



## INFORME GUÍA APE N°2

### I. PORTADA

Tema: APE 2. Tratamiento de transacciones  
Unidad de Organización Curricular: PROFESIONAL  
Nivel y Paralelo: 5 A  
Alumnos participantes:  
Analuiza Castillo Jimmy Sebastián  
Gordillo Guevara Luis Josué  
Manobanda Chango Ana Patricia  
Peñaloza Narváez Johnny Alexander  
Sistemas de Base de Datos Distribuidos  
Ing. José Caiza  
Asignatura:  
Docente:

### II. INFORME DE GUIA APE

#### 2.1 Objetivos

##### **General:**

Determinar el comportamiento de un SGBD con transacciones.

##### **Específicos:**

#### 2.2 Instrucciones

- Conectar al motor de base de datos.
- Crear una BD llamada Universidad.
- Ingresar datos y verificar la integridad referencial.
- Crear transacciones y probar las características de atomicidad Commit y Rollback.
- Habilitar una nueva sesión para pruebas.
- Manejar errores On\_error, set xact\_abort y Try.

#### 2.3 Listado de materiales

Listado de equipos y materiales generales empleados en la guía práctica:

- Computadora
- Diapositivas
- SQL Server

TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

- Plataformas educativas
- Simuladores y laboratorios virtuales
- Aplicaciones educativas
- Recursos audiovisuales
- Gamificación
- Inteligencia Artificial

#### 2.4 Desarrollo de la actividad

##### **1. Creación de base de datos**

**Paso 1.-** Crear una Base de Datos llamada Universidad en la cual se crean las tablas con sus respectivas relaciones como se muestra en la Fig. 1.

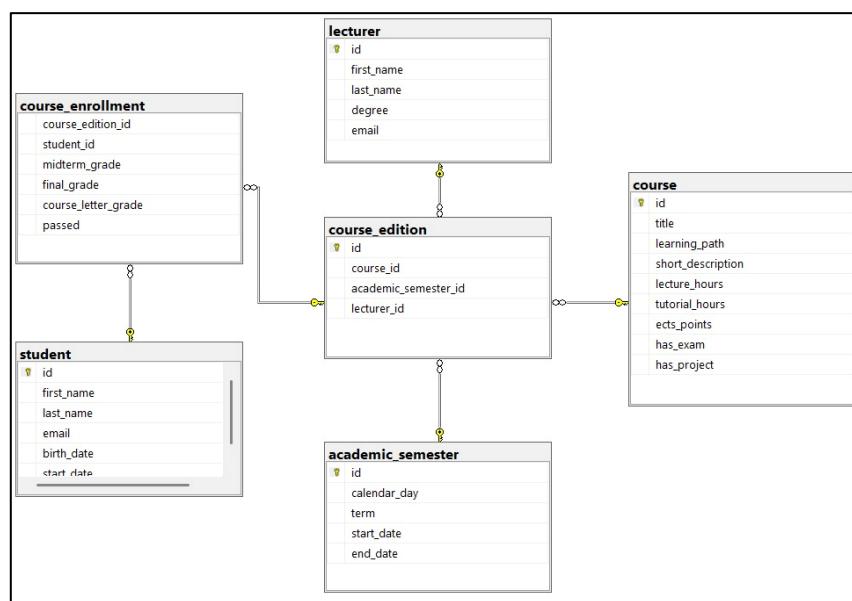


Fig. 1. Modelo Entidad Relación de BD Universidad.

**Paso 2.-** Insertar datos de prueba para poder realizar las diferentes transacciones a lo largo de la guía (Fig. 2).

```
-- =====
-- Datos
-- =====

-- academic_semester
INSERT INTO academic_semester (id, calendar_day, term, start_date, end_date) VALUES
(1, '2023-01-01', 'N'Primavera', '2023-02-01', '2023-06-30'),
(2, '2023-07-01', 'N'Otoño', '2023-08-01', '2023-12-20'),
(3, '2024-01-01', 'N'Primavera', '2024-02-01', '2024-06-30');

-- course
INSERT INTO course (id, title, learning_path, short_description, lecture_hours, tutorial_hours, ects_points, has_exam, has_project) VALUES
(1, 'N'Programación I', 'N'Informática', 'N'Curso introductorio de programación', 40, 20, 5, 1, 1),
(2, 'N'Bases de Datos', 'N'Informática', 'N'Fundamentos de bases de datos relacionales', 35, 15, 5, 1, 0),
(3, 'N'Redes de Computadoras', 'N'Informática', 'N'Conceptos básicos de redes', 30, 10, 4, 0, 1),
(4, 'N'Algoritmos y Estructuras', 'N'Informática', 'N'Curso avanzado de programación', 45, 15, 6, 1, 1),
(5, 'N'Sistemas Operativos', 'N'Informática', 'N'Introducción a sistemas operativos', 40, 20, 5, 1, 0);

-- lecturer
INSERT INTO lecturer (id, first_name, last_name, degree, email) VALUES
(1, 'N'Ana', 'N'Martinez', 'N'PhD', 'N'ana.martinez@example.com'),
(2, 'N'Luis', 'N'Ramirez', 'N'MSc', 'N'luis.ramirez@example.com'),
(3, 'N'Carlos', 'N'Santos', 'N'PhD', 'N'carlos.santos@example.com');
```

Fig. 2. Inserción de Datos de Ejemplo en las tablas de BD Universidad.

