PADHAAII

COMPUTER SCIENCE With python

PADHAAII.COM
WE MAKE EXAM EASY

Textbook for Class XII

- Programming & Computational Thinking
- Computer Networks
- Data Management (SQL, Django)
- Society, Law and Ethics

SUMITA ARORA

DHANPAT RAI & Co.

Scanned by CamScanner

PADHAAII

13 MySQL SQL Revision Tour

431 – 456

13.1 Introduction 431
13.2 Relational Data Model 432
13.3 MySQL – A popular Relational DataBase Management System 433
13.3.1 MySQL and SQL 433
13.3.2 Common MySQL Data Types 434
13.4 Accessing Database in MySQL 434
13.5 Creating Tables in MySQL 435
13.6 Inserting Data into Table 435
13.7 Making Simple Queries Through Select Command 436
13.7.1 Selecting All Data 436
13.7.2 Selecting Particular Rows 436
13.7.3 Selecting Particular Columns 437
13.7.4 Eliminating Redundant Data (with Keyword DISTINCT) 437
13.7.5 Selecting From All the Rows – ALL Keyword 437
13.7.6 Viewing Structure of a Table 437
13.7.7 Performing Simple Calculations 438
13.7.8 Using Column Aliases 438
13.7.9 Condition Based on a Range 438
13.7.10 Condition Based on a List 438
13.7.11 Condition Based on Pattern Matches 438
13.7.12 Searching for NULL 439
13.8 Creating Tables with SQL Constraints 440
13.8.1 SQL Constraints 440
13.8.2 Applying Table Constraints 444
13.9 Viewing a Table Structure 444
13.10 Inserting Data into Another Table 445
13.11 Modifying Data in Tables 445
13.12 Deleting Data from Tables 445
13.13 Altering Tables 446
13.14 Dropping Tables 447
13.15 SQL joins 447
13.16 Indexes in Database 448

padhaaii

Scanned by CamScanner

PADHAAII

13 MySQL SQL Revision Tour

In This Chapter

13.1 Introduction
13.2 Relational Data Model
13.3 MySQL–A Popular Relational Database Management System
13.4 Accessing Database in MySQL
13.5 Creating Tables in MySQL
13.6 Inserting Data into Table
13.7 Making Simple Queries Through Select Command
13.8 Creating Tables with SQL Constraints
13.9 Viewing a Table Structure
13.10 Inserting Data Into Another Table
13.11 Modifying Data in Tables
13.12 Deleting Data from Tables
13.13 Altering Tables
13.14 Dropping Tables
13.15 SQL joins
13.16 Indexes in Database

13.1 INTRODUCTION

A database system is basically a computer based record keeping system. The collection of data, usually referred to as the *database*, contains information about one particular enterprise. In a typical file-processing system, permanent records are stored in various files. A number of different application programs are written to extract records from files and add records to the appropriate files. But this scheme has a number of major limitations and disadvantages, such as data redundancy (duplication of data), data inconsistency, unsharable data, unstandardized data, insecure data, incorrect data etc. A *database management system* is answer to all these problems as it provides a centralized control of the data.

Various advantages of database systems are :

- Database systems reduce data redundancy (*data duplication*) to a large extent.
- Database systems control data inconsistency to a large extent.
- Databases facilitate sharing of data.
- Databases enforce standards.
- Centralized databases can ensure data security.
- Integrity can be maintained through databases.

Scanned by CamScanner

PADHAAII

13.2 RELATIONAL DATA MODEL

A data model refers to a set of concepts to describe the structure of a database, and certain *constraints* (restrictions) that the database should obey.

Most commonly used data model for databases is the **relational data model**.

In relational data model, the data is organized into tables (*i.e.*, rows and columns). These tables are called *relations*. A row in a table represents a *relationship* among a set of values. Since a table is a collection of such relationships, it is generally referred to using the mathematical term *relation*, from which the relational data model derives its name.

Let us now revise different terms used in relational model.

- *Relation* — A table storing logically related data ; data must be atomic in a cell ; all rows of this table are distinct ; ordering of rows and columns is immaterial.
- *Domain* — This is a pool of values from which the actual values appearing in a given column are drawn.
- *Tuple* — A row of a relation is generally referred to as a *tuple*.
- *Attribute* — A column of a relation is generally referred to as an *attribute*.
- *Degree* — This refers to the number of attributes in a relation.
- *Cardinality* — This refers to the number of tuples in a relation.
- *View* — It is a virtual table that does not really exist in its own right but is instead derived from one or more underlying base table(s).
- *Primary Key* — This refers to a set of one or more attributes that can uniquely identify tuples within the relation.
- *Candidate Key* — All attribute combinations inside a relation that can serve as primary key are *candidate keys* as these are candidates for primary key position.
- *Alternate Key* — A candidate key that is not primary key, is called an *alternate key*.
- *Foreign Key* — A non-key attribute, whose values are derived from the primary key of some other table, is known as *foreign key* in its current table.

Referential Integrity

A referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data.

You can set referential integrity when all of the following conditions are met :

- The matching field from the primary table is a primary key or has a unique index.
- The related fields have the same data type.
- Both tables belong to the same database. Referential integrity can't be enforced for linked table from databases in other formats.

When referential integrity is enforced, you must observe the following rules :

- You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.
- You can't delete a record from a primary table if matching records exist in a related table.
- You can't change a primary key value in the primary table, if that record has related records.

Scanned by CamScanner

PADHAAII

13.3 MYSQL – A POPULAR RELATIONAL DATABASE MANAGEMENT SYSTEM

MySQL is a freely available open source *Relational Database Management System* (RDBMS) that uses Structured Query Language (SQL). It is downloadable from site www.mysql.org. In a MySQL database, information is stored in *Tables*. A single MySQL database can contain many tables at once and store thousands of individual records. MySQL provides you with a rich set of features that support a secure environment for storing, maintaining, and accessing data. MySQL is a fast, reliable, scalable alternative to many of the commercial RDBMSs available today.

MySQL Database System

MySQL database system refers to the combination of a MySQL server instance and a MySQL database. MySQL operates using client/server architecture in which the server runs on the machine containing the databases and clients connect to the server over a network.

- **The server** (MySQL server) listens for client requests coming in over the network and accesses database contents according to those requests and provides that to the clients.
- **Clients** are programs that connect to the database server and issue queries in a pre-specified format. MySQL is compatible with the standards based SQL (Structured Query Language).

13.3.1 MySQL and SQL

In order to access data within the MySQL database, all programs and users must use, *Structured Query Language* (SQL). SQL is the set of commands that is recognised by nearly all RDBMSs.

The Structured Query Language (SQL) is a language that enables you to create and operate on relational databases, which are sets of related information stored in tables.

Classification of SQL Statements

SQL provides many different types of commands used for different purposes. SQL commands can be mainly divided into following categories :

(*i*) **Data Definition Language (DDL) Commands.** Commands that allow you to perform tasks related to data definition *e.g.*,

- creating, altering and dropping.
- granting and revoking privileges and roles.
- maintenance commands

(*ii*) **Data Manipulation Language (DML) Commands.** Commands that allow you to perform data manipulation *e.g.*, retrieval, insertion, deletion and modification of data stored in a database.

(*iii*) **Transaction Control Language (TCL) Commands.** Commands that allow you to manage and control the transactions (a transaction is one complete unit of work involving many steps), *e.g.*,

- making changes to database, permanent
- undoing changes to database, permanent
- creating savepoints
- setting properties for current transactions

There are other categories of SQL commands also but above three categories of commands are mainly used by learners.

