**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](#_Toc310513273)

[1.1 Supuestos 1](#_Toc310513274)

[1.2 Modelo Relacional 1](#_Toc310513275)

[1.3 Restricciones 3](#_Toc310513276)

[1.4 Justificación de las restricciones globales 5](#_Toc310513277)

[1.4.1 Salon 5](#_Toc310513278)

[1.4.2 Inscribe 5](#_Toc310513279)

[1.4.3 Aprueba 5](#_Toc310513280)

[1.4.4 Rinde 6](#_Toc310513281)

[2. DDL de la solución 7](#_Toc310513282)

[3. Casos de prueba 9](#_Toc310513283)

[3.1 Salon 9](#_Toc310513284)

[3.2 Inscribe 10](#_Toc310513285)

[3.3 Aprueba 11](#_Toc310513286)

[3.4 Rinde 12](#_Toc310513287)

[4. Cambios para la segunda entrega 18](#_Toc310513288)

[4.2 Supuestos 18](#_Toc310513289)

[4.1 Modelo relacional 18](#_Toc310513290)

[4.2 Restricciones 18](#_Toc310513291)

[4.3 Restricciones Globales 19](#_Toc310513292)

[4.3.1 Nuevas 19](#_Toc310513293)

[4.3.2 Correcciones y modificaciones 19](#_Toc310513294)

[4.4 DDL 20](#_Toc310513295)

[5. Análisis y diseño de procedimientos y funciones 20](#_Toc310513296)

[5.1 Procedimiento para inscripción a instancias de examen 20](#_Toc310513297)

[5.1.1 Análisis y diseño 20](#_Toc310513298)

[5.2 Informe de Exámenes 22](#_Toc310513299)

[5.2.1 Análisis y diseño 22](#_Toc310513300)

[5.2.2 Salidas de ejemplo 23](#_Toc310513301)

[5.3 Informe de Aprobación 25](#_Toc310513302)

[5.3.1 Análisis y diseño 25](#_Toc310513303)

[5.3.2 Salidas de ejemplo 26](#_Toc310513304)

[5.4 Mejoramiento de calidad 27](#_Toc310513305)

[5.4.1 Análisis y diseño 27](#_Toc310513306)

[7. Código 29](#_Toc310513307)

[7.1 Procedimiento para inscripción a instancias de examen 29](#_Toc310513308)

[7.2 Informe de Exámenes 33](#_Toc310513309)

[7.3 Informe de Aprobación 36](#_Toc310513310)

[7.4 Convertir fecha y hora a la fecha y hora locales de una zona horaria 38](#_Toc310513311)

[7.5 Mejoramiento de calidad 38](#_Toc310513312)

[7.5.1 Procedimiento para cargar tabla calidad 38](#_Toc310513313)

[7.5.2 Procedimiento auxiliar incrementar\_si\_aprueba 40](#_Toc310513314)

# 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, timezone)

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 (dependía de nro\_examen de la tabla examen cuando lo correcto era nro\_estudiante de esa tabla) (fue un error de copy-paste).

# 5. Análisis y diseño de procedimientos y funciones

## 5.1 Procedimiento para inscripción a instancias de examen

### 5.1.1 Análisis y diseño

El algoritmo utilizado para obtener el salón y la silla, ya se encontraba en el código del trigger de la tabla rinde, en el que se tiraban excepciones con los valores que deberían pasarse en el insert/update para poder efectuar la opción sin problemas. Por lo que efectuamos una reingeniería, en la que sacamos a un procedimiento BUSCAR\_SALON\_SILLA dicho algoritmo y retornando el salón y la silla que se le asigna.

En el caso de trigger si el salón y la silla no son los que vienen en “:new”, entonces se tira una excepción, de esta manera mantenemos el comportamiento del trigger y podemos re-usar el código en el procedimiento de inscripción.

El procedimiento de inscripción (INSERT\_RINDE ) recibe los siguientes parámetros:

* Nro. de examen de entrada y obligatorio,
* Nro. de estudiante de entrada y obligatorio
* Nombre de la institución de entrada y Opcional (en caso de no pasarse una institución, el parámetro debe tener el valor NULL, y el procedimiento se ajusta para primero buscar la última institución en que rindió examen, en caso de no encontrarse se retorna error (parámetro status que es de salida en 1)
* Nro. De salón de salida y contiene el nro de salón asignado si no se produjo error.
* Nro. De silla de salida y contiene el nro de silla asignado si no se produjo error.
* Status de salida e indica si la operación fue exitosa (0) o si dio error (1).

El código se consta tres secciones:

1. Búsqueda de salón y silla in inscripciones anteriores (BUSCAR\_ULTIMO\_SALON\_SILLA)
2. Búsqueda de salón y silla libre (BUSCAR\_SALON\_SILLA), la que se encuentra en una sección crítica del procedimiento.
3. finalmente la inserción y commit de la inscripción (en caso que se produzca error, rollback)
4. BUSCAR\_ULTIMO\_SALON\_SILLA

Realiza la búsqueda del último salón y silla en que rindió examen un estudiante, en la institución pasada por parámetro, si no se pasa la institución, se el salón y silla de institución del último examen rendido.

Este procedimiento tiene tres secciones, en las que :

* Se controla la existencia de la instancia de examen, si no existe se tira excepción
* Se busca la institución, numero de salón y numero de silla de la última vez que rindió examen y esos son los valores que se retornan seteando status en 0.

