**Programming 14, 15, 16, & 17**

**Database Programming with SQL  
14-1: Intro to Constraints; NOT NULL and UNIQUE Constraints  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **Unique Constraint** | Every value in a column or set of columns (a composite key) must be unique |
| **NOT NULL Constraint** | For every row entered into the table, there must be a value for that column |
| **Primary Key Constraint** | Constraint ensures that the column contains no null values and uniquely identifies each row of the table |
| **Check Constraint** | Specifies a condition for a column that must be true for each row of data |
| **Foreign Key** | Identifies that table and column in the parent table |
| **Unique Constraint** | An integrity constraint that requires every value in a column or set of columns be unique |
| **Foreign Key Constraint** | Designates a column (child table) that establishes a relationship between a primary key in the same table and a different table (parent table) |
| **Table-Level Constraint** | References one or more columns and is defined separately from the definitions of the columns in the table |
| **Constraint** | Database rule. |
| **Column-Level Constraint** | Database rule that references a single column |

Try It / Solve It  
Global Fast Foods has been very successful this past year and has opened several new stores. They need to add a table to their database to store information about each of their store’s locations. The owners want to make sure that all entries have an identification number, date opened, address, and city and that no other entry in the table can have the same email address. Based on this information, answer the following questions about the global\_locations table. Use the table for your answers.

1. What is a “constraint” as it relates to data integrity?

A constraint is a rule applied to a database column or table to enforce data integrity. Constraints ensure that the data adheres to specific rules or conditions, such as ensuring uniqueness, preventing null values, or maintaining relationships between tables.

1. What are the limitations of constraints that may be applied at the column level and at the table level?

 **Column-level constraints**: Can only be applied to a single column during its definition. They are limited to rules affecting that specific column.

 **Table-level constraints**: Can involve multiple columns and are defined separately from the column definitions, providing more flexibility for composite constraints.

1. Why is it important to give meaningful names to constraints?

Meaningful names make it easier to understand the purpose of constraints, simplify debugging when an error occurs, and improve database documentation and maintenance.

1. Based on the information provided by the owners, choose a datatype for each column. Indicate the length, precision, and scale for each NUMBER datatype.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Datatype** | **Length/Precision** | **Nullable** | **Description** |
| id | NUMBER | 10 | NOT NULL | Store identification number, primary key |
| name | VARCHAR2 | 100 | NOT NULL | Store name |
| date\_opened | DATE | - | NOT NULL | Opening date |
| address | VARCHAR2 | 200 | NOT NULL | Store address |
| city | VARCHAR2 | 100 | NOT NULL | City where the store is located |
| Zip\_postal\_code | VARCHAR2 | 10 | NULL | ZIP or postal code |
| phone | VARCHAR2 | 15 | NULL | Contact phone number |
| email | VARCHAR2 | 100 | NOT NULL UNIQUE | Email address, must be unique |
| Manager\_id | NUMBER | 10 | NULL | Manager identification number |
| Emergency\_contact | VARCHAR2 | 15 | NULL | Emergency contact number |

1. Use “nullable” to indicate those columns that can have null values.

 Columns that **cannot** have null values: id, name, date\_opened, address, city, email.

 Columns with null values: zip\_postal\_code, phone, manager\_id, emergency\_contact.

1. Write the CREATE TABLE statement for the Global Fast Foods locations table to define the constraints at the column level.

CREATE TABLE global\_locations (

id NUMBER(10) CONSTRAINT pk\_global\_locations\_id PRIMARY KEY,

name VARCHAR2(100) NOT NULL,

date\_opened DATE NOT NULL,

address VARCHAR2(200) NOT NULL,

city VARCHAR2(100) NOT NULL,

zip\_postal\_code VARCHAR2(10),

phone VARCHAR2(15),

email VARCHAR2(100) NOT NULL CONSTRAINT uq\_global\_locations\_email UNIQUE,

manager\_id NUMBER(10),

emergency\_contact VARCHAR2(15)

);

1. Execute the CREATE TABLE statement in Oracle Application Express.

A screenshot of a computer program

Description automatically generated

1. Execute a DESCRIBE command to view the Table Summary information.

A screenshot of a computer program

Description automatically generated

1. Rewrite the CREATE TABLE statement for the Global Fast Foods locations table to define the UNIQUE constraints at the table level. Do not execute this statement.

CREATE TABLE global\_locations (

id NUMBER(10) CONSTRAINT pk\_global\_locations\_id PRIMARY KEY,

name VARCHAR2(100) NOT NULL,

date\_opened DATE NOT NULL,

address VARCHAR2(200) NOT NULL,

city VARCHAR2(100) NOT NULL,

zip\_postal\_code VARCHAR2(10),

phone VARCHAR2(15),

email VARCHAR2(100) NOT NULL,

manager\_id NUMBER(10),

emergency\_contact VARCHAR2(15),

CONSTRAINT uq\_global\_locations\_email UNIQUE (email)

);