Scanned by CamScanner

PADHAAII

13.3.2 Common MySQL Data Types

Data types are means to identify the type of data and associated operations for handling it. A value's *datatype* associates a fixed set of properties with the value. Some commonly used data types of MySQL are briefly listed below.

MySQL Data Types

Data Type	Spec	Data Type	Spec
CHAR	String (0 – 255)	INT	Integer (– 2147483648 to 2147483647)
VARCHAR	String (0 – 255)	BIGINT	Integer (– 9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 – 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 – 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 – 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 – 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 – 16777215)	DATETIME	YYYY-MM-DDHH:MM:SS
LONGTEXT	String (0 – 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 – 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (– 128 to 127)	ENUM	One of preset options
SMALLINT	Integer (– 32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (– 8388608 to 8388607)	BOOLEAN	TINYINT(1)

Difference between Char and Varchar

The difference between CHAR and VARCHAR is that of *fixed length* and *variable length*. The CHAR datatype specifies a *fixed length* character string. When a column is given datatype as CHAR(*n*), then MySQL ensures that all values stored in that column have this length *i.e.*, *n* bytes. If a value is shorter than this length *n* then blanks are added, but the size of value remains *n* bytes.

VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(*n*), then the maximum size a value in this column can have is *n* bytes. Each value that is stored in this column stores exactly as you specify it *i.e.*, no blanks are added if the length is shorter than maximum length *n*. However, if you exceed the maximum length *n*, then an error message is displayed.

13.4 ACCESSING DATABASE IN MYSQL

Before you start writing SQL commands or making queries upon the data in tables of a database, you need to open the database for use. For this, after logging into MySQL, you need to issue a command

```
Use <databasename> ;
```

For example, we want to work on our sample database namely *menagerie*, so we shall write the following command after logging in *MySQL*.

```
mysql> USE menagerie ;
Database changed
mysql>
```

Scanned by CamScanner

PADHAAII

13.5 CREATING TABLES IN MYSQL

Tables are defined with the CREATE TABLE command. When a table is created, its columns are named, data types and sizes are supplied for each column. Each table must have at least one column.

The syntax of CREATE TABLE command is :

```
CREATE TABLE <table-name>
    (<column name> <data type> [ (<size>)],
    <columnname> <data type> [ ( <size> ) ... ]) ;
```

NOTE

Before issuing a CREATE TABLE command, make sure that its parent database has been opened using USE <database name> command.

To create an *employee* table whose schema is as follows :

```
employee (ecode, ename, sex, grade, gross)
```

the *SQL* command will be

```
CREATE TABLE employee
(   ecode    integer,
    ename    char(20),
    sex      char(1),
    grade    char(2),
    gross    decimal ) ;
```

13.6 INSERTING DATA INTO TABLE

The rows (tuples) are added to relations using INSERT command of *SQL*. In its simplest form, INSERT takes the following syntax :

```
INSERT INTO <tablename> [ <column list> ]
VALUES ( <value>, <value> ... ) ;
```

For example, to enter a row into *employee* table (defined earlier), you could use the following statement :

```
INSERT INTO employee
VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;
```

See the order of values matches the order of columns in the CREATE TABLE command of *employee*. The same can be done with an alternate command as shown below :

```
INSERT INTO employee (ecode, ename, sex, grade, gross)
VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;
```

The INSERT statement adds a new row to *employee* giving a value for every column in the row. Note that the data values are in the same order as the column names in the table. Data can be added only to some columns in a row by specifying the columns and their data.

For instance, if you want to insert only *ecode, ename* and *sex* columns, you use the command :

```
INSERT INTO employee (ecode, ename, sex)
VALUES (2014, 'Manju', 'F') ;
```

Scanned by CamScanner

PADHAAII

The columns that are not listed in the INSERT command will have their default value, if it is defined for them, otherwise, NULL value.

If any other column (that does not have a **default value** and is defined NOT NULL) is skipped or omitted, an error message is generated and the row is not added.

NOTE

In an INSERT statement, only those columns can be omitted that have either default value defined or they allow NULL values.

Inserting NULL values

To insert value NULL in a specific column, you can type NULL without quotes and NULL will be inserted in that column.

Consider the following statement :

```
INSERT INTO EMPL (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
VALUES (8100, 'YASH', 'ANALYST', NULL, '10-MAY-03', 6000, NULL, 20) ;
```

See, for **Mgr** and **Comm** columns, NULL values have been inserted.

Inserting Dates

Dates are by default entered in 'YYYY-MM-DD' format *i.e.*, first four digits depicting *year*, followed by a hyphen, followed by 2 digits of *month*, followed by a hyphen and a two digit day. All this is enclosed in single quotes.

13.7 MAKING SIMPLE QUERIES THROUGH SELECT COMMAND

The SELECT statement is used to pull information from a table. The general form of the statement is :

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy ;
```

13.7.1 Selecting All Data

The simplest form of SELECT retrieves everything from a table. You just need to specify asterisk in the *select-list*(what_to_select), *e.g.*, :

```
mysql> SELECT * FROM pet ;
```

13.7.2 Selecting Particular Rows

You can select particular rows from a table by specify filtering condition through WHERE clause of the SELECT statement, *e.g.*,

1. Select all pets with gender(sex) as *male*("m").

```
mysql>SELECT * FROM pet
      WHERE sex = 'm' ;
```

2. Select all pets that were born on or after Jan 1, 2019.

```
mysql>SELECT * FROM pet
      WHERE birth >= '2019-1-1' ;
```

← *date in yyyy-mm-dd format*

Scanned by CamScanner

PADHAAII

3. Select all female-dogs.

```
mysql> SELECT * FROM pet
       WHERE species = 'dog' AND sex = 'f';
```

4. Select all snakes or birds.

```
mysql> SELECT * FROM pet
       WHERE species = 'snake' OR species = 'bird';
```

5. Select all male cats.

```
mysql> SELECT * FROM pet
       WHERE (species = 'cat' AND sex = 'm') ;
```

13.7.3 Selecting Particular Columns

You can select particular columns by specifying column-names (*i.e.*, attributes) in the select-list of the SELECT command, *e.g.*,

1. Display names and birth-dates of all pets.

```
mysql> SELECT name, birth FROM pet ;
```

2. Display owners of pets born after Dec 2018.

```
mysql> SELECT owner FROM pet
       WHERE birth >'2018-12-31' ;
```

← *date in yyyy-mm-dd format*

13.7.4 Eliminating Redundant Data (with Keyword DISTINCT)

By default, data is selected from all the rows of the table, even if the data appearing in the result gets duplicated. The DISTINCT keyword eliminates duplicate rows from the results of a SELECT statement. *For example,*

1. Display names of all pet-owners (non-redundant).

```
mysql> SELECT DISTINCT owner FROM pet;
```

2. Display distinct species of pets from table pet.

```
mysql> SELECT DISTINCT(species) FROM pet.
```

13.7.5 Selecting From All the Rows – ALL Keyword

If in place of keyword DISTINCT, you give keyword ALL then the result retains the duplicate output rows. It is just the same as when you specify neither DISTINCT nor ALL ; ALL is essentially a clarifier rather than a functional argument. Thus if you give

```
SELECT ALL city FROM suppliers ;
```

it will give values of *city* column from every row of the table without considering the duplicate entries.

13.7.6 Viewing Structure of a Table

If you want to know the structure of a table, you can use *Describe* or *Desc* command as per following syntax :

```
DESCRIBE | DESC <table name> ;
```

For instance, the commands : DESCRIBE pet ; or DESC pet ; will display the structure of table *pet*.

