CS245: Databases

SQL

Vijaya saradhi

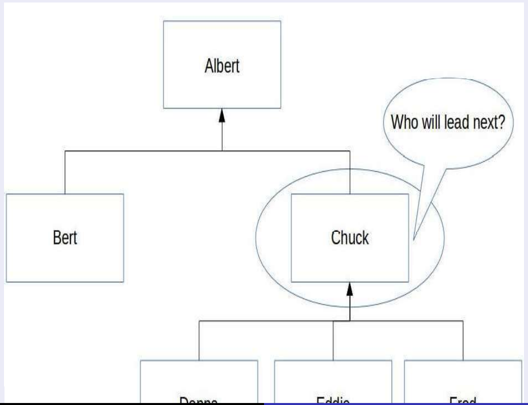
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Supervisor-supervisee

Manages Relation

Employee	Boss	Salary
Albert	$\perp$	1000.00
Bert	Albert	900.00
Chuck	Albert	900.00
Donna	Chuck	800.00
Eddie	Chuck	700.00
Fred	Chuck	600.00

Manages Relation



Supervisor-supervisee

Anomalies

INSERT	Can include cycles in the graph
UPDATE	
DELETE	
Structural	

Insertion Anomaly Example

Employee	Boss	Salary
Albert	$\perp$	1000.00
Albert	Fred	100.00
Bert	Albert	900.00
Chuck	Albert	900.00
Donna	Chuck	800.00
Eddie	Chuck	700.00
Fred	Chuck	600.00

Supervisor-supervisee

Anomalies

INSERT	Can include cycles in the graph
UPDATE	`UPDATE manager set Employee='Charles' where Employee = 'Chuck';`
DELETE	
Structural	

UPDATE Anomaly Example

Employee	Boss	Salary
Albert	$\perp$	1000.00
Bert	Albert	900.00
Charles	Albert	900.00
Donna	Chuck	800.00
Eddie	Chuck	700.00
Fred	Chuck	600.00

In atomic fashion

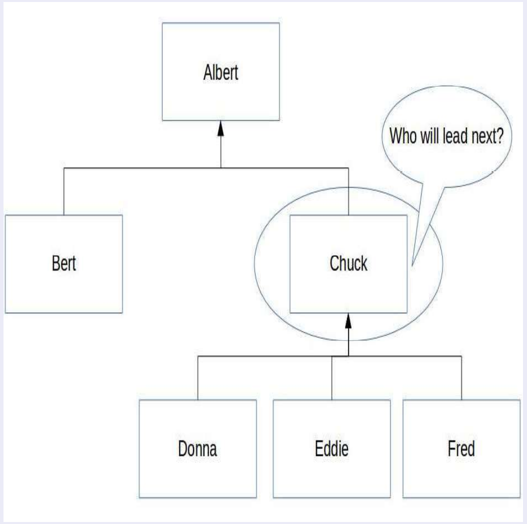
```
UPDATE manager set Employee='Charles' where Employee =
'Chuck';
UPDATE manager set Boss='Charles' where Boss = 'Chuck';
```

Supervisor-supervisee

Anomalies

INSERT	Can include cycles in the graph
UPDATE	`UPDATE manager set Employee='Charles' where Employee = 'Chuck';`
DELETE	Chuck left the organization. What should be the right way?
Structural	

DELETE Anomaly Example



Supervisor-supervisee

Structural Anomalies

- `INSERT INTO Manager (Employee, Boss) VALUES (‘a’, ‘a’);`
- Create simple cycles
- `INSERT INTO Manager (Employee, Boss) VALUES (‘b’, ‘c’);`
- `INSERT INTO Manager (Employee, Boss) VALUES (‘c’, ‘b’);`

Supervisor-supervisee: Solution - Part I

Modify relation

- Employee details and organization hierarchy must be separated
- Create table for `Employee(eid, ename, address)`
- Create table for hierarchy `Manages(role, eid, boss_eid)`
- role should be primary key
- (eid, boss_eid) should be unique
- eid should be foreign key referring Employee
- eid default value should be 0 to indicate vacant position
- eid should not be NULL

