


# CSCI 5707 - Homework 3

 Names:	Kelsey Neis (neis), Bela Demtchouk (derga002), Ashwin Sridhar (sridh052)
---	--

**Question 16.2 Consider the following actions taken by transaction T1 on database objects X and Y:**

**T1: R(X), W(X), R(Y), W(Y)**

**1. Give an example of another transaction T2 that, if run concurrently to transaction T1 without some form of concurrency control, could interfere with T1.**

T2: R(X), W(X)

If in T1 W(X) = writing  $X*10$  and in T2: W(X) = writing  $X+2$  then hypothetically

T1: R(X)                      W(X) R(Y) W(Y)

T2:        R(X) W(X)

would result in T1 overwriting T2's result not in the intended manner

**2. Explain how the use of Strict 2PL would prevent interference between the two transactions.**

T1 would get an exclusive lock on X and Y before writing to them, blocking T2 from getting a lock on X until T1 is complete. This would force T1 & T2 to run serially in this case.

**3. Strict 2PL is used in many database systems. Give two reasons for its popularity.**

It ensures that transactions are recoverable by only allowing transactions to read committed changes. It also guarantees that regardless of how transactions are interleaved, the state of the database will be the same as though the transactions were run serially.

In short, 2PL allows:

1) Save Interleaves

2) Repeatable Reads

3) Prevents reading uncommitted data.

**Question 16.4 We call a transaction that only reads database object a read-only transaction, otherwise the transaction is called a read-write transaction. Give brief answers to the following questions**

**1. What is lock thrashing and when does it occur?**

Thrashing is a point at which adding more transactions actually decreases transaction throughput as the new transactions are blocked and compete with existing ones.

Lock thrashing is when the amount of concurrent transactions is so large that the amount of them that are blocked waiting for locks reaches 30%.

**2. What happens to the database system throughput if the number of read-write transactions is increased?**

Throughput increases until the number of RW transactions reaches the thrashing point. This point is reached because when the number of read-write transactions goes up because those additional transactions begin to compete for locks with existing transactions. The closer it gets, the slower the rate of increase. Then when the thrashing point is reached, throughput starts to decline.

**3. What happens to the database system throughput if the number of read-only transactions is increased?**

It increases indefinitely, since read-only transactions only use shared locks, thus no blocking occurs.

**4. Describe three ways of tuning your system to increase transaction throughput.**

- By locking only the objects you need to lock for a given transaction
- By limiting the amount of time that a transaction can hold locks
- By reducing hotspots, database objects that are frequently read and updated
- Buying a faster system

## Question 21.2

You are the DBA for the VeryFine Toy Company and create a relation called **Employees** with fields **ename, e, dept, and Sala1~1j**. For authorization reasons, you also define **views EmployeeNames (with ena:rne as the only attribute) and DeptInfo with fields dept and avgsalary**. The latter lists the average salary for each department.

1. Show the view definition statements for **EmployeeNames** and **DeptInfo**.

```
CREATE VIEW EmployeeNames
AS SELECT ename
FROM Employees;

CREATE VIEW DeptInfo
AS SELECT dept, AVG(salary)
FROM Employees
GROUP BY dept;
```

2. What privileges should be granted to a user who needs to know only average department salaries for the Toy and CS departments?

```
CREATE VIEW csToyDeptInfo
AS SELECT dept, AVG(salary)
FROM Employees e
WHERE e.dept = 'Toys'
OR e.dept = 'CS'
GROUP BY dept;

GRANT SELECT ON EmployeeNames TO toyCSManager;
```

3. You want to authorize your secretary to fire people (you will probably tell him whom to fire, but you want to be able to delegate this task), to check on who is an employee, and to check on average department salaries. What privileges should you grant?

```
GRANT SELECT ON EmployeeNames TO Secretary;
GRANT SELECT ON DeptInfo TO Secretary;
GRANT DELETE ON Employees TO Secretary;
```

**4. Continuing with the preceding scenario, you do not want your secretary to be able to look at the salaries of individuals. Does your answer to the previous question ensure this? Be specific: Can your secretary possibly find out salaries of some individuals (depending on the actual set of tuples)", or can your secretary always find out the salary of any individual he wants to?**

The only scenario the secretary could find salaries of individuals is the unlikely one where an employee is the only member of the department.

