RELATIONAL DATA MODEL
Objectives

- Terminology +
- Characteristics of Relations +
- Relational Data Model Notations +
- Key constraints +
- Other Constraints +
- Relational Database Schema and State +
- Relational Data Model Operations +
- Terminology

- A **relation** is a table (logical) with columns and rows.

- An **attribute** is a named column of a relation.

- A **domain** is a set of allowable values for one or more attributes.

- A **tuple** is a row of a relation.

- **Degree** is a number of attributes in a relation.

- **Cardinality** is a number of tuples in a relation.

- **Relational Database** is a collection of relations.
- Terminology

Relation name

STUDENT

Attributes

Tuples

Cardinality

Degree

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>NAT_ID</th>
<th>DOB</th>
<th>DEPTCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>201111</td>
<td>KHALED</td>
<td>1111111111</td>
<td>16-05-1984</td>
<td>ICS</td>
</tr>
<tr>
<td>202222</td>
<td>ADEL</td>
<td>2222222222</td>
<td>23-09-1984</td>
<td>COE</td>
</tr>
<tr>
<td>203333</td>
<td>AHMED</td>
<td>3333333333</td>
<td>11-08-1984</td>
<td>COE</td>
</tr>
<tr>
<td>211111</td>
<td>HASSAN</td>
<td>4444444444</td>
<td>23-10-1985</td>
<td>SWE</td>
</tr>
<tr>
<td>204444</td>
<td>MUSTAFA</td>
<td>5555555555</td>
<td>16-11-1984</td>
<td>ICS</td>
</tr>
</tbody>
</table>
- Characteristics of Relations

- Each relation in the same relational database schema has a distinct name
- Each value in a tuple is atomic
- Each attribute in a relation has a distinct name.
- Values of an attribute are all from the same domain.
- Each tuple is distinct.
- Order of attributes has no significance.
- Order of tuples has no significance, theoretically.
- Relational Data Model Notations

- The letters Q, R, S denote the abstract relation names.

- \( R(A_1, A_2, A_3, \ldots, A_n) \) denotes a relation schema \( R \) of degree \( n \).
  - Example: \( \text{STUDENT}(\text{Name, Id, Phone, Address, Mobile, DOB}) \)

- Both \( t[A_i] \) and \( t.A_i \) refers to the value \( v_i \) in \( t \) for attribute \( A_i \).
  - Example: in second tuple, both \( t[\text{name}] \) and \( t.\text{name} \) refers to “Adel”
- Key Constraints

- Superkey
- Candidate Key
- Primary Key
- Alternate Key
- Foreign Key
-- Superkey

An Attribute or a set of attributes that uniquely identify a tuple within a relation.

Example:
- STU_ID, NAME, NAT_ID, DOB, DEPTCODE
- STU_ID, NAME, NAT_ID
- STU_ID
- NAT_ID

STUDENT

<table>
<thead>
<tr>
<th>STU_ID</th>
<th>NAME</th>
<th>NAT_ID</th>
<th>DOB</th>
<th>DEPTCODE</th>
</tr>
</thead>
</table>

-- Candidate Key

A superkey (K) such that no proper subset is a superkey within the relation.

- In each tuple of R, the values of K uniquely identify that tuple (uniqueness).
- No proper subset of K has the uniqueness property (irreducibility).

Example:

- STU_ID
- NAT_ID

### STUDENT

<table>
<thead>
<tr>
<th>STU_ID</th>
<th>NAME</th>
<th>NAT_ID</th>
<th>DOB</th>
<th>DEPTCODE</th>
</tr>
</thead>
</table>

-- Primary Key

- Is a candidate key selected to identify tuples uniquely within a relation.

- Example:
  - STU_ID

**STUDENT**

<table>
<thead>
<tr>
<th>STU_ID</th>
<th>NAME</th>
<th>NAT_ID</th>
<th>DOB</th>
<th>DEPTCODE</th>
</tr>
</thead>
</table>

March 23, 2008
-- Alternate Key

- candidate keys that are not selected to be the primary key.