**Database Programming with SQL  
14-2: PRIMARY KEY, FOREIGN KEY, and CHECK Constraints  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **ON DELETE CASCADE** | Allows a foreign key row that is referenced to a primary key row to be deleted |
| **Check Constraint** | Explicitly defines a condition that must be met |
| **Primary Key** | A column or set of columns that uniquely identifies each row in a table |
| **NOT NULL Constraint** | Constraint ensures that the column contains no null values |
| **ON DELETE SET NULL** | Allows a child row to remain in a table with null values when a parent record has been deleted |
| **Foreign Key** | Establishes a relationship between the foreign key column and a primary key or unique key in the same table or a different table |

Try It / Solve It

1. What is the purpose of a  
   a. PRIMARY KEY

A PRIMARY KEY uniquely identifies each row in a table. It ensures no duplicate values and prevents NULL values in the specified columns.

b. FOREIGN KEY

A FOREIGN KEY creates a relationship between two tables by linking a column in the child table to a PRIMARY KEY in the parent table. It ensures referential integrity.

c. CHECK CONSTRAINT

A CHECK CONSTRAINT enforces a condition that all values in a column must satisfy. For example, it can ensure that a column only contains positive numbers or dates within a certain range.

1. Using the column information for the animals table below, name constraints where applicable at the table level, otherwise name them at the column level. Define the primary key (animal\_id). The license\_tag\_number must be unique. The admit\_date and vaccination\_date columns cannot  
   contain null values.

CREATE TABLE animals (

animal\_id NUMBER(6) CONSTRAINT pk\_animals PRIMARY KEY,

name VARCHAR2(25),

license\_tag\_number NUMBER(10) CONSTRAINT uq\_license\_tag UNIQUE,

admit\_date DATE NOT NULL,

adoption\_id NUMBER(5),

vaccination\_date DATE NOT NULL

);

1. Create the animals table. Write the syntax you will use to create the table.

CREATE TABLE animals (

animal\_id NUMBER(6) CONSTRAINT pk\_animals PRIMARY KEY,

name VARCHAR2(25),

license\_tag\_number NUMBER(10) CONSTRAINT uq\_license\_tag UNIQUE,

admit\_date DATE NOT NULL,

adoption\_id NUMBER(5),

vaccination\_date DATE NOT NULL

);

1. Enter one row into the table. Execute a SELECT \* statement to verify your input. Refer to the graphic below for input.

INSERT INTO animals (animal\_id, name, license\_tag\_number, admit\_date, adoption\_id, vaccination\_date)

VALUES (101, 'Spot', 35540, TO\_DATE('10-Oct-2004', 'DD-Mon-YYYY'), 205, TO\_DATE('12-Oct-2004', 'DD-Mon-YYYY'));

A screenshot of a computer program

Description automatically generated

1. Write the syntax to create a foreign key (adoption\_id) in the animals table that has a  
   corresponding primary- key reference in the adoptions table. Show both the column-level and table-level syntax. Note that because you have not actually created an adoptions table, no adoption\_id primary key exists, so the foreign key cannot be added to the animals table.

Column-Level Syntax:

CREATE TABLE animals (

animal\_id NUMBER(6) CONSTRAINT pk\_animals PRIMARY KEY,

name VARCHAR2(25),

license\_tag\_number NUMBER(10) CONSTRAINT uq\_license\_tag UNIQUE,

admit\_date DATE NOT NULL,

adoption\_id NUMBER(5) CONSTRAINT fk\_adoptions REFERENCES adoptions(adoption\_id),

vaccination\_date DATE NOT NULL

);

Table-Level Syntax:

ALTER TABLE animals ADD CONSTRAINT fk\_adoptions FOREIGN KEY (adoption\_id) REFERENCES adoptions(adoption\_id);

1. What is the effect of setting the foreign key in the ANIMAL table as:  
   a. ON DELETE CASCADE

If a referenced row in the parent table (adoptions) is deleted, all rows in the child table (animals) that reference it will also be automatically deleted.

b. ON DELETE SET NULL

If a referenced row in the parent table (adoptions) is deleted, the foreign key value (adoption\_id) in the child table (animals) will be set to NULL.

1. What are the restrictions on defining a CHECK constraint?

 CHECK constraints cannot reference columns in other tables. They are limited to conditions within the same row of the table.

 Expressions in a CHECK constraint must evaluate to a Boolean result (e.g., salary > 0).

 The condition cannot include subqueries, sequence values, or PL/SQL functions.

 A column with a CHECK constraint cannot violate the condition at the time of insertion or update.

**Database Programming with SQL  
14-3: Managing Constraints  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **DISABLE** | To deactivate an integrity constraint |
| **CASCADE** | Disables dependent integrity constraints |
| **ALTER** | To add, modify, or drop columns from a table |
| **ENABLE** | To activate an integrity constraint currently disabled |
| **DROP** | Removes a constraint from a table |
| **ALTER** | Allows user to delete a column from a table |
| **ON DELETE/UPDATE Action** | Defines the actions the database server takes when a user attempts to delete or update a key to which existing foreign keys point |

Try It / Solve It

Using Oracle Application Express, click the SQL Workshop tab in the menu bar. Click the Object  
Browser and verify that you have a table named copy\_d\_clients and a table named copy\_d\_events. If you don’t have these tables in your schema, create them before completing the exercises below. Here is how the original tables are related. The d\_clients table has a primary key client\_number. This has a primary-key constraint and it is referenced in the foreign-key constraint on the d\_events table.

