CSE 180: Lab Section

- What must not happen?
- Transaction Isolation levels (Postgres)
- Adding and Dropping foreign key constraints
- Actions
- Adding CHECK constraints
- Unit testing

Tuesday, November 14th
Utkarsh Gupta

Student, Courses, & Enrolment

```
utgupta=# select * from student;
 s_id | s_name |    ssn     |    dob     | gpa  | has_grad |   phone    |      email
------+--------+------------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234       | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567       | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468       | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412       | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    5 | ABY    | 4321       | 2000-01-01 | 3.32 | t        |            |
(5 rows)
```

```
utgupta=# select * from courses;
 c_id |   c_name    | credits
------+-------------+---------
  180 | DB 1        |       5
  181 | DB 2        |       5
  215 | DB GRAD 1   |       5
  160 | CG          |       5
  280 | CV SEM      |       2
(5 rows)
```

```
utgupta=# select * from enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)
```

What must not happen?

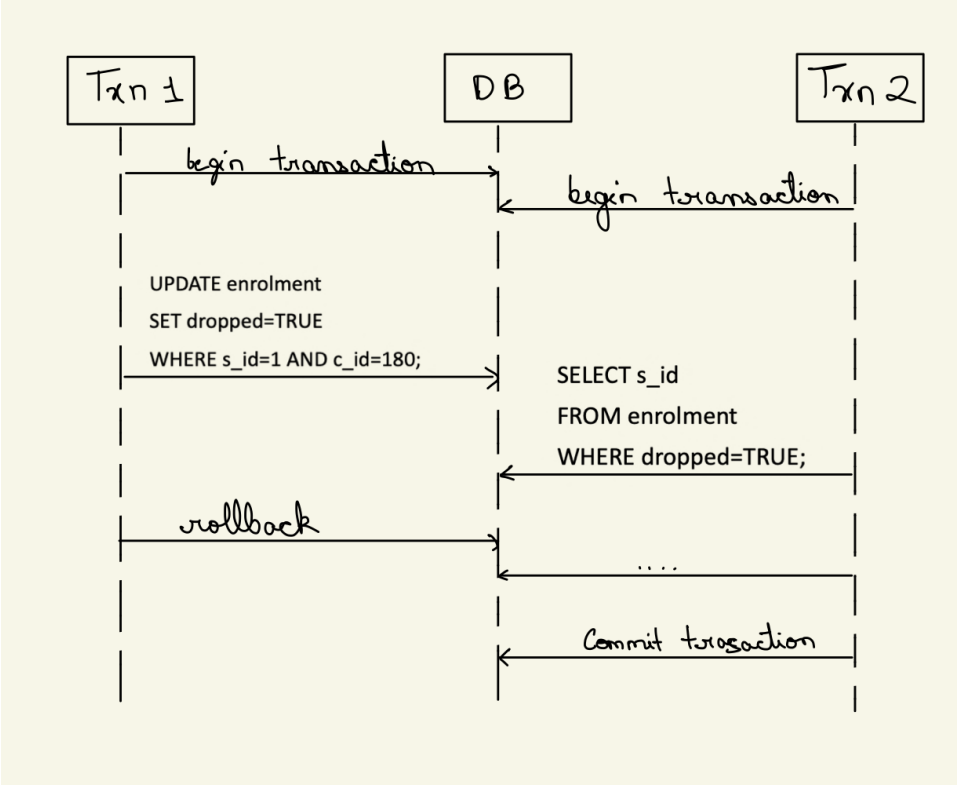
Complex backend software often leads to business-logic mistakes.

Those possible mistakes are as follows:

- Dirty read
- Lost update
- Non-repeatable read
- Phantoms
- Serialization anomaly

What must not happen? (cont.)

- **Dirty read:**
- Read uncommitted changes of other transactions



What must not happen? (cont.)

- **Lost update:**
- A lost update is when first transaction reads data into its local memory, and then the second transaction changes this data and commits its changes.
- After this, the first transaction updates the same data based on what it read into memory before the second transaction was executed.
- The update performed by the second transaction can be considered as a lost update.

