Relational Model 2

12 November 2012
Lecture 5

Topics for Today

- Enforcing Integrity Constraints
  - Transactions and Constraints
- Querying Relational Data
- Introduction to Views
  - Updates on Views
- Destroying/Altering Tables and Views

Source: Ramakrishnan and Gehrke Chapter 3

What to check

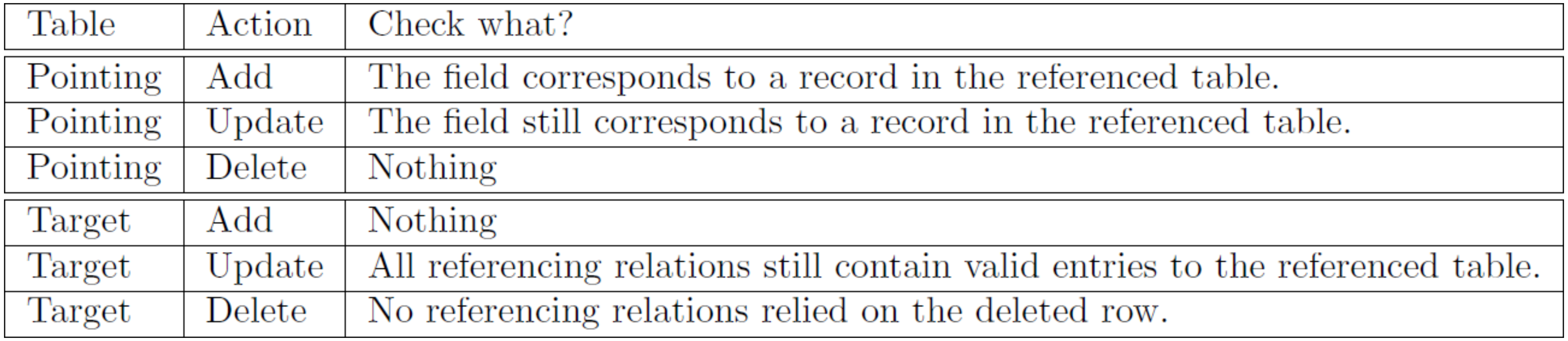


Table	Action	Check what?
Pointing	Add	The field corresponds to a record in the referenced table.
Pointing	Update	The field still corresponds to a record in the referenced table.
Pointing	Delete	Nothing
Target	Add	Nothing
Target	Update	All referencing relations still contain valid entries to the referenced table.
Target	Delete	No referencing relations relied on the deleted row.

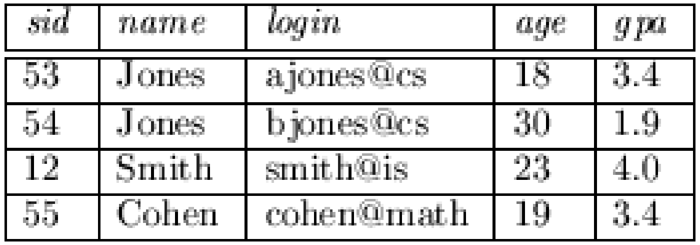
What happens

- In the pointing relation:
  - On Add/Update: If the new value is not in the target relation, the action is blocked
- In the target relation, the user can decide:
  - NO ACTION (the default condition)
  - CASCADE
  - SET DEFAULT
  - SET NULL
- One choice for Delete, one choice for Update
- Let's see some examples…

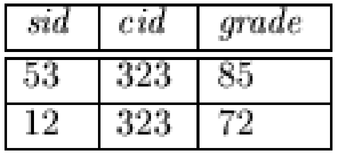
Foreign Key Example

- DELETE FROM Students
  WHERE sid = 53

- UPDATE Students SET sid = 13 WHERE sid = 12



sid	*name*	*login*	*age*	*gpa*
53	Jones	ajones@cs	18	3.4
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