1. What are four functions that an ALTER statement can perform on constraints?

* Add a new constraint to an existing table.
* Drop an existing constraint from a table.
* Disable a constraint to deactivate its enforcement.
* Enable a constraint to reactivate its enforcement.

2. Since the tables are copies of the original tables, the integrity rules are not passed onto the new  
tables; only the column datatype definitions remain. You will need to add a PRIMARY KEY  
constraint to the copy\_d\_clients table. Name the primary key copy\_d\_clients\_pk . What is the  
syntax you used to create the PRIMARY KEY constraint to the copy\_d\_clients.table?

ALTER TABLE copy\_d\_clients

ADD CONSTRAINT copy\_d\_clients\_pk PRIMARY KEY (client\_number);

3. Create a FOREIGN KEY constraint in the copy\_d\_events table. Name the foreign key  
copy\_d\_events\_fk. This key references the copy\_d\_clients table client\_number column. What is  
the syntax you used to create the FOREIGN KEY constraint in the copy\_d\_events table?

ALTER TABLE copy\_d\_events

ADD CONSTRAINT copy\_d\_events\_fk FOREIGN KEY (client\_number)

REFERENCES copy\_d\_clients (client\_number);

4. Use a SELECT statement to verify the constraint names for each of the tables. Note that the tablenames must be capitalized.

SELECT constraint\_name, constraint\_type, table\_name

FROM user\_constraints

WHERE table\_name IN ('COPY\_D\_CLIENTS', 'COPY\_D\_EVENTS');

A screenshot of a computer program

Description automatically generated

a. The constraint name for the primary key in the copy\_d\_clients table is \_\_\_COPY\_D\_CLIENTS\_PK\_\_\_\_\_\_\_\_\_\_\_\_.  
b. The constraint name for the foreign key in the copy\_d\_events table is \_\_\_\_COPY\_D\_EVENTS\_FK\_\_\_\_\_\_\_\_\_\_\_.

5. Drop the PRIMARY KEY constraint on the copy\_d\_clients table. Explain your results.

ALTER TABLE copy\_d\_clients

DROP CONSTRAINT copy\_d\_clients\_pk;

**Explanation:**

Dropping the PRIMARY KEY removes the enforcement of the unique and non-null requirement for the client\_number column. This allows duplicate or null values in the column.

6. Add the following event to the copy\_d\_events table. Explain your results.  
ID NAME EVENT\_DATE DESCRIPTION COST VENUE

INSERT INTO copy\_d\_events (id, name, event\_date, description, cost, venue\_id, package\_code, theme\_code, client\_number)

VALUES (140, 'Cline Bas Mitzvah', TO\_DATE('15-Jul-2004', 'DD-Mon-YYYY'), 'Church and Private Home', 4500, 105, 87, 77, 7125);

**Explanation:**  
This query will fail if the foreign key copy\_d\_events\_fk is enabled and the value 7125 for client\_number does not exist in the copy\_d\_clients table.  
  
7. Create an ALTER TABLE query to disable the primary key in the copy\_d\_clients table. Then add the values from #6 to the copy\_d\_events table. Explain your results.

ALTER TABLE copy\_d\_clients

DISABLE CONSTRAINT copy\_d\_clients\_pk;

INSERT INTO copy\_d\_events (id, name, event\_date, description, cost, venue\_id, package\_code, theme\_code, client\_number)

VALUES (140, 'Cline Bas Mitzvah', TO\_DATE('15-Jul-2004', 'DD-Mon-YYYY'), 'Church and Private Home', 4500, 105, 87, 77, 7125);

**Explanation:**  
Disabling the PRIMARY KEY in the parent table (copy\_d\_clients) does not affect the foreign key relationship. If 7125 is not a valid client\_number in copy\_d\_clients, the query will still fail.

8. Repeat question 6: Insert the new values in the copy\_d\_events table. Explain your results

INSERT INTO copy\_d\_events (id, name, event\_date, description, cost, venue\_id, package\_code, theme\_code, client\_number)

VALUES (140, 'Cline Bas Mitzvah', TO\_DATE('15-Jul-2004', 'DD-Mon-YYYY'), 'Church and Private Home', 4500, 105, 87, 77, 7125);

**Explanation:**  
If the foreign key is still enabled, the insert will fail unless client\_number = 7125 exists in the copy\_d\_clients table.

.  
9. Enable the primary-key constraint in the copy\_d\_clients table. Explain your results

ALTER TABLE copy\_d\_clients

ENABLE CONSTRAINT copy\_d\_clients\_pk;

**Explanation:**  
The PRIMARY KEY constraint is now reactivated, ensuring that all client\_number values in copy\_d\_clients are unique and non-null.

10. If you wanted to enable the foreign-key column and reestablish the referential integrity between these two tables, what must be done?

**Steps to reenable foreign key and reestablish integrity:**

* Ensure all client\_number values in copy\_d\_events exist in copy\_d\_clients.
* Enable the foreign key constraint:

11. Why might you want to disable and then re-enable a constraint?

* To temporarily allow operations that violate the constraint, such as bulk data imports or updates.
* To troubleshoot errors related to constraints.
* To perform maintenance tasks like removing or updating related records.