Example:

1. The first transaction reads the number of dropped students for CSE 180.
2. The second transaction updates the dropped column for CSE 180 class.
3. The first transaction drops the course CSE 180 from the course table because there are very less students enrolled in it.

What must not happen? (cont.)

- **Non repeatable read:**
- One of the rows you have queried at different stages of transaction may be updated by other transactions.

Example:

1. Transaction 1 reads the student ids of CSE 180 from Enrolment table.
2. Transaction 2 modifies the dropped column for a few student of CSE 180 class.
3. Transaction 1 again reads the student ids of CSE 180 from Enrolment table.

In the above case, transaction 1 will see different/updated data.

What must not happen? (cont.)

- **Phantom:**
- New rows are added or removed by another transaction to the set of records being read.

Example:

1. Transaction 1 reads the student ids of CSE 180 from Enrolment table.
2. Transaction 2 deletes all the courses taken by s_id 1 from the Enrolment table.
3. Transaction 1 again reads the student ids of CSE 180 from Enrolment table.

In the above case, transaction 1 will see lesser data because s_id 1 has been removed from the Enrolment table.

What must not happen? (cont.)

- **Serialization Anomaly:**
- The result of successfully committing a group of transactions is inconsistent with all possible orderings of running those transactions one at a time.

Example:

1. Transaction 1 reads the data of s_id=1 from enrolment with dropped=TRUE.
2. Transaction 2 reads the data of s_id=1 from enrolment with dropped=FALSE.
3. Based on the above records, transaction 1 WILL NOT charge the fee for CSE 180 but transaction 2 WILL charge the fee for the same course.

Transaction Isolation Levels (Postgres)

- The term isolation refers to how the concurrent transactions affect each other.
- In Postgres, the Isolation levels are defined in the following way:

Level	Dirty read	Non-repeatable read	Lost update	Phantoms	Serialization Anomaly
Read committed	YES	NO	NO	NO	NO
Repeatable Read	YES	YES	YES	YES	NO
Serializable	YES	YES	YES	YES	YES

Transaction Isolation Levels (Postgres) (cont.)

- Syntax:

Read Committed:

BEGIN;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

<Query 1>

<Query 2>

…

...

…

<Query N>

COMMIT;

Transaction Isolation Levels (Postgres) (cont.)

Example

Read Committed:

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT * FROM Student;
SELECT * FROM Enrolment;
COMMIT;
```

Transaction Isolation Levels (Postgres) (cont.)

```
utgupta=# BEGIN;
BEGIN
utgupta=*# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
utgupta=*# SELECT * FROM Student;
 s_id | s_name |    ssn    |    dob     | gpa  | has_grad |   phone    |      email
------+--------+-----------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234      | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567      | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468      | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412      | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    5 | ABY    | 4321      | 2000-01-01 | 3.32 | t        |            |
(5 rows)

utgupta=*# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)

utgupta=*# COMMIT;
COMMIT
utgupta=#
```

Transaction Isolation Levels (Postgres) (cont.)

- Syntax:

Repeatable read:

BEGIN;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

<Query 1>

<Query 2>

…

…

…

<Query N>

COMMIT;

Transaction Isolation Levels (Postgres) (cont.)

Example

Repeatable read:

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT * FROM Student;
SELECT * FROM Enrolment;
COMMIT;
```

Transaction Isolation Levels (Postgres) (cont.)

```
utgupta=# BEGIN;
BEGIN
utgupta=*# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
utgupta=*# SELECT * FROM Student;
 s_id | s_name |    ssn    |    dob     | gpa  | has_grad |   phone    |      email
------+--------+-----------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234      | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567      | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468      | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412      | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    5 | ABY    | 4321      | 2000-01-01 | 3.32 | t        |            |
(5 rows)

utgupta=*# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)

utgupta=*# COMMIT;
COMMIT
utgupta=#
```

Transaction Isolation Levels (Postgres) (cont.)

