Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Bases de Datos Avanzadas

Informe etapa 2

Fabián Olivares, Erik Regla
{folivares13,eregla09}@alumnos.utalca.cl

19 de Abril del 2016

1. Modelo final de la base de datos

En base a los resultados obtenidos en la etapa anterior se optó por mantener las siguientes desnormalizaciones:

1. *Relación 1-1:* Separación de columna `plain_text_content` de `problems`. Esta resultó en una reducción del tamaño de la tabla de problemas como también en una mejora de la velocidad de consulta.
2. *Relación 1-N clave:* Columna `country_id` añadida a tabla `teams` dado que es muy frecuente consultar el país atribuido a un equipo.
3. *Relación N-M:* Nueva columna `team_name` a tabla `team_members` ya que es frecuente consultar los equipos en los que ha participado un competidor.
4. *Tabla de búsqueda:* Se separa al malescrito `ballon_colour` de la tabla `problems` ya que son 20 colores que se utilizan. Esta resultó en una reducción del tamaño de la tabla de problemas.

La *Relación 1-N no clave:* Nueva columna `contest_name` en tabla `problems`, no fue exitosa dado que si bien consultar a que competencia pertenece un problema dado es bastante frecuente, la cantidad de tuplas en la tabla hizo que agregar una nueva columna de texto fuese muy costoso en espacio.

En un experimento adicional de la primera etapa se comprobó que no es costoso indexar las tablas más consultadas de la base de datos (problemas, sitios, equipos y miembros del equipo). Una de las opciones barajadas además era la posibilidad de dejar `GIST` como índice para la tabla de descripción de problemas, sin embargo, dada la envergadura del texto el proceso de indexado tomaba mucho tiempo como para ser considerado una opción viable.

```
1  CREATE TABLE teams_2 AS SELECT teams.id,teams.institution,teams.coach_id,teams.name,teams.site_id,teams.approved,teams.
       include_coach_cert,teams.make_coach_individual_cert,sites.country_id AS country_id FROM teams,sites WHERE teams.site_id =
       sites.id;
2  CREATE TABLE team_members_2 AS SELECT team_members.id,team_members.contestant_id,team_members.team_id,team_members.role_id,
       team_members.registration_complete,team_members.on_team_certificate,team_members.on_individual_certificate,teams.name AS
       team_name FROM teams,team_members WHERE team_members.team_id = teams.id;
3  CREATE TABLE colours AS SELECT row_number()over() AS id,problems.ballon_colour AS name FROM problems GROUP BY ballon_colour;
4  CREATE TABLE problems_2 AS SELECT problems.id,problems.letter,problems.pdf_file,colours.id AS colour_id,problems.description,
       problems.codename FROM problems,colours WHERE problems.ballon_colour = colours.name;
5  CREATE TABLE problems_content AS SELECT id,plain_text_content FROM problems;
6  DROP TABLE teams;
7  DROP TABLE team_members;
8  DROP TABLE problems;
9  ALTER TABLE teams_2 RENAME TO teams;
10 ALTER TABLE team_members_2 RENAME TO team_members;
11 ALTER TABLE problems_2 RENAME TO problems;
```

Figura 1: Consulta SQL para desnormalizar base de datos original.

