Comenzado el	sábado, 7 de junio de 2025, 15:27
Estado	Finalizado
Finalizado en	sábado, 7 de junio de 2025, 16:17
Tiempo empleado	49 minutos 25 segundos

Pregunta 1

Finalizado

Se puntúa como 0 sobre 1,00

Utilizando el esquema unc_esq_voluntarios. Mostrar los tres voluntarios (su PK) que más horas aportadas tienen y posean coordinador.

Tiempo estimado 20 min

a. No hay registros que cumplan con este requerimiento

b. 100, 101, 102

c. Ninguna de las opciones es correcta

d. 10, 101, 125

e. 102, 216, 101

Respuesta incorrecta.

SQL que lo resuelve:

```
SELECT v.nro_voluntario, v.horas_aportadas
FROM unc_esq_voluntario.voluntario v
WHERE v.id_coordinador is not null
ORDER BY v.horas_aportadas DESC
LIMIT 3;
```

(101,102,145)

La respuesta correcta es:

Ninguna de las opciones es correcta

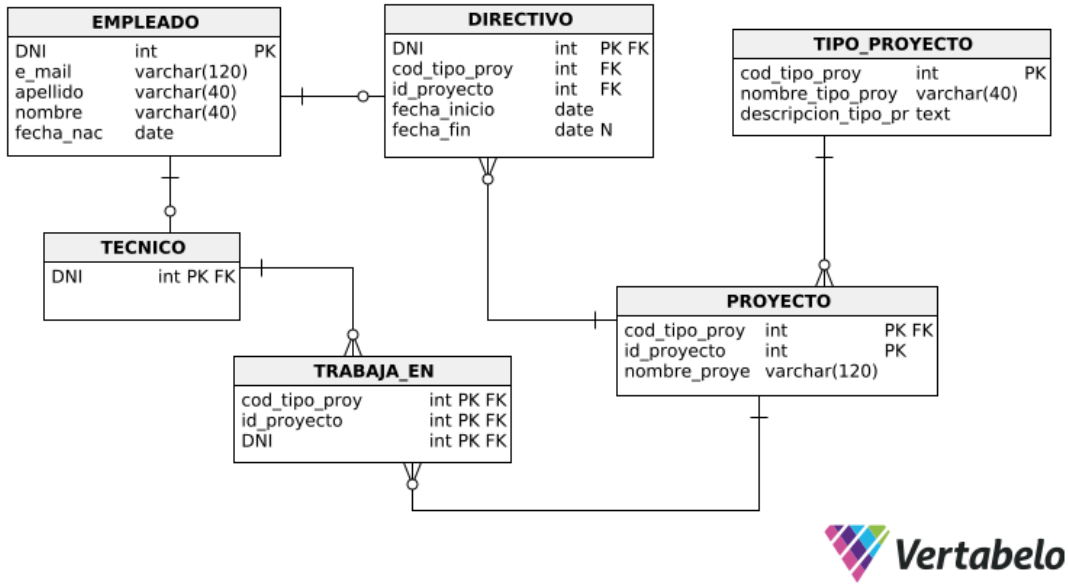
Pregunta 2

Finalizado

Se puntúa como 0 sobre 1,00

Tiempo estimado 45 min

Dado el siguente esquema - Script de Creación de Tablas



1) De la sentencia declarativa mas restrictiva que controle lo siguiente

- Todo Proyecto debe tener un Directivo, si tiene personal trabajando en él

2) En caso de que no pueda ser implementado en PostgreSQL declarativamente, de la solución procedural mas eficiente.

```
/* Pregunta 2
1) De la sentencia declarativa mas restrictiva que controle lo siguiente
- To do Proyecto debe tener un Directivo, si tiene personal trabajando en él */
/*
CREATE ASSERTION proyecto_con_directivo_y_personal
CHECK (NOT EXISTS (
    SELECT 1
    FROM TRABAJA_EN te
    WHERE NOT EXISTS (
       SELECT 1
       FROM DIRECTIVO d
       WHERE te.cod_tipo_proyecto = d.cod_tipo_proyecto
       AND te.id_proy = d.id_proy
    )
)); */

-- 2) En caso de que no pueda ser implementado en PostgreSQL declarativamente, de la soluci
-- mas eficiente.
-- TRABAJA_EN
CREATE OR REPLACE FUNCTION fn_trabaja_en()
RETURNS TRIGGER AS $$
BEGIN
    IF NOT EXISTS (
        SELECT 1
        FROM DIRECTIVO d
        WHERE NEW.cod_tipo_proy = d.cod_tipo_proy
        AND NEW.id_proyecto = d.id_proyecto
    )
    THEN RAISE EXCEPTION 'ERROR! El Proyecto id %, código tipo %, no tiene Directivo a carg
    END IF;
```

