

Using MySQL

Session 4



Objectives

- ◆ *Explain database*
- ◆ *Explain the data types*
- ◆ *Identify the different types of data*
- ◆ *Explain the creation of a table*
- ◆ *Explain normalization*
- ◆ *Identify the different forms of normalization*
- ◆ *Explain indexes and referential integrity*

- ◆ A database is a collection of organized and related data
- ◆ Databases store information in an organized manner in the form of a table
- ◆ A database management system can be used to handle the data in a database
- ◆ A database can contain duplicate records in tables if the database is not normalized and tables are not indexed
- ◆ You can normalize the database and index the tables to eliminate redundancy and enhance the speed of data search and retrieval operations

- ◆ While naming a database and its components, such as tables, columns, and so forth you will have to follow certain rules

For Aptech Center Use Only

- ◆ The following naming conventions are complied by MySQL:

- ◆ **Legal Characters –**

- ◆ You can use any alphabet (a-z or A-Z) or digits (0-9) in the name
- ◆ A name can start with a digit but cannot contain only digits because you cannot distinguish it from a number
- ◆ You cannot use `\.`, `\/` or `\ \` within the name because it acts as the separator

◆ **Length of names –**

- ◆ MySQL restricts the length of the name of the databases, indexes, or columns
- ◆ It can have a maximum length of 64 characters
- ◆ MySQL allows alias names to be 256 characters in length and compound statement labels are restricted to 16 characters

- ◆ You can enter SQL commands on a single line or split them across multiple lines
- ◆ All MySQL commands should be terminated by a semicolon
- ◆ MySQL waits for a semicolon (;) to execute a command
- ◆ Alternatively, you can use the '\g' command

- ◆ In MySQL, case sensitivity varies with the object in question

- ◆ **SQL Statements and Keywords**

- ◆ The SQL statements and keywords are not case sensitive
- ◆ A user can write SQL statements in upper or lower case
- ◆ Consider the statements as shown:

```
SELECT VERSION ( );
```

```
select version( );
```

- ◆ Both the statements are equivalent
- ◆ Both the commands will return the version of MySQL being used
- ◆ The following combinations are also valid:

```
Select VERSION ( );
```

```
SeLeCT version ( );
```


◆ Database and Table names

- ◆ Case sensitivity of databases, tables, and trigger names depend on the operating system
- ◆ If the underlying operating system is case sensitive, then the names of databases, tables, and triggers
- ◆ For example, Microsoft Windows is not case sensitive and so databases, tables, and triggers names in MySQL running on Microsoft Windows are not case-sensitive
- ◆ Linux is case sensitive and so databases, tables, and triggers names under a Linux installation of MySQL are case-sensitive
- ◆ This is because these objects are stored as files on the file system

◆ Columns, Column Aliases, Index, Stored Routines, and Events Names

- ◆ Names of columns, column aliases, index, stored routines, and events are not case sensitive
- ◆ You can use one of the following SQL statements to display the column 'FNAME' from a SAMPLE table:

```
SELECT FNAME from SAMPLE ( );
```

```
SELECT fname from SAMPLE ( );
```

◆ Log Files

- ◆ Log file names are case sensitive

- ◆ MySQL is a Relational Database Management System
- ◆ A relational database consists of different tables in which data is stored such that there is minimal redundancy
- ◆ A relational database also contains relationships or links to related tables
- ◆ The syntax for creating a database is:

```
CREATE DATABASE [IF NOT EXISTS] <dbname>;
```

where,

CREATE DATABASE – adds a new database to the MySQL instance

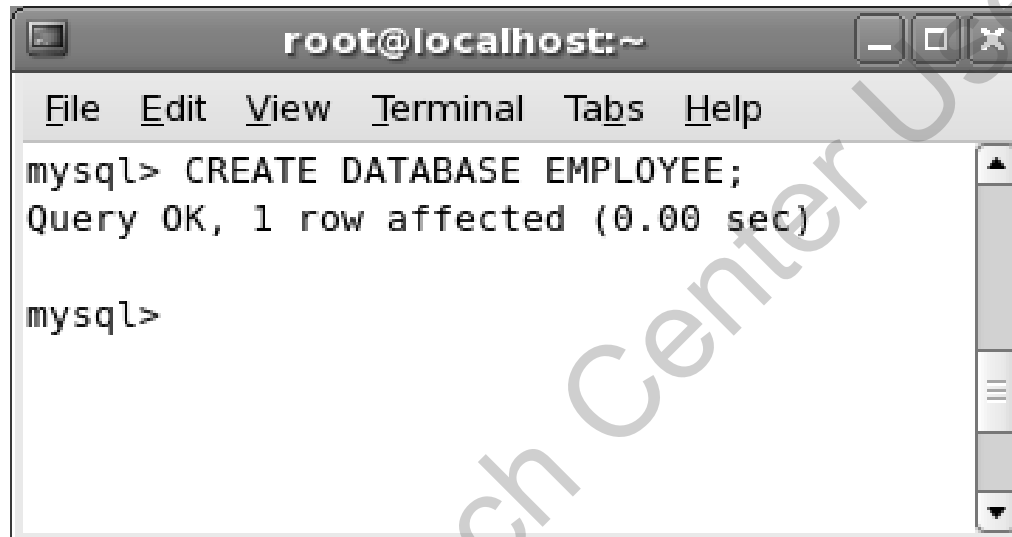
IF NOT EXISTS – checks for the existence of databases with the same name before adding the new database

dbname – specifies a name for the new database

- ◆ For example, to create a database named `EMPLOYEE` for storing employee information, such as personal details, qualifications, salary details, and so on, enter the following command at the command prompt:

```
CREATE DATABASE EMPLOYEE;
```

Figure displays the number of rows affected, and the time taken to execute the command