12. Query the data dictionary for some of the constraints that you have created. How does the data dictionary identify each constraint type?

SELECT constraint\_name, constraint\_type, status, table\_name

FROM user\_constraints

WHERE table\_name IN ('COPY\_D\_CLIENTS', 'COPY\_D\_EVENTS');

The constraint\_type column identifies each constraint type:

* P: PRIMARY KEY
* R: FOREIGN KEY
* C: CHECK CONSTRAINT
* U: UNIQUE CONSTRAINT

**Database Programming with SQL  
15-1: Creating Views  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **View** | A subset of data from one or more tables that is generated from a query and stored as a virtual table |
| **View name** | Name of view |
| **Create or Replace View** | Creates a view regardless of whether or not the base tables exist |
| **Simple view** | Derives data from a table, no functions or groups, performs DML operations through the view |
| **Create View** | Creates the view only if the base table exists |
| **Create View** | Statement used to create a new view |
| **Column alias** | Specifies a name for each expression selected by the view’s query |
| **Subquery** | A complete SELECT statement |
| **Complex View** | Derives data from more than one table, contains functions or groups of data, and does not always allow DML operations through the view |
| **Create or Replace View** | Re-creates the view if it already exists |

Try It / Solve It

1. What are three uses for a view from a DBA’s perspective?

* **Data Security**: Restrict access to specific columns or rows of a table by exposing only the required subset of data.
* **Simplification**: Simplify complex queries by creating a reusable view.
* **Data Aggregation**: Provide summarized or aggregated data using group functions for business analysis.

2. Create a simple view called view\_d\_songs that contains the ID, title, and artist from the DJs on Demand table for each “New Age” type code. In the subquery, use the alias “Song Title” for the title column.

CREATE VIEW view\_d\_songs AS

SELECT id, title AS "Song Title", artist

FROM djs\_on\_demand

WHERE type\_code = 'New Age';

3. SELECT \*  
FROM view\_d\_songs.

What was returned?

A screenshot of a computer

Description automatically generated

4. REPLACE view\_d\_songs. Add type\_code to the column list. Use aliases for all columns.

CREATE OR REPLACE VIEW view\_d\_songs AS

SELECT id AS "Song ID", title AS "Song Title", artist AS "Artist Name", type\_code AS "Type Code"

FROM djs\_on\_demand;

5. Jason Tsang, the disk jockey for DJs on Demand, needs a list of the past events and those planned for the coming months so he can make arrangements for each event’s equipment setup.  
As the company manager, you do not want him to have access to the price that clients paid for  
their events. Create a view for Jason to use that displays the name of the event, the event date,  
and the theme description. Use aliases for each column name.

CREATE VIEW view\_events\_for\_jason AS

SELECT event\_name AS "Event Name", event\_date AS "Event Date", theme\_description AS "Theme"

FROM events;

6. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that  
displays the needed information for department managers.

CREATE VIEW view\_department\_salaries AS

SELECT department\_id AS "Department ID",

MIN(salary) AS "Minimum Salary",

MAX(salary) AS "Maximum Salary",

AVG(salary) AS "Average Salary"

FROM employees

GROUP BY department\_id;

**Database Programming with SQL  
15-2: DML Operations and Views  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **ROWNUM** | A pseudocolumn which assigns a sequential value starting with 1 to each of the rows returned from the subquery |
| **WITH CHECK OPTION** | Specifies that INSERTS and UPDATES performed through the view can’t create rows which the view cannot select |
| **WITH READ ONLY** | Ensures that no DML operations can be performed on this view |

Try It / Solve It  
Use the DESCRIBE statement to verify that you have tables named copy\_d\_songs, copy\_d\_events, copy\_d\_cds, and copy\_d\_clients in your schema. If you don't, write a query to create a copy of each.

1. Query the data dictionary USER\_UPDATABLE\_COLUMNS to make sure the columns in the base tables will allow UPDATE, INSERT, or DELETE. Use a SELECT statement. All table names in the data dictionary are stored in uppercase.

A screenshot of a computer

Description automatically generated

2. Use the CREATE or REPLACE option to create a view of all the columns in the copy\_d\_songs table called view\_copy\_d\_songs.

CREATE OR REPLACE VIEW view\_copy\_d\_songs AS

SELECT \*

FROM copy\_d\_songs;

3. Use view\_copy\_d\_songs to INSERT the following data into the underlying copy\_d\_songs table. Execute a SELECT \* from copy\_d\_songs to verify your DML command. See the graphic.

A screenshot of a computer

Description automatically generated

4. Create a view based on the DJs on Demand COPY\_D\_CDS table. Name the view  
read\_copy\_d\_cds. Select all columns to be included in the view. Add a WHERE clause to restrict  
the year to 2000. Add the WITH READ ONLY option.

