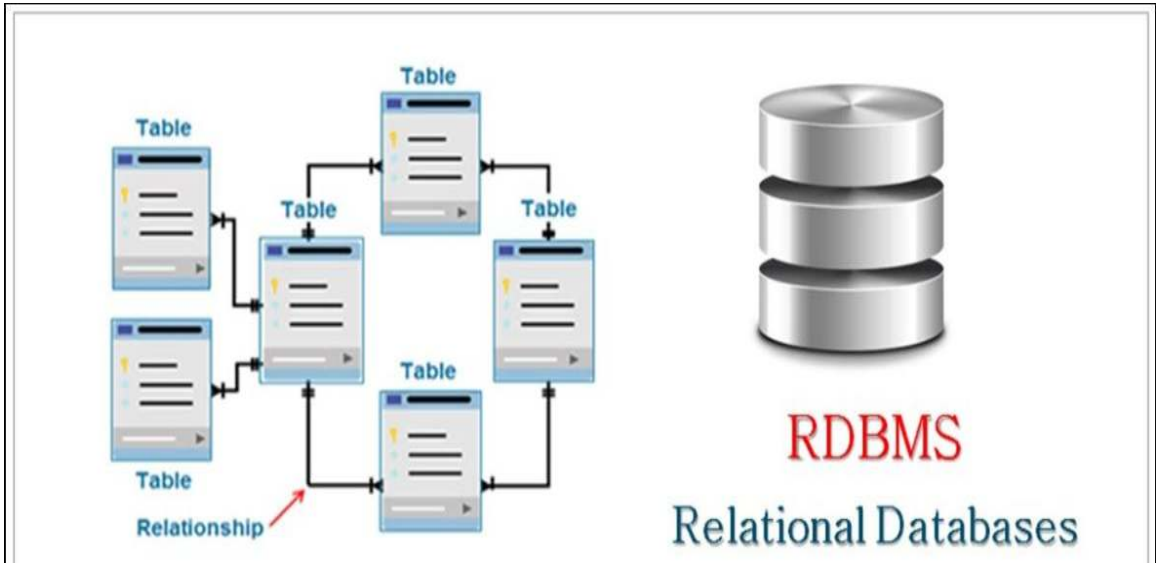
University of Bisha
Faculty Of Computing and Information Technology

Part (5)

Relational Database Management System (RDBMS)



3/3/2020

Outlines of Part 5 → use Chapter 5

- Introduction
- RDBMS Terminology
- Relational Data Structure
- Relational Data Manipulation
- Codd's Rules

Introduction

- **Relational Model Concepts**
- ***A relational database management system (RDBMS)*** is a database management system (DBMS) that is based on the relational model as introduced by *E. F. Codd*.
- The relational Model of Data is based on the concept of a ***Relation.***
- A short definition of an RDBMS may be a DBMS in which data is stored in the form of **tables** and the **relationship** among the data is also stored in the form of tables.
- A Relation is a mathematical concept based on the ideas of sets.
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:
  - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970

RDBMS Terminology

The relations we have been dealing with are known as base relations.

- **Base relation**

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

We can define a view in terms of base relations.

- **View**

The dynamic result of one or more relational operations operating on the base relations to produce another relation.

A view is a *virtual relation* that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request.

Relational Data Structure

❖ Relation:

- Informally, A relation is a ***table*** with columns and rows.
-Relation name is distinct from all other relation names in relational schema.
-Each cell of the relation contains exactly one atomic (single) value;

❖ Attribute:

- Informally, An attribute is a named ***column*** of a relation.
- Each attribute has a distinct name.
- the order of attributes has no significance

❖ Domain:

- is the set of allowable values for an attribute.
- 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

Relational Data Structure…..Cont

❖ Degree:

- The degree of a relation is the number of attributes of its schema.(***unary***, ***binary, ternary***, and after that the term ***n-ary*** is usually used)
- ex.: Car(Licence#, EngineSerial#, Make, Model, Year) is of degree 5.
- The degree of a relation is a property of the ***intension*** of the relation.