En el coso que no se encuentre resultado (nunca rindió ese examen), se va a producir una excepción NO\_DATA\_FOUND que es capturada y manejada (ver siguiente punto), para encontrar salón y silla usándola institución pasada por parámetro.

* Captura de la excepción NO\_DATA\_FOUND: En el caso que no haya rendido el examen con anterioridad, se asume que es una inscripción nueva y se va a utilizar el parámetro nombre de institución recibido para buscar un salón y una silla cualquiera.

Si el parámetro es null, se tira una excepción NO SE RINDIO EXAMEN ANTES Y NO SE PASA LA INSTITUCION => ERROR.

En Caso que se no sea null, se busca entre las instancias de examen en la institución, un salón y una silla y se retornan esos valores y estatus en 0. Luego en el procedimiento de inscripción se buscará la primera silla libre.

Si no se encuentran resultados, la institución pasada por parámetro no es validad y se tira excepción.

1. Búsqueda de una silla disponible según algoritmo requerimiento.

Para asignar un salón y una silla se invoca al procedimiento BUSCAR\_SALON\_SILLA (que es usado también por el trigger de rinde).

Pero aquí tenemos un problema , que como la búsqueda de un salón y una silla y la inserción en la tabla rinde debe efectuarse de manera atómica, no se debe permitir que se inserte otra instancia de rinde con los mismos valores de silla y salón, y como cada procedimiento se ejecuta en su propia sesión y cada uno tiene una versión de la base de datos, al momento de buscar el salón y la silla puede ocurrir que se asignen a dos estudiantes la misma silla y salón (en sesiones distintas), y que luego como el trigger de inserción se corre sobre la copia de la sesión, la inserción de los datos no falle y cuando finalmente se comitea, se inserten salón y silla de ambas sesiones, queden una silla de un salón ocupada por dos estudiantes repetidos.

Para evitar esto, creamos unas sección critica sobre la instancia de examen, usando select for update sobre la instancia de examen, ejecutando esto antes de buscar el salón y silla nos aseguramos que para una instancia de examen la búsqueda e inserción se haga de a uno por vez (para dicha instancia), con lo que al comitear no van a quedar salones y sillas repetidos para distintos estudiantes.

SELECT 1

INTO tmp

FROM instancia\_ex iex

WHERE

iex.nombre\_institucion = nom\_inst

AND iex.fecha = p\_FECHA

FOR UPDATE;

Se bloquean todas la tuplas de instancia de examen para una institución y una fecha, de esta manera no corro el peligro de que dos o más sesiones estén asignando la misma silla del mismo salón a estudiantes distintos.

Luego de tener exclusividad en la sección de código se invoca el procedimiento de búsqueda de salón y silla, para finalmente insertar y comitear. Retornando El salón y la silla asignados y status= 0.

Al realizar el commit o rollback el lock del select for update se libera y se continua con la ejecución del siguiente llamada al procedimiento de inscripción.

## 5.2 Informe de Exámenes

### 5.2.1 Análisis y diseño

El procedimiento recibe los parámetros:

* Numero de examen, de entrada y obligatorio.
* Fecha de inicio, de entrada y obligatorio.
* Fecha de fin, de entrada y opcional, en caso de no usarlo pasar NULL.

Se iteran ordenadas por fecha ascendente las instancias de examen que cumplan con el rango de fechas pasadas por parámetro (si no se especifico una fecha de fin entonces se toma como fecha de fin la máxima fecha soportada por Oracle).

Para cada una de las instancias se iteran las instituciones donde se realizaron, ordenados por fecha y hora.

Por último, para cada instancia de examen e institucion, se iteran los registros de la tabla rinde y alumno ordenados por el numero de salón y numero de silla, con un corte de control por salón para imprimir los resultados en el orden pedido.

Se utiliza una función auxiliar para convertir la fecha y hora de la instancia del examen a la fecha y hora locales del pais de la institucion donde se está tomando el examen (la fecha convertida solo se imprime si cambia respecto a la original cuando se le aplica la diferencia horaria).

### 5.2.2 Salidas de ejemplo

**Entradas:**

**Numero de Examen**: 142

**Fecha inicio**: '2011/07/01'

**Fecha fin**: NULL (fin de los datos)

-----------------------

- Informe de Examenes -

-----------------------

Examen: FCE - First Certificate (142)

Desde: 01.JUL.2011

Hasta: Fin De Datos

-----------------------

-- 15.AUG.2011

----- Anglo (UY) - 07:00 am

------- Salon: 1

Silla - NroEst - Nombre

------------------------------------------

1 - 1234 - Juan Perez

2 - 2983 - Jose Alvarez

------------------------------------------

------- Salon: 23

Silla - NroEst - Nombre

------------------------------------------

11 - 8945 - Alicia Perez

12 - 9523 - Jose Rodriguez

------------------------------------------

----- British School (CL) - 05:00 am

------- Salon: 123

Silla - NroEst - Nombre

------------------------------------------

8 - 3234 - Antonio Marquez

12 - 3483 - Silvia Jumint

------------------------------------------

-- 25.SEP.2011

----- Metropolitano de Barranquilla (CO) - 04:00 am

------- Salon: 35

Silla - NroEst - Nombre

------------------------------------------

11 - 9999 - Carlos Fernando Navarro Montoya

15 - 8888 - Roberto Abbondanzieri

------------------------------------------

------- Salon: 70

Silla - NroEst - Nombre

------------------------------------------

20 - 6666 - Marco Antonio Etcheverry

21 - 7777 - Jose Luis Felix Chilavert

------------------------------------------

-----------------------

- Fin del Informe -

-----------------------

**Entradas:**

**Numero de Examen**: 142

**Fecha inicio**: '2011/07/01'

**Fecha fin**: '2011/09/24'

-----------------------

- Informe de Examenes -

-----------------------

Examen: FCE - First Certificate (142)