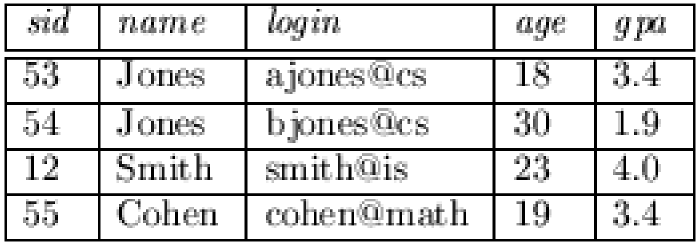


sid	*cid*	*grade*
53	323	85
12	323	72

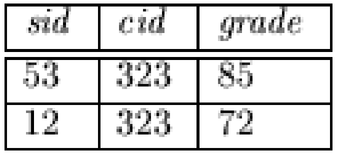
ON Delete No Action

- DELETE FROM Students WHERE sid = 53

Before

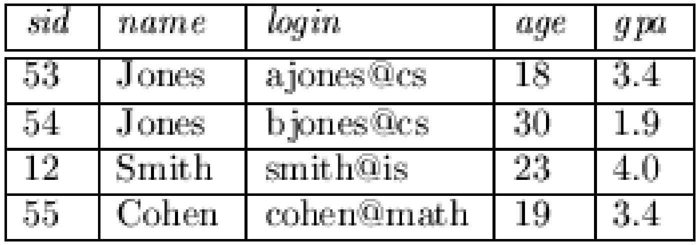


sid	*name*	*login*	*age*	*gpa*
53	Jones	ajones@cs	18	3.4
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

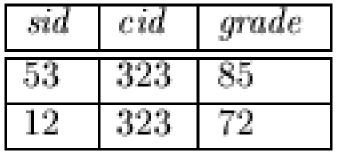


sid	*cid*	*grade*
53	323	85
12	323	72

After



sid	*name*	*login*	*age*	*gpa*
53	Jones	ajones@cs	18	3.4
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

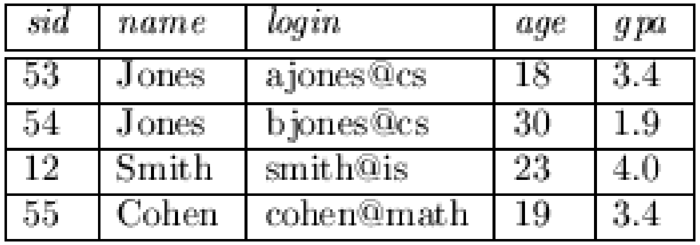


sid	*cid*	*grade*
53	323	85
12	323	72

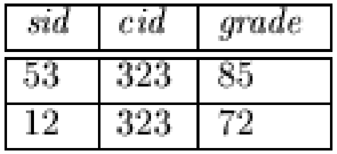
ON Delete Cascade

- DELETE FROM Students WHERE sid = 53

Before

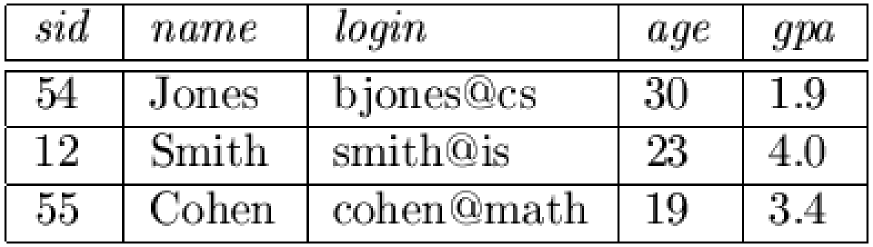


sid	*name*	*login*	*age*	*gpa*
53	Jones	ajones@cs	18	3.4
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