A terminal window titled 'root@localhost:~' with standard window controls. The menu bar includes 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The terminal text shows a MySQL prompt 'mysql>' followed by the command 'CREATE DATABASE EMPLOYEE;'. The response is 'Query OK, 1 row affected (0.00 sec)'. Another 'mysql>' prompt is shown below. A large, light-gray watermark 'For Apteck Center Use Only' is diagonally across the image.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> CREATE DATABASE EMPLOYEE;  
Query OK, 1 row affected (0.00 sec)  
  
mysql>
```

Working with MySQL Tables

- ◆ As compared to a DBMS, RDBMS stores data in different tables to ensure consistency and faster search and retrieval operations
- ◆ A table stores data in rows and columns
- ◆ A field in a table is called a column and row is called a record
- ◆ A record is also defined as collection of fields

- ◆ Data types define the type of the data that will be stored in a column
- ◆ Data types also define the size and the type of data that can be stored in the column
- ◆ MySQL supports different data types to store values
- ◆ Following are the different categories under which the data can be categorized:
 - ◆ Numeric data types
 - ◆ Date and time data types
 - ◆ String data types
 - ◆ Complex data types

◆ Numeric Data Type

- ◆ A numeric data type stores data in number form
- ◆ There are several numeric data types available in MySQL

Table lists the numeric data types in MySQL

Numeric Data Type	Storage In Bytes	Description	Synonyms
BIGINT (length)	8	Stores unsigned numbers in the range of 0 to 18,446,744,073,709,551,615, and signed numbers in the range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	DEC NUMERIC
DECIMAL (length, decimal)	Precision +2	Stores floating point numbers and cannot be unsigned	

DOUBLE	8	Stores double precision floating point numbers. Range for a negative number is between -1.7976931348623157E+308 and -2.2250738585072014E-308,0. Range for a positive number is between 2.2250738585072014E-308 and 1.7976931348623157E+308	DOUBLE PRECISION REAL
FLOAT(length, decimal)	4	Stores single precision floating point numbers. Range for a negative number is between -3.402823466E+38 and -1.175494351E-38,0. Range for a positive number is between 1.175494351E-38 and 3.402823466E+38	
INT(length)	4	Stores integer numbers. Range for a signed number is from -2,147,483,648 to 2,147,483,647. Range for an unsigned number is from 0 to 4,294,967,295	INTEGER

MEDIUMINT (length)	3	Stores integer numbers. Range for a signed number is from -8,388,608 to 8,388,607. Range for an unsigned number is from 0 to 16,777,215	
SMALLINT (length)	2	Stores integer numbers. Range for a signed number is from -32,768 to 32,767. Range for an unsigned number is from 0 to 65,535	
TINYINT	1	Stores integer numbers. Range for a signed number is from -128 to 127. Range for an unsigned number is from 0 to 255	BOOLEAN

◆ String Data Type

- ◆ A string represents a sequence of characters
- ◆ A string data type is enclosed in single quotes or double quotes
- ◆ It enables you to enter alphabets from a-z, A-Z, and numbers from 0-9

Table lists the string data types supported by MySQL

Data Type	Storage In Bytes	Description
CHAR (Length)	Length	Stores character data set and can have a maximum length of 255 characters
BINARY (Length)	Length	Stores binary byte string upto 255 characters
VARCHAR ()	Length+1	Stores variable length string values and can have a length from 0 to 65,535. Uses one byte to store 255 characters. Uses additional two bytes to if values require more than 255 characters
VARBINARY	Length+2	Stores binary strings and has a maximum length same as that of VARCHAR

BLOB	Length+2	Stores large amount of variable binary data such as images and can have a maximum length of 65,535. Uses an additional byte of storage if values exceed 65,536
TINYBLOB	Length+1	Stores upto 255 bytes
MEDIUMBLOB	Length+3	Stores text in size upto 16 MB. Uses additional three bytes of storage if values exceed the specified length.
LONGBLOB	Length+4	Stores text values whose size exceeds 4 GB. Uses additional four bytes of storage if values exceed the specified length
TEXT	Length+2	Stores characters and can be upto 65,535 characters in length. Uses additional two bytes of storage if values exceed the specified length
TINYTEXT	Length+1	Stores short text values and can have a length upto 255 characters
MEDIUMTEXT	Length+3	Stores medium-sized text whose size is upto 16 MB. Uses additional three bytes of storage if values exceed the specified length.
LONGTEXT	Length+4	Stores large text values whose size exceeds 4 GB. Uses additional four bytes of storage if values exceed the specified length

◆ Date Data Type

- ◆ A date data type stores date and time information
- ◆ A date data type column stores date as string value in the default format, 'YYYY-MM-DD'

Table lists the date data types in MySQL

Numeric Data Type	Format	Storage In Bytes	Description
DATE	YYYY-MM-DD	3	Stores a date type of data in the range of January 1,1000 to December 31,9999
DATETIME	YYYY-MM-DD hh:mm:ss	8	Stores date and time. The range for it is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIME	hh:mm:ss	3	Stores time
TIMESTAMP	YYYY-MM-DD hh:mm:ss	4	Stores timestamp
YEAR	YYYY	1	Stores a year in the range of 1901 to 2155

◆ Complex Data Type

- ◆ Complex data type is a special data type
- ◆ It defines the list of possible values that can be inserted in a column

Table lists the complex data types in MySQL

Complex Data Type	Storage In Bytes	Description
ENUM	1,2	Stores one value out of a possible 65535 number of options in a predefined list of possible values. This data type can contain only one distinct string value from the predefined list of values. The values are represented internally as integers. Eg: ENUM('abc','def','ghi')
SET	1,2, 3, 4 or 8	Stores a list of values from a predefined set of values. Maximum of 64 values can be stored in a SET data type column. The SET data type can contain zero or more values that can be specified in the list.

- ◆ The SET data type is similar to ENUM
- ◆ While an ENUM data type restricts you to selecting one of many predefined values, SET data type allows you to choose more than one of the predefined values

- ◆ Consider the following example:

```
CREATE TABLE MY_CITY  
(CITY_ID INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
CITY_NAME ENUM('CHICAGO','DALLAS','COLUMBUS'));
```

- ◆ The code will create a table MY_CITY with two columns, CITY_ID and CITY_NAME
- ◆ The data type for CITY_NAME has been defined as ENUM
- ◆ When you add records to the table using the INSERT command, this data type will allow you to use one of the city names specified in the ENUM list as the column value when you add rows to the table

- ◆ Consider the same code with SET as the data type for the CITY_NAME field

```
CREATE TABLE MY_CITY  
(CITY_ID INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
CITY_NAME SET ( 'CHICAGO' , ' DALLAS' , 'COLUMBUS' ) ) ;
```

- ◆ When you assign SET data type to the CITY_NAME field, you can use one or more city names from the SET list as a column value when you add rows to the table

- ◆ The `SET` data type uses storage space in the form of bytes to store the number of elements defined in the data type

Table lists the storage structure of `SET` data type depending upon the number of elements

Number Of Elements	Storage In Bytes
1-8	1
9-16	2
17-24	3
25-32	4
33-64	8

- ◆ A relational database contains several related tables
- ◆ The maximum length of a field name is 64 characters
- ◆ MySQL supports two types of tables:
 - ◆ **Transaction-safe tables:** enables you to execute transactions or data manipulation commands without losing data. You can undo the changes made to transaction-safe tables
 - ◆ **Non-transaction safe tables:** also allows you to execute transaction or data manipulation commands. However, you cannot undo the changes made to non-transaction safe tables because the changes are permanent

- ◆ Following are the features of transaction-safe tables:
 - ◆ Provides safety from data loss and allows MySQL to recover data, from a backup and the transaction log, in case of a crash or hardware failure
 - ◆ Enables to combine and execute statements using the `COMMIT` command
 - ◆ Enables to undo the changes made to the data using the `ROLLBACK` command
 - ◆ Enables concurrency while simultaneously reading data from tables
 - ◆ Examples of transaction-safe tables are InnoDB

- ◆ Following are the features of non-transaction safe tables:
 - ◆ Faster than the transaction-safe tables as no transaction overhead is required
 - ◆ Require less disk space
 - ◆ Require less memory for updates

◆ Storage Engines

- ◆ MySQL supports storage engines for different table types
- ◆ Storage engines handle both transaction-safe and non-transaction safe tables

Table lists the storage engines supported in MySQL:

Storage Engine	Feature
MyISAM	Is the default storage engine to handle tables
InnoDB	Provides commit, rollback, and crash recovery characteristics to secure data
MERGE	Allows developers to create tables with identical column and index information
MEMORY (HEAP)	Allows developers to create tables that store contents in the memory. MEMORY tables can have upto 64 indexes per table, 16 columns per index, and a maximum key length of 3072 bytes
EXAMPLE	Allows developers to create dummy tables where no data will be stored
FEDERATED	Provides access to data from tables in remote databases without copying data to the local server. This engine does not create a replica or a copy of the database or tables to the local server while executing a query on the remote server
ARCHIVE	Allows developers to store large amount of data without indexing
CSV	Allows developers to store data in text files using comma-separated values
BLACKHOLE	Allows developers to accept data without storing. Tables that have this storage engine will always return an empty result. This storage engine can be used in distributed database design where data is automatically replicated, but not stored locally

- ◆ You will use the `CREATE` command to add a new table to a database
- ◆ **The syntax for `CREATE` command is:**

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
(create_clause,...) [table_options] [IGNORE | REPLACE]  
[AS] Select ...];
```

where,

`CREATE TABLE` – adds a new table to the database

`tbl_name` – specifies a name for the table

Table lists the description for the options in the CREATE TABLE syntax:

Option	Description
TEMPORARY	Table exists as long as the current user is connected
IF NOT EXISTS	Creates table only if it does not exists in the database
create_clause	Defines column
table_options	Creates different types of tables such as ISAM, INNODB, and so on
IGNORE	If table already exists system ignores and does not replace the previous table
REPLACE	Replaces the table if a table with the same name exists in the database
Select	Enables to copy records from an existing table

- ◆ The syntax for create_clause in the **CREATE TABLE** command is:

```
column type [NOT NULL | NULL] [DEFAULT  
value] [AUTO_INCREMENT] [PRIMARY KEY] [REFERENCE] ;
```

where,

column type – defines the data type for the column

Table lists the options in the create_clause:

Option	Description
AUTO_INCREMENT	Specifies that the column value must be auto incremented. This option works only if the columns contain positive values
DEFAULT value	Inserts the specified value when no value for that column is entered. The default value has to be a constant. The values cannot be a function or expression
NOT NULL	Specifies that a column cannot contain a null value. When a null value is inserted, the system prompts with an error message
NULL	Specifies that the column can contain null values
PRIMARY KEY	Defines a column as the primary key while creating a table
REFERENCES	Assigns the InnoDB storage engine

- ◆ The syntax for REFERENCE options is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
REFERENCE tbl_name[(index_col_name,...)] [MATCH FULL  
|MATCH PARTIAL| MATCH SIMPLE] [ON DELETE  
reference_option] [ON UPDATE reference_option];
```

where,

CREATE TABLE – adds a new table to the database

tbl_name – specifies a name for the table

REFERENCE – defines a relationship between the index and another table

tbl_name – specifies the name for the referenced table

Index_col_name – specifies the column to be indexed

Table lists the options in the REFERENCE clause:

Option	Description
index_col_name	Defines an index on the column
ON DELETE	Used for cascading tables
ON UPDATE	Used for cascading tables

- ◆ Consider an example of a customer table with columns, such as `customer_id` and `name` and the table is assigned to a storage engine, **INNODB**