Desde: 01.JUL.2011

Hasta: 24.SEP.2011

-----------------------

-- 15.AUG.2011

----- Anglo (UY) - 07:00 am

------- Salon: 1

Silla - NroEst - Nombre

------------------------------------------

1 - 1234 - Juan Perez

2 - 2983 - Jose Alvarez

------------------------------------------

------- Salon: 23

Silla - NroEst - Nombre

------------------------------------------

11 - 8945 - Alicia Perez

12 - 9523 - Jose Rodriguez

------------------------------------------

----- British School (CL) - 05:00 am

------- Salon: 123

Silla - NroEst - Nombre

------------------------------------------

8 - 3234 - Antonio Marquez

12 - 3483 - Silvia Jumint

------------------------------------------

-----------------------

- Fin del Informe -

-----------------------

## 5.3 Informe de Aprobación

### 5.3.1 Análisis y diseño

El procedimiento recibe los parámetros:

* Numero de examen, de entrada y obligatorio.

Se iteran las instituciones donde se haya rendido el examen que se pasa por parámetro, para cada una de ellas se iteran los registros de rinde y estudiante para poder obtener los datos de los estudiantes que rindieron el examen pedido.

Obteniendo la MAX(fecha) de la tabla rinde que está en la tabla aprueba para el estudiante que se está iterando y el examen buscado se obtiene la fecha de la instancia del examen en que el alumno salvo el examen (si lo salvo).

Luego se iteran todas las rendiciones del estudiante y se va imprimiendo los datos y por último se imprime un texto especial si aprobó el examen o no (si aprobó se muestra la calificación).

Se utiliza una función auxiliar para convertir la fecha y hora de la instancia del examen a la fecha y hora locales del pais de la institucion donde se está tomando el examen (la fecha convertida solo se imprime si cambia respecto a la original cuando se le aplica la diferencia horaria).

### 5.3.2 Salidas de ejemplo

**Entradas:**

**Numero de Examen**: 142

------------------------

- Informe de Aprobados -

------------------------

Examen: FCE - First Certificate (142)

-----------------------

-- Anglo (UY)

-----------------------

1234 - Juan Perez

15.DEC.2009 - 07:00 am

20.DEC.2010 - 07:00 am

15.AUG.2011 - 07:00 am

>>> Aprobado: 25.AUG.2011 Calif: 100

3234 - Antonio Marquez

20.DEC.2010 - 07:00 am

15.AUG.2011 - 07:00 am

>>> Aprobado: 25.AUG.2011 Calif: 80

8945 - Alicia Perez

15.DEC.2009 - 07:00 am

20.DEC.2010 - 07:00 am

>>> NO Aprobado <<<

-- British School (CL)

-----------------------

8888 - Roberto Abbondanzieri

01.JAN.2009 - 06:00 am

01.JAN.2010 - 06:00 am

>>> Aprobado: 01.JAN.2011 Calif: 95

9999 - Carlos Fernando Navarro Montoya

>>> Aprobado: 01.JAN.2011 Calif: 92

-----------------------

- Fin del Informe -

-----------------------

## 5.4 Mejoramiento de calidad

### 5.4.1 Análisis y diseño

Para cumplir con el requerimiento de performance decidimos que la carga de datos en la tabla de calidad sea incremental, y no truncar la tabla y re calcular todo cada vez, por lo que se obtiene del sistema solo las instancias de examen que no se hayan ya ingresado en el sistema de calidad.

Otra consideración importante es si crear un índice o no por lo campos <nro de examen , fecha > en la tabla rinde , ya que eso impacta directamente sobre este procedimiento. El problema que trae esto es el costo al momento de la dada de alta de inscripciones a exámenes, que con volúmenes mundiales de datos, puede ser que le baje la performance al sistema considerablemente, y poniendo en la balanza el uso del sistema vs sacar un reporte de calidad, esta se inclina hacia no tener índice para no impactar en la performance en el uso normal.

Para cargar la tabla de calidad se creó el procedimiento ROCEDURE CARGA\_CALIDAD , que recibe la fecha hasta la que se quiere procesar.

Este procedimiento hace uso de una tabla temporal para almacenar resultados intermedios, que está definida en el DDL y que se limpia automáticamente luego de realizar commit o rollback.

Cuenta con un cursor que retorna la cantidad de inscriptos agrupado por examen y por fecha de examen, que cumpla que tengan una fecha menor a la fecha pasada -7 y que no estén ya dados de alta en la tabla de calidad.

