**Universidad ORT Uruguay**

**Facultad de Ingeniería**

**Bernard Wand Polak**

**Base de Datos 2**

**Obligatorio - Entrega 2**

**Rodrigo Demicheli, Nro. est. 103036**

**Mauricio Carballo, Nro. est. 169199**

**Grupo N6B**

**Docente: Gabriel Fialco**

**Formulario de Antecedentes**

**Curso Base de Datos II**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nro. Estudiante** | 103036 | |  | | --- | | C:\Users\tanquista\Desktop\QUI69AB.tmp.jpg | |
| **Nombre:** | Rodrigo |
| **Apellido:** | Demicheli |
| **Grupo / Turno:** | N6B |
| **¿Trabaja en algo relacionado con la carrera?**  **¿Qué tareas desempeña?** | Si, programación y bases de datos. |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nro. Estudiante** | 169199 | |  | | --- | | QUI1338 | |
| **Nombre:** | Mauricio |
| **Apellido:** | Carballo |
| **Grupo / Turno:** | N6B |
| **¿Trabaja en algo relacionado con la carrera?**  **¿Qué tareas desempeña?** | Si, programación y bases de datos. |
|  |  |

Tabla de contenido

[1. Análisis del problema 1](#_Toc310448534)

[1.1 Supuestos 1](#_Toc310448535)

[1.2 Modelo Relacional 1](#_Toc310448536)

[1.3 Restricciones 3](#_Toc310448537)

[1.4 Justificación de las restricciones globales 5](#_Toc310448538)

[1.4.1 Salon 5](#_Toc310448539)

[1.4.2 Inscribe 5](#_Toc310448540)

[1.4.3 Aprueba 5](#_Toc310448541)

[1.4.4 Rinde 6](#_Toc310448542)

[2. DDL de la solución 7](#_Toc310448543)

[3. Casos de prueba 9](#_Toc310448544)

[3.1 Salon 9](#_Toc310448545)

[3.2 Inscribe 10](#_Toc310448546)

[3.3 Aprueba 11](#_Toc310448547)

[3.4 Rinde 12](#_Toc310448548)

[4. Cambios para la segunda entrega 18](#_Toc310448549)

[4.2 Supuestos 18](#_Toc310448550)

[4.1 Modelo relacional 18](#_Toc310448551)

[4.2 Restricciones 18](#_Toc310448552)

[4.3 Restricciones Globales 19](#_Toc310448553)

[4.3.1 Nuevas 19](#_Toc310448554)

[4.3.2 Correcciones y modificaciones 19](#_Toc310448555)

[4.4 DDL 20](#_Toc310448556)

# 1. Análisis del problema

## 1.1 Supuestos

**SUP1:** Para rendir un examen, este debe estar disponible.

**SUP2:** La calificación máxima en un examen es 100 puntos.

**SUP3:** La calificación de un examen para aprobarlo debe ser mayor o igual a 70 puntos.

**SUP4:** Todas las fechas y horas son GMT-0.

**SUP5:** En un mismo salón pueden tomarse diferentes exámenes al mismo tiempo.

## 1.2 Modelo Relacional

**Estudiante** (nroEstudiante, nroPasaporte, pais, nombre, apellido, ciudadResidencia)

nroEstudiante PK

(nroPasaporte, pais) AK

**Examen** (nroExamen, descripcion, disponible)

nroExamen PK

**Institucion** (nombre, pais, ciudad, direccion)

nombre PK

**Salon** (nombreInstitucion, nroSalon, nroSillaMaximo, nroSillaMinimo)

(nombreInstitucion, nroSalon) PK

nombreInstitucion FK Institucion (nombre)

**Inscribe** (nroEstudiante, nroExamen)

(nroEstudiante, nroExamen) PK

nroEstudiante FK Estudiante

nroExamen FK Examen

**Aprueba** (nroEstudiante, nroExamen, fecha, calificacion)

(nroEstudiante, nroExamen) PK

nroEstudiante FK Estudiante

nroExamen FK Examen

**Instancia\_Ex** (nroExamen, nombreInstitucion, fecha, hora)

(nroExamen, nombreInstitucion, fecha) PK

nroExamen FK Examen

nombreInstitucion FK Institucion (nombre)