- Syntax:

Serializable:

BEGIN;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

<Query 1>

<Query 2>

…

...

…

<Query N>

COMMIT;

Transaction Isolation Levels (Postgres) (cont.)

Example

Serializable:

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT * FROM Student;
SELECT * FROM Enrolment;
COMMIT;
```

Transaction Isolation Levels (Postgres) (cont.)

```
utgupta=# BEGIN;
BEGIN
utgupta=*# SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
utgupta=*# SELECT * FROM Student;
 s_id | s_name |    ssn    |    dob     | gpa  | has_grad |   phone    |      email
------+--------+-----------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234      | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567      | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468      | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412      | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    5 | ABY    | 4321      | 2000-01-01 | 3.32 | t        |            |
(5 rows)

utgupta=*# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)

utgupta=*# COMMIT;
COMMIT
utgupta=#
```

Adding and dropping a Foreign Key constraint

Dropping a foreign key constraint:

- Let us drop the constraint **enrolment_c_id_fkey.**

```
utgupta=# \d enrolment;
            Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           | not null |
 dropped | boolean |           | not null |
Indexes:
    "enrolment_pkey" PRIMARY KEY, btree (s_id, c_id)
Foreign-key constraints:
    "enrolment_c_id_fkey" FOREIGN KEY (c_id) REFERENCES courses(c_id)
    "enrolment_s_id_fkey" FOREIGN KEY (s_id) REFERENCES student(s_id)
```

Adding and dropping a Foreign Key constraint (cont.)

Dropping a foreign key constraint:

ALTER TABLE Enrolment

DROP CONSTRAINT enrolment_c_id_fkey;

```
utgupta=# ALTER TABLE Enrolment
utgupta-#     DROP CONSTRAINT enrolment_c_id_fkey;
ALTER TABLE
utgupta=# \d enrolment
            Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           | not null |
 dropped | boolean |           | not null |
Indexes:
    "enrolment_pkey" PRIMARY KEY, btree (s_id, c_id)
Foreign-key constraints:
    "enrolment_s_id_fkey" FOREIGN KEY (s_id) REFERENCES student(s_id)
```

Adding and dropping a Foreign Key constraint (cont.)

Adding a foreign key constraint:

ALTER TABLE Enrolment

ADD CONSTRAINT new_enrolment_course FOREIGN KEY (c_id)

REFERENCES Courses (c_id);

```
utgupta=# ALTER TABLE Enrolment
utgupta-#     ADD CONSTRAINT new_enrolment_course FOREIGN KEY (c_id)
utgupta-#         REFERENCES Courses (c_id);
ALTER TABLE
utgupta=# \d enrolment;
            Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           | not null |
 dropped | boolean |           | not null |
Indexes:
    "enrolment_pkey" PRIMARY KEY, btree (s_id, c_id)
Foreign-key constraints:
    "enrolment_s_id_fkey" FOREIGN KEY (s_id) REFERENCES student(s_id)
    "new_enrolment_course" FOREIGN KEY (c_id) REFERENCES courses(c_id)
```

Actions

- Actions defines the steps for the child table when a row from a parent table is deleted.
- In our case, Student is the parent table and Enrolment is the child table.

Following are the plausible actions:

1. CASCADE:
    1. If a parent row is deleted then the data of the child row will also be deleted.
    2. If a parent row is updated then the data of the child row will also be updated.
2. RESTRICT:
    1. A parent row will not get deleted if there is a row in the child table which references the row in the parent.
    2. A parent row will not get updated if there is a row in the child table which references the row in the parent.
3. SET NULL:
    1. The data of the child row will be set to NULL if the parent row is deleted.
    2. The data of the child row will be set to NULL if the parent row is updated.

Actions (cont.)

Drop the old foreign key constraint

```
ALTER TABLE Enrolment
    DROP CONSTRAINT new_enrolment_course;
ALTER TABLE Enrolment
    DROP CONSTRAINT enrolment_s_id_fkey;
```