1)

```
CREATE ASSERTION ck_proyecto_con_personal_tiene_directivo
       CHECK ( NOT EXISTS( SELECT 1
                   FROM trabaja_en t
                   WHERE NOT EXISTS (SELECT 1
                           FROM directivo d
                           WHERE d.cod_tipo_proy = t.cod_tipo_proy
                           AND d.id_proyecto = t.id_proyecto)));
```

```
--Al ser atributos no nulos, se puede plantear como un NOT IN también
CREATE ASSERTION ck_proyecto_con_personal_tiene_directivo
       CHECK ( NOT EXISTS( SELECT 1
                   FROM trabaja_en
                   WHERE (cod_tipo_proy, id_proyecto)
                           NOT IN (SELECT cod_tipo_proy, id_proyecto
                                   FROM directivo)));
```

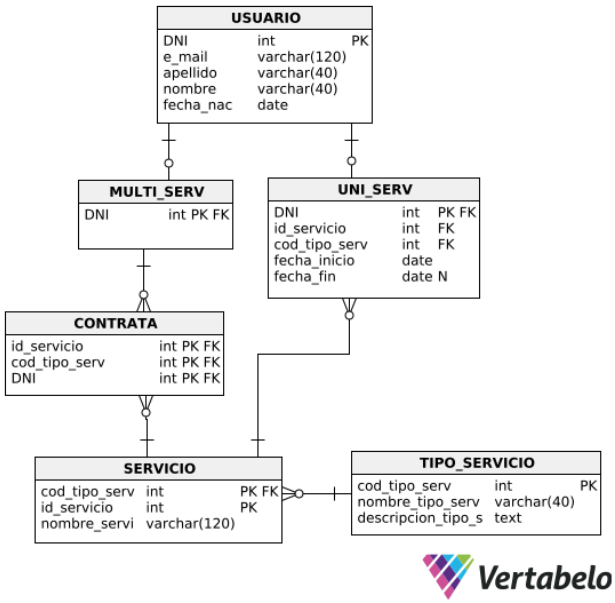
2)

```
create function fn_directivo() returns trigger as $$
begin
    if (not exists (select 1
                from directivo d
                where (d.cod_tipo_proy, d.id_proyecto) = (new.cod_tipo_proy, new.id_proyecto))) then
        raise exception 'Error';
    end if;
    return new;
end; $$ language plpgsql;

create trigger tr_biu_trabaja_en
    before insert or update of cod_tipo_proy, id_proyecto on trabaja_en
    for each row
    execute function fn_directivo();

create function fn_trabaja_en() returns trigger as $$
declare cod_proy int; id_proy int;
begin
    if (tg_op = 'DELETE') then
        cod_proy := old.cod_tipo_proy;
        id_proy := old.id_proyecto;
    else
        cod_proy := new.cod_tipo_proy;
        id_proy := new.id_proyecto;
    end if;
    if (exists (
            select 1
            from trabaja_en t
            where (t.cod_tipo_proy, t.id_proyecto) = (cod_proy, id_proy))) then
        raise exception 'Error';
    end if;
    if (tg_op = 'DELETE') then
        return old;
    end if;
    return new;
end; $$ language plpgsql;

create trigger tr_biu_unisum
    before delete or update of cod_tipo_proy, id_proyecto on directivo
    for each row
    execute function fn_trabaja_en();
```

Pregunta 3

Finalizado

Se puntúa como 0 sobre 1,00



Tiempo estimado 15 min

Para el esquema de la figura y dadas las siguientes definiciones de vistas:

```
CREATE OR REPLACE VIEW v_usuario_mayor
AS
SELECT DNI, e_mail, apellido, nombre, fecha_nac
FROM usuario
WHERE DNI > 12345678;

CREATE OR REPLACE VIEW v_mayor_gmail
AS
SELECT *
FROM v_usuario_mayor
WHERE e_mail like '%gmail%'
WITH LOCAL CHECK OPTION;

CREATE OR REPLACE VIEW v_mayor_parcial
AS
SELECT *
FROM v_usuario_mayor
WHERE apellido like 'Alva%'
WITH CASCADED CHECK OPTION;
```

Para las siguientes sentencias ejecutadas de manera independiente señalar las opciones que son FALSAS. Nota: Las respuestas incorrectas restan del puntaje total. Tenga cuidado al cortar y pegar las sentencias con las comillas simples ' '.