**Rinde** (nroExamen, nroEstudiante, nombreInstitucion, nroSalon, fecha, nroSillaAsignado)

(nroExamen, nroEstudiante, nombreInstitucion, nroSalon, fecha) PK

(nroExamen, nroEstudiante) FK Inscribe

(nroExamen, nombreInstitucion, fecha) FK InstanciaEx

(nombreInstitucion, nroSalon) FK Salon

(nroEstudiante, fecha) AK Rinde

## 1.3 Restricciones





## 1.4 Justificación de las restricciones globales

### 1.4.1 Salon

**Motivo:** al modificar el rango máximo o mínimo de números de silla del salón, no puede quedar ninguna silla asignada a un estudiante para rendir un examen fuera del nuevo rango.

**Tablas involucradas:** Rinde, Salon.

Al rendir un Examen, a un Estudiante se le asigna un número de silla. Ese número de silla debe estar dentro del rango mínimo y máximo de números de silla del Salon donde se va a rendir el Examen. Si se modificara dicho rango, eso no puede provocar que las sillas asignadas al Salon queden fuera del nuevo rango.

### 1.4.2 Inscribe

**Motivo:** al inscribirse un estudiante a un examen, este debe estar disponible.

**Tablas involucradas:** Inscribe, Examen.

Asumimos que un estudiante puede inscribirse a un examen solo si este está disponible, entonces al insertar o actualizar una tupla en la tabla Inscribe, hay que controlar que el examen al que se está inscribiendo el estudiante esté disponible en la tabla Examen.

### 1.4.3 Aprueba

**Motivo**: La fecha de aprobación de un examen debe ser como mínimo siete días posteriores a la fecha del examen.

**Tablas involucradas:** Rinde, Instancia\_ex, Aprueba.

Esta restricción esta especificada explícitamente en la realidad del problema, y para controlar que se cumpla generamos un trigger sobre la tabla Aprueba, que en el insert o update de alguna tupla, chequea que el examen que se está señalando como aprobado, haya sido rendido como mínimo siete días antes de la fecha de aprobación del mismo.

### 1.4.4 Rinde

**Motivo**: El número de silla debe estar dentro del rango de los números de silla de Salón.

**Tablas involucradas:** Salon, Instancia\_ex, Inscribe, Rinde.

Se está controlando que al insertar una tupla en la tabla Rinde, el campo nro\_silla\_asignado tenga un valor que este dentro del rango mínimo y máximo de números de silla del Salon donde se Rinde el Examen.

**Motivo**: Se debe controlar que si el estudiante ya aprobó el examen no pueda volver a rendirlo.

**Tablas involucradas:** Rinde, Aprueba.

Esta restricción se especifica en la letra del problema.

**Motivo**: A cada estudiante la primera vez que se inscribe a un examen, se le otorga un número de silla dentro de un salón de la institución elegida. Luego por razones de control, cada examen debe rendirlo en el mismo salón y silla a excepción de que se solape con otro estudiante, en dicho caso el sistema deberá buscar otro salón, en la misma institución, con el número de silla disponible, si no es posible deberá conseguir en el salón inicial otra silla, de no haber buscará una disponible en otro salón. Si al final no se puede encontrar silla el estudiante no podrá en dicha instancia rendir el examen.

**Tablas involucradas:** Rinde, Salon.

Esta restricción se especifica en la letra del problema.