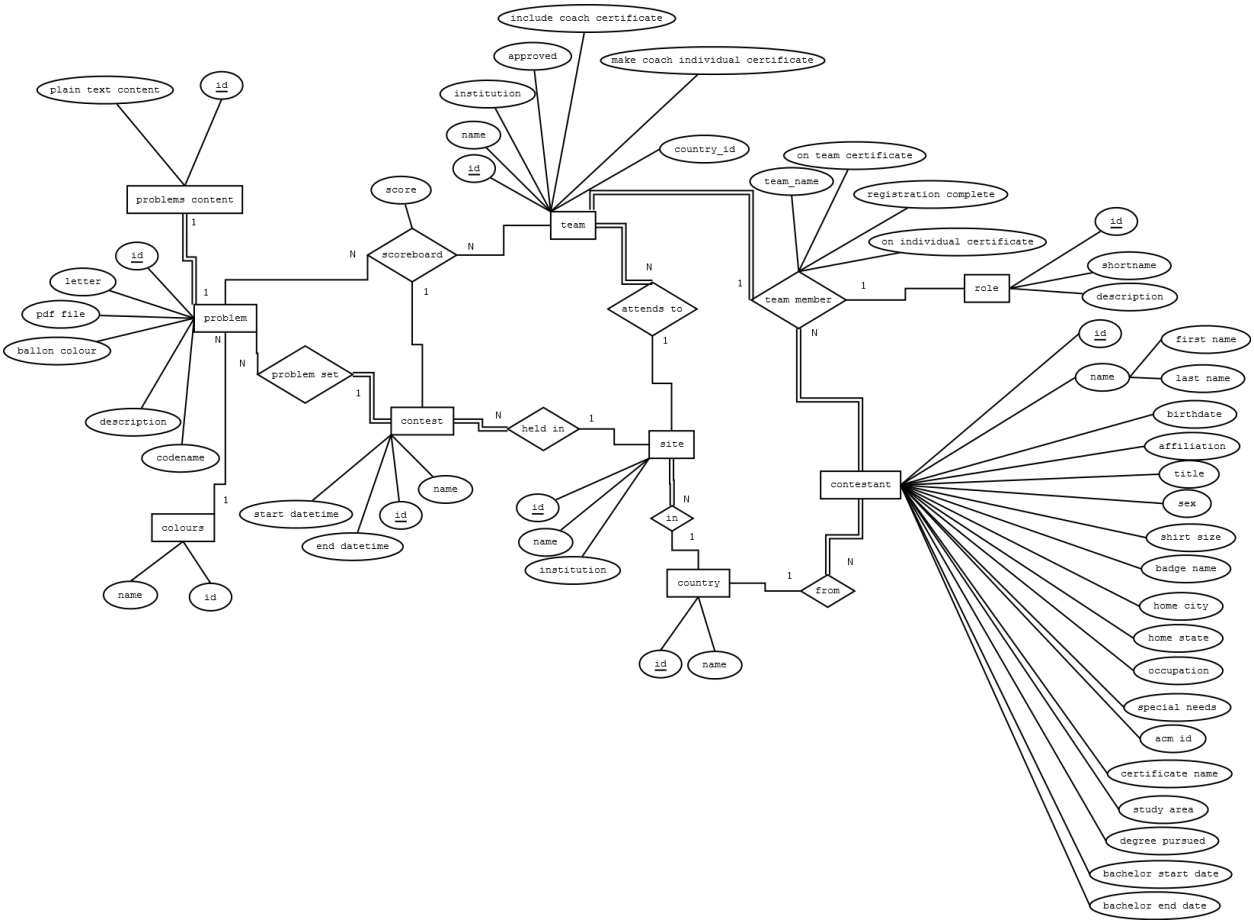

Figura 2: Diagrama ER final.

```sql
-- PROBLEM INDICES
CREATE INDEX problem_id_idx ON problems USING hash (id);
CREATE INDEX problem_letter_idx ON problems USING hash (letter);
CREATE INDEX problem_pdf_file_idx ON problems USING hash (pdf_file);
CREATE INDEX problem_colour_id_idx ON problems USING hash (colour_id);
CREATE INDEX problem_description_id_idx ON problems USING hash (description);
CREATE INDEX problem_codename_idx ON problems USING hash (codename);

-- Problem content :p
CREATE INDEX problems_content_id_idx ON problems_content USING hash (id);
--CREATE INDEX problems_content_plain_text_content_idx ON problems_content USING gist (plain_text_content);

-- Sites
CREATE INDEX sites_institution_idx ON sites USING hash (institution);

-- Team members
CREATE INDEX team_members_id_idx ON team_members USING hash (id);
CREATE INDEX team_members_contestant_id_idx ON team_members USING hash (contestant_id);
CREATE INDEX team_members_team_id_idx ON team_members USING hash (team_id);
CREATE INDEX team_members_role_id_idx ON team_members USING hash (role_id);
CREATE INDEX team_members_registration_complete_idx ON team_members USING hash (registration_complete);
CREATE INDEX team_members_on_team_certificate_idx ON team_members USING hash (on_team_certificate);
CREATE INDEX team_members_on_individual_certificate_idx ON team_members USING hash (on_individual_certificate);
CREATE INDEX team_members_team_name_idx ON team_members USING hash (team_name);

-- TEAMS!
CREATE INDEX team_id_idx ON teams USING hash (id);
CREATE INDEX team_institution_idx ON teams USING hash (institution);
CREATE INDEX team_coach_id_idx ON teams USING hash (coach_id);
CREATE INDEX team_name_idx ON teams USING hash (name);
CREATE INDEX team_site_id_idx ON teams USING hash (site_id);
CREATE INDEX team_approved_idx ON teams USING hash (approved);
CREATE INDEX team_include_coach_cert_idx ON teams USING hash (include_coach_cert);