❖ Tuple:

- A tuple is a row of a relation.
- Each tuple is distinct; there are no duplicate tuples;

❖ Cardinality:

- The cardinality of a relation is the number of tuples it contains.
- The cardinality is a property of the ***extension*** of the relation and is determined from the particular instance of the relation at any given moment (changes as tuples are added or deleted).

Relational Data Structure…..Cont

❖ Relational database:

- A collection of normalized relations with distinct relation names.

❖ Relational Database Schema:

- A set S of relation schemas that belong to the same database.
- S is the name of the whole **database schema**
- S = {R1, R2, …, Rn}
- R1, R2, …, Rn are the names of the individual **relation schemas** within the database S
- A COMPANY database schema with 6 relation schemas is shown in figer 5.5

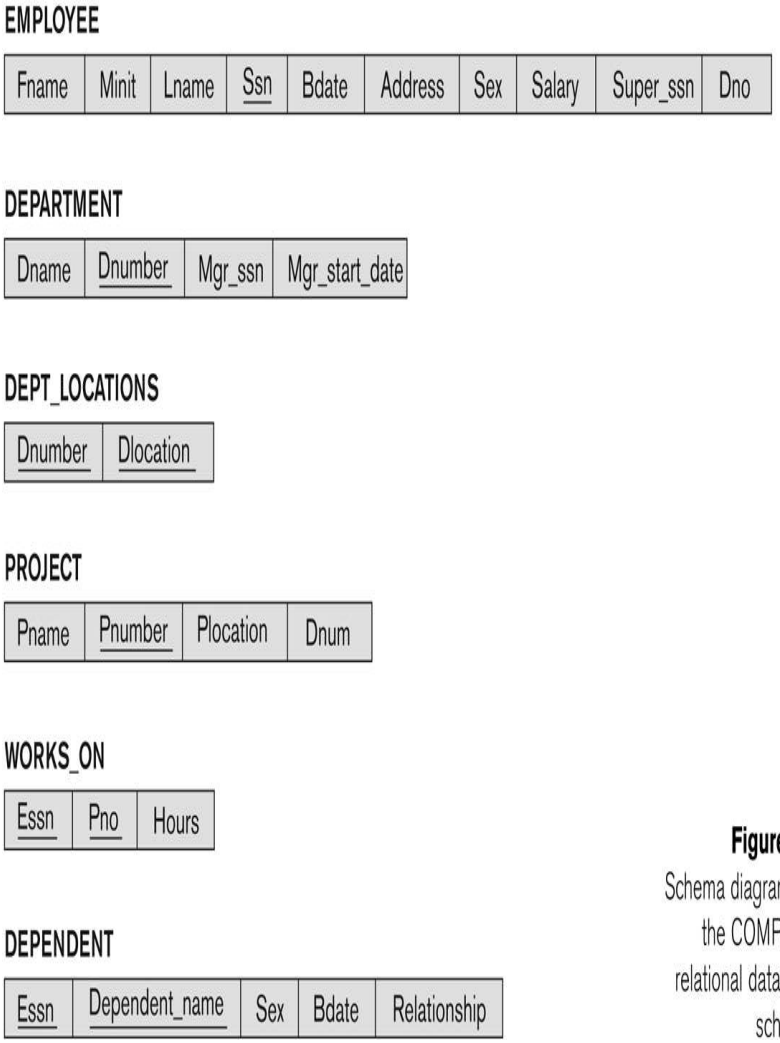


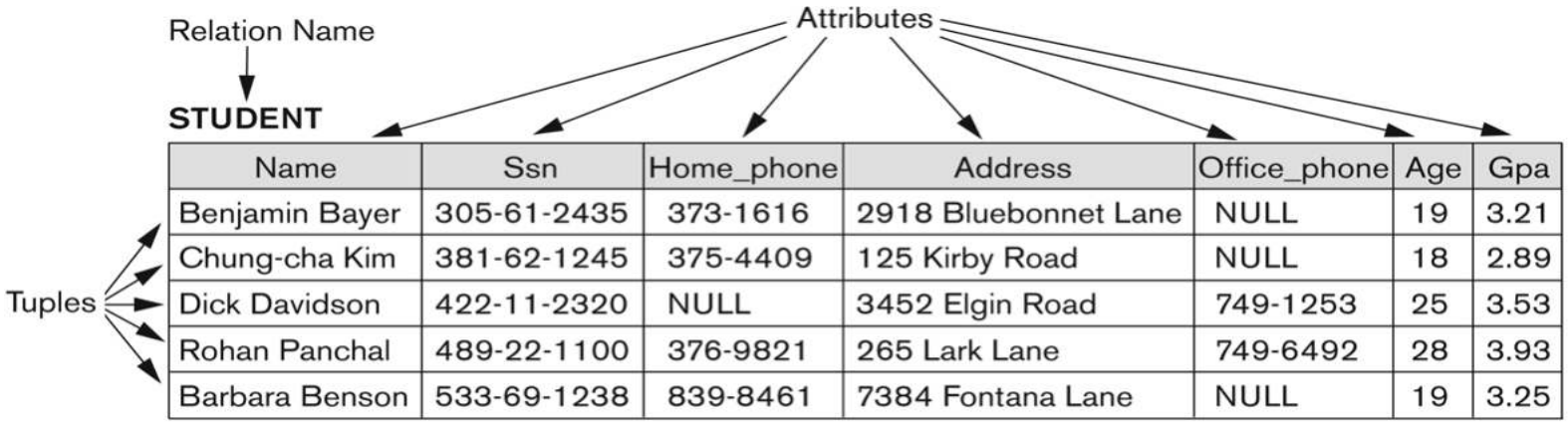
Figure 5.5
Schema diagram for the COMPANY relational database schema.

Relational Data Structure…..Cont

❖Alternative Terminology for Relational Model:

Informal Terms	Formal Terms	Alternative
Table	Relation	File
Column Header	Attribute	Field
All possible Column Values	Domain	
Row	Tuple	Record

❖Example – 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

Figure 5.1
The attributes and tuples of a relation STUDENT.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three ***main types*** of constraints in the relational model:
  - ***Key*** constraints
  - ***Entity integrity*** constraints