- [ ] a. INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
  VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
  Procede, inserta los datos en la tabla usuario pero no se muestran en la vista v_mayor_gmail

- [x] b. INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
  VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
  NO Procede, da error.

  Es falsa, ya que la operación procede insertando en la tabla base si error alguno al cumplir con el WHERE de v_mayor_gmail (LOCAL CHECK OPTION) y no evaluar el WHERE de v_usuario_mayor por no tener opción de chequeo en su definición

c. INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
NO Procede, porque no cumple con la condición de la vista v_usuario_mayor

Es falsa ya que la vista v_usuario_mayor no tiene opción de chequeo y por lo tanto no se comprueba la condición de su WHERE. La operación procede y la tupla se inserta en la tabla base.

d. INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
Procede, inserta los datos en la tabla usuario pero no se muestran en la vista v_usuario_mayor

e. INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
Procede, inserta los datos en la tabla usuario y no se muestran en la vista v_usuario_mayor, ni en en la vista v_mayor_gmail

f. INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
Procede, inserta los datos en la tabla usuario y se muestran en todas las vistas

Es Falsa, ya que la operación procede e inserta los datos en la tabla usuario ** pero la tupla no se muestra en la vista v_usuario_mayor porque DNI 11 millones no es mayor a DNI 12345678, por ende no se muestra en ninguna de las vistas que dependen de ella.
** La tupla se inserta en la tabla base porque cumple con el WHERE de v_mayor_gmail (LOCAL CHECK OPTION) y no evalúa el WHERE de v_usuario_mayor al no tener opción de chequeo en su definición.

Respuesta correcta

Las respuestas correctas son:

INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
Procede, inserta los datos en la tabla usuario y se muestran en todas las vistas,

INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
NO Procede, porque no cumple con la condición de la vista v_usuario_mayor,

INSERT INTO v_mayor_gmail (DNI, e_mail, apellido, nombre, fecha_nac)
VALUES (11234567, 'cc@gmail.com', 'Alvarado', 'J', to_date('20170103','YYYYMMDD'))
NO Procede, da error.

Pregunta 4

Finalizado

Se puntúa como 0 sobre 1,00

Utilizando el esquema unc_esq_voluntario. Cuántos voluntarios por cada tarea, nacidos entre 1998 y 1999, que pertenecen a alguna FUNDACION han realizado tareas cuyo identificador contiene MAN

Tiempo estimado 20 min

- a. MK_MAN 3, ST_MAN 1
- b. MK_MAN
- c. SD_MAN 1
- d. SA_MAN, 1; ST_MAN, 1
- e. Ninguna de las opciones es correcta

Respuesta correcta

SQL que lo resuelve:

```
SELECT t.id_tarea, COUNT(*) AS voluntarios
  FROM voluntario v
  JOIN institucion i ON v.id_institucion = i.id_institucion
  JOIN tarea t ON t.id_tarea = v.id_tarea
  WHERE EXTRACT(YEAR FROM v.fecha_nacimiento) BETWEEN 1998 AND 1999
    AND i.nombre_institucion like 'FUNDACION%'
    AND t.id_tarea LIKE '%MAN%'
  GROUP BY t.id_tarea
```