# 2. DDL de la solución

DROP TABLE aprueba CASCADE CONSTRAINTS;

DROP TABLE estudiante CASCADE CONSTRAINTS;

DROP TABLE examen CASCADE CONSTRAINTS;

DROP TABLE inscribe CASCADE CONSTRAINTS;

DROP TABLE instancia\_ex\_data CASCADE CONSTRAINTS;

DROP VIEW instancia\_ex CASCADE CONSTRAINTS;

DROP TABLE institucion CASCADE CONSTRAINTS;

DROP TABLE salon CASCADE CONSTRAINTS;

DROP TABLE rinde\_data CASCADE CONSTRAINTS;

DROP VIEW rinde CASCADE CONSTRAINTS;

CREATE TABLE estudiante (

nro\_estudiante NUMBER CHECK(nro\_estudiante >=0),

nro\_pasaporte NUMBER NOT NULL CHECK(nro\_pasaporte >=0),

pais VARCHAR2(30) NOT NULL,

nombre VARCHAR2(50) NOT NULL,

apellido VARCHAR2(50) NOT NULL,

ciudad\_residencia VARCHAR2(50),

CONSTRAINT estudiante\_pk PRIMARY KEY (nro\_estudiante),

CONSTRAINT estudiante\_ak UNIQUE (nro\_pasaporte, pais)

);

CREATE TABLE examen (

nro\_examen NUMBER CHECK(nro\_examen >=0),

descripcion VARCHAR2(200),

disponible CHAR(1) CHECK (disponible IN ('S', 'N')) NOT NULL,

CONSTRAINT examen\_pk PRIMARY KEY (nro\_examen)

);

CREATE TABLE institucion (

nombre VARCHAR2(50),

pais VARCHAR2(30) NOT NULL,

ciudad VARCHAR2(50) NOT NULL,

direccion VARCHAR2 (100) NOT NULL,

timezone VARCHAR2(6) NOT NULL CHECK (REGEXP\_LIKE(timezone, '^(\+((0[127]:00)|(0[3-69]:[03]0)|(1[34]:00)|(1[01]:[03]0)|((12|0[58]):(00|45))))|(\-((0[1258]:00)|(0[349]:[03]0)|(1[0-2]:[03]0)))|(00:00)$')),

CONSTRAINT institucion\_pk PRIMARY KEY (nombre)

);

CREATE TABLE salon (

nombre\_institucion VARCHAR2(50),

nro\_salon NUMBER CHECK(nro\_salon >=0),

nro\_silla\_max NUMBER NOT NULL CHECK (nro\_silla\_max >= 0),

nro\_silla\_min NUMBER NOT NULL CHECK (nro\_silla\_min >= 0),

CONSTRAINT salon\_pk PRIMARY KEY (nombre\_institucion, nro\_salon),

CONSTRAINT salon\_fk FOREIGN KEY (nombre\_institucion) REFERENCES institucion (nombre),

CONSTRAINT check\_nroSillaMax CHECK (nro\_silla\_max >= nro\_silla\_min)

);

CREATE TABLE inscribe (

nro\_estudiante NUMBER CHECK(nro\_estudiante >=0),

nro\_examen NUMBER CHECK(nro\_examen >=0),

CONSTRAINT inscribe\_pk PRIMARY KEY (nro\_estudiante, nro\_examen),

CONSTRAINT inscribe\_fk\_nro\_est FOREIGN KEY (nro\_estudiante) REFERENCES estudiante (nro\_estudiante),

CONSTRAINT inscribe\_fk\_nro\_ex FOREIGN KEY (nro\_examen) REFERENCES examen (nro\_examen)

);

CREATE TABLE aprueba (

nro\_estudiante NUMBER CHECK(nro\_estudiante >=0),

nro\_examen NUMBER CHECK(nro\_examen >=0),

fecha DATE NOT NULL,

calificacion NUMBER (3) NOT NULL CHECK(calificacion BETWEEN 70 and 100),

CONSTRAINT aprueba\_pk PRIMARY KEY (nro\_estudiante, nro\_examen),

CONSTRAINT aprueba\_fk\_nro\_est FOREIGN KEY (nro\_estudiante) REFERENCES examen (nro\_estudiante),

CONSTRAINT aprueba\_fk\_nro\_ex FOREIGN KEY (nro\_examen) REFERENCES examen (nro\_examen)

);

CREATE TABLE instancia\_ex (

nro\_examen NUMBER CHECK(nro\_examen >=0),

nombre\_institucion VARCHAR2(50),

fecha DATE,

hora TIMESTAMP NOT NULL,

CONSTRAINT instancia\_ex\_pk PRIMARY KEY (nro\_examen, nombre\_institucion, fecha),