```
utgupta=# ALTER TABLE Enrolment
utgupta-#     DROP CONSTRAINT new_enrolment_course;
ALTER TABLE
utgupta=# ALTER TABLE Enrolment
utgupta-#     DROP CONSTRAINT enrolment_s_id_fkey;
ALTER TABLE
utgupta=# \d enrolment;
            Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           | not null |
 dropped | boolean |           | not null |
Indexes:
    "enrolment_pkey" PRIMARY KEY, btree (s_id, c_id)
```

Actions (cont.)

Add the new foreign key constraints

```
ALTER TABLE Enrolment
   ADD CONSTRAINT new_enrolment_course FOREIGN KEY (c_id)
      REFERENCES Courses (c_id)
         ON UPDATE CASCADE;

ALTER TABLE Enrolment
   ADD CONSTRAINT new_enrolment_student FOREIGN KEY (s_id)
      REFERENCES Student (s_id)
         ON DELETE CASCADE;
```

Actions (cont.)

```
utgupta=# ALTER TABLE Enrolment
utgupta-#     ADD CONSTRAINT new_enrolment_course FOREIGN KEY (c_id)
utgupta-#         REFERENCES Courses (c_id)
utgupta-#             ON UPDATE CASCADE;
ALTER TABLE
utgupta=# ALTER TABLE Enrolment
utgupta-#     ADD CONSTRAINT new_enrolment_student FOREIGN KEY (s_id)
utgupta-#         REFERENCES Student (s_id)
utgupta-#             ON DELETE CASCADE;
ALTER TABLE
utgupta=# \d enrolment;
            Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           | not null |
 dropped | boolean |           | not null |
Indexes:
    "enrolment_pkey" PRIMARY KEY, btree (s_id, c_id)
Foreign-key constraints:
    "new_enrolment_course" FOREIGN KEY (c_id) REFERENCES courses(c_id) ON UPDATE CASCADE
    "new_enrolment_student" FOREIGN KEY (s_id) REFERENCES student(s_id) ON DELETE CASCADE
```

Actions (cont.)

- Let us **DELETE** a row from Student table where **s_id=5;**

```
utgupta=# select * from student;
 s_id | s_name |    ssn     |    dob     | gpa  | has_grad |   phone    |      email
------+--------+------------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234       | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567       | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468       | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412       | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    5 | ABY    | 4321       | 2000-01-01 | 3.32 | t        |            |
(5 rows)
```

```
utgupta=# select * from enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)
```

Actions (cont.)

- DELETE FROM Student WHERE s_id = 5;

```
utgupta=# select * from student;
 s_id | s_name |    ssn     |    dob     | gpa  | has_grad |   phone    |      email
------+--------+------------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234       | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567       | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468       | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412       | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    5 | ABY    | 4321       | 2000-01-01 | 3.32 | t        |            |
(5 rows)
```

```
utgupta=# select * from enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    5 |  160 | f
    5 |  215 | f
    1 |  181 | t
    2 |  160 | f
    5 |  180 | f
    2 |  280 | f
    5 |  181 | f
    2 |  181 | t
    3 |  180 | f
    5 |  280 | f
(11 rows)
```

Actions (cont.)

- DELETE FROM Student WHERE s_id = 5;

```
utgupta=# DELETE FROM Student WHERE s_id = 5;
DELETE 1
utgupta=# SELECT * FROM Student;
 s_id | s_name |    ssn    |    dob     | gpa  | has_grad |   phone    |      email
------+--------+-----------+------------+------+----------+------------+-----------------
    1 | DAVID  | 1234      | 1997-01-01 | 3.80 | f        | 0123456    | david@pqrs.com
    2 | JULIA  | 4567      | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468      | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412      | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
(4 rows)

utgupta=# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    1 |  181 | t
    2 |  160 | f
    2 |  280 | f
    2 |  181 | t
    3 |  180 | f
(6 rows)
```

Actions (cont.)

- Let us update Course c_id 180 to 130.

```
utgupta=# select * from courses;
 c_id |   c_name    | credits
------+-------------+---------
  180 | DB 1        |       5
  181 | DB 2        |       5
  215 | DB GRAD 1   |       5
  160 | CG          |       5
  280 | CV SEM      |       2
(5 rows)
```

```
utgupta=# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  180 | f
    1 |  181 | t
    2 |  160 | f
    2 |  280 | f
    2 |  181 | t
    3 |  180 | f
(6 rows)
```

Actions (cont.)

```
UPDATE Courses
    SET c_id = 130
        WHERE c_id = 180;
```

```
utgupta=# UPDATE Courses
utgupta-#     SET c_id = 130
utgupta-#         WHERE c_id = 180;
UPDATE 1
utgupta=# SELECT * FROM Courses;
 c_id |  c_name   | credits
------+-----------+---------
  181 | DB 2      |       5
  215 | DB GRAD 1 |       5
  160 | CG        |       5
  280 | CV SEM    |       2
  130 | DB 1      |       5
(5 rows)

utgupta=# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  181 | t
    2 |  160 | f
    2 |  280 | f
    2 |  181 | t
    1 |  130 | f
    3 |  130 | f
(6 rows)
```

Actions (cont.)

Drop the old foreign key constraint:

```
ALTER TABLE Enrolment
    DROP CONSTRAINT new_enrolment_course;
```

Drop the primary key constraint: (Primary key can't have NULL values)

```
ALTER TABLE Enrolment
    DROP CONSTRAINT enrolment_pkey;
```

DROP the NOT NULL constraint:

```
ALTER TABLE ENROLMENT
    ALTER COLUMN c_id DROP NOT NULL;
```

```
utgupta=# \d enrolment;
           Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           |          |
 dropped | boolean |           | not null |
Foreign-key constraints:
    "new_enrolment_course" FOREIGN KEY (c_id) REFERENCES courses(c_id) ON UPDATE SET NULL
    "new_enrolment_student" FOREIGN KEY (s_id) REFERENCES student(s_id) ON DELETE CASCADE
```

Actions (cont.)

Add new foreign key constraints:

ALTER TABLE Enrolment

ADD CONSTRAINT new_enrolment_course FOREIGN KEY (c_id)

REFERENCES Courses (c_id)

ON UPDATE SET NULL;

```
utgupta=# ALTER TABLE Enrolment
utgupta-#     ADD CONSTRAINT new_enrolment_course FOREIGN KEY (c_id)
utgupta-#         REFERENCES Courses (c_id)
utgupta-#             ON UPDATE SET NULL;
ALTER TABLE
utgupta=# \d enrolment;
            Table "university.enrolment"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 s_id    | integer |           | not null |
 c_id    | integer |           |          |
 dropped | boolean |           | not null |
Foreign-key constraints:
    "new_enrolment_course" FOREIGN KEY (c_id) REFERENCES courses(c_id) ON UPDATE SET NULL
    "new_enrolment_student" FOREIGN KEY (s_id) REFERENCES student(s_id) ON DELETE CASCADE
```

Actions (cont.)

- Our Courses and Enrolment tables are as follows:
- Let us update Courses c_id = 130 to c_id = 180.

```
utgupta=# SELECT * FROM Courses;
 c_id |   c_name    | credits
------+-------------+---------
  181 | DB 2        |       5
  215 | DB GRAD 1   |       5
  160 | CG          |       5
  280 | CV SEM      |       2
  130 | DB 1        |       5
(5 rows)
```

```
utgupta=# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  181 | t
    2 |  160 | f
    2 |  280 | f
    2 |  181 | t
    1 |  130 | f
    3 |  130 | f
(6 rows)
```

Actions (cont.)

```
UPDATE Courses
    SET c_id = 180
        WHERE c_id = 130;
```

```
utgupta=# UPDATE Courses
utgupta-#     SET c_id = 180
utgupta-#         WHERE c_id = 130;
UPDATE 1
utgupta=#
utgupta=# SELECT * FROM Courses;
 c_id |  c_name   | credits
------+-----------+---------
  181 | DB 2      |       5
  215 | DB GRAD 1 |       5
  160 | CG        |       5
  280 | CV SEM    |       2
  180 | DB 1      |       5
(5 rows)

utgupta=# SELECT * FROM Enrolment;
 s_id | c_id | dropped
------+------+---------
    1 |  181 | t
    2 |  160 | f
    2 |  280 | f
    2 |  181 | t
    1 |      | f
    3 |      | f
(6 rows)
```

Adding CHECK constraint

Adding a CHECK constraint on the GPA column:

The current description of the student table is as follows:

```
utgupta=# \d student;
                        Table "university.student"
  Column  |         Type          | Collation | Nullable |      Default
----------+-----------------------+-----------+----------+--------------------
 s_id     | integer               |           | not null |
 s_name   | character varying(30) |           |          |
 ssn      | character(9)          |           |          |
 dob      | date                  |           | not null | '2000-01-01'::date
 gpa      | numeric(3,2)          |           |          |
 has_grad | boolean               |           |          |
 phone    | character(10)         |           |          |
 email    | character varying(50) |           |          |
Indexes:
    "student_pkey" PRIMARY KEY, btree (s_id)
    "student_email_key" UNIQUE CONSTRAINT, btree (email)
    "student_phone_s_name_key" UNIQUE CONSTRAINT, btree (phone, s_name)
    "student_ssn_key" UNIQUE CONSTRAINT, btree (ssn)
Referenced by:
    TABLE "enrolment" CONSTRAINT "new_enrolment_student" FOREIGN KEY (s_id) REFERENCES student(s_id) ON DELETE CASCADE
```

Adding CHECK constraint (cont.)

Adding a CHECK constraint on the GPA column:

```
ALTER TABLE Student
    ADD CONSTRAINT gpa_check_constraint
        CHECK (gpa >= 0 AND gpa <= 4);
```

```
utgupta=# \d student;
                      Table "university.student"
  Column  |         Type          | Collation | Nullable |      Default
----------+-----------------------+-----------+----------+--------------------
 s_id     | integer               |           | not null |
 s_name   | character varying(30) |           |          |
 ssn      | character(9)          |           |          |
 dob      | date                  |           | not null | '2000-01-01'::date
 gpa      | numeric(3,2)          |           |          |
 has_grad | boolean               |           |          |
 phone    | character(10)         |           |          |
 email    | character varying(50) |           |          |
Indexes:
    "student_pkey" PRIMARY KEY, btree (s_id)
    "student_email_key" UNIQUE CONSTRAINT, btree (email)
    "student_phone_s_name_key" UNIQUE CONSTRAINT, btree (phone, s_name)
    "student_ssn_key" UNIQUE CONSTRAINT, btree (ssn)
Check constraints:
    "gpa_check_constraint" CHECK (gpa >= 0::numeric AND gpa <= 4::numeric)
Referenced by:
    TABLE "enrolment" CONSTRAINT "new_enrolment_student" FOREIGN KEY (s_id) REFERENCES student(s_id) ON DELETE CASCADE
```

Unit testing

Let us update the GPA of Student with s_id = 1 to 2.0.

(This will meet the CHECK constraint)

```
UPDATE Student
   SET gpa = 2
      WHERE s_id = 1;
```

```
utgupta=# UPDATE Student
utgupta-#     SET gpa = 2
utgupta-#         WHERE s_id = 1;
UPDATE 1
utgupta=# SELECT * FROM Student;
 s_id | s_name |    ssn     |    dob     | gpa  | has_grad |   phone    |      email
------+--------+------------+------------+------+----------+------------+-----------------
    2 | JULIA  | 4567       | 2000-02-18 | 3.90 | f        | 1234589    | julia@pqrs.com
    3 | DAVID  | 2468       | 2000-02-01 | 3.80 | f        | 9827123    | david1@pqrs.com
    4 | JOEL   | 3412       | 2002-03-23 | 3.34 | f        |            | joel@pqrs.com
    1 | DAVID  | 1234       | 1997-01-01 | 2.00 | f        | 0123456    | david@pqrs.com
(4 rows)
```

Unit testing (cont.)

Let us update the GPA of Student with s_id = 1 to -2.0.

(This will violate the CHECK constraint)