La respuesta correcta es: SA_MAN, 1; ST_MAN, 1

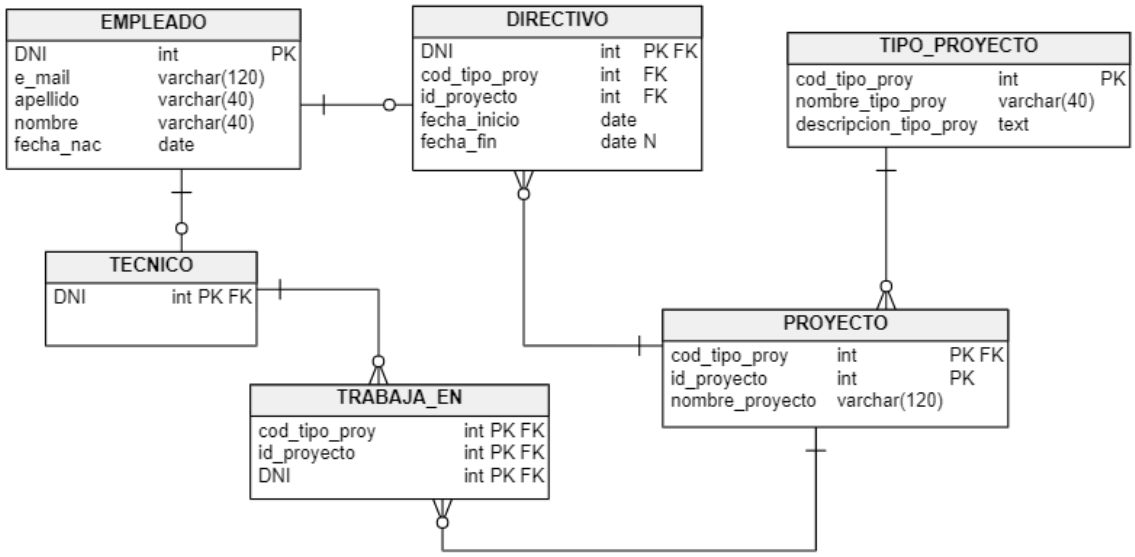
Pregunta 5

Finalizado

Se puntúa como 0 sobre 1,00

Tiempo estimado 25 min

Considere el siguiente esquema (script creación solo de tablas) correspondiente a un sistema de contratación de servicios, cuyo diagrama en Vertabelo es el siguiente:



Considere las siguientes sentencias SQL de declaración de claves primarias (PK) y claves extranjeras (FK) sobre el esquema dado:

```
-- Primary Key
ALTER TABLE DIRECTIVO ADD CONSTRAINT PK_DIRECTIVO PRIMARY KEY (DNI);
ALTER TABLE EMPLEADO ADD CONSTRAINT PK_EMPLEADO PRIMARY KEY (DNI);
ALTER TABLE TECNICO ADD CONSTRAINT PK_TECNICO PRIMARY KEY (DNI);
ALTER TABLE TIPO_PROYECTO ADD  CONSTRAINT PK_TIPO_PROYECTO PRIMARY KEY (cod_tipo_proy);

-- foreign keys
ALTER TABLE DIRECTIVO ADD CONSTRAINT FK_DIRECTIVO_PROYECTO
    FOREIGN KEY (id_proyecto, cod_tipo_proy)
    REFERENCES PROYECTO (id_proyecto, cod_tipo_proy);

ALTER TABLE PROYECTO ADD CONSTRAINT FK_PROYECTO_TIPO_PROYECTO
    FOREIGN KEY (cod_tipo_proy)
    REFERENCES TIPO_PROYECTO (cod_tipo_proy);

ALTER TABLE TRABAJA_EN ADD CONSTRAINT FK_TRABAJA_EN_TECNICO
    FOREIGN KEY (DNI)
    REFERENCES TECNICO (DNI);
```

