Lecture 3

The Relational Data Model and Relational Database Constraints

Agenda

- Relational Model Concepts
- Formal Definitions
- Characteristics of Relations
- Relational Model Constraints
- Schema-based Constraints
- Dealing with Constraint Violations

Relational Model Concepts

Relational Model Concepts

- The relational Model of Data is based on the concept of a ***Relation.***
- A ***Relation*** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - Rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - The column header is called an **attribute name** (or just **attribute**)

Relational Model Concepts

Relation Name

Attributes

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

Tuples

Figure 5.1
The attributes and tuples of a relation STUDENT.

Formal Definitions

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
  - Denoted by **R(A1, A2, ...An)**
  - R is the **name** of the relation
  - The **attributes** of the relation are A1, A2, ..., An
- Example:

  CUSTOMER (Cust-id, Cust-name, Address, Phone#)
  - CUSTOMER is the relation name
  - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
  - For example, the domain of Cust-id is 6 digit numbers.

Formal Definitions - Tuple

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

Formal Definitions - Domain

- A **domain** has a logical definition:
  - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- 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.
- The attribute name designates the role played by a domain in a relation:
  - Used to interpret the meaning of the data elements corresponding to that attribute
  - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

Formal Definitions - State

- **A Relation state** *r* of the relation schema $R(A1, A2, \ldots, An)$, is a set of *n*-tuples $r = \{t1, t2, \ldots, tm\}$.
- Each ***n*-tuple** *t* is an ordered list of *n* values $t = <v1, v2, \ldots, vn>$, where each value *vi*, $1 \leq i \leq n$, is an element of dom (Ai) or is a special NULL value.
- *n* is the number of values within each tuple while *m* is the total number of tuples (or rows).

Formal Definitions - Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Characteristics of Relations

Characteristics of Relations

- **Ordering of Tuples in a Relation.** A relation is defined as a *set* of tuples. Mathematically, elements of a set have *no order* among them; hence, tuples in a relation do not have any particular order. In other words, a relation is not sensitive to the ordering of tuples.

- **Ordering of Values within a Tuple.** According to the definition of a relation, an *n*-tuple is an *ordered list* of *n* values, so the ordering of values in a tuple is important.

Characteristics of Relations

- **Values in the Tuples.** Each value in a tuple is an **atomic** value; that is, it is not divisible into components within the framework of the basic relational model. Hence, composite and multivalued attributes are not allowed.

Relational Model Constraints

Relational Model Constraints

- Constraints on databases can generally be divided into three main categories:
  1. Constraints that are inherent in the data model. We call these **inherent model-based constraints** or **implicit constraints**.
  2. Constraints that can be directly expressed in the schemas of the data model. We call these **schema-based constraints** or **explicit constraints**.
  3. Constraints that *cannot* be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs or in some other way. We call these **application-based** or **semantic constraints** or **business rules**.

Relational Model Constraints

1. **Inherent model-based constraints:**
   - The **characteristics of relations** that we discussed are the inherent constraints of the relational model and belong to the first category.
   - For example, the constraint that a relation cannot have duplicate tuples is an inherent constraint (this is because a relation is a *set* of tuples).
2. **Schema-based constraints:**
   - The schema-based constraints include **Domain constraints**, **Key constraints**, **constraints on NULLs**, **Entity Integrity constraints**, and **Referential Integrity constraints**.

Relational Model Constraints

3. **Semantic constraints:**
    - Constraints in the third category are more general, relate to the **meaning** as well as **behavior** of attributes, and are difficult to express and enforce within the data model, so they are usually checked within the application programs that perform database updates.

Lets now discuss the **Schema-based constraints**

Schema-based Constraints

Schema-based Constraints - Domain Constraints

- Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).
- The data types associated with domains typically include standard numeric data types for integers (such as short integer, integer, and long integer) and real numbers (float and double-precision float).
- Characters, Booleans, fixed-length strings, and variable-length strings are also available, as are date, time, timestamp, and other special data types.
- Domains can also be described by a subrange of values from a data type or as an enumerated data type in which all possible values are explicitly listed.

Schema-based Constraints - Key Constraints

- **Superkey** of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
    - This condition must hold in *any valid state* r(R)
- **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 (does not possess the superkey uniqueness property)

Schema-based Constraints - Key Constraints

- Hence, a key is a superkey but not vice versa. A superkey may be a key (if it is minimal) or may not be a key (if it is not minimal).
- Consider the STUDENT relation of Figure 5.1. The attribute set {Ssn} is a key of STUDENT because no two student tuples can have the same value for Ssn.
- Any set of attributes that includes Ssn—for example, {Ssn, Name, Age}—is a superkey. However, the superkey {Ssn, Name, Age} is not a key of STUDENT because removing Name or Age or both from the set still leaves us with a superkey.

Schema-based Constraints - Key Constraints

- In general:
  - Any *key* is a *superkey* (but not vice versa)
  - Any set of attributes that *includes a key* is a *superkey*
  - A *minimal* superkey is also a key
- Hence, a key satisfies two properties:

  1. Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key. This *uniqueness* property also applies to a superkey.