El cuerpo principal del procedimiento consiste en un bucle sobre el cursor, en el que para cada elemento retornado (<examen, fecha, cantidad inscriptos>), se entra en otro bucle donde para cada estudiante que rindió en dicha instancia (<examen, fecha>) , se van sumando en un contador para esa instancia de examen si aprobó (para lograr esto se invoca al procedimientos INCREMENTAR\_SI\_APRUEBA.

Al terminar el bucle sobre estudiante, tenemos contado cuantos aprobaron y tenemos también la cantidad total que rindieron (sale del dato del cursor) en esa fecha. Luego (todavía dentro del bucle del cursos) se insertan estos valores en una tabla temporal (que se limpia automáticamente al comitear o hacer rollback).

Esto se ejecuta para todas las instancias que faltan completar en la tabla calidad.

Finalmente al terminar el bucle sobre el cursor, se realiza un select \* de la tabla temporal into la tabla calidad, y se realiza el commit.

En caso de error se invoca un rollback.

**Procedimiento auxiliar:** incrementar\_si\_aprueba

**Parámetros:**

* Numero de examen, de entrada y obligatorio.
* Numero de estudiante, de entrada y obligatorio.
* Posible fecha de aprobación, de entrada y obligatorio.
* Contador, de entrada-salida y obligatorio.

**Comportamiento:**

Este procedimiento se utiliza en el procedimiento principal del requerimiento de calidad, dado un numero de examen, un numero de estudiante y una posible fecha de aprobación, si el estudiante aprobó el examen en la fecha que se paso por parámetro (que es una de las fechas en que rindió el examen), entones se aumenta el contador pasado por parámetro en una unidad.

El estudiante puede no haber aprobado el examen en la fecha que se paso por parámetro, en ese caso se captura la excepción NO\_DATA\_FOUND para controlar que no es necesario aumentar el contador.

# 7. Código

## 7.1 Procedimiento para inscripción a instancias de examen

CREATE OR REPLACE

PROCEDURE INSERT\_RINDE

(

p\_NRO\_EXAMEN IN NUMBER

, p\_NRO\_ESTUDIANTE IN NUMBER

, p\_NOMBRE\_INSTITUCION IN VARCHAR2

, p\_FECHA IN DATE

, p\_NRO\_SALON OUT NUMBER

, p\_NRO\_SILLA OUT NUMBER

, p\_STATUS OUT NUMBER

) AS

nom\_inst VARCHAR2(50);

nro\_salonFinal NUMBER;

nro\_sillaFinal NUMBER;

estado\_interno NUMBER;

tmp NUMBER;

BEGIN

p\_STATUS := 1;

nom\_inst := p\_NOMBRE\_INSTITUCION;

dbms\_output.put\_line('nom\_inst='||nom\_inst);

-- si no se pasa la institucion y nunca rindio el examen, da error

-- si no se pasa o si se pasa y ya rindio el examen, se retorna la ultima

-- institucion en la que se rindio, la silla y el salon en que rindio.

-- si se pasa la institucion y nunca rindio el examen, retorna la institucion pasada,

-- el primer salon y la primer silla de dicha institucion, para que luego se busque una libre

BUSCAR\_ULTIMO\_SALON\_SILLA(p\_NRO\_EXAMEN, p\_NRO\_ESTUDIANTE, p\_FECHA, nom\_inst, nro\_salonFinal, nro\_sillaFinal, estado\_interno);

-- BLOQUEO los rinde para la institucion seleccionada, para que nadie

-- pueda leer ni escibir datos en rinde y relacionados con la institucion

dbms\_output.put\_line('BLOQUEO FOR UPDATE instancia examen(int, ex, fecha) ='

||nom\_inst||', '||p\_NRO\_EXAMEN||', '||p\_FECHA);

-- Seccion critica para la instancia de examen, para evitar que me

-- llamen al procedimientos mientras estoy buscando un lugar vacio para insertar

SELECT 1

INTO tmp

FROM instancia\_ex iex

WHERE

iex.nombre\_institucion = nom\_inst

AND iex.nro\_examen = p\_NRO\_EXAMEN

AND iex.fecha = p\_FECHA

FOR UPDATE;

BEGIN

dbms\_output.put\_line('BUSCAR\_SALON\_SILLA('|| p\_NOMBRE\_INSTITUCION ||','||

p\_FECHA||','||

nro\_salonFinal||','||

nro\_sillaFinal||')');

BUSCAR\_SALON\_SILLA(

p\_NOMBRE\_INSTITUCION,

p\_FECHA,

nro\_salonFinal,

nro\_sillaFinal

);

INSERT INTO rinde VALUES (p\_NRO\_EXAMEN, p\_NRO\_ESTUDIANTE,

nom\_inst, nro\_salonFinal, p\_FECHA, nro\_sillaFinal);

COMMIT;

dbms\_output.put\_line('hizo commit');

END;

p\_NRO\_SALON := nro\_salonFinal;

p\_NRO\_SILLA := nro\_sillaFinal;

p\_STATUS := 0;

EXCEPTION

WHEN OTHERS THEN

dbms\_output.put\_line( 'CODE - '||SQLCODE||' -ERROR- '||SQLERRM);

ROLLBACK;

p\_STATUS:=1;

END INSERT\_RINDE;

CREATE OR REPLACE

PROCEDURE BUSCAR\_ULTIMO\_SALON\_SILLA(

p\_NRO\_EXAMEN IN NUMBER

, p\_NRO\_ESTUDIANTE IN NUMBER

, p\_FECHA IN DATE

, p\_NOMBRE\_INSTITUCION IN OUT VARCHAR2

, p\_NRO\_SALON OUT NUMBER

, p\_NRO\_SILLA OUT NUMBER

, p\_STATUS OUT NUMBER

) AS

-- si no se pasa la institucion y nunca rindio el examen, da error

-- si no se pasa o si se pasa y ya rindio el examen, se retorna la ultima institucion en la que se rindio, la silla y el salon en que rindio.

