Soft skills-Aug 14

Monday, August 14, 2023 10:01 AM

Professionalism

DevOps

Thursday, August 24, 2023 9:23 AM

What is DevOps

Culture to promote development
And operation process collaboratively

Increases speed
Better management and communication

DevOps Principles
Automation-workflows, testing new code
Iteration-write chunks of code during time box
Continuous improvement
collaboration

Github actions

CI/CD platform allowing us to automate build,
Test and deployment pipeline

Create a workflow that build and test every pull req
To your repo

Runner-individual vm

Dev-pushes
Operations team-tests and deploys

SQL

Tuesday, August 29, 2023 9:15 AM

DATA
Collection of symbols,characters etc. stored electronically

WHAT?
Database
Collection of related data and metadata organized in a structured format
For optimized info mgmt.

DBMS
s/w for easy creation, access and modification of db.
For efficient mgmt.

Database System
Integrated system of hardware s/w people procedures
That define and regulate collection storage and mgmt. and use of data
Within database environment

WHY?

PURPOSE OF DB
Optimizes data mgmt.
Transforms data to info

FUNCTIONS
Stores data and related data entry forms, report def etc.
Hides complexities of RDB model from user
Enforces integrity
Implements data security mgmt.--->access, privacy

Data Models

Entity
Thing about which data are to be collected and stored

ATTRIBUTE
Association, like emp name, phone

Constraints
Restrictions

Hiearchial model

Adv
Simple understanding
Centralized

Disadv
Limited rep of relationships only one to many
Complex implement
Lack of standard

NORMALIZATION
Converting bigger tables to smaller
To reduce redundancy
Repeated data is redundancy

1NF
Atomicity maintained.
Data present in each attribute cannot have more than one value

2nf
1nf
No partial dependency-when non primary attribute are partially dependent on a part of candidate key
All should depend on primary key only

CANDIDATE KEY
Attribute/set of attributes uniquely identifying a tuple
Except primary other attr are primary key

Superkey
Superset of candidate key

ER
Analyze structure of db.
Relation b/w entities and attributes

Entity
Thing/object in real world

Online Transactional Processing Database(OLTP)
Process transaction as quicky as possible
Uses ER model

OLAP
Online analytical processing db
All data from OLTP DB->batch process>PERFORM ETL->put in OLAP
As data increases number of joins increase to retrieve info(N-1)
Hence we require Denormalization

Dimensional Modelling

Sql-2

Wednesday, August 30, 2023 9:10 AM

Schema
Collection of dimensional table and fact table(contains fk and numerical measure)--->star schema and flat schema

STAR SCHEMA
All tables are fully normalized
All tables connected to a single table

Snow flake schema
Dimesnion+flat table
Some dimension table are partially normalized

Fact Table:
- A fact table contains quantitative data or measures. These are the numerical values that represent business transactions or events.
- Fact tables usually store aggregated data, such as sales revenue, quantities sold, or profit margins.
- Fact tables are typically quite large because they store a lot of data over time.
- Each row in a fact table represents a specific event or transaction and contains foreign keys that link to dimension tables.
- Fact tables are the heart of the data warehouse and provide the context for analysis by associating measures with dimensions.
- Examples of fact table attributes: sales revenue, profit, quantity, cost, date, time, etc.

Dimension Table:
- A dimension table contains descriptive attributes that provide context to the measures stored in the fact table.
- Dimension tables hold information that helps to categorize, filter, and analyze the data in the fact table.
- Dimension tables are usually smaller than fact tables and can be used to group and filter data.
- Each row in a dimension table represents a unique category, such as a product, customer, location, or time period.
- Dimension tables are connected to the fact table through foreign key relationships.
- Examples of dimension table attributes: product name, customer name, location, date, category, etc.

In simpler terms, think of the fact table as the "what" and the dimension tables as the "who," "where," "when," and "how." The fact table contains the measurable data you want to analyze, while the dimension tables provide the context and details that allow you to understand and slice the data from different angles.

Remember, both fact and dimension tables are crucial components of a data warehouse schema, and they work together to enable efficient analysis and reporting on large datasets.

Certainly! Star schema and snowflake schema are two common data warehousing design approaches.

Star Schema:
The star schema is a type of database schema used in data warehousing. It consists of a central fact table surrounded by dimension tables. The fact table contains measures (numerical data) and foreign keys to the dimension tables. Dimension tables store descriptive information related to the measures. The star schema is simple to understand and query, making it efficient for reporting and analysis.