CREATE OR REPLACE VIEW read\_copy\_d\_cds AS

SELECT \*

FROM copy\_d\_cds

WHERE year = 2000

WITH READ ONLY;

5. Using the read\_copy\_d\_cds view, execute a DELETE FROM read\_copy\_d\_cds WHERE  
cd\_number = 90;

A screenshot of a computer

Description automatically generated

6. Use REPLACE to modify read\_copy\_d\_cds. Replace the READ ONLY option with WITH CHECK OPTION CONSTRAINT ck\_read\_copy\_d\_cds. Execute a SELECT \* statement to verify that the view exists.

A screenshot of a computer program

Description automatically generated

7. Use the read\_copy\_d\_cds view to delete any CD of year 2000 from the underlying copy\_d\_cds.

DELETE FROM read\_copy\_d\_cds WHERE year = 2000;

8. Use the read\_copy\_d\_cds view to delete cd\_number 90 from the underlying copy\_d\_cds table.

DELETE FROM read\_copy\_d\_cds WHERE cd\_number = 90;

9. Use the read\_copy\_d\_cds view to delete year 2001 records.

DELETE FROM read\_copy\_d\_cds WHERE year = 2001;

10. Execute a SELECT \* statement for the base table copy\_d\_cds. What rows were deleted?

A screenshot of a computer

Description automatically generated

11. What are the restrictions on modifying data through a view?

* A view must be based on a single table with no group functions or aggregate operations.
* Columns in the view must allow updates in the base table.
* WITH READ ONLY prevents DML operations through the view.
* Views with WITH CHECK OPTION restrict DML operations to only rows visible in the view.

12. What is Moore’s Law? Do you consider that it will continue to apply indefinitely? Support your opinion with research from the internet.

Moore's Law states that the number of transistors on a microchip doubles approximately every two years, resulting in exponential growth in computing power.

**Will it continue indefinitely?** No, Moore’s Law is approaching physical and economic limits due to:

* Miniaturization limits of silicon-based chips.
* Increased costs of manufacturing.
* The rise of alternative technologies like quantum computing and neuromorphic chips.

13. What is the “singularity” in terms of computing?

The "singularity" is the idea that there could come a time when artificial intelligence becomes smarter than humans. At that point, AI might be able to improve itself without our help, which could lead to swift advancements in technology and changes that are hard to predict

**Database Programming with SQL  
15-3: Managing Views  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **Top-N Analysis** | Asks for the N largest or smallest values in a column |
| **DROP VIEW** | Removes a view |
| **Inline View** | Subquery with an alias that can be used within a SQL statement |

Try It / Solve It  
1. Create a view from the copy\_d\_songs table called view\_copy\_d\_songs that includes only the title and artist. Execute a SELECT \* statement to verify that the view exists.

CREATE VIEW view\_copy\_d\_songs AS

SELECT title, artist

FROM copy\_d\_songs;

-- Verify the view

SELECT \* FROM view\_copy\_d\_songs;

A screenshot of a computer

Description automatically generated

2. Issue a DROP view\_copy\_d\_songs. Execute a SELECT \* statement to verify that the view has  
been deleted.

DROP VIEW view\_copy\_d\_songs;

-- Verify the view is deleted

SELECT \* FROM view\_copy\_d\_songs;

3. Create a query that selects the last name and salary from the Oracle database. Rank the salaries  
from highest to lowest for the top three employees.

SELECT last\_name, salary

FROM (

SELECT last\_name, salary, RANK() OVER (ORDER BY salary DESC) AS rank

FROM employees

)

WHERE rank <= 3;

A screenshot of a computer program

Description automatically generated

4. Construct an inline view from the Oracle database that lists the last name, salary, department

ID, and maximum salary for each department. Hint: One query will need to calculate maximum salary by department ID.

SELECT e.last\_name, e.salary, e.department\_id, d.max\_salary

FROM employees e

JOIN (

SELECT department\_id, MAX(salary) AS max\_salary

FROM employees

GROUP BY department\_id

) d

ON e.department\_id = d.department\_id;

A screenshot of a computer program

Description automatically generated

5. Create a query that will return the staff members of Global Fast Foods ranked by salary from  
lowest to highest.

SELECT staff\_name, salary, RANK() OVER (ORDER BY salary ASC) AS rank

FROM global\_fast\_foods\_staff;

**Extension Exercises**

1. Create a new table called my\_departments and add all columns and all rows to it using a  
subquery from the Oracle departments table. Do a SELECT \* from my\_departments to confirm  
that you have all the columns and rows.

CREATE TABLE my\_departments AS

SELECT \* FROM departments;

-- Verify the data

SELECT \* FROM my\_departments;

A screenshot of a computer

Description automatically generated

2. To view any constraints that may affect the my\_departments table, DESCRIBE my\_departments to check if any constraints were carried over from the departments table. If there are constraints on my\_departments, use an ALTER TABLE command to DISABLE all constraints on my\_departments.

DESCRIBE my\_departments;

-- Disable all constraints

ALTER TABLE my\_departments DISABLE CONSTRAINT ALL;

SELECT constraint\_name, constraint\_type

FROM user\_constraints

WHERE table\_name = 'MY\_DEPARTMENTS';

A screenshot of a computer

Description automatically generated

No constraints were copied when the table was created.

