CS 1555
www.cs.pitt.edu/~nlf4/cs1555/
Checks, assertions, and triggers
Semantic integrity constraints

- A constraint is expressed as a predicate, a condition similar to the one at the WHERE clause of a query
- We have three DDL constructs to express semantic integrity constraints:
  - Checks
  - Assertions
  - Triggers
CREATE TABLE DEPARTMENT ( 
    Dname VARCHAR(15) UNIQUE, 
    Dnumber INT PRIMARY KEY, 
    Created_date DATE, 
    Budget DECIMAL(15,2), 
    Mgr_ssn CHAR(9) NOT NULL, 
    Mgr_start_date DATE, 
    CONSTRAINT dept_fk
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn), 
    CONSTRAINT dept_budget_IC1 
        CHECK ((Budget >= 0) AND (Budget IS NOT NULL)) DEFERRABLE, 
    CONSTRAINT mgr_IC 
        CHECK (Created_date <= Mgr_start_date) DEFERRABLE
);
Assertions

- CREATE OR REPLACE ASSERTION <aname>
  CHECK <predicate>;
- DROP ASSERTION <aname>;
- CREATE OR REPLACE ASSERTION budget_constraint
  CHECK (NOT EXISTS
    (SELECT * FROM DEPARTMENT D WHERE Budget <
    (SELECT SUM(Salary) FROM EMPLOYEE E
    WHERE E.Dno = D.Dnumber)));
- Assertion predicates often built around EXISTS and NOT EXISTS
Triggers

- Triggers consist of two parts: a condition and an action
  - Action is executed when condition becomes true

- CREATE or REPLACE TRIGGER <trigname>
  
  <time> <event> ON <tname>
  [REFERENCING {NEW | OLD} AS <refname>]
  [FOR EACH ROW]
  [WHEN (<predicate>)]
  <action>
Triggers

- **<time>**
  - BEFORE
  - AFTER
  - INSTEAD OF

- **<event>**
  - INSERT
  - DELETE
  - UPDATE [OF <alist>]
Triggers

- FOR EACH ROW classifies the trigger as a row trigger
  - In contrast to a statement trigger
- NEW and OLD allow reference to be made to new and old values of tuples affected by the trigger (and satisfy the WHEN clause) in a row trigger
  - Can also be aliased using REFERENCING ... AS ...
- WHEN specifies a condition that must be met for the trigger to fire
- <action>
  - Either a sequence of SQL statements
  - Or a stored procedure
Trigger maintenance

- ALTER TABLE <tname> ENABLE ALL TRIGGERS;
- ALTER TABLE <tname> DISABLE ALL TRIGGERS;
- ALTER TRIGGER <trigname> {ENABLE | DISABLE};
- DROP TRIGGER <trigname>;
 Assertions and Triggers

- CREATE OR REPLACE ASSERTION budget_constraint
  CHECK (NOT EXISTS
  (SELECT * FROM DEPARTMENT D WHERE Budget <
   (SELECT SUM(Salary) FROM EMPLOYEE E
   WHERE E.Dno = D.Dnumber)));

- CREATE OR REPLACE TRIGGER budget_constraint_trig
  AFTER INSERT OR UPDATE OF Budget
  ON DEPARTMENT
  WHEN (EXISTS
   (SELECT * FROM DEPARTMENT D WHERE Budget <
    (SELECT SUM(Salary) FROM EMPLOYEE E
    WHERE E.Dno = D.Dnumber)));
  ROLLBACK;
CREATE OR REPLACE TRIGGER budget_constraint_trig
AFTER INSERT OR UPDATE OF Salary
ON EMPLOYEE
WHEN (EXISTS
(SELECT *
FROM DEPARTMENT D
WHERE Budget <
(SELECT SUM(Salary)
FROM EMPLOYEE E
WHERE E.Dno = D.Dnumber)));
ROLLBACK;
Another trigger utility example

- Combining triggers with *stored procedures* can prove useful

- **CREATE TRIGGER** salary_trig
  **AFTER UPDATE OF** Salary
  **ON** EMPLOYEE
  **WHEN** (EXISTS (SELECT *
    FROM EMPLOYEE E, EMPLOYEE F, DEPARTMENT D
    WHERE E.Salary > F.Salary
    AND E.Dno = D.Dnumber
    AND D.Mgr_ssn = F.Ssn
    AND E.Ssn <> F.Ssn))
  **CALL** Email_director (E.Ssn, Mgr_ssn);
Statement-level triggers

- CREATE OR REPLACE TRIGGER Audit_Updater
  AFTER INSERT OR DELETE OR UPDATE ON STUDENTS
  BEGIN
  INSERT INTO AUDIT_Table
  VALUES ('STUDENT', sysdate);
  END;
/

CREATE OR REPLACE TRIGGER trigger_deans_list
AFTER INSERT ON STUDENTS
REFERENCING NEW AS newRow
FOR EACH ROW
WHEN (newRow.GPA > 3.5)
BEGIN
    INSERT INTO DL
    VALUES ( :newRow.ID, :newRow.GPA );
END;
/
CREATE OR REPLACE TRIGGER bad_auto_sid
AFTER INSERT ON STUDENTS
FOR EACH ROW
BEGIN
    SELECT MAX(ID) + 1 INTO :NEW.ID
    FROM STUDENTS;
END;
/

ERROR at line 1:
ORA-04084: cannot change NEW values for this trigger type
An improved trigger

- CREATE OR REPLACE TRIGGER bad_auto_sid
  
  BEFORE INSERT ON STUDENTS
  
  FOR EACH ROW
  
  BEGIN
  
  SELECT MAX(ID) + 1 INTO :NEW.ID
  
  FROM STUDENTS;
  
  END;
  
  /
  
  /
Achieving AUTO_INCREMENT in Oracle

- We can use a trigger!
- We'll need to use two special Oracle constructs:
  - A sequence, example:
    ```sql
    CREATE SEQUENCE seq_ex
    NOCYCLE MAXVALUE 99999 START WITH 1;
    ```
  - The dual table:
    ```sql
    SQL> DESCRIBE DUAL;
    Name Null? Type
    -------- ------- ---------------
    DUMMY     NO      VARCHAR2(1)
    SQL> SELECT * FROM DUAL;
    D
    -
    X
    ```
Achieving AUTO_INCREMENT in Oracle

- **CREATE OR REPLACE TRIGGER** `auto_increment_ex`  
  **BEFORE INSERT ON** `STUDENTS`  
  **FOR EACH ROW**  
  **DECLARE** `next_id` `integer`  
  **BEGIN**  
  ```sql
  SELECT seq_ex.NEXTVAL INTO next_id
  FROM DUAL
  :new.ID := next_id;
  END;
  /
  ```
In closing

- CHECK constraints and ASSERTIONS follow a declarative approach to integrity constraint enforcement
- TRIGGERs take a procedural approach