In a star schema, the relationship between the fact table and dimension tables is straightforward, resembling a star when visualized. The dimension tables are denormalized, meaning they may contain redundant data to speed up query performance.

Snowflake Schema:
The snowflake schema is an extension of the star schema. It involves normalizing dimension tables by breaking down hierarchies into multiple related tables. This normalization reduces data redundancy but increases the complexity of querying since it requires joining more tables.

In the snowflake schema, the dimension tables are connected in a snowflake-like structure, with various levels of sub-dimensions. While it may help save storage space, it can make querying more complex and slightly slower due to the increased number of joins.

In summary:
- **Star Schema:** Simple and denormalized, easy to query, suitable for quick reporting and analysis.
- **Snowflake Schema:** Normalized and more complex, reduces redundancy, but queries may involve more joins and be somewhat slower.

The choice between star and snowflake schemas depends on factors like performance requirements, data complexity, and the balance between query efficiency and data storage.

Sure, here are examples of a star schema and a snowflake schema based on a sales data scenario:

Star Schema Example:

In this example, we have a central fact table called "Sales" surrounded by two dimension tables: "Products" and "Time."

Fact Table - Sales:

Date	ProductID	Quantity	Revenue
2023-08-01	101	10	$500
2023-08-02	102	5	$250
...	...	...	...

Dimension Table - Products:

ProductID	ProductName	Category
101	Laptop	Electronics
102	Smartphone	Electronics
...	...	...

Dimension Table - Time:

Date	Day	Month	Year
2023-08-01	1	8	2023
2023-08-02	2	8	2023
...	...	...	...

Snowflake Schema Example:

In this example, we extend the star schema to a snowflake schema by normalizing the "Products" dimension further into a "Categories" dimension.

Fact Table - Sales:

(as above)

Dimension Table - Products:

ProductID	ProductName	CategoryID
101	Laptop	1
102	Smartphone	1
...	...	...

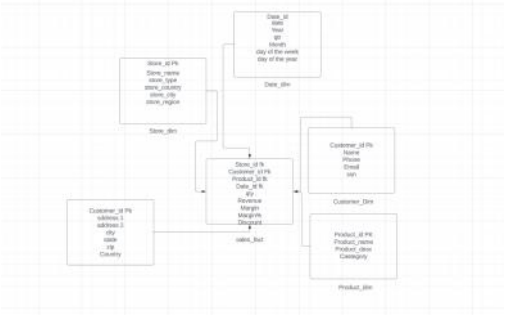
Dimension Table - Categories:

CategoryID	CategoryName
1	Electronics
...	...

Dimension Table - Time:

(as above)

In the snowflake schema, the "Categories" dimension is normalized, leading to more efficient storage, but queries might involve more joins between the "Products" and "Categories" tables.



Surrogate key: is unique sequential number it replaces oltp Primary and fk

Using primary keys as IDs is a common approach, but there are cases where using surrogate keys offers advantages. Let's explore why you might choose one over the other:

Primary Key as an ID:
- A primary key is a unique identifier for a record within a table.
- It can be based on a natural attribute of the data, like a person's social security number or a product's barcode.
- If the natural attribute is guaranteed to be unique and stable, using it as a primary key can simplify the database design.
- Primary keys often have meaning, which can help people understand the data without needing to look up additional information.

Surrogate Key as an ID:
- A surrogate key is a simple number generated by the database system, with no inherent meaning to users.

Slowly changing dimension
Technique to track changes in the dimension table

With SCD, you create a system to track these changes:

Type 1 (No History): You update the record, and the old author's name is gone. But you lose the history of the original author.

Type 2 (Full History): You keep the old record, mark it as "historical," and add a new record for the book with the new author. Now you can see both authors, and your history is intact.

Type 3 (Partial History): You update the record but keep a spot for the old author's name. This way, you know the change happened, but you don't keep all the history.

- Surrogate keys are typically smaller in size compared to some natural keys, which can improve performance.
- They provide a stable identifier even if the underlying data changes (e.g., a person changes their name).
- Using surrogate keys can simplify joins and relationships between tables, leading to more efficient queries.
- They provide an extra level of security, as they don't expose any sensitive information about the record.

When to Use Which:
- Use primary keys based on natural attributes when those attributes are guaranteed to be unique, unchanging, and aren't too large.
- Use surrogate keys when you want a stable, efficient, and secure way to identify records, particularly when working with larger databases or complex relationships.