CREATE INDEX team_make_coach_individual_cert_idx ON teams USING hash (make_coach_individual_cert);
CREATE INDEX team_country_id_idx ON teams USING hash (country_id);
```

Figura 3: Consulta SQL para agregar indices a tablas

2. Restricciones de dominio

Muchas de las restricciones fueron ya implementadas durante la etapa uno[1], por tanto acá solo se mencionarán las mas importantes y nuevas implementadas.

1. *Restricción de dominio COLOURS:NAME*: Los colores de los globos están restringidos a ser una cadena de texto sin caracteres especiales. Sin embargo si están permitidos los espacios dado que existen colores como "VERMILION RED" o "PALE GREEN", los cuales están compuestos además de un espacio en blanco. Para esto se creó una restricción de dominio sobre un tipo nuevo de dato denominado `baloon_type` (Ver Figura 2).

2. *Restricción de dominio DEFAULT*: Debido a la implementación de la etapa 1, muchas de las tablas ya contaban con secuencias para establecer los valores por defecto de sus llaves primarias. Solo fueron agregadas a las nuevas tablas generadas producto de la desnormalización (ver Figuras 2 y 2).

```
1  CREATE SEQUENCE colours_id_seq
2    INCREMENT 1
3    MINVALUE 1
4    MAXVALUE 9223372036854775807
5    START 0
6    CACHE 1;
7  ALTER TABLE colours_id_seq OWNER TO postgres;
8  ALTER TABLE colours ALTER COLUMN id SET DEFAULT nextval('colours_id_seq');
9  CREATE DOMAIN baloon_type AS text CONSTRAINT baloon_type_check CHECK (VALUE ~ '([A-Za-z]+(_)*)+');
10 ALTER TABLE colours ALTER COLUMN name SET DATA TYPE baloon_type;
```

Figura 4: Consulta SQL para agregar secuencia a colours junto con su restricción de dominio.

```
1  CREATE SEQUENCE problems_content_id_seq
2    INCREMENT 1
3    MINVALUE 1
4    MAXVALUE 9223372036854775807
5    START 880000
6    CACHE 1;
7  ALTER TABLE problems_content_id_seq OWNER TO postgres;
8  ALTER TABLE problems_content ALTER COLUMN id SET DEFAULT nextval('problems_content_id_seq');
```

Figura 5: Consulta SQL para agregar secuencia a tabla problems_content.