  - ***Referential integrity*** constraints

- Another implicit constraint is the ***domain*** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute

Relational Integrity Constraints….Cont.
Key Constraints

❖ **Superkey:**

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

❖ **Key of R:**

- A "minimal" superkey
- That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey .

Relational Integrity Constraints….Cont.
Key Constraints

❖ **Example:** Consider the CAR relation schema:

- CAR(State, Reg#, SerialNo, Make, Model, Year)
- CAR has two keys:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
- Both are also superkeys of CAR
- {SerialNo, Make} is a superkey but *not* a key.

❖ In general:

- Any *key* is a *superkey* (but not vice versa)
- Any set of attributes that *includes a key* is a *superkey*

Relational Integrity Constraints….Cont.
Key Constraints

❖ **Candidate key:**

- A superkey such that no proper subset is a superkey within the relation.

❖ **Primary key**

- The candidate key that is selected to identify tuples uniquely within the relation.
- It cannot contain null values.
- The primary key attributes are underlined.

Example: CAR table with two candidate keys – LicenseNumber chosen as Primary Key

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Relational Integrity Constraints….Cont.

Key Constraints

Simple Key & Composite Key

❖ **Simple Key**

- A candidate key that consists of one attribute.

❖ **Composite Key**

- A key that consists of two or more attributes.