In summary, the choice between using a primary key based on a natural attribute or a surrogate key depends on factors like data stability, performance needs, and the complexity of relationships in your database. Both approaches have their merits, and the decision should be based on the specific requirements of your application.

Surrogate key is req in olap
Because emp no allows only unique values
So we use surrogate key--->it is in every dim table
After change we have new surrogate key-->with changes made
No point in increasing column

Oldp to olap
Denormalize
Removing null and duplicate calues
Decoding the data
Timming data

Data Warehouse
Enterprise wide collection of historical data
Centralized database
Subject oriented
Intigrated
Time varient
Not volatile collection

Data mart
Departmental wise historical data basically subset of d/w

BIG DATA

HADOOP
Terminology
Node-individual server for processing
Collection of node->Rack
Collection of Racks->cluster

Architecture
1)HDFS
2)MapReduce engine------>a)framework for performing calculations on the data in the file system
b)Has built in resource management and schedule

A file system is a method used by computers to organize and store files on a storage device, such as a hard drive, solid-state drive, or external storage.

HDFS
- Designed to tolerate high component failure rate
- Through replication
- Designed to handle large files
- No random access
- Use blocks to store a file or part of file

Not same as OS file blocks
Default->64mb
Size of the file can be larger than single disk in cluster

HDFS REPLICATION
Blocks with data are replicated to multiple nodes

NAME NODE
Stores metadata
Data doesn't go through name node
Data does not store in name node
Single point of failure--->hence mirror name node and do not use that in expensive commodity h/w
Has large memory req-->File system metadata is maintained in RAM

Data Node
Many for a cluster
Blocks from diff files can be on same data node

Job tracker
One per hadoop cluster
Manages mapreduce job in a single cluster
Receives job req submitted by client

YARN
Allocates system resources to various applications
Running in Hadoop cluster and scheduling tasks to be executed
On diff cluster nodes

Topology awareness

N1	N1
N2	N2
N3	N3
N4	N4

Rack1	Rack2

Mapreduce
Scalability and easy data processing
Used to map and reduce
Larger->smaller chunks->reduce(key value pairs)

Hive --->declarative
Completely relational

Hdfs

Invoke:hadoop fs <args>
List current directory-> hadoop fs -ls

Data lake
Replacable to hdfs
Centralized repo to store process and secure
Large amounts of structured semi structured and unstructured data
Inexpensive and scalable commodity h/w clusters
Stores all formats of data whereas wearhouse does only structured

While both data lakes and data warehouses are used to store and manage data, they serve different purposes and have distinct characteristics:

Data Lake:

A data lake is a storage repository that can hold vast amounts of structured, semi-structured, and unstructured data. It's designed to store data in its raw, original form without a predefined structure. Here are some key points about data lakes:

1. **Data Variety:** Data lakes can store various types of data, including text, images, videos, sensor data, and more, without the need for extensive data transformation.

2. **Schema Flexibility:** Data lakes allow you to store data without imposing a fixed schema upfront. This is particularly useful when dealing with diverse and evolving data sources.

3. **Scalability:** Data lakes can scale out easily by adding more storage capacity, making them suitable for handling large volumes of data.

Azure Data Lake Storage and Hadoop Distributed File System (HDFS) are both designed to handle large volumes of data, but they belong to different environments. Here's a comparison:

Azure Data Lake Storage:

- **Platform:** Azure Data Lake Storage is a part of the Microsoft Azure cloud platform.
- **Storage Type:** It's a scalable and secure data lake storage service built for the cloud. It supports both structured and unstructured data.
- **Integration:** Seamlessly integrates with other Azure services and tools like Azure Databricks, Azure HDInsight, and Azure Data Factory.
- **Security:** Provides advanced security features like Azure Active Directory integration, encryption, and access control.
- **Management:** Managed by Microsoft, so you don't need to

data, and more, without the need for extensive data transformation.

2. **Schema Flexibility:** Data lakes allow you to store data without imposing a fixed schema upfront. This is particularly useful when dealing with diverse and evolving data sources.

3. **Scalability:** Data lakes can scale out easily by adding more storage capacity, making them suitable for handling large volumes of data.

4. **Data Exploration:** Data lakes are often used for data exploration, as they provide a repository for raw data that data scientists and analysts can work with to uncover insights.

Data Warehouse:

A data warehouse, on the other hand, is designed for efficient querying and analysis of structured data. It's organized with a predefined schema and optimized for complex querying and reporting. Here are some key points about data warehouses:

1. **Structured Data:** Data warehouses are designed to store structured data in a consistent and organized manner.