3. Create a view called view\_my\_departments that includes: department\_id and department\_name.

CREATE VIEW view\_my\_departments AS

SELECT department\_id, department\_name

FROM my\_departments;

4. Add the following data to the my\_departments table using view\_my\_departments.

INSERT INTO view\_my\_departments (department\_id, department\_name)

VALUES (105, 'Advertising');

INSERT INTO view\_my\_departments (department\_id, department\_name)

VALUES (120, 'Custodial');

INSERT INTO view\_my\_departments (department\_id, department\_name)

VALUES (130, 'Planning');

5. Create or enable the department\_id column as the primary key.

ALTER TABLE my\_departments ADD CONSTRAINT pk\_department\_id PRIMARY KEY (department\_id);

6. Enter a new department named Human Resources into the my\_departments table using  
view\_my\_departments. Do not add a new department ID.

INSERT INTO view\_my\_departments (department\_id, department\_name)

VALUES (220, 'Human Resources');

7. Add the Human Resources department, department ID 220, to my\_departments using  
view\_my\_departments.

INSERT INTO view\_my\_departments (department\_id, department\_name)

VALUES (220, 'Human Resources');

8. Verify that the new additions to my\_departments were added using view\_my\_departments.  
See chart below

A screenshot of a computer

Description automatically generated

9. Modify view\_my\_departments to include location ID. Do a SELECT \* command to show what columns are present and a DESCRIBE command to view the columns and associated constraints.

CREATE OR REPLACE VIEW view\_my\_departments AS

SELECT department\_id, department\_name, location\_id

FROM my\_departments;

-- Verify the updated view

SELECT \* FROM view\_my\_departments;

DESCRIBE view\_my\_departments;

A screenshot of a computer

Description automatically generated

10. Make location\_id a NOT NULL column in the my\_departments table.

ALTER TABLE my\_departments MODIFY location\_id NOT NULL;

11. Using the Oracle database, create a complex view between locations and departments with only the following columns: department\_name, street\_address, city, and state. Include only U.S. cities. Verify that the view was created using a SELECT \* statement

CREATE VIEW view\_locations\_departments AS

SELECT d.department\_name, l.street\_address, l.city, l.state

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

WHERE l.country\_id = 'US';

-- Verify the view

SELECT \* FROM view\_locations\_departments;

**Database Programming with SQL  
16-1: Working with Sequences  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **SEQUENCE** | Command that automatically generates sequential numbers |
| **NEXTVAL** | Generates a numeric value |
| **NEXTVAL** | Returns the next available sequence value |
| **INCREMENT BY** | Specifies the interval between sequence numbers |
| **MAXVALUE** | Specifies a maximum value of 10^27 for an ascending sequence and -1 for a descending sequence (default) |
| **CURRVAL** | returns the current sequence value |
| **MINVALUE** | specifies the minimum sequence value |
| **CYCLE / NOCYCLE** | specifies whether the sequence continues to generate values after reaching its maximum or minimum values |
| **MINVALUE** | specifies a minimum value of 1 for an ascending sequence and – (10^26) for a descending sequence (default) |
| **MAXVALUE** | specifies a maximum or default value the sequence can generate |
| **START WITH** | specifies the first sequence number to be generated |
| **CACHE / NOCACHE** | specifies how many values the Server pre-allocates and keeps in memory |

Try It / Solve It

1. Using CREATE TABLE AS subquery syntax, create a seq\_d\_songs table of all the columns in the DJs on Demand database table d\_songs. Use the SELECT \* in the subquery to make sure that  
you have copied all of the columns.

CREATE TABLE seq\_d\_songs AS

SELECT \*

FROM d\_songs;

2. Because you are using copies of the original tables, the only constraints that were carried over  
were the NOT NULL constraints. Create a sequence to be used with the primary-key column of  
the seq\_d\_songs table. To avoid assigning primary-key numbers to these tables that already  
exist, the sequence should start at 100 and have a maximum value of 1000. Have your sequence  
increment by 2 and have NOCACHE and NOCYCLE. Name the sequence seq\_d\_songs\_seq.

CREATE SEQUENCE seq\_d\_songs\_seq

START WITH 100

INCREMENT BY 2

MAXVALUE 1000

NOCACHE

NOCYCLE;

3. Query the USER\_SEQUENCES data dictionary to verify the seq\_d\_songs\_seq SEQUENCE  
settings.

SELECT \*

FROM USER\_SEQUENCES

WHERE SEQUENCE\_NAME = 'SEQ\_D\_SONGS\_SEQ';

A screen shot of a computer

Description automatically generated

4. Insert two rows into the seq\_d\_songs table. Be sure to use the sequence that you created for the ID column. Add the two songs shown in the graphic.

INSERT INTO seq\_d\_songs (ID, TITLE, DURATION, ARTIST, TYPE\_CODE)

VALUES (seq\_d\_songs\_seq.NEXTVAL, 'Island Fever', '5 min', 'Hawaiian Islanders', 12);

INSERT INTO seq\_d\_songs (ID, TITLE, DURATION, ARTIST, TYPE\_CODE)

VALUES (seq\_d\_songs\_seq.NEXTVAL, 'Castle of Dreams', '4 min', 'The Wanderers', 77);

5. Write out the syntax for seq\_d\_songs\_seq to view the current value for the sequence. Use the  
DUAL table. (Oracle Application Developer will not run this query.)

SELECT seq\_d\_songs\_seq.CURRVAL

FROM DUAL;

A screenshot of a computer program

Description automatically generated

6. What are three benefits of using SEQUENCEs?

**Benefits of Using Sequences**

* Ensures unique values for primary keys.
* Eliminates manual number assignment errors.
* Improves performance by automating number generation.

7. What are the advantages of caching sequence values?

**Advantages of Caching Sequence Values**

* Reduces I/O operations.
* Enhances performance by storing sequence values in memory.
* Minimizes contention in multi-user environments.

8. Name three reasons why gaps may occur in a sequence?

**Reasons for Gaps in a Sequence**

* Rollback of a transaction that generated a sequence value.
* A sequence being used in multiple sessions or transactions concurrently.
* Sequence values not reused after being cached in memory.

**Extension Exercise**

1. Create a table called “students”. You can decide which columns belong in that table and what  
datatypes these columns require. (The students may create a table with different columns;  
however, the important piece that must be there is the student\_id column with a numeric datatype. This column length must allow the sequence to fit, e.g. a column length of 4 with a sequence that starts with 1 and goes to 10000000 will not work after student #9999 is entered.)

CREATE TABLE students (

student\_id NUMBER(6),

first\_name VARCHAR2(50),

last\_name VARCHAR2(50),

enrollment\_date DATE

);

2. Create a sequence called student\_id\_seq so that you can assign unique student\_id numbers for  
all students that you add to your table.

CREATE SEQUENCE student\_id\_seq

START WITH 1

INCREMENT BY 1

MAXVALUE 999999

NOCACHE

NOCYCLE;

3. Now write the code to add students to your STUDENTS table, using your sequence “database  
object.”

INSERT INTO students (student\_id, first\_name, last\_name, enrollment\_date)

VALUES (student\_id\_seq.NEXTVAL, 'John', 'Doe', SYSDATE);

INSERT INTO students (student\_id, first\_name, last\_name, enrollment\_date)

VALUES (student\_id\_seq.NEXTVAL, 'Jane', 'Smith', SYSDATE);

**Database Programming with SQL  
16-2: Indexes and Synonyms  
Practice Activities**

Vocabulary  
Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **USER\_INDEXES** | Confirms the existence of indexes from the USER\_INDEXES data dictionary view |
| **INDEX** | Schema object that speeds up retrieval of rows |
| **SYNONYM** | To refer to a table by another name to simplify access |
| **COMPOSITE INDEX** | An index that you create on multiple columns in a table |
| **UNIQUE INDEX** | The Oracle Server automatically creates this index when you define a column in a table to have a PRIMARY KEY or a UNIQUE KEY constraint |
| **B-TREE INDEX** | Stores the indexed values and uses the index based on a SELECT statement to retrieve the data |
| **DROP INDEX** | Removes an index |
| **SYNONYM** | Gives alternative names to objects |

Try It / Solve It

1. What is an index and what is it used for?

An **index** is a schema object used to improve the performance of data retrieval operations by speeding up access to rows in a table. It’s similar to a table of contents in a book, and helps locate data efficiently without scanning the entire table.

2. What is a ROWID, and how is it used?

A **ROWID** is a unique identifier assigned to a row in a database table. It provides the physical address of a row and is used for fast data access by pointing directly to the storage location of the row in the database.

3. When will an index be created automatically?

Indexes are created automatically:

* When a **PRIMARY KEY** constraint is defined on a column.
* When a **UNIQUE KEY** constraint is defined on a column.

4. Create a nonunique index (foreign key) for the DJs on Demand column (cd\_number) in the  
D\_TRACK\_LISTINGS table. Use the Oracle Application Developer SQL Workshop Data Browser to confirm that the index was created.

CREATE INDEX idx\_cd\_number

ON d\_track\_listings(cd\_number);

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

5. Use the join statement to display the indexes and uniqueness that exist in the data dictionary for the DJs on Demand D\_SONGS table.

SELECT a.table\_name, a.index\_name, b.column\_name, a.uniqueness

FROM user\_indexes a

JOIN user\_ind\_columns b

ON a.index\_name = b.index\_name

WHERE a.table\_name = 'D\_SONGS';

6. Use a SELECT statement to display the index\_name, table\_name, and uniqueness from the data dictionary USER\_INDEXES for the DJs on Demand D\_EVENTS table.

SELECT index\_name, table\_name, uniqueness

FROM USER\_INDEXES

WHERE table\_name = 'D\_EVENTS';

7. Write a query to create a synonym called dj\_tracks for the DJs on Demand d\_track\_listings table.

CREATE SYNONYM dj\_tracks

FOR d\_track\_listings;

8. Create a function-based index for the last\_name column in DJs on Demand D\_PARTNERS table that makes it possible not to have to capitalize the table name for searches. Write a SELECT statement that would use this index.

CREATE INDEX idx\_last\_name\_lower

ON d\_partners(LOWER(last\_name));

SELECT \*

FROM d\_partners

WHERE LOWER(last\_name) = 'smith';

9. Create a synonym for the D\_TRACK\_LISTINGS table. Confirm that it has been created by  
querying the data dictionary.

CREATE SYNONYM track\_listings\_syn

FOR d\_track\_listings;

SELECT synonym\_name, table\_name

FROM USER\_SYNONYMS

WHERE synonym\_name = 'TRACK\_LISTINGS\_SYN';

A screenshot of a computer

Description automatically generated

10. Drop the synonym that you created in question 9.

DROP SYNONYM track\_listings\_syn;

**Database Programming with SQL  
17-1: Controlling User Access  
Practice Activities**

Try It / Solve It

1. What are system privileges concerned with?

System privileges are concerned with actions that can be performed at the database level, such as creating tables, views, or users. These privileges control access to system-level operations and are not tied to specific objects.

2. What are object privileges concerned with?

Object privileges are concerned with the rights to perform actions on specific database objects, such as tables, views, sequences, or procedures. Examples include SELECT, INSERT, UPDATE, and DELETE on a particular table.

3. What is another name for object security?

Another name for object security is **data security**.

4. What commands are necessary to allow Scott access to the database with a password of tiger?

CREATE USER scott IDENTIFIED BY tiger;

GRANT CONNECT TO scott;

5. What are the commands to allow Scott to SELECT from and UPDATE the d\_clients table?

GRANT SELECT, UPDATE ON d\_clients TO scott;

6. What is the command to allow everybody the ability to view the d\_songs table?

GRANT SELECT ON d\_songs TO PUBLIC;

7. Query the data dictionary to view the object privileges granted to you the user.

SELECT \*

FROM USER\_TAB\_PRIVS;

8. What privilege should a user be given to create tables?

The CREATE TABLE system privilege should be granted:

GRANT CREATE TABLE TO username;

9. If you create a table, how can you pass along privileges to other users just to view your table?

GRANT SELECT ON table\_name TO username;

10. What syntax would you use to grant another user access to your copy\_employees table?

GRANT SELECT, INSERT, UPDATE, DELETE ON copy\_employees TO username;

11. How can you find out what privileges you have been granted for columns in the tables belonging to others?

SELECT \*

FROM ALL\_COL\_PRIVS

WHERE GRANTEE = 'YOUR\_USERNAME';

**Database Programming with SQL  
17-2: Creating and Revoking Object Privileges  
Practice Activities**

Try It / Solve It

1. What is a role?

A **role** is a group of privileges that can be assigned to users. It simplifies the management of privileges by allowing administrators to grant and revoke multiple privileges through a single entity rather than assigning them individually.

2. What are the advantages of a role to a DBA?

* **Simplified privilege management**: Roles allow privileges to be grouped and managed collectively, reducing administrative overhead.
* **Consistency**: Ensures consistent privilege assignments across users with similar requirements.
* **Scalability**: Makes it easier to manage privileges as the number of users grows.
* **Flexibility**: Allows for the addition or removal of privileges from a role without directly modifying user privileges.

3. Give the ability to another user in your class to look at one of your tables. Give him the right to let other students have that ability.

GRANT SELECT ON my\_table TO student\_user WITH GRANT OPTION;

4. You are the DBA. You are creating many users who require the same system privileges. What  
should you use to make your job easier?

To make the job easier, you should use a **role**. Create a role with the necessary privileges and assign it to all the users:

CREATE ROLE common\_role;

GRANT privilege\_name TO common\_role;

GRANT common\_role TO user1, user2, user3;

5. What is the syntax to accomplish the following?

a. Create a role of manager that has the privileges to select, insert, and update and delete  
from the employees table

CREATE ROLE manager;

GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO manager;

b. Create a role of clerk that just has the privileges of select and insert on the employees  
table

CREATE ROLE clerk;

GRANT SELECT, INSERT ON employees TO clerk;

c. Grant the manager role to user scott

GRANT manager TO scott;

d. Revoke the ability to delete from the employees table from the manager role

REVOKE DELETE ON employees FROM manager;

6. What is the purpose of a database link?

A **database link** is a schema object in one database that enables access to objects in another database. It allows users to:

* Query and manipulate data in remote databases as if it were local.
* Simplify distributed database operations.
* Create seamless integration between multiple database systems.

**Database Programming with SQL  
17-3: Regular Expressions  
Practice Activities**

Try It / Solve It

1. Working with the employees table, and using regular expressions, write a query that returns  
employees whose first names start with a “S” (uppercase) followed by either a “t” (lowercase) or  
“h” (lowercase).

SELECT \*

FROM employees

WHERE REGEXP\_LIKE(first\_name, '^S[th]');

2. Investigate the LOCATIONS table.

a. Describe the table.

DESCRIBE LOCATIONS;

b. Perform a select that returns all rows and all columns of that table.

SELECT \*

FROM LOCATIONS;

c. Write a query using regular expressions that removes the spaces in the street\_address  
column in the LOCATIONS table.

SELECT REGEXP\_REPLACE(street\_address, ' ', '') AS street\_address\_no\_spaces

FROM LOCATIONS;