**Name:**

**CSIT 2520 - SQL Applications Using Oracle: Chapter 9 & 11 Homework (40 Pts)**

**Please download and save this document to your class drive. Then type your name at the top and record your responses below each question. Upload to the online dropbox when you’ve completed the assignment. To complete this homework successfully, you must read and refer to material presented in chapters 9 & 11 of the Price textbook. Related slide shows are also posted on our online [Course Content] page.**

1. Describe three of the fundamental properties (ACIDs) of valid transactions. (3 pts)

Atomic means the transaction and it’s SQL statements make up a single unit of work.

Consistent transaction make sure the database stays in a consistent state, meaning all integrity constraints are satisfied.

Isolated pertains transaction locking levels. “Separate transactions should not interfere with each other.”

Durable means changes remain permanent even if the system crashes.

1. List or describe three features provided by Oracle SQL to support valid transactions that correspond to the properties you have listed in question#1. (6 pts)

Concurrent transactions are transactions run at the same time that are run on the same table. The example in the book describes a situation where T1 inserts a new row into the table but T2 can’t “see” it until T1 commits it. Then, T2 can still only “see” it if they run another SELECT.

Another feature is transaction locking which has something similar to a stair step effect. At least, that’s the best way to visually explain it. T1 initiates its SQL statement and then moves up the staircase, applying each necessary lock one step at a time. T2 currently cannot access the row because T1 has obtained a lock on it (this actually depends on what level of locking there is.). When T1 has obtained all its locks, or it has reached the top of the staircase, then the SQL statement is performed and committed. Now T1 can release all the locks stepping back down the staircase and T2 can now access the row.

Keeping T2 from accessing the row is an example of transaction isolation levels. There are 5 main levels of locking:

1. Locking the whole database (“safest”) but very constrictive. It would likely slow down the system to an unbearable degree.
2. Table level
3. Disk page level locking (most common)
4. Row level
5. Cell level
6. Explain the difference between a simple view and a complex view including a comparison of how attempts to insert data into the base tables by using a view are affected by the type of view that is being used. (3 pts)

A simple view contains a subquery the shows rows from ONE table. While a complex view’s subquery shows rows from multiple tables, uses a GROUP BY or DISTINCT clause, and contains a function call. An INSERT statement can be performed on simple views ONLY. Complex views don’t support DML statements. If you include the CHECK OPTION constraint the database will return an error if the row the user is trying to INSERT isn’t retrievable by the view.

**Use the following table definition to answer questions 4 through 10.**

**CREATE TABLE demo\_emp (**

**empno NUMBER(4) NOT NULL,**

**ename VARCHAR2(10),**

**job VARCHAR2(9),**

**mgr NUMBER(4),**

**hiredate DATE DEFAULT SYSDATE,**

**sal NUMBER(7,2),**

**deptno NUMBER(2),**

**CONSTRAINT job\_type\_chk**

**CHECK(job IN('SALES', 'SUPERVISION','SUPPORT')),**

**CONSTRAINT emp\_self\_key FOREIGN KEY (mgr)**

**REFERENCES demo\_emp (empno),**

**CONSTRAINT emp\_foreign\_key FOREIGN KEY (deptno)**

**REFERENCES demo\_dept (deptno),**

**CONSTRAINT emp\_primary\_key PRIMARY KEY (empno));**

1. List each constraint that has been included in the definition of the demo\_emp table above and explain in reasonable detail the purpose of each of those constraints. (6 pts)

Empno contains a column level NOT NULL constraint which I feel is pretty self explanatory.

Hiredate contains a column level DEFAULT “constraint”. If the user doesn’t enter a hiredate then the default date is the SYSDATE (system date).

Job contains a table level CHECK constraint called job\_type\_chk the checks to make sure job is set to ‘SALES’, ‘SUPERVISION’, or ‘SUPPORT’.

Mgr contains table level FOREIGN KEY constraint called emp\_self\_key that references (ties) the mgr column in this table to the empno column in the demo\_emp table (this table).

Deptno contains a table level FOREIGN KEY constraint called emp\_foreign\_key that references the deptno column in this table to the deptno column the demo\_dept table.

Empno contains a table level PRIMARY KEY constraint that automatically applies a UNIQUE and NOT NULL constraint.

5. Regarding the NOT NULL constraint for the *empno* column:

1. Is it necessary to include this constraint for the demo\_emp table? Why or why not? (1 pts)

No because it is also given a table level PRIMARY KEY constraint which automatically applies a NOT NULL and UNIQUE constraint.

b. Is this constraint a column constraint or a table constraint. Please explain your answer.

(2 pts)

Column level because it is defined when the column is defined. Table level constraints are defined separately from the column definition.

1. Are the data type and size chosen for the *ename* column reasonable? Please justify your response. (Note: Names are entered into the table in the format last-name, first-name as for example *Doe, Jane*.) (2 pts)

The data type is fine but the size is not. What if you had customer’s name you had to enter that had a name like this; Janice Keihanaikukauakahihuliheekahaunaele (It looks Hawaiian. It’s really funny to actually try and pronounce haha!)? The last name by itself has 35 characters. But in all seriousness, according to the census bureau, 53% of surnames contain 5-7 characters. Using the max estimated value is best for size so this would mean most people, if we assumed a surname with a length of 7 characters, would have a first name with only 3 letters. I could also argue with the format in which you chose to store the names because of this guy 🡪Paul Ludwig Hans Anton von Beneckendorff und von Hindenburg. How the heck would you store that in the database? At this point I’m actually just trying to amuse you. ☺

7. Suppose I have successfully entered several employee records into my new demo\_emp table. Now I need to change the value of the salary column, *sal*, for one of the supervisors (Jane Doe) to $120,500.50. (4 pts)

1. Write the SQL statement(s) that I will use to perform this task.

UPDATE demo\_emp

SET sal = 120500.50

WHERE ename = ‘Doe, Jane’;

I used the format you mentioned above.