Escriba las sentencias sentencias SQL de **declaración de claves primarias (PK), claves alterativas (AK) y claves extranjeras (FK)** que faltan teniendo en cuenta los siguiente:
a) Deben definirse las acciones referenciales para que cada vez que se elimine o modifique (PK) de un empleado debe hacerse también de las tablas subtipo de la jerarquía.
b) Los nombres de proyectos deben ser únicos.

```
-- Primary Key
ALTER TABLE DIRECTIVO ADD CONSTRAINT PK_DIRECTIVO PRIMARY KEY (DNI);
ALTER TABLE EMPLEADO ADD CONSTRAINT PK_EMPLEADO PRIMARY KEY (DNI);
ALTER TABLE TECNICO ADD CONSTRAINT PK_TECNICO PRIMARY KEY (DNI);
ALTER TABLE TIPO_PROYECTO ADD  CONSTRAINT PK_TIPO_PROYECTO PRIMARY KEY (cod_tipo_proy);
ALTER TABLE TRABAJA_EN ADD CONSTRAINT PK_TRABAJA_EN PRIMARY KEY (id_proyecto, cod_tipo_proy, DNI);
ALTER TABLE PROYECTO ADD CONSTRAINT PK_PROYECTO PRIMARY KEY (id_proyecto, cod_tipo_proy);

-- Foreign keys
ALTER TABLE DIRECTIVO ADD CONSTRAINT FK_DIRECTIVO_PROYECTO
    FOREIGN KEY (id_proyecto, cod_tipo_proy)
    REFERENCES PROYECTO (id_proyecto, cod_tipo_proy);
-- Inicio resuelto
ALTER TABLE DIRECTIVO ADD CONSTRAINT FK_DIRECTIVO_EMPLEADO
    FOREIGN KEY (DNI)
    REFERENCES EMPLEADO (DNI)
    ON UPDATE CASCADE
    ON DELETE CASCADE;
-- Fin resuelto

ALTER TABLE PROYECTO ADD CONSTRAINT FK_PROYECTO_TIPO_PROYECTO
    FOREIGN KEY (cod_tipo_proy)
    REFERENCES TIPO_PROYECTO (cod_tipo_proy);

ALTER TABLE TRABAJA_EN ADD CONSTRAINT FK_TRABAJA_EN_TECNICO
    FOREIGN KEY (DNI)
    REFERENCES TECNICO (DNI);
-- Inicio resuelto
ALTER TABLE TRABAJA_EN ADD CONSTRAINT FK_TRABAJA_EN_PROYECTO
    FOREIGN KEY (id_proyecto, cod_tipo_proy)
    REFERENCES PROYECTO (id_proyecto, cod_tipo_proy);

ALTER TABLE TECNICO ADD CONSTRAINT FK_TECNICO_EMPLEADO
    FOREIGN KEY (DNI)
    REFERENCES EMPLEADO (DNI)
    ON UPDATE CASCADE
    ON DELETE CASCADE;
-- Fin resuelto

-- End of file.

/* Pregunta 5

Escriba las sentencias sentencias SQL de declaración de claves primarias (PK), claves alterativas (AK)
y claves extranjeras (FK) que faltan teniendo en cuenta los siguiente:

a) Deben definirse las acciones referenciales para que cada vez que se elimine o modifique (PK) de un empleado debe
hacerse también de las tablas subtipo de la jerarquía.

b) Los nombres de proyectos deben ser únicos. */
ALTER TABLE PROYECTO ADD CONSTRAINT AK_PROYECTO UNIQUE (nombre_proyecto);
```

```
ALTER TABLE PROYECTO ADD CONSTRAINT PK_PROYECTO PRIMARY KEY (id_proyecto,cod_tipo_proy);
```

```
ALTER TABLE TRABAJA_EN ADD CONSTRAINT PK_TRABAJA_EN PRIMARY KEY (id_proyecto,cod_tipo_proy,DNI);
```

```
ALTER TABLE DIRECTIVO ADD CONSTRAINT FK_DIRECTIVO_EMPLEADO
    FOREIGN KEY (DNI)
    REFERENCES EMPLEADO (DNI)
    ON DELETE  CASCADE
    ON UPDATE  CASCADE;
```

```
ALTER TABLE TECNICO ADD CONSTRAINT FK_TECNICO_EMPLEADO
```

```
    FOREIGN KEY (DNI)
    REFERENCES EMPLEADO (DNI)
    ON DELETE  CASCADE
    ON UPDATE  CASCADE;
```

```
ALTER TABLE TRABAJA_EN ADD CONSTRAINT FK_TRABAJA_EN_PROYECTO
    FOREIGN KEY (id_proyecto, cod_tipo_proy)
    REFERENCES PROYECTO (id_proyecto, cod_tipo_proy);
```

```
ALTER TABLE PROYECTO ADD CONSTRAINT AK_PROYECTO UNIQUE (nombre_proyecto);
```