2. **Schema Design:** Data warehouses require a defined schema before data is loaded, ensuring data consistency and integrity.

3. **Performance:** Data warehouses are optimized for querying, reporting, and analyzing data, often involving complex operations.

4. **Aggregation:** Data warehouses often store aggregated and summarized data, making them well-suited for business intelligence and reporting.

In summary, a data lake is more like a repository for diverse and raw data, allowing for flexibility in data storage and exploration. A data warehouse, on the other hand, is focused on providing structured and organized data optimized for efficient querying and reporting. Organizations often use both data lakes and data warehouses in conjunction to meet different data storage and analysis needs.

ADL
Ingest->Prepare->store->analyze

unstructured data.
- **Integration:** Seamlessly integrates with other Azure services and tools like Azure Databricks, Azure HDInsight, and Azure Data Factory.
- **Security:** Provides advanced security features like Azure Active Directory integration, encryption, and access control.
- **Management:** Managed by Microsoft, so you don't need to worry about hardware provisioning and maintenance.
- **Scalability:** Easily scales storage capacity up or down based on your needs.

Hadoop Distributed File System (HDFS):

- **Platform:** HDFS is the distributed file system of the Apache Hadoop ecosystem, commonly used in on-premises or private cloud environments.
- **Storage Type:** It's designed for storing and managing large datasets across a cluster of commodity hardware.
- **Integration:** Works with Hadoop-based tools like MapReduce and Hive.
- **Security:** Basic security mechanisms exist, but they might require additional configurations or third-party tools for more advanced security features.
- **Management:** Requires management of the underlying infrastructure and hardware.
- **Scalability:** Scales by adding more machines to the Hadoop cluster.

Key Considerations:

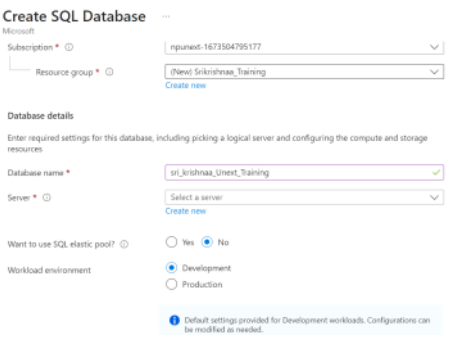
- **Deployment:** Azure Data Lake Storage is suitable for cloud-based deployments, while HDFS is often used in on-premises or private cloud environments.
- **Integration:** Azure Data Lake Storage seamlessly integrates with Azure services, while HDFS is tightly integrated with the Hadoop ecosystem.
- **Management:** Azure Data Lake Storage offers managed services, while HDFS requires more hands-on management.
- **Scalability:** Both systems offer scalability, but Azure Data Lake Storage offers the cloud's elasticity for easier scalability.

In summary, the choice between Azure Data Lake Storage and HDFS depends on your environment (cloud vs. on-premises), existing tools and technologies, integration needs, security requirements, and the level of management you're comfortable with.