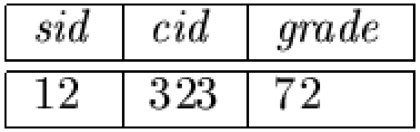


sid	*cid*	*grade*
53	323	85
12	323	72

After



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

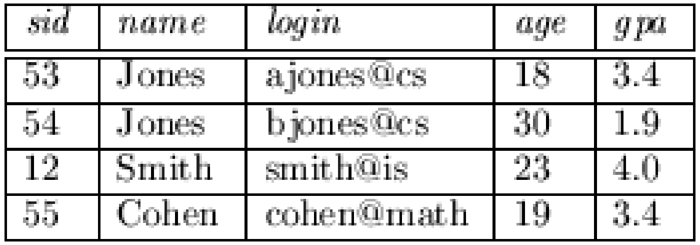


sid	*cid*	*grade*
12	323	72

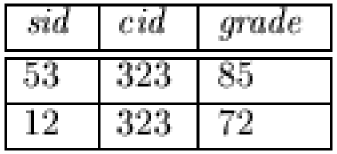
ON Delete Set Default

- DELETE FROM Students WHERE sid = 53

Before

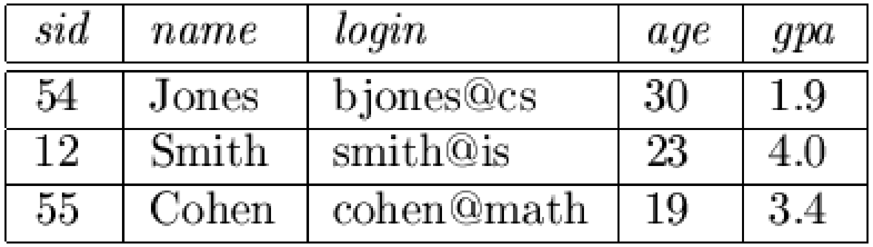


sid	*name*	*login*	*age*	*gpa*
53	Jones	ajones@cs	18	3.4
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

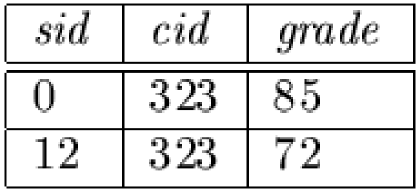


sid	*cid*	*grade*
53	323	85
12	323	72

After



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4



sid	*cid*	*grade*
0	323	85
12	323	72

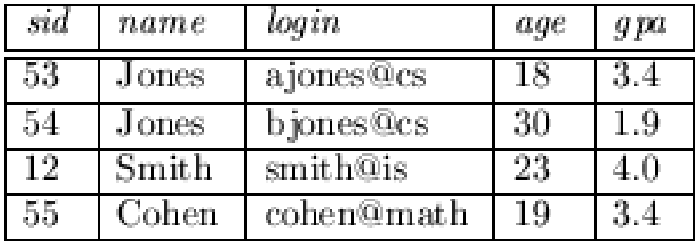
Assuming Enrolled defined sid as follows:

sid char(20) default ‘0’

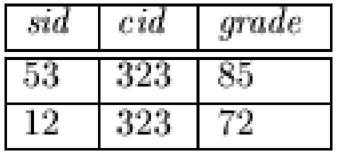
ON Delete Set Null

- DELETE FROM Students WHERE sid = 53

Before

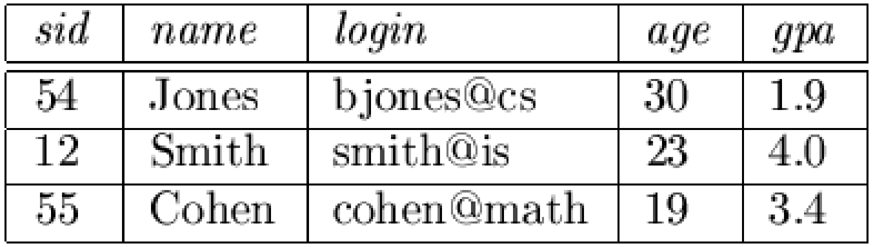


sid	*name*	*login*	*age*	*gpa*
53	Jones	ajones@cs	18	3.4
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

