys
- Integrity Constraints
- Views

Relational Keys

- **Superkey**
  - Super key stands for superset of a key. A Super Key is a set of one or more attributes that are taken collectively and can identify all other attributes uniquely.
  - An attribute, or set of attributes, that uniquely identifies a tuple within a relation.
  - A superkey is any set of attributes for which the values are guaranteed to be unique for all possible sets of tuples in a table at all times.

Relational Keys

- **Candidate Key**
  - A candidate key is a superkey K such that no proper subset of K is also a superkey.
  - This is often summarised by saying that a superkey is unique whereas a candidate key is both unique and irreducible.

Relational Keys

For Example, We have a table called Employee where we know that each *Emp_SSN* is unique, and each Emp_Number is unique.

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Robert

Relational Keys

- In this table we can have the following super keys:
  - {Emp_SSN}
  - {Emp_Number}
  - {Emp_SSN, Emp_Number}
  - {Emp_SSN, Emp_Name}
  - {Emp_SSN, Emp_Number, Emp_Name}
  - {Emp_Number, Emp_Name}

- All of the above sets are able to uniquely identify rows of the employee table.

Relational Keys

- And in this table we can have the following candidate keys:
  - {Emp_SSN}
  - {Emp_Number}
- Only these two sets are candidate keys as all other sets are having redundant attributes that are not necessary for unique identification.

Relational Keys

- **Primary Key**
  - Candidate key selected to identify tuples uniquely within a relation.

- **Alternate Keys**
  - Candidate keys that are not selected to be the primary key.

- **Foreign Key**
  - Attribute, or set of attributes whose value within one relation matches the primary key value of another relation.

Relational Keys

Relational Database Keys

KEY TYPE	DEFINITION
Superkey	An attribute (or combination of attributes) that uniquely identifies each entity in a table.
Candidate key	A minimal superkey. A superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

Instances of Branch and Staff Relations

Primary key

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Primary key

Foreign key

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Topics List

- Relational Model Terminology
- Properties of Relations
- Relational Keys
- Integrity Constraints
- Views

Integrity Constraints

- **Null**
  - Represents value for an attribute that is currently unknown or not applicable for this 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.

Integrity Constraints

- **Entity Integrity**
  - In a base relation, no attribute of a primary key can be null. Primary key must also be unique.

- **Referential Integrity**
  - If foreign key exists in a relation, either foreign key value must match the primary (or alternate) key value of some tuple in its home relation or foreign key value must be wholly null**.**

Integrity Constraints

INTEGRITY RULES

ENTITY INTEGRITY	DESCRIPTION
Requirement	All primary key entries are unique, and no part of a primary key may be null.
Purpose	Guarantees that each entity will have a unique identity and ensures that foreign key values can properly reference primary key values.
Example	No invoice can have a duplicate number, nor can it be null. In short, all invoices are uniquely identified by their invoice number.
REFERENTIAL INTEGRITY	**DESCRIPTION**
Requirement	A foreign key may have either a null entry—as long as it is not a part of its table's primary key—or an entry that matches the primary key value in a table to which it is related. (Every non-null foreign key value *must* reference an *existing* primary key value.)
Purpose	Makes it possible for an attribute NOT to have a corresponding value, but it will be impossible to have an invalid entry. The enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not (yet) have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

Integrity Constraints

An Illustration of Integrity Rules

Table name: CUSTOMER
Primary key: CUS_CODE
Foreign key: AGENT_CODE

Database name: Ch03_InsureCo

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	12-Mar-02	502
10011	Dunne	Leona	K	713	894-1238	23-May-02	501
10012	Smith	Kathy	W	615	894-2285	05-Jan-03	502
10013	Olowski	Paul	F	615	894-2180	20-Sep-02	
10014	Orlando	Myron		615	222-1672	04-Dec-02	501
10015	O'Brian	Amy	B	713	442-3381	29-Aug-02	503
10016	Brown	James	G	615	297-1228	01-Mar-03	502
10017	Williams	George		615	290-2556	23-Jun-02	503
10018	Farriss	Anne	G	713	382-7185	09-Nov-02	501
10019	Smith	Olette	K	615	297-3809	18-Feb-03	503

Table name: AGENT
Primary key: AGENT_CODE
Foreign key: none

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	$1,735,453.75
502	615	882-1244	Hahn	$4,967,003.28
503	615	123-5589	Okon	$3,093,980.41

Integrity Constraints

- **General Constraints**
  - Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.
  - For Example:
    In the PropertyForRent table, type must be House, Flat or Apartment.

Topics List

- Relational Model Terminology
- Properties of Relations
- Relational Keys
- Integrity Constraints
- Views

Views

- **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.

Views

- A virtual relation that does not actually exist in the database but is produced upon 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 reflected in the view.

Views

- A view name may be used in exactly the same way as a table name in any SELECT query. Once stored, the view can be used again and again, rather than re-writing the same query many times.
- One of the most important uses of views is in large multi-user systems, where they make it easy to control access to data for different types of users.

Views - Example

- As a very simple example, suppose that you have a table of employee information

Employee(PPS, fName, lName, phone, jobTitle, payRate, managerID).

- Obviously, you can’t let everyone in the company look at all of this information, let alone make changes to it.

- Only a very few trusted people would have SELECT, UPDATE, INSERT, and DELETE privileges on the entire Employee base table; everyone else would have exactly the access that they need, but no more.

Views - Example

- You could create separate views on just the Employee table, and control access to it like this:

```
CREATE VIEW phone_view AS
  (SELECT fName, lName, phone
  FROM Employee);
```

```
GRANT SELECT ON phone_view TO public;
```

Views - Example

```
CREATE VIEW job_view AS
 (SELECT PPS, fName, lName, jobTitle,
 managerID
 FROM Employee);
```

```
GRANT SELECT, UPDATE ON job_view TO
managers;
```

Views - Example

```
CREATE VIEW pay_view AS
 (SELECT PPS, fName, lName, payRate
 FROM Employee);
```

```
GRANT SELECT, UPDATE ON pay_view TO payroll;
```

Views - Example

```
CREATE VIEW sales_rate AS
 (SELECT PPS, fName, lName, payRate
 FROM Employee
 where jobTitle = ‘Sales’);
```

```
GRANT SELECT ON sales_rate TO managers;
```

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. For example, rather than writing a query that involved 2 or more tables. Create the view once and then query the view as often as required.

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.