-- si se pasa la institucion y nunca rindio el examen, retorna la institucion pasada, el primer salon y la primer silla de dicha institucion, para que luego se busque una libre,

institucion VARCHAR2(50);

cantInstancias NUMBER;

BEGIN

p\_STATUS := -1;

p\_NRO\_SALON := -1;

p\_NRO\_SILLA := -1;

BEGIN --CHEQUEO que exista una instancia de examen para esa fecha, para no seguir trabajando sin necesidad

IF p\_NOMBRE\_INSTITUCION IS NOT NULL THEN

SELECT count(\*)

INTO cantInstancias

FROM instancia\_ex iex

WHERE iex.fecha = p\_FECHA

AND iex.nro\_examen = p\_NRO\_EXAMEN

AND iex.nombre\_institucion = p\_NOMBRE\_INSTITUCION;

IF cantInstancias = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001,'LA INSTANCIA DE EXAMEN NO EXISTE=> ERROR.');

END IF;

END IF;

END;

BEGIN

--en rinde nro\_est, fecha son unique por lo que al fijar una

--fecha y un estudiante se obtiene ninguno o solo un registro

SELECT r.nombre\_institucion, r.nro\_salon, r.nro\_silla\_asignado

INTO institucion, p\_NRO\_SALON, p\_NRO\_SILLA

FROM rinde r

WHERE r.nro\_examen = p\_NRO\_EXAMEN

AND r.nro\_estudiante = p\_NRO\_ESTUDIANTE

AND r.fecha IN

(--busco la ultima fecha en que el estudinte dio el examen

SELECT MAX(fecha)

FROM rinde

WHERE fecha < p\_FECHA

AND nro\_estudiante = p\_NRO\_ESTUDIANTE

AND nro\_examen = p\_NRO\_EXAMEN

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN-- NUNCA RINDIO EL EXAMEN

dbms\_output.put\_line('CODIGO MANEJO EXCEPCION');

IF p\_NOMBRE\_INSTITUCION IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001,

'NO SE RINDIO EXAMEN ANTES Y NO SE PASA LA INSTITUCION => ERROR.');

ELSE

BEGIN

--en caso que se pase la institucion me quedo

--con el primer salon que aparezca (rownum < 2) y obtengo la silla minima

SELECT s.nombre\_institucion, s.nro\_salon, s.nro\_silla\_min

INTO institucion, p\_NRO\_SALON, p\_NRO\_SILLA

FROM instancia\_ex iex, salon s

WHERE iex.fecha = p\_FECHA

AND iex.nro\_examen = p\_NRO\_EXAMEN

AND iex.nombre\_institucion = p\_NOMBRE\_INSTITUCION

AND iex.nombre\_institucion = s.nombre\_institucion

AND rownum < 2;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN -- NO hay salones en la institucion

RAISE\_APPLICATION\_ERROR(-20001,

'NO HAY SALONES EN LA

INSTITUCION o es una

instancia invalida de examen=> ERROR.');

END;

END IF;

END;

p\_NOMBRE\_INSTITUCION := institucion;

p\_STATUS:= 0;

END;

## 7.2 Informe de Exámenes

CREATE OR REPLACE PROCEDURE informe\_examenes (nro\_examen NUMBER, fecha\_inicio DATE, fecha\_fin DATE) AS

var\_descripcion VARCHAR2(200);

var\_nombre\_inst VARCHAR2(50);

var\_nombre\_inst\_ant VARCHAR2(50);

var\_fecha DATE;

var\_hora TIMESTAMP;

var\_pais VARCHAR2(30);

var\_nro\_salon NUMBER;

var\_nro\_salon\_anterior NUMBER;

var\_nro\_silla NUMBER;

var\_nro\_estudiante NUMBER;

var\_nombre\_estudiante VARCHAR2(50);

var\_apellido VARCHAR2(50);

var\_primero NUMBER;

var\_aux\_fecha\_fin DATE;

var\_timezone VARCHAR2(6);

var\_fecha\_local VARCHAR2(11);

var\_hora\_local VARCHAR2(8);

--itera las diferentes instancias de examen para el rango de fechas dado

CURSOR cursor\_instancias\_examenes (par\_fecha\_fin IN DATE) IS

SELECT i.fecha

FROM instancia\_ex i

WHERE i.nro\_examen = nro\_examen

AND i.fecha >= fecha\_inicio

AND i.fecha <= par\_fecha\_fin

GROUP BY fecha

ORDER BY fecha ASC;

--itera los datos de las instancias de examen del primer cursor

CURSOR cursor\_examenes (par\_fecha IN DATE) IS

SELECT i.nombre\_institucion, i.hora, t.pais, t.timezone

FROM instancia\_ex i, institucion t

WHERE i.nro\_examen = nro\_examen

AND i.fecha = par\_fecha

AND t.nombre = i.nombre\_institucion

ORDER BY i.fecha ASC, i.hora DESC;