**5. You want to give your secretary the authority to allow other people to read the EmployeeNames view. Show the appropriate command.**

```
GRANT SELECT ON EmployeeNames TO secretary WITH GRANT OPTION;
```

**6. Your secretary defines two new views using the EmployeeNames view. The first is called AtoRNames and simply selects names that begin with a letter in the range A to R. The second is called HowManyNames and counts the number of names. You are so pleased with this achievement that you decide to give your secretary the right to insert tuples into the EmployeeNames view. Show the appropriate command and describe 'what privileges your secretary has after this command is executed.**

```
GRANT INSERT ON EmployeeNames TO secretary;
```

Since granting insert access to EmployeeNames gives access to the underlying tables, the secretary can now add tuples to the Employees table, but still can't read columns in the Employees table other than ename.

**7. Your secretary allows Todd to read the EmployeeNames relation and later quits. You then revoke the secretary's privileges. What happens to Todd's privileges?**

Unless another user grants access to Todd, he would lose access to EmployeeNames, because of the cascading rule.

**8. Give an example of a view update on the preceding schema that cannot be implemented through updates to Employees.**

One example of a view update that cannot be implemented through updates to Employees is changing the average salary for a department since one doesn't know which salaries to change.

**9. You decide to go on an extended vacation, and to make sure that emergencies can be handled, you want to authorize your boss Joe to read and modify the Employees relation and the EmployeeNames relation (and Joe must be able to delegate authority, of course, since he is too far up the management hierarchy to actually do any work). Show the appropriate SQL statements. Can Joe read the DeptInfo view?**

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Employees TO Joe WITH GRANT OPTION
GRANT SELECT, INSERT, UPDATE, DELETE ON EmployeeNames TO Joe WITH GRANT OPTION
```

Joe cannot read the DeptInfo view, but can read an identical view. In other words, Joe cannot read the DeptInfo view but can read the view that is aggregated to create it, the more granular version of DeptInfo — EmployeeNames

**10. After returning from your (wonderful) vacation, you see a note from Joe, indicating that he authorized his secretary Mike to read the Employees relation. You want to revoke Mike's SELECT privilege on Employees, but you do not want to revoke the rights you gave to Joe, even temporarily. Can you do this in SQL?**

You cannot do this in SQL because even though you granted privileges to Joe, Joe granted privileges to Mike. Thus, you cannot revoke Joe's privileges without also revoking Mike's.

**11. Later you realize that Joe has been quite busy. He has defined a view called All-Names using the view EmployeeNames, defined another relation called StaffNames that he has access to (but you cannot access), and given his secretary Mike the right to read from the AllNames view. Mike has passed this right on to his friend Susan. You decide that, even at the cost of annoying Joe by revoking some of his privileges, you simply have to take away Mike and Susan's rights to see your data. What REVOKE statement would you execute? What rights does Joe have on Employees after this statement is executed? What views are dropped as a consequence?**

Since you don't own AllNames, you can only prevent Mike and Susan from accessing it by revoking Joe's right to read EmployeeNames:

```
REVOKE SELECT ON EmployeeNames FROM Joe
```





The view AllNames is dropped as a consequence of Joe losing his permissions to the parent View. Joe can still modify EmployeeNames without reading it.

## Exercise 25.2

**Consider the instance of the Sales relation shown in Figure 25.2.**

**1. Show the result of pivoting the relation on pid and timeid**

**Pivot on pid and timeid**

<u>Aa</u> pid/timeid	 1	 2	 3	 total
<u>11</u>	60	30	25	115
<u>12</u>	56	65	70	191
<u>13</u>	28	50	15	93
<u>total</u>	144	145	110	
<u>Untitled</u>				

**2. Write a collection of SQL queries to obtain the same result as in the previous part.**