CONSTRAINT instancia\_ex\_fk\_nro\_ex FOREIGN KEY (nro\_examen) REFERENCES examen (nro\_examen),

CONSTRAINT instancia\_ex\_fk\_nombre FOREIGN KEY (nombre\_institucion) REFERENCES institucion (nombre)

);

CREATE TABLE rinde\_data (

nro\_examen NUMBER CHECK(nro\_examen >=0),

nro\_estudiante NUMBER check(nro\_estudiante >=0),

nombre\_institucion VARCHAR2(50),

nro\_salon NUMBER check(nro\_salon >=0),

fecha DATE,

nro\_silla\_asignado NUMBER check(nro\_silla\_asignado >=0) NOT NULL,

CONSTRAINT rinde\_pk

PRIMARY KEY (nro\_examen, nro\_estudiante, nombre\_institucion, nro\_salon, fecha),

CONSTRAINT rinde\_fk\_inscribe

FOREIGN KEY (nro\_examen, nro\_estudiante)

REFERENCES inscribe (nro\_examen, nro\_estudiante),

CONSTRAINT rinde\_fk\_instancia\_ex

FOREIGN KEY (nro\_examen, nombre\_institucion, fecha)

REFERENCES instancia\_ex (nro\_examen, nombre\_institucion, fecha),

CONSTRAINT rinde\_fk\_salon

FOREIGN KEY (nombre\_institucion, nro\_salon)

REFERENCES salon (nombre\_institucion, nro\_salon),

CONSTRAINT rinde\_unique UNIQUE (nro\_estudiante, fecha)

);

CREATE VIEW rinde AS SELECT \* FROM rinde\_data;

Para la relación Rinde decidimos usar una tabla mutante porque uno de los triggers que se crearon en esa relación, hacia una consulta a la tabla en el momento de la ejecución del trigger. Para esto seguimos el procedimiento visto en el práctico.

# 3. Casos de prueba

## 3.1 Salon

**Motivo:** al modificar el rango máximo o mínimo de números de silla del salón, no puede quedar ninguna silla asignada a un estudiante para rendir un examen fuera del nuevo rango.

**Considerando las tablas inicialmente vacías, inserts previos necesarios para las pruebas**:

INSERT INTO estudiante VALUES (1, 2, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO institucion VALUES ('inst1', 'paraguay', 'asuncion', 'chilavert road 1354');

INSERT INTO salon VALUES ('inst1', 1, 30, 1);

INSERT INTO institucion VALUES ('inst2', 'ARGENTINA', 'BS', 'Kichener');

INSERT INTO salon VALUES ('inst2', 1, 5, 1);

INSERT INTO examen VALUES (1, 'test de cooper', 'S');

INSERT INTO inscribe VALUES (1, 1);

INSERT INTO instancia\_ex VALUES (1, 'inst1',

to\_date('2011/01/01', 'yyyy/mm/dd'),

to\_date('2011/01/01 16:00:', 'YYYY/MM/DD HH24:MI:');

INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 5);

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| Actualizo los rangos de número de silla de un Salon de manera que todas las tuplas de Rinde quedan consistentes. | UPDATE salon  SET nro\_silla\_min = 10, nro\_silla\_max = 17  WHERE nombre\_institucion = 'inst2'; | 1 fila actualizada. | Si. |
| Actualizo el rango mínimo de número de silla de un Salon de manera que alguna tupla de Rinde queda inconsistente. | UPDATE salon  SET nro\_silla\_min = 10  WHERE nombre\_institucion = 'inst1'; | Error -20001 ,'NO SE PUEDE ODIFICAR EL RANGO DE SILLAS DEL SALON PUES YA HAY SILLAS ASIGNADAS EN LA RELACION RINDE' | Si. |
| Actualizo el rango máximo de número de silla de un Salon de manera que alguna tupla de Rinde queda inconsistente. | UPDATE salon  SET nro\_silla\_max = 2  WHERE nombre\_institucion = 'inst1'; | Error -20001, 'NO SE PUEDE ODIFICAR EL RANGO DE SILLAS DEL SALON PUES YA HAY SILLAS ASIGNADAS EN LA RELACION RINDE' | Si. |

