# Relations and Operations


pm jat @ daiict


# Relations

- Database is Collection of Relations
- Every Relation has Schema: “Structure” and “Constraints”
  - Note term “Integrity Constraint” is often used for Constraints.
- Constraints
  - Every tuple is Unique: implemented through **Keys**
    - Keys are: **Super Key, Candidate Key, Key**
    - Primary Key does not belong here
  - Referential Integrity
  - Not Null


# Relations - Keys

- Super Key: is a set of attribute SK for which no two tuples can have same value (put together)
- Suppose $t1 \in R$ and $t2 \in R$ then $t1[SK] \neq t2[SK]$
- Candidate Key, or Key: Minimal Super Key!
- Also note:
  - Key is often called "Composite" when it contains more than one attribute. For example relation
    Offers(CouseNo, TermID, InstructorID) has key {CourseNo, TermID}
  - A relation can have multiple Keys.
    For instance R(UUID, ENO, Name, DOB, Address, Phone) has two keys: UUID and ENO

# Referential Integrity Constraints

- Relationship between relations are implemented through **Foreign Keys**.
- Consider a relation **R** has a Foreign Keys FK and refers into relation **S**.
- Then referential integrity constraint requires that $t[FK]$ should (where $t \in R$)
  - either it is NULL, or
  - There is some tuple $s \in S$ such that $t[FK] = s[K]$, where K is key of relation S.
- Foreign key can also be composite. Example Registration refers to Offers


# Examples

- Schema examples are given in other PDF.
- Understand following from examples
  - Interpretation of a tuple in each of Relation
  - Keys
  - Foreign Keys. Kind of relationships captured by Foreign Keys.
- Running Examples: XIT, Company, DA-ACAD

# Relation Operations

- Three types of Operations.
- Data Definition: this is basically Defining Schema for all relations (and hence for Database)
- Data Update operations: updates relations, that are
  - Add new Tuple(s)
  - Modify Tuples(s)
  - Delete Tuple(s)
- Answering Queries. Query here typically mean, getting answer of an “Information Question” from database (i.e. relations)
- Note: the term query may also be used for SQL update operations


# [Meta] **Data Definition**


# Data Definition

- Here we define Schemas for all relations that are to be there in a database.
- DDL (Data Definition Language) part of SQL, referred as SQL-DDL is used for this purpose.
- Following are commonly used commands -
  - CREATE/DROP SCHEMA
  - **CREATE/ALTER/DROP TABLE**
  - CREATE/DROP DOMAIN/TYPE
  - CREATE/DROP VIEWS
- Note: SQL is case insensitive!

# CREATE SCHEMA command

- CREATE SCHEMA command creates a container to hold all elements of a “database instance”.
- For each database, you create a new schema.
- Examples
  - **CREATE SCHEMA XIT;**
- DROP Schema command is used to remove an Schema. Removing an Schema, also require deleting its content
  - **DROP SCHEMA XIT;**
  - **DROP SCHEMA XIT CASCADE;**


# CREATE TABLE command

- CREATE TABLE is used to specify a new relation.
- Typically requires specifying following parameters
  - Name of relation
  - Its attributes, domain for attributes, and
  - Constraints.
- This command does following-
  - Creates empty relation in relation instance.
  - Makes appropriate entries in database catalogue; that is schema definition is put into database catalogue


# CREATE TABLE command

- **CREATE TABLE** *R* ($A_1$ dom($A_1$), $Aa_2$ dom($A_2$), ..., $A_n$ dom($A_n$), ($constraint_1$), .., ($constraint_k$));
  - *R* is the name of the relation
  - each $A_i$ is an attribute name in the schema of relation R
  - dom($A_i$) is the data type(domain) for attribute $A_i$

- Data Types in SQL are used for defining domain.
  - CHAR, VARCHAR, BINARY, DATE, TIME, TIMESTAMP, CLOB, BLOB

- Common Constraints: Primary Key, Foreign Key, NOT NULL, UNIQUE, CHECK, DEFAULT VALUE


# Example CREATE TABLE

```
CREATE TABLE department (
     DID CHAR(2) PRIMARY KEY,
     DNAME VARCHAR(30) NOT NULL
);

CREATE TABLE program (
     PID CHAR(3),
     PNAME VARCHAR(20) NOT NULL,
     INTAKE SMALLINT,
     DID CHAR(2) REFERENCES department(did),
     PRIMARY KEY (PID)
);

CREATE TABLE student (
     StudID CHAR(3),
     Name VARCHAR(20) NOT NULL,
     ProgID CHAR(3) REFERENCES program(pid),
     "class" smallint,
     cpi decimal(4,2)
);
```

# Primary Key and Keys

- In theory, we have concept of “Keys” (or Candidate Keys)
- When we implement database, we define Primary Key.
- Any of the Key of the relation can be chosen as Primary Key.
- Primary Key implies following-
  - No tuple can have same value for PK attributes
  - None of attribute of PK (as PK can be composite) can have NULL value in a tuple. That means NOT NULL constraint is also applicable to Primary Key attributes.
  - Your DBMS automatically creates an “Index” on Primary Key attributes.

# Referential Integrity Constraints

- Defined in varied [syntax] manner.
- However, we typically say following here
  - What attribute(s) are FK in a relation
  - A relation can have multiple Foreign Keys
  - What relation a FK refers to [optionally attribute name in referred relation, by default PK of other relation]
  - We also [optionally] specify **Referential Actions**


# Referential Integrity Constraints

- **Referential Actions:** change in values of FK attribute can cause constraint violation. Referential actions defines possible actions when such a situation occurs.
- In absence of this, operation leading FK constraint violation will be denied to be executed by DBMS.
- Referential Actions are specified by following clause-
  ON UPDATE <action> ON DELETE <action> as part of CREATE TABLE statement.


# Referential Actions

- And actions could be one of following
  - SET NULL – sets any referencing FK to null
  - SET DEFAULT - sets any referencing FK to default value, which may be null
  - CASCADE – On delete, this deletes any rows with referencing foreign key values. On update this updates, this updates referencing foreign key values to new values.
  - NO ACTION- rejects any update that cause referential integrity violation
  - RESTRICT- same as NO ACTION with the additional restriction that the integrity check cannot be deferred.

# Example - Referential Actions

- A WORKS_ON table from company database.

```
CREATE TABLE works_on(
    essn DECIMAL(9,0),
    pno SMALLINT,
    hours DECIMAL(5,1),
    FOREIGN KEY (essn) REFERENCES employee(ssn)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (pno) REFERENCES project(pno)
        ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (essn,pno)
);
```


# Other Commands

- ALTER TABLE
  - ADD COLUMN, ALTER COLUMN, DROP COLUMN
  - ADD CONSTRAINT, DROP CONSTRAINT
- DROP TABLE
- DROP SCHEMA
- Use of “CASCADE” to force the ACTION
  - WITH CAUTION


# Data Update Operations


# Data Update Operations

- Update operation modify state of a *relation*.
- There are three types of update operations. Names are drawn from their keyword in SQL.
  - INSERT: add new tuples to a relation.
  - UPDATE: modify data of existing tuples of a relation.
  - DELETE: remove tuples from a relation.
- Some examples and related discussions are available in other PDF
- CONCERN: Operations violating database integrity constraints should not be allowed to be executed.
  - New data may not agree with database constraints.


# Possible Constraint violations

- INSERT:
  - Leading duplicate values in PK
  - Not having values for attributes that are constrained by NOT NULL
  - Value for FK attribute are not referring to a valid tuple
- UPDATE
  - Most of above are applicable to here also
- DELETE
  - Tuple being deleted might be referred by other tuple!