## 2. Verificación de Integridad Referencial

**Verificación 1.-** Intentar insertar un registro con una clave foránea que no existe en este caso al querer asignar una edición de curso a un curso que no existe (Fig. 3).

```
36  -- Esto debe FALLAR por violar integridad referencial
37  ✓ INSERT INTO course_edition
38    (id, course_id, academic_semester_id, lecturer_id)
39    VALUES (99, 999, 1, 1);  -- course_id 999 no existe
40

sajes
Mens. 547, Nivel 16, Estado 0, Línea 137
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_course_ed_cours_3F466844".
The statement has been terminated.
```

Fig. 3. Fallo por Integridad Referencial.

**Verificación 2.-** Intentar eliminar un curso que ya está siendo usado en course\_edition (Fig. 4).



```
142 -- Este curso (id = 1) está siendo usado en course_edition
143 DELETE FROM course WHERE id = 1;
144
145 Mens. 547, Nivel 16, Estado 0, Línea 143
146 The DELETE statement conflicted with the REFERENCE constraint "FK_course_ed_cours_3F466844"
147 The statement has been terminated.
```

Fig. 4. Fallo por Integridad Referencial.

**Verificación 3.-** Verificar que no se puede cambiar el valor de una clave primaria (por ejemplo, id de lecturer) si ya está usada como clave foránea en otra tabla (course\_edition) (Fig. 5).

```
145 -- El profesor con id = 1 está en course_edition
146 UPDATE lecturer SET id = 99 WHERE id = 1;
147
148 Mens. 547, Nivel 16, Estado 0, Línea 146
149 The UPDATE statement conflicted with the REFERENCE constraint "FK_course_ed_lectu_412EB0B6".
150 The statement has been terminated.
```

Fig. 5. Fallo por Integridad Relacional.

### 3. Crear transacciones y probar COMMIT/ROLLBACK

Una transacción agrupa operaciones para que todas se ejecuten o ninguna (principio de atomicidad).

**Transacción 1.-** Insertar un registro correspondiente en la tabla course\_enrollment que refleje la matrícula del estudiante en la edición del curso con ID 1(Fig. 6).

```
151 BEGIN TRANSACTION;
152
153 -- Insertamos un nuevo estudiante
154 INSERT INTO student
155 (id, first_name, last_name, email, birth_date, start_date)
156 VALUES
157 (10, 'N'Lucía', 'N'Vera', 'N'lucia.vera@example.com', '2002-05-10', '2024-03-01');
158
159 -- Insertamos una matrícula relacionada
160 INSERT INTO course_enrollment
161 (course_edition_id, student_id, midterm_grade, final_grade, course_letter_grade, passed)
162 VALUES (1, 10, 90.00, 95.00, 'N'A', 1);
163
164 COMMIT TRANSACTION; -- Confirma todos los cambios
165
166
167 Mens. 1, Nivel 16, Estado 0, Línea 162
168 (1 fila afectada)
169 (1 fila afectada)
```

Fig. 6. Inserción de Estudiante y Asignación a una Matrícula.

Se realiza una consulta específica con los datos del estudiante creado y su asignación a una matrícula para poder comprobar que la Transacción se haya realizado correctamente (Fig. 7).



**UNIVERSIDAD TÉCNICA DE AMBATO**  
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL  
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN  
CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026



The screenshot shows a MySQL Workbench interface. The SQL editor contains the following code:

```
166 --- Verificación
167 SELECT * FROM student WHERE id = 10;
168 SELECT * FROM course_enrollment WHERE student_id = 10;
```

The Results tab displays two tables:

	id	first_name	last_name	email	birth_date	start_date
1	10	Lucia	Vera	lucia.vera@example.com	2002-05-10	2024-03-01

	course_edition_id	student_id	midterm_grade	final_grade	course_letter_grade	passed
1	1	10	90.00	95.00	A	1

Fig. 7. Verificación de Estudiante y Asignación de Matrícula.

**Rollback 1.-** Insertar correctamente un nuevo estudiante en la tabla student e intentar insertar una matrícula en la tabla course\_enrollment con una edición de curso no existente (Fig. 8).