## 3.2 Inscribe

**Motivo:** al inscribirse un estudiante a un examen, este debe estar disponible.

**Considerando las tablas inicialmente vacías, inserts previos necesarios para las pruebas**:

INSERT INTO estudiante VALUES (1, 2, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO examen VALUES (1, 'test de cooper', 'S');

INSERT INTO examen VALUES (2, 'tiro al arco', 'N');

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| Inscribo un Estudiante en un Examen que está disponible. | INSERT INTO inscribe VALUES (1, 1); | 1 fila insertada. | Si. |
| Inscribo un Estudiante en un Examen que no está disponible. | INSERT INTO inscribe VALUES (1, 2); | Error -20001, 'El examen debe estar disponible para poder inscribirse a él.' | Si. |

## 3.3 Aprueba

**Motivo:** : La fecha de aprobación de un examen debe ser como mínimo siete días posteriores a la fecha del examen.

**Considerando las tablas inicialmente vacías, inserts previos necesarios para las pruebas**:

INSERT INTO estudiante VALUES (1, 2, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO institucion VALUES ('inst1', 'paraguay', 'asuncion', 'chilavert road 1354');

INSERT INTO salon VALUES ('inst1', 1, 30, 1);

INSERT INTO examen VALUES (1, 'test de cooper', 'S');

INSERT INTO inscribe VALUES (1, 1);

INSERT INTO instancia\_ex VALUES (1, 'inst1',

to\_date('2011/01/01', 'yyyy/mm/dd'),

to\_date('2011-12-02 16:00:', 'YYYY-MM-DD HH24:MI:'));

INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 5);

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| Apruebo un Examen 15 días después de la fecha en que se rindió. | INSERT INTO aprueba VALUES (1, 1, to\_date('2011/01/15', 'yyyy/mm/dd'), 95); | 1 fila insertada. | Si. |
| Apruebo un Examen 5 días después de la fecha en que se rindió. | INSERT INTO aprueba VALUES (1, 1, to\_date('2011/01/05', 'yyyy/mm/dd'), 95); | Error -20001, 'La fecha de aprobación de un examen debe ser al menos siete días posterior a la fecha del mismo.' | Si. |

## 3.4 Rinde

**Motivo**: El número de silla debe estar dentro del rango de los números de silla de Salón.

**Considerando las tablas inicialmente vacías, inserts previos necesarios para las pruebas**:

INSERT INTO estudiante VALUES (1, 1, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (2, 2, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (3, 3, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO institucion VALUES ('inst1', 'paraguay', 'asuncion', 'chilavert road 1354');

INSERT INTO salon VALUES ('inst1', 1, 30, 1);

INSERT INTO examen VALUES (1, 'test de cooper', 'S');

INSERT INTO inscribe VALUES (1, 1);

INSERT INTO inscribe VALUES (2, 1);

INSERT INTO inscribe VALUES (3, 1);

INSERT INTO instancia\_ex VALUES (1, 'inst1',

to\_date('2011/01/01', 'yyyy/mm/dd'),

to\_date('2011-12-02 16:00:', 'YYYY-MM-DD HH24:MI:'));

INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| Inserto una tupla en Rinde donde el numero de silla esta dentro del rango de números de silla del salon. | INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2); | 1 fila insertada. | Si. |
| Actualizo una tupla en Rinde donde el numero de silla esta dentro del rango de números de silla del salon. | UPDATE rinde  SET nro\_silla\_asignado = 5  WHERE nro\_examen = 1  AND nro\_estudiante = 1  AND nombre\_institucion = 'inst1'  AND nro\_salon = 1  AND fecha = to\_date('2011/01/01', 'yyyy/mm/dd'); | 1 fila actualizada. | Si. |
| Inserto una tupla en Rinde donde el número de silla esta fuera del rango de números de silla del salon. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1500); | Error -20001, 'El numero de silla no es válido, debe estar dentro del rango máximo y mínimo del salon.' | Si. |
| Actualizo una tupla en Rinde donde el número de silla esta fuera del rango de números de silla del salon. | UPDATE rinde  SET nro\_silla\_asignado = 700  WHERE nro\_examen = 1  AND nro\_estudiante = 1  AND nombre\_institucion = 'inst1'  AND nro\_salon = 1  AND fecha = to\_date('2011/01/01', 'yyyy/mm/dd'); | Error -20001, 'El numero de silla no es válido, debe estar dentro del rango máximo y mínimo del salon.' | Si. |