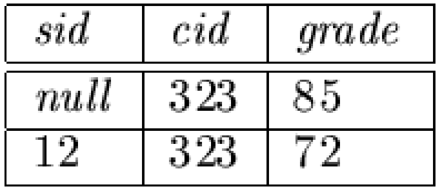


sid	*cid*	*grade*
53	323	85
12	323	72

After



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

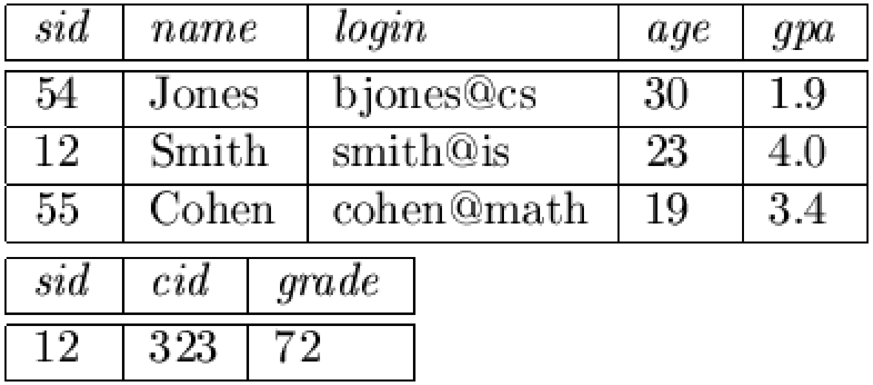


sid	*cid*	*grade*
null	323	85
12	323	72

ON Update No Action

- UPDATE Students SET sid = 13 WHERE sid = 12

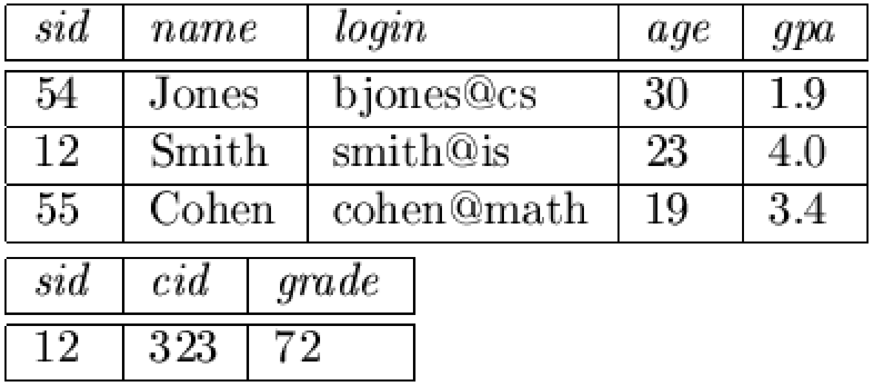
Before



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
12	323	72

After



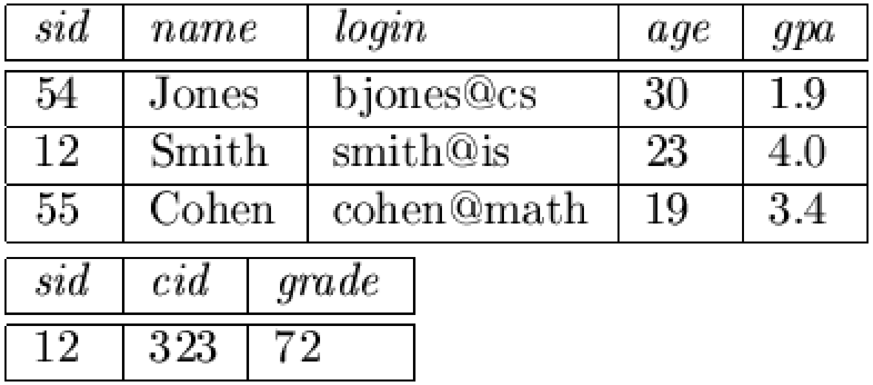
sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
12	323	72

ON Update Cascade

- UPDATE Students SET sid = 13 WHERE sid = 12

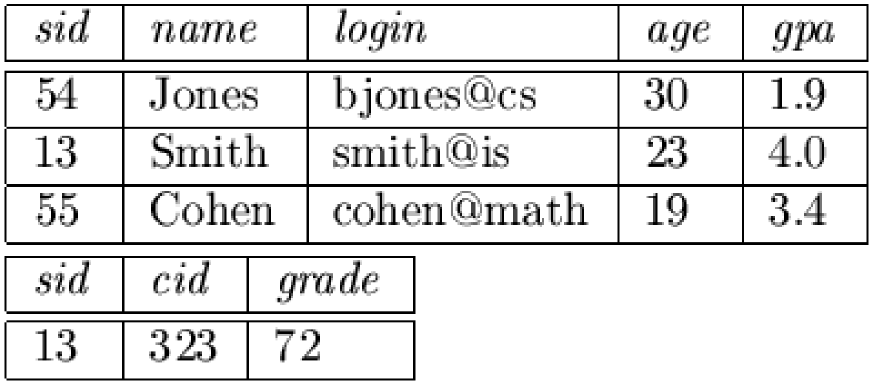
Before



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
12	323	72

After



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
13	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
13	323	72

