# CHAPTER 1

# DATABASES

Database is heart of pharmacoinformatics, provides adequate and updated information. In everyday life, we are in the need of information ranging from basic information to scientific information. Print and electronic indices (books and journals) developed to retain the published information, which facilitates quick identification of relevant information.

Technically database can be defined as

1. Physical collection of logically related records (data).
2. Collection of database files and each database file is a collection of records.

Four classes of databases known are

| | | |
|---|---|---|
| Archieval | - | It accepts data as it is and most are public. eg: GeneBank |
| Curated | - | Mostly private |
| Public | | |
| Private | | |

## 1.1 Types of Databases

Three major types of pharmacoinformatics databases are known and are

1. Pharmaceutical databases
   (a) Literature database
   (b) Chemical database
2. Biological databases
   (a) Structure databases
   (b) Sequence databases

3. Relational databases
    (a) Structured Query Language (SQL)
    (b) Practical Extraction and Report Language (PERL)

## 1.2 Components of Databases

Database has two main components

1. Field: Each piece of information in a database is called field, which means the smallest unit in a database. In database there can be five categories of fields, they are:
    - Numeric
    - Character
    - Logic
    - Memo
    - Date
2. Record: All the related events of particular field constitute a record, which is a collection of logically related fields.

## 1.3 Database Schema

The overall description of a database is called as database schema and 3 types of schemas are known.

1. Internal schema: It contains definitions of the stored records and specifies how the data is stored.
2. External schema: It describes external views of the data, there are many external schema for the given database. It excludes irrelevant data as well as data which the user is not authorized to access.
3. Conceptual schema: It describes the types of data stored in database and relationships between them.

### 1.3.1 Database Architecture

- The database architecture defines the nature of the data and structure of the data.
- The database architecture specifies, set of rules and processes that dictate how data should be stored in a database and how data is accessed by components of a system.
- The database architecture includes data types, relationships and naming conventions. It describes the organization of all database objects and their working methodologies. It affects integrity, reliability, scalability and performance of database.

### 1.3.2 ANSI-SPARC 3 Level Architecture

American National Standard Institute - Standards Planning And Requirements Committee (ANSI - SPARC) is an abstract design standard for Database Management System (DBMS). Most commercial DBMS are based on this system.

1. It allows independent customized user views – each user should be able to access the same data.
2. It hides the physical storage details from users.
3. The database administrator should be able to change the database storage structure without affecting the user views.
4. The internal structure of the database should be able to changes to the physical aspects of the storage.

The database administrator should be able to change the conceptual or global structure of the database without affecting the users.

### 1.3.3 Data Redundancy

Identical data stored in two or more files are known as data redundancy.

Dependencies between attribute (column) causes data redundancy. It occurs in database systems which have a field that is repeated in two or more tables. It leads to data anomalies and corruption, hence should be avoided. Database normalization prevents redundancy by making use of proper foreign keys.

## 1.4 Database Applications

- Redundancy can be reduced
- Data inconsistencies can be avoided
- Allows data sharing and data migration between systems
- Provides security restriction
- Data integrity can be maintained
- Balances conflicting requirements.

## 1.5 Database Management System (DBMS)

The stress is given to creation as well as management of database. Database management involves creating, modifying, deleting and adding data in files and using this data to generate reports or answer the queries.

The software allows to perform these functions easily are called as database management system (DBMS). DBMS attempt to make the physical data non-redundant and also optimizes resource utilization. The windows based database available in the market is MS ACCESS, which is a part of Microsoft windows.

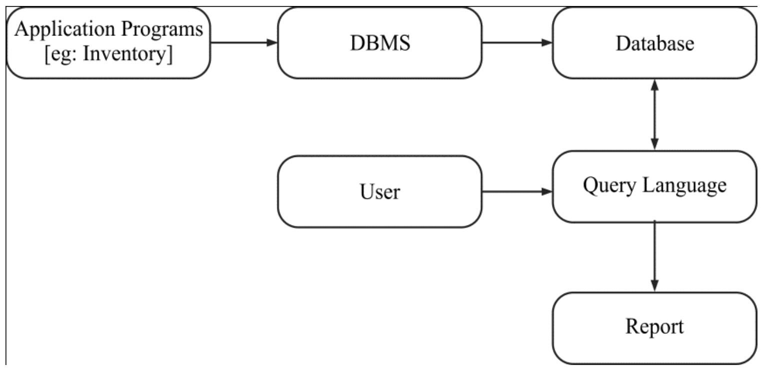


**Fig. 1.1** Schema represents DBMS approach.

DBMS allows inserting, updating, deleting and processing of data. Some of the important DBMS are Oracle, Ingress, Sybase, Dbase 3+, Foxpro, MS access, Dataease, Dataflex, Advanced revelation., etc. Primary tasks of DBMS are:

- Database development: Define and organize the contents, relationships and structure of the data needed to build s database.
- Database interrogation: It involves information retrieval and report generation. End users can selectively display information, produce reports and documents.
- Database maintenance: It helps in adding, deleting, updating, correcting and protecting the data in a database.
- Application development: It is used to develop prototypes of data entry careens queries, forms, reports, tables and lebels for a prototype application / use 4th Generation Language (4GL) or application generator to develop program codes.

**Benefits of DBMS**

1. The amount of data redundancy in stored data can be reduced.
2. Data inconsistencies can be removed.
3. Stored data can be shared by a single or multiple users.
4. Standards can be set and followed.

5. Data integrity can be maintained.
6. Security of data can be implemented.
7. Data independence can be achieved.

## 1.6 Relational Database Management System

A relational database matches data by using common characteristics found within the data set. The resulting groups of data are organized and much easier for people to understand. Such a grouping uses the relational model / schema and this database is called as relational database. The soft ware used to do this grouping is called as relational database management system (RDMS).

## 1.7 Normalization

Database normalization is a critical part of good database architecture, which ensures data integrity and avoids data redundancy.

## 1.8 Relationship

Database relationship creates a history for the tables. A properly normalized database has a well-organized hierarchy. For each relationship between two tables, one table is the parent and one table is the child. For example, a patient table may be the parent of prescription detail table.

## 1.9 Primary and Foreign Keys

Database relationships are embodied through primary keys in parent tables and foreign keys in child tables.

- Primary key is a field in a table whose value uniquely identifies each record and defines relationship within a database.
- Foreign key is a field in a table that refers to parent records in another table. It need not have unique values in the referencing relation.

## 1.10 Elements of DBMS

- Data Definition Language (DDL): It provides link between logical and physical views of the database. It is used to define the physical characteristics of each record and fields.

- Data Manipulation Language (DML): It provides techniques for retrieval, sorting, display and deletion of data/records.
- Data Query Language (DQL): It allows retrieving data from the database and imposing ordering upon it.
- Data Control Language (DCL): It controls the access to data and to the database.

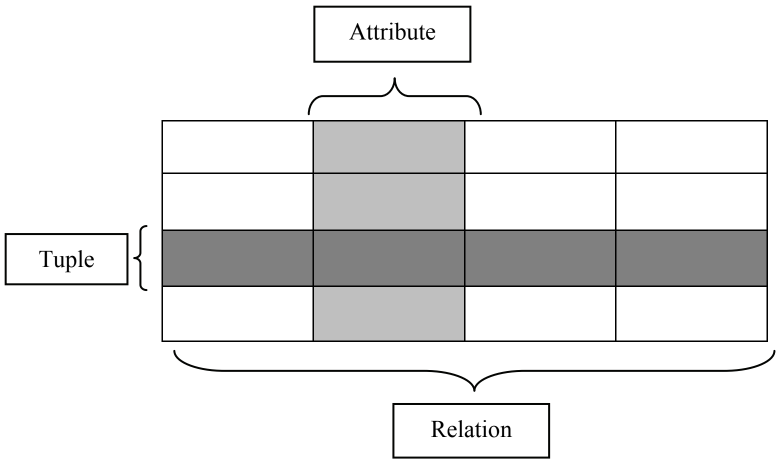
## 1.11 Relational Databases

A relational database uses relationally two-dimensional tables to store different pieces of information inside tables and nothing more. All operations on data are done on the tables themselves or produce other tables as a result. A relational database contains one or many tables, which is a basic storage structure of relational database management system (RDMS).

### 1.11.1 Relational Database Terminology

Relational database theory uses set of mathematical terms, which are roughly equivalent to Structured Query Language (SQL) database terminology.

| **Relational term** | **SQL equivalent** |
|---|---|
| Realtion, base | table |
| Derived | view, query result, result set |
| Tuple | Row |
| Attribute | coloumn |



A table is set of rows and columns. Each row is set of column with only one value for each. All rows from the same table have the same set of columns. The row from a relational table is analogous to a record, and the columns to a field. Relational database serve in two ways.

1. Retrieving subset of its column.
2. Retrieving subset of its row.

## 1.12 Rules Governing Relational Database

CODD rules and Normalization rules are the two important rules which governs the functions of relational database.

### 1.12.1 Codd Rules

They specify set of rules that relational database must comply in order to be relational.

**Rule 1- The information rule**

All data should be presented to the user in table form. Data are represented only one way; as values within columns within rows

1. The basic requirement of the relational database
2. Simple, consistent and versatile.

**Rule 2- Guarranteed access rule**

All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key and column name.

1. Every value can be accessed by providing table name, column name and key.
2. All data are uniquely identified and accessible via this identity.

**Rule 3- Systematic insert, update and delete**

All fields should be allowed to remain empty. This involves the support of a null value, which is distinct from an empty string or a number with a value of zero. This can't apply to primary keys. In addition, most database implementations support the concept of a nun-null field constraint that prevents null values in a specific table column.

**Rule 4- Dynamic on-line catalogue**

A relational database must provide access through the same tools that are used to access data. This is usually accomplished by storing the structure definition within special system tables. Catalog can be queried by authorized users as part of the database.

**Rule 5- Comprehensive data sublanguage rule**

The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity and database transaction control. All commercial relational databases use forms of the standard SQL as their supported comprehensive language.

1. Used interactively and embedded within programs.
2. Supports data definition, data manipulation, security, integrity constraints and transaction processing.

**Rule 6- View updating rule**

Data can be presented to the user in different logical combinations, called views. Each view should support the same full range of data manipulation that direct-access to a table has available. In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.

**Rule 7- High-level insert, update and delete**

Data can be retrieved from a relational database in sets constructed of data from multiple rows and or multiple tables. This rule states insert, update and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

**Rule 8- Physical data independence**

The user is isolated from the physical method of storing and retrieving information from the database. Change can be made to the underlying architecture without affecting the user access.

**Rule 9- Logical data independence**

Data should not change when logical structure changes. Users and programs are independent of the logical structure of the database.

**Rule 10- Integrity independence**

SQL should support constraints on user input that maintain database integrity. All databases do preserve two constraints through SQL.

**Rule 11-Distribution independence**

A user should be completely unaware of whether or not the database is distributed.

### Rule 12-No subversion rule

There should be no way to modify the database structure other than through the multiple row database language.

### Codd's ZERO rule

The system be able to manage databases entirely through its relational capabilities, no matter what additional capabilities the system may support.

## 1.12.2 Normalization

Normalization is the process of simplifying the relationship between data elements in the records.

| Title | Author name 1 | Author name 2 | ISBN | Subject | Pages | Publishers |
|---|---|---|---|---|---|---|
| PHP and MySQL Web Development | Luke Welling | Laura Thomson | 0672317842 | PHP, MySQL | 867 | Sams |
| MySQL tutorial | Luke Welling | Laura Thomson | 0672317845 | MySQL | 300 | Sams |

This table is not very efficient with storage. The names of the authors are too long, contains many characters. The repetition of their names will occupy more storage space, and there is chance for entering their name with spell mistake. This leads to less efficient searching and gives results missing with some data. Normalization helps in this regard to reduce the redundant data, to improve storage efficiency, data integrity and scalability.

Normalization is a step by step decomposition of complex records into simple records and are carried out

1. To structure the data between tables so that data maintenance is simplified.
2. To allow data retrieval at optimal speed.
3. To simplify data maintenance through updates, inserts and deletes.
4. To reduce the need to restructure tables as new application by rationalization arise.
5. To improve the quality of design for an application by rationalization of table data.

Normalization decomposes data into two dimensional tables, eliminates any relationship in which table data does fully depend upon the primary key of a record and contains transitive dependencies. The database community developed series of guidelines called as normal forms. They are

- First normal form (1 NF)
- Second normal form (2 NF)
- Third normal form (3 NF)
- Fourth normal form (4 NF)
- Fifth normal form (5 NF)
- Domain / key normal form (DKNF)
- Sixth normal form (6 NF)

**First normal form (1 NF)**

The above table contains two (more than one) author field and contains more than one piece of information. This complicates the search operation and First normal form helps in overcoming these problems by modifying the table. 1NF involves the removal of redundant data from horizontal rows and ensures that there is no duplication of data in the table. 1NF creates separate tables for each group of related data and identify each row with a column or set of columns.

**Normalized by 1NF**

| Title | Author | ISBN | Subject | Pages | Publishers |
|---|---|---|---|---|---|
| PHP and MySQL Web Development | Luke Welling | 0672317842 | MySQL | 867 | Sams |
| PHP and MySQL Web Development | Laura Thomson | 0672317842 | PHP | 867 | Sams |
| MySQL tutorial | Laura Thomson | 0672317845 | MySQL | 300 | Sams |
| MySQL tutorial | Luke Welling | 0672317845 | MySQL | 300 | Sams |

In this case author and subject columns are reduced into one. Splitting the table into author table, subject table and book table will improves the normalization process.

**Author ID**

| Author ID | First name | Last name |
|---|---|---|
| 1 | Luke | Welling |
| 2 | Laura | Thomson |

Author table is splitted into two columns in order to store little information in each coloumn. Each table has a primary key, which connects all the tables when querying the data. A primary key must be unique (no two books will be given same ISBN number).

**Subject table**

| Subject ID | Subject |
|---|---|
| 1 | MySQL |
| 2 | PHP |

**Book Table**

| ISBN | Title | Pages | Publisher |
|---|---|---|---|
| 0672317842 | PHP and MySQL Web Development | 867 | Sams |
| 0672317845 | MySQL Tutorial | 300 | Sams |

**Second normal form (2 NF)**

Second normal form deals with redundancy of data in vertical columns. In this table publisher column contains repeating data. Second normal form breaks this table and creates separate publisher information table.

| ISBN | Title | Pages | Publisher |
|---|---|---|---|
| 0672317842 | PHP and MySQL Web Development | 867 | Sams |
| 0672317845 | MySQL Tutorial | 300 | Sams |

**Normalised Table**

| Publisher ID | Publisher |
|---|---|
| 1 | Sams |

1. 2NF meets all the requirements of the first normal form.
2. It removes subsets of data that apply to multiple rows of a table and places them in separate tables.
3. 2NF creates relationships between these new tables and their predecessors through the use of foreign keys.

**Third normal form (3 NF)**

3NF looks for data which is not dependent on the primary key and remove those columns.

## 1.13 Scripting Languages

Every computer architecture has its own 'machine language' such as FORTRAN, C, C++. A disadvantage of compilers is, whenever modification is made to source code, it must be recompiled before it can run. The computer scripts offer an 'alternative strategy', scripting languages, use interpreters instead of compilers. Python, Ruby and PERL are some of the popular scripting languages useful in pharmaco-informatics.

**Script – What you give an actor ?**

**Program – What you give an audience ?**

**PERL - P**ractical **E**xtraction and **R**eport **L**anguage

PERL is a high level but easy to use programming language and available for most operating systems. It makes easy things easier and hard things possible where as professional programming languages make all things equally difficult. Using PERL series of complex tasks can be reduced to a single statement. PERL is preferred for processing sequence analysis and database management.

**Bio-PERL**

A popular tool-kit developed as a collection of integrated PERL modules for transforming and manipulating sequence data and annotations, accession remote databases and parsing output from programs such as BLAST, FASTA etc. Bio-PERL also facilitates local execution of programs from the EMBOSS suite. Bio-PERL modules save time and effort.

**Applications of Bio-PERL**

- Bioperl provides access to sequence data and transforming formats of databases
- Bioperl assists in sequence similarity search
- Bioperl creates and manipulates sequence alignment
- Bioperl is useful in searching structures of genome
- Bioperl develops machine readable annotations

**Sequence analysis:** Parsing and annotation is useful in the analysis of simple patterns and Bioperl provides mechanisms for parsing and running.

**BLAST parsing:** Parsing NCBI, WuBlast, bl2seq and psi-blast can be done through the bioperls modules

- Bio::Tools::Blast
- Bio::Tools::Bplite
- Bio::Tools::BPbl2seq
- Bio::Tools::BPpsilite

**BLAST running:** Running BLAST locally and remotely is built-in to bioperl through the modules

- Bio::Tools::Run::StandAloneBLast
- Bio::Tools::Run::RemoteBlast

**Multiple sequence alignment parsing:** Multiple sequence alignment are also staple of bioinformatics research. Bioperl offers the Bio::AlignIO system for reading and writing MSA reports produced by a variety of sources include ClustalW and CCG.

**ClustalW and TCoffee:** Parsing ClustalW and CCG can be done through the bioperls modules

- Bio::Tools::Run::Alignment::Clustal
- Bio::Tools::Run:: Alignment::TCoffee

**Gene prediction and parsing:** Genscan and Mzef produce Bio:: SeqFeature::GeneStructure for gene prediction and parsing

## 1.14 Structured Query Language (SQL)

Structured Query Language (SQL) has been a command language, provides an interface to relational database systems. In common usage of SQL also encompasses Data Manipulations Language (DML), for INSERTs, UPDATEs, DELETEs and Data Definition Language (DDL), used for creating and modifying tables and other database structures.

A relational database uses, relationally or two-dimensional tables to store different pieces of information inside tables and nothing more. All operations on data are done on the tables themselves or produce other tables as the result. A relational database contains one or many tables, which is a basic storage structure of relational database management system (RDMS). Each row is set of column with only one value for each.

All rows from the same table have the same set of columns. The row from a relational table is analogous to a record, and the columns to a field. Relational database serves in two ways

- Retrieving subset of its column.
- Retrieving subset of its row.

| S. No | Medicine name | Category | Cost | Expiry date |
|---|---|---|---|---|
| | | | | |
| | | | | |

**Properties**

1. It can be accessed and modified by executing Structured Query Language (SQL) statements.
2. It contains a collection of tables with no physical pointers and uses a set of operators.
3. A single row or table representing all data required for a particular medicine, each row in a table should be identified by a primary key, which allows no duplicate key / values.
4. A column or an attribute contains the medicine name.
5. The serial number identifies a medicine in the table. In this example, the serial number column is designated as primary key.
6. A primary key must contain value and the value must be unique.
7. A column containing cost value is a foreign key which defines how table relate to each other.
8. A field may have no value in it, this is called a null value.
9. A field can be found at the intersection of row and column, there can be one value in it.

**General guideline for executing SQL commands**

- SQL commands may be on one / many line.
- Clauses are usually placed on separate lines.
- Tabulation can be used.
- Command words can't be split across lines.
- SQL commands are not case sensitive.
- Place a semi-colon (;) at the end of the last clause.

**SQL features**

- SQL can be used by a range of users, including those with little or no program knowledge.
- It is non procedural language.
- It reduces the amount of time required for creating and maintaining systems

**SQL rules**

1. It starts with a verb, each verb is followed by number of clauses and a space ( ) separates clauses.
2. A comma (**,**) separates parameters without a clause.
3. A semicolon (**;**) is used to end SQL statements.
4. Statement may be split across lines but keywords may not.
5. Lexical units such as identifiers, operator names, literals are separated by one or more spaces or other delimiters that will not be confused with the lexical unit.
6. Reserved words cannot be used as identifiers unless enclosed with double quotes.
7. Identifier can contain up to 30 characters and must start with an alphabetic character.
8. Character and date literals must be enclosed within single quotes.
9. Numeric literals can be represented by simple values.
10. Scientific notation as $2 \times 10^5$.
11. Comments may be enclosed between /*and*/ symbols and may be multi line.
12. Single line comments may be prefixed with a – symbol.

**Components of SQL**

**Data definition language (DDL):** It is set of SQL commands used to create, modify and delete database structures but not data.

Examples:

**Create:** To create objects / tables in the database.

**Alter:** Alter the structure of the database.

**Drop:** Delete objects from the database.

**Truncate:** Remove all records from a table, including all spaces allocated for the records are removed.

**Data Manipulation Language (DML):** It is area of SQL that changing data within the database.

Examples:

**Insert:** Insert records (database) into a table

**Update:** Updates existing data within a table.

**Delete**: Deletes specified records from a table, the space for the records remain.

**Call:** Call a PL/SQL or Java program.

**Explain Plan:** Explain access path to data.

**Lock:** TABLE control concurrency.

**Data Control Language (DCL):** It is the component of SQL statement that control access to data and to the database.

***Examples:***

**Commit:** Save work done

**Save Point:** Identify a point in a transaction to which we can roll back

**Rollback:** Restore database to original since the last COMMIT

**Set Transaction:** Change transaction options like what rollback segment to use.

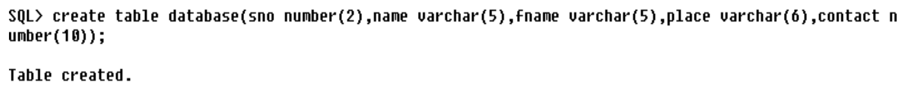
**Grant /Revoke:** Grant or take back permissions to or from the oracle users.

**Data Query Language (DQL):** It is the component of SQL statement that allows retrieving data from the database and imposing ordering upon it.

***Example***

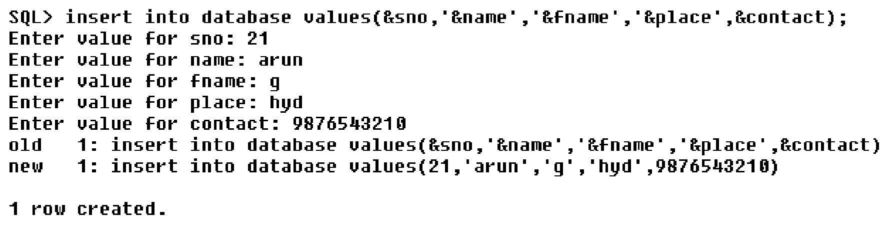
**Select:** Retrieve data from the database.

**Using CREATE command**



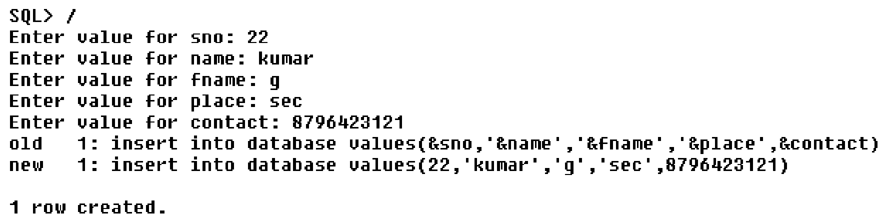
```
SQL> create table database(sno number(2),name varchar(5),fname varchar(5),place varchar(6),contact n
umber(10));

Table created.
```

## Using INSERT command

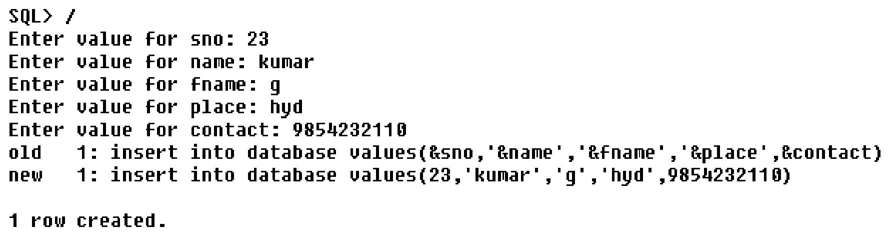


```
SQL> insert into database values(&sno,'&name','&fname','&place',&contact);
Enter value for sno: 21
Enter value for name: arun
Enter value for fname: g
Enter value for place: hyd
Enter value for contact: 9876543210
old   1: insert into database values(&sno,'&name','&fname','&place',&contact)
new   1: insert into database values(21,'arun','g','hyd',9876543210)

1 row created.
```

Using "/" helps adding next set of data (record) without writing new insert command

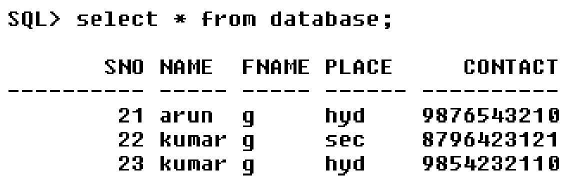


```
SQL> /
Enter value for sno: 22
Enter value for name: kumar
Enter value for fname: g
Enter value for place: sec
Enter value for contact: 8796423121
old   1: insert into database values(&sno,'&name','&fname','&place',&contact)
new   1: insert into database values(22,'kumar','g','sec',8796423121)

1 row created.
```



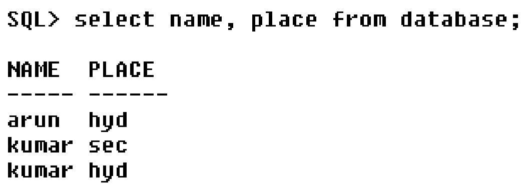
```
SQL> /
Enter value for sno: 23
Enter value for name: kumar
Enter value for fname: g
Enter value for place: hyd
Enter value for contact: 9854232110
old   1: insert into database values(&sno,'&name','&fname','&place',&contact)
new   1: insert into database values(23,'kumar','g','hyd',9854232110)

1 row created.
```

## Using SELECT command



```
SQL> select * from database;

       SNO NAME  FNAME PLACE     CONTACT
---------- ----- ----- ------ ----------
        21 arun  g     hyd    9876543210
        22 kumar g     sec    8796423121
        23 kumar g     hyd    9854232110
```

## Using SELECT command



```
SQL> select name, place from database;

NAME  PLACE
----- ------
arun  hyd
kumar sec
kumar hyd
```

## Using DISTINCT command

```
SQL> select distinct place from database;

PLACE
------
hyd
sec
```

## Using AND command

```
SQL> select * from database where name='kumar' and fname='g';

       SNO NAME  FNAME PLACE     CONTACT
---------- ----- ----- ------ ----------
        22 kumar g     sec    8796423121
        23 kumar g     hyd    9854232110
```

## Using OR command

```
SQL> select * from database where name='kumar' or fname='m';

       SNO NAME  FNAME PLACE     CONTACT
---------- ----- ----- ------ ----------
        22 kumar g     sec    8796423121
        23 kumar g     hyd    9854232110
        24 mani  m     hyd    9876453423
```

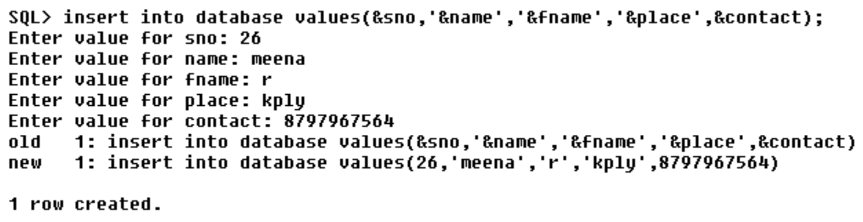
## Using DELETE command

```
SQL> delete from database where sno=23;

1 row deleted.

SQL> select * from database;

       SNO NAME  FNAME PLACE     CONTACT
---------- ----- ----- ------ ----------
        21 aruna g     hyd    9876543210
        22 kumar g     sec    8796423121
        24 mani  m     hyd    9876453423
        25 msdhu g     bply   8746453533
```

## Using TRUNCATE command

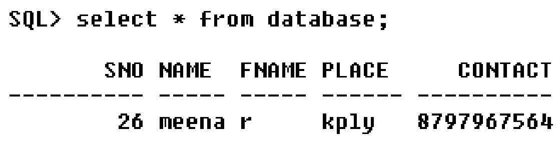
```
SQL> truncate table database;

Table truncated.

SQL> select * from database;

no rows selected
```

Truncate command removes data only and retains the constructed table, so that data can be added further.



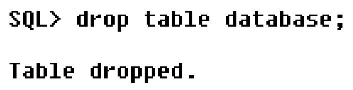
```
SQL> insert into database values(&sno,'&name','&fname','&place',&contact);
Enter value for sno: 26
Enter value for name: meena
Enter value for fname: r
Enter value for place: kply
Enter value for contact: 8797967564
old   1: insert into database values(&sno,'&name','&fname','&place',&contact)
new   1: insert into database values(26,'meena','r','kply',8797967564)

1 row created.
```

### Using SELECT command



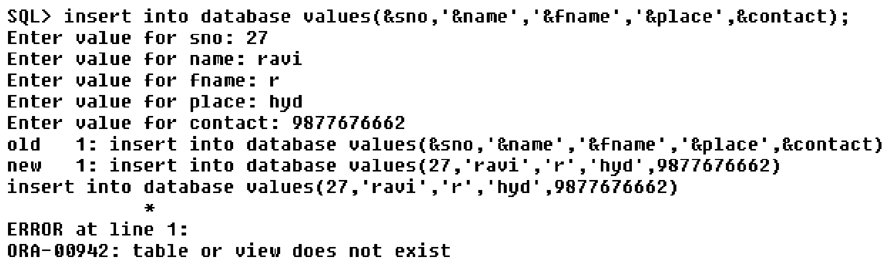
```
SQL> select * from database;

       SNO NAME  FNAME PLACE     CONTACT
---------- ----- ----- ------ ----------
        26 meena r     kply   8797967564
```

### Using DROP command



```
SQL> drop table database;

Table dropped.
```

Drop command completely removes the constructed table along with stored data.



```
SQL> insert into database values(&sno,'&name','&fname','&place',&contact);
Enter value for sno: 27
Enter value for name: ravi
Enter value for fname: r
Enter value for place: hyd
Enter value for contact: 9877676662
old   1: insert into database values(&sno,'&name','&fname','&place',&contact)
new   1: insert into database values(27,'ravi','r','hyd',9877676662)
insert into database values(27,'ravi','r','hyd',9877676662)
            *
ERROR at line 1:
ORA-00942: table or view does not exist
```