  2. It is a *minimal superkey*—that is, a superkey from which we cannot remove any attributes and still have the uniqueness constraint hold. This *minimality* property is required for a key but is optional for a superkey.

Schema-based Constraints - Key Constraints

- We call such key a **Primary Key**.

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the ***primary key***.
  - The primary key attributes are <u>underlined</u>.

- The ***primary key*** value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity

Schema-based Constraints - Key Constraints

- Example: Consider the CAR relation:
  - It has two candidate keys: License_number and Engine_serial_number.
  - We choose License_number (i.e. SerialNo) as the **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.

Schema-based Constraints - NULLs

- Another constraint on attributes specifies whether NULL values are or are not permitted.

- For example, if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.

Schema-based Constraints - Entity Integrity

- A **relational database schema** S is a set of relation schemas $S = \{R1, R2, \ldots, Rm\}$ and a set of **integrity constraints** IC.
- A **relational database state** must satisfy the integrity constraints specified in IC.
- Figure 5.5 shows a relational database schema that we call COMPANY = {EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON, DEPENDENT}.
- In each relation schema, the underlined attribute represents the primary key.

Schema-based Constraints - Entity Integrity

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Schema-based Constraints - Entity Integrity

- **Entity Integrity:**
  - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple.
    - This is because primary key values are used to *identify* the individual tuples.
    - t[PK] $\neq$ null for any tuple t
    - 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.

Schema-based Constraints - Referential Integrity

- 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**.

Schema-based Constraints - Referential Integrity

- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].

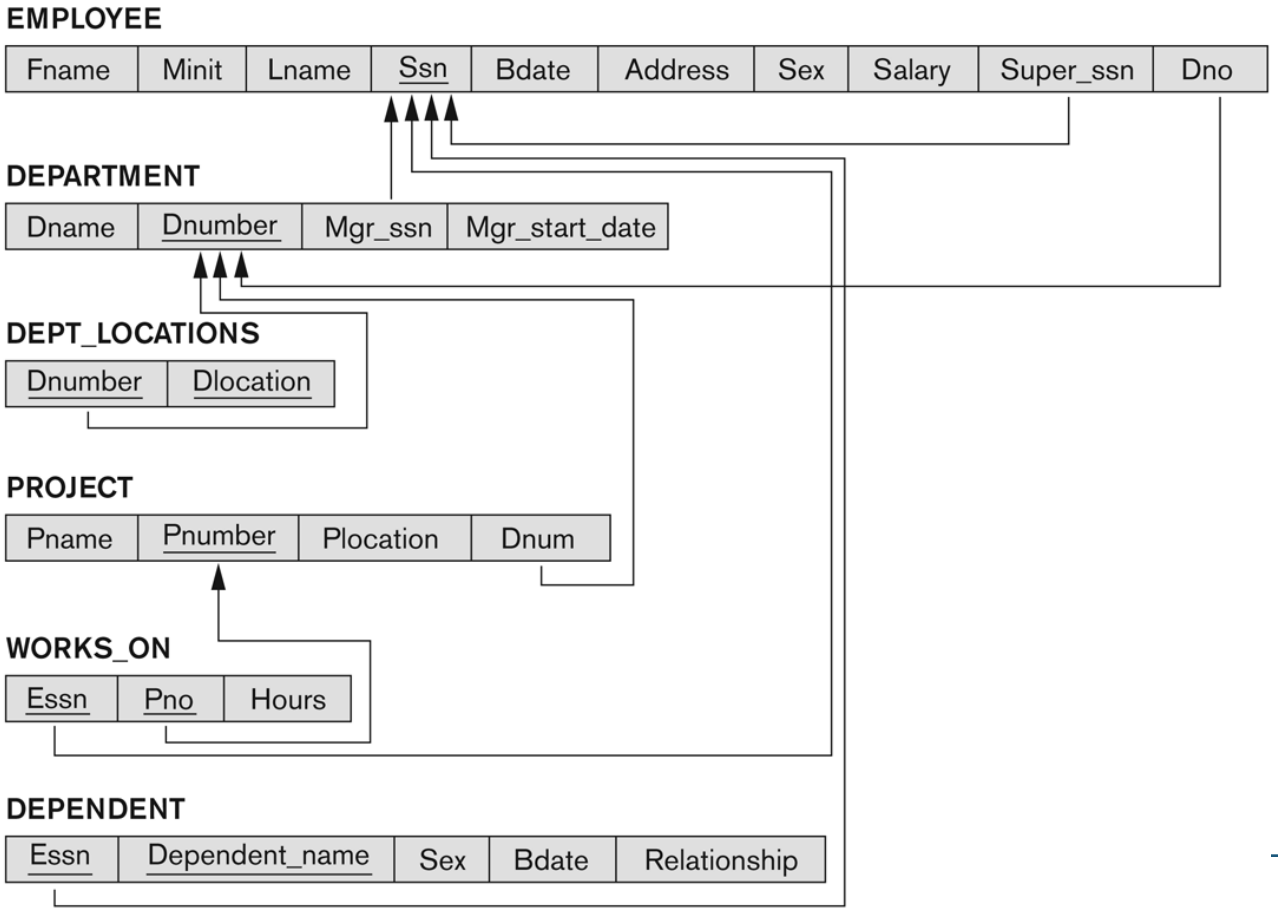
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Schema-based Constraints - Referential Integrity

- **Referential Integrity:**
  - The value in the *foreign key* column (or columns) FK of the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

Schema-based Constraints - Referential Integrity

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Dealing with Constraint Violations

Constraint Violations

- Each *relation* will have many tuples in its current relation state
- 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
  - UPDATE an attribute of an existing tuple
- Next slide shows an example state for the COMPANY database

Figure 5.6
One possible database state for the COMPANY relational database schema.