The screenshot shows a MySQL Workbench interface. The SQL editor contains the following code:

```
173 BEGIN TRANSACTION;
174
175 -- Insert correcto
176 INSERT INTO student
177 (id, first_name, last_name, email, birth_date, start_date)
178 VALUES
179 (11, 'Roberto', 'Mejia', 'roberto.mejia@example.com', '2001-02-02', '2024-03-01');
180
181 -- Esto fallará porque el course_edition_id no existe
182 INSERT INTO course_enrollment
183 (course_edition_id, student_id, midterm_grade, final_grade, course_letter_grade, passed)
184 VALUES (999, 11, 80.00, 85.00, 'N/B', 1);
185
186 ROLLBACK TRANSACTION; -- Revierte todo
187
```

The Results tab shows an error message:

(1 fila afectada)  
Mens. 547, Nivel 16, Estado 0, Línea 182  
The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_course\_en\_cours\_4316F928". The conflict occurred in database "Universidad\_Tecnica\_de\_Ambato".  
The statement has been terminated.

Fig. 8. Inserción de Estudiante y Asignación de Matrícula fallida.

Se realiza una consulta específica con los datos del estudiante creado para poder comprobar que el Rollback se haya realizado correctamente. El registro se revirtió, garantizando que la atomicidad ayuda a que ningún cambio parcial quedó en la base. (Fig. 9).

The screenshot shows a MySQL Workbench interface. The SQL editor contains the following code:

```
186 --- Verificación ROLLBACK
187
188 SELECT * FROM student WHERE id = 11; -- No debe aparecer
189
```

The Results tab shows an empty table:

	id	first_name	last_name	email	birth_date	start_date
--	----	------------	-----------	-------	------------	------------

Fig. 9. Verificación de Rollback exitoso al no aparecer ningún Estudiante con id 11.

**Transacción 2.-** Insertar correctamente un nuevo profesor y una nueva edición de curso asociada (Fig. 10).



**UNIVERSIDAD TÉCNICA DE AMBATO**  
**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL**  
**CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN**  
**CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026**



```
92 --- EJEMPLO 2
93 BEGIN TRANSACTION;
94
95     -- 1º Insertar un nuevo profesor
96     INSERT INTO lecturer (id, first_name, last_name, degree, email)
97     VALUES (10, 'N'Sofia', 'N'García', 'N'MSc', 'N'sofia.garcia@example.com');
98
99     -- 2º Insertar una edición de curso asignada a la profesora Sofia
100    INSERT INTO course_edition (id, course_id, academic_semester_id, lecturer_id)
101    VALUES (100, 2, 2, 10); -- course_id=2, semester_id=2, lecturer_id=10 (válidos)
102
103    COMMIT TRANSACTION;
104
105    PRINT 'Transacción completada con éxito.';
```

(1 fila afectada)

(1 fila afectada)

Transacción completada con éxito.

Fig. 10. Inserción de Profesor y Asignación a una Edición del curso.

Se realiza una consulta específica con los datos del profesor creado y su asignación a una edición del curso para poder comprobar que la Transacción se haya realizado correctamente (Fig. 11).

```
207 --- VERIFICACION
208
209
210    SELECT * FROM lecturer WHERE id = 10;
211    SELECT * FROM course_edition WHERE id = 100;
```

	id	first_name	last_name	degree	email
1	10	Sofia	García	MSc	sofia.garcia@example.com

	id	course_id	academic_semester_id	lecturer_id
1	100	2	2	10

Fig. 11. Verificación de Profesor creado y Asignación a una Edición del curso.

**Rollback 2.-** Intentar insertar una edición de curso con un lecturer\_id (profesor) inexistente (Fig. 12).

```
12 BEGIN TRANSACTION;
13
14     -- Insertamos un nuevo curso
15     INSERT INTO course
16     (id, title, learning_path, short_description, lecture_hours, tutorial_hours, ects_points, has_exam, has_project)
17     VALUES
18     (11, 'N'Computación Gráfica', 'N'Informática', 'N'Introducción al modelado y gráficos por computadora', 40, 15, 5, 1, 1);
19
20     -- Intentamos asignarle un profesor que no existe (id = 999)
21     INSERT INTO course_edition (id, course_id, academic_semester_id, lecturer_id)
22     VALUES (101, 11, 1, 999); -- Error, no existe el profesor 999
23
24     ROLLBACK TRANSACTION;
25
26     PRINT 'Transacción revertida debido a error.';
```

(1 fila afectada)

Mens. 547, Nivel 16, Estado 0, Líneas 221  
The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_\_course\_ed\_\_lecto\_\_412EB0B6". The conflict occurred in database "UniversidadTAE", table "dbo.lecturer", column 'id'.  
The statement has been terminated.  
Transacción revertida debido a error.

Fig. 12. Insertar un Curso y Asignar un Profesor inexistente.

Se realiza una consulta específica con los datos del curso creado y de la asignación de una edición de curso para poder comprobar que el Rollback se haya realizado correctamente. Ambos registros se revirtieron, garantizando que la atomicidad ayuda a que ningún cambio parcial quedó en la base. (Fig. 13).