```
CREATE TABLE customer (customer_id INT NOT NULL, name CHAR  
(15)) ENGINE = INNODB;
```

- ◆ You will create three tables namely `EMP_DETAILS`, `EMP_DEPARTMENT`, and `SALARY_DETAILS` in the **EMPLOYEE** database
- ◆ The data stored in the three respective tables are as follows:
 - ◆ **EMP_DETAILS:** Stores the personal details of the employees, such as identification number, name, address, and phone number
 - ◆ **EMP_DEPARTMENT:** Stores the department information, such as identification number, department, date of joining, and designation of the employee
 - ◆ **SALARY_DETAILS:** Stores the salary information of the employee, such as identification number, basic salary, house rent allowance, traveling allowance, and gross salary

- ◆ You will first have to select the database to create the tables
- ◆ You will select the `EMPLOYEE` database with the help of `USE` command
- ◆ **The syntax for `USE` command is:**

```
USE DATABASE;
```

where,

`USE` – switches to the specified database

`DATABASE` - specifies the name of the database

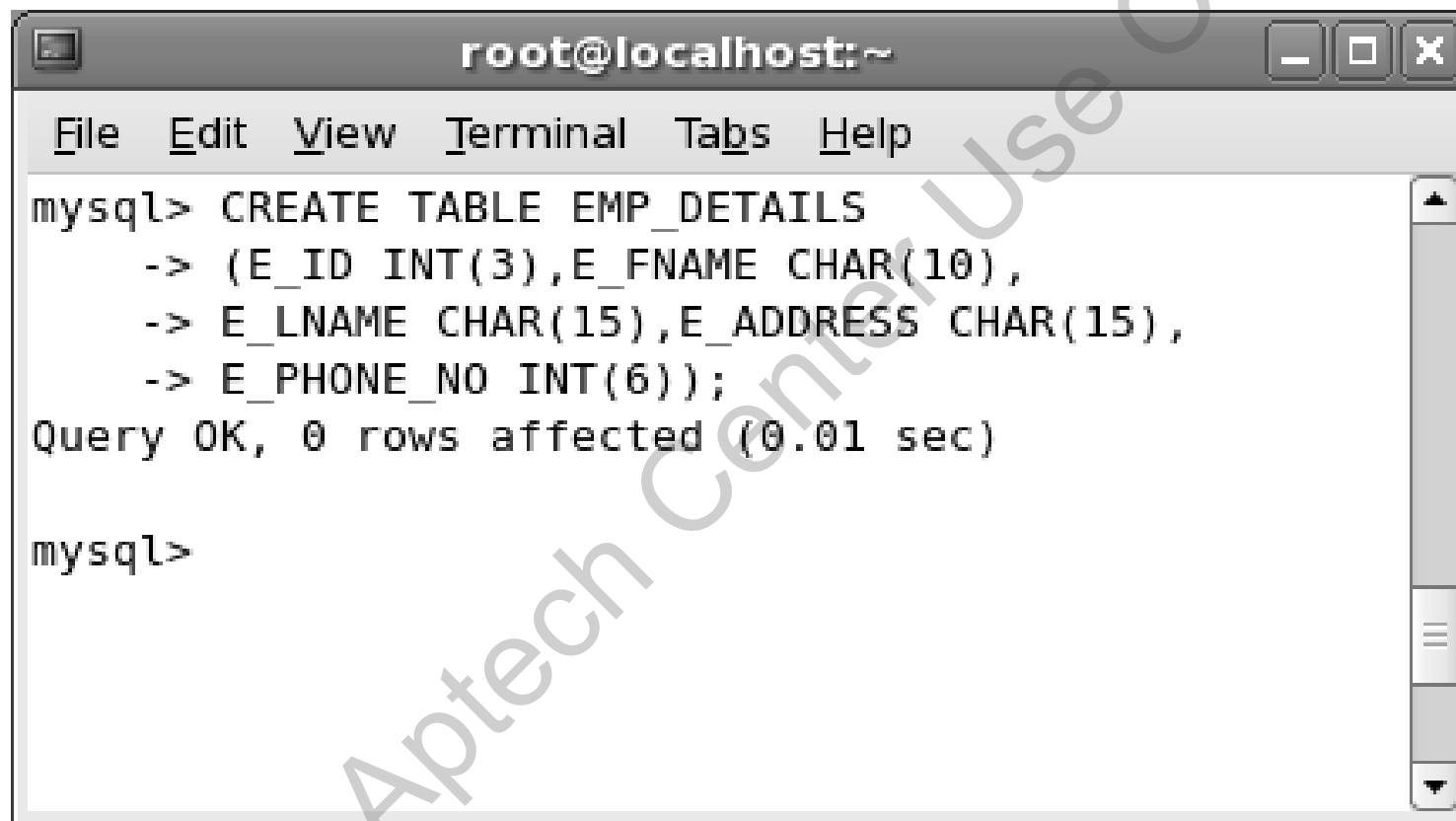
Create an **EMP_DETAILS** table with the structure listed in the table

Column Name	Data Type	Length
E_ID	INT	3
E_FNAME	CHAR	10
E_LNAME	CHAR	15
E_ADDRESS	CHAR	15
E_PHONE_NO	INT	6

- ◆ To create the EMP_DETAILS table, enter the following command at the command prompt:

```
CREATE TABLE EMP_DETAILS (E_ID INT(3), E_FNAME CHAR(10),  
E_LNAME CHAR(15), E_ADDRESS CHAR(15), E_PHONE_NO INT(6));
```

Figure displays the output of the command

A screenshot of a terminal window titled 'root@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The terminal content shows a MySQL command to create a table named EMP_DETAILS with columns E_ID, E_FNAME, E_LNAME, E_ADDRESS, and E_PHONE_NO. The command is executed successfully, and the output indicates that 0 rows were affected in 0.01 seconds. The prompt 'mysql>' is shown at the bottom.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> CREATE TABLE EMP_DETAILS  
-> (E_ID INT(3),E_FNAME CHAR(10),  
-> E_LNAME CHAR(15),E_ADDRESS CHAR(15),  
-> E_PHONE_NO INT(6));  
Query OK, 0 rows affected (0.01 sec)  
  
mysql>
```

