DATABASE ANALYSIS AND DESIGN 5

Week 6

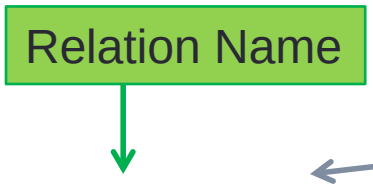
Relational Model

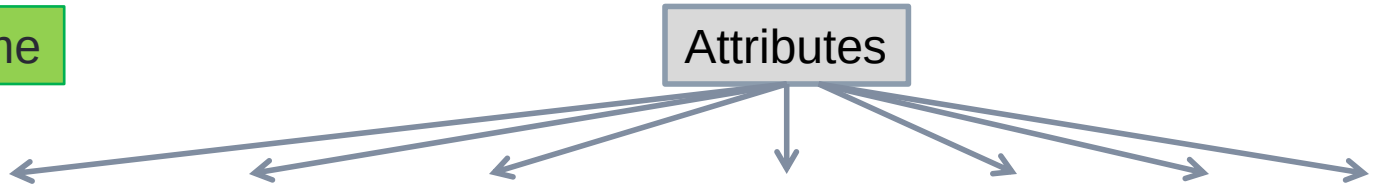
Outline

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations

Relational Model Concepts

- Relational model represents the database as a collection of relations.
- Informally, each relation resembles a table of values.
- Each row in the table represents a collection of relational data values.
  - Represents a fact that corresponds to a real world entity or relationship.
  - Table name and Column name are used to interpret the values in each row.
- All the values in a column are same data type.
- In *formal relational model terminology*, row is called a tuple, column header is called an attribute and a table is called a relation.
- The data type describing the type of values that appear in each column is represented by a domain of possible values.





Student	Name	AddmissionNo	HomePhone	Address	OfficePhone	Age	GPA
	Jones	53-666L	011 2685043	Sarana Road, Colombo 7	null	20	3.4
	William	53-688K	081 2225741	Dalada Veediya, Kandy	null	21	3.2
	Jane	54-650V	null	Pagoda Road, Nugegoda	011 2709834	26	3.8
	Jessica	54-320V	091 2233222	Rampart Street, Galle	011 2503421	28	3.9
	George	55-633V	033 2455042	Mangala Road, Gampaha	null	20	3.6

Domains

- A domain D is a set of atomic values.
- A domain is specified with a data type and a name for the domain
- E.g. domains
  - Set_of_phone_numbers: The set of valid phone numbers in Sri Lanka
  - Local_phone_numbers: The set of valid phone numbers within a particular area code in Sri Lanka.
  - Student_admission_numbers: The set of valid student admission numbers.
  - Names: The set of character strings that represent names of persons.
  - Grade_point_averages: Possible values of computed grade point averages; each must be real (floating point) number between 0 and 4.2
  - Employee_ages: Possible ages of employees in a company; each must be a value between 18 and 65 years old.

Domains

Logical definition of domain.

- Employee_ages: Possible ages of employees in a company; each must be a value between 18 and 65 years old
- Data type of following?
  - Employee_ages : Integer number between 18 and 65
  - Employee_names: Character strings
  - Student_admission_numbers: Character strings that represent valid admission numbers
- A domain also has a data-type or a **format** defined for it
  - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
  - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.
- Additional information
  - E.g. for numeric domain like Person_weights – should have unit of measurement, kilo grams
- dom(A_i) – Domain of A_i attribute

Relation

- Made up of 2 parts
  - **Schema** is used to describe a relation.
    - Denoted by $R(A_1, A_2, \ldots, A_n)$. It is made up of a relation name R and a list of attributes $A_1, A_2, \ldots A_n$.
    - Specifies name of relation, plus name and type of each column.
    - E.g. User(uid: string, name: string, login: string, age: integer).
  - **State (or Instance)** : a table, with rows and columns.
    - #fields/ attributes of a relation = degree / arity.
- Can think of a **relation** as a ***set of rows or tuples*** *(i.e., all rows are distinct).*

For Relation Schema Student…

- Student(Name, AdmissionNo, HomePhone, Address, OfficePhone, Age, GPA)
- Relation Degree?
- Using data type can also be written as
  - Student(Name: string, AdmissionNo: string, HomePhone: string, Address: string, OfficePhone: string, Age: integer, GPA: real)
- Domains (*From slide 5*)
  - dom(Name) = Names
  - dom(AdmissionNo) = Student_admission_numbers
  - dom(HomePhone) = Local_phone_numbers
  - dom(GPA) = Grade_point_averages