- Simple Key example:

Academic_no	***address***	***Student_name***
3131144	Bisha	Sara
3156643	Bisha	Amira
3156432	Al-Aleya	Ruba

- Composite Key example:

Academic_no	***University***	***address***	***Name***
3131144	4	Bisha	Sara
3156643	7	Bisha	Amira
3156432	2	Al-Aleya	Ruba

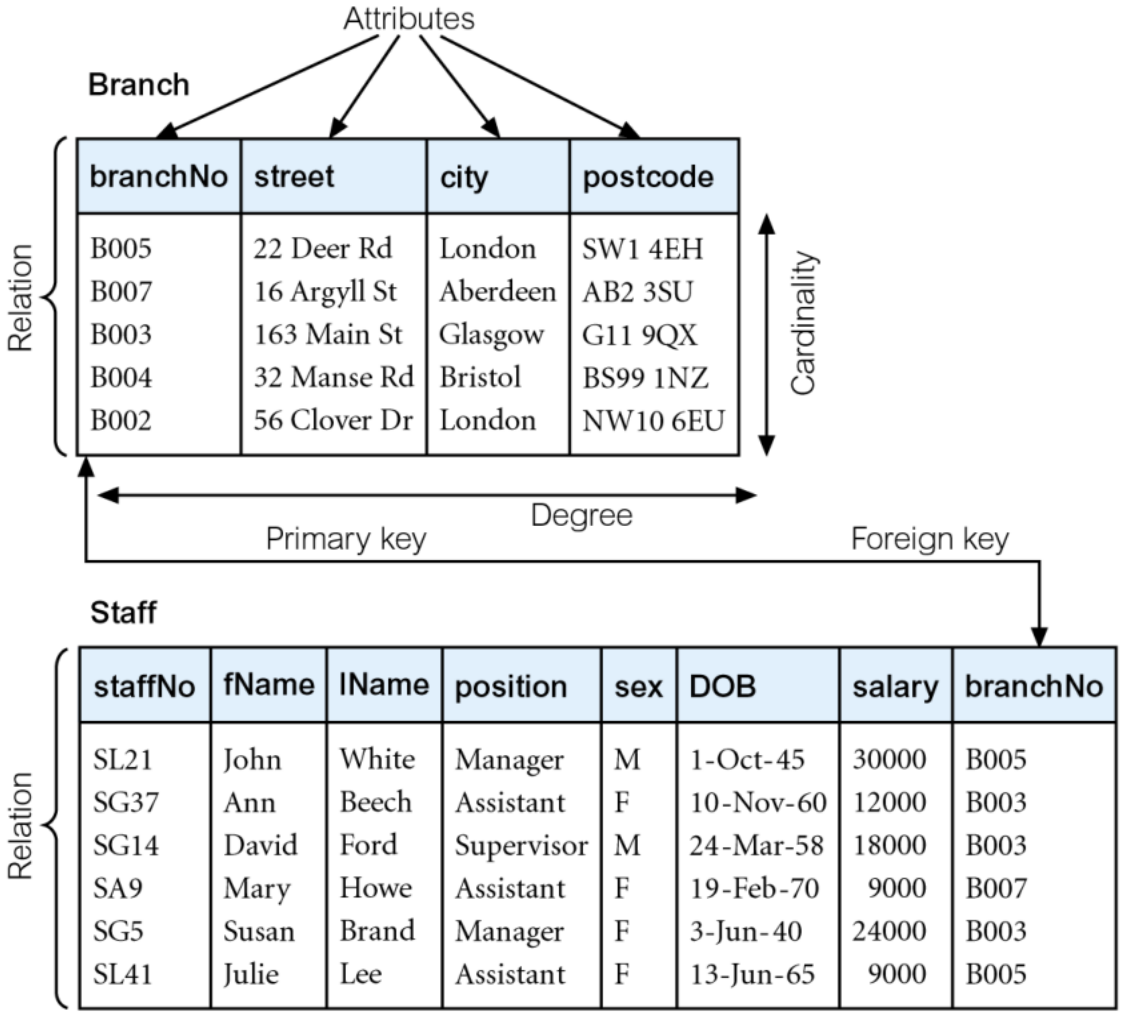
Relational Integrity Constraints….Cont.
Key Constraints

❖ Foreign Key

- is a attribute ,or set of attributes used to link two tables together (To preserve relationships).