Supervisor-supervisee: Solution - Part II

Constraints

- Self boss is not allowed. `CHECK(eid <> boss_eid);`
- boss_eid and eid should not be 0; `CHECK( boss_eid != 0 AND eid != 0 )`
- Number of nodes in tree: `SELECT COUNT(*) FROM Manages`
- Number of edges in tree: `SELECT COUNT(boss_eid) FROM Manages`
- Number of edges = number of nodes - 1; `CHECK( (SELECT COUNT(*) FROM Manages) - 1 = (SELECT COUNT(boss_eid) FROM Manages) )`
- Only one root:
  `CHECK(SELECT COUNT(*) FROM Manages where ISNULL(boss_eid) = 1)`

Supervisor-supervisee: Solution - Part III

Constraints - Check for Cycles

```
CREATE FUNCTION TreeTest() RETURNS CHAR(6)
 BEGIN ATOMIC
-- put a copy in a temporary table
        INSERT INTO Tree SELECT eid, boss_id FROM Manages

-- prune the leaves
           WHILE ((SELECT COUNT(*) FROM Tree) - 1) = (SELECT COUNT(boss_id) FROM Tree)
           DO
                       DELETE FROM Tree
                    -- Check employee is not the boss
                    WHERE Tree.eid
                    NOT IN (
                                -- Select all the bosses
                                SELECT T2.boss_id
                                FROM Tree AS T2
                                WHERE NOT ISNULL(T2.boss_id)
                     );

     IF NOT EXISTS (SELECT * FROM Tree)
     THEN
        RETURN ('Tree');
20␣␣␣␣ELSE
23␣␣␣␣␣␣␣RETURN␣('Cycles');
     END IF;
  END WHILE;
END;
```

Supervisor-supervisee: Steps

Detailed Steps

```
Iteration #1
----------------------------------------------------------------------
Albert Not in {Albert, Albert, Chuck, Chuck, Chuck}? No;
Bert   Not in {Albert, Albert, Chuck, Chuck, Chuck}? Yes; Delete
Chuck Not in {Albert, Albert, Chuck, Chuck, Chuck}? No;
Donna Not in {Albert, Albert, Chuck, Chuck, Chuck}? Yes; Delete
Eddie Not in {Albert, Albert, Chuck, Chuck, Chuck}? Yes; Delete
Fred   Not in {Albert, Albert, Chuck, Chuck, Chuck}? Yes; Delete
----------------------------------------------------------------------
```

Supervisor-supervisee: Steps

Detailed Steps

```
Iteration #2
------------------------------------------------------------------------
Albert NULL
Chuck  Albert

Albert  Not in {Albert} No;
Chuck   Not in {Albert} Yes; Delete
------------------------------------------------------------------------
```

Supervisor-supervisee: Steps

Detailed Steps

```
Iteration #3
-----------------------------------------------------------------------
Albert NULL
Albert  Not in {} Yes; Delete
-----------------------------------------------------------------------
```

Exceptions

SQL exception - 01

- An SQL system indicates error conditions by setting non-zero sequence of digits in SQLSTATE
- Example 02000 `no tuple found`
- Example 21000 `single row select has returned more than one row`
- We can declare user defined exceptions called exception handler
- Invoked whenever one of a list of these error codes appear in SQLSTATE during execution of a statement
- Each exception handler is associated with a block of code
- delineated by `BEGIN` … `END`

Exceptions

SQL exception - 02

- The form of a handler declaration is
- DECLARE [where to go] HANDLER FOR [condition list] [statement]
- where to go:

CONTINUE
: means that after executing the statement in the handler declaration, we execute the statement after the one raised the exception

EXIT
: after executing the handler's statement, control leaves `BEGIN` ... `END` block in which the handler is declared

UNDO
: Same as EXIT which a difference that any changes to the database or local variables that were made by the block executed so far are undone