```
mysql> DESC pet ;
```

Scanned by CamScanner

PADHAAII

13.7.7 Performing Simple Calculations

To perform simple calculations, you can write the expression/formula to be calculated next to keyword SELECT, *e.g.*,

1. To calculate 3.14159*6*6

```
mysql> SELECT 3.14159*6*6 ;
```

2. To obtain current system date

```
mysql> SELECT curdate();
```

13.7.8 Using Column Aliases

The columns that you select in a query can be given a different name *i.e.*, **column alias name** for output purposes. As per following syntax :

```
Select <columnname> AS [columnalias] [, <columnname> AS [columnalias]]
      .....
From <tablename> ;
```

For example,

```
mysql> SELECT date, type AS "Event Type"
    -> FROM event ;
```

13.7.9 Condition Based on a Range

The BETWEEN operator defines a range of values that the column values must fall in to make the condition true. The range includes both lower value and the upper value. For example, to list the items whose *QOH* falls between 30 to 50 (both inclusive), the command would be :

```
SELECT icode, descp, QOH
FROM items
WHERE QOH BETWEEN 30 AND 50 ;
```

13.7.10 Condition Based on a List

To specify a list of values, IN operator is used. The IN operator selects values that match any value in a given list of values. For example, to display a list of members from 'DELHI', 'MUMBAI', 'CHENNAI' or 'BANGALORE' cities, you may give

```
SELECT * FROM members
WHERE city IN ('DELHI', 'MUMBAI', 'CHENNAI', 'BANGALORE') ;
```

The NOT IN operator finds rows that do not match in the list. So if you write

```
SELECT * FROM members
WHERE city NOT IN ('DELHI', 'MUMBAI', 'CHENNAI') ;
```

it will list members not from the cities mentioned in the list.

13.7.11 Condition Based on Pattern Matches

SQL also includes a string-matching operator, LIKE, for comparisons on character strings using patterns. Patterns are described using two special wildcard characters :

- **percent (%).** The % character matches any substring.
- **underscore (_).** The _ character matches any character.

Scanned by CamScanner

PADHAAII

The LIKE keyword is used to select rows containing columns that match a wildcard pattern.

Examples :

1. To list members which are in areas with pin codes starting with 13, the command is :

```
SELECT firstname, lastname, city
FROM members
WHERE pin LIKE '13%' ;
```

2. To list names of pets who have names ending with '*y*', the command would be :

```
SELECT name
FROM emp
WHERE name LIKE '%y' ;
```

13.7.12 Searching for NULL

The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause. (Relational operators like =, <> etc. can't be used with NULL). For example, to list details of all employees whose departments contain NULL (*i.e.*, novalue), you use the command :

```
SELECT empno, empname, job
FROM emp WHERE DeptNo IS NULL ;
```

Non-NULL values in a table can be listed using IS NOT NULL.

LET US REVISE

- *The basic elements of MySQL SQL are : literals, datatypes, nulls and comments.*
- *Literals are fixed data values.*
- *Data types of MySQL SQL include : NUMERIC (INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, FLOAT(M, D), DOUBLE(M, D) AND DECIMAL(M, D), DATE and TIME (DATE, DATETIME, TIMESTAMP, TIME and YEAR(M) and STRING (CHAR(M), VARCHAR(M), BLOB or TEXT, TINYBLOB or TINYTEXT, MEDIUMBLOB or MEDIUMTEXT and ENUM).*
- *Column having no value is said to have NULL value.*
- *The SELECT command of SQL lets you make queries on the database.*
- *The DISTINCT keyword eliminates redundant data.*
- *To perform calculations, the expressions can be written along with SELECT without specifying any table name.*
- *A small work table **Dummy** can be used for making simple calculations.*
- *The **curdate** pseudo-column returns the current system date.*
- *The WHERE clause is used to select specific rows.*
- *The logical operators OR (||) , AND (&&) and NOT (!) are used to connect search conditions in the WHERE clause.*
- *The BETWEEN operator is used for making range checks in queries.*
- *The IN operator is used for selecting values from a list of values.*
- *The LIKE operator is used for making character comparisons using strings.*
- *The null value in a column can be searched for in a table using IS NULL in the WHERE clause.*

Scanned by CamScanner

PADHAAII

13.8 CREATING TABLES WITH SQL CONSTRAINTS

You already know that a table is created using CREATE TABLE command. The SQL syntax for **CREATE TABLE** is

```
CREATE TABLE "table_name"
("column 1" "data_type_for_column_1",
"column 2" "data_type_for_column_2", ... ) ;
```

So, if we are to create the customer table specified as above, we would type in

```
CREATE TABLE customer
(    First_Name char(50),
     Last_Name char(50),
     Address char(50),
     City char(50),
     Country char(25),
     Birth_Date date   ) ;
```

But while creating tables, we may need to apply certain conditions on columns *e.g.*, this column must not be blank or NULL for any record. To apply conditions on columns, SQL constraints are used.

13.8.1 SQL Constraints

Common types of constraints that are applied on table columns include the following :

S.No.	Constraints	Description
1.	*NOT NULL*	❖ Ensures that a column cannot have NULL value.
2.	*DEFAULT*	❖ Provides a default value for a column when none is specified.
3.	*UNIQUE*	❖ Ensures that all values in a column are different.
4.	*CHECK*	❖ Makes sure that all values in a column satisfy certain criteria.
5.	*Primary Key*	❖ Used to uniquely identify a row in the table.
6.	*Foreign Key*	❖ Used to ensure referential integrity of the data.

13.8.1A SQL NOT NULL Constraint

By default, a column can hold NULL. If you do not want to allow NULL value in a column, you will want to place a constraint on this column specifying that NULL is now not an allowable value. For example, in the following statement,

```
CREATE TABLE Customer
(    SID integer NOT NULL,
     Last_Name varchar (30) NOT NULL,
     First_Name varchar(30)  );
```

CONSTRAINT

A **Constraint** is a condition or check applicable on a field or set of fields.

Columns *SID* and *Last_Name* cannot include NULL, while *First_Name* can include NULL.

An attempt to execute the following SQL statement,

```
INSERT INTO Customer (Last_Name, First_Name)
VALUES ('Wang','Pedro');
```

will result in an error because this will lead to column *SID* being NULL, which violates the NOT NULL constraint on that column.

Scanned by CamScanner

PADHAAI

13.8.1B SQL DEFAULT Constraint

The DEFAULT constraint provides a *default value* to a column when the INSERT INTO statement does not provide a specific value.

For example, if we create a table as below :

```
CREATE TABLE Student
(    Student_ID integer Unique,
     Last_Name varchar (30),
     First_Name varchar (30),
     Score DEFAULT 80 );
```

and execute the following SQL statement,

```
INSERT INTO Student (Student_ID, Last_Name, First_Name)
VALUES ('10','Qureshi','Zeeshan');
```

See, no value has been provided for Score field

After this SQL query, the table will look like the following :

Student_ID	Last_Name	First_Name	Score
10	Qureshi	Zeeshan	80

See, Score field has got the default value

Even though we didn't specify a value for the *Score* column in the INSERT INTO statement, it does get assigned the *default value* of *80* since we had already set *80* as the *default value* for this column.

13.8.1C SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are distinct. In other words, no two rows can hold the same value for a column with UNIQUE constraint.

For example, in the following CREATE TABLE statement,

```
CREATE TABLE Customer
(    SID integer Unique,
     Last_Name varchar (30),
     First_Name varchar(30) );
```

column *SID* has a **unique** constraint, and hence cannot include duplicate values. Such constraint does not hold for columns *Last_Name* and *First_Name*. So, if the table already contains the following rows :

SID	Last_Name	First_Name
1	Wang	Pedro
2	Qureshi	Zeeshan
3	Rastogi	Rajiv

Scanned by CamScanner

PADHAAII

Executing the following SQL statement,