ON Update Set Default

- UPDATE Students SET sid = 13 WHERE sid = 12

Before

sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
12	323	72

After

sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
13	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
0	323	72

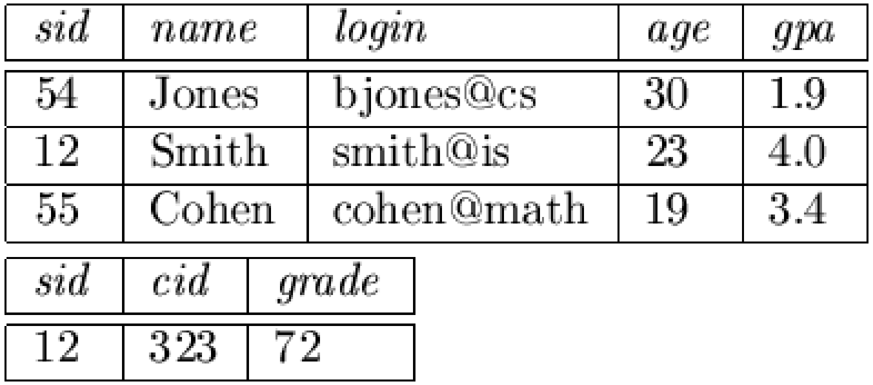
Assuming Enrolled defined sid as follows:

sid char(20) default ‘0’

ON Update Set Null

- UPDATE Students SET sid = 13 WHERE sid = 12

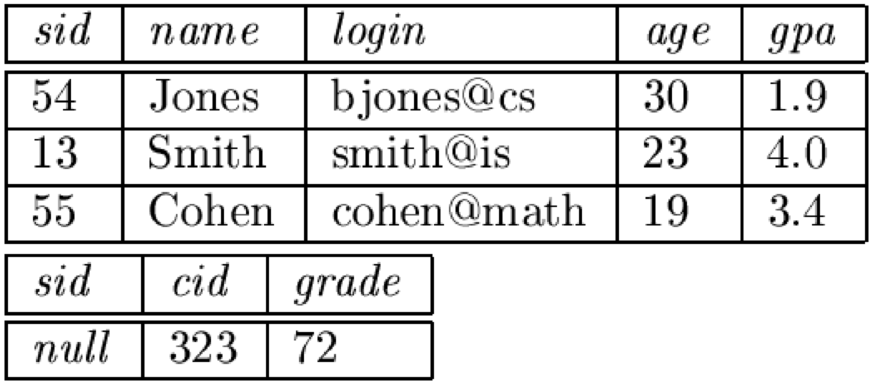
Before



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
12	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
12	323	72

After



sid	*name*	*login*	*age*	*gpa*
54	Jones	bjones@cs	30	1.9
13	Smith	smith@is	23	4.0
55	Cohen	cohen@math	19	3.4

sid	*cid*	*grade*
null	323	72

Transactions and Constraints

- By default, constraints are checked at the end of each SQL statement

- Sometimes that is not the best time to check constraints
  - Complex conditions
  - Deleting, then adding rows in a complex network of foreign keys

- We can instruct the DBMS to wait to check the constraints until the <u>the end of the transaction</u> instead
  - If the constraint is called “CstNames” we write:

```
SET CONSTRAINT CstNames DEFERRED
```

[Valid only for Oracle DB]

So Far

- Enforcing Integrity Constraints
  - Transactions and Constraints
- Querying Relational Data
- Introduction to Views
  - Updates on Views
- Destroying/Altering Tables and Views

