Semantic Constraints

- Used for application dependent conditions
- Each row in table must satisfy condition
- Example: limit attribute values

```
CREATE TABLE Transcript (
   StudId: INTEGER,
   CrsCode: CHAR(6),
   Semester: CHAR(6),
   Grade: CHAR(1),
   CHECK (Grade IN ('A', 'B', 'C', 'D', 'F')),
   CHECK (StudId > 0 AND StudId < 1000000000) )
```

32

Semantic Constraints (con't)

- Example: relate values of attributes in different columns

```
CREATE TABLE Employee (
   Id: INTEGER,
   Name: CHAR(20),
   Salary: INTEGER,
   MngrSalary: INTEGER,
   CHECK ( MngrSalary > Salary))
```

33

Constraints - Problems

- **Problem 1**: Empty table always satisfies all CHECK constraints

```
CREATE TABLE Employee (
   Id: INTEGER,
   Name: CHAR(20),
   Salary: INTEGER,
   MngrSalary: INTEGER,
   CHECK ( 0 < (SELECT COUNT (*) FROM Employee)) )
```

  - If Employee is empty, there are no rows on which to evaluate the CHECK condition.

34

Constraints - Problems

- **Problem 2**: Interrelational constraints are asymmetric

```
CREATE TABLE Employee (
   Id: INTEGER,
   Name: CHAR(20),
   Salary: INTEGER,
   MngrSalary: INTEGER,
   CHECK ((SELECT COUNT (*) FROM Manager) <
            (SELECT COUNT (*) FROM Employee)) )
```

  - Why should constraint be in Employee an not Manager?
  - What if Employee is empty?

35

Assertion

- Element of schema (like table)
- Symmetrically specifies an inter-relational constraint
- Applies to entire database (not just the individual rows of a single table)
  - hence it works even if Employee is empty

```
CREATE ASSERTION DontFireEveryone
   CHECK (0 < SELECT COUNT (*) FROM Employee)
```

36

Assertion

```
CREATE ASSERTION KeepEmployeesDown
   CHECK (NOT EXISTS(
            SELECT * FROM Employee E
            WHERE E.Salary > E.MngrSalary))
```

37

Assertions and Inclusion Dependency

```
CREATE ASSERTION NoEmptyCourses
  CHECK (NOT EXISTS (
                SELECT * FROM Teaching T
                WHERE  -- for each row in T check
                       -- the following condition
                   NOT EXISTS (
                SELECT * FROM Transcript R
                WHERE T.CrsCode = R.CrsCode
                  AND T.Semester = R.Semester) ) )
```

Courses with no students

Students in a particular course.

Constraint Checking

- Automatically checked
  - After each statement (IMMEDIATE) or
  - After a transaction completes (DEFERRED)
- Protects database from errors
- Enforces enterprise rules

Domains

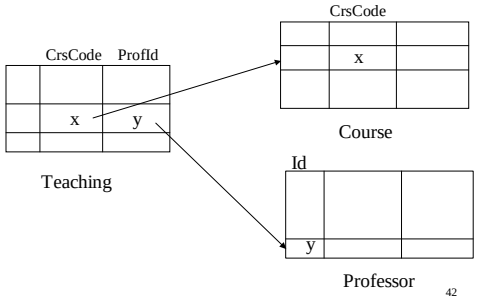
- Possible attribute values can be specified
  - Using a CHECK constraint or
  - Creating a new domain
- Domain can be used in several declarations
- Domain is a schema element

```
CREATE DOMAIN Grades CHAR (1)
   CHECK (VALUE IN ('A', 'B', 'C', 'D', 'F'))
CREATE TABLE Transcript (
   …. Grade: GRADES, …..)
```

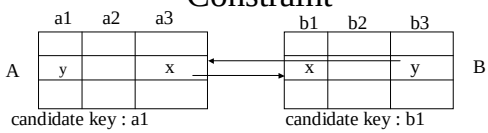

Foreign Key Constraint

```
CREATE TABLE Teaching (
   ProfId: INTEGER,
   CrsCode: CHAR (6),
   Semester: CHAR (6),
   PRIMARY KEY (CrsCode, Semester),
   FOREIGN KEY (CrsCode) REFERENCES Course,
   FOREIGN KEY (ProfId) REFERENCES Professor (Id) )
```