3. Llaves primarias

[1] Ver `https://github.com/eregla/3407B414_A-databases_management/tree/master/StageOne`

```sql
ALTER TABLE colours ADD CONSTRAINT colours_pk PRIMARY KEY (id);
ALTER TABLE contest_sites ADD CONSTRAINT contest_sites_pk PRIMARY KEY (id);
ALTER TABLE contestants ADD CONSTRAINT contestants_pk PRIMARY KEY (id);
ALTER TABLE contests ADD CONSTRAINT contests_pk PRIMARY KEY (id);
ALTER TABLE countries ADD CONSTRAINT countries_pk PRIMARY KEY (id);
ALTER TABLE problem_set ADD CONSTRAINT problem_set_pk PRIMARY KEY (id);
ALTER TABLE problems ADD CONSTRAINT problems_pk PRIMARY KEY (id);
ALTER TABLE problems_content ADD CONSTRAINT problems_content_pk PRIMARY KEY (id);
ALTER TABLE roles ADD CONSTRAINT roles_pk PRIMARY KEY (id);
ALTER TABLE scoreboards ADD CONSTRAINT scoreboards_pk PRIMARY KEY (id);
ALTER TABLE sites ADD CONSTRAINT sites_pk PRIMARY KEY (id);
ALTER TABLE team_members ADD CONSTRAINT team_members_pk PRIMARY KEY (id);
ALTER TABLE teams ADD CONSTRAINT teams_pk PRIMARY KEY (id);
```

Figura 6: Consulta SQL para agregar llaves primarias

4. Llaves foráneas y restricciones de integridad

```sql
ALTER TABLE contest_sites ADD FOREIGN KEY (id_site) REFERENCES sites ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE contest_sites ADD FOREIGN KEY (id_contest) REFERENCES contests ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE contestants ADD FOREIGN KEY (home_country) REFERENCES countries ON DELETE SET NULL ON UPDATE CASCADE;
ALTER TABLE problem_set ADD FOREIGN KEY (contest_id) REFERENCES contests ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE problem_set ADD FOREIGN KEY (problem_id) REFERENCES problem ON DELETE RESTRICT ON UPDATE CASCADE;
ALTER TABLE problems ADD FOREIGN KEY (colour_id) REFERENCES colour ON DELETE SET NULL ON UPDATE CASCADE;
ALTER TABLE scoreboards ADD FOREIGN KEY (problem_id) REFERENCES problems ON DELETE RESTRICT ON UPDATE CASCADE;
ALTER TABLE scoreboards ADD FOREIGN KEY (contest_id) REFERENCES contests ON DELETE RESTRICT ON UPDATE CASCADE;
ALTER TABLE scoreboards ADD FOREIGN KEY (team_id) REFERENCES teams ON DELETE RESTRICT ON UPDATE CASCADE;
ALTER TABLE sites ADD FOREIGN KEY (country_id) REFERENCES countries ON DELETE SET NULL ON UPDATE CASCADE;
ALTER TABLE team_members ADD FOREIGN KEY (contestant_id) REFERENCES contestants ON DELETE RESTRICT ON UPDATE RESTRICT;
ALTER TABLE team_members ADD FOREIGN KEY (role_id) REFERENCES roles ON DELETE SET NULL ON UPDATE CASCADE;
ALTER TABLE team_members ADD FOREIGN KEY (team_id) REFERENCES teams ON DELETE RESTRICT ON UPDATE CASCADE;
ALTER TABLE teams ADD FOREIGN KEY (coach_id) REFERENCES contestants ON DELETE SET NULL ON UPDATE CASCADE;
ALTER TABLE teams ADD FOREIGN KEY (site_id) REFERENCES sites ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE teams ADD FOREIGN KEY (country_id) REFERENCES countries ON DELETE SET NULL ON UPDATE CASCADE;
```

Figura 7: Consulta SQL para agregar llaves foráneas

5. Reglas de negocio

Las reglas de negocio que hemos diseñado se pueden clasificar en tres tipos:

1. Consistencia de problemas y sets de problemas
   a) *No se puede repetir la letra de un problema en un mismo problem set*
   b) *No se puede repetir el color del globo en un mismo problem set*
   c) *Un mismo problema no se puede asignar más de una vez al mismo contest*
2. Borrado de problemas y actualización de los marcadores
   a) *Al borrar un problema, se borran las entradas asociadas a éste en el scoreboard*
3. Consistencia de equipos y miembros
   a) *Un team debe estar compuesto por exactamente tres contestants*
   b) *Un contestant asociado a un team debe pertenecer a la misma institución que dicho team*

```
CREATE OR REPLACE FUNCTION baloon_and_letter() RETURNS trigger AS $baloon_and_letter$
        DECLARE
            repeated_letters int;
            the_letter text;
            repeated_problems int;
            the_balloon int;
            repeated_balloons int;
        BEGIN
            SELECT letter into the_letter FROM problems WHERE NEW.problem_id = problems.id;
            SELECT colour_id into the_balloon FROM problems WHERE NEW.problem_id = problems.id;
            SELECT COUNT (*) into repeated_letters FROM problems, problem_set WHERE the_letter = problems.letter AND problems.id =
                   problem_set.problem_id AND problem_set.contest_id = NEW.contest_id;
            SELECT COUNT (*) into repeated_balloons FROM problems, problem_set WHERE the_balloon = problems.colour_id AND problems
                  .id = problem_set.problem_id AND problem_set.contest_id = NEW.contest_id;
            SELECT COUNT (*) into repeated_problems FROM problem_set WHERE contest_id = NEW.contest_id;
        IF repeated_letters != 0 THEN
            RAISE EXCEPTION 'Letter␣%␣is␣already␣present␣on␣the␣problem␣set␣for␣the␣contest␣%', the_letter, NEW.contest_id;
        END IF;
        IF repeated_balloons != 0 THEN
            RAISE EXCEPTION 'Balloon␣%␣is␣already␣present␣on␣the␣problem␣set␣for␣the␣contest␣%', (SELECT colours.name FROM
                  problems, colours WHERE colours.id = problems.id AND NEW.problem_id = problems.id) , NEW.contest_id;
        END IF;
        IF repeated_problems != 0 THEN
            RAISE EXCEPTION 'Problem␣%␣is␣already␣present␣on␣the␣problem␣set␣for␣the␣contest␣%', NEW.problem_id, NEW.contest_id;
        END IF;
        RETURN NEW;
    END;
$baloon_and_letter$ LANGUAGE plpgsql;
DROP TRIGGER baloon_and_letter ON problem_set;
CREATE TRIGGER baloon_and_letter BEFORE INSERT OR UPDATE ON problem_set
    FOR EACH ROW EXECUTE PROCEDURE baloon_and_letter();
```

Figura 8: Definición del trigger para verificación de consistencia para problemas y sets de problemas.

```
CREATE OR REPLACE FUNCTION delete_problem_scoreboard() RETURNS TRIGGER AS $$
    BEGIN
        DELETE FROM scoreboards WHERE scoreboards.problem_id = OLD.id;
        RETURN OLD;
    END;
$$ LANGUAGE plpgsql;
DROP TRIGGER t_delete_problem_scoreboard ON problems;
CREATE TRIGGER t_delete_problem_scoreboard
    BEFORE DELETE ON problems
    FOR EACH ROW EXECUTE PROCEDURE delete_problem_scoreboard();
```

Figura 9: Definición del trigger para borrado de problemas y actualización de marcadores.

```
CREATE OR REPLACE FUNCTION check_members_from_team() RETURNS TRIGGER AS $$
        DECLARE
            the_institution text;
            the_affiliation text;
    BEGIN
        SELECT institution into the_affiliation FROM teams WHERE NEW.team_id = teams.id;
        SELECT affiliation into the_institution FROM contestants WHERE NEW.contestant_id = contestants.id;

        IF (the_institution != the_affiliation) THEN
            RAISE EXCEPTION 'The␣contestant␣does␣not␣share␣the␣same␣institution␣as␣the␣team';
        END IF;

        IF((SELECT COUNT(*) FROM team_members WHERE team_id = NEW.team_id) < 3.0 )
        THEN
            RETURN NEW;
        END IF;
        RAISE EXCEPTION 'A␣team␣cannot␣have␣more␣than␣3␣members';
    END;
$$ LANGUAGE plpgsql;
DROP TRIGGER t_max_members_per_team ON team_members;
CREATE TRIGGER t_max_members_per_team
    BEFORE INSERT ON team_members
    FOR EACH ROW EXECUTE PROCEDURE check_members_from_team();
```

Figura 10: Definición del trigger para verificación de consistencia para equipos y miembros.