```
INSERT INTO Customer
VALUES ('3','Cyrus','Grace');
```

will result in an error because the value *3* already exists in the *SID* column, thus trying to insert another row with that value violates the UNIQUE constraint.

NOTE

Please note that a column that is specified as a primary key must also be unique. At the same time, a column that's unique may or may not be a primary key. In addition, multiple UNIQUE constraints can be defined on a table but PRIMARY KEY constraint is defined only once in the table.

13.8.1D SQL CHECK Constraint

The CHECK constraint ensures that all values in a column satisfy certain conditions. Once defined, the database will only insert a new row or update an existing row if the new value satisfies the CHECK constraint. The CHECK constraint is used to ensure data quality For example, in the following CREATE TABLE statement,

```
CREATE TABLE Customer
(    SID integer CHECK (SID > 0),
     Last_Name varchar (30),
     First_Name varchar(30) );
```

NOTE

Please note that MySQL does not enforce the CHECK constraint although MySQL syntax allows the definition of CHECK constraint. As per MySQL documentation, 'Check constraint gets parsed but IGNORED'.

13.8.1E PRIMARY KEY Constraint

A primary key is used to uniquely identify each row in a table. It can either be part of the actual record itself, or it can be an artificial field (one that has nothing to do with the actual record). A primary key can consist of one or more fields on a table. When multiple fields are used as a primary key, they are called a **composite key**.

Primary keys can be specified either when the table is created (using **CREATE TABLE**) or by changing the existing table structure (using **ALTER TABLE**).

Defining Primary Key through Create Table Command

You can define a primary key in CREATE TABLE command through keywords PRIMARY KEY.

Below are examples for specifying a primary key when creating a table :

```
     CREATE TABLE Customer
     (    SID integer not null PRIMARY KEY,
          Last_Name varchar(30),
          First_Name varchar(30)  );
Or   CREATE TABLE Customer
     (    SID integer not null,
          Last_Name varchar(30),
          First_Name varchar(30),
          PRIMARY KEY (SID)   );
```

The latter way is useful if you want to specify a *composite primary key* (*i.e.*, having a group of fields) *e.g.*,

```
CREATE TABLE Customer
(    Branch integer not null,
     SID integer not null,
     Last_Name varchar(30),
     First_Name varchar(30),
     PRIMARY KEY (Branch, SID)  );
```

*Combination of fields **Branch and SID** is the composite primary key of the table.*

Scanned by CamScanner

PADHAAII

Defining Primary Key through Alter Table Command

You can define a primary key in ALTER TABLE command through keywords

```
ADD PRIMARY KEY (<key-field>)
```

Below are examples for specifying a primary key by altering a table :

```
ALTER TABLE Customer
ADD PRIMARY KEY (SID);
```

NOTE

Before using the ALTER TABLE command to add a primary key, you'll need to make sure that the field is defined as 'NOT NULL' – in other words, NULL cannot be an accepted value for that field.

13.8.1F Foreign Key Constraint

In an RDBMS, tables reference one another through common fields and to ensure validity of references, *referential integrity* is enforced. *Referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data.* Referential integrity is ensured through FOREIGN KEY constraint. This is implemented as explained in the following paragraph.

Whenever two tables are related by a common column (or set of columns), then the related column(s) in the **parent table** (or **primary table**) should be either declared a PRIMARY KEY or UNIQUE key and the related column(s) in the **child table** (or **related table**) should have FOREIGN KEY constraint. For instance, if we have two tables having structures as given below :

Table : CUSTOMER

column name	characteristic
SID	Primary Key
Last_Name	
First_Name	

Table : ORDERS

column name	characteristic
Order_ID	Primary Key
Order_Date	
Customer_SID	Foreign Key
Amount	

In the above example, the **Customer_SID** column in the *ORDERS* table is a foreign key pointing to the **SID** column in the *CUSTOMER* table.

Just like primary key, Foreign key can also be created in *two* ways : through CREATE TABLE and ALTER TABLE commands.

Defining Foreign key through Create Table

In Create Table command, you can add foreign key's definition through following syntax :

```
Foreign Key (<column-to-be-designated-as-foreign-key>) references
Master-Table(<primary-key-of master-table>);
```

Following example shows how to specify the foreign key when creating the ORDERS table :

```
CREATE TABLE ORDERS
(    Order_ID integer,
     Order_Date date,
     Customer_SID integer,
     Amount double,
     Primary Key (Order_ID),
     Foreign Key (Customer_SID) references CUSTOMER(SID) );
```

Scanned by CamScanner

PADHAAII

The above code will designate **Customer_SID** field of *ORDERS* table as foreign key referencing **SID** field of *CUSTOMER* table.

Defining Foreign key through Alter Table

In Alter Table command, you can add foreign key's definition through following syntax :

```
ALTER TABLE <table-name>
ADD FOREIGN KEY (<column-to-be-designated-as-foreign-key>)
references Master-Table(<primary-key-of master-table>) ;
```

Following example specifies a foreign key by altering a table. This assumes that the ORDERS table has been created, and the foreign key has not yet been put in :

```
ALTER TABLE ORDERS
ADD FOREIGN KEY (customer_sid) REFERENCES CUSTOMER(SID);
```

IMPORTANT

Foreign Key and Storage Engine

MySQL is capable of creating databases in many different storage engines that offer different features. The default storage engine is ISAM but ironically it does not support foreign keys. So, if you find that your foreign key constraint are being ignored or not implemented, you need to change the storage engine of your database tables to InnoDB storage engine.

To do so you simply need to write following command for your connecting tables :

```
ALTER TABLE <tablename> ENGINE = innodb ;
```

e.g.,

```
ALTER TABLE table1 ENGINE = innodb ;
```

Once done, you can create and implement foreign keys for your tables.

13.8.2 Applying Table Constraints

When a constraint is to be applied on a group of columns of the table, it is called ***table constraint***. The table constraints appear in the end of table definition. For instance, if you want combination of *icode* and *descp* of table *items* to be unique, you may write it as follows :

```
CREATE TABLE items
(   icode    char (5)     NOT NULL,
    descp    char (20)    NOT NULL,
    ROL      integer,
    QOH      integer,
    CHECK    (ROL < QOH),
    UNIQUE   (icode, descp) ) ;
```

these are a table constraints

The above statement ensures that the combination of *icode* and *desc* in each row must be unique.

A constraint applied on one column (*e.g.*, as you define **not null** with a column definition) is known as **column constraints.**

13.9 VIEWING A TABLE STRUCTURE

Once you have created a table, you may want to view its structure. Anytime, if you want to view an already created/existing table's structure, you may use DESC[RIBE] command of MySQL.

Scanned by CamScanner

PADHAAII

You can use DESC[RIBE] command as follows :

```
DESC[RIBE] <tablename> ;
```

For example,

```
DESC empl ;
```

or

```
DESCRIBE empl ;
```

13.10 INSERTING DATA INTO ANOTHER TABLE

You already know that data is added to tables using INSERT INTO command *e.g.,*

```
INSERT INTO employee
VALUES (1001, 'Rahman', 'M', 'E4', 14700.00) ;
```

Or

```
INSERT INTO employee (ecode, ename, sex)
VALUES (3005, 'Meera', 'F') ;
```

INSERT command can also be used to take or derive values from one table and place them in another by using it with a query. To do this, simply replace the VALUES clause with an appropriate query as shown in the following example :

```
INSERT INTO branch1
SELECT * FROM branch2
WHERE gross > 7000.00 ;
```

This query will generate data to be inserted into table

It will extract all those rows from *branch2* that have *gross* more than 7000.00 and insert this produced result into the table *branch1*. But for above command to work, table namely **branch1** must be an existing table of the database.

13.11 MODIFYING DATA IN TABLES

You can modify data in tables using UPDATE command of SQL. The UPDATE command specifies the rows to be changed using the WHERE clause, and the new data using the SET keyword. The new data can be a specified constant, an expression or data from other tables. For example, to change the reorder level *ROL* of all items to 250, you would write

```
UPDATE items
SET ROL = 250 ;
```

To update the *ROL* and *QOH* for items having icode less than 'I040', we shall write

```
UPDATE items
SET ROL = 400, QOH = 700
WHERE icode < 'I040' ;
```

To double the gross pay of employees of grade 'E3' and 'E4', you use the command :

```
UPDATE employee
SET gross = gross * 2
WHERE (grade = 'E3' OR grade = 'E4') ;
```

13.12 DELETING DATA FROM TABLES

To delete some data from tables, you can use SQL DELETE commands. The DELETE command removes rows from a table. This removes the entire rows, not individual field values, so no field argument is needed or accepted.

Scanned by CamScanner

PADHAAII

The DELETE statement takes the following general form :

```
DELETE FROM <tablename>
[WHERE <predicate> ] ;
```

For instance, to remove the tuples from *employee* table that have *gross* (salary) less than 2200.00, the following command is used :

```
DELETE FROM employee
WHERE gross < 2200.00 ;
```

If you specify no condition with WHERE, then all the rows of the table will be deleted, *e.g.*, To remove all the contents of *items* table, you use the command :

```
DELETE FROM items ;
```

13.13 ALTERING TABLES

The ALTER TABLE command is used to change definitions of existing tables. Usually, it is used to add columns to a table. Sometimes it is used to delete columns (depending on privileges) or change their sizes. In general, in MySQL SQL, ALTER TABLE command is used :

- to add a column
- to add an integrity constraint
- to redefine a column (datatype, size, default value).

You can use ALTER TABLE **for adding columns** as follows :

```
ALTER TABLE <table name> ADD <column name> <data type><size> [<constraint name>] ;
```

For instance, to add a new column *tel_number* of type *integer* in table *Empl* you may give :

```
ALTER TABLE Empl
ADD (tel_number integer) ;
```

To *modify existing columns of table*, ALTER TABLE command can be used according to following syntax :

```
ALTER TABLE <tablename>
MODIFY (columname newdatatype (newsize) ) [FIRST|AFTER column] ;
```

To modify column *Job* of table *Empl* (given just before the *Assignment* at the end of this chapter) to have new width of 30 characters, you may give :

```
ALTER TABLE Empl
MODIFY (Job char(30) ) ;
```

Sometimes you may need to **change the name of one of your columns**. For this you can use CHANGE clause of ALTER TABLE command as per following syntax :

```
ALTER TABLE
CHANGE [COLUMN] old_col_name new_col_name column_definition ;
```

For instance, to change the existing column namely *First_Name* of table *Student*, to *FirstName* you may write :

```
ALTER TABLE Customers
CHANGE First_Name FirstName VARCHAR(20);
```

See, complete column description is given along with the new name.

Scanned by CamScanner

PADHAAII

13.14 DROPPING TABLES

The DROP TABLE command of *SQL* lets you drop a table from the database. The syntax for using a DROP TABLE command is :

```
DROP TABLE [IF EXISTS] <tablename>
```

That is, to drop a table *items*, you need to write :

```
DROP TABLE items ;
```

Once this command is given, the table name is no longer recognized and no more commands can be given on that object.

The IF EXISTS clause of DROP TABLE first checks whether the given table exists in the database or not. If it does, then it drops the mentioned table from the database. For instance, consider the following query :

```
DROP TABLE IF EXISTS players ;
```

The above query will first check for existence of *players* table in current database. If it exists, then it (table *players*) will be dropped from the database.

13.15 SQL JOINS

An SQL Join is a query that fetches data from two or more tables whose records are joined with one another based on a condition. The common syntax for getting JOIN of two or more tables is :

```
SELECT <field list>
FROM <table1>, <table2> [, <table3>...]
WHERE <join condition for the tables>
```

For example,

```
mysql>  SELECT ENAME, LOC
        FROM EMPL, DEPT
        WHERE ENAME = 'ANOOP'
        AND EMPL.DEPTNO = DEPT.DEPTNO ;          -- see this is join condition
```

The above SQL statement will first join the tables **EMPL** and **DEPT** on the basis of join condition **EMPL.DEPTNO = DEPT.DEPTNO**. That is a record from table **EMPL** will join with a record from **DEPT** table whose **DEPTNO** is equal to its **DEPTNO**.

There can be many types of JOINS possible. These are :

(*i*) **Cartesian Product.** An SQL join query without any join condition returns all the records of joined with all the records of the other table. It is known as a Cartesian Product, This type of join is also known as a CROSS JOIN.

(*ii*) **Equi Join.** An SQL join query that joins two or more tables based on a condition using equality operator.

(*iii*) **Inner Join.** An INNER JOIN implements an equi join. In this join, only those rows are returned from both the tables that satisfy the join condition. The Join conditions can match records based on one or more columns.

(*iv*) **Natural Join.** A NATURAL JOIN is a type of equi join where the join condition compares all the same names columns in both tables.

Scanned by CamScanner

PADHAAII

(*v*) **Left Join.** The LEFT JOIN is a particular type of join that selects rows from both left and right tables that are matched, plus **all rows from the left table** (Table A) even with no matching rows found in the right table (Table B).

(*vi*) **Right Join.** The RIGHT JOIN is a particular type of join that selects rows from both left and right tables that are matched, plus **all rows from the right table** (Table B) even with no matching rows found in the left table (Table A).

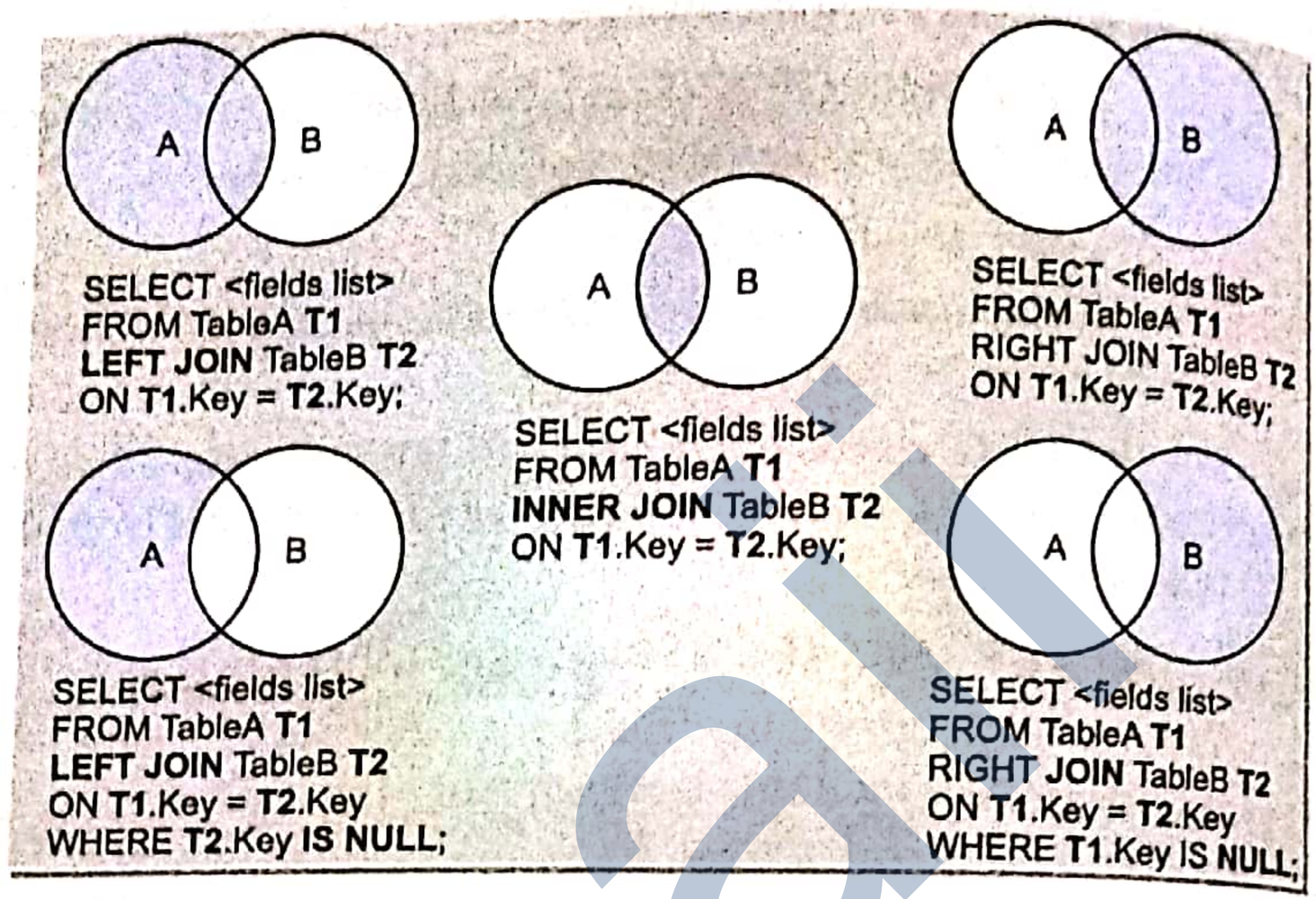

Figure 13.1 SQL Joins

13.16 INDEXES IN DATABASE

An **index** is a data structure maintained by a database that helps it find records within a table more quickly. An index stores the sorted/ordered values within the index field and their location in the actual table. An **index** in a database is also a table (a database structure) which stores arranged values of one or more columns in a specific order.

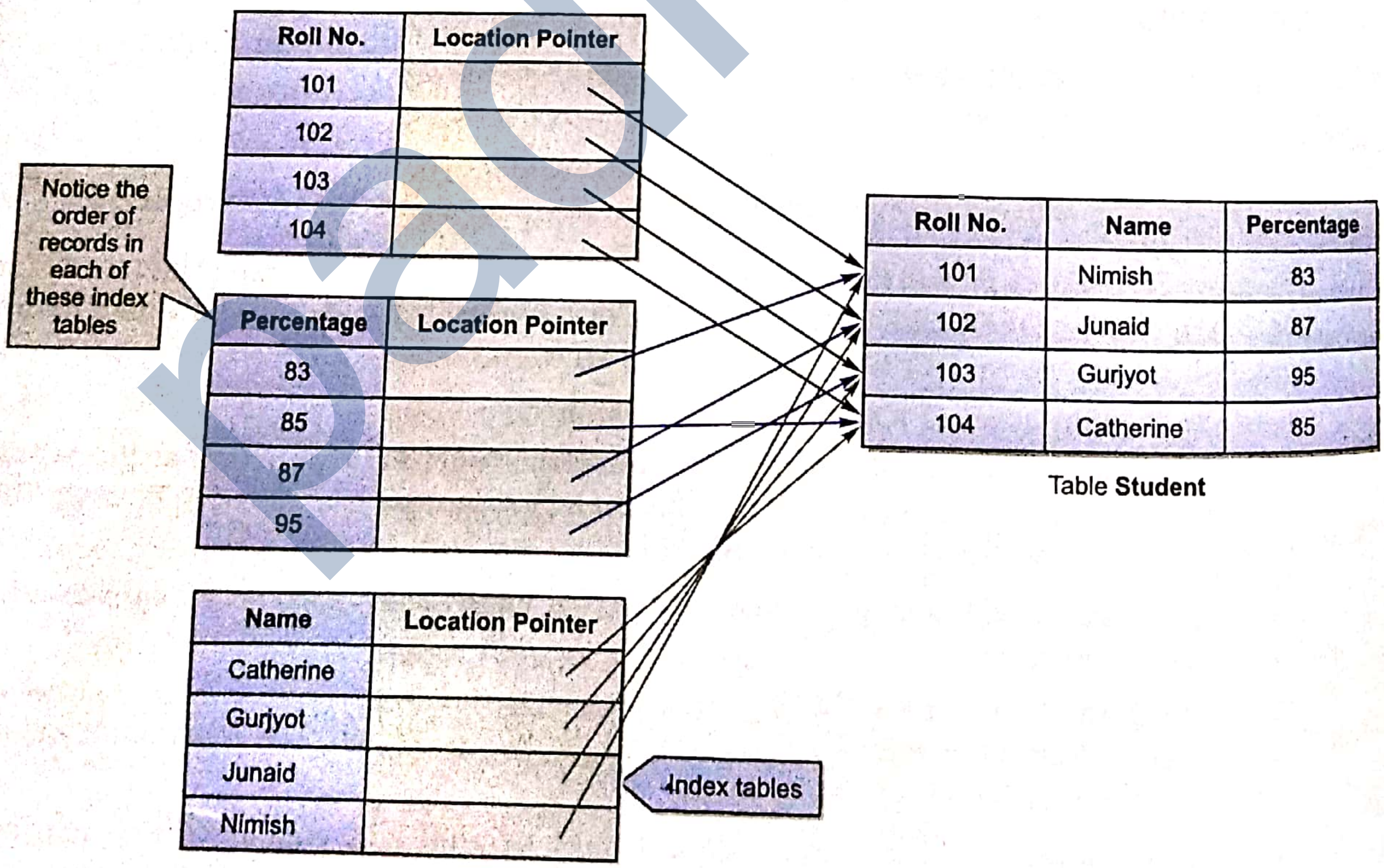

Figure 13.2 Multiples indexes of single data table

Scanned by CamScanner

PADHAAI

Creating Indexes in MySQL

MySQL also provides commands to create indexes in a database. You can create indexes in MySQL using following commands. MySQL provides *two* ways to create indexes :

(*i*) Create index at the time of table creation *i.e.,* along with CREATE TABLE command.

(*ii*) Create index on an already existing table.

(*i*) Creating Index at Time of Table Creation (*i.e.*, with CREATE TABLE)

To create index on a field at the time of table creation, you can specify the index-fields with INDEX clause of CREATE TABLE command, whose syntax is given below :

```
CREATE TABLE table_name
(
    column1 datatype [ NULL | NOT NULL ],
    column2 datatype [ NULL | NOT NULL ],
    ...
    column_n datatype [ NULL | NOT NULL ],
    ...
    INDEX index_name (index_col1 [(length)] [ASC | DESC],
        index_col2 [(length)] [ASC | DESC],
    ...
    index_col_n [(length)] [ASC | DESC])
);
```

Recall that in a syntax optional parts are given in square brackets [ ]

Please note that a CREATE TABLE statement with INDEX clause creates both the table as well as the index at the same time. Consider the following example :