Foreign Key Constraint



Circularity in Foreign Key Constraint



candidate key : a1
foreign key : a3 references B(b1)

candidate key : b1
foreign key : b3 references A(a1)

Problem 1: Creation of A requires existence of B and vice versa

Solution 1:

```
CREATE TABLE A ( ……),   (* no foreign key *)
CREATE TABLE B ( ……),   (* foreign key included *)
ALTER TABLE A
   ADD CONSTRAINT cons
      FOREIGN KEY (a3) REFERENCES B (b1)
```

Circularity in Foreign Key Constraint

- Problem 2 : Insertion of row in A requires prior existence of row in B and vice versa
- Solution 2 : DEFERRED constraint check - insert both rows within a single transaction

```
CREATE TABLE  Dept (…..
   DeptId: CHAR (4),
   MngrId : INTEGER,
   PRIMARY KEY DeptId,
   FOREIGN KEY (MngrId) REFERENCES Employee (Id) )
CREATE TABLE Employee ( ….
   DeptId : CHAR (4),
   Id : INTEGER,
   PRIMARY KEY Id,
   FOREIGN KEY (DeptId) REFERENCES Dept ( DeptId) )
```

44

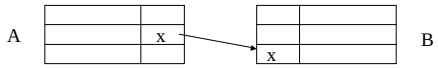
Recognizing vs. Handling Events

- Constraints enable DBMS to recognize a bad state and reject the statement or transaction that creates it
- More generally, it would be nice to have a mechanism that allows a user to specify an action to be taken if a particular situation (possibly a transition to a bad state) occurs
- SQL-92 provides a limited form of such a mechanism for handling foreign key violations

45

Handling Foreign Key Violations

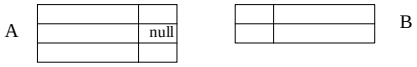
- Insertion into A: Reject if no row exists in B containing foreign key of inserted row
- Deletion from B:
  - NO ACTION: Reject if row(s) in A references row to be deleted (default response)



46

Handling Foreign Key Violations

- Deletion from B (con't):
  - SET NULL: Set value of foreign key in referencing row(s) in A to null



47

Handling Foreign Key Violations

- Deletion from B (con't):
  - SET DEFAULT: Set value of foreign key in referencing row(s) in A to default value (y)

48

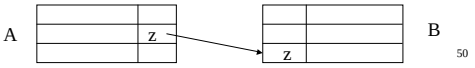
Handling Foreign Key Violations

- Deletion from B (con't):
  - CASCADE: Delete referencing row(s) in A as well

49

Handling Foreign Key Violations

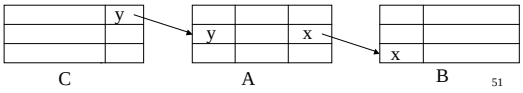
- Update foreign key in A: Reject if no row exists in B containing new foreign key
- Update candidate key in B (to z)
  - NO ACTION: Reject if row(s) in A references row to be updated (default response)
  - SET NULL: Set value of foreign key to null
  - SET DEFAULT: Set value of foreign key to default
  - CASCADE: Propagate z to foreign key



50

Handling Foreign Key Violations

- The action taken to repair the violation of a foreign key constraint in A may cause a violation of a foreign key constraint in C
  - The action specified in C controls how that violation is handled;
  - If the entire chain of violations cannot be resolved, the initial deletion from B is rejected.



51

Specifying Actions

```
CREATE TABLE Teaching (
   ProfId INTEGER,
   CrsCode CHAR (6),
   Semester CHAR (6),
   PRIMARY KEY (CrsCode, Semester),
   FOREIGN KEY (ProfId) REFERENCES Professor (Id)
      ON DELETE NO ACTION
      ON UPDATE CASCADE,
   FOREIGN KEY (CrsCode) REFERENCES Course (CrsCode)
      ON DELETE SET NULL
      ON UPDATE CASCADE )
```

52

Triggers

- A more general mechanism for handling events
  - Not in SQL-92, but is in SQL:1999
- Schema element (like table, assertion, …)

```
CREATE TRIGGER CrsChange
   AFTER UPDATE OF CrsCode, Semester ON Transcript
   WHEN  (Grade IS NOT NULL)
      ROLLBACK
```

53