6. Definicion de transaciones aplicables

1. Cuando se efectúa un cambio en un problema, por reglamento, todos los puntajes asociados a ese problema deben ser eliminados. De no ser así, el problema no puede cambiar de estado. En caso de ser interrumpido el proceso requiere de savepoints para poder reanudar o bien abortar la operación al vuelo.

2. Para crear un set de problemas, es necesario incluirlos todos y este problem set debe estar ligado a una competencia. No puede existir un problem set sin competencia ni un problema que no pertenezca a un set. De fallar alguno de estos pasos, es necesario revertir los cambios para evitar lanzar una competencia con datos inválidos. En este caso, si bien está utilizando un nivel de aislamiento `READ UNCOMMITED` para evitar problemas con otras transacciones, es necesario realizar todas las comprobaciones de llaves de inmediato (`ALL IMMEDIATE`) para evitar conflictos con otras transacciones concurrentes.

3. Para crear correctamente un equipo, se debe crear la instancia y además conectar a los participantes. En caso de fallar uno, todo debe ser deshecho.

```
import psycopg2
import random
import string
from sys import stdout
from time import sleep

params = {
    'database': 'icpcdb',
    'user': 'postgres',
    'password': '1234',
    'host': '10.211.55.4',
    'port': 5432
}

conn = psycopg2.connect(**params)
cur = conn.cursor()
teams_id = -1;
con1_id = -1;
con2_id = -1;
con3_id = -1;
# create team
cur.execute("INSERT INTO teams VALUES (DEFAULT,'University of Westeros',1,'Not Candy',1,'t','t','t',1);")
cur.execute("SELECT currval('teams_id_seq');");
if(cur.rowcount > 0):
    teams_id = cur.fetchone()[0] ##store the assigned id

# insert first contestant
cur.execute("INSERT INTO contestants VALUES (DEFAULT,'Juan','Nieves','1990-05-12','University of Westeros',NULL,1,'M',NULL,1,'
     Winterfell','WESTEROS',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);")
cur.execute("SELECT currval('contestants_id_seq');")
if(cur.rowcount > 0):
    con1_id = cur.fetchone()[0]

# insert second contestant
cur.execute("INSERT INTO contestants VALUES (DEFAULT,'Tyrion','Lannister','1985-07-09','University of Westeros',NULL,1,'M',NULL
     ,1,'Casterly Rock','WESTEROS',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);")
cur.execute("SELECT currval('contestants_id_seq');")
if(cur.rowcount > 0):
    con2_id = cur.fetchone()[0]

# insert third contestant
cur.execute("INSERT INTO contestants VALUES (DEFAULT,'Daenerys Queen of Meereen, Queen of the Andals, the Rhoynar and the First
     Men, Lady Regnant of the Seven Kingdoms, Khaleesi of the Great Grass Sea, Mhysa, Breaker of Chains, the Unburnt, Mother of
     Dragons','Targaryen','1991-12-18','University of Westeros',NULL,2,'M',NULL,1,'Meereen','Essos',NULL,NULL,NULL,NULL,NULL,NULL
     ,NULL,NULL);")
cur.execute("SELECT currval('contestants_id_seq');")
if(cur.rowcount > 0):
    con3_id = cur.fetchone()[0]

try:
    #assign team members
    cur.execute("INSERT INTO team_members VALUES(DEFAULT, %s, %s, 1, 't', 't', 't','University of Westeros');", (con1_id,teams_id)
        )
    cur.execute("INSERT INTO team_members VALUES(DEFAULT, %s, %s, 1, 't', 't', 't','University of Westeros');", (con2_id,teams_id)
        )
    cur.execute("INSERT INTO team_members VALUES(DEFAULT, %s, %s, 1, 't', 't', 't','University of Westeros');", (con3_id,teams_id)
        )

except Exception, e:
    cur.rollback()
    print "An error has ocurred, rollin'back!"
    print e
conn.commit()
cur.close()
conn.close()
print "Insertion of test data finished."
```

