Views

- Used as a relation, but rows are not physically stored.
  - The contents of a view is *computed* when it is used within an SQL statement
- View is the result of a SELECT statement over other views and base relations
- When used in an SQL statement, the view definition is substituted for the view name in the statement
  - SELECT statement can be nested in FROM clause

53

View - Example

```
CREATE VIEW CumGpa (StudId, Cum) AS
  SELECT T.StudId, AVG (T.Grade)
  FROM Transcript T
  GROUP BY T.StudId

SELECT S.Name, C.Cum
FROM CumGpa C, Student S
WHERE C.StudId = S.StudId AND C.Cum > 3.5
```

54

View Benefits

- *Access Control*: Users not granted access to base tables. Instead they are granted access to the view of the database appropriate to their needs.
  - *External schema* is composed of views.
  - View allows owner to provide SELECT access to a subset of columns (analogous to providing UPDATE and INSERT access to a subset of columns)

55

Views - Limiting Visibility

```
CREATE VIEW PartOfTranscript (StudId, CrsCode, Semester) AS
   SELECT T. StudId, T.CrsCode, T.Semester      -- limit columns
    FROM Transcript T
    WHERE T.Semester = 'S2000'                  -- limit rows
```

Give permissions to access data through view:

```
GRANT SELECT ON PartOfTranscript TO joe
```

This is analogous to:

```
GRANT SELECT (Grade) ON Transcript TO joe
```

on regular tables

56

View Benefits (con't)

- *Customization*: Users need not see full complexity of database. View creates the illusion of a simpler database customized to the needs of a particular category of users
- A view is *similar in many ways to a subroutine* in standard programming
  - Can be used in multiple queries

57

Nulls

- *Conditions*: *x op y* (where *op* is <, >, <>, =, etc.) has value *unknown* (*U*) when either x or y is null
  - WHERE T.*cost* > T.*price*
- *Arithmetic expression*: x *op y* (where *op* is +, -, *, etc.) has value NULL if x or y is NULL
  - WHERE (T. *price*/T.*cost*) > 2
- *Aggregates*: COUNT counts NULLs like any other value; other aggregates ignore NULLs

```
SELECT COUNT (T.CrsCode), AVG (T.Grade)
FROM   Transcript T
WHERE T.StudId = '1234'
```

58

Nulls (con't)

- WHERE clause uses a *three-valued logic* to filter rows. Portion of truth table:

C1	*C2*	*C1* AND *C2*	*C1* OR *C2*
T	U	U	T
F	U	F	U
U	U	U	U

- Rows are discarded if WHERE condition is *false* or *unknown*
- Ex: WHERE T.*CrsCode* = 'CS305' AND T.*Grade* > 2.5

59

Modifying Tables - Insert

- Inserting a single row into a table
  - Attribute list can be omitted if it is the same as in CREATE TABLE (but do not omit it)
  - NULL and DEFAULT values can be specified

```
INSERT INTO  Transcript(StudId, CrsCode, Semester, Grade)
VALUES (12345, 'CSE305', 'S2000',  NULL)
```

60

Bulk Insertion

- Insert the rows output by a SELECT

```
CREATE TABLE DeansList (
      StudId       INTEGER,
      Credits      INTEGER,
      CumGpa       FLOAT,
      PRIMARY KEY StudId )

INSERT INTO  DeansList (StudId, Credits, CumGpa)
SELECT       T.StudId, 3 * COUNT (*), AVG(T.Grade)
FROM         Transcript T
GROUP BY     T.StudId
HAVING  AVG (T.Grade) > 3.5 AND COUNT(*) > 30
```

61

Modifying Tables - Delete

- Similar to SELECT except:
  - No project list in DELETE clause
  - No Cartesian product in FROM clause (only 1 table name)
  - Rows satisfying WHERE clause (general form, including subqueries, allowed) are deleted instead of output

```
DELETE FROM Transcript T
WHERE T.Grade IS NULL AND
      T.Semester <> 'S2000'
```

62

Modifying Data - Update

```
UPDATE  Employee E
SET     E.Salary = E.Salary * 1.05
WHERE   E.Department = 'research'
```

- Updates rows in a single table
- All rows satisfying WHERE clause (general form, including subqueries, allowed) are updated

63

Updating Views

- Question: Since views look like tables to users, can they be updated?
- Answer: Yes – a view update changes the underlying base table to produce the requested change to the view

```
CREATE VIEW   CsReg (StudId, CrsCode, Semester) AS
SELECT        T.StudId, T. CrsCode, T.Semester
FROM          Transcript T
WHERE    T.CrsCode LIKE 'CS%' AND T.Semester='S2000'
```

64

Updating Views - Problem 1

INSERT INTO CsReg (*StudId*, *CrsCode*, *Semester*)
VALUES (1111, 'CSE305', 'S2000')

- **Question**: What value should be placed in attributes of underlying table that have been projected out (e.g., *Grade*)?
- **Answer**: NULL (assuming null allowed in the missing attribute) or DEFAULT

65

Updating Views - Problem 2

INSERT INTO CsReg (*StudId*, *CrsCode*, *Semester*)
VALUES (1111, 'ECO105', 'S2000')

- **Problem**: New tuple not in view
- **Solution**: Allow insertion (assuming the WITH CHECK OPTION clause has not been appended to the CREATE VIEW statement)

66

Updating Views - Problem 3

- Update to the view might *not* *uniquely* specify the change to the base table(s) that results in the desired modification of the view

```
CREATE VIEW   ProfDept (PrName, DeName)  AS
SELECT    P.Name, D.Name
FROM      Professor P, Department D
WHERE     P.DeptId = D.DeptId
```

67

Updating Views - Problem 3 (con't)

- Tuple <Smith, CS> can be deleted from ProfDept by:
  - Deleting row for Smith from Professor (but this is inappropriate if he is still at the University)
  - Deleting row for CS from Department (not what is intended)
  - Updating row for Smith in Professor by setting *DeptId* to null (seems like a good idea)

68

Updating Views - Restrictions

- Updatable views are restricted to those in which
  - No Cartesian product in FROM clause
  - no aggregates, GROUP BY, HAVING
  - …

For example, if we allowed:

```
CREATE VIEW AvgSalary (DeptId, Avg_Sal ) AS
   SELECT  E.DeptId, AVG(E.Salary)
    FROM   Employee E
    GROUP BY E.DeptId
```

then how do we handle:

```
UPDATE AvgSalary
   SET Avg_Sal = 1.1 * Avg_Sal
```

69