- A foreign key is a primary key from one table placed into another table
- The key is called a foreign key in the table that received the key.

Instances of Branch and Staff Relations



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

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

Relational Integrity Constraints….Cont.
Entity Integrity constraints

- The ***primary key attributes*** PK of each relation schema R in S ***cannot*** have ***null*** values in any tuple of r(R).
  - This is because primary key values are used to *identify* the individual tuples.
  - If PK has several attributes, null is not allowed in any of these attributes

- **Note:** Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Relational Integrity Constraints….Cont.
Referential Integrity Constraints

- A constraint involving **two** relations.(The previous constraints involve a single relation).
- Used to specify a **relationship** among tuples in two relations:
  - The **referencing relation** and the **referenced relation**.
- Referential integrity ensure that every value of a foreign key column must match a value of an existing primary key
  - **Example:** If a 5 was entered as a Mgr_ID in the PROJECT table (a foreign key), a Manager having a MgrID value of 5 must exist in the Manager table.
  - The primary key value must exist before the foreign key value is entered

Relational Integrity Constraints….Cont.
Displaying a relational database schema and its constraints

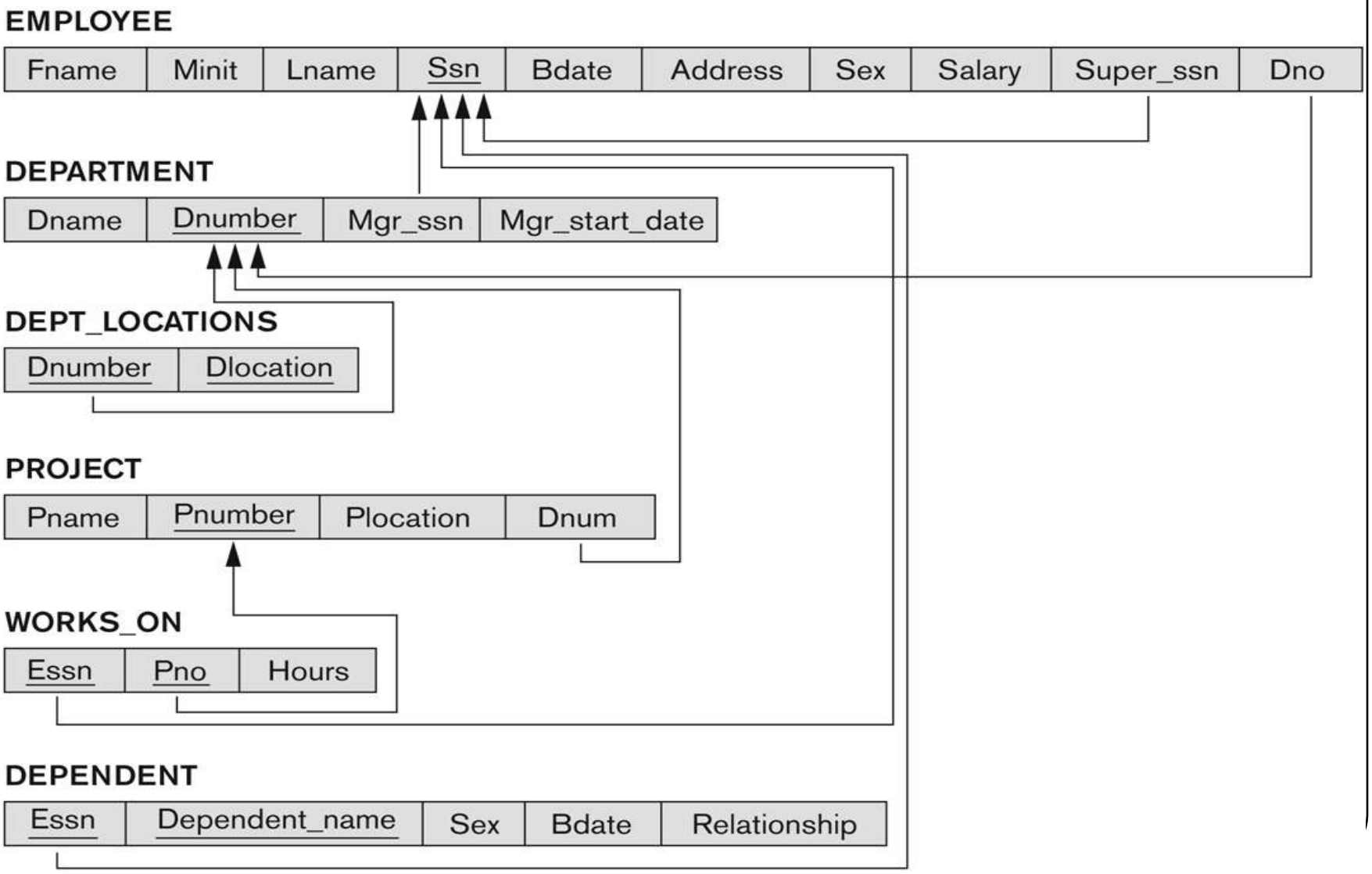
- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table
  - Can also point the the primary key of the referenced relation for clarity
- Next slide shows the COMPANY **relational schema diagram**