Queries and SQL

- We saw some implicit queries before, now let’s talk a bit more about them
  - We’ll see SQL in more depth next lecture
- SQL is the most popular **relational query language**
  - There are others
  - There are also many varieties of SQL (dialects) per vendor: PL/SQL, Transact-SQL, etc.
  - The standard for all of what we’ll see this semester is SQL-92
- SQL is a (4th generation) programming language
  - We describe what data we want to see and the DBMS goes out and fetches it
  - Learning to express what you want is the hardest part
  - It’s easy to write very computationally intensive queries

Example Queries

- Show the students who are older than 18

- SELECT * FROM Students S WHERE S.age > 18
  - * means to show all of the columns in the relation
  - S is an alias for Students (we could have chosen anything)

- SELECT T.name, T.age FROM Students T WHERE T.age > 18
  - This shows only two columns – name and age
  - T is an alias now (note that it works backwards)

- SELECT S.name FROM Students S WHERE S.age > 18
  - This shows just name (we don’t need to use the filtering condition column)

Example Queries

- Show the students who are over 18 and have a GPA less than 80

- SELECT * FROM Students S WHERE S.age > 18 AND S.gpa < 80
  - Two conditions – only rows which fulfill both of the conditions are included

= =

- For each student who received a 90 in a course, print the student's name and course number
- SELECT S.name, E.cid FROM Students S, Enrolled E
  WHERE S.sid = E.sid AND E.grade = 90
  - This combines data from two relations
  - Called a **JOIN** on Students and Enrolled (Inner Join)
  - Computes all pairs of rows between Students and Enrolled
  - Filters the pairs based on the WHERE condition
  - One of the most common uses of foreign keys

So Far

- Enforcing Integrity Constraints
  - Transactions and Constraints
- Querying Relational Data
- Introduction to Views
  - Updates on Views
- Destroying/Altering Tables and Views

Introduction to Views

- A view is a *virtual relation* built from other relations (or views)

- To the user, a view behaves exactly like a normal relation
  - Under the hood, it is recalculated each time it is accessed

- To define a view of all students whose have an 80 in a course:

```
CREATE VIEW B-Students (name, sid, course)
    AS SELECT S.name, S.sid, E.cid
    FROM Students S, Enrolled E
    WHERE S.sid = E.sid AND E.grade = 80
```

Why use views?

- Provide *Logical Data Independence*
  - We can rewrite the view's logic and not change the reading applications
  - We can modify or update the queried relations without affecting the view's interface

- Example: We update the relations, breaking grades out of Enrolled and putting them in separate relation:
  - Students (sid, name, age, gpa)
  - Enrolled(sid, cid)
  - Grades(sid, cid, semester, grade)
- We can then rewrite the view:
  - CREATE VIEW B-Students (name, sid, course)

    AS SELECT S.name, S.sid, G.cid

    FROM Students S, Grades G

    WHERE S.sid = G.sid AND G.grade = 80

Why use views?

- Provides information hiding and security
  - Can produce derivative relations which hide sensitive data