For Relation Schema Student… (Cont'd)

- Several attributes can be from the same domain

  e.g. HomePhone, OfficePhone in the same domain
  - They indicate different roles or interpretations for the domain Local_phone_numbers

- It is possible to refer to attributes of a relation **by its position** within the relation
  - Second attribute of the Student relation is AdmissionNo
  - Fourth attribute of the Student relation is Address

Relation State

- Also known as **relation instance**
- It is denoted by *r(R)*, where R is a relation
- Relation state is a set of *n*-tuples, r= $\{t_1, t_2, \ldots, t_m\}$
- Relation state, *r(R)* is a subset of ($dom(A_1)$ x $dom(A_2)$ x…x $dom(A_n)$)

- **Current relation state** – a relation state at a given time
  - Reflects only the valid tuples that represent a particular state of the real world
  - Relation state changes as the state of the real world changes

- However, schema is relatively static.
  - E.g. adding a new attribute to represent new information that was not originally stored in the relation

Tuple

- A **tuple** is an ordered set of values
  - Enclosed in angled brackets ‘< … >’
- Each value is derived from an appropriate *domain*.
- A row in the Customer relation is a 4-tuple and would consist of four values, for example:
  - <632895, "John Smith", "101 Main St., GA 30332", "(404) 894-2000">
  - This is called a 4-tuple as it has 4 values
  - A tuple in the Customer relation.
- A relation is a **set** of such tuples.

Summary

Informal Terms	Formal Terms
Table	
Column Header	
All possible Column Values	
Row	
Table Definition	
Populated Table	

Characteristics of Relations

What makes a relation different from a file or a table?

1. **Ordering of Tuples in a Relation**
   - Relation is a set of tuples
   - Mathematically, elements of a set have no order among them
   - So, tuples in a Relation do not have any particular order (Tuple ordering is not part of a relation definition)
   - However, when we display a relation as a table, then the rows are displayed in a certain order
   - Many logical orders can be specified on a relation – e.g. by Name, Age, AdmissionNo
   - When implemented as a file or displayed as table, a particular order may be specified on the records of the file or rows of the table

A

Student	Name	AddmissionNo	HomePhone	Age	GPA
	Jones	53-666-234 L	011 2685043	20	3.4
	William	53-688-231 K	081 2225741	21	3.2
	Jane	54-650-211V	null	26	3.8
	Jessica	54-320-441V	091 2233222	28	3.9
	George	55-633-238V	033 2455042	20	3.6

A, B are Identical Relations

B

Student	Name	AddmissionNo	HomePhone	Age	GPA
	Jane	54-650-211V	null	26	3.8
	Jessica	54-320-441V	091 2233222	28	3.9
	Jones	53-666-234 L	011 2685043	20	3.4
	George	55-633-238V	033 2455042	20	3.6
	William	53-688-231 K	081 2225741	21	3.2

Characteristics of Relations (Cont'd)

2. **Ordering of Values within a Tuple, and an Alternative Definition of a Relation**
   - First Definition:
     - An *n*-tuple is an *ordered list* of *n* values. $t = <v_1, v_2, v_3 \ldots v_n>$ where each value v_i, is an element of $dom(A_i)$ or a special **null** value. $1 \leq i \leq n$
     - i^{th} value in tuple *t* corresponds to attribute A_i, is referred to as $t[A_i]$ (or t[i] if we use the positional notation).
     - Ordering of values in a tuple, hence ordering of a relational schema is important.
   - However, at a logical level -> ordering of attributes and their values is not important as long as the correspondence between attributes and values is maintained.
   - Alternative Definition: Ordering of values in a tuple is unnecessary

Characteristics of Relations (Cont'd)

2. **Ordering of Values within a Tuple, and an Alternative Definition of a Relation**
   - Alternative Definition: Ordering of values in a tuple is unnecessary
     - Each tuple t_i is a mapping from R to D.
     - Where R is the relational schema is a set of attributes, $R(A_1, A_2, \ldots, A_n)$ and D is the union of attribute domains. $D = dom(A_1) \cup dom(A_2) \cup \ldots \cup dom(A_n)$
     - Therefore, a tuple can be considered as a set of (<attrbiute>, <value>) pairs
     - E.g.

t = <(Name, Jane), (AdmissionNo, 54-650-211V), (HomePhone, null), (age, 26), (GPA, 3.8)>

≡

t = < (AdmissionNo, 54-650-211V), (Name, Jane), (age, 26), (HomePhone, null), (GPA, 3.8)>

Ordering is not important as attribute name appears with its value.