228 --- VERIFICACION ROLLBACK 2  
229 SELECT \* FROM course WHERE id = 11; -- No debe aparecer  
230 SELECT \* FROM course\_edition WHERE id = 101; -- Tampoco debe existir  
231

Resultados Mensajes

id	title	learning_path	short_description	lecture_hours	tutorial_hours	ects_points	has_exam	has_project

id	course_id	academic_semester_id	lecturer_id

Fig. 13. Verificación de Rollback exitoso al no aparecer ninguno de los registros.

#### 4. Habilitar una nueva sesión para pruebas

**Paso 1.-** Abrir dos ventanas de consulta (sesión 1 y sesión 2), la Ventana 1, será Sesión A (Fig. 14) y la Ventana 2, será Sesión B (Fig. 15). Verificar el id de cada una.

```
SELECT @@SPID AS 'ID de Sesión Actual';
```

SQLQuery3.s...R\User (66)\* -> X SQLQuery2.sql...SR\User (69)\* DESKTO  
1 --- Habilitar una nueva sesión para pruebas  
2  
3 USE UniversidadAPE;  
4 GO  
5  
6 ✓ SELECT @@SPID AS 'ID de Sesión Actual';  
109 % ✓ No se encontraron problemas.

Resultados Mensajes

ID de Sesión Actual
1 66

Fig. 14. Sesión A y Verificación de su id.

232  
233 SELECT @@SPID AS 'ID de Sesión Actual';  
234  
109 % ✓ 12 0 ↑ ↓  
Resultados Mensajes

ID de Sesión Actual
1 69

Fig. 15. Sesión B y Verificación de su id.

**Paso 2.-** En la Sesión A (Transacción abierta), ejecutar la siguiente transacción sin hacer COMMIT todavía (Fig. 16).

```
8 BEGIN TRANSACTION;  
9  
10 ✓ INSERT INTO student (id, first_name, last_name, email, birth_date, start_date)  
11 VALUES (20, N'Pedro', N'Flores', N'pedro.flores@example.com', '2001-05-10', '2025-01-01');  
12  
13 --- No hacer COMMIT todavía  
14  
No se encontraron problemas.
```

(1 fila afectada)

Hora de finalización: 2025-10-06T15:13:31.9884940-05:00

Fig. 16. Transacción efectuada sin Commit.



**Paso 3.-** En la Sesión B (otra ventana de consulta), probar si se puede ver el registro (Fig. 17).

**SELECT \* FROM student WHERE id = 20;**

```
-- 234 | SELECT * FROM student WHERE id = 20;
235 |
09 % ① 12 ▲ 0 ↑ ↓
Resultados Mensajes
```

Fig. 17. Verificación en Sesión B de la Transacción realizada.

No se verá el registro todavía, porque la transacción de la sesión A no ha hecho COMMIT. Esto demuestra el aislamiento de transacciones (cada sesión ve solo los datos confirmados).

**Paso 4.-** Ahora vuelvov a la Sesión A y ejecutar el siguiente comando.

**COMMIT TRANSACTION;**

**Paso 5.-** En la Sesión B (otra ventana de consulta), probar si se puede ver el registro (Fig 18), Ahora sí aparecerá el registro, porque ya se confirmó en la otra sesión..

```
-- 234 | SELECT * FROM student WHERE ID = 20;
90 % ① 12 ▲ 0 ↑ ↓
Resultados Mensajes
id first_name last_name email birth_date start_date
1 20 Pedro Flores pedro.flores@example.com 2001-05-10 2025-01-01
```

Fig. 18. Verificación de Transacción reflejada en Sesión B.

## 5. Manejo de errores ON\_ERROR, SET XACT\_ABORT, TRY...CATCH

**Opción 1.- SET XACT\_ABORT ON:** Si ocurre un error, SQL Server haga automáticamente ROLLBACK.

```
--Opción 1: SET XACT_ABORT ON
SET XACT_ABORT ON;

BEGIN TRANSACTION;
    INSERT INTO student (id, first_name, last_name, email, birth_date, start_date)
    VALUES (12, N'Laura', N'Cedeño', N'laura.cedeno@example.com', '2000-07-22', '2024-03-01');

    -- Esta insert fallará
    INSERT INTO course_enrollment (course_edition_id, student_id, midterm_grade, final_grade, course_letter_grade, passed)
    VALUES (888, 12, 70.00, 75.00, N'C', 1);
COMMIT TRANSACTION;
```

(1 fila afectada)
Estado: 347, Nivel 16, Estado 0, Línea 280
The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_course\_en\_\_course\_4916F928". The conflict occurred in database "UniversidadSAFE", table "dbo.course\_edition", column

Fig. 19. Transacción abortada debido a que no existe esa Edición de Curso.

Se realiza una consulta específica con los datos del estudiante creado y de la asignación de una Matrícula para poder comprobar que el XACT\_ABORT se haya realizado correctamente (Fig. 20).