```
CREATE TABLE PLAYERS
(   PLAYERNO INTEGER NOT NULL,
    NAME CHAR(15) NOT NULL,
    DOB DATE,
    SEX CHAR(1) NOT NULL,
    ADDRESS VARCHAR(100) NOT NULL,
    PHONE CHAR(15),
    TEAMNO CHAR(4) NOT NULL,
    PRIMARY KEY (PLAYERNO),
    INDEX Player_idx (TEAMNO)
);
```

The above SQL statement will create a table by the name **Players** and at the same time it will also create an index namely **Player_idx** on the basis of field **TeamNO**. Since, we have not specified any order ASC or DESC, it will take default order ASC *i.e.,* ascending for the index field.

Check Point 13.1

1. What is SQL ? What are the different categories of SQL commands ?
2. What is a datatype ? Name some datatypes available in MySQL.
3. How would you calculate 13 * 15 in SQL ?
4. Is NULL value the same as 0 (zero) ? Write the reason for your answer.
5. Write the UPDATE command to increase the commission (Column name : COMM) by 500 of all the Salesmen who have achieved Sales (Column name : SALES) more than 200000. The table's name is COMPANY.
6. While using SQL pattern matching, what is the difference between '_' (underscore) and '%' wildcard symbols ?
7. Write one similarity and one difference between CHAR and VARCHAR data types. [CBSE OD 15]
8. Write SQL statement to display
   Today, the date is <current date>
9. Write a command to add a NOT NULL constraint on FEES column of a student table.
10. What is a constraint ? Name some constraints that you can apply to enhance database integrity.
11. What is primary key ? What is PRIMARY KEY constraint ?
12. What is NOT NULL constraint ? What is DEFAULT constraint ?
13. When a column's value is skipped in an INSERT command, which value is inserted in the database ?
14. Write MySql command to display the list of existing databases. [CBSE D 12]
15. Write MySql command will be used to open an already existing database "CONTACTS". [CBSE OD 11]

Scanned by CamScanner

PADHAAII

(ii) Create Index on an Already Existing Table

Another way of creating index is to create it for already existing tables using following command:

```
CREATE INDEX <indexname> ON <tablename> (<column>[ASC | DESC],
        <column>[ASC | DESC] ...);
```

To enforce unique values, add the UNIQUE keyword :

```
CREATE UNIQUE INDEX <indexname> ON <tablename> (<column>[ASC | DESC],
      <column> [ASC | DESC]...);
```

For example,

```
CREATE UNIQUE INDEX Players_idx
ON Players (TeamNo);
```

The above statement will create a Unique index on the basis of field **TeamNo** on an existing table namely **Players**.

LET US REVISE

- *CREATE TABLE command is used to create tables in database.*
- *INSERT INTO command is used to insert data in the table.*
- *To insert data from other tables, subquery can be used inside INSERT INTO command.*
- *Existing data in tables can be changed with UPDATE command.*
- *Tuples in a table can be deleted using DELETE command.*
- *ALTER TABLE command is used to alter the definition of already created tables.*
- *With ALTER TABLE, new columns can be added, existing columns can be redefined.*
- *DROP TABLE command drops a table from a database.*
- *A join is an SQL query that fetches records from two or more tables based on a condition.*
- *An index is a data structure maintained by a database, that stores the sorted values within the index field and their location in actual data.*
- *You can create indexes either by using INDEX clause of CREATE TABLE command or by using command CREATE INDEX.*

Solved Problems

1. *What is SQL ? What are different categories of commands available in SQL ?*

Solution. In order to access data within the Oracle database, all programs and users must use, Structured Query Language (SQL). SQL is the set of commands that is recognised by nearly all RDBMSs. SQL commands can be divided into following categories :

1. Data Definition Language (DDL) Commands.
2. Data Manipulation Language (DML) Commands.
3. Transaction Control Language (TCL) Commands.
4. Session Control Commands.
5. System Control Commands.

Scanned by CamScanner

PADHAAII

2. *Differentiate between DDL and DML commands.*

Solution. The Data Definition Language (DDL) commands, as the name suggests, allow you to perform tasks related to data definition. That is, through these commands, you can perform tasks like, create, alter and drop schema objects, grant and revoke privileges etc.

The Data Manipulation Language (DML) commands, as the name suggests, are used to manipulate data. That is, DML commands query and manipulate data in existing schema objects.

3. *Differentiate between CHAR and VARCHAR datatypes.*

Solution. The difference between CHAR and VARCHAR is that of *fixed length* and *variable length*. The CHAR datatype specifies a *fixed length* character string. When a column is given datatype as CHAR(*n*), then MySQL ensures that all values stored in that column have this length *i.e.*, *n* bytes. If a value is shorter than this length *n* then blanks are added, but the size of value remains *n* bytes.

VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(*n*), then the maximum size a value in this column can have is *n* bytes. Each value that is stored in this column stores exactly as we specify it *i.e.*, no blanks are added if the length is shorter than maximum length *n*. However, if we exceed the maximum length *n*, then an error message is displayed.

4. *What is the "data type"? What are main objectives of datatypes ?*

Solution. Data type is defined as a set of values along with the operations that can be performed on those values.

Some common data types are *Integer, Float, Varchar, Char, String*, etc.

Main objectives of datatypes are :

- Optimum usage of storage space
- Represent all possible values
- Improve data integrity

5. *Consider the following table named "GARMENT". Write command of SQL for (i) to (iv).*

Table : GARMENT

GCODE	GNAME	SIZE	COLOUR	PRICE
111	TShirt	XL	Red	1400.00
112	Jeans	L	Blue	1600.00
113	Skirt	M	Black	1100.00
114	Ladies Jacket	XL	Blue	4000.00
115	Trousers	L	Brown	1500.00
116	Ladies Top	L	Pink	1200.00

(i) *To display names of those garments that are available in 'XL' size.*

(ii) *To display codes and names of those garments that have their names starting with 'Ladies'.*

(iii) *To display garment names, codes and prices of those garments that have price in the range 1000.00 to 1500.00 (both 1000.00 and 1500.00 included).*

(iv) *To change the colour of garment with code as 116 to "Orange".*

Scanned by CamScanner

PADHAAII

Solution.

(i)
```
SELECT GNAME
FROM GARMENT
WHERE SIZE = 'XL'
```

(ii)
```
SELECT GCODE, GNAME
FROM GARMENT
WHERE GNAME LIKE 'Ladies%' ;
```

(iii)
```
SELECT GNAME, GCODE, PRICE
FROM GARMENT
WHERE PRICE BETWEEN 1000.00 AND 1500.00 ;
```

(iv)
```
UPDATE garment
SET COLOUR = 'orange'
WHERE GCODE = 116 ;
```

6. *Consider the following table named "SOFTDRINK". Write commands of SQL for (i) to (iv).*

Table : SOFTDRINK

DRINKCODE	DNAME	PRICE	CALORIES
101	Lime and Lemon	20.00	120
102	Apple Drink	18.00	120
103	Nature Nectar	15.00	115
104	Green Mango	15.00	140
105	Aam Panna	20.00	135
106	Mango Juice Bahaar	12.00	150

(i) *To display names and drink codes of those drinks that have more than 120 calories.*

(ii) *To display drink codes, names and calories of all drinks, in descending order of calories.*

(iii) *To display names and price of drinks that have price in the range 12 to 18 (both 12 and 18 included).*

(iv) *Increase the price of all drinks in the given table by 10%.*

Solution.

(i)
```
SELECT DNAME, DRINKCODE
FROM SOFTDRINK
WHERE CALORIES > 120 ;
```

(ii)
```
SELECT DRINKCODE, DNAME, CALORIES
FROM SOFTDRINK
ORDER BY CALORIES DESC ;
```

(iii)
```
SELECT DNAME, PRICE
FROM SOFTDRINK
WHERE PRICE BETWEEN 12 and 18 ;
```

(iv)
```
UPDATE SOFTDRINK
SET PRICE = PRICE + 0.10 * PRICE ;
```

7. (a) *Write two examples of DBMS software.* [CBSE D 2014]

(b) *What is meant by NULL value in MySQL ?*

(c) *Table 'Club' has 4 rows and 3 columns. Table 'Member' has 2 rows and 5 columns. What will be the cardinality of the Cartesian product of them ?*

Solution.

(a) (i) Oracle (ii) SQL Server (iii) MySQL

(b) Null value signifies a legal empty value.

(c) 8

Scanned by CamScanner

PADHAAII

GLOSSARY

DataType	Means to identify type of data and associated operations for handling it.
Query	A command given to produce certain specified information from the database table(s).
DDL	Data Description Language Subset of SQL commands that are used to describe various entities of database.
DML	Data Manipulation Language Subset of SQL commands that are used to manipulate data in tables.
Join	A query that combines rows from one or more tables.

Consider a table Empl as given below. It will be required for many assignment questions.

empno	ename	job	mgr	hiredate	sal	comm	deptno
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30
8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	10
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20
8934	MITA	CLERK	8882	1992-01-23	1300.00	NULL	10

padhaalii

Assignment

1. How are SQL commands classified ?
2. Differentiate between DDL and DML commands.
3. (*a*) What is the use of UPDATE statement in SQL ? How is it different from ALTER statement ?
   (*b*) Mr. Shankar created a table VEHICLE with 3 rows and 4 columns. He added 1 more row to it and deleted one column. What is the Cardinality and Degree of the Table VEHICLE ?

Scanned by CamScanner

PADHAAII

(c) Consider the following table named "GYM" with details about fitness items being sold in the store. Write command of SQL for (*i*) to (*iv*).

Table : GYM

ICODE	INAME	PRICE	BRANDNAME
G101	Power Fit Exerciser	20000	Power Gymea
G102	Aquafit Hand Grip	1800	Reliable
G103	Cycle Bike	14000	Ecobike
G104	Protoner Extreme Gym	30000	Coscore
G105	Message Belt	5000	Message Expert
G106	Cross Trainer	13000	GTC Fitness

(*i*) To display the names of all the items whose name starts with "A".

(*ii*) To display ICODEs and INAMEs of all items, whose Brandname is Reliable or Coscore.

(*iii*) To change the Brandname to "Fit Trend India" of the item, whose ICODE as "G101".

(*iv*) Add a new row for new item in GYM with the details :

"G107", "Vibro exerciser", 21000, "GTCFitness"

[CBSE D 2013]

4. (*a*) Mr. James created a table **CLIENT** with 2 rows and 4 columns. He added 2 more rows to it and deleted one column. What is the Cardinality and Degree of the Table **CLIENT** ?

(*b*) Consider the following table **FITNESS** with details about fitness products being sold in the store. Write command of SQL for (*i*) to (*iv*).

Table : FITNESS

PCODE	PNAME	PRICE	Manufacturer
P1	Treadmill	21000	Coscore
P2	Bike	20000	Aone
P3	Cross Trainer	14000	Reliable
P4	Multi Gym	34000	Coscore
P5	Massage Chair	5500	Regrosene
P6	Belly Vibrator Belt	6500	Ambawya

(*i*) To display the names of all the products with price more than 20000.

(*ii*) To display the names of all products by the manufacturer "Aone".

(*iii*) To change the price data of all the products by applying 25% discount reduction.

(*iv*) To add a new row for product with the details :

"P7", "Vibro Exerciser", 28000, "Aone".

Scanned by CamScanner

PADHAAII

5. Write SQL commands for the following on the basis of given table CLUB. [CBSE OD 2013]

Table : CLUB

COACH_ID	COACHNAME	AGE	SPORTS	DATOFAPP	PAY	SEX
1.	KUKREJA	35	KARATE	27/03/1996	1000	M
2.	RAVINA	34	KARATE	20/01/1998	1200	F
3.	KARAN	34	SQUASH	19/02/1998	2000	M
4.	TARUN	33	BASKETBALL	01/01/1998	1500	M
5.	ZUBIN	36	SWIMMING	12/01/1998	750	M
6.	KETAKI	36	SWIMMING	24/02/1998	800	F
7.	ANKITA	39	SQUASH	20/02/1998	2200	F
8.	ZAREEN	37	KARATE	22/02/1998	1100	F
9.	KUSH	41	SWIMMING	13/01/1998	900	M
10.	SHAILYA	37	BASKETBALL	19/02/1998	1700	M

(*a*) To show all information about the swimming coaches in the club.

(*b*) To list names of all coaches with their date of appointment (DATOFAPP) in descending order.

(*c*) To display a report, showing coachname, pay, age and bonus (15% of pay) for all the coaches.

6. Write SQL commands for the following on the basis of given table STUDENT.

Table : STUDENT1

No.	Name	Stipend	Stream	AvgMark	Grade	Class
1	Karan	400.00	Medical	78.5	B	12B
2	Divakar	450.00	Commerce	89.2	A	11C
3	Divya	300.00	Commerce	68.6	C	12C
4	Arun	350.00	Humanities	73.1	B	12C
5	Sabina	500.00	Nonmedical	90.6	A	11A
6	John	400.00	Medical	75.4	B	12B
7	Robert	250.00	Humanities	64.4	C	11A
8	Rubina	450.00	Nonmedical	88.5	A	12A
9	Vikas	500.00	Nonmedical	92.0	A	12A
10	Mohan	300.00	Commerce	67.5	C	12C

(*a*) Select all the Nonmedical stream students from STUDENT1.

(*b*) List the names of those students who are in class 12 sorted by Stipend.

(*c*) List all students sorted by AvgMark in descending order.

Scanned by CamScanner

PADHAAII

7. What is foreign key ? How do you define a foreign key in your table ?
8. How is FOREIGN KEY commands different from PRIMARY KEY command ?
9. How is FOREIGN KEY commands related to the PRIMARY KEY ?
10. What are table constraints ? What are column constraints ? How are these *two* different ?
11. Insert all those records of table Accounts into table *Pending* where amt_outstanding is more than 10000.
12. Increase salary of employee recrods by 10% (*table employee*).
13. Add a constraint (NN-Grade) in table Empl (given before assignment) that declares column Grade not null.
14. Drop the table Empl.
15. Differentiate between :
    (*i*) DROP TABLE, DROP DATABASE
    (*ii*) DROP TABLE, DROP clause of ALTER TABLE.
16. Mr. Mittal is using a table with following columns :

    Name, Class, Stream_Id, Stream_name

    He needs to display names of students who have not been assigned any stream or have been assigned stream_name that ends with "computers".

    He wrote the following command, which did not give the desired result.

```
SELECT Name, Class FROM Students
WHERE Stream_name = NULL OR Stream_name = "%computers" ;
```

    Help Mr. Mittal to run the query by removing the error and write correct query. [CBSE D 12]

17. The Doc_name Column of a table Hospital is given below : [CBSE OD 11]

Doc_name
Avinash
Hariharan
Vinayak
Deepak
Sanjeev

    Based on the information, find the output of the following queries :
    (*i*) SELECT doc_name FROM HOSPITAL WHERE Doc_name like "%v";
    (*ii*) SELECT doc_name FROM HOSPITAL WHERE doc_name like "%e%";
18. Sarthak, a student of class XII, created a table "Class". Grade is one of the columns of this table. To find the details of students whose Grades have not been entered, he wrote the following MySql query, which did not give the desired result : [CBSE OD 11]

```
SELECT * FROM Class WHERE Grade = "Null";
```

    Help Sarthak to run the query by removing the errors from the query and write the correct query.
19. What is the purpose of DROP TABLE command in MySql ? How is it different from DELETE command ? [CBSE D 11]

Scanned by CamScanner