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

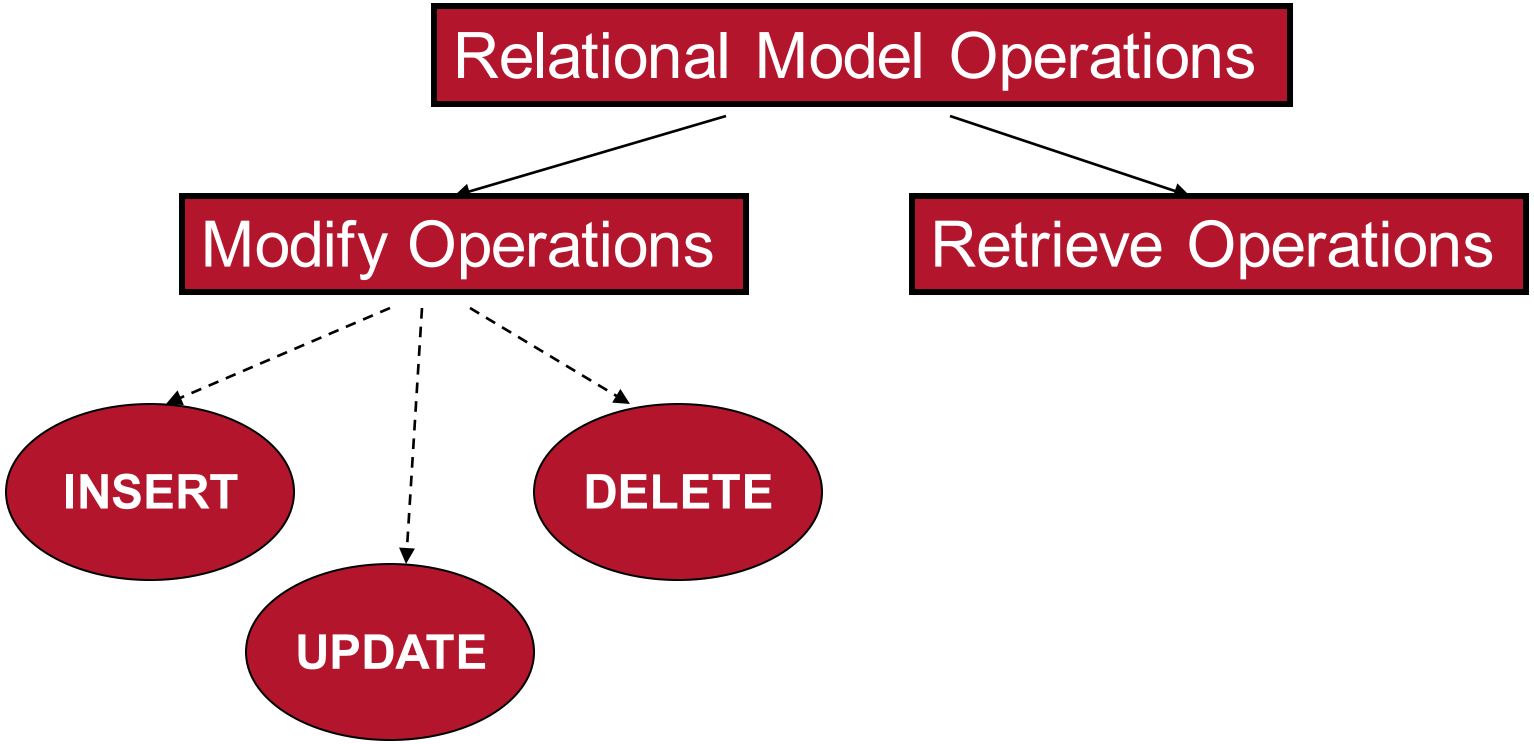
PROJECT

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

Constraint Violations



Whenever modify operations are applied, the ICs should not be violated

Constraint Violations

- 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)
  - Execute a user-specified error-correction routine

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

- 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 resolved by several actions: RESTRICT, CASCADE, SET NULL
      - **RESTRICT** option: reject the deletion
      - **CASCADE** option: propagate the deletion by deleting tuples that reference the tuple that is being deleted.
      - **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

Possible violations for each operation

- UPDATE is similar to a DELETE operation followed by an INSERT operation.
- Therefore, UPDATE can violate any constraints violated by INSERT or DELETE in the previous slides.
- Constraints may be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - May violate domain, key, referential integrity, entity integrity constraints
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - May violate domain constraints

Thank You