- Example:
  - NAT_ID

STUDENT

<table>
<thead>
<tr>
<th>STU_ID</th>
<th>NAME</th>
<th>NAT_ID</th>
<th>DOB</th>
<th>DEPTCODE</th>
</tr>
</thead>
</table>

March 23, 2008
-- Foreign Key

An attribute or a set of attributes within one relation that matches the candidate key of some (possibly the same) relation.

**DEPARTMENT**

<table>
<thead>
<tr>
<th>DEPTCODE</th>
<th>DEPTNAME</th>
</tr>
</thead>
</table>

**STUDENT**

<table>
<thead>
<tr>
<th>STU_ID</th>
<th>NAME</th>
<th>NAT_ID</th>
<th>DOB</th>
<th>DEPTCODE</th>
</tr>
</thead>
</table>

Foreign key
- Other Constraints

- **Null**: Represents a value of an attribute that is currently unknown or is no applicable for this tuple.

- **Entity integrity Constraint**: In a base relation, no attribute of a primary key can be *null*.

- **Referential Integrity Constraint**: If a foreign key exists in a relation, either the foreign key value *must match a candidate key* value of some tuple in its home relation or the foreign key value must be *wholly null*.

- **Domain Constraint**: Specifies that the value of attribute \( A \) must be an atomic value from the domain \( \text{DOM}(A) \).
A relational Database Schema is a non-empty set of relations schemas \{R_1, R_2, R_3, \ldots, R_n\} and a set of integrity constraints that include domain, null, keys, entity, and referential.

A relational database state \( S \) is a set of relation states \( \{r_1, r_2, r_3, \ldots, r_n\} \) such that \( r_i \) is a relation state of \( R_i \) and it satisfies the constraints specified for \( R_i \).
There are two categories of relational data model operations:

- **Retrieval operations** extract information from the relational database.
- **Update operations** causes the relation (and the relational database) state changes. They include:
  - Insert
  - Delete
  - Update
-- Update Operations

- **Insert Operation +**
  - Possible violations Caused by Insert Operation +
  - Handling Violations Caused by Insert Operation +

- **Delete Operation +**
  - Possible Violations Caused by Delete Operation +
  - Handling Violations Caused by Delete Operation +

- **Update Operation +**
  - Possible Violations Caused by Update Operation +
  - Handling Violations Caused by Update Operation +
--- Insert Operation

- Insert operation inserts a list of attribute values for a new tuple \( t \) into relation \( R \).
- It is denoted by:
  
  \[
  \text{INSERT} \langle \text{list of attribute values} \rangle \ \text{INTO} \ \text{R};
  \]

- Example:
  
  \[
  \text{INSERT} \langle \text{SWE}, \text{'Software Engineering'} \rangle \ \text{INTO} \ \text{DEPARTMENT};
  \]
--- INSERT <‘SWE’, ‘Software Eng.’> INTO DEPARTMENT

Before INSERT operation

<table>
<thead>
<tr>
<th>DEPTCODE</th>
<th>DEPTNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>COE</td>
<td>Comp. Eng</td>
</tr>
<tr>
<td>ICS</td>
<td>Info. Comp. Sci.</td>
</tr>
</tbody>
</table>

DEPARTMENT

After INSERT operation

<table>
<thead>
<tr>
<th>DEPTCODE</th>
<th>DEPTNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>COE</td>
<td>Comp. Eng</td>
</tr>
<tr>
<td>ICS</td>
<td>Info. Comp. Sci.</td>
</tr>
<tr>
<td>SWE</td>
<td>Software Eng.</td>
</tr>
</tbody>
</table>

DEPARTMENT
--- Possible INSERT Operation Violations

- **Domain constraint violation** can occur when a given value of an attribute does not belong to the specified domain of that attribute.
  - Example: `INSERT <1234, 'Software Eng.'> INTO DEPARTMENT`

- **Key constraint violation** can occur when a key value in the tuple to be inserted already exists in the relation.
  - Example: `INSERT <'COE', 'Software Eng.'> INTO DEPARTMENT`

- **Entity Integrity constraint violation** can occur if the primary key value of the new tuple is null.
  - Example: `INSERT <null, 'Software Eng.'> INTO DEPARTMENT`

- **Referential Integrity constraint violation** if the foreign key value of the new tuple reference to a value that does’t exist as a valid value in the corresponding attribute in the referenced relation.
If an INSERT operation violates one or more constraints, there are two options to handle that:

- Reject the operation (default)
- Prompt the user to correct the values that cause the violation.
--- Delete Operation

- Deletes one or many tuples in a relation.
  - It is denoted as: `DELETE FROM R WHERE <where clause>`

- Example
  - `DELETE FROM department WHERE deptcode = 'SWE';`
DELETE FROM department WHERE deptcode = 'SWE'

Before DELETE operation

<table>
<thead>
<tr>
<th>DEPTCODE</th>
<th>DEPTNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>COE</td>
<td>Comp. Eng</td>
</tr>
<tr>
<td>ICS</td>
<td>Info. Comp. Sci.</td>
</tr>
<tr>
<td>SWE</td>
<td>Software Eng.</td>
</tr>
</tbody>
</table>

After DELETE operation

<table>
<thead>
<tr>
<th>DEPTCODE</th>
<th>DEPTNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>COE</td>
<td>Comp. Eng.</td>
</tr>
<tr>
<td>ICS</td>
<td>Info. Comp. Sci.</td>
</tr>
</tbody>
</table>
--- Possible Delete Operation Violations

Referential integrity constraint violation can occur if the tuple being deleted by the foreign keys from other tuples in the database.

Example: `DELETE FROM lecturer WHERE LID = 111;`

<table>
<thead>
<tr>
<th>LID</th>
<th>Lname</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Salah</td>
</tr>
<tr>
<td>222</td>
<td>Ejaz</td>
</tr>
<tr>
<td>333</td>
<td>Yahya</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCODE</th>
<th>SNAME</th>
<th>LID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICS334</td>
<td>Databases</td>
<td>333</td>
</tr>
<tr>
<td>ICS431</td>
<td>Operating Systems</td>
<td>111</td>
</tr>
<tr>
<td>ICS490</td>
<td>Data warehouse</td>
<td>222</td>
</tr>
<tr>
<td>ICS202</td>
<td>Data Structures</td>
<td></td>
</tr>
</tbody>
</table>
--- Handling Delete Operation Violations

There are three options to handle violations of a DELETE operation:

- Reject the DELETE operation.
- Cascade the DELETE operation by deleting tuples that refers to the deleted tuple.
- Replace the referencing attribute value by null, provided it is not a primary key or other valid exist-tuple value.
--- Update Operation

- Changes the value of one or more attributes in one or more tuples of some relation R.
- Its is denoted by: UPDATE R SET <expression>  WHERE <where clause>;

Example: UPDATE lecturer set lname = 'Adam' WHERE LID = 111;

Before UPDATE

<table>
<thead>
<tr>
<th>LID</th>
<th>Lname</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Salah</td>
</tr>
<tr>
<td>222</td>
<td>Ejaz</td>
</tr>
<tr>
<td>333</td>
<td>Yahya</td>
</tr>
</tbody>
</table>

After UPDATE

<table>
<thead>
<tr>
<th>LID</th>
<th>Lname</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Adam</td>
</tr>
<tr>
<td>222</td>
<td>Ejaz</td>
</tr>
<tr>
<td>333</td>
<td>Yahya</td>
</tr>
</tbody>
</table>
--- Possible Update Operation Violations

- In UPDATE operation two possible conditions that cause violations.
  - **Modifying non-key attribute** can cause:
    - Domain constraint violations
    - **Example**: Updating `sex` to ‘B’ instead to ‘M’ or ‘F’.
      
      ```sql
      UPDATE employee SET sex = 'B' WHERE EMP_ID = 11111
      ```
  
  - **Modifying key attribute** can cause:
    - Key constraint violation
    - Entity Integrity Constraint violation
    - Referential Integrity Constraint violation
UPDATE lecturer SET lid = 111 WHERE LID = 222

Violates primary key constraint because the new value 111 already exists in the lecturer table. It also violates referential integrity constraint because there are foreign keys which refer to 111.
--- Handling Update Operation Violations

There are three options to handle violations of a UPDATE operation:

- Reject the UPDATE operation.
- Prompt the user to correct the values that cause the violation.
- Replace the referencing attribute value by null, provided it is not a primary key or other valid exist-tuple value.