Views

- Schema element
- Part of external schema
- A (virtual) table constructed from (actual) tables
  - Can be accessed in queries like any other table
  - Not materialized, constructed when accessed
  - Similar to a subroutine in ordinary programming

54

Views - Examples

```
CREATE VIEW CoursesTaken (StudId, CrsCode, Semester) AS
  SELECT  T.StudId, T.CrsCode, T.Semester
  FROM Transcript T
```

-- part of external schema suitable for use in Bursar's office

```
CREATE VIEW CoursesITook (CrsCode, Semester, Grade) AS
  SELECT  T.CrsCode, T.Semester, T.Grade
  FROM  Transcript T
  WHERE  T.StudId = '123456789'
```

-- part of external schema suitable for use by student with Id 123456789

55

Modifying the Schema

```
ALTER TABLE Student
  ADD COLUMN Gpa INTEGER DEFAULT 0

ALTER TABLE Student
  ADD CONSTRAINT GpaRange
    CHECK (Gpa >= 0 AND Gpa <= 4)

ALTER TABLE  Transcript
  DROP CONSTRAINT Cons    --constraint names are useful

DROP TABLE Employee

DROP ASSERTION DontFireEveryone
```

56

Access Control

- Database might contain sensitive information
- Access has to be limited:
  - Users have to be identified – ***authentication***
    - Generally done with passwords
  - Each user must be limited to modes of access appropriate to that user - ***authorization***
- SQL:92 provides tools for specifying an authorization policy but does not support authentication (vendor specific)

57

Controlling Authorization in SQL

```
GRANT access_list
  ON table
  TO user_list
```

access modes: SELECT, INSERT, DELETE, UPDATE, REFERENCE

```
GRANT UPDATE (Grade) ON Transcript TO prof_smith
```

- Only the Grade column can be updated

```
GRANT SELECT ON Transcript TO joe
```

- Individual columns cannot be specified for SELECT access; all columns of Transcript can be read
- Access to individual columns controlled through views

```
GRANT SELECT ON CoursesTaken TO joe
```

58

Controlling Authorization in SQL Using Views

```
GRANT access
  ON view
  TO user_list
```

59

Authorization: REFERENCE

- Foreign key constraint enforces relationship between tables that can be exploited to:

  - Control access:

```
CREATE TABLE DontDismissMe (
  Id INTEGER,
  FOREIGN KEY (Id) REFERENCES Student
    ON DELETE NO ACTION
        --prevents deletion of referenced row )
```

  - Reveal information: successful insertion into DontDissmissMe means a row with foreign key value exists in Student

```
INSERT INTO DontDismissMe ('111111111')
```

60