```
UPDATE Student
    SET gpa = -2
        WHERE s_id = 1;
```

```
utgupta=# UPDATE Student
utgupta-#     SET gpa = -2
utgupta-#         WHERE s_id = 1;
ERROR:  new row for relation "student" violates check constraint "gpa_check_constraint"
DETAIL:  Failing row contains (1, DAVID, 1234     , 1997-01-01, -2.00, f, 0123456   , david@pqrs.com).
utgupta=#
```

Create commands (Students)

```
CREATE TABLE STUDENT (
        S_ID INT,
        S_NAME VARCHAR(30),
        SSN CHAR(9),
        DOB DATE DEFAULT '2000-01-01' NOT NULL,
        GPA NUMERIC(3,2),
        HAS_GRAD BOOL,
        PHONE CHAR(10),
        EMAIL VARCHAR(50),
        PRIMARY KEY (S_ID),
        UNIQUE (EMAIL),
        UNIQUE (PHONE, S_NAME),
        UNIQUE (SSN)
);
```

```
CREATE TABLE COURSES (
        C_ID INT PRIMARY KEY,
        C_NAME CHAR(9) UNIQUE,
        CREDITS INT DEFAULT 5 NOT NULL
);
```

```
CREATE TABLE ENROLMENT (
        S_ID INT REFERENCES STUDENT,
        C_ID INT REFERENCES COURSES,
        DROPPED BOOL NOT NULL,
        PRIMARY KEY(S_ID, C_ID)
);
```

Insert Commands (Students)

Student

```
INSERT INTO STUDENT VALUES (1, 'DAVID', '1234', '1997-01-01', 3.8, false, '0123456', 'david@pqrs.com');
INSERT INTO STUDENT VALUES (2, 'JULIA', '4567', '2000-02-18', 3.9, false, '1234589', 'julia@pqrs.com');
INSERT INTO STUDENT VALUES (3, 'DAVID', '2468', '2000-02-01', 3.8, false, '9827123', 'david1@pqrs.com');
INSERT INTO STUDENT VALUES (4, 'JOEL', '3412', '2002-03-23', 3.34, false, NULL, 'joel@pqrs.com');
INSERT INTO STUDENT (S_ID, S_NAME, SSN, GPA, HAS_GRAD, PHONE, EMAIL) VALUES (5, 'ABY', '4321', 3.32, true, NULL, NULL);
```

Courses

```
INSERT INTO COURSES VALUES (180, 'DB 1');
INSERT INTO COURSES VALUES (181, 'DB 2');
INSERT INTO COURSES VALUES (215, 'DB GRAD 1');
INSERT INTO COURSES VALUES (160, 'CG');
INSERT INTO COURSES VALUES (280, 'CV SEM', 2);
```

Enrolment

```
INSERT INTO ENROLMENT VALUES (1, 180, false);
INSERT INTO ENROLMENT VALUES (5, 160, false);
INSERT INTO ENROLMENT VALUES (5, 215, false);
INSERT INTO ENROLMENT VALUES (1, 181, true);
INSERT INTO ENROLMENT VALUES (2, 160, false);
INSERT INTO ENROLMENT VALUES (5, 180, false);
INSERT INTO ENROLMENT VALUES (2, 280, false);
INSERT INTO ENROLMENT VALUES (5, 181, false);
INSERT INTO ENROLMENT VALUES (2, 181, true);
INSERT INTO ENROLMENT VALUES (3, 180, false);
INSERT INTO ENROLMENT VALUES (5, 280, false);
```