- When the relation is implemented as a file, first definition is used.

Characteristics of Relations (Cont'd)

3. Values and Nulls in the Tuples

- Each value in a tuple is an **atomic** value
- Hence, composite and multivalued attributes are not allowed.

- **Null** is used to represent values of attributes that may be unknown or may not apply to a tuple.
  - Value unknown, value exists but is not available, attribute doe not apply to the tuple.

Characteristics of Relations (Cont'd)

4. Interpretation (Meaning) of a Relation

- E.g. from Slide 4
  - Schema of the student relation asserts, that in general, a student entity has a Name, AdmissionNo, HomePhone, Address, OfficePhonen, Age, GPA

- Then each tuple in the relation can be interpreted as a fact or a particular instance of the assertion (or declaration).
  - First tuple asserts that there is a student whose name is Jones, AdmissionNo is 53-666-234 L, Age is 20 and so on….

- Includes facts about entities and relationships

Relational Model Constraints

Three main categories

- **Inherent model based** constraints – Constraints that are inherent in the data model
  - Characteristics of Relations (Slide 13 to 18)
  - E.g. Relation cannot have duplicate tuples

- **Schema based** constraints – Constraints that can be directly expressed in the schemas of the data model
  - Include domain constraints, key constraints, constraints on nulls, entity integrity constraints and referential integrity constraints

- **Application based** constraints – Constraints that cannot be expressed in the schemas of the data model. Therefore, they are expressed and enforced by the application programs.

Domain Constraints

- Within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).

- Data types associated with a domain
  - Standard numeric data types – for integers (short integer, integer, long integer) and for real (float and double precision float)
  - Characters
  - Boolean
  - Fixed length string and variable length strings
  - Date, time
  - Some cases money type

Key Constraints and Constraints on Null Values

- A relation is set of tuples
- All elements of a set are distinct
- Therefore, all tuples in a relation should be distinct

- **Superkey** (SK) is a subset of attributes in a relation where $t_1[SK] \neq t_2[SK]$
  - It specifies a uniqueness constraint that no two distinct tuples in any state r of R can have the same value for SK
  - Every relation has at least one default superkey. That is all its attributes.
  - A SK can have redundant attributes.

Key Constraints and Constraints on Null Values (Cont'd)

- **Key** (K) – is a superkey of a relation R with the additional property that removing any attribute A for K leaves a set of attributes K` that is not a superkey any more.

- A key satisfies two constraints:
  1. Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key.
  2. It is a minimal superkey. It is a superkey from which we cannot remove any attributes and still have the uniqueness constraint in condition 1 hold.

- E. g. Attribute set {AdmissionNo} is a key

  Attribute set {AdmissionNo, Name, Age} is a super key but not a key.

Key Constraints and Constraints on Null Values (Cont'd)

- **Candidate key** – If there are more than 1 key, each of keys is called a candidate key.
  - E.g. for a Person relation schema: Key1 ={NIC}, Key2= {PassportNo}
  - For a Car relation schema: Key1 = {State, RegNo} , Key2 = {SerialNo}

- **Primary Key** (PK) – Select one of the candidate keys as PK
  - Attributes that form the primary key are underlined.
  - E.g CAR(State, RegNo, SerialNo, Make, Model, Year)
  - Select a key with a single attribute or small number of attributes

- **NOT NULL** constraint – specifies whether null values are allowed or not on an attribute.
  - E.g. Name of the Student Relation is constrained to be NOT NULL.

Relational databases and relational database schemas

- **Relational database schema** (S), is a set of relational schemas, S= {R_1, R_2, … , R_m} and set of integrity constraints (IC).
  - E.g. Company = {Employee, Department, Dept_Location, Project, Works_On, Dependent}

- **Relational database State** (or instance) (DB) is a set of relation states, DB = {r_1, r_2, … , r_m} where each r_i is a state of R_i and r_i satisfies the integrity constraints in IC

- **Valid State** – a database state that satisfies all the constraints in IC
- **Invalid State** – a state that does not obey all the constraints in IC

Company Relational Database Schema

Employee

FName	Initial	LName	ENO	DOB	Address	Sex	Salary	SuperENO	DNO

Department

DName	DNumber	MgrNO	MgrStartingDate

Department_Location

DNumber	DLocation

Project

PName	PNumber	PLocation	DNum

Works_On

EENO	PNO	Hours

Dependent

EENO	Dependent_Name	Sex	BDate	Relationship

Represent the same real world concept. Those attributes may or not have the same name.