Figura 11: Programa python que implementa la transacción 3.

```
1
2  import psycopg2
3  import random
4  import string
5  from sys import stdout
6  from time import sleep
7
8  params = {
9      'database': 'icpcdb',
10     'user': 'postgres',
11     'password': '1234',
12     'host': '10.211.55.4',
13     'port': 5432
14 }
15
16 conn = psycopg2.connect(**params)
17 cur = conn.cursor()
18 teams_id = -1;
19 con1_id = -1;
20 con2_id = -1;
21 con3_id = -1;
22 # create team
23 conn.set_isolation_level(psycopg2.extensions.ISOLATION_LEVEL_READ_UNCOMMITTED)
24 pid = -1;
25 cur.execute("SET␣CONSTRAINTS␣ALL␣IMMEDIATE;")
26 cur.execute("INSERT␣INTO␣problems␣VALUES(DEFAULT,␣'A',␣'A.pdf',␣1,␣'Elmano␣parece␣candy␣pero␣no␣es␣candy␣asi␣que␣candy␣candy␣
       aprende␣algo␣candy','notCandy');")
27 cur.execute("SELECT␣currval('problems_id_seq');")
28 if(cur.rowcount > 0):
29     pid = cur.fetchone()[0] ##store the assigned id
30
31 cur.execute("INSERT␣INTO␣problems_content␣VALUES(%s,'aennginif␣si␣␣isngua␣ugn␣augouneaiurgn␣uonrg␣arug␣ar␣auiguoireaig␣ueagurarug␣
       eauib␣uah␣baurgb␣auireg␣be␣uibg␣bvananinaviunriu␣aiunr␣auiagui␣lljio');", pid)
32 cur.execute("INSERT␣INTO␣problem_set␣VALUES(DEFAULT,1,%s);", pid)
33 cur.execute("SAVEPOINT␣sp1;")
34
35 #if insertion fails, then rollsback to the last updated problem, that way we don't need to insert it again
36 try:
37     cur.execute("INSERT␣INTO␣problems␣VALUES(DEFAULT,␣'B',␣'B.pdf',␣2,␣'Aca␣habría␣una␣descripción...␣Si␣se␣me␣ocurriera␣una!','
           notDesc');")
38     cur.execute("SELECT␣currval('problems_id_seq');")
39     cur.execute("INSERT␣INTO␣problems_content␣VALUES(%s,'ef␣pjpg␣reijg␣pirgisgji␣sgi');", pid)
40     cur.execute("INSERT␣INTO␣problem_set␣VALUES(DEFAULT,1,%s);", pid)
41 except Exception, e:
42     cur.execute("ROLLBACK␣TO␣sp1;")
43     print "An␣error␣has␣ocurred,␣rollin'back␣to␣sp1!"
44     print e
45 cur.execute("SAVEPOINT␣sp2;")
46 try:
47     cur.execute("INSERT␣INTO␣problems␣VALUES(DEFAULT,␣'C',␣'C.pdf',␣3,␣'soa␣bashele,␣haga␣algo!␣T-T','soaBashele');")
48     cur.execute("SELECT␣currval('problems_id_seq');")
49     cur.execute("INSERT␣INTO␣problems_content␣VALUES(%s,'lala␣lalala␣lala␣lalala');", pid)
50     cur.execute("INSERT␣INTO␣problem_set␣VALUES(DEFAULT,1,%s);", pid)
51 except Exception, e:
52     cur.execute("ROLLBACK␣TO␣sp2;")
53     print "An␣error␣has␣ocurred,␣rollin'back␣to␣sp2!"
54     print e
55
56 conn.commit()
57 cur.close()
58 conn.close()
59 print "Insertion␣of␣test␣data␣finished."
```

Figura 12: Programa python que implementa la transacción 2.