CMR Institute of Technology, Bengalore – 560 037
Department of Computer Applications
Scheme and Solutions for Internal Assessment Test I / II

18MCA31 – DATABASE MANAGEMENT SYSTEM **Semester / Section: III A** **Date of Test: 02-11-2020**

---

1. Bring out the different clauses of SELECT – FROM – WHERE – GROUP – HAVING with an example for each. (10)

i. The select command is used to retrieve a specified column or all the columns from the table name or from the system generated object.

Eg. Select sysdate from dual;

ii. i. The select command is used to retrieve a specified column or all the columns from the table name

Eg. Select * from student;

Select USN, studname from student;

iii. The select command with where clause is used to retrieve the columns from the table with the condition specified in the where clause.

Eg. Select * from instructor

Where salary > 50000;

iv. The select command offers to summarise the data based on some specific attribute. It is noted that the group by clause should have any one of the aggregate functions supported by the RDBMS.

Eg. Select deptid, sum(dalary) from instructor

Group by deptname;

v. The select command with having clause can be used to group the data for a specified condition.

Eg. Select deptid, avg(salary) from instructor

Group by deptname

Having count(deptid) > 2;

2. Explain SQL data definition and data types in brief and explain DROP and ALTER command. (10)

- **SQL Data Definition:**
- The set of relations in a database must be specified to the system by means of a data definition language (DDL).
- It allows specification of not only a set of relations, but also information about each relation, including :
- The schema for each relation.
- The types of values associated with each attribute.
- The integrity constraints.
- The set of indices to be maintained for each relation.
- The security and authorization information for each relation.
- The physical storage structure of each relation on disk
- Basic Types: The SQL standard supports a variety of built-in types, including:
  - Char(n): A fixed-length character string with user-specified length n
  - varchar(n): A variable-length character string with user-specified maximum length n
  - int: An integer (a finite subset of the integers that is machine dependent)
  - smallint: A small integer (a machine-dependent subset of the integer type)
  - numeric(p, d): A fixed-point number with user-specified precision.
  - real, double precision: Floating-point and double-precision floating-point numbers with machine-dependent precision

---

- float(n): A floating-point number, with precision of at least n digits
- The general form of the create table command is:
- create table r (A1 D1, A2 D2, . . . , An Dn, <integrity-constraint1>, . . . , <integrity-constraintk);
- **Data Types:**
- the SQL standard supports several data types relating to dates and times:
- date: A calendar date containing a (four-digit) year, month, and day of the month
- time: The time of day, in hours, minutes, and seconds
- timestamp: A combination of date and time
- Default Values
- SQL allows a default value to be specified for an attribute as illustrated by the following create table statement:
- Large-Object Types
- SQL provides large-object (LOB) data types for character data (clob) and binary data (blob).
- book_review clob(10KB)
- image blob(10MB)
- movie blob(2GB)
- User-Defined Types
- SQL supports two forms of user-defined data types.

(i) Distinct types

(ii) structured data types: allows the creation of complex data types with nested record structures, arrays, and multi-sets

(iii) Index Creation: An index on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation

3. Explain about all integrity constraints with examples. (10)

(i) primary key (Aj1, Aj2, . . . , Ajm):

The primarykey attributes are required to be non-null

no two tuples in the relation can be equal on all the primary-key attributes

(ii) foreign key (Ak1, Ak2, . . . , Akn) references s

specification says that the values of attributes (Ak1, Ak2, . . . , Akn) for any tuple in the relation must correspond to values of the primary key attributes of some tuple in relation s.

(iii) not null: constraint on an attribute specifies that the null value is not allowed for that attribute

(iv) default: set the default value for an attribute

(v) check: the check constraint checks the specified value to be assigned to the column

4. Give all four informal design guidelines for relation schemas. (10)

(i) Imparting Clear Semantics to Attributes in Relations

- Guideline -1:

Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation. If a relation schema corresponds to one entity type or one relationship type, it is straightforward to interpret and to explain its meaning. Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

(ii) Redundant Information in Tuples and Update Anomalies

- Guideline 2

Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

(iii) NULL Values in Tuples

- Guideline-3:

As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation

(iv) Generation of Spurious Tuples

- Guideline 4:

Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

5. What is normalization? Define 1st, 2nd, and BCNF. (10)

- Normalization of data can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of:
  - minimizing redundancy and
  - minimizing the insertion, deletion, and update anomalies
- Definition: The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized

(i) First Normal Form (1NF)

Definition: The relation is said to be in 1NF if all the attributes in a relation must have atomic domains

(ii) Second Normal Form (2NF)

Definition: A relation schema R is in 2NF if every non-prime attribute A in R is fully functionally dependent on the primary key of R

(iii) Third Normal Form (3NF)

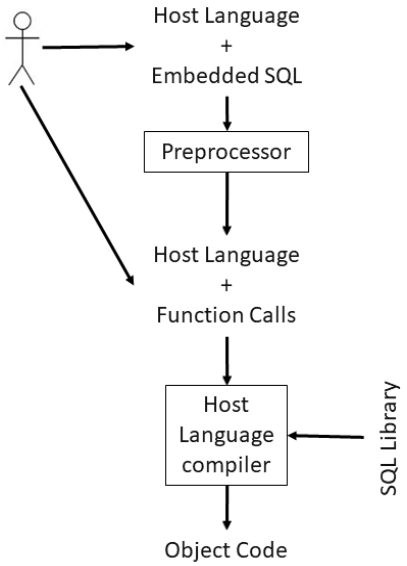
A relation schema R is in third normal form (3NF) if, whenever a nontrivial functional dependency X → A holds in R, either (a) X is a superkey of R, or (b) A is a prime attribute of R