Different concepts may have same name in different relations. E.g. Name instead of PName in Project.

Database State (Database Instance)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Integrity Constraint (ICs)

- Condition that must be true for *any instance* of the database; e.g., *domain constraints*.
  - ICs are specified when schema is defined
  - ICs are checked when relations are modified.

- A *legal instance of a relation is one that satisfies* all specified ICs
  - DBMS should not allow illegal instances

- If the DBMS checks ICs, stored data is more faithful to real-world meaning
  - Avoids data entry errors, too!

Entity Integrity Constraint

- 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.
  - t[PK] ≠ null for any tuple t in r(R)
  - 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.

Primary Key Constraints

- A set of fields is a *key for a relation if :*
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.

  - Part 2 false? A *superkey*
  - If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the primary key.

- E.g., *sid is a key for Students. (What about name?) The set {sid, name} is a superkey.*

- Possibly many *candidate keys (specified using* UNIQUE), one of which is chosen as the *primary key.*

Referential Integrity Constraint

- 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**.
- Tuples in the **referencing relation** R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R_2.
  - A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.

Referential Integrity Constraint

- *Foreign key : Set of fields in one relation that is used* to `refer' to a tuple in another relation. (Must correspond to primary key of the second relation.)

- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R_1.FK to R_2.

- E.g. *sid is a foreign key referring to Student:*
  - Enrolled(*sid: string, cid: string, grade: string)*
  - If all foreign key constraints are enforced, referential integrity is achieved.