**Motivo**: Se debe controlar que si el estudiante ya aprobó el examen no pueda volver a rendirlo.

**Considerando las tablas inicialmente vacías, inserts previos necesarios para las pruebas**:

INSERT INTO estudiante VALUES (1, 1, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (2, 2, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (3, 3, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO institucion VALUES ('inst1', 'paraguay', 'asuncion', 'chilavert road 1354');

INSERT INTO salon VALUES ('inst1', 1, 30, 1);

INSERT INTO examen VALUES (1, 'test de cooper', 'S');

INSERT INTO inscribe VALUES (1, 1);

INSERT INTO inscribe VALUES (2, 1);

INSERT INTO inscribe VALUES (3, 1);

INSERT INTO instancia\_ex VALUES (1, 'inst1', to\_date('2011/01/01', 'yyyy/mm/dd'), to\_date('2011-12-02 16:00:', 'YYYY-MM-DD HH24:MI:'));

INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);

INSERT INTO aprueba VALUES (1, 1, to\_date('2011/01/10', 'yyyy/mm/dd'), 80);

INSERT INTO instancia\_ex VALUES (1, 'inst1', to\_date('2011/02/01', 'yyyy/mm/dd'), to\_date('2011-12-02 18:00:', 'YYYY-MM-DD HH24:MI:'));

INSERT INTO examen VALUES (2, 'test drive MINI cooper XD', 'S');

INSERT INTO instancia\_ex VALUES (2, 'inst1', to\_date('2011/01/01', 'yyyy/mm/dd'));

INSERT INTO inscribe VALUES (1, 2);

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| Aprueba un examen que no aprobó. | INSERT INTO aprueba VALUES (1, 2, to\_date('2011/02/10', 'yyyy/mm/dd'), 80); | 1 fila insertada. | Si. |
| Rinde un examen que ya tiene aprobado. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/02/01', 'yyyy/mm/dd'), 1); | Error, -20001,'El estudiante ya tiene aprobado el examen ' | Si. |
| Cambio un estudiante a un examen que ya tiene aprobado. | UPDATE rinde  SET nro\_examen = 2  WHERE  nro\_estudiante = 1  AND nombre\_institucion = 'inst1'  AND nro\_salon = 1  AND nro\_silla\_asignado = 1  AND fecha = to\_date('2011/01/01', 'yyyy/mm/dd'); | Error, -20001,'El estudiante ya tiene aprobado el examen ' | Si. |

**Motivo**: A cada estudiante la primera vez que se inscribe a un examen, se le otorga un número de silla dentro de un salón de la institución elegida. Luego por razones de control, cada examen debe rendirlo en el mismo salón y silla a excepción de que se solape con otro estudiante, en dicho caso el sistema deberá buscar otro salón, en la misma institución, con el número de silla disponible, si no es posible deberá conseguir en el salón inicial otra silla, de no haber buscará una disponible en otro salón. Si al final no se puede encontrar silla el estudiante no podrá en dicha instancia rendir el examen.