(iv) Boyce-Codd Normal Form:

Definition. A relation schema R is in BCNF if whenever a nontrivial functional dependency X → A holds in R, then X is a superkey of R

6. Explain in detail about embedded SQL. (10)

- SQL statements can be embedded in a general-purpose programming language (Embedded SQL)
- Host language: the programming language used in the embedded approach (SQLJ for Java)
- It allows to execute any SQL statement from an application program
- an embedded SQL statement is distinguished from programming language statements by prefixing it with the keywords EXEC SQL so that a preprocessor (or precompiler) can separate embedded SQL statements from the host language code
- Shared variables: special variables declared in C program, within the embedded SQL command
- Connecting to the Database:
- CONNECT TO <server name> AS <connection name>
- AUTHORIZATION <user account name and password> ;
- SET CONNECTION <connection name> ;
- The two special communication variables that are used by the DBMS to communicate exception or error conditions to the program are (i) SQLCODE and (ii) SQLSTATE
- SQLCODE: After each database command is executed, the DBMS returns a value in SQLCODE
- SQLCODE = 0, the statement was executed successfully
- SQLCODE > 0, no more data are available in a query result
- SQLCODE < 0, some error has occurred
- In Oracle, SQLCODE is a field in a record structure called SQL Communication Area (SQLCA)
- the definition of SQLCA must be included in the C program by including the following line:
- EXEC SQL include SQLCA ;
- SQLSTATE: a communication variable added in the later versions of the SQL standard
- It is a string of five characters
- 00000: no error and 02000: no more data
- Retrieving Multiple Tuples with Embedded SQL Using Cursors



- a cursor as a pointer that points to a single tuple (row) from the result of a query that retrieves multiple tuples
- The cursor is declared when the SQL query command is declared in the program
- an OPEN CURSOR command fetches the query result from the database and sets the cursor to a position before the first row in the result of the query
- FETCH commands are issued in the program which moves the cursor to the next row in the result of the query, making it the current row and copying its attribute values into the C (host language) program variables specified in the FETCH command by an INTO clause
- CLOSE CURSOR command is issued to indicate that we are done with processing the result of the query associated with that cursor

7. Create the following table and perform the tasks as specified below: (10)
ACCOUNT(account_number, name, account_type, contact_number, balance)
TRANSACTION(transaction_number, transaction_date, transaction_type, amount, account_number)
i. Create the table with necessary integrity constraints
ii. account_type should be any one of: {savings bank, current account}
iii. transaction_type should be any one of: {withdrawal, deposit}
iv. set the transaction_date should be the current date.
v. The balance should be minimum 5000
vi. Count the number of transactions
vii. Count the number of unique account holders
viii. Count the number of customers based on account_type
ix. Find the account holder's name who maintains the maximum balance
x. Display the transactions based on transaction_type.

i. create table account
(account_number number(10) primary key,
name varchar2(20),
account_type char(2),
contact_number number(10),
balance number(7,2)
);
create table transaction
(transaction_number number(10) primary key,
transaction_date date,
transaction_type char(3),
amount number(10,2),
account_number number(10),
foreign key account_number references account(account_number)
);
ii. alter table account
add constraint const1 check acctype in ('sb', 'ca');

iii. alter table transaction
add constraint const2 check transaction_type in ('wdl', 'dep');
iv. alter table transaction add transdate date default sysdate;
v. alter table account add balance number(10)
check (balance > 5000);
vi. select count(*) from transaction;
vii. select distinct(count(name)) from account;
viii. select account_type, count(*) from account
group by account_type
order by account_type;
ix. select account_number, max(blance) from account;
x. select * from transaction
where transaction_type = 'wdl';

8. Create the table named 'employee' with the attributes: empno, empname, dept, designation, salary, doj, place. Execute the following queries: (10)
i. Display all the fields of employee table.
ii. Display details of employee number and their salary.
iii. Display average salary of all employee.
iv. Display name of employee in descending order.
v. Display distinct number of employee.
vi. Display the employees whose name contains second and third letters as 'um'.
vii. Display total salary of employee group by employee designation.
viii. Display total salary of employee which is greater than 120000.
ix. Display details of employee whose name is 'Amit' and salary greater than 50000.
x. Find department-wise minimum salary, maximum salary, average salary and sum of salary. Display the output in the ascending order of the department name.

i. select * from employee;
ii. select empno, salary from employee;
iii. select avg(salary) from employee;
iv. select empname from employee
order by empname desc;
v. select count(distinct(empname)) from employee;
vi. select * from employee where empname like '_um%';
vii. select dept, sum(salary) from employee group by dept order by dept;
viii. select emp_name, sum(salary) from employee
group by(emp_name) having sum(salary)>120000;
ix. select * from employee
where emp_name='Amit' and salary>50000;
x. select dept, sum(salary), min(salary), max(salary), avg(salary)
from employee group by dept order by dept;