Relational Integrity Constraints….Cont.

Referential Integrity Constraints for COMPANY database

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational database schema.



Relational Data Manipulation

- Each ***relation*** will have many tuples in its current relation state
- The ***relational database state*** is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple

Relational Data Manipulation
Update Operations on Relations

- Integrity constraints should not be violated by the update operations.

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)

Relational Data Manipulation
Possible violations for each operation

- **INSERT** may violate ***any*** of the constraints:
  - **Domain constraint:**
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - **Key constraint:**
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - **Referential integrity:**
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - **Entity integrity:**

    ifthe primary key value is null in the new tuple

Relational Data Manipulation
Possible violations for each operation

- **DELETE** may violate ***only*** referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
- Can be remedied by several actions: RESTRICT, CASCADE, SET NULL:
  - **RESTRICT** option: reject the deletion
  - **CASCADE** option: propagate the new primary key value into the foreign keys of the referencing tuples
  - **SET NULL** option: set the foreign keys of the referencing tuples to NULL
- One of the above options must be specified during database design for each foreign key constraint

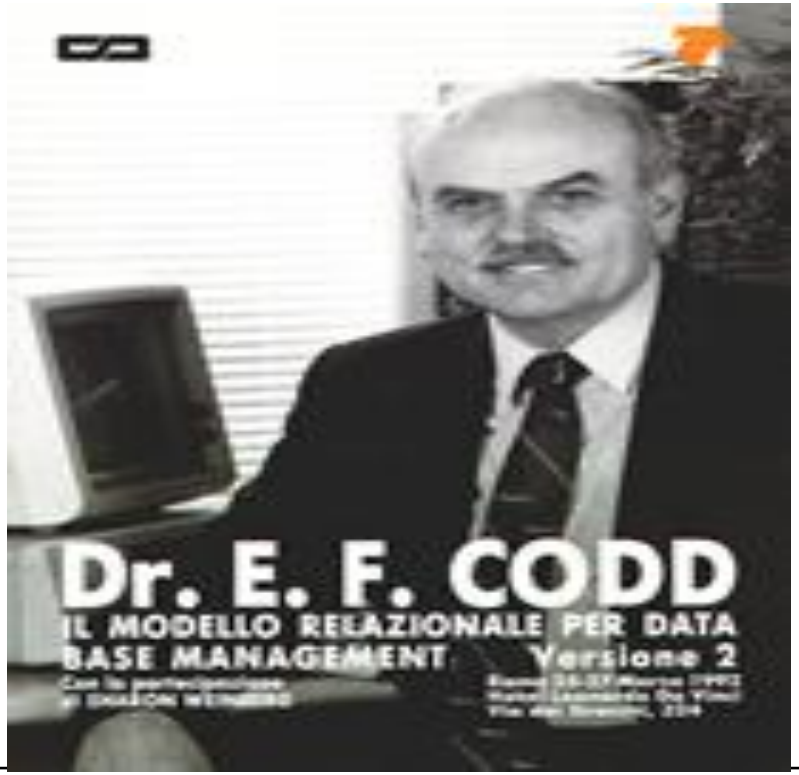
Relational Data Manipulation
Possible violations for each operation

