Module 13: Create server-side programming constructs in Azure Cosmos DB SQL API

Build multi-item transactions with the Azure Cosmos DB SQL API

Understand transactions

In a database, a transaction is typically defined as a sequence of point operations grouped together into a single unit of work. It's expected that a transaction provides ACID guarantees.

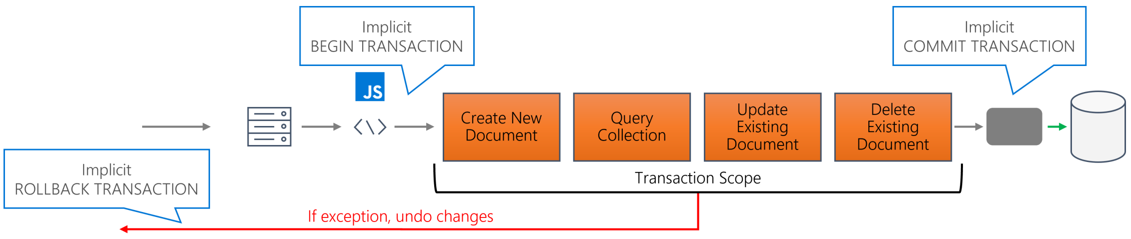
- **Atomicity** guarantees that all the work done inside a transaction is treated as a single unit where either all of it is committed or none.
- **Consistency** makes sure that the data is always in a healthy internal state across transactions.
- **Isolation** guarantees that no two transactions interfere with each other – generally, most commercial systems provide multiple isolation levels that can be used based on the application's needs.
- **Durability** ensures that any change that's committed in the database will always be present.

Author Stored procedures

```
function createProduct(item) {
    var context = getContext();
    var container = context.getCollection();
    var accepted = container.createDocument(
        container.getSelfLink(),
        item,
        (error, newItem) => {
            if (error) throw error;
            context.getResponse().setBody(newItem)
        }
    );
    if (!accepted) return;
}
```

Rollback transactions

Azure Cosmos DB’s SQL API will roll back the entire transaction if a single exception is thrown from the stored procedure script.



Check with following SQL Query if item was created:

```
SELECT * FROM c
where c.name = "Bike"
```

Expand query and transaction functionality in Azure Cosmos DB SQL API

Create User-defined functions (UDFs)

UDFs are used to extend the Azure Cosmos DB SQL API's query language grammar and implement custom business logic and can only be called from inside queries.

Suppose you have the following item

```
{
  "name": "Black Bib Shorts (Small)",
  "price": 80.00
}
```

Create a UDF that calculates 15% tax

```
function addTax(preTax)
{
    return preTax * 1.15;
}
```

Run this query that returns the price and the tax

```
SELECT
    p.name,
    p.price,
    udf.addTax(p.price) AS priceWithTax
FROM products p
WHERE p.name = "Black Bib Shorts (Small)“
```

Add triggers to an operation – Pre-trigger

Pre-triggers are the core way that Azure Cosmos DB SQL API can inject business logic before an operations and cannot have any input parameters.

Suppose you want to insert the following item

```
{
  "id": "caab0e5e-c037-48a4-a760-140497d19452",
  "name": "Handlebar",
  "categoryId": "e89a34d2-47ee-4da8-bcf6-10f552604b79",
  "categoryName": "Accessories",
  "price": 50
}
```

You want to automatically insert a "label" so that the document becomes

```
{
  "id": "caab0e5e-c037-48a4-a760-140497d19452",
  "name": "Handlebar"
```