1. When I do this task, am I performing *Data Manipulation(DML)*, *Data Definition(DDL)*, and/or (TC)*Transaction Control*? Please justify your response.

Yes it is a DML because you are changing the data in the table but it is not a DDL statement because it is not altering any meta-data and it is not a transaction control statement because it is not a COMMIT, ROLLBACK, savepoint, etc. You actually still have to commit the UPDATE for it to become permanent.

1. Is the *sal* column big enough to accept the salary of $120,500.50 that I need to enter for Ms. Doe? If not, what is the result of my attempt to change it in step a?

Yes because the format of the column is NUMBER(7, 2) which means 7 places before the decimal and 2 places after. $120,500.50 is only 6 places before the decimal and 2 places after. No change is needed in step a.

8. I’ve been using my database for a while now and today I’ve been told that our company is reorganizing. Management plans to eliminate department #40 which currently has several employees assigned to it. (5 pts)

a. What SQL statement(s) will I use to remove a department number from my demo\_dept table?

ON DELETE CASCADE

1. How do the current structure and constraint definitions of my demo\_emp table affect my attempt to perform this task?

The system will currently not let you. It will give an error that will say there are other tables using the rows you are trying to delete. This is enforcement of referential integrity.

1. Does the result of my attempt to remove a department number from my demo\_dept table change if I have previously added an ON DELETE CASCADE clause to *my emp\_foreign\_key* constraint in the original table creation statement? Please explain.

Yes because that statement will make it where you can delete departments from the demo\_dept table and also remove them from the demo\_emp table. This is the DELETE statement cascading to all other tables that reference it. However, if you remove departments from the demo\_emp table and it will not remove rows from the demo\_dept table.

1. Does Oracle SQL provide me with statements that will allow me to retain employees from department #40 in my demo\_emp table while I remove that department number from my demo\_dept table and then move those employees to department #30 in a subsequent step? If so, please give me a hint on how I can do this (exact syntax is not necessary).

I would think not but the way you phrased it makes me think there is a way but I’m not sure how. The way I would do it would be to change the deptno of all employees in department #40 to #30 and THEN delete department #40.

--If the department hasn’t been added to the demo\_dept yet

INSERT INTO demo\_dept

VALUES (30, ‘deptname’, ‘other data in the table’)

WHERE NOT EXISTS;

UPDATE demo\_emp

SET deptno = 30

WHERE deptno = 40;

DELETE FROM demo\_dept

WHERE deptno = 40;

COMMIT;

That’s also known as a transaction

1. The task I am trying to perform is considered a transaction that must be completed as a unit or aborted in its entirety. List the steps that I must complete or abort to keep my database consistent.

I think I actually did that above in d.

9. My company has grown over the years and I now have 8000 employees working for me. On a daily basis, my assistants perform many database queries that involve retrieving information from the demo\_emp table. Most of these queries retrieve between 1 and 100 rows from my demo\_emp table.

a. Assuming that the demo\_emp table was created using the original statement provided to you, which of the following conditions is it most reasonable to assume I might be experiencing? (1 pt)

\_X\_\_ queries that search for records based on *empno* are faster than queries that

search for records based on *ename*

\_\_\_\_ queries that search for records based on *empno* are about the same speed as

searches for records based on *ename*

\_\_\_\_ queries that search for records based on *ename* are faster than queries that

search for records based on *empno*

**Why?**

Numbers are generally faster for the computer to process and the empno is unique while there can be multiples of ename.

b. What feature(s) of Oracle can I use to try to improve the performance of queries such as those described above? (1 pt)

I’m honestly not sure.

1. Given the circumstances I’ve described, do you feel it would be worthwhile to attempt to add the performance enhancing code identified in step b. to my environment? (Please justify your response. 2 pts)

Yes. If the system has multiple transactions going on at once, each retrieving up to 100 rows at a time, performance is definitely something to take into consideration. The developer then also has to think about locks which take up even more time.

10. Suppose that before I have entered any data into my demo\_emp table, I have decided that I would like the *empno* automatically generated for me.

1. What feature of Oracle allows me to perform this task? (1 pt)

The AUTO NUMBER constraint

1. If I want to begin my employee numbers at 100 and increment each subsequent employee number by 10 without reusing any employee numbers, will Oracle allow me to do this? (1 pt)

Yes

1. List the steps or show me the approximate syntax that I will need to use to set up this automatic employee number generation system. Please also tell or show me how I will add records to my tables that contain these system generated employee numbers. (2 pts)

CREATE SEQUENCE empno\_sequence

MINVALUE 100

MAXVALUE 999999999

START WITH 100

INCREMENT BY 10

CACHE 20;

INSERT INTO demo\_emp

VALUES (empno\_sequence.NEXTVAL, ‘Doe, John’, etc.);