--itera los datos de la rendicion y alumnos del segundo cursor

CURSOR cursor\_salon\_alumno (par\_nombre\_inst IN VARCHAR2, par\_fecha IN DATE) IS

SELECT r.nro\_salon, r.nro\_silla\_asignado, r.nro\_estudiante, e.nombre, e.apellido

FROM rinde r, estudiante e

WHERE r.nro\_examen = nro\_examen

AND r.nro\_estudiante = e.nro\_estudiante

AND r.fecha = par\_fecha

AND r.nombre\_institucion = par\_nombre\_inst

ORDER BY r.nro\_salon ASC, r.nro\_silla\_asignado ASC;

BEGIN

--imprimo el mensaje inicial

SELECT e.descripcion

INTO var\_descripcion

FROM examen e

WHERE e.nro\_examen = nro\_examen;

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

DBMS\_OUTPUT.PUT\_LINE('- Informe de Examenes -');

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

DBMS\_OUTPUT.PUT\_LINE('Examen: ' || var\_descripcion || ' (' || nro\_examen || ')');

DBMS\_OUTPUT.PUT\_LINE('Desde: ' || TO\_CHAR(fecha\_inicio, 'DD.MON.YYYY'));

IF fecha\_fin IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Hasta: Fin De Datos');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Hasta: ' || TO\_CHAR(fecha\_fin, 'DD.MON.YYYY'));

END IF;

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

IF (fecha\_fin IS NULL)

THEN

--La fecha maxima soportada por Oracle: 31/12/9999

var\_aux\_fecha\_fin := to\_date(5373484, 'J');

ELSE

var\_aux\_fecha\_fin := fecha\_fin;

END IF;

OPEN cursor\_instancias\_examenes (var\_aux\_fecha\_fin);

LOOP

FETCH cursor\_instancias\_examenes INTO var\_fecha;

EXIT WHEN cursor\_instancias\_examenes%NOTFOUND;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('-- ' || TO\_CHAR(var\_fecha, 'DD.MON.YYYY'));

var\_nombre\_inst\_ant := 'xxxxxxxxxxxxxxxxxxxxxx';

OPEN cursor\_examenes(var\_fecha);

LOOP

FETCH cursor\_examenes INTO var\_nombre\_inst, var\_hora, var\_pais, var\_timezone;

EXIT WHEN cursor\_examenes%NOTFOUND;

BEGIN

--convierto la hora y fecha de la instancia de examen a la hora y fecha

--local del pais de la institucion donde se esta tomando el examen

FORMATO\_UTC\_A\_LOCAL(var\_hora, var\_timezone, var\_fecha\_local, var\_hora\_local);

DBMS\_OUTPUT.PUT('----- ' || var\_nombre\_inst || ' (' || var\_pais || ') - ');

IF (TO\_CHAR(var\_fecha, 'DD.MON.YYYY') <> var\_fecha\_local) THEN

DBMS\_OUTPUT.PUT (var\_fecha\_local || ' - ');

END IF;

DBMS\_OUTPUT.PUT\_LINE(var\_hora\_local);

var\_nro\_salon\_anterior := -1;

var\_primero := 1;

OPEN cursor\_salon\_alumno(var\_nombre\_inst, var\_fecha);

LOOP

FETCH cursor\_salon\_alumno INTO var\_nro\_salon, var\_nro\_silla, var\_nro\_estudiante, var\_nombre\_estudiante, var\_apellido;

EXIT WHEN cursor\_salon\_alumno%NOTFOUND;

BEGIN

--solo imprimo el los datos del salon una vez

IF (var\_nro\_salon <> var\_nro\_salon\_anterior)

THEN

IF (var\_primero <> 1)

THEN

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

END IF;

var\_primero := 0;

DBMS\_OUTPUT.PUT\_LINE('------- Salon: ' || var\_nro\_salon);

DBMS\_OUTPUT.PUT\_LINE(' Silla - NroEst - Nombre');

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

END IF;

--imprimo los datos pedidos

DBMS\_OUTPUT.PUT\_LINE(' ' || var\_nro\_silla || ' - '

|| var\_nro\_estudiante || ' - ' || var\_nombre\_estudiante

|| ' ' || var\_apellido);

var\_nro\_salon\_anterior := var\_nro\_salon;

END;

END LOOP;

CLOSE cursor\_salon\_alumno;

--corte de control por institucion para cumplir con el formato pedido

IF (var\_nombre\_inst <> var\_nombre\_inst\_ant)

THEN

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('');

END IF;

var\_nombre\_inst\_ant := var\_nombre\_inst;

END;

END LOOP;

CLOSE cursor\_examenes;

END;

END LOOP;

CLOSE cursor\_instancias\_examenes;

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

DBMS\_OUTPUT.PUT\_LINE('- Fin del Informe -');

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

END informe\_examenes;