- Example: Produce a list of students enrolled in a particular class, but don't show their GPA

  -
    ```
    CREATE VIEW DB-Enrolled (name, sid, age)
    AS SELECT S.name, S.sid, S.age
    FROM Students S, Enrolled E
    WHERE S.sid = E.sid AND E.cid = '102322'
    ```

Updating Views

- What about updating views?
  - In a word – don't
  - But if you must…

- SQL-92 allows view updates **if:** <u>They are derived from a single relation and have no aggregation</u>
  - Aggregation – averaging columns, counting rows, etc. (next time)

- If you meet the condition:
  - If one update (or delete) one which row represents multiple rows in the source view, *all source rows are updated (or deleted)*
  - If you add a new row – missing columns are put in **null**
    - If that's illegal, you can't add a row

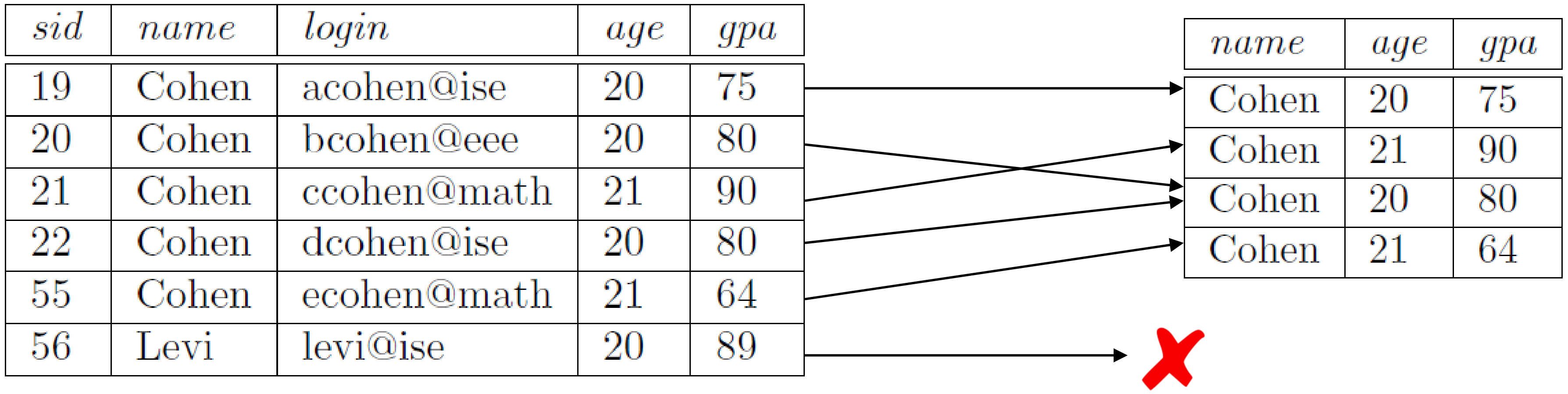
One row is many

Removes duplicate rows

Cohens:

CREATE VIEW Cohens(name,
    age, gpa) AS
SELECT DISTINCT S.name,
    S.age, S.gpa
FROM Students S
WHERE S.name = ’Cohen’

Students:



sid	*name*	*login*	*age*	*gpa*
19	Cohen	acohen@ise	20	75
20	Cohen	bcohen@eee	20	80
21	Cohen	ccohen@math	21	90
22	Cohen	dcohen@ise	20	80
55	Cohen	ecohen@math	21	64
56	Levi	levi@ise	20	89

name	*age*	*gpa*
Cohen	20	75
Cohen	21	90
Cohen	20	80
Cohen	21	64

One row is many

UPDATE Cohens SET gpa = 81 WHERE age = 20 and gpa = 80

Before:

name	*age*	*gpa*
Cohen	20	75
Cohen	21	90
Cohen	20	80
Cohen	21	64

sid	*name*	*login*	*age*	*gpa*
19	Cohen	acohen@ise	20	75
20	Cohen	bcohen@eee	20	80
21	Cohen	ccohen@math	21	90
22	Cohen	dcohen@ise	20	80
55	Cohen	ecohen@math	21	64
56	Levi	levi@ise	20	89

After:

name	*age*	*gpa*
Cohen	20	75
Cohen	21	90
Cohen	20	**81**
Cohen	21	64

sid	*name*	*login*	*age*	*gpa*
19	Cohen	acohen@ise	20	75
20	Cohen	bcohen@eee	20	**81**
21	Cohen	ccohen@math	21	90
22	Cohen	dcohen@ise	20	**81**
55	Cohen	ecohen@math	21	64
56	Levi	levi@ise	20	89

Adding a row

INSERT INTO Cohens (name, age, gpa) VALUES (‘Cohen’, 21, 89)

Before:

sid	*name*	*login*	*age*	*gpa*
19	Cohen	acohen@ise	20	75
20	Cohen	bcohen@eee	20	80
21	Cohen	ccohen@math	21	90
22	Cohen	dcohen@ise	20	80
55	Cohen	ecohen@math	21	64
56	Levi	levi@ise	20	89

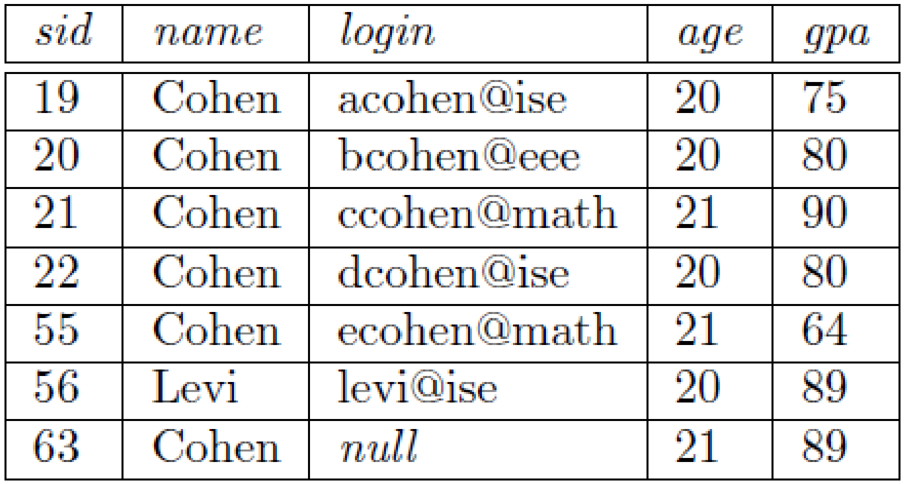
After:

sid	*name*	*login*	*age*	*gpa*
19	Cohen	acohen@ise	20	75
20	Cohen	bcohen@eee	20	80
21	Cohen	ccohen@math	21	90
22	Cohen	dcohen@ise	20	80
55	Cohen	ecohen@math	21	64
56	Levi	levi@ise	20	89
null	Cohen	*null*	21	89

Adding a row

