Assignment 5
Due December 6, 2017 at 11 am

Part 1: Indexes /40

The last thing we'll be doing with our employee database is adding a few indexes. After adding each index, run the associated query/queries and record the performance (planning time and execution time). Also look at the explain plan of the queries. *You'll probably need to rewrite the queries slightly to fit your database (if there are different columns or tables).*

Index 1: Add an index to the employee_histories table first_name and last_name fields.
Index 2: Add an index to the employee_jobs table employee_id and job_id fields.
Index 3: Add an index to the employees table birthdate field.

For each index, answer the following questions:

a. Fill out the tables below describing how adding the index affected the planning and execution timings.
b. Did adding the index change the explain plans? What changed?
c. Was this what you expected to happen for the timing and the execution plans? What is a possible reason for this change (or lack of change)?

Index 1

Execution Time	Without index	With index
Query 1		
Query 2		

Index 2

Execution Time	Without index	With index
Query 3		

Index 3

Execution Time	Without index	With index
Query 4		

Part 2: Normalization /40

Let’s pretend that the company whose employees we’ve been managing so far is an engineering firm. The company manages multiple projects at a time, and assigns its employees to tasks on the different projects. Only one employee can be assigned to a project task. Below is some un-normalized data used to manage projects in a company. After analyzing this sample data, structure it in 1st normal, 2nd normal, and 3rd normal form one step at a time, showing the results of each step. So you should have 3 diagram – one for your data in 1st normal, one for 2nd normal, and one for 3rd normal.

Team Member Id	Team Member First Name	Team Member Last Name	Project Code	Project Name	Project Status	Project Manager	Task Number	Task Status
1	John	Smith	DDL	Darren & Darren Ltd	Active	Garth Butler	10 132 133 134	Resolved In Progress Not Started In Progress
2	Dave	Richter	DDL KMI	Darren & Darren Ltd Kristen Motors Inc.	Active Active	Garth Butler Jim David	100 110 10 13	In Progress Not Started Not Started Resolved
3	Janie	Klotter	KMI	Kristen Motors Inc.	Active	Jim David	1 2 15	In Progress Resolved Resolved

Part 3: Concurrency /20

1. Scenario – Transaction A and B are being run concurrently in separate sessions.

Below is the initial state of the Accounts table before any transaction is run

Account Number	Account Nickname	Account Balance
1	Chequing	450
2	Chequing	200

Transaction A	Transaction B

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN
 	
SELECT
  a.account_number,
  a.account_nickname,
  a.account_balance
FROM accounts;
 	
UPDATE accounts
SET account_balance = 0
WHERE account_number = 2;
 	
END;
COMMIT;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
	BEGIN
	
	SELECT
	  a.account_number,
	  a.account_nickname,
	  a.account_balance
	FROM accounts;
	
	UPDATE accounts
	SET account_balance = account_balance – 100
	WHERE account_number = 1;
	
	UPDATE accounts
	SET account_balance = account_balance + 100
	WHERE account_number = 2;
	
	END;
	COMMIT;

a. What would the Accounts table look like after these transactions are finished?

Account Number	Account Nickname	Account Balance

b. What type(s) of data inconsistency is caused in this case (lost update, dirty read, non-repeatable read, or phantom read)?

2. Transaction C and D are being run concurrently in separate sessions

Below is the initial state of the Accounts table before any transaction is run:

Account Number	Account Nickname	Account Balance
1	Chequing	450
2	Chequing	200

Transaction C	Transaction D

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN
	SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
	BEGIN

SELECT
  a.account_number,
  a.account_nickname,
  a.account_balance
FROM accounts;	

	INSERT INTO accounts (account_number, account_nickname, account_balance)
	VALUES(3, 'Savings', 50);

	UPDATE accounts
	SET account_balance = 300
	WHERE account_number = 1;

SELECT
  a.account_number,
  a.account_nickname,
  a.account_balance
FROM accounts;
	END;
	COMMIT;

END;
COMMIT;	

a. What type(s) of data inconsistency is caused in this case (lost update, dirty read, non-repeatable read, or phantom read)?

3. Transaction E and F are being run concurrently in separate sessions

Below is the initial state of the Accounts table before any transaction is run:

Account Number	Account Nickname	Account Balance
1	Chequing	450
2	Chequing	200

Transaction E	Transaction F

SET TRANSACTION ISOLATION LEVEL UNCOMMITTED READ;
BEGIN
	SET TRANSACTION ISOLATION LEVEL UNCOMMITTED READ;
	
	BEGIN

SELECT
  a.account_number,
  a.account_nickname,
  a.account_balance
FROM accounts;	

UPDATE accounts
SET account_balance = 300
WHERE account_number = 1;	

	SELECT
	  a.account_number,
	  a.account_nickname,
	  a.account_balance
	FROM accounts;

SELECT
  a.account_number,
  a.account_nickname,
  a.account_balance
FROM accounts;
 	
END;
ROLLBACK ;	

	INSERT INTO accounts (account_number, account_nickname, account_balance)
	VALUES(3, ‘Savings’, 50);
	
	END;
	COMMIT;

a. What type(s) of data inconsistency is caused in this case (lost update, dirty read, non-repeatable read, or phantom read)?