- ◆ MySQL provides the `DESCRIBE` command to view the structure of a table
- ◆ The syntax for the **DESCRIBE** command is:

```
DESCRIBE table [column];
```

Or

```
DESC table [column];
```

where,

`DESCRIBE` – displays the field structure of the table

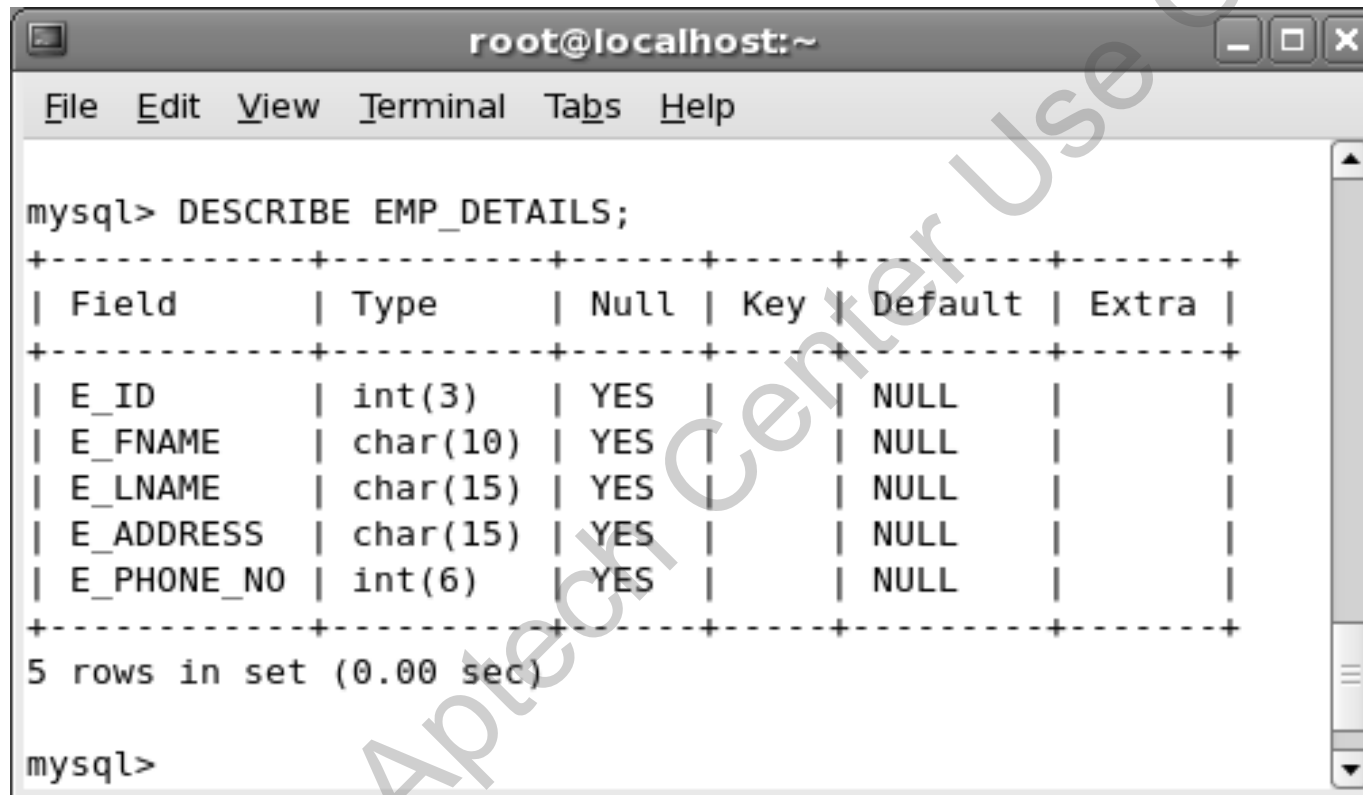
`table` – specifies the name of the table to display the structure

`column` – displays information for a specific column of the table

- ◆ To view the structure of the EMP_DETAILS table, enter the following command at the prompt:

```
DESC EMP_DETAILS;
```

Figure displays the output of the DESCRIBE command



```

mysql> DESCRIBE EMP_DETAILS;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| E_ID       | int(3)    | YES  |     | NULL    |       |
| E_FNAME    | char(10)  | YES  |     | NULL    |       |
| E_LNAME    | char(15)  | YES  |     | NULL    |       |
| E_ADDRESS  | char(15)  | YES  |     | NULL    |       |
| E_PHONE_NO | int(6)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
    
```


- ◆ Figure displays the following information for a table:
 - ◆ Columns in the table
 - ◆ Data type used for a column
 - ◆ Acceptance of null values for a column.
 - ◆ Type of key such as Primary key
 - ◆ Default value for a column
 - ◆ Extra information or unique characteristics of a column
 - ◆ Number of rows in the table