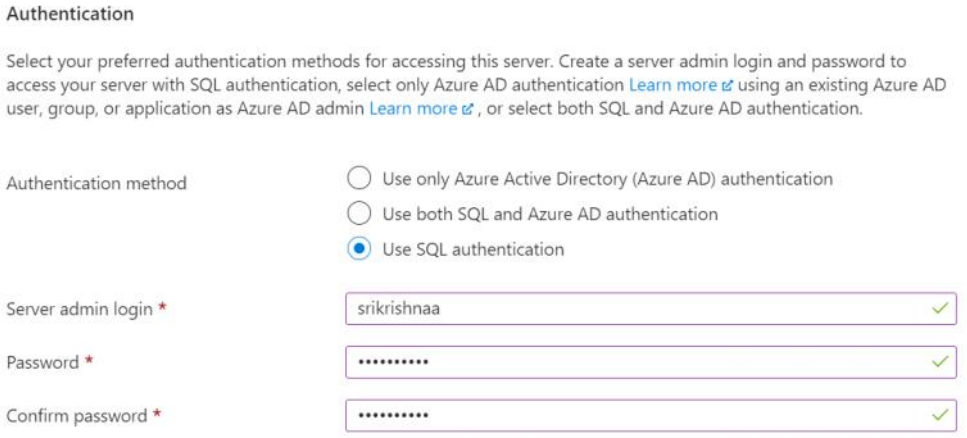
Azure sql-1

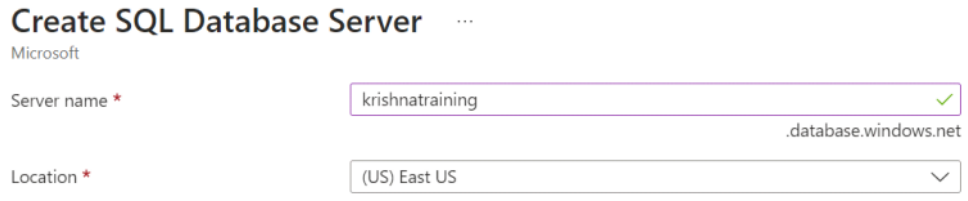
Thursday, August 31, 2023 9:10 AM

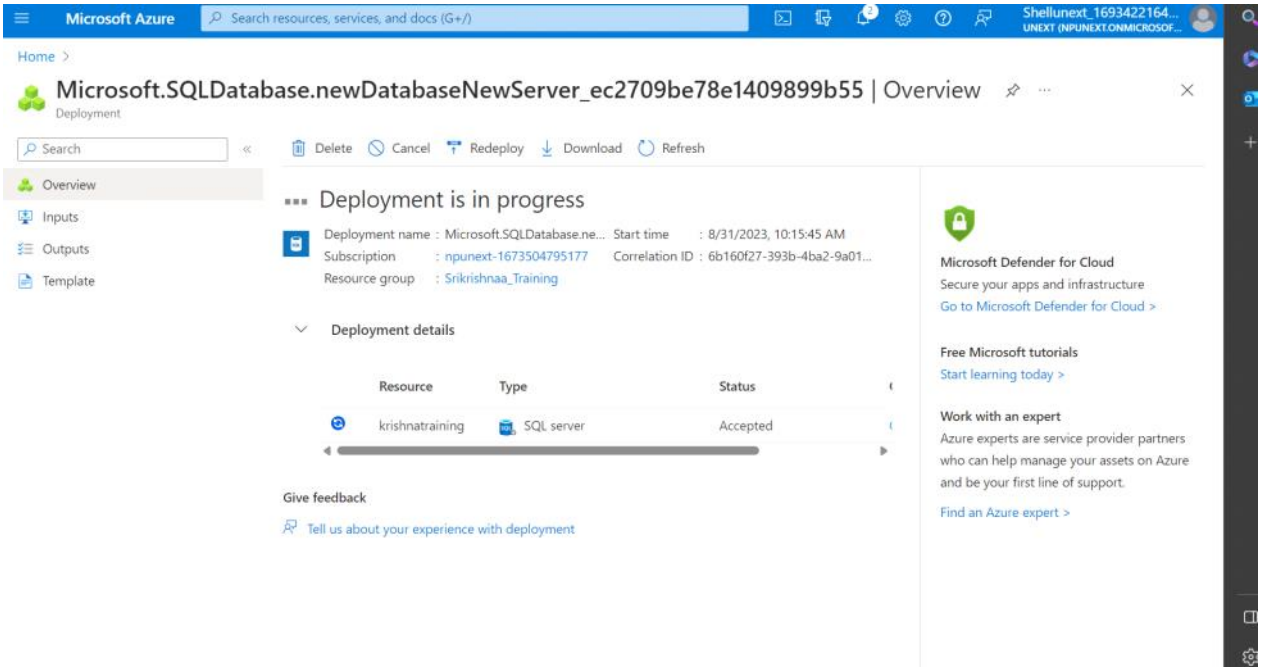
Resource group
Group databases-custom name

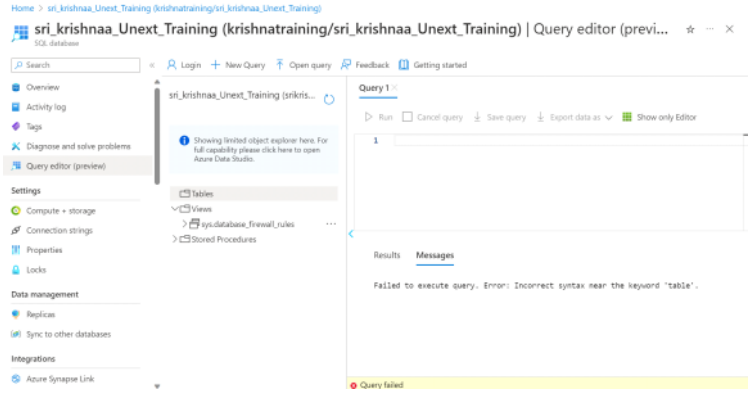


Pass-