❖ **UPDATE** may violate ***domain*** constraint and ***NOT NULL*** constraint on an attribute being modified

- Any of the other constraints may also be violated, depending on the attribute being updated:
  - ***Updating the primary key (PK):***
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - ***Updating a foreign key (FK):***
    - May violate referential integrity
  - ***Updating an ordinary attribute (neither PK nor FK):***
    - Can only violate domain constraints

Codd's Rules

- Codd proposed 13 rules popularly known as **Codd's 12 rules** to test DBMS's concept against his relational model. Codd's rule actualy define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS).



Codd's Rules…..Cont.

- **Rule zero**

  This rule states that for a system to qualify as an ***RDBMS***, it must use it's relational facilities exclusively to manage the database.

- **Rule 1: Information rule**

- All information in an RDB is represented as ***values*** in the tables.
- This is achieved by values in column and rows of tables.
- All information including table names, column names and column data types should be available in same table within the database.

- **Rule 2: Guaranteed Access**

Each unique piece of data should be accesible by : ***Table Name + Primary Key(Row) + Attribute(column).***

Codd's Rules…..Cont.

- **Rule 3: Systematic treatment of NULL**

-Null has several meanings, it can mean missing data, not applicable or no value.

- Should be handled consistently - Not Zero or Blank

-Primary key must not be null, ever.

-Expression on NULL must give null.

- **Rule 4: Location independence**

-The user should be able to access the database from any site.

-The user should be able to access all data as if it were stored at the user's site, no matter where it is physically stored.

Codd's Rules…..Cont.

- **Rule 5: Powerful and Well-Structured Language**

One well structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, etc.

- **Rule 6: View Updating Rule**

-View = "Virtual table", temporarily derived from base tables.

-Example: If a view is formed as join of 3 tables, changes to view should be reflected in base tables.

- **Rule 7: Relational Level Operation**

-This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

-Set operation like Union, Intersection and minus should also be supported.

- **Example:** Suppose if we need to change ID then it will reflect everywhere automatically.

Codd's Rules…..Cont.

- **Rule 8: Physical Data Independence**

-The physical storage of data should not matter to the system.

- A change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.

- **Rule 9: Logical Data Independence**

-If there is change in the logical structure(table structures) of the database the user view of data should not change.

-The addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

Codd's Rules…..Cont.

- **Rule 10: Integrity Independence (Catalog)**

-The database should be able to enforce its own integrity rather than using other programs.

Key and Check constraints, trigger etc, should be stored in Data Dictionary.

- **Rule 11: Distribution Independence**

-A database should work properly regardless of its distribution across a network.

-it should be possible to run the DB on a variety of different communication networks.

-Application program will work even if the programs and data are moved on different site.

- **Rule 12: Nonsubversion Rule**

-It should not be able to subvert integrity rules to change the data. This can be achieved by some sort of encryption.

-In other words, the system should support heterogeneity


Thank you