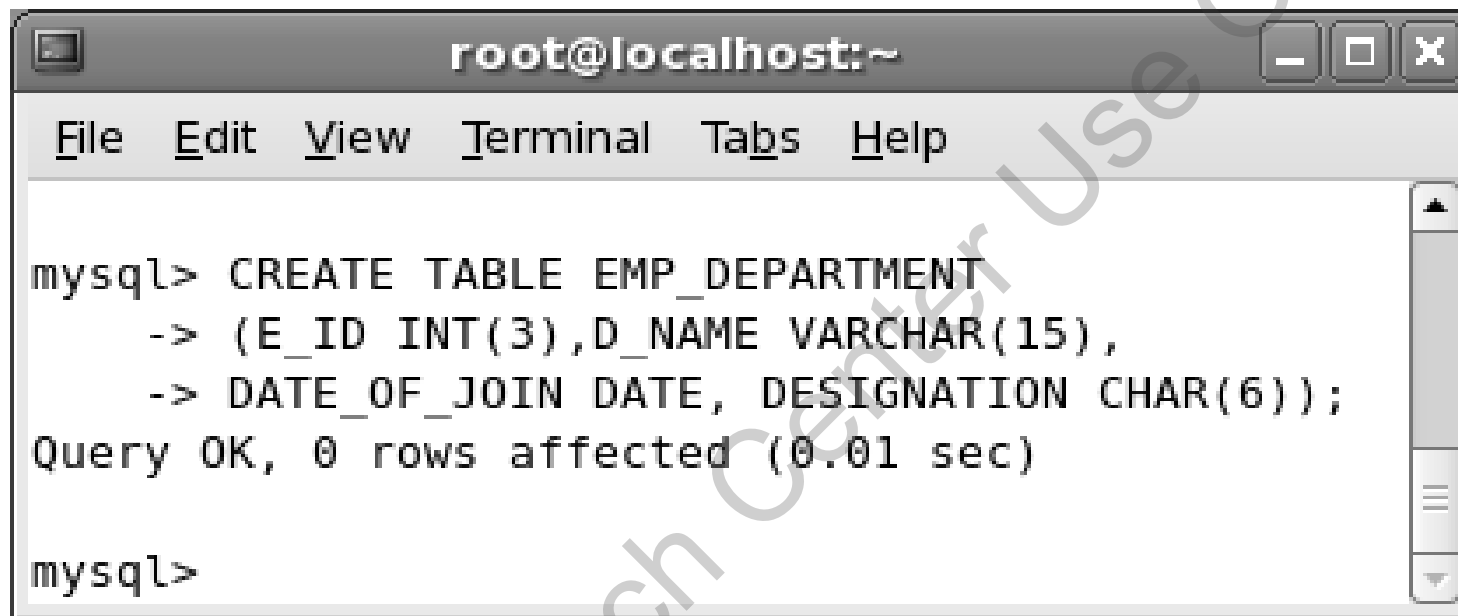
You will create an EMP_DEPARTMENT table with the structure listed in the table

Column Name	Data Type	Length
E_ID	INT	3
D_FNAME	VARCHAR	15
DATE_OF_JOIN	DATE	
DESIGNATION	CHAR	6

- ◆ To create the EMP_DEPARTMENT table, enter the following command at the command prompt:

```
CREATE TABLE EMP_DEPARTMENT (E_ID INT(3), D_NAME  
VARCHAR(15), DATE_OF_JOIN DATE, DESIGNATION CHAR(6));
```

Figure displays the output of the command



The image shows a terminal window titled 'root@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The terminal content shows the following commands and output:

```
mysql> CREATE TABLE EMP_DEPARTMENT  
      -> (E_ID INT(3), D_NAME VARCHAR(15),  
      -> DATE_OF_JOIN DATE, DESIGNATION CHAR(6));  
Query OK, 0 rows affected (0.01 sec)  
  
mysql>
```

A large, diagonal watermark reading 'For Aptech Center Use Only' is overlaid across the terminal window.

Create a `SALARY_DETAILS` table with the structure listed in table

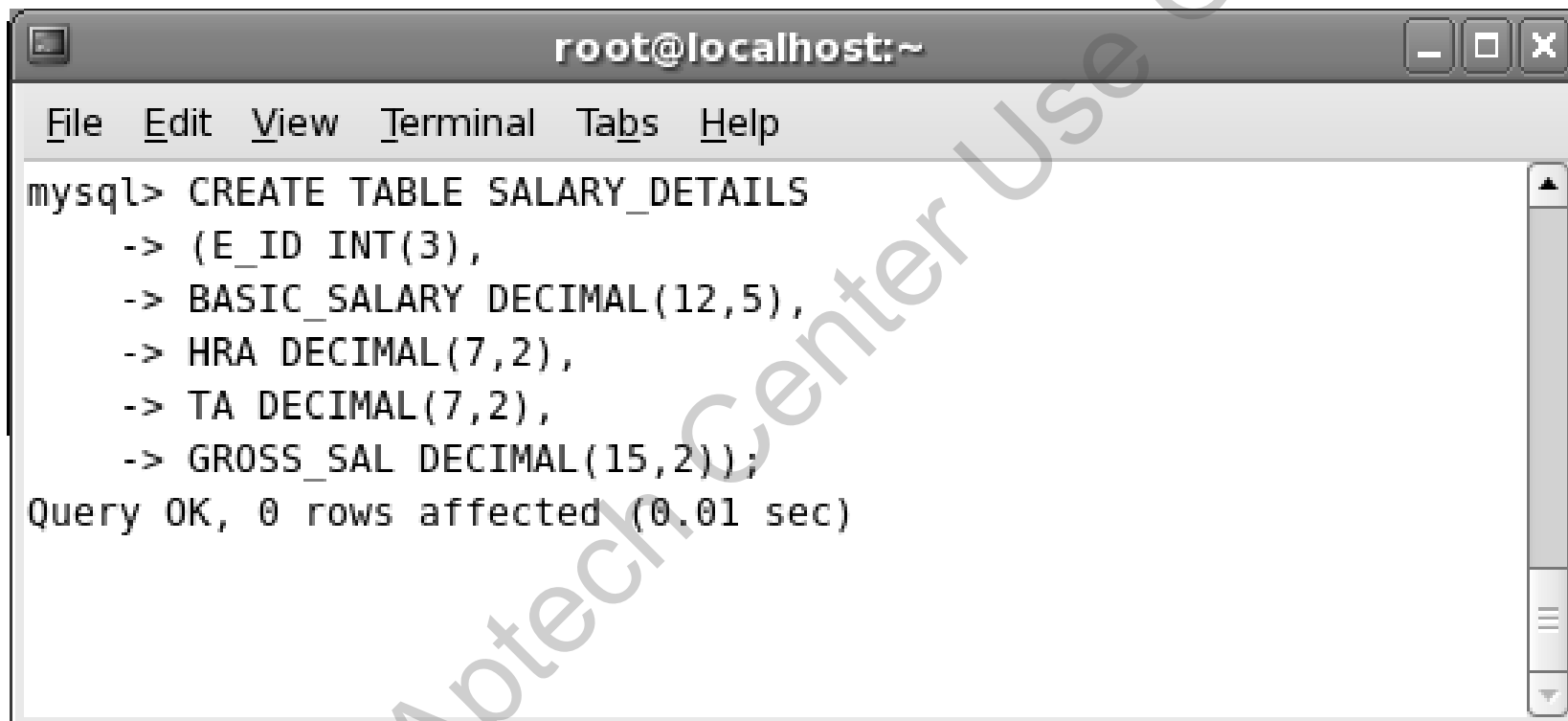
Column Name	Data Type	Length
E_ID	INT	3
BASIC_SAL	DECIMAL	(12,2)
HRA	DECIMAL	(7,2)
TA	DECIMAL	(7,2)
GROSS_SAL	REAL	(15,2)

