COMP 430
Intro. to Database Systems
Modifying data & schemas
Add, modify, delete data
Data not static

Databases are built to be used.
Forgetting this is a common student mistake.
Adding data – review

INSERT INTO Student VALUES ("S123456789", "John", "Doe", "Houston", "TX");

INSERT INTO Student VALUES ("S111111111", "Mary", "Jones", "Houston", "TX"),
    ("S222222222", "Joe", "Wallis", "Kansas City", "KS");

INSERT INTO Student (id, first_name, last_name) VALUES ("S987654321", "Jane", "Smith");
Saving query results into a table

```
SELECT ...;
INTO new_table_name
...;

INSERT INTO Student
SELECT ...;

INSERT INTO Student (id, first_name, last_name)
SELECT ...;
```

Existing table

New table
Modifying & deleting data

```sql
UPDATE Student
SET city = "Austin", state = "TX"
WHERE id = "S123456789";
```

```sql
DELETE FROM Student;
```

```sql
DELETE FROM Student
WHERE ...;
```
What’s the issue?

Student (id, first_name, last_name, city, state)
Enrollment (student_id, crn)
Course (crn, dept_code, number, title)

DELETE FROM Student
WHERE id = “S123456789”;
Modifying & deleting data affects other tables

Student (id, first_name, last_name, city, state)
Enrollment (student_id, crn)
Course (crn, dept_code, number, title)

Need to maintain referential integrity!

DELETE FROM Student
WHERE id = “S123456789”;

By default, UPDATE & DELETE fail when they would break referential integrity.
Modifying & deleting data affects other tables

Student \((id, \text{first\_name}, \text{last\_name}, \text{city}, \text{state})\)
Enrollment \((\text{student\_id}, \text{crn})\)
Course \((\text{crn}, \text{dept\_code}, \text{number}, \text{title})\)

```
CREATE TABLE Enrollment (  
    ...  
    FOREIGN KEY student_id REFERENCES Student (id)  
    ON UPDATE ...,  
    FOREIGN KEY crn REFERENCES Course (crn)  
    ON DELETE ...  
);
```

- RESTRICT
- NO ACTION
- SET NULL
- SET DEFAULT
- CASCADE
<table>
<thead>
<tr>
<th>Natural</th>
<th>Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can get a new SSN in cases of stolen identity.</td>
<td>Should never change.</td>
</tr>
<tr>
<td>Account numbers can change when banks merge.</td>
<td></td>
</tr>
<tr>
<td>Fits with ON UPDATE CASCADE.</td>
<td>Fits with ON UPDATE RESTRICT / NO ACTION.</td>
</tr>
</tbody>
</table>
Hard vs. soft deletes

Hard I want to delete the data, so dang it, delete it!
    Fits with ON DELETE CASCADE.

Soft I might want this data later, after all.
    Delete = Flag this data as “inactive”.
    E.g., closed customer account or discontinued product.
    Fits with ON DELETE RESTRICT / NO ACTION.
Add, modify, delete from schema
Database schemas not static

Needs change over time.
Think about long-term plans & maintenance.
Add, modify, delete attribute

ALTER TABLE Student
ADD COLUMN zip INT;

ALTER TABLE Student
MODIFY COLUMN zip VARCHAR(9);

ALTER TABLE Student
ALTER COLUMN zip VARCHAR(9);

ALTER TABLE Student
DROP COLUMN state;
DANGER! Can change query semantics!

SELECT *
FROM ...
...;

Dangerous – avoid!
Specify columns explicitly.
Add, modify, delete constraints

See previous set of notes.
Add, modify, delete views

CREATE VIEW ... AS
SELECT ...;

ALTER VIEW ... AS
...

CREATE OR REPLACE VIEW ... AS
SELECT ...;

DROP VIEW ...;
What’s the difference?

CREATE VIEW StudentName AS
SELECT first_name, last_name
FROM Student;

SELECT first_name, last_name
INTO StudentName
FROM Student;
Process for changing schemas
Traditional – Regular shutdowns

1. Make & test changes on a test server.
2. Have regularly scheduled system shutdown.
   a) Typically late at night to minimize user disruption.
   b) Disable user access to live server.
   c) Make & test changes on the live server.

Takes time. Lots of data, possibly replicated & distributed.
Reducing downtime

Instead of a full shutdown, just block users from DB temporarily.
  • Shutdown only as long as needed.
  • “Lock” the DB.

Only need to lock tables that are changing.
  • See COMP 322 & 421 for more about locks.

For UX, assumes lock is only needed briefly.
Updating without any user disruption

E.g.:
• Create “shadow table” incorporating changes
• Create triggers in original table that forwards data updates to shadow table
• Copy original table’s data to shadow table
• Rename shadow table to replace original table

Can choose to only let some users see the changes.

Some DBMS provide support for such “online” schema changes.