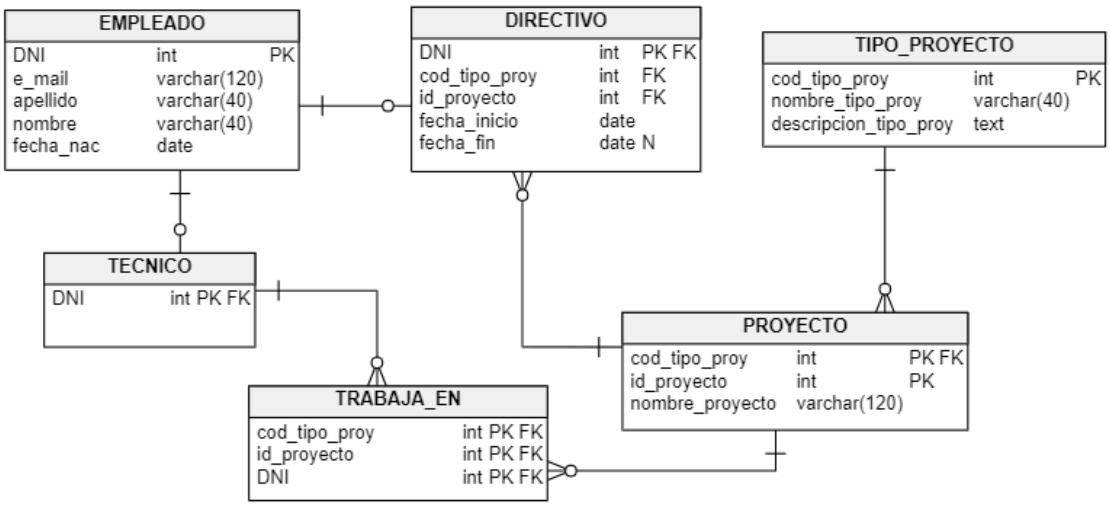
Pregunta 6

Finalizado

Se puntúa como 0 sobre 1,00

Tiempo estimado 15 min

Dado el esquema de la figura:



Si las foreign key fueran las que se indican debajo y se encuentran cargados los siguientes datos:

TIPO_PROYECTO

cod_tipo_proy	*nombre_tipo_proy*	....
C	nombre1	
B	nombre2	
F	nombre3	
D	nombre4	
E	nombre5	

PROYECTO

cod_tipo_proy	*id_proyecto*	*nombre_proye*
D	2	proyecto1
B	3	proyecto2
E	1	proyecto3
D	1	proyecto4
C	4	proyecto5

TRABAJA_EN

cod_tipo_proy	*id_proyecto*	*DNI*
B	3	263546
D	1	345689
B	3	405263
E	1	345689
D	1	375621

```
FK_PROYECTO_TIPO_PROYECTO
   ON UPDATE CASCADE
   ON DELETE RESTRICT
FK_TRABAJA_EN_PROYECTO
   ON UPDATE RESTRICT
   ON DELETE CASCADE
```

¿Cuál/es de las siguientes operaciones proceden y cuál/es fallan?

DELETE FROM TIPO_PROYECTO WHERE cod_tipo_proy ='E';

Falla

DELETE FROM PROYECTO WHERE cod_tipo_proy = 'D';

Procede

DELETE FROM TRABAJA_EN WHERE id_proyecto = 3;

Procede

DELETE FROM PROYECTO WHERE cod_tipo_proy ='E';

Procede

UPDATE PROYECTO SET cod_tipo_proy = 'B' WHERE cod_tipo_proy = 'D' AND id_proyecto = 2;

Procede

UPDATE TRABAJA_EN SET id_proyecto = 2 WHERE cod_tipo_proy = 'E' AND id_proyecto = 1;

Falla

UPDATE TIPO_PROYECTO SET cod_tipo_proy ='A' WHERE cod_tipo_proy ='D';

Falla

Respuesta correcta

La respuesta correcta es:
DELETE FROM TIPO_PROYECTO WHERE cod_tipo_proy ='E'; → Falla,

DELETE FROM PROYECTO WHERE cod_tipo_proy = 'D';

→ Procede,
DELETE FROM TRABAJA_EN WHERE id_proyecto = 3; → Procede,
DELETE FROM PROYECTO WHERE cod_tipo_proy ='E'; → Procede,

UPDATE PROYECTO SET cod_tipo_proy = 'B' WHERE cod_tipo_proy = 'D' AND id_proyecto = 2;

→ Procede,

UPDATE TRABAJA_EN SET id_proyecto = 2 WHERE cod_tipo_proy = 'E' AND id_proyecto = 1;

→ Falla,

UPDATE TIPO_PROYECTO SET cod_tipo_proy ='A' WHERE cod_tipo_proy ='D'; → Falla

Actividad previa

◀ Práctico 9 - SQL Avanzado

Ir a...

Mantente en contacto

Facultad, Pabellón Central Paraje Arroyo Seco. Campus Universitario. (B7001BBO) Tandil. Buenos Aires, Argentina

https://exa.unicen.edu.ar/
(+54) (0249) 438-5650 Conmutador: int. 2000
moodle@exa.unicen.edu.ar

Descargar la app para dispositivos móviles

Facultad de Ciencias Exactas – UNICEN

Contacto administradores plataforma: E-mail moodle@exa.unicen.edu.ar – Tel. +54 0249 4385650 int. 2098