UNIT II

Relational Model: Introduction to relational model, concepts of domain, attribute, tuple, relation, importance of null values, constraints (Domain, Key constraints, integrity constraints) and their importance BASIC SQL: Simple Database schema, data types, table definitions (create, alter), different DML operations (insert, delete, update), basic SQL querying (select and project) using where clause, arithmetic & logical operations, SQL functions(Date and Time, Numeric, String conversion).

Various data models

Data Model: Data Model is a collection of concepts that can be used to describe the structure of database.

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS.

An **Entity** is a real-world object that are represented in database. It can be any object,place,person or class.Data are stored about such entities.

Entity Set: The collection of similar entities is called as Entity Set.

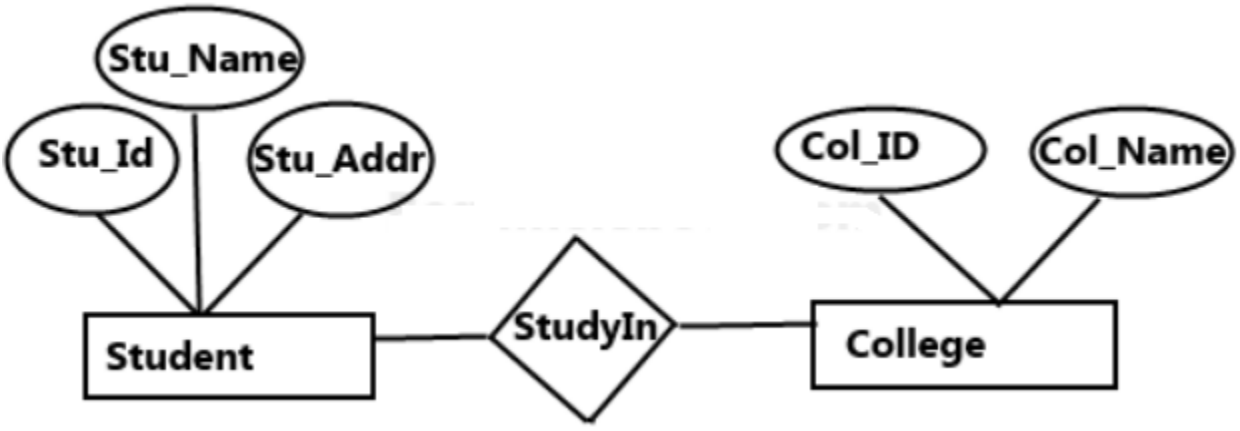
In DBMS we store data in the form of table containing information about entity type like students,teachers,employees etc. In a school database, a table containing information about all the students,students here are entities.

Entity is represented by set of **Attributes** which are properties used to describe an entity. **It basically describes an entity.** All entities in a given entity set have the same attributes. For example student have properties like name, redgno, address etc they are attributes which combined make table.

Entity Relationship Data Model

Entity relationship model is based on the notion of the real world entities and their relationships. While formulating the real world scenario in to the database model an entity set is created and this model is dependent on two vital things and they are

- Entity and their attributes
- Relationships among entities

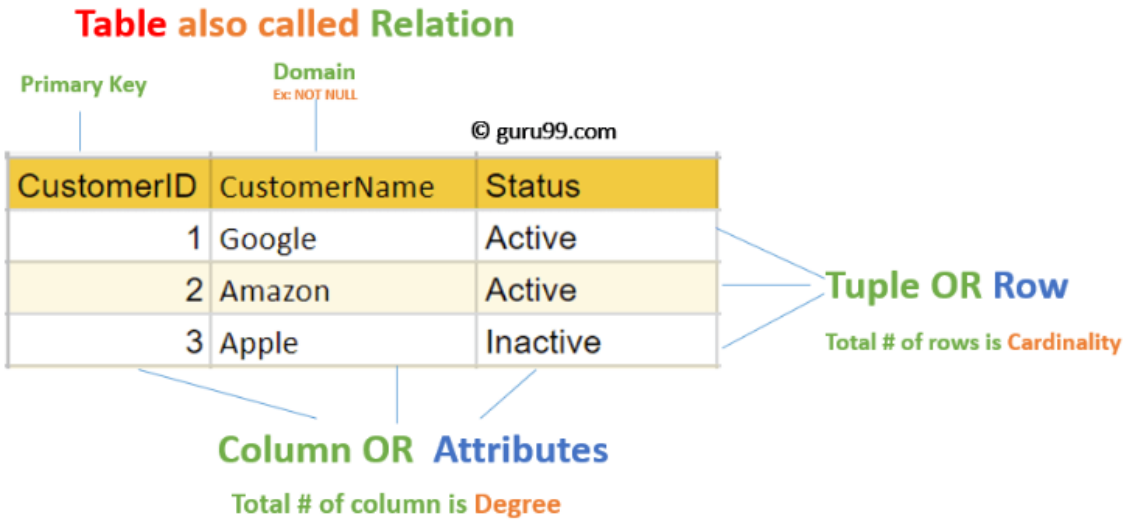


An entity has a real world property called attribute and attribute define by a set of values called domain. For example, in a university a student is an entity, university is the database, name and age and sex are the attributes. The relationships among entities define the logical association between entities.

Relational Data Model

Relational model is the **most popular model** and the most extensively used model. The relational model represents the database as a collection of **relations(tables).** Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.



Difference between E-R Model and Relational Model

The basic difference between E-R Model and Relational Model is that E-R modelspecifically deals with entities and their relations. On the other hand, the Relational Model deals with Tables and relation between the data of those tables. The main difference between E-R Model and Relational Model is that E-R Model is entity specific, and Relational Model is table specific.

BASIC SQL

SQL stands for Structured Query Language
SQL lets you access and manipulate databases
SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.

What Can SQL do?
SQL can execute queries against a database
SQL can retrieve data from a database
SQL can insert records in a database
SQL can update records in a database
SQL can delete records from a database
SQL can create new databases
SQL can create new tables in a database
SQL can create stored procedures in a database
SQL can create views in a database
SQL can set permissions on tables, procedures, and views

SQL Data Types

Each column in a database table is required to have a name and a data type.

In MySQL there are three main data types: string, numeric, and date and time.

String Data Types

1)CHAR(size) :A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
2)VARCHAR(size):A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
3)int or integer
4)number
5)date

Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated.

Types of SQL Commands : Data Definition Language(DDL)

- These are used for creation, deletion, and modification of definitions for database tables and views.

- In order to make/perform changes on the physical structure of any table residing inside a database, DDL is used. These commands when executed are auto commit in nature and all the changes in the table are reflected and saved immediately. DDL commands includes :

S.No	DDL Commands	Description	Sample Query
1.	CREATE	Used to create tables or databases.	CREATE table student;
2.	ALTER	Used to modify the values in the tables.	ALTER table student add column roll_no int;
3.	RENAME	Used to rename the table or database name.	RENAME student to student_details;
4.	DROP	Deletes the table from the database.	DROP table student_details;
5.	TRUNCATE	Used to delete a table from database.	TRUNCATE table student_details;

Alter table table_name add (column_name data type);
Alter table table_name modify (column_name data type);
Drop table deletes all rows including structure.
Truncate table deletes all rows only.

Types Of SQL Commands : Data Manipulation Language(DML)

- Once the tables are created and database is generated using DDL commands, manipulation inside those tables and databases is done using DML commands. The advantage of using DML commands is, if in case any wrong changes or values are made, they can be changes and rolled back easily. DML commands includes :

S.No	DML Command	Description	Sample Query
1.	INSERT	Used to insert new rows in the tables.	INSERT into student(roll_no, name) values(1, Anoop);
2.	DELETE	Used to delete a row or entire table.	DELETE table student;
3.	UPDATE	Used to update values of existing rows of tables.	UPDATE students set s_name = 'Anurag' where s_name like 'Anoop';
4.	LOCK	Used to lock the privilege as either read or write.	LOCK tables student read;
5.	MERGE	Used to merge two rows of existing tables in database.	

Delete from tablename where condition;

Update tableName set column=new value where condition;

Types Of SQL Commands : Data Control Language(DCL)

- DCL commands as the name suggests manages the matters and issues related to the data control in any database. TCL commands mainly provides special privilege access to users and is also used to specify the roles of users accordingly. There are two commonly used DCL commands, these are:

S.No	DCL Commands	Description	Sample Query
1.	**GRANT**	**Used to provide access to users.**	**GRANT CREATE table to user1;**
2.	**REVOKE**	**Used to take back the access privileges from the users.**	**REVOKE CREATE table from user1;**

Grant previlages on objectName to username;
Grant select on faculty to user1;
Revoke previlages on objectName from username;
Revoke select on faculty from user1;

Types Of SQL Commands : Data Query Language(DQL)

- Data query language consists of only one command over which data selection in SQL relies. SELECT command in combination with other SQL clauses is used to retrieve and fetch data from database/tables on the basis of certain conditions applied by user.

S.No	DQL Command	Description	Sample Query
1.	**SELECT**	**Used to fetch data from tables/database.**	**SELECT * from student_details;**

Types Of SQL Commands : Transaction Control Language(TCL)

- Transaction Control Language as the name suggests manages the issues and matters related to the transactions in any database. They are used to rollback or commit the changes in the database.
- Roll back means “Undo” the changes and Commit means “Applying” the changes. There are three major TCL commands.

S.No	TCL Commands	Description	Sample Query
1.	**ROLL BACK**	**Used to cancel or UNDO the changes made in the database.**	**ROLLBACK;**
2.	**COMMIT**	**Used to deploy or apply or save the changes in the database.**	**COMMIT;**
3.	**SAVEPOINT**	**Used to save the data on temporary basis in the database.**	**SAVEPOINT roll_no;**

Savepoint <save point name>;

Savepoint A;

Rollback to savepoint A;

The form of a Basic SQL query:

```
SELECT      [DISTINCT]  select-list
FROM from-list
WHERE       qualification
```

From-list : From-list in the FROM clause is a list of table name(relation names). A table name can be followed by a range variable; a range variable particularly useful when the same table name appears more than once in the from-list.

select-list A list of column names of tables named in the table-list. Column names can be prefixed by range variable name.

qualification in the WHERE clause is Boolean expression using relational operators(<,>,<=,>=, <>, =) and logical operators AND, OR and NOT. The qualification starts with WHERE clause that is used to specify selection conditions on the tables mentioned in the FROM clause.

DISTINCT is optional keyword indicating that answer should not contain duplicates.

Examples:

```
create table dept(
  deptno     number(2,0),
  dname      varchar2(14),
  loc        varchar2(13),
  constraint pk_dept primary key (deptno)
);
```

```
create table emp(
  empno    number(4,0),
  ename    varchar2(10),
  job      varchar2(9),
  mgr      number(4,0),
  hiredate date,
  sal      number(7,2),
  comm number(7,2),
  deptno   number(2),
  constraint pk_emp primary key (empno),
```

```
  constraint fk_deptno foreign key (deptno) references dept (deptno)
);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	smith	clerk	7902	17-Dec-80	800		20
7499	allen	salesman	7698	20-Feb-81	1600		30
7521	ward	salesman	7698	22-Feb-81	1250	500	30
7566	jones	manager	7839	02-Apr-81	2975		20
7654	martin	salesman	7698	28-Sep-81	1250	1400	30
7698	blake	manager	7839	01-May-81	2850		30
7782	clark	manager	7839	09-Jan-81	2450		10
7788	scott	analyst	7566	19-Apr-87	3000	200	20
7839	king	president		17-Nov-81	5000		10
7844	turner	salesman	7698	08-Sep-81	10000		30
7876	adems	clerk	7788	23-May-87	1100		20
7900	james	clerk	7698	03-Dec-81	950		30
7902	ford	analyst	7566	03-Dec-81	3000		20
7934	miller	clerk	7782	23-Jan-82	1300		10

SQL Functions

Numberic Functions

Function	**Input Argument**	**Value Returned**
ABS (m)	m = value	Absolute value of m
MOD (m, n)	m = value, n = divisor	Remainder of m divided by n
POWER (m, n)	m = value, n = exponent	m raised to the nth power
ROUND (m [, n])	m = value, n = number of decimal places, default 0	m rounded to the nth decimal place
TRUNC (m [, n])	m = value, n = number of decimal places, default 0	m truncated to the nth decimal place
SIN (n)	n = angle expressed in radians	sine (n)
COS (n)	n = angle expressed in radians	cosine (n)
TAN (n)	n = angle expressed in radians	tan (n)
SQRT (n)	n = value	positive square root of n
EXP (n)	n = value	e raised to the power n
LN (n)	n > 0	natural logarithm of n
LOG (n2, n1)	base n2 any positive value other than 0 or 1, n1 any	logarithm of n1, base n2

	positive value	
CEIL (n)	n = value	smallest integer greater than or equal to n
FLOOR (n)	n = value	greatest integer smaller than or equal to n

Here are some examples of the use of some of these numeric functions:

```
select round (83.28749, 2) from dual;

select sqrt (3.67) from dual;

select power (2.512, 5) from dual;

select ceil(15.25) from dual;
```

String Functions

Function	Input Argument	Value Returned
INITCAP (s)	s = character string	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	s = character string	All letters are changed to lowercase.
UPPER (s)	s = character string	All letters are changed to uppercase.
CONCAT (s1, s2)	s1 and s2 are character strings	Concatenation of s1 and s2. Equivalent to s1 || s2
LTRIM (s [, set])	s is a character string and set is a set of characters	Returns s with characters removed up to the first character not in set; defaults to space
RTRIM (s [, set])	s is a character string and set is a set of characters	Returns s with final characters removed after the last character not in set; defaults to space
REPLACE (s, search_s [, replace_s])	s = character string, search_s = target string, replace_s = replacement string	Returns s with every occurrence of search_s in s replaced by replace_s; default removes search_s
SUBSTR (s, m [, n])	s = character string, m = beginning position, n = number of characters	Returns a substring from s, beginning in position m and n characters long; default returns to end of s.
LENGTH (s)	s = character string	Returns the number of characters in s.

Here are some examples of the use of String functions:

```
Select length ('Alan') from dual;

select concat ('Alan', 'Turing') as "NAME" from dual;

select substr ('Alan Turing', 1, 4) from dual;

select upper ('alan Turing') from dual;
```

String / Number Conversion Functions

Function	Input Argument	Value Returned
TO_CHAR (m [, fmt])	m = numeric value, fmt = format	Number m converted to character string as specified by the format
TO_NUMBER (s [, fmt])	s = character string, fmt = format	Character string s converted to a number as specified by the format

Formats for TO_CHAR Function

Symbol	Explanation
9	Each 9 represents one digit in the result
0	Represents a leading zero to be displayed
$	Floating dollar sign printed to the left of number
L	Any local floating currency symbol
.	Prints the decimal point
,	Prints the comma to represent thousands

Examples:

```
SELECT TO_CHAR (SYSDATE,'YYYY') from dual;
SELECT TO_CHAR(SYSDATE,'DD MON YYYY') from dual;
```

Group Functions

Function	Input Argument	Value Returned

AVG ([DISTINCT | ALL] col)	col = column name	The average value of that column
COUNT (*)	none	Number of rows returned including duplicates and NULLs
COUNT ([DISTINCT | ALL] col)	col = column name	Number of rows where the value of the column is not NULL
MAX ([DISTINCT | ALL] col)	col = column name	Maximum value in the column
MIN ([DISTINCT | ALL] col)	col = column name	Minimum value in the column
SUM ([DISTINCT | ALL] col)	col = column name	Sum of the values in the column

Examles:

```
Select max(salary) from Faculty;

Select sum(salary) from faculty;

Select min(salary) from faculty;
```

Date and Time Functions

Function	Input Argument	Value Returned
ADD_MONTHS (d, n)	d = date, n = number of months	Date d plus n months
LAST_DAY (d)	d = date	Date of the last day of the month containing d
MONTHS_BETWEEN (d, e)	d and e are dates	Number of months by which e precedes d
NEW_TIME (d, a, b)	d = date, a = time zone (char), b = time zone (char)	The date and time in time zone b when date d is for time zone a
NEXT_DAY (d, day)	d = date, day = day of the week	Date of the first day of the week after d
SYSDATE	none	Current date and time
GREATEST (d1, d2, ..., dn)	d1 ... dn = list of dates	Latest of the given dates
LEAST (d1, d2, ..., dn)	d1 ... dn = list of dates	Earliest of the given dates

Date Conversion Functions

Function	Input Argument	Value Returned
TO_CHAR (d [, fmt])	d = date value, fmt = format for string	The date d converted to a string in the given format
TO_DATE (s [, fmt])	s = character string, fmt = format for date	String s converted to a date value
ROUND (d [, fmt])	d = date value, fmt = format for string	Date d rounded as specified by the format
TRUNC (d [, fmt])	d = date value, fmt = format for string	Date d truncated as specified by the format

Date Formats

Format Code	Description	Range of Values
DD	Day of the month	1 - 31
DY	Name of the day in 3 uppercase letters	SUN, ..., SAT
DAY	Complete name of the day in uppercase, padded to 9 characters	SUNDAY, ..., SATURDAY
MM	Number of the month	1 - 12
MON	Name of the month in 3 uppercase letters	JAN, ..., DEC
MONTH	Name of the month in uppercase padded to a length of 9 characters	JANUARY, ..., DECEMBER
RM	Roman numeral for the month	I, ..., XII
YY or YYYY	Two or four digit year	71 or 1971
HH:MI:SS	Hours : Minutes : Seconds	10:28:53
HH 12 or HH 24	Hour displayed in 12 or 24 hour format	1 - 12 or 1 - 24
MI	Minutes of the hour	0 - 59
SS	Seconds of the minute	0 - 59
AM or PM	Meridian indicator	AM or PM
SP	A suffix that forces the number to be spelled out.	e.g. TWO THOUSAND NINE
TH	A suffix meaning that the ordinal number is to be added	e.g. 1st, 2nd, 3rd, ...

FM	Prefix to DAY or MONTH or YEAR to suppress padding	e.g. MONDAY with no extra spaces at the end

Here are some examples of the use of the Date functions:

```
select to_char ( sysdate, 'MON DD, YYYY' ) from dual;

select to_char ( sysdate, 'HH12:MI:SS AM' ) from dual;

select to_char ( new_time ( sysdate, 'CDT', 'GMT'), 'HH24:MI' ) from dual;

select greatest ( to_date ( 'JAN 19, 2000', 'MON DD, YYYY' ),
                  to_date ( 'SEP 27, 1999', 'MON DD, YYYY' ),
                 to_date ( '13-Mar-2009', 'DD-Mon-YYYY' ) )
from dual;

select next_day ( sysdate, 'FRIDAY' ) from dual;
select last_day ( add_months ( sysdate, 1 ) ) from dual;
```

Unit - III

Relational Model, Relational Algebra and Calculus

Syllabus

Introduction to the Relational Model Integrity Constraint over Relations, Enforcing Integrity Constraints, Querying Relational Data, Logical Database Design, Introduction to Views, Destroying/Altering Tables and Views.

Relational Algebra Selection and Projection Set Operations, Renaming, Joins, Division, Examples of Algebra Overviews, Relational Calculus, Tuple Relational Calculus, Domain Relational Calculus, Expressive Power of Algebra and Calculus.

3.1 Introduction to the Relational Model

Codd proposed the relational data model in 1970. At that time, most database systems were based on one or two older data models (the hierarchical model and the network model), the relational model revolutionized the database field and largely supplanted (replaced with) these earlier models. Today, the relational model is by far the dominant data model and the foundation for the leading DBMS products, including IBM's DB2 family, Informix, Oracle, Sybase, Microsoft's Access, SQL server, Foxbase and Paradox. Relational database systems are ubiquitous (very common) in the market place and represent a multibillion dollar industry.

The relational model is very simple and elegant. A database is a collection of one or more relations, where each relation is a table with rows and columns. This simple tabular representation enables even novice users to understand the contents of database and it permits the use of simple, high level languages to query the data.

The relational model uses Data Definition Language and Data Manipulation Language (Query Language), the standard language for creating, accessing, manipulating and querying (i.e., retrieving) data in a relational DBMS.

The main construct for representing data in the relation model is a relation (table).

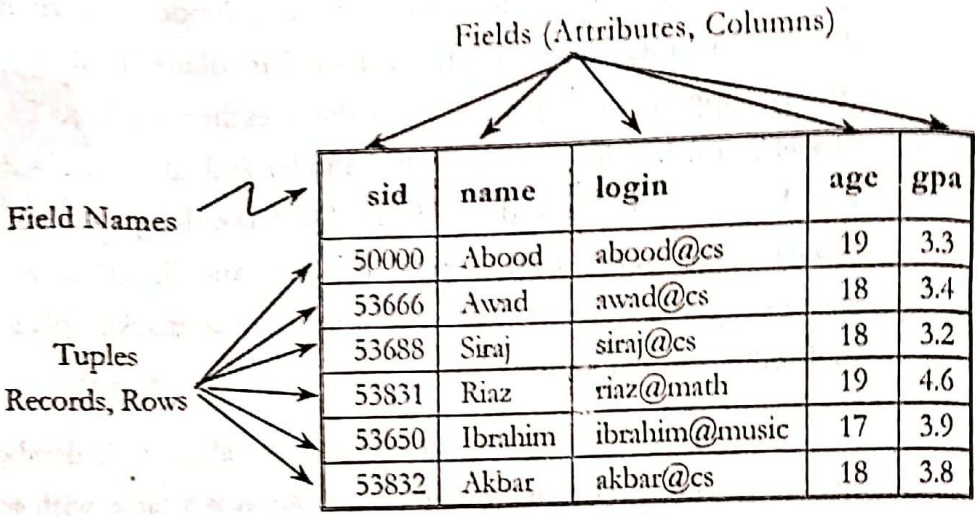
The SQL-92 language standard uses the word table to denote relation. The subset of SQL that supports the creation, deletion and modification of tables is called the Data Definition Language (DDL).

The create table statement is used to define a new table. To create the students relation, we can use the following statement,

```
sql> create table students(sid    char(20),
                           name   char(30),
                           login  char(20),
                           age    integer,
                           gpa    real);
```

Thus, the above statement creates the students relation shown in Table 3.1.1 by accepting the values from the user.

Table 3.1.1 An Instance of Student Relation



sid	name	login	age	gpa
50000	Abood	abood@cs	19	3.3
53666	Awad	awad@cs	18	3.4
53688	Siraj	siraj@cs	18	3.2
53831	Riaz	riaz@math	19	4.6
53650	Ibrahim	ibrahim@music	17	3.9
53832	Akbar	akbar@cs	18	3.8

An important component of a relational model is that it specifies some conditions that must be satisfied while accepting the values from the user which prevents the entry of incorrect information. Such conditions, are called as Integrity Constraints (ICs), which enable the DBMS to reject operations that might corrupt the data.

3.1.1 Relation : Fundamental Concepts

1) Relation [IQ 1]

[April/May – 2009, Set – 4]

The main construct for representing data in the relation model is a relation (table). A relation can be thought of as a set of records in the form of two-dimensionable table containing rows and columns of data. A relation consists of two things, a relation schema and a relation instance.

2) Relation Schema

[IQ 1]

[April/May – 2009, Set – 4]

A relation schema contains the basic information of a table or relation. This information includes the name of the entire table, the names of the column (also known as fields or attributes) and the data types (also known as domain) associated with each column.

For Example : A relation schema for the relation called students could be expressed using the following representation,

students(sid : string, name : string, login : string, age : integer, gpa : real)

The name of the entire table or relation is "students". There are five column names sid, name, login, age, gpa with respective data types string, string, string, integer, real associated with them.

3) Relation Instance

[IQ 1]

[April/May – 2009, Set – 4]

A relation instance is a set of rows (also known as records or rows) that when combined together forms the schema of the relation. A relation instance can be thought of as a table in which each tuple is a row and all rows have same number of fields (columns).

4) Relational Database Schema : A relational database schema is a collection of relation schemas, describing one or more relations.

5) Domain : Domain is synonymous with data type. Attributes can be thought of as columns in a table. Therefore, an attribute domain refers to the database associated with a column.

6) Relation Cardinality : The relation cardinality is the number of tuples (rows or records) in the relation.

7) Relation Degree **[IQ 1]**

[April/May – 2009, Set – 4]

The relation degree is the number of fields (column or attributes) in the relation.

8) Tuples / Records : The rows of the table is also known as records or tuples.

9) Fields / Attributes : The columns of the table is also known as fields or attributes.

3.1.2 Creating and Modifying Relations using SQL [IQ 2]

[April/May – 2009, Set – 4]

1) **Creating Relations in SQL (Structured Query Language) :** Relations (tables) are created in SQL using create table statement as,

Example

```
create table customers(cid     integer,
                       cname   char(30),
                       accno   integer,
                       bname   char(40),
                       amt     real);
```

2) **Inserting Tuples in a Table :** Tuples are inserted in a table using an insert command as,

Example

```
insert into customers(cid, cname, accno, bname, amt)
        values(6001, 'shazia', 41, 'UTI', 15000.00);
```

Note : Listing of columns is not compulsory.

3) **Deleting Tuples from a Table :** Tuples in the table can be deleted using a delete command in SQL as,

Example

```
delete from    customers C
        where  C.accno = 41;
```

4) **Modifying the Column Values :** Values in a particular row can be changed using an update command as,

Examples

1)
```
update  customers C
set     C.cname = 'Zainab', C.amt = 20000.50
where   C.cid = 6001;
```

Table has to be Added : The keywords where and set are used to determine the modifying rows and the modifying procedure (i.e., how the rows can be changed).

2) When a customer deposits some money in his account its instance can be updated as,

```
update  customers C
set     C.amt = C.amt + 1000
where   C.accno <= 60;
```

Updated instance is shown in the Table 3.1.2.

Table 3.1.2 Customers Instance C_2 after Performing Update Operation

cid	cname	accno	bname	amt
6001	Shazia	41	UTI	16000.00
6312	Zainab	52	SBI	21000.50
6448	Aasia	60	SBH	11500.00
6015	Neelima	72	Darussalam Bank	18050.50

3.2 Integrity Constraints over Relations

Integrity constraint is a condition that ensures the correct insertion of the data and prevents unauthorized data access thereby preserving the consistency of the data.

For example, the roll number of a student cannot be a decimal value. The database enforces the constraint that the instance of roll number can have only integer values.

Integrity constraints can be enforced in the following cases,

1) As soon as the database schema is defined, the associated integrity constraints must be specified.

2) Validity of the changes made to the data must be checked while executing the application. Hence, the time of ICs check, the changing statement and the associated transaction must be necessarily specified.

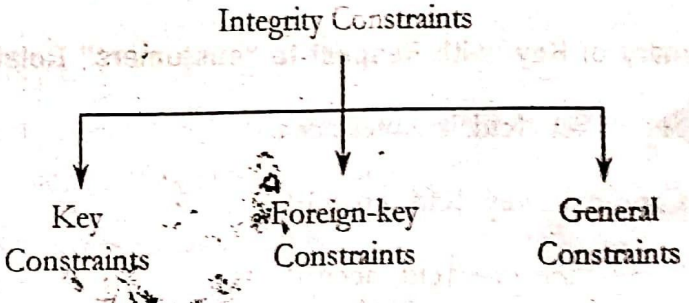


Fig. 3.2.1 Types of Integrity Constraints

3.2.1 Key Constraints

Key constraints are used in relations to uniquely identify some records of the relation. A key constraint can be defined as a statement that consist of minimal subset of attribute that uniquely determine a record in a table. This set of attributes that work in accordance with the key constraint is called as candidate key. A candidate key is a collection of fields/columns/attributes that uniquely identifies a tuple. For example, in "customer" relation the attribute "cid" is a key, it uniquely defines a tuple in a relation. No two rows in a relation "customer" can have the same "cid" value. The set of attributes that form a candidate key need not be all keys. The attributes may be treated as candidates to be taken as key. For example, the set (cid, cname) is a candidate key which means either cid or cname can be taken as key but not both. Each of them independently and uniquely identifies a particular row. The alternate keys are the candidate keys that are not taken as keys.

Composite Key : Composite key consist of more than one attribute that uniquely identifies a tuple in a relation. All the attributes that form a set of keys and all of them taken together determines a unique row in a table. For example, the set (cid, accno) is a composite key which maintains the uniqueness of each row. Both cid, accno are taken as keys.

Super Key : A super key is a combination of both candidate key and composite key. That is a super key is a set of attributes or a single attribute that uniquely identifies a tuple in a relation. For example, consider the super key,

{cid, accno, cname}

Here, all the three attributes taken together can identify a particular record or a combination of any two attributes can identify a particular record or any one of the attribute can identify a particular record.

Primary Key : Only a single attribute can uniquely identify a particular record. More specifically, it can be defined as the candidate key which has been selected as key to identify unique records. For example, "cid" attribute in "customer" relation can be treated as a primary key.

Summary of Key (with Respect to "customers" Relation)

- Super key {cid, cname, accno}
- Candidate key {cid, cname}
- Composite key {cid, accno}
- Primary key {cid}

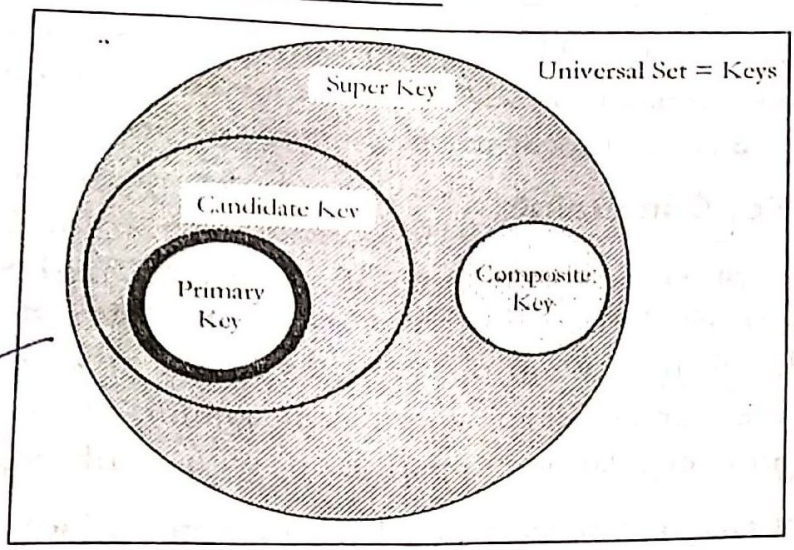


Fig. 3.2.2 Representing Relation between Different Keys