- ◆ In the table the data type of `BASIC_SAL` field is `DECIMAL`
- ◆ The length of the field has been specified as (12, 2)
- ◆ The value before the comma in the bracket defines the total length of the column including the decimal point
- ◆ The value after the comma specifies the number of digits after the decimal point

- ◆ To create a SALARY_DETAILS table, enter the following command at the command prompt:

```
CREATE TABLE SALARY_DETAILS (E_ID INT(3), BASIC_SALARY  
DECIMAL(12,5), HRA DECIMAL(7,2), TA DECIMAL(7,2),  
GROSS_SAL DECIMAL(15,2));
```

Figure displays the output of the command

A screenshot of a terminal window titled 'root@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The terminal content shows a MySQL command to create a table named 'SALARY_DETAILS' with five columns: 'E_ID' (INT(3)), 'BASIC_SALARY' (DECIMAL(12,5)), 'HRA' (DECIMAL(7,2)), 'TA' (DECIMAL(7,2)), and 'GROSS_SAL' (DECIMAL(15,2)). The command is entered as 'mysql> CREATE TABLE SALARY_DETAILS' followed by five lines of column definitions, each preceded by a hyphen and an arrow. The final output is 'Query OK, 0 rows affected (0.01 sec)'. A large, diagonal watermark 'For Aptech Center Use Only' is overlaid across the terminal window.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> CREATE TABLE SALARY_DETAILS  
-> (E_ID INT(3),  
-> BASIC_SALARY DECIMAL(12,5),  
-> HRA DECIMAL(7,2),  
-> TA DECIMAL(7,2),  
-> GROSS_SAL DECIMAL(15,2));  
Query OK, 0 rows affected (0.01 sec)
```


- ◆ Normalization is the process of structuring the data in a table to minimize duplication and inconsistency
- ◆ Normalization requires breaking down of a larger table into two or more smaller tables
- ◆ Normalization improves the clarity in a database

- ◆ You must understand the following terms before normalizing a database:
 - ◆ **Entity** - represents an object about which you want to store the information
 - ◆ **Attributes** - represents the component of entity type or identifying qualities
 - ◆ **Domain** - defines the range for the attributes
 - ◆ **Relationship** - represents association between entities

- ◆ Keys help to establish a relationship between columns
- ◆ You can define a field as a key
- ◆ These keys uniquely identify a record

For Aptech Center Use Only

- ◆ The different types of keys include:
 - ◆ **Primary Key** - Is used as the unique identifier and has only one attribute
 - ◆ **Composite Key** - Is a primary key having more than one attribute
 - ◆ **Foreign Key** - Is a column or a set of column in a table which refers to a column or a set of column in another table

- ◆ To normalize a database, you will use a sample table where you store department number, department name, employee identification number, employee name, employee category, and wages of the employee per hour

Table lists the data stored in a sample table for normalization

D_No	D_Name	E_Id	E_Name	Emp_Category	Wages
10	Research	41	Charles	I	100
		46	Jack	II	60
		50	Peter	III	40
20	Marketing	51	Mike	I	100
		71	George	III	40

- ◆ You can apply different forms of normalization on a database to remove redundancy and improve clarity
- ◆ You must follow certain set of rules while normalizing the database so that the database contains a minimum number of duplicate records

◆ First Normal Form or 1NF:

- ◆ An entity is said to be in the first normal form, if all the attributes are single valued
- ◆ The following are the rules for a table to be in the first normal form:
 - ◆ **All attributes are atomic:** Field holds a value for one object
 - ◆ **All columns have only one instance:** Columns must be defined to contain a specific type of record

Table displays the sample table structure after applying the First Normal Form

D_No	D_Name	E_Id	E_Name	Emp_Category	Wages
10	Research	41	Charles	I	100
10	Research	46	Jack	II	60
10	Research	50	Peter	III	40
20	Marketing	51	Mike	I	100
20	Marketing	71	George	III	40

◆ Second Normal Form or 2NF:

- ◆ An entity is in the second normal form, if it is in the first normal form and all non key attributes entirely depend on the unique identifier of the entity
 - ◆ After normalizing the sample table using the first normal form, the value for D_NO repeats several times
 - ◆ the field D_NAME depends only on the D_NO field which is part of the primary key and not the entire primary key
- ◆ You will apply the second normal form and split the tables

EMP_DEPARTMENT TABLE

D_No	D_Name	E_Id	E_Name	Emp_Category	Wages
10	Research	41	Charles	I	100
10	Research	46	Jack	II	60
10	Research	50	Peter	III	40
20	Marketing	51	Mike	I	100
20	Marketing	71	George	III	40

DEPARTMENT TABLE

D_No	D_Name
10	Marketing
20	Research

EMPLOYEE TABLE

D_No	E_Id
10	41
10	50
10	46
20	51
20	70

◆ Third Normal Form or 3NF:

- ◆ An entity is in the third normal form, if it is already in the second normal form and no non-identifying attributes are dependent on any other non-identifying attributes

Consider the following data in the EMPLOYEE table, as shown in the table

E_Id	E_Name	Emp_Category	Wages
41	Charles	I	100
46	Jack	II	60
50	Peter	III	40
51	Mike	I	100
71	George	III	40

- ◆ In table, the employee Mike is of EMP_CATEGORY I and the wages should be 100 instead of 60
- ◆ All the attributes should depend only on the primary key according to the third normal form
- ◆ You will split the table, as shown in the following tables

EMPLOYEE table

E_Id	E_Name	Emp_Category
41	Charles	I
46	Jack	II
50	Peter	III
51	Mike	I
71	George	III

RATE table

Emp_Category	Wages
I	100
II	60
III	40
I	100
III	40

◆ **Boyce-Codd Normal Form or BCNF**

- ◆ An entity is said to be in the Boyce-Codd normal form, if it is already in the third normal form and contains only one unique identifier

- ◆ Referential integrity is a feature that prevents you from storing inconsistent data
- ◆ You implement referential integrity when you define a relationship between two tables in a database
- ◆ You can use indexing on databases to make search operations faster
- ◆ You index the fields in a table, on which the search is to be performed
- ◆ You can index several fields in a table at a time

- ◆ Indexes are lists that specify the location of data in a table within a database
- ◆ Indexes support faster data storage and retrieval capabilities
- ◆ Indexes in a database are similar to the indexes in books
- ◆ Indexes in a database use the same concept
- ◆ You can define indexes on a field in a table within a database

- ◆ The disadvantages of indexes are:
 - ◆ It uses more storage space
 - ◆ Data manipulation commands like insert, update, and delete require longer duration to execute
- ◆ You can create an index in a table by indexing the fields
- ◆ The primary key serves as the index key when the index key is not specified explicitly

- ◆ You must remember the following points while indexing fields in a table:
 - ◆ Fields should have unique values
 - ◆ Fields on which the table is to be sorted
 - ◆ Fields should have data types such as text, number, date, or time

- ◆ You can use referential integrity to maintain consistency of records in different tables
- ◆ Referential integrity can be defined by linking a foreign key of a table to the primary key of another

- ◆ Consider the following records stored in the EMP_DETAILS and EMP_DEPARTMENT table

EMP_DETAILS

E_ID	E_FNAME	E_LNAME	E_ADDRESS	E_PHONE_NO
101	Jack	Williams		765432
102	Peter	Adams		712832

EMP_DEPARTMENT

E_ID	D_NAME	DATE_OF_JOIN	DESIGNATION
101	Sales	2001-01-01	Supervisor
102	Sales	2000-02-05	Worker
103	Research	2003-04-06	Worker

- ◆ You can identify inconsistency in the data from the tables
- ◆ The employee with the `E_ID` as 103 does not have a corresponding entry in the `EMP_DETAILS`
- ◆ You will use referential integrity to remove inconsistency from the tables
- ◆ To implement referential integrity, you will have to use a foreign key

- ◆ The syntax for defining a foreign key while creating a table is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table  
(create_clause,...) [table_options] [IGNORE [REPLACE]  
select], FOREIGN KEY(col_name) REFERENCES  
<table_name>(referenced_column);
```

