Section 6

Triggers & Security
Triggers

Trigger = a procedure invoked by the DBMS in response to an update to the database

Trigger = Event + Condition + Action
Triggers in SQL

• Event = INSERT, DELETE, UPDATE

• Condition = any WHERE condition
  o Refers to the old and the new values

• Action = more inserts, deletes, updates
  o May result in cascading effects!
Example: Row Level Trigger

CREATE TRIGGER InsertPromotions AFTER UPDATE OF price ON Product

REFERENCING
OLD AS x
NEW AS y

FOR EACH ROW
WHEN (x.price > y.price)
INSERT INTO Promotions(name, discount)
VALUES x.name,
(x.price-y.price)*100/x.price
EVENTS

INSERT, DELETE, UPDATE

• Trigger can be:
  o AFTER event
  o INSTEAD of event
Scope

• FOR EACH ROW = trigger executed for every row affected by update
  o OLD ROW
  o NEW ROW

• FOR EACH STATEMENT = trigger executed once for the entire statement
  o OLD TABLE
  o NEW TABLE
Statement Level Trigger

CREATE TRIGGER avg-price INSTEAD OF UPDATE OF price ON Product

REFERENCING
OLD_TABLE AS OldStuff
NEW_TABLE AS NewStuff

FOR EACH STATEMENT
WHEN (1000 < (SELECT AVG (price)
FROM ((Product EXCEPT OldStuff) UNION NewStuff))
DELETE FROM Product
WHERE (name, price, company) IN OldStuff;
INSERT INTO Product
(SELECT * FROM NewStuff)
Triggers v.s. Integrity
Constraints

Active database = a database with triggers

• Triggers can be used to enforce ICs
• Triggers are more general: alerts, log events
• But hard to understand: recursive triggers
• Syntax is vendor specific, and may vary significantly
  o Postgres has *rules* in addition to *triggers*
Postgres & Triggers

• Procedural Language
  – PL/pgSQL

• Parts
  1. Write a PL/pgSQL function
  2. Create trigger to use the function
Postgres Trigger Example

Employee Salary Table

CREATE TABLE emp (  
    empname text,  
    salary integer,  
    last_date timestamp,  
    last_user text );
CREATE FUNCTION emp_stamp() RETURNS trigger AS $emp_stamp$
BEGIN
    IF NEW.salary < 0 THEN
        RAISE EXCEPTION 'Employee % cannot have a negative salary', NEW.empname;
    END IF;

    NEW.last_date := current_timestamp;
    NEW.last_user := current_user;
    RETURN NEW;
END;
$emp_stamp$ LANGUAGE plpgsql;
Postgres Trigger Example (Step 2)

CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON emp FOR EACH ROW EXECUTE PROCEDURE emp_stamp();
Security

Goal:
Only allow users to see the information they need to see, no more.

- Create views that reveal only what the users are allowed to know
- Grant users access only to relevant views
Views and Security

CREATE VIEW PublicCustomers
    SELECT Name, Address
    FROM Customers

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Huston</td>
<td>450.99</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>-240</td>
</tr>
<tr>
<td>Joan</td>
<td>Seattle</td>
<td>333.25</td>
</tr>
<tr>
<td>Ann</td>
<td>Portland</td>
<td>-520</td>
</tr>
</tbody>
</table>

Fred is not allowed to see this.

Fred is allowed to see this.

CREATE VIEW PublicCustomers
    SELECT Name, Address
    FROM Customers
Views and Security

Customers

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Huston</td>
<td>450.99</td>
</tr>
<tr>
<td>Sue</td>
<td>Seattle</td>
<td>-240</td>
</tr>
<tr>
<td>Joan</td>
<td>Seattle</td>
<td>333.25</td>
</tr>
<tr>
<td>Ann</td>
<td>Portland</td>
<td>-520</td>
</tr>
</tbody>
</table>

CREATE VIEW BadCreditCustomers
SELEC
FROM Customers
WHERE Balance < 0

John is not allowed to see balances >0
Access Control

• Role
  – A group with specific privileges (eg. DataEntry, CustomerSupport)

• User
  – The individual (eg. John, Fred, Program)
Access Control

CREATE ROLE BadCreditEnforcers;

GRANT SELECT,UPDATE
    ON BadCreditCustomers
    TO BadCreditEnforcers;

CREATE USER John WITH
    PASSWORD ‘john-password’
    IN ROLE BadCreditEnforcers;