**Considerando las tablas inicialmente vacías, inserts previos necesarios para las pruebas**:

INSERT INTO estudiante VALUES (1, 1, 'paraguay', 'est1', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (2, 2, 'paraguay', 'est2', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (3, 3, 'paraguay', 'est3', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (4, 4, 'paraguay', 'est4', 'apellido', 'asuncion');

INSERT INTO estudiante VALUES (5, 5, 'paraguay', 'est5', 'apellido', 'asuncion');

INSERT INTO institucion VALUES ('inst1', 'paraguay', 'asuncion', 'chilavert road 1354');

INSERT INTO salon VALUES ('inst1', 1, 2, 1);

INSERT INTO salon VALUES ('inst1', 2, 2, 1);

INSERT INTO examen VALUES (1, 'test de cooper', 'S');

INSERT INTO inscribe VALUES (1, 1);

INSERT INTO inscribe VALUES (2, 1);

INSERT INTO inscribe VALUES (3, 1);

INSERT INTO inscribe VALUES (4, 1);

INSERT INTO inscribe VALUES (5, 1);

INSERT INTO instancia\_ex VALUES (1, 'inst1', to\_date('2011/01/01', 'yyyy/mm/dd'), to\_date('2011/01/01 16:00:', 'yyyy/mm/dd HH24:MI:'));

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| La silla está disponible en el salón en el que se quiere insertar. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  INSERT INTO rinde VALUES (1, 3, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1); | 3 filas insertadas. | Si. |
| No había ningún salón que tuviera el número de silla disponible, entonces busco una silla disponible en el salón inicial. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  --Este es el insert problemático  INSERT INTO rinde VALUES (1, 3, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1); | 2 filas insertadas.  Error, -20001, 'MISMO SALON NUEVA SILLA, DEBE INSERTAR EN EL SALON 1, SILLA 2 ' | Si. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| La silla en el salón inicial está ocupada, entonces se busca otro salón que tenga disponible el mismo número de silla. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el insert problemático  INSERT INTO rinde VALUES (1, 3, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1); | 2 filas insertadas.  Error, -20001, 'NUEVO SALON MISMA SILLA, DEBE INSERTAR EN EL SALON 2, SILLA 1 ' | Si. |
| No se encontró la silla buscada en ningún salón, entonces se busca la primera silla libre en cualquier salón. | --aumento la cantidad de sillas disponibles  UPDATE SALON SET nro\_silla\_max = 3 where nro\_salon = 1 and nombre\_institucion = 'inst1';  INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  INSERT INTO rinde VALUES (1, 3, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 4, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el insert problemático.  INSERT INTO rinde VALUES (1, 5, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1); | 1 fila actualizada.  4 filas insertadas.  Error, -20001 'CUALQUIER SALON, CUALQUIER SILLA, DEBE INSERTAR EN EL SALON 1, SILLA 3' | Si. |
| No puede dar el examen, no hay sillas disponibles en ningún salón de la Institución. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  INSERT INTO rinde VALUES (1, 3, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 4, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el insert problemático  INSERT INTO rinde VALUES (1, 5, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1); | 4 filas insertadas.  Error -20001, 'NO PUEDES DAR EL EXAMEN'. | SI. |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| La silla está disponible en el salón en el que se quiere actualizar. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el update exitoso  UPDATE rinde  SET nro\_salon= 2  WHERE nro\_estudiante = 2; | 2 filas insertadas.  1 filas actualizadas. | Si. |
| No permite el update y en el mensaje informa que se debe usar la  silla 2 del salón 1, si se quiere realizar un update exitoso. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  --Este es el update problemático  UPDATE rinde  SET nro\_salon= 1  WHERE nro\_estudiante = 2; | 2 filas insertadas.  Error, -20001, 'MISMO SALON NUEVA SILLA, DEBE INSERTAR EN EL SALON 1, SILLA 2 ' | Si. |
| No permite el update y en el mensaje informa que se debe usar la  silla 1 del salón 2, si se quiere realizar un update exitoso. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el update problemático  UPDATE rinde  SET nro\_salon= 1  WHERE nro\_estudiante = 1; | 2 filas insertadas.  Error, -20001, 'NUEVO SALON MISMA SILLA, DEBE INSERTAR EN EL SALON 2, SILLA 1 ' | SI. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de prueba** | **Sentencia SQL** | **Resultado Esperado** | **Resultado obtenido ok?** |
| No permite el update y en el mensaje informara que se debe usar la silla 3 del salón 1, si se quiere realizar un update exitoso. | --aumento la cantidad de sillas disponibles  UPDATE SALON SET nro\_silla\_max = 3 where nro\_salon = 1 and nombre\_institucion = 'inst1';  INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  INSERT INTO rinde VALUES (1, 3, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 4, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el update problemático.  UPDATE rinde  SET nro\_salon= 2  WHERE nro\_estudiante = 3; | 1 fila actualizada.  4 filas insertadas.  Error, -20001 'CUALQUIER SALON, CUALQUIER SILLA, DEBE INSERTAR EN EL SALON 1, SILLA 3' | Si. |
| No permite el update y en el mensaje informa  que para esa instancia de examen no hay mas lugares. | INSERT INTO rinde VALUES (1, 1, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 2, 'inst1', 1, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  INSERT INTO rinde VALUES (1, 3, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 1);  INSERT INTO rinde VALUES (1, 4, 'inst1', 2, to\_date('2011/01/01', 'yyyy/mm/dd'), 2);  --Este es el update problemático  UPDATE rinde  SET nro\_salon= 2  WHERE nro\_estudiante = 3; | 4 filas insertadas.  Error -20001, 'NO PUEDES DAR EL EXAMEN'. | SI. |

# 4. Cambios para la segunda entrega

## 4.2 Supuestos

**SUP6:** No puede haber dos instancias de examen del mismo examen consecutivas sin que se haya realizado la corrección de la primera.

## 4.1 Modelo relacional

Se agregó el atributo timezone a la relación Institucion para representar la zona horaria en la que se encuentra cada institución, ya que tenemos un supuesto que dice que el sistema trabaja con las fechas y horas en GMT-0.

**Institucion** (nombre, pais, ciudad, direccion, timezone)

nombre PK

## 4.2 Restricciones

Se agrego una restricción para controlar que un alumno no puede aprobar un examen que no rindió.



Se agregaron restricciones para el atributo timezone en la relación Institucion , se controla que tenga valor y además que el valor sea una zona horaria valida, para esto último usamos una expresión regular.

Ejemplos de zonas horarias validas: "-03:00", "-04:00", "+02:00".



A partir de la corrección de la primer entrega se agrego una restricción para controlar que un examen se esté tomando en el mismo momento en todo el mundo en la relación InstanciaEx.



## 4.3 Restricciones Globales

### 4.3.1 Nuevas

**Motivo**: Un examen debe tomarse en todo el mundo en el mismo momento

**Tablas involucradas:** Instancia\_ex.

Esta restricción surge de la corrección de la primer entrega, para implementarla tuvimos que convertir la tabla instancia\_ex en mutante, renombrándola a instancia\_ex\_data y creando la vista instancia\_ex.

Luego se codifico un trigger INSTEAD OF INSERT OR UPDATE en el que se controla que las instancias de examen existentes para un numero de examen y la nueva (o actualizada) sean todas a la misma hora, de lo contrario no se ejecuta INSERT o UPDATE y se muestra un mensaje de error.

### 4.3.2 Correcciones y modificaciones

**Trigger sobre la tabla aprueba:**

Había un problema cuando se seleccionaba el campo fecha del join entre rinde e instancia\_ex, se estaban seleccionando múltiples fechas en lugar de una sola, se corrigió haciendo MAX(fecha).

Con esto efectivamente controlamos que se apruebe en la última instancia de examen rendida.

Además se agrego el control de que si no se rindió el examen tampoco se puede aprobarlo.

## 4.4 DDL

El DDL completo y corregido está en la sección ["DDL de la solucion",](#_2._DDL_de) estos son los cambios y correcciones:

* Se renombro la tabla instancia\_ex a instancia\_ex\_data.
* Se creó la vista instancia\_ex.
* Se agrego el campo timezone en la tabla institucion.
* Se agrego un CHECK sobre el campo timezone de la tabla institucion.
* Se corrigió un error en una clave foránea de la tabla aprueba, aprueba\_fk\_nro\_est estaba dependiendo de un campo incorrecto (dependia de nro\_examen de la tabla examen cuando lo correcto era nro\_estudiante de esa tabla).

# 4. Cambios para la segunda entrega