**UNIVERSIDAD TÉCNICA DE AMBATO**  
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL  
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN  
CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026



```
253  
254 |   SELECT * FROM STUDENT WHERE id = 12;  
255 |   SELECT * FROM course_enrollment WHERE student_id = 12;  
  
0 %  6  0  ↑ ↓  Resultados Mensajes  
| id | first_name | last_name | email | birth_date | start_date |  
  
| course_edition_id | student_id | midterm_grade | final_grade | course_letter_grade | passed |
```

Fig. 20. Verificación de auto Rollback por XACT\_ABORT.

**Opción 2.- TRY...CATCH (manejo controlado):** Permite capturar el error y hacer rollback manual (Fig. 21).

```
258 --- Opción 2: TRY...CATCH (manejo controlado)  
259  
260 BEGIN TRY  
261     BEGIN TRANSACTION;  
262  
263         INSERT INTO student (id, first_name, last_name, email, birth_date, start_date)  
264             VALUES (13, 'N'Andrés', 'N'Vera', 'N'andres.vera@example.com', '2001-04-15', '2024-03-01');  
265  
266         -- Error intencional: referencia inexistente  
267         INSERT INTO course_enrollment (course_edition_id, student_id, midterm_grade, final_grade, course_letter_grade, passed)  
268             VALUES (999, 13, 90.00, 95.00, N'A', 1);  
269  
270         COMMIT TRANSACTION;  
271     END TRY  
272     BEGIN CATCH  
273         PRINT 'Error detectado, ejecutando ROLLBACK...';  
274         ROLLBACK TRANSACTION;  
275  
276         PRINT 'Mensaje de error: ' + ERROR_MESSAGE();  
277     END CATCH;  
278  
0 6 0  ↑ ↓  Resultados Mensajes  
1 fila afectada  
0 filas afectadas  
Error detectado, ejecutando ROLLBACK...  
Mensaje de error: The INSERT statement conflicted with the FOREIGN KEY constraint "FK_course_enrollment". The conflict occurred in database "UniversidadADE", table "dbo.course_enrollment", column 'id'.  
Hora de finalización: 2025-10-04T15:37:05.7687842-05:00
```

Fig. 21. Transacción detenida debido al error detectado en Edición de Curso inexistente.

Se realiza una consulta específica con los datos del estudiante creado y de la asignación de una Matrícula para poder comprobar que el TRY...CATCH se haya realizado correctamente (Fig. 22).

```
279  
280 |   SELECT * FROM student WHERE id = 13; -- No debe estar insertado  
281 |  
6  0  0  ↑ ↓  Resultados Mensajes  
| id | first_name | last_name | email | birth_date | start_date |
```

Fig. 22. Verificación de Rollback por Try...Catch.

**Opción 3.- ON\_ERROR:** SQL Server no tiene una instrucción directa llamada ON\_ERROR como Oracle. Pero podemos simular su comportamiento usando BEGIN TRAN + ROLLBACK en triggers o transacciones, y manejando el error con ERROR\_NUMBER() / ERROR\_MESSAGE().

## 2.5 Resultados obtenidos

A lo largo de la práctica se logró:

- Crear y configurar correctamente la base de datos Universidad, incluyendo sus tablas y relaciones según el modelo entidad-relación propuesto.



- Verificar la integridad referencial mediante intentos de inserción y modificación de registros con claves foráneas inválidas, comprobando que el sistema rechaza estas operaciones para mantener la consistencia de los datos.
- Implementar transacciones que demostraron el principio de atomicidad, asegurando que todas las operaciones dentro de una transacción se ejecuten completamente o ninguna en caso de error.
- Utilizar COMMIT para confirmar cambios y ROLLBACK para revertirlos en caso de fallos, garantizando que no se guarden estados inconsistentes en la base de datos.
- Realizar pruebas de aislamiento entre sesiones, donde se evidenció que una transacción no confirmada no es visible para otras sesiones hasta que se realiza el COMMIT.
- Aplicar diferentes estrategias de manejo de errores:
  - SET XACT\_ABORT ON para rollback automático.
  - TRY...CATCH para control manual de errores y rollback explícito.
  - Simulación de un comportamiento similar a ON\_ERROR mediante el uso de funciones de error de SQL Server.

Todos los ejercicios se ejecutaron de manera exitosa, validando tanto el comportamiento transaccional del SGBD como la correcta aplicación de los conceptos de atomicidad, consistencia, aislamiento y durabilidad (propiedades ACID).

## 2.6 Habilidades blandas

- Liderazgo
- Trabajo en equipo
- Comunicación assertiva
- La empatía
- Pensamiento crítico
- Flexibilidad
- La resolución de conflictos
- Adaptabilidad
- Responsabilidad

## 2.7 Conclusiones

- Las transacciones en SQL Server son esenciales para mantener la consistencia y atomicidad de las operaciones sobre la base de datos, especialmente en entornos donde múltiples usuarios acceden concurrentemente a los datos.
- La integridad referencial es fundamental para evitar inconsistencias en los datos, y el SGBD la garantiza mediante restricciones que impiden operaciones inválidas.
- El uso de COMMIT y ROLLBACK permite controlar explícitamente la persistencia de los cambios, lo que es crucial en operaciones complejas o críticas.
- El manejo de errores mediante XACT\_ABORT y TRY...CATCH ofrece flexibilidad para decidir cómo responder ante fallos, ya sea de forma automática o controlada.

## 2.8 Referencia Bibliográfica

- [1] hdeleon.net, ⚡ La magia de las transacciones SQL | Ejemplo en Sql Server, (el 21 de agosto de 2020). Consultado: el 3 de octubre de 2025. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=keL9-EtE-zE>



- [2] Microsoft Learn, “BEGIN TRANSACTION (Transact-SQL) - SQL Server”. Consultado: el 3 de octubre de 2025. [En línea]. Disponible en: <https://learn.microsoft.com/es-es/sql/t-sql/language-elements/begin-transaction-transact-sql?view=sql-server-ver17>
- [3] Microsoft Learn, “ROLLBACK TRANSACTION (Transact-SQL) - SQL Server”. Consultado: el 3 de octubre de 2025. [En línea]. Disponible en: <https://learn.microsoft.com/es-es/sql/t-sql/language-elements/rollback-transaction-transact-sql?view=sql-server-ver17>

## 2.9 Anexos

### Script SQL

```
-- =====
-- Base de datos: Universidad
-- =====

CREATE DATABASE UniversidadAPE;
GO

USE UniversidadAPE;
GO

-- =====
-- Tablas
-- =====

CREATE TABLE academic_semester (
    id INT NOT NULL PRIMARY KEY,
    calendar_day DATE NOT NULL,
    term NVARCHAR(128) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL
);

CREATE TABLE course (
    id INT NOT NULL PRIMARY KEY,
    title NVARCHAR(128) NOT NULL,
    learning_path NVARCHAR(128) NOT NULL,
    short_description NVARCHAR(1200) NOT NULL,
    lecture_hours INT NOT NULL,
    tutorial_hours INT NOT NULL,
    ects_points INT NOT NULL,
    has_exam BIT NOT NULL,
    has_project BIT NOT NULL
);

CREATE TABLE lecturer (
    id INT NOT NULL PRIMARY KEY,
    first_name NVARCHAR(128) NOT NULL,
    last_name NVARCHAR(128) NOT NULL,
    degree NVARCHAR(32),
    email NVARCHAR(128) NOT NULL
);

CREATE TABLE student (
    id INT NOT NULL PRIMARY KEY,
    first_name NVARCHAR(128) NOT NULL,
    last_name NVARCHAR(128) NOT NULL,
    email NVARCHAR(128) NOT NULL,
```



```
birth_date DATE NOT NULL,
start_date DATE NOT NULL
);

CREATE TABLE course_edition (
    id INT NOT NULL PRIMARY KEY,
    course_id INT NOT NULL,
    academic_semester_id INT NOT NULL,
    lecturer_id INT NOT NULL,
    FOREIGN KEY (course_id) REFERENCES course(id),
    FOREIGN KEY (academic_semester_id) REFERENCES academic_semester(id),
    FOREIGN KEY (lecturer_id) REFERENCES lecturer(id)
);

CREATE TABLE course_enrollment (
    course_edition_id INT NOT NULL,
    student_id INT NOT NULL,
    midterm_grade DECIMAL(5,2),
    final_grade DECIMAL(5,2),
    course_letter_grade NVARCHAR(3),
    passed BIT,
    FOREIGN KEY (course_edition_id) REFERENCES course_edition(id),
    FOREIGN KEY (student_id) REFERENCES student(id)
);

-- =====
-- Datos
-- =====

-- academic_semester
INSERT INTO academic_semester (id, calendar_day, term, start_date, end_date) VALUES
(1, '2023-01-01', N'Primavera', '2023-02-01', '2023-06-30'),
(2, '2023-07-01', N'Otoño', '2023-08-01', '2023-12-20'),
(3, '2024-01-01', N'Primavera', '2024-02-01', '2024-06-30');

-- course
INSERT INTO course (id, title, learning_path, short_description, lecture_hours, tutorial_hours, ects_points, has_exam, has_project) VALUES
(1, N'Programación I', N'Informática', N'Curso introductorio de programación', 40, 20, 5, 1, 1),
(2, N'Bases de Datos', N'Informática', N'Fundamentos de bases de datos relacionales', 35, 15, 5, 1, 0),
(3, N'Redes de Computadoras', N'Informática', N'Conceptos básicos de redes', 30, 10, 4, 0, 1),
(4, N'Algoritmos y Estructuras', N'Informática', N'Curso avanzado de programación', 45, 15, 6, 1, 1),
(5, N'Sistemas Operativos', N'Informática', N'Introducción a sistemas operativos', 40, 20, 5, 1, 0);

-- lecturer
INSERT INTO lecturer (id, first_name, last_name, degree, email) VALUES
(1, N'Ana', N'Martínez', N'PhD', N'ana.martinez@example.com'),
(2, N'Luis', N'Ramírez', N'MSc', N'luis.ramirez@example.com'),
(3, N'Carlos', N'Santos', N'PhD', N'carlos.santos@example.com');

-- student
INSERT INTO student (id, first_name, last_name, email, birth_date, start_date) VALUES
```



**UNIVERSIDAD TÉCNICA DE AMBATO**  
**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL**  
**CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN**  
**CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026**



```
(1, N'Juan', N'Pérez', N'juan.perez@example.com', '2000-03-15', '2022-08-01'),  
(2, N'María', N'Gómez', N'maria.gomez@example.com', '1999-11-20', '2022-08-01'),  
(3, N'Carlos', N'López', N'carlos.lopez@example.com', '2001-06-05', '2023-01-15'),  
(4, N'Ana', N'Torres', N'ana.torres@example.com', '2000-09-10', '2022-08-01'),  
(5, N'Luis', N'Ramírez', N'luis.ramirez@student.com', '1998-12-05', '2023-01-15');  
  
-- course_edition  
INSERT INTO course_edition (id, course_id, academic_semester_id, lecturer_id) VALUES  
(1, 1, 1, 1),  
(2, 2, 1, 2),  
(3, 3, 2, 2),  
(4, 4, 3, 1),  
(5, 5, 3, 3),  
(6, 1, 2, 3),  
(7, 2, 3, 2);  
  
-- course_enrollment  
INSERT INTO course_enrollment (course_edition_id, student_id, midterm_grade, final_grade, course_letter_grade, passed) VALUES  
(1, 1, 85.00, 90.00, N'A', 1),  
(1, 2, 70.00, 75.00, N'B', 1),  
(1, 3, 50.00, 55.00, N'F', 0),  
(2, 1, 60.00, 65.00, N'C', 1),  
(2, 3, 80.00, 85.00, N'A', 1),  
(2, 4, 55.00, 60.00, N'D', 1),  
(3, 5, 75.00, 80.00, N'B', 1),  
(3, 2, 40.00, 50.00, N'F', 0),  
(4, 1, 88.00, 92.00, N'A', 1),  
(4, 2, 78.00, 80.00, N'B', 1),  
(5, 3, 70.00, 75.00, N'B', 1),  
(6, 4, 50.00, 55.00, N'F', 0),  
(7, 1, 95.00, 98.00, N'A', 1),  
(7, 2, 85.00, 87.00, N'A', 1);  
  
--- Verificar integridad referencial  
  
-- Esto debe FALLAR por violar integridad referencial  
INSERT INTO course_edition  
(id, course_id, academic_semester_id, lecturer_id)  
VALUES (99, 999, 1, 1); -- course_id 999 no existe  
  
-- Este curso (id = 1) está siendo usado en course_edition  
DELETE FROM course WHERE id = 1;  
  
-- El profesor con id = 1 está en course_edition  
UPDATE lecturer SET id = 99 WHERE id = 1;  
  
---- Crear transacciones y probar COMMIT / ROLLBACK (atomicidad)  
  
--- EJEMPLO 1  
BEGIN TRANSACTION;  
  
-- Insertamos un nuevo estudiante
```



**UNIVERSIDAD TÉCNICA DE AMBATO**  
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL  
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN  
CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026



```
INSERT INTO student
(id, first_name, last_name, email, birth_date, start_date)
VALUES
(10, N'Lucía', N'Vera', N'lucia.vera@example.com', '2002-05-10',
'2024-03-01');

-- Insertamos una matrícula relacionada
INSERT INTO course_enrollment
(course_edition_id, student_id, midterm_grade, final_grade,
course_letter_grade, passed)
VALUES (1, 10, 90.00, 95.00, N'A', 1);

COMMIT TRANSACTION; -- Confirma todos los cambios

--- Verificación
SELECT * FROM student WHERE id = 10;
SELECT * FROM course_enrollment WHERE student_id = 10;

BEGIN TRANSACTION;

-- Insert correcto
INSERT INTO student
(id, first_name, last_name, email, birth_date, start_date)
VALUES
(11, N'Roberto', N'Mejía', N'roberto.mejia@example.com', '2001-02-
02', '2024-03-01');

-- Esto fallará porque el course_edition_id no existe
INSERT INTO course_enrollment
(course_edition_id, student_id, midterm_grade, final_grade,
course_letter_grade, passed)
VALUES (999, 11, 80.00, 85.00, N'B', 1);

ROLLBACK TRANSACTION; -- Revierte todo

--- Verificación ROLLBACK
SELECT * FROM student WHERE id = 11; -- No debe aparecer

-----
-----
--- EJEMPLO 2
BEGIN TRANSACTION;

-- Insertar un nuevo profesor
INSERT INTO lecturer (id, first_name, last_name, degree, email)
VALUES (10, N'Sofía', N'García', N'MSc',
N'sofia.garcia@example.com');

-- Insertar una edición de curso asignada a la profesora Sofía
INSERT INTO course_edition (id, course_id, academic_semester_id,
lecturer_id)
VALUES (100, 2, 2, 10); -- course_id=2, semester_id=2,
lecturer_id=10 (válidos)

COMMIT TRANSACTION;

PRINT 'Transacción completada con éxito.';

--- VERIFICACION TRANSACCION 2
```



**UNIVERSIDAD TÉCNICA DE AMBATO**  
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL  
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN  
CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026



```
SELECT * FROM lecturer WHERE id = 10;
SELECT * FROM course_edition WHERE id = 100;

BEGIN TRANSACTION;

    -- Insertamos un nuevo curso
    INSERT INTO course
        (id, title, learning_path, short_description, lecture_hours,
        tutorial_hours, ects_points, has_exam, has_project)
    VALUES
        (11, N'Computación Gráfica', N'Informática', N'Introducción al
        modelado y gráficos por computadora', 40, 15, 5, 1, 1);

    -- Intentamos asignarle un profesor que no existe (id = 999)
    INSERT INTO course_edition (id, course_id, academic_semester_id,
    lecturer_id)
    VALUES (101, 11, 1, 999); -- Error, no existe el profesor 999

ROLLBACK TRANSACTION;

PRINT 'Transacción revertida debido a error.';

--- VERIFICACION ROLLBACK 2
SELECT * FROM course WHERE id = 11; -- No debe aparecer
SELECT * FROM course_edition WHERE id = 101; -- Tampoco debe existir

---- Habilitar una nueva sesión para pruebas
SELECT @@SPID AS 'ID de Sesión Actual';
SELECT * FROM student WHERE ID = 20;

--- Manejo de errores: ON_ERROR, SET XACT_ABORT, TRY...CATCH

---Opción 1: SET XACT_ABORT ON

SET XACT_ABORT ON;

BEGIN TRANSACTION;
    INSERT INTO student (id, first_name, last_name, email, birth_date,
    start_date)
    VALUES (12, N'Laura', N'Cedeño', N'laura.cedeno@example.com', '2000-
    07-22', '2024-03-01');

    -- Este insert fallará
    INSERT INTO course_enrollment (course_edition_id, student_id,
    midterm_grade, final_grade, course_letter_grade, passed)
    VALUES (888, 12, 70.00, 75.00, N'C', 1);
COMMIT TRANSACTION;

SELECT * FROM STUDENT WHERE id = 12;
SELECT * FROM course_enrollment WHERE student_id = 12;

--- Opción 2: TRY...CATCH (manejo controlado)

BEGIN TRY
    BEGIN TRANSACTION;

        INSERT INTO student (id, first_name, last_name, email,
        birth_date, start_date)
```



**UNIVERSIDAD TÉCNICA DE AMBATO**  
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL  
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN  
CICLO ACADÉMICO: AGOSTO 2025 – ENERO 2026



```
VALUES (13, N'Andrés', N'Vera', N'andres.vera@example.com',
'2001-04-15', '2024-03-01');

-- Error intencional: referencia inexistente
INSERT INTO course_enrollment (course_edition_id, student_id,
midterm_grade, final_grade, course_letter_grade, passed)
VALUES (999, 13, 90.00, 95.00, N'A', 1);

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    PRINT 'Error detectado, ejecutando ROLLBACK...';
    ROLLBACK TRANSACTION;

    PRINT 'Mensaje de error: ' + ERROR_MESSAGE();
END CATCH;

SELECT * FROM student WHERE id = 13; -- No debe estar insertado

--- OPCION 3. ON_ERROR
BEGIN TRANSACTION;
BEGIN
    -- Insert válido
    INSERT INTO student (id, first_name, last_name, email, birth_date,
start_date)
    VALUES (30, N'Paula', N'Mendoza', N'paula.mendoza@example.com',
'2001-08-20', '2025-01-01');

    -- Insert que rompe la FK (curso edición 999 no existe)
    INSERT INTO course_enrollment (course_edition_id, student_id,
midterm_grade, final_grade, course_letter_grade, passed)
    VALUES (999, 30, 85.00, 90.00, N'A', 1);

    -- Si todo sale bien
    COMMIT TRANSACTION;
END
-- Simulando ON_ERROR
IF @@ERROR <> 0
BEGIN
    PRINT 'Error detectado, ejecutando rollback...';
    ROLLBACK TRANSACTION;
END
```