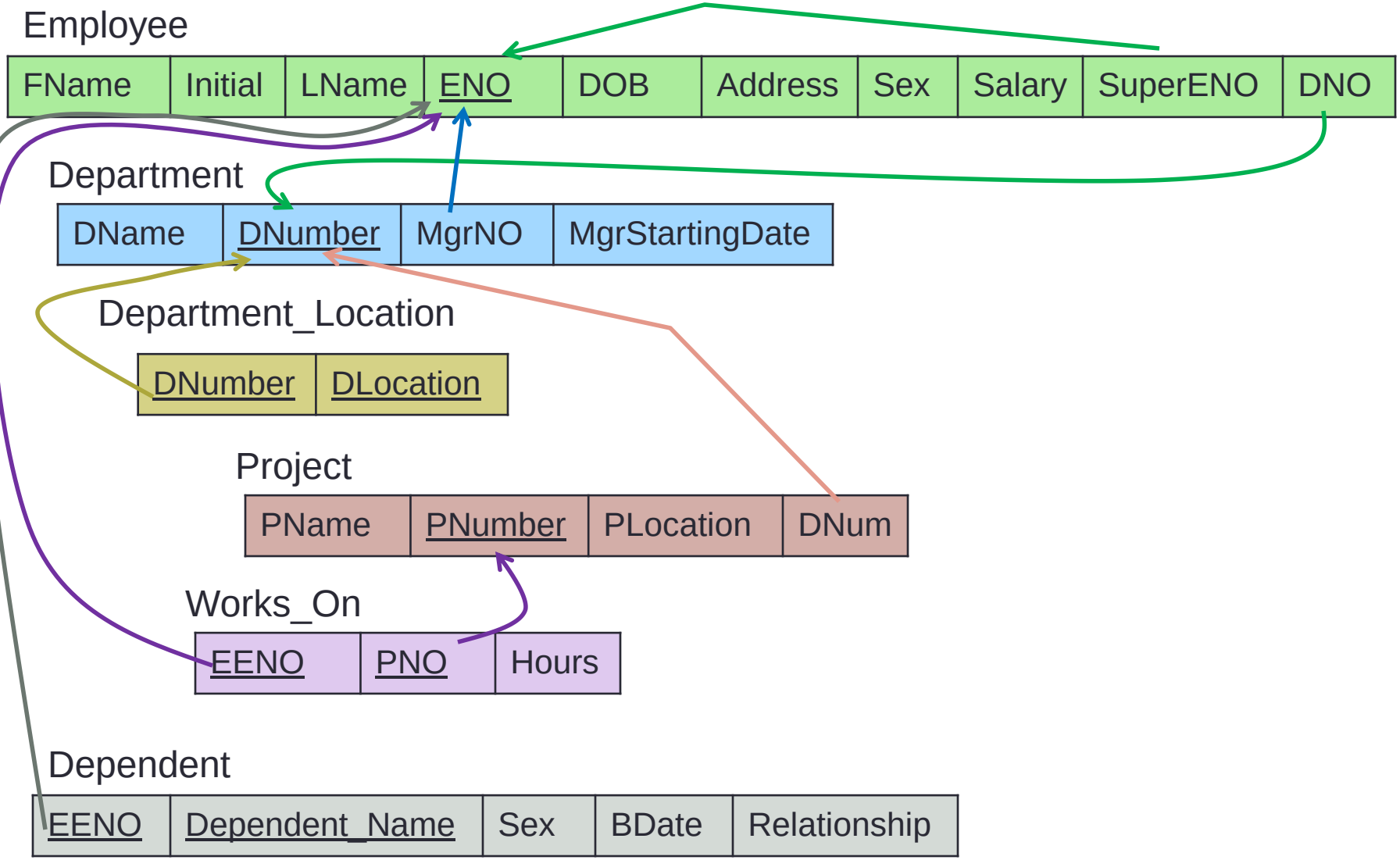
Referential Integrity Constraint

- A set of attributes FK in relation schema R_1 is a foreign key of R_1 that references relation R_2 if it satisfies the following two rules:
  - The attributes in FK have the same domain as the primary key attributes PK of R_2
  - A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ or is null
- In case (2), the FK in R_1 should **not** be a part of its own primary key
- A foreign key can refer to its own relation

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 to the primary key of the referenced relation for clarity

Company Schema



Enforcing Referential Integrity

- Consider Students and Enrolled; *sid in Enrolled is a* foreign key that references Students.

- What should be done if an Enrolled tuple with a *non-existent student id is inserted*? (*Reject it!)*

- What should be done *if a Students tuple is deleted*?
  - Also delete all Enrolled tuples that refer to it.
  - Or Disallow deletion of a Students tuple that is referred to.
  - Or Set sid in Enrolled tuples that refer to it to a *default sid.*

  (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value null, denoting `unknown' or `inapplicable'.)

- Similar if primary key of Students tuple is updated.

Other type of Constraints

- Semantic Integrity Constraints:
  - Based on application semantics and cannot be expressed by the model per se
  - E.g. “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”

- A **constraint specification** language may have to be used to express these

Update Operations on Relations

- **Operations on a relational model**: updates and retrievals

- Three types of update operations
  - INSERT a tuple(s).
  - DELETE a tuple(s).
  - MODIFY a tuple(s).

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

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:
    - if the primary key value is null in the new tuple

Possible violations for each operation

- In case of integrity violation in inserts, 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)
  - Execute a user-specified error-correction routine

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 deletion by deleting the tuples that reference the tuple that is being deleted
      - SET NULL option: set the foreign keys of the referencing tuples to NULL or change to reference another valid tuple.
        - Could violate entity integrity if foreign key is part of the primary key of the referencing relation
  - One of the above options must be specified during database design for each foreign key constraint

Possible violations for each operation

- **MODIFY/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

Task 1

- Consider the following relations for a database that keeps track of business trips of sales person in a sales office

  - SalesPerson(Sid, Name, Start_year, DeptNo)
  - Trip(Sid, From_city, To_city, Departure_date, Return_date, TripID)
  - Expense(Trip_Id, AccountNo, Amount)

- Specify Foreign Keys

Task 2

- Consider the following 6 relations of for an order processing database application in a company
  - Customer (CustNo, Cname, City)
  - Order(OrderNo, CustNo, Order_Amt)
  - Order_item(OrderNo, ItemNo, Quantity)
  - Item(ItemNo, Unit_price)
  - Shipment(OrderNo, WarehouseNo, Ship_date)
  - Warehouse(WarehouseNo, City)

Assume that an order can be shipped from several warehouses.

Specify the Foreign Keys for the schema

Task 3

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)

COURSE(CourseNo, Cname, Dept)

ENROLL(SSN, CourseNo, Quarter, Grade)

BOOK_ADOPTION(CourseNo, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema.