



INFORME DE GUÍA PRÁCTICA

I. PORTADA

Tema:	APE 2. Tratamiento de transacciones
Unidad de Organización Curricular:	PROFESIONAL
Nivel y Paralelo:	5 ^{to} – ‘A’
Alumnos participantes:	Ases Tiban Jeremy Damián Palate Moreta Kevin Damián Poveda Gomez William Alberto Pullupaxi Chango Daniel
Asignatura:	Sistemas de Base de Datos Distribuidas
Docente:	Ing. José Caiza

II. INFORME DE GUÍA PRÁCTICA

2.1 Objetivos

General:

Determinar el comportamiento de un SGBD con transacciones.

Específicos:

- Analizar el funcionamiento de las transacciones en SQL Server mediante el uso de los comandos COMMIT y ROLLBACK, garantizando la integridad de los datos.
- Implementar mecanismos de control de errores utilizando las sentencias SET XACT_ABORT ON y TRY...CATCH para asegurar la consistencia de la base de datos ante fallos durante una transacción.
- Comprobar la atomicidad y aislamiento de las transacciones ejecutadas en diferentes sesiones de SQL Server, evaluando su impacto en la lectura y escritura simultánea de datos.

2.2 Modalidad

Presencial.

2.3 Tiempo de duración

Presenciales: 6 horas.

No presenciales: N/A.

2.4 Instrucciones

- Conéctese al motor de base de datos
- Cree una BD llamada Universidad
- Cree las tablas A(a char(1) PK, B(b char(1) referenciada a A, C(c char(1)
- Ingrese algunos datos y verifiquemos la integridad referencial
- Creamos transacciones y probamos las características de atomicidad Commit y Rollback
- Habilitamos una nueva sesión para pruebas
- Manejamos errores On_error, set xact_abort y Try

2.5 Listado de equipos, materiales y recursos

- Computadora.
- 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
- Otros (Especifique): _____

2.6 Actividades desarrolladas

Primero conectarse al motor de base de datos en este caso será SQL Server.

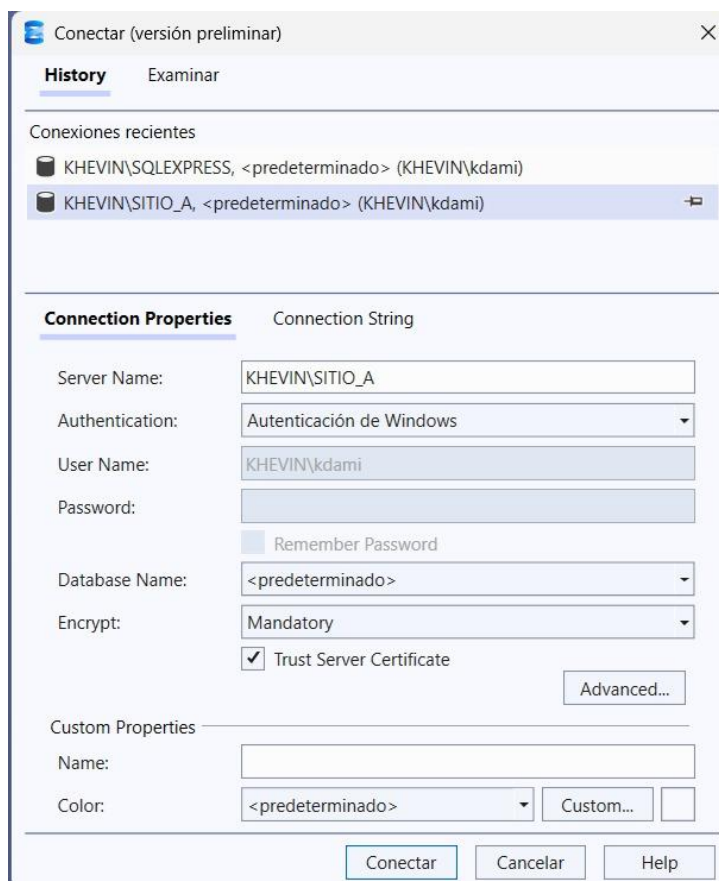


Fig. 1 Conectarnos al motor de base de datos

Para realizar la práctica crear una Base de Datos llamada “Universidad”, tal como se muestra la fig. 2.

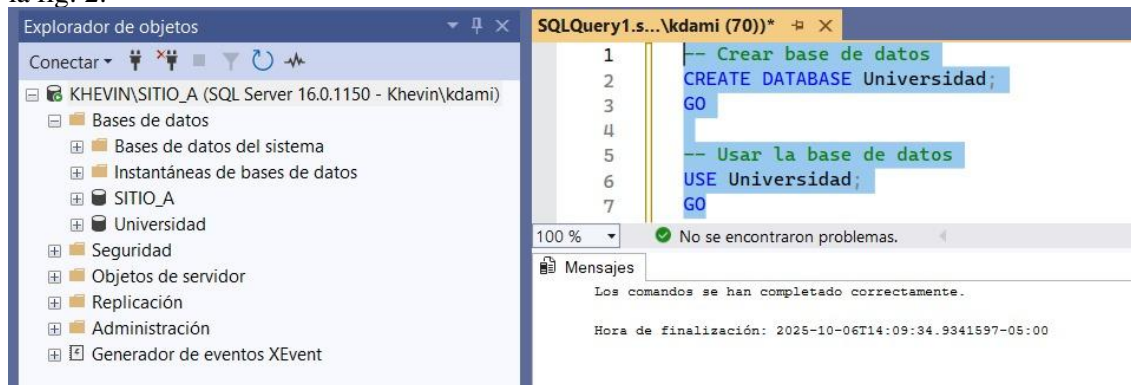


Fig. 2 Creación de BD Universidad

Creación de las tablas que se usarán en esta práctica.



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

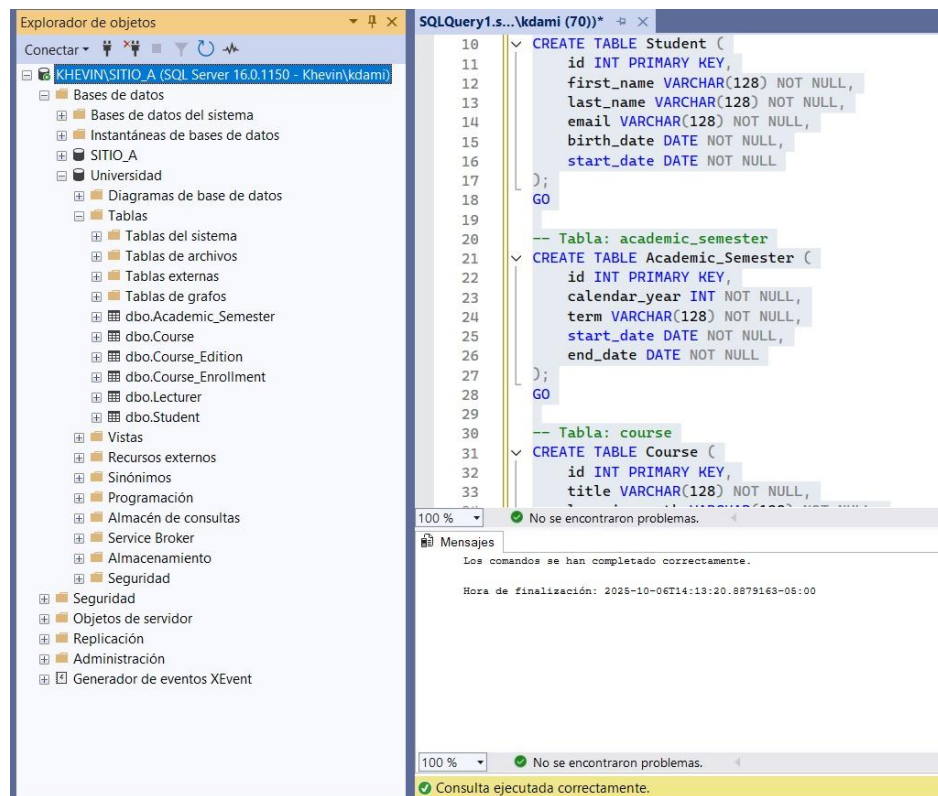


Fig. 3 Creación de Tablas

Insertión de datos en las tablas creadas.

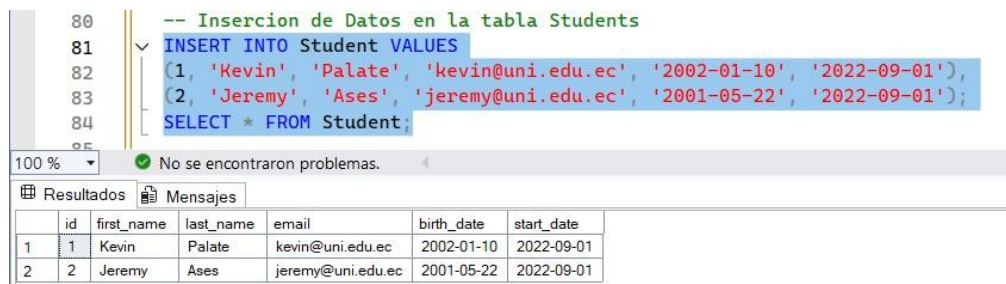


Fig. 4 Inserciones en la tabla Student

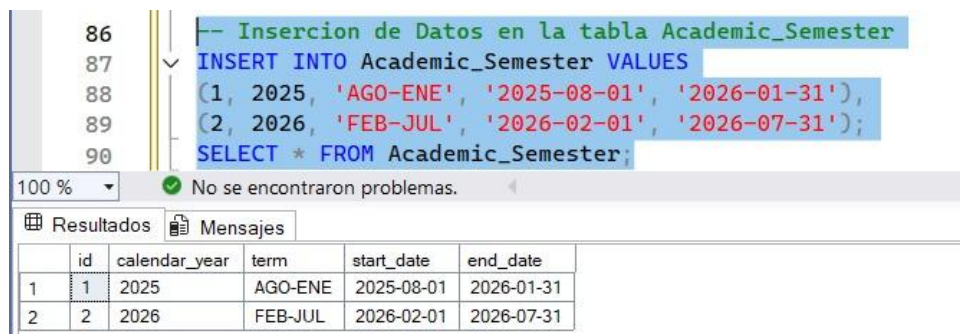


Fig. 5 Inserciones en la Tabla Academic_Semester



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  -- Insercion de Datos en la tabla Course
93  INSERT INTO Course VALUES
94  (1, 'Sistemas de Bases de Datos Distribuidos', 'Tecnologías de la Información',
95  'Curso sobre transacciones, replicación y consistencia en BDD.',
96  40, 10, 6, 1, 1),
97  (2, 'Programación Avanzada', 'Tecnologías de la Información',
98  'Programación orientada a objetos y patrones de diseño.',
99  50, 12, 5, 1, 1);
100 SELECT * FROM Course;
101
```

100 % No se encontraron problemas.

	id	title	learning_path	short_description	lecture_hours	tutorial_hours	ects_points	has_exam	has_project
1	1	Sistemas de Bases de Datos Distribuidos	Tecnologías de la Información	Curso sobre transacciones, replicación y consiste...	40	10	6	1	1
2	2	Programación Avanzada	Tecnologías de la Información	Programación orientada a objetos y patrones de d...	50	12	5	1	1

Fig. 6 Inserciones en la tabla Course

```
103  INSERT INTO Lecturer VALUES
104  (1, 'María', 'Zamora', 'MSc', 'mzamora@uni.edu.ec'),
105  (2, 'Carlos', 'Jiménez', 'PhD', 'cjimenez@uni.edu.ec');
106  SELECT * FROM Lecturer;
107
108
```

100 % No se encontraron problemas.

	id	first_name	last_name	degree	email
1	1	María	Zamora	MSc	mzamora@uni.edu.ec
2	2	Carlos	Jiménez	PhD	cjimenez@uni.edu.ec

Fig. 7 Inserciones en la tabla Lecturer

```
108  INSERT INTO Course_Edition VALUES
109  (1, 1, 1, 1), -- Curso 1, Semestre 1, Docente 1
110  (2, 2, 2, 2); -- Curso 2, Semestre 2, Docente 2
111  SELECT * FROM Course_Edition;
```

100 % No se encontraron problemas.

	id	course_id	academic_semester_id	lecturer_id
1	1	1	1	1
2	2	2	2	2

Fig. 8 Inserciones en tabla Course_Edition

```
120  INSERT INTO Course_Enrollment VALUES
121  (1, 1, 8.5, 9.0, 'A', 1),
122  (1, 2, 7.0, 8.0, 'B', 1),
123  (2, 2, 6.0, 4.0, 'F', 0);
124  SELECT * FROM Course_Enrollment;
125
```

100 % No se encontraron problemas.

	course_edition_id	student_id	midterm_grade	final_grade	course_letter_grade	passed
1	1	1	8.50	9.00	A	1
2	1	2	7.00	8.00	B	1
3	2	2	6.00	4.00	F	0

Fig. 9 Inserciones en tabla Course_Enrollment



Verificar que los datos ingresados en las diferentes tablas tengan una correcta integridad referencial, primero en la creación de un **Course_Edition** se intentará insertar un curso que no existe. En la Fig. 10 se muestra el error, ya que el curso 99 no existe con esto comprobamos la integridad referencial.

```
116 -- Prueba de integridad referencial (error intencional)
117 INSERT INTO Course_Edition VALUES
118 (3, 99, 1, 1); -- Error ya que el curso 99 no existe
119
120
```

100 % No se encontraron problemas. Línea: 116 Carácter: 1

Mensajes

Mens. 547, Nivel 16, Estado 0, Línea 117
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Course_Ed_course_3F466844". The conflict occurred in database "Universidad", table "dbo.Course", column 'id'.
The statement has been terminated.

Hora de finalización: 2025-10-06T14:25:20.1394339-05:00

Fig. 10 Prueba de Integridad 1

Ahora se intentará insertar un **Course_Enrollment** enviando como parámetro un **Course_Edition** que no existe, En la Fig. 11 se muestra como da error de clave foránea y con esto se demuestra la integridad de la base de datos Universidad.

```
127 -- Prueba de error (violando integridad)
128 INSERT INTO Course_Enrollment VALUES
129 (5, 1, 9.0, 9.5, 'A', 1); -- Course_Edition 5 no existe
130
131
```

100 % No se encontraron problemas. Línea: 127 Carácter: 1 SP

Mensajes

Mens. 547, Nivel 16, Estado 0, Línea 128
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Course_En_course_44081D61". The conflict occurred in database "Universidad", table "dbo.Course_Edition", column 'id'.
The statement has been terminated.

Hora de finalización: 2025-10-06T14:31:09.2352103-05:00

Fig. 11 Prueba de Integridad 2

Ahora iniciamos una transacción, luego insertamos un nuevo curso con su respectiva edición con el **COMMIT** confirmamos los cambios que se realizaron y con unas subconsultas podemos verificar que los datos se han guardado correctamente, tal como se muestra la Fig. 12.

```
SQLQuery2.sql...N(kdami (69))* SQLQuery1.s...N(kdami (70))*
135 -- Iniciamos la transacción
136 BEGIN TRANSACTION;
137
138 -- Insertamos un nuevo curso y su edición
139 INSERT INTO Course VALUES
140 (3, 'Comutación y Enrutamiento Básico', 'Tecnologías de la Información',
141 'Curso sobre redes TCP/IP, routers y seguridad en redes.',
142 45, 15, 6, 1, 0);
143
144 INSERT INTO Course_Edition VALUES
145 (3, 3, 1, 2); -- curso_id = 3, semestre_id = 1, docente_id = 2
146
147 -- Confirmamos los cambios
148 COMMIT;
149
150 -- Verificamos que se guardaron correctamente
151 SELECT * FROM Course;
152 SELECT * FROM Course_Edition;
```

100 % No se encontraron problemas. Línea: 135

Resultados Mensajes

id	title	learning_path	short_description	lecture_hours	tutorial_hours	ects_points	has_exam	has_project	
1	1	Sistemas de Bases de Datos Distribuidos	Tecnologías de la Información	Curso sobre transacciones, replicación y consiste...	40	10	6	1	1
2	2	Programación Avanzada	Tecnologías de la Información	Programación orientada a objetos y patrones de d...	50	12	5	1	1
3	3	Comutación y Enrutamiento Básico	Tecnologías de la Información	Curso sobre redes TCP/IP, routers y seguridad en ...	45	15	6	1	0

id	course_id	academic_semester_id	lecturer_id
1	1	1	1
2	2	2	2
3	3	1	2

Fig. 12 Transacción exitosa (COMMIT) de un course asignado a course_edition



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



Ahora empezar otra transacción insertar datos de un curso, una edición con un error referencial, como se ve la fig. 13, ver que en las inserciones no se realizó ninguna con 99, entonces aquí dará error pero con el **ROLLBACK** se puede revertir los cambios, Luego verificar si los datos se guardaron correctamente, al haber existido una falla el **ROLLBACK** no realizará ningún cambio en la base de datos.

```
SQLQuery2.sql...N\kdami (69))* SQLQuery1.s...N\kdami (70))*
155
156
157 -- Iniciamos una nueva transacción
158 BEGIN TRANSACTION;
159
160 -- Intentamos insertar un curso y una edición con error referencial
161 INSERT INTO Course VALUES
162 (4, 'Inteligencia Artificial', 'Ciencias Computacionales',
163 'Curso sobre algoritmos inteligentes y aprendizaje automático.',
164 40, 10, 5, 1, 1);
165
166 -- Esta línea generará un error porque el docente 99 no existe
167 INSERT INTO Course_Edition VALUES
168 (4, 4, 1, 99);
169
170 -- Si ocurre un error, revertimos los cambios
171 ROLLBACK;
172
173 -- Verificamos si se guardaron los datos (no deberían aparecer)
174 SELECT * FROM Course WHERE id = 4;
175 SELECT * FROM Course_Edition WHERE id = 4;
176
177
```

100 % No se encontraron problemas. Línea: 157 Carácter: 1 SPC CRLF

Resultados Mensajes

(1 fila afectada)
Mens. 547, Nivel 16, Estado 0, Línea 167
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Course_Ed_Lectu_612B086". The conflict occurred in database "Universidad", table "dbo.Lecturers", column "id".
The statement has been terminated.

(0 filas afectadas)
(0 filas afectadas)

Hora de finalización: 2025-10-06T14:39:34.7810867-05:00

100 % No se encontraron problemas. Línea: 12 Carácter: 1 TABULACIONES MIXTO

Fig. 13 Transacción con error (ROLLBACK) sobre asignación de un docente inexistente a una edición

Primero se abre una nueva sesión para realizar una transacción en la cual actualiza una nota final al estudiante con id=1, para verificar las transacciones entre diferentes sesiones.

```
SQLQuery4.s...N\kdami (65))* SQLQuery2.sql ... Ejecutando... SQLQuery1.sql...N\kdami (70))*
1 USE Universidad;
2 GO
3
4
5 BEGIN TRANSACTION;
6
7 -- Actualizamos la nota final de Kevin
8 UPDATE Course_Enrollment
9 SET final_grade = 10.0,
10 course_letter_grade = 'A+',
11 passed = 1
12 WHERE student_id = 1;
13
14 -- No hacemos COMMIT todavía
15
```

100 % No se encontraron problemas. Línea: 1 Carácter: 1 SPC CRLF

Mensajes

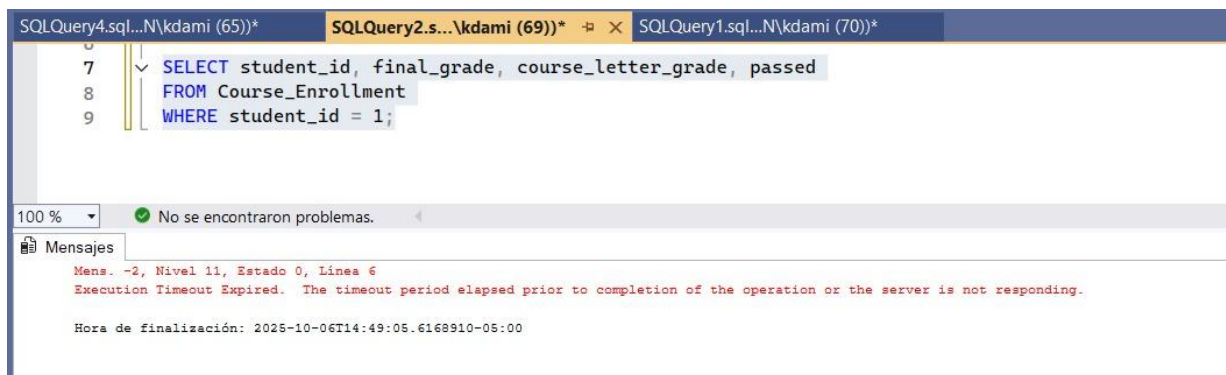
(1 fila afectada)

Hora de finalización: 2025-10-06T14:46:01.4204167-05:00

Fig. 14 Sesión de pruebas_1



Se abre una nueva sesión a la cual se denominara sesión 2, en la cual verificamos que la transacción se haya realizado o no y podemos observar en la fig.15 que debido a que la sesión 1 no finalizo la transacción, no se pudo acceder a la lectura de datos de la sesión 2.



```
SQLQuery4.sql...N\kdami (65))* SQLQuery2.s...N\kdami (69))* SQLQuery1.sql...N\kdami (70))*
7 SELECT student_id, final_grade, course_letter_grade, passed
8 FROM Course_Enrollment
9 WHERE student_id = 1;

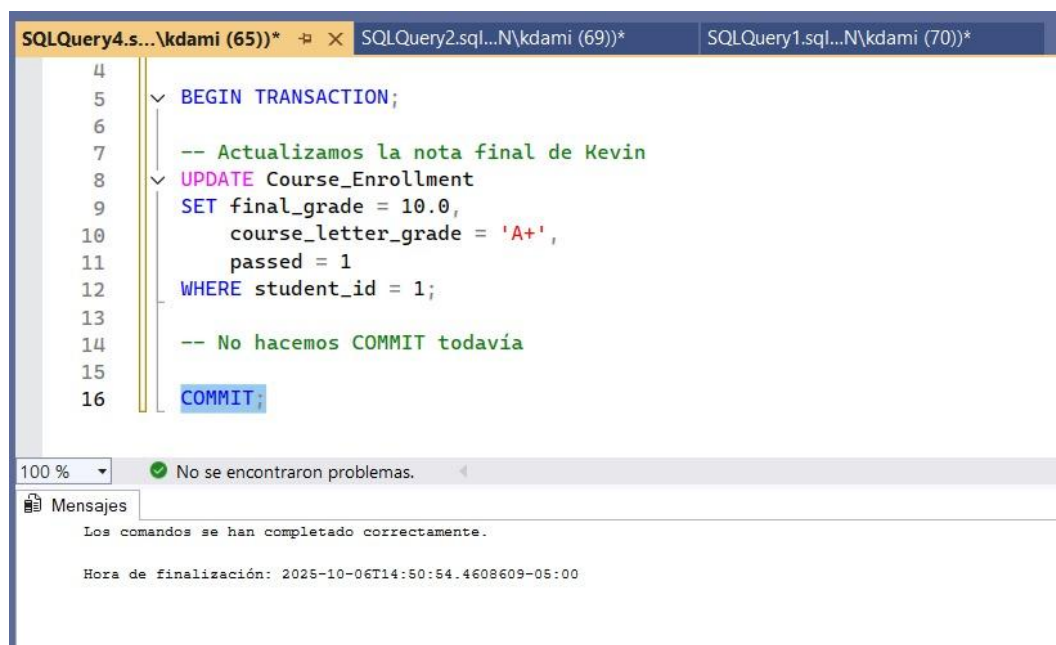
100 % No se encontraron problemas.

Mensajes
Mens. -2, Nivel 11, Estado 0, Línea 6
Execution Timeout Expired. The timeout period elapsed prior to completion of the operation or the server is not responding.

Hora de finalización: 2025-10-06T14:49:05.6168910-05:00
```

Fig. 15 Sesión de pruebas_2

Se realiza el commit (guardado de datos) para completar la transacción en la sesión 1.



```
SQLQuery4.s...N\kdami (65))* SQLQuery2.sql...N\kdami (69))* SQLQuery1.sql...N\kdami (70))*
4
5 BEGIN TRANSACTION;
6
7 -- Actualizamos la nota final de Kevin
8 UPDATE Course_Enrollment
9 SET final_grade = 10.0,
10 course_letter_grade = 'A+',
11 passed = 1
12 WHERE student_id = 1;
13
14 -- No hacemos COMMIT todavía
15
16 COMMIT;

100 % No se encontraron problemas.

Mensajes
Los comandos se han completado correctamente.

Hora de finalización: 2025-10-06T14:50:54.4608609-05:00
```

Fig. 16 Sesión 1: Confirmamos los cambios



En la sesión 2, ya se puede realizar la lectura de datos, debido a que ya se completó la sesión 1

```
7 SELECT student_id, final_grade, course_letter_grade, passed
8 FROM Course_Enrollment
9 WHERE student_id = 1;
```

100 % No se encontraron problemas.

Resultados Mensajes

	student_id	final_grade	course_letter_grade	passed
1	1	10.00	A+	1

Fig. 17 Sesión 2: Verificamos nuevamente

Este bloque SQL utiliza `SET XACT_ABORT ON` para asegurar que toda la transacción se revierta automáticamente si ocurre cualquier error durante la operación. Intenta insertar un nuevo curso y su edición, pero al fallar (por datos referenciales inexistentes), toda la operación se cancela, garantizando que no queden registros inconsistentes en la base de datos y manteniendo la integridad de los datos.

```
19 -- Activamos XACT_ABORT
20 SET XACT_ABORT ON;
21 GO
22
23 BEGIN TRANSACTION;
24
25 -- Inserción válida
26 INSERT INTO Course VALUES
27 (7, 'Big Data', 'Tecnologías de la Información',
28 'Curso sobre análisis de grandes volúmenes de datos.',
29 40, 10, 6, 1, 0);
30
31 -- Inserción con error: el semestre 99 no existe
32 INSERT INTO Course_Edition VALUES
33 (7, 7, 1, 1);
34
35 -- Commit (no llegará a ejecutarse si hay error)
36 COMMIT;
37
38 -- Verificamos si el curso fue guardado
39 SELECT * FROM Course WHERE id = 7;
40 SELECT * FROM Course_Edition WHERE id = 7;
```

100 % No se encontraron problemas.

Resultados Mensajes

	id	title	learning_path	short_description	lecture_hours	tutorial_hours	ects_points	has_exam	has_project
1	7	Big Data	Tecnologías de la Información	Curso sobre análisis de grandes volúmenes de dat...	40	10	6	1	0

	id	course_id	academic_semester_id	lecturer_id
1	7	7	1	1

Fig. 18 Control de errores con XACT_ABORT

Este código utiliza un bloque TRY...CATCH para manejar errores en transacciones SQL. Al intentar insertar un curso y su edición, se genera un error deliberado por un docente inexistente. El bloque CATCH detecta automáticamente la falla, muestra el mensaje de error y ejecuta ROLLBACK, revirtiendo toda la transacción para mantener la integridad de la base de datos y evitar cambios parciales o inconsistentes.



```
45 BEGIN TRY
46     BEGIN TRANSACTION;
47
48     -- Inserción válida
49     INSERT INTO Course VALUES
50     (8, 'Machine Learning', 'Tecnologías de la Información',
51     'Curso sobre algoritmos de ML y AI aplicada.',
52     35, 8, 5, 1, 1);
53
54     -- Error intencional: docente 99 no existe
55     INSERT INTO Course_Edition VALUES
56     (8, 8, 1, 2);
57
58     COMMIT; -- Solo si no hay errores
59 END TRY
60 BEGIN CATCH
61     PRINT 'Error detectado: ' + ERROR_MESSAGE();
62     ROLLBACK;
63 END CATCH;
64
65 -- Verificamos los datos
66 SELECT * FROM Course WHERE id = 8;
67 SELECT * FROM Course_Edition WHERE id = 8;
68
```

100 % No se encontraron problemas. Link

Resultados Mensajes

id	title	learning_path	short_description	lecture_hours	tutorial_hours	ects_points	has_exam	has_project	
1	8	Machine Learning	Tecnologías de la Información	Curso sobre algoritmos de ML y AI aplicada.	35	8	5	1	1

id	course_id	academic_semester_id	lecturer_id	
1	8	8	1	2

Fig. 19 Control de errores con TRY...CATCH

Este código implementa una transacción manual que inserta un curso y luego valida con IF EXISTS si ya existe un identificador duplicado. Al detectar duplicidad, usa RAISERROR para lanzar un mensaje personalizado y ejecutar ROLLBACK, revirtiendo toda la transacción. Esto garantiza que no se guarden registros duplicados y mantiene la integridad de la base de datos mediante validaciones explícitas.

```
73
74 BEGIN TRANSACTION;
75
76 -- Inserción de curso
77 INSERT INTO Course VALUES
78 (11, 'Gestión de Proyectos TI', 'Tecnologías de la Información',
79 'Curso administración de Proyectos TI',
80 40, 10, 5, 1, 1);
81
82 -- Forzamos un error si el curso ya existe (simulación)
83 IF EXISTS (SELECT 1 FROM Course WHERE id = 9)
84 BEGIN
85     RAISERROR ('El curso ya existe. Transacción revertida.', 16, 1);
86 END
87
88 -- Commit
89 COMMIT;
```

0 % No se encontraron problemas.

Mensajes

(1 fila afectada)
Mens. 50000, Nivel 16, Estado 1, Línea 85
El curso ya existe. Transacción revertida.

Hora de finalización: 2025-10-06T15:05:38.6871192-05:00

Fig. 20 Control de errores con ON_ERROR



Resultados obtenidos

Un sistema transaccional siempre finaliza con Commit o Rollback, caso contrario puede generar un estado inconsistente de la BD. Para el manejo de errores existen varias alternativas que funciona de acuerdo al contexto.

2.7 Habilidades blandas empleadas en la práctica

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

2.8 Conclusiones

Al manejar transacciones se debe cuidar de no producir un estado inconsistente de la BD. La captura de errores va a depender del tipo de error y su severidad.

2.9 Recomendaciones

Un sistema transaccional se complica muchísimo en ambientes distribuidos

2.10 Referencias bibliográficas

[1]“Transactions (Transact-SQL) - SQL Server”. Microsoft Learn: Build skills that open doors in your career. Accedido el 6 de octubre de 2025. [En línea]. Disponible: <https://learn.microsoft.com/es-es/sql/t-sql/language-elements/transactions-transact-sql?view=sql-server-ver17>

[2]“TRY...CATCH (Transact-SQL) - SQL Server”. Microsoft Learn: Build skills that open doors in your career. Accedido el 6 de octubre de 2025. [En línea]. Disponible: <https://learn.microsoft.com/es-es/sql/t-sql/language-elements/try-catch-transact-sql?view=sql-server-ver17>

2.11 Anexos

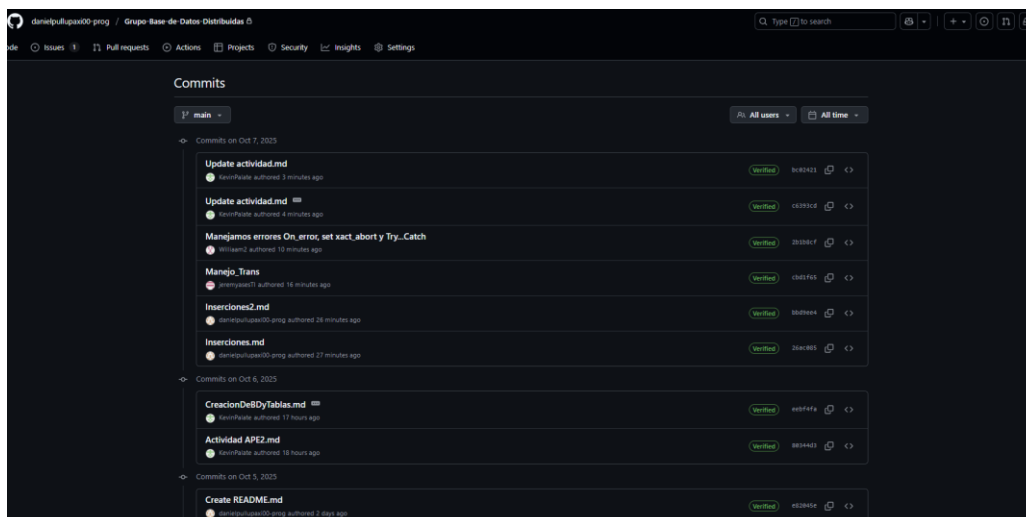


Fig 1 Registro de commits



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

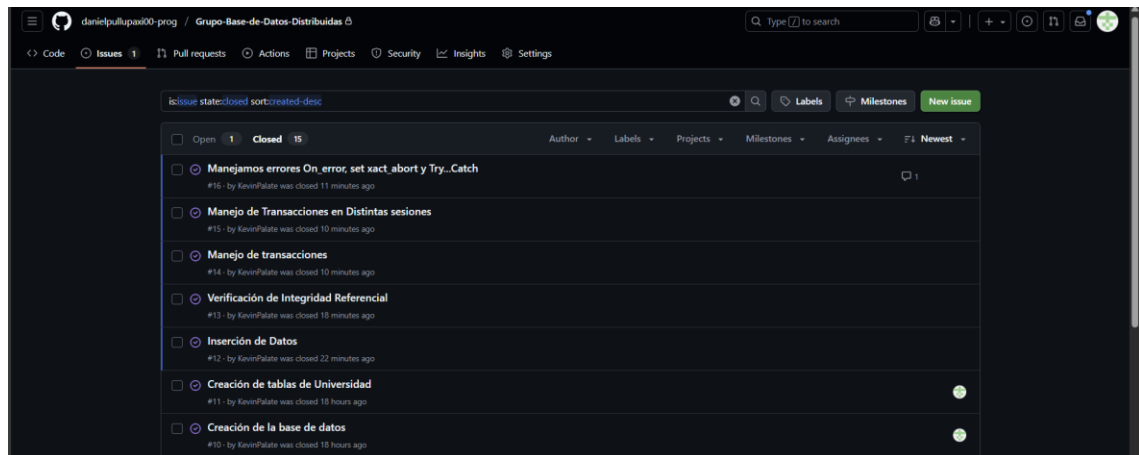


Fig 2 Registro de tareas

-- Crear base de datos

```
CREATE DATABASE Universidad;
```

```
GO
```

-- Usar la base de datos

```
USE Universidad;
```

```
GO
```

-- Tabla: student

```
CREATE TABLE Student (  
    id INT PRIMARY KEY,  
    first_name VARCHAR(128) NOT NULL,  
    last_name VARCHAR(128) NOT NULL,  
    email VARCHAR(128) NOT NULL,  
    birth_date DATE NOT NULL,  
    start_date DATE NOT NULL
```

```
);
```

```
GO
```

-- Tabla: academic_semester

```
CREATE TABLE Academic_Semester (  
    id INT PRIMARY KEY,  
    calendar_year INT NOT NULL,  
    term VARCHAR(128) NOT NULL,  
    start_date DATE NOT NULL,  
    end_date DATE NOT NULL
```

```
);
```

```
GO
```

-- Tabla: course

```
CREATE TABLE Course (  
    id INT PRIMARY KEY,  
    title VARCHAR(128) NOT NULL,  
    learning_path VARCHAR(128) NOT NULL,  
    short_description VARCHAR(1200),  
    lecture_hours INT NOT NULL,  
    tutorial_hours INT NOT NULL,  
    ects_points INT NOT NULL,  
    has_exam BIT NOT NULL,
```



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



```
has_project BIT NOT NULL
);
GO

-- Tabla: lecturer
CREATE TABLE Lecturer (
    id INT PRIMARY KEY,
    first_name VARCHAR(128) NOT NULL,
    last_name VARCHAR(128) NOT NULL,
    degree VARCHAR(32) NOT NULL,
    email VARCHAR(128)
);
GO

-- Tabla: course_edition
CREATE TABLE Course_Edition (
    id INT 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)
);
GO

-- Tabla: course_enrollment
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 VARCHAR(3),
    passed BIT,
    PRIMARY KEY (course_edition_id, student_id),
    FOREIGN KEY (course_edition_id) REFERENCES Course_Edition(id),
    FOREIGN KEY (student_id) REFERENCES Student(id)
);
GO

-- Insercion de Datos en la tabla Students
INSERT INTO Student VALUES
(1, 'Kevin', 'Palate', 'kevin@uni.edu.ec', '2002-01-10', '2022-09-01'),
(2, 'Jeremy', 'Ases', 'jeremy@uni.edu.ec', '2001-05-22', '2022-09-01');
SELECT * FROM Student;

-- Insercion de Datos en la tabla Academic_Semester
INSERT INTO Academic_Semester VALUES
(1, 2025, 'AGO-ENE', '2025-08-01', '2026-01-31'),
(2, 2026, 'FEB-JUL', '2026-02-01', '2026-07-31');
SELECT * FROM Academic_Semester;

-- Insercion de Datos en la tabla Course
INSERT INTO Course VALUES
```




(1, 'Sistemas de Bases de Datos Distribuidos', 'Tecnologías de la Información',

'Curso sobre transacciones, replicación y consistencia en BDD.',

40, 10, 6, 1, 1),

(2, 'Programación Avanzada', 'Tecnologías de la Información',

'Programación orientada a objetos y patrones de diseño.',

50, 12, 5, 1, 1);

SELECT * FROM Course;

-- Insercion de Datos en la tabla Students

INSERT INTO Lecturer VALUES

(1, 'María', 'Zamora', 'MSc', 'mzamora@uni.edu.ec'),

(2, 'Carlos', 'Jiménez', 'PhD', 'cjimenez@uni.edu.ec');

SELECT * FROM Lecturer;

INSERT INTO Course_Edition VALUES

(1, 1, 1, 1), -- Curso 1, Semestre 1, Docente 1

(2, 2, 2, 2); -- Curso 2, Semestre 2, Docente 2

SELECT * FROM Course_Edition;

-- Prueba de integridad referencial (error intencional)

INSERT INTO Course_Edition VALUES

(3, 99, 1, 1); -- Error ya que el curso 99 no existe

INSERT INTO Course_Enrollment VALUES

(1, 1, 8.5, 9.0, 'A', 1),

(1, 2, 7.0, 8.0, 'B', 1),

(2, 2, 6.0, 4.0, 'F', 0);

SELECT * FROM Course_Enrollment;

-- Prueba de error (violando integridad)

INSERT INTO Course_Enrollment VALUES

(5, 1, 9.0, 9.5, 'A', 1); -- Course_Edition 5 no existe

-- Iniciamos la transacción

BEGIN TRANSACTION;

-- Insertamos un nuevo curso y su edición

INSERT INTO Course VALUES

(3, 'Conmutación y Enrutamiento Básico', 'Tecnologías de la Información',



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



'Curso sobre redes TCP/IP, routers y seguridad en redes.',
45, 15, 6, 1, 0);

INSERT INTO Course_Edition VALUES
(3, 3, 1, 2); -- curso_id = 3, semestre_id = 1, docente_id = 2

-- Confirmamos los cambios
COMMIT;

-- Verificamos que se guardaron correctamente
SELECT * FROM Course;
SELECT * FROM Course_Edition;

-- Iniciamos una nueva transacción

BEGIN TRANSACTION;

-- Intentamos insertar un curso y una edición con error referencial

INSERT INTO Course VALUES
(4, 'Inteligencia Artificial', 'Ciencias Computacionales',
'Curso sobre algoritmos inteligentes y aprendizaje automático',
40, 10, 5, 1, 1);

-- Esta línea generará un error porque el docente 99 no existe

INSERT INTO Course_Edition VALUES
(4, 4, 1, 99);

-- Si ocurre un error, revertimos los cambios
ROLLBACK;

-- Verificamos si se guardaron los datos (no deberían aparecer)

SELECT * FROM Course WHERE id = 4;
SELECT * FROM Course_Edition WHERE id = 4;

--sesion 1

USE Universidad;
GO

BEGIN TRANSACTION;

-- Actualizamos la nota final de Kevin
UPDATE Course_Enrollment
SET final_grade = 10.0,



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



```
course_letter_grade = 'A+',  
passed = 1  
WHERE student_id = 1;
```

```
-- No hacemos COMMIT todavía
```

```
COMMIT TRANSACTION;
```

```
-- Activamos XACT_ABORT  
SET XACT_ABORT ON;  
GO
```

```
BEGIN TRANSACTION;
```

```
-- Inserción válida
```

```
INSERT INTO Course VALUES
```

```
(7, 'Big Data', 'Tecnologías de la Información',
```

```
'Curso sobre análisis de grandes volúmenes de datos.',
```

```
40, 10, 6, 1, 0);
```

```
-- Inserción con error: el semestre 99 no existe
```

```
INSERT INTO Course_Edition VALUES  
(7, 7, 1, 1);
```

```
-- Commit (no llega a ejecutarse si hay error)
```

```
COMMIT;
```

```
-- Verificamos si el curso fue guardado  
SELECT * FROM Course WHERE id = 7;  
SELECT * FROM Course_Edition WHERE id = 7;
```

```
BEGIN TRY  
BEGIN TRANSACTION;
```

```
-- Inserción válida
```

```
INSERT INTO Course VALUES
```

```
(8, 'Machine Learning', 'Tecnologías de la Información',
```

```
'Curso sobre algoritmos de ML y AI aplicada.',  
35, 8, 5, 1, 1);
```

```
-- Error intencional: docente 99 no existe  
INSERT INTO Course_Edition VALUES  
(8, 8, 1, 2);
```



```
COMMIT; -- Solo si no hay errores
END TRY
BEGIN CATCH
    PRINT 'Error detectado: ' + ERROR_MESSAGE();
    ROLLBACK;
END CATCH;

-- Verificamos los datos
SELECT * FROM Course WHERE id = 8;
SELECT * FROM Course_Edition WHERE id = 8;

delete from Course where id = 10;

BEGIN TRANSACTION;

-- Inserción de curso
INSERT INTO Course VALUES
(11, 'Gestión de Proyectos TI', 'Tecnologías de la Información',
'Curso administración de Proyectos TI',
40, 10, 5, 1, 1);

-- Forzamos un error si el curso ya existe (simulación)
IF EXISTS (SELECT 1 FROM Course WHERE id = 9)
BEGIN
    RAISERROR ('El curso ya existe. Transacción revertida.', 16, 1);
END

-- Commit
COMMIT;

--sesion2
SELECT * FROM Course;
SELECT * FROM Course_Edition;
SELECT * FROM Course_Enrollment;
SELECT * FROM Lecturer;
SELECT * FROM Student;

SELECT student_id, final_grade, course_letter_grade, passed
FROM Course_Enrollment
WHERE student_id = 1;
```