where,

FOREIGN KEY – defines a foreign key

col_name – specifies the name of the column to be used for
the foreign key

REFERENCES – defines a relationship with other tables

table_name – specifies the name of the table to define a
relationship

referenced_column – specifies the column of the table
to define a relationship

- ◆ Consider the following example where the E_ID is defined as a primary key in the EMP_DETAILS table

```
CREATE TABLE EMP_DETAILS (E_ID INT, E_FNAME CHAR  
(10), E_LNAME CHAR(15), ADDRESS CHAR(15), E_PHONE_NO  
INT(6), PRIMARY KEY (E_ID));
```

- ◆ Consider the following example, where the E_ID is defined as a Foreign key referencing E_ID column of EMP_DETAILS table

```
CREATE TABLE EMP_DEPARTMENT (E_ID INT, D_NAME  
CHAR(), DATE_OF_JOIN DATE, DESIGNATION CHAR(6), FOREIGN KEY  
(E_ID) REFERENCES EMP_DETAILS (E_ID));
```


- ◆ EMP_DETAILS is the master table and EMP_DEPARTMENT is the child table
- ◆ The relationship enables to store consistent records in the tables
- ◆ Referential integrity monitors addition and deletion of records from these tables
- ◆ MySQL will generate an error message when you attempt to add a new record in the child table that does not have a corresponding record in the parent table
- ◆ MySQL will generate an error message when you delete a record from the child table that has a corresponding entry in the master table

- ◆ Databases are used to store information and can contain more than one table
- ◆ A table consists of rows and columns. A column in a table is known as a field and row is known as a record
- ◆ MySQL supports data types, such as numeric, string, date, and complex
- ◆ Numeric data type column consists of different types of numbers. A number can be classified as signed or unsigned. A signed number stores a negative or positive value
- ◆ String data type columns consist of data in the form of characters
- ◆ Date data type column have date values entered as string values in the form of 'YYYY-MM-DD' format
- ◆ MySQL provides the **CREATE** command to generate new databases and tables
- ◆ Normalization removes redundancies from the database
- ◆ A Primary key is a field in the table which is used as the unique identifier. A primary key having more than one attribute is called Composite key

- ◆ A Foreign key in a table is a Primary Key in another table
- ◆ An entity is said to be in the first normal form if all the attributes are single valued
- ◆ An entity is said to be in the second normal form, if it is in the first normal and all non key attributes entirely depend on the unique identifier of the entity
- ◆ An entity is said to be in the third normal form, if it is in the second normal form and no non-identifying attributes are dependent on any other non-identifying attributes
- ◆ An entity is said to be in the Boyce-Codd normal form if it is already in the third normal form and only one unique identifier exists
- ◆ Referential integrity means that when you have a record in a table that refers to corresponding record in another table then that particular record must exist
- ◆ Indexing enables faster search and retrieval operations in a database