

**Modul Number: 0750362**

**Module Name: Database Applications**

**Teacher: Eman Alnaji**

**Part I**

**Revision of Database Concepts**

**Basic Definitions:**

* **Data:** Known facts that can be recorded and have an implicit meaning.
* **Database :** A collection of related data
* **Database management system (DBMS):** A collection of computer programs, which enables users to create and maintain databases.
* **Schema**: description of data at some level (*e.g.,*  tables, attributes, constraints, domains)

**A Relation:**

Relation is a set of tuples/records

* + Tuple ordering irrelevant
	+ Cardinality of relation = number of tuples
* All tuples in a relation have the same structure; constructed from the same set of attributes
	+ Attributes named (=> ordering irrelevant)
	+ Value of an attribute drawn from the attribute’s domain



Figure 1: Student Relation.

**Relation as A Table**

Relation 🡺 Table

Tuple 🡺 Row

Attribute 🡺 Column

**Relational Database Management System (RDBMS)**

* Finite set of relations
* Database schema = set of relations (and other things)

**Database Schema (Example):**

* Student (Id: INT, Name: STRING, Address: STRING, Status: STRING)
* Professor (Id: INT, Name: STRING, DeptId: DEPTS)
* Course (DeptId: DEPTS, CrsName: STRING, CrsCode: COURSES)
* Transcript (CrsCode: COURSES, StudId: INT, Grade: GRADES, Semester: SEMESTERS)
* Department(DeptId: DEPTS, Name: STRING)



Figure2: Three-schema Architecture of an RDBMS.

**DDL: Data Definition Language**

Used by DBAs or designers to define schema

(CREATE, DROP, ALTER)

**DML: Data Manipulation Language**

 Used to insert, update and delete data in database tables.

 (INSERT, UPDATE, DELETE, SELECT)

**DCL: Data Control Language**

Used to create roles, permissions, and referential integrity and to control access on a database.

 (GRANT, REVOKE)

**DSL: Data Storage Language**

 Specifies Internal Schema (used by the RDBMS)

**TCL: Transactional Control Language**

Used to manage different transactions occurring within a database.

 (COMMIT, ROLLBACK)

**Integrity Constraints**

* Part of schema
* Restriction on state (or sequence of states) of database
* Enforced by DBMS

**Database Integrity**

Implies that the data held in the tables of the database is consistent in terms of the Relational Data Model

* Two Types
	+ Entity integrity (PK)
	+ Referential Integrity (FK)

**Relation Keys Constraints**

* **Key Constraint**: Values in a column (or columns) of a relation are unique: at most one row in a relation instance can contain a particular value(s)
* **What is a Key?**
	+ A minimal set of attributes satisfying key constraint
* Every relation has a key
* **Candidate Key-**
	+ No two tuples of the relation will have identical entries in all attributes of the key.
	+ The number of attributes that comprises the key must be minimal.
* **Primary Key-** Since a table may have more than one candidate key, one should be designated as the primary key (PK) of the relation.

**Examples:**

* + **primary key** (Id in Student) –
	+ **candidate key** ((Name, Address) in Student)
* A RDBMS allows only one primary key per table.
	+ Once a PK has been selected, any remaining candidate keys are called alternate keys.
* A primary key may be composed of
	+ a single attribute (single primary key)
		- E.g. ID
	+ More than one attribute (composite primary key)
		- E.g. Code + Serial
* An attribute that is a primary key can not have a null value
* An attribute that is part of any key is called a prime attribute.



Figure3: Example of Primary Key



Figure4: Example of Composite Key

* Primary keys are
	+ defined using Data Definition Language (DDL)
	+ Automatically enforced by the RDBMS
	+ Generally are defined at the time the tables are created.
* When selecting primary keys, we need to choose attributes that satisfy the uniqueness and minimalist conditions for all permissible data.

**NULL**

* What is NULL?
	+ a NULL value is used to represent missing information, unknown, or inapplicable data.
	+ A NULL value is not a zero value
	+ A NULL value doesn’t represent a particular value within the computer.

**Foreign Key Constraint**

* R**eferential integrity** => Item named for **primary key** attribute/s in one relation must correspond to tuple(s) in another that describes the item
	+ Employee (ProjId) references Project(ProjId)
	+ Professor(DeptId) references Department(DeptId)
* a1 is a **foreign key** of R1 referring to a2 in R2 => if v is a value of a1, there is a ***unique* tuple** of R2 in which a2 has value v
	+ This is a special case of referential integrity: a2 must be a **candidate key** of R2 (DeptId is a key of Department)
	+ If no row exists in R2 => violation of referential integrity
	+ Not all rows of R2 need to be referenced: relationship is not symmetric
	+ Value of a foreign key might not be specified (DeptId column of some professor might be null)



Figure5: Example of Foreign key constraint



Figure6: Company Database (Foreign key constraints)

* Names of a1 and a2 need not be the same.
	+ With SQL tables:

 Foreign key *<name>* professor (DeptId) references

 Department (Dno)

**DeptId** attribute of **Professor Table** references to **Dno** attribute in

**Department Table**

* R1 and R2 need not be distinct.
* The attributes of where the referential integrity exists must have the same data type and length.
* Foreign key might consist of several columns
	+ (CrsCode, Semester) of Transcript references (CrsCode, Sem) of Teaching
* R1(a1, …an) references R2(b1, …bn)
	+ There exists a 1 - 1 relationship between a1,…an and b1,…bn
	+ ai and bi have same domains (although not necessarily the same names)
	+ For every tuple **T** in R1 over ai’s there exists a unique tuple **S** in R2 over bi’s, with **T = S**
	+ b1,…bn is a candidate key of R2

**Semantic Constraints**

* Domain, primary key, and foreign key are examples of structural (syntactic) constraints
* **Semantic constraints** express rules of application:
	+ e.g., number of registered students ≤ maximum enrollment
	+ SQL calls them Check constraint