## 7.3 Informe de Aprobación

CREATE OR REPLACE PROCEDURE informe\_aprobados (par\_nro\_examen NUMBER) AS

descripcion VARCHAR2(200);

var\_nombre\_institucion VARCHAR2(50);

pais VARCHAR2(30);

nombre VARCHAR2(50);

var\_nro\_estudiante NUMBER;

apellido VARCHAR2(50);

fecha DATE;

var\_ultima\_rendida DATE;

hora TIMESTAMP;

var\_calificacion NUMBER;

var\_timezone VARCHAR2(6);

var\_fecha\_local VARCHAR2(11);

var\_hora\_local VARCHAR2(8);

--itera todas las instituciones donde se rindio el examen que se paso por parametro

CURSOR cursor\_institucion IS

SELECT i.nombre, i.pais, i.timezone

FROM rinde r, institucion i

WHERE r.nro\_examen = par\_nro\_examen

GROUP BY i.nombre, i.pais, i.timezone

ORDER BY i.nombre ASC, i.pais ASC;

--itera los estudiantes que rindieron el exament que se paso por parametro

CURSOR cursor\_rindieron (par\_nombre\_institucion IN VARCHAR2) IS

SELECT DISTINCT r.nro\_estudiante, e.nombre, e.apellido

FROM rinde r, estudiante e

WHERE r.nro\_examen = par\_nro\_examen

AND r.nombre\_institucion = par\_nombre\_institucion

AND r.nro\_estudiante = e.nro\_estudiante

ORDER BY r.nro\_estudiante ASC;

--itera los datos del estudiante, instancia examen y rendicion de los cursores anteriores

CURSOR cursor\_datos\_rindieron (par\_nombre\_institucion IN VARCHAR2, par\_nro\_estudiante IN NUMBER) IS

SELECT DISTINCT i.fecha, i.hora

FROM rinde r, estudiante e, instancia\_ex i

WHERE r.nro\_estudiante = par\_nro\_estudiante

AND r.nro\_estudiante = e.nro\_estudiante

AND r.nombre\_institucion = par\_nombre\_institucion

AND r.nombre\_institucion = i.nombre\_institucion

AND i.nro\_examen = par\_nro\_examen

AND r.fecha = i.fecha

ORDER BY i.fecha ASC;

BEGIN

--imprimo mensaje inicial

SELECT e.descripcion

INTO descripcion

FROM examen e

WHERE e.nro\_examen = par\_nro\_examen;

DBMS\_OUTPUT.PUT\_LINE('------------------------');

DBMS\_OUTPUT.PUT\_LINE('- Informe de Aprobados -');

DBMS\_OUTPUT.PUT\_LINE('------------------------');

DBMS\_OUTPUT.PUT\_LINE('Examen: ' || descripcion || ' (' || par\_nro\_examen || ')');

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

OPEN cursor\_institucion;

LOOP

FETCH cursor\_institucion INTO var\_nombre\_institucion, pais, var\_timezone;

EXIT WHEN cursor\_institucion%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('-- ' || var\_nombre\_institucion || ' (' || pais || ')');

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

OPEN cursor\_rindieron (var\_nombre\_institucion);

LOOP

FETCH cursor\_rindieron INTO var\_nro\_estudiante, nombre, apellido;

EXIT WHEN cursor\_rindieron%NOTFOUND;

--imprimo nombre y apellido del estudiante

DBMS\_OUTPUT.PUT\_LINE(var\_nro\_estudiante || ' - ' || nombre || ' ' || apellido);

BEGIN

--obtengo la fecha de la instancia de examen en que aprobo

--si existe ademas me quedo con la calificacion

SELECT calificacion INTO var\_calificacion

FROM aprueba p

WHERE p.nro\_estudiante = var\_nro\_estudiante

AND p.nro\_examen = par\_nro\_examen;

SELECT MAX(fecha) INTO var\_ultima\_rendida

FROM rinde r

WHERE r.nro\_estudiante = var\_nro\_estudiante

AND r.nro\_examen = par\_nro\_examen

AND r.nombre\_institucion = var\_nombre\_institucion;

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN var\_calificacion := 0;

END;

OPEN cursor\_datos\_rindieron (var\_nombre\_institucion, var\_nro\_estudiante);

LOOP

FETCH cursor\_datos\_rindieron INTO fecha, hora;

EXIT WHEN cursor\_datos\_rindieron%NOTFOUND;

--itero todas las rendiciones del examen pedido del estudiante en la institucion

--imprimo los datos pedidos

IF (fecha = var\_ultima\_rendida AND var\_calificacion > 0) THEN

DBMS\_OUTPUT.PUT\_LINE(' >>> Aprobado: '

|| TO\_CHAR(fecha, 'DD.MON.YYYY') || ' Calif: ' || var\_calificacion);

ELSE

--convierto la fecha y hora de la instancia

--en la fecha y hora locales al pais de la institucion

FORMATO\_UTC\_A\_LOCAL(hora, var\_timezone, var\_fecha\_local, var\_hora\_local);

DBMS\_OUTPUT.PUT(' ' || TO\_CHAR(fecha, 'DD.MON.YYYY') || ' - ');

IF (TO\_CHAR(fecha, 'DD.MON.YYYY') <> var\_fecha\_local) THEN

DBMS\_OUTPUT.PUT (var\_fecha\_local || ' - ');

END IF;

DBMS\_OUTPUT.PUT\_LINE(var\_hora\_local);

END IF;

END LOOP;

CLOSE cursor\_datos\_rindieron;

IF (var\_calificacion = 0) THEN

DBMS\_OUTPUT.PUT\_LINE(' >>> NO Aprobado <<<');

END IF;

END LOOP;

CLOSE cursor\_rindieron;

END LOOP;

CLOSE cursor\_institucion;

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

DBMS\_OUTPUT.PUT\_LINE('- Fin del Informe -');

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

END informe\_aprobados;

## 7.4 Convertir fecha y hora a la fecha y hora locales de una zona horaria

CREATE OR REPLACE PROCEDURE FORMATO\_UTC\_A\_LOCAL

(

P\_I\_FECHA\_HORA IN TIMESTAMP

, P\_I\_TIME\_ZONE IN VARCHAR2

, P\_O\_FECHA\_LOCAL OUT VARCHAR2

, P\_O\_HORA\_LOCAL OUT VARCHAR2

) AS

temp\_timestamp TIMESTAMP;

tmp\_string varchar2(100);

BEGIN

--variables auxiliares intermedias

tmp\_string := TO\_CHAR (P\_I\_FECHA\_HORA) || P\_I\_TIME\_ZONE;

temp\_timestamp := SYS\_EXTRACT\_UTC (tmp\_string);

--variables de salida con el resultado de la conversion

P\_O\_FECHA\_LOCAL := to\_char(temp\_timestamp, 'DD.MON.YYYY');

P\_O\_HORA\_LOCAL := LOWER(to\_char(temp\_timestamp, 'HH:MI PM')) ;

END FORMATO\_UTC\_A\_LOCAL;

## 7.5 Mejoramiento de calidad

### 7.5.1 Procedimiento para cargar tabla calidad

CREATE OR REPLACE

PROCEDURE CARGA\_CALIDAD (p\_fecha IN DATE)

IS

encontre NUMBER;

nro\_examenTMP NUMBER;

fecha\_TMP DATE;

cant\_rendidos\_TMP NUMBER;

cant\_aprobados\_TMP NUMBER;

--Curoso para calcular la cantidad de inscripciones a una instancia de examen,

--que cumpla que no se haya corregido aun y que no esten ya dados de alta en la tabla calidad

CURSOR cursor\_rinde\_not\_in\_calidad (par\_fecha in DATE ) IS

SELECT r.nro\_examen, r.fecha, count(\*)

FROM rinde r

WHERE

(par\_fecha - 7) >r.fecha

AND

NOT EXISTS

(

SELECT 1

FROM calidad cal

WHERE

cal.fecha = r.fecha

AND

cal.nroexamen = r.nro\_examen

)

GROUP BY r.nro\_examen, r.fecha;

BEGIN

OPEN cursor\_rinde\_not\_in\_calidad(p\_fecha);

LOOP

FETCH cursor\_rinde\_not\_in\_calidad

INTO nro\_examentmp, fecha\_tmp, cant\_rendidos\_tmp;

EXIT WHEN cursor\_rinde\_not\_in\_calidad%NOTFOUND;

BEGIN

cant\_aprobados\_TMP:= 0;

for i in (

select distinct(r.nro\_estudiante)

from rinde r

where

r.nro\_examen = nro\_examentmp

and

r.fecha = fecha\_tmp

)

loop

incrementar\_si\_aprueba( nro\_examentmp,i.nro\_estudiante, fecha\_tmp, cant\_aprobados\_TMP);

end loop;

dbms\_output.put\_line('Aprobados = '||cant\_aprobados\_TMP);

INSERT INTO calidad\_temp

VALUES(nro\_examentmp,fecha\_tmp, cant\_rendidos\_tmp,

cant\_aprobados\_tmp, (cant\_rendidos\_tmp - cant\_aprobados\_tmp));

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN

dbms\_output.put\_line('NO DATA FOUND en aprueba para nro\_examen='

||nro\_examentmp||' fecha=' ||fecha\_tmp);

cant\_aprobados\_TMP := 0;

END;

end loop;

close cursor\_rinde\_not\_in\_calidad;

INSERT INTO calidad (NROEXAMEN, FECHA, TOTALALUMNOS, TOTALAPROBADOS,TOTALELIMINADOS)

SELECT NROEXAMEN, FECHA, TOTALALUMNOS, TOTALAPROBADOS,TOTALELIMINADOS

FROM calidad\_temp;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'ERROR DE CALIDAD');

close cursor\_rinde\_not\_in\_calidad;

ROLLBACK ;

END CARGA\_CALIDAD;

### 7.5.2 Procedimiento auxiliar incrementar\_si\_aprueba

CREATE OR REPLACE

PROCEDURE incrementar\_si\_aprueba

(

par\_nro\_examen IN NUMBER,

par\_nro\_estudiante IN NUMBER,

par\_posible\_fecha\_aprobacion IN DATE,

par\_contador IN OUT NUMBER

) AS

var\_fecha DATE;

BEGIN

SELECT MAX(r.fecha)

INTO var\_fecha

FROM rinde r, aprueba a

WHERE r.nro\_estudiante = par\_nro\_estudiante

AND r.nro\_examen = par\_nro\_examen

AND r.nro\_estudiante = a.nro\_estudiante

AND r.nro\_examen = a.nro\_examen;

IF (var\_fecha = par\_posible\_fecha\_aprobacion) THEN

par\_contador := par\_contador + 1;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

--dummy

par\_contador := par\_contador;

END incrementar\_si\_aprueba;