```
CREATE VIEW Cohens2 (id, name, age, gpa)
AS SELECT S.sid, S.name, S.age, S.gpa
FROM Students S
WHERE S.name = ’Cohen’

INSERT INTO Cohens2 (id, name, age, gpa) VALUES (63,
   ’Cohen’, 21, 89)
```



sid	*name*	*login*	*age*	*gpa*
19	Cohen	acohen@ise	20	75
20	Cohen	bcohen@eee	20	80
21	Cohen	ccohen@math	21	90
22	Cohen	dcohen@ise	20	80
55	Cohen	ecohen@math	21	64
56	Levi	levi@ise	20	89
63	Cohen	*null*	21	89

Invisible Updates

- You can add a row to view which is invisible

- INSERT INTO Cohens2 (id, name, age, gpa) VALUES (65, ’Levi’, 30, 88)
  - The new row is inserted into Students, but is invisible in Cohens2

- You can prevent this using the WITH CHECK OPTION command at the end of the CREATE VIEW command
  - What if a view is built on another view and one has the WITH CHECK OPTION and the other doesn’t?
  - We won’t talk more about this – just be careful

So Far

- Enforcing Integrity Constraints
  - Transactions and Constraints
- Querying Relational Data
- Introduction to Views
  - Updates on Views
- Destroying/Altering Tables and Views

Destroying Tables and Views

- To delete a table:
  - DROP TABLE Students
- To delete a view:
  - DROP VIEW Cohens

- What about ICs?
  - DROP TABLE Students RESTRICT (only drops if it won't affect others) *(Default)*
  - DROP TABLE Students CASCADE (drops all other dependent tables)
    - Not supported in MS SQL
- Same for views

- To just delete all rows and leave the schema – use TRUNCATE or DELETE FROM Students

Altering Tables and Views

- To alter a table – use ALTER TABLE
- To add a column(s):
  - ALTER TABLE Students ADD maiden-name CHAR(10)
  - ALTER TABLE Students ADD birthYear int, birthMonth int
- To delete a column:
  - ALTER TABLE Students DROP COLUMN maiden-name
- To modify a column:
  - ALTER TABLE Students ALTER COLUMN birthMonth varchar(10)
- To add a foreign key:
  - ALTER TALBE Courses ADD FOREIGN KEY (sid) REFERENCES Students

- Similar for views – ALTER VIEW

Conclusion

- Enforcing Integrity Constraints
  - Transactions and Constraints
- Querying Relational Data
- Introduction to Views
  - Updates on Views
- Destroying/Altering Tables and Views