```
SELECT
  s.pid,
  s.timeid,
  SUM(s.sales)
FROM
  sales
GROUP BY
  CUBE(s.pid,s.timeid)
```

**3. Show the result of pivoting the relation on pid and locid.**

**Pivot on pid and locid**

<u>Aa</u> pid/locid	 1	 2	 total
---------------------	---	---	---

Aa pid/locid	≡ 1	≡ 2	# total
<u>11</u>	48	67	115
<u>12</u>	100	91	191
<u>13</u>	28	65	93
<u>total</u>	176	223	

## Exercise 25.10

Consider the Locations, Products, and Sales relations in Figure 25.2.

1. To decide whether to materialize a view, what factors do we need to consider?

You need to determine whether the queries or views in question are expensive and frequently used enough to justify materializing them.

2. Assume that we have defined the following materialized view:

```
SELECT L.state, S.sales
FROM Locations L, Sales S
WHERE S.locid=L.locid
```

- a. Describe what auxiliary information the algorithm for incremental view maintenance from Section 25.10.1 maintains and how this data helps in maintaining the view incrementally

The count for how many times each row from the view can be derived from state/sales combinations from the Locations and Sales tables. This would help determine if a new row should be added when either the Location or Sales table gets a new row, or if a row should be removed, if the count is 1, and a deletion from Location or Sales removes the last remaining such combination. Modifications to Locations or Sales then only need to compute  $L_i \bowtie S$ ,  $L_d \bowtie S$ ,  $L \bowtie S_i$ , or  $L \bowtie S_d$

- b. Discuss the pros and cons of materializing this view.

- Pros: Joins are expensive operations, so materializing a view for it would lead to faster queries related to state and sales

- Cons: It does not seem that this view would be useful for queries involving the other tables, and would only serve the narrow purpose of showing the sales by state. Furthermore, there would likely be many rows with the same location, but with different sales, and one couldn't distinguish between them after the projection of state and sales.

**3. Consider the materialized view in the previous question. Assume that the relations**

**Locations and Sales are stored at one site, but the view is materialized on a second site.**

**Why would we ever want to maintain the view at a second site? Give a concrete example**

**where the view could become inconsistent.**

If the second site is far away, querying the tables directly may be too slow, so a materialized view would help. It needs to be maintained so that the data remains accurate even if changes happen to the Locations and Sales tables that would affect the materialized view. For example, if a Sales tuple is added or deleted. If added, and the combination of sales and location are not already in the view as a row, a row needs to be added. If deleted, where the combination of location and sales has a count of 1 in the view, the row in the view needs to be deleted as well, otherwise the view does not represent an accurate picture of the data.

**4. Assume that we have defined the following materialized view:**

```
SELECT T.year, L.state, SUM (S.sales)
FROM Sales S, Time T, Locations L
WHERE S.timeid=T.timeid AND S.locid=L.locid
GROUP BY T.year, L.state
```

**a. Describe what auxiliary information the algorithm for incremental view maintenance from Section 25.10.1 maintains, and how this data helps in maintaining the view incrementally.**

- The count for how many times the combination of year and state can be derived from the data. This would allow for adding or removing individual view rows when new rows are added or removed from sales. If a new row is added, and it is not represented in the view, a new row needs to be added to the view. It would work similarly for deletions.



- The sum of sales. If the row already exists, then the sales can be added to the sum of sales for the existing row in the view. Similarly with deletions of rows, the sum needs to be subtracted from the sum of sales in the view.

**b. Discuss the pros and cons of materializing this view**

- Pros: As this view has two joins on the sales table, it would be an expensive operation to compute this view every time. This view seems more useful than the one from #3, since the time is included and it is aggregate rather than just a flat list of sales/location pairs. So, it is a better candidate for materializing. It could be used in queries involving sales, location and time, rather than just location and sales.
- Cons: Maintenance would be harder to maintain, since there is more auxiliary information to maintain, and more tables are involved, thus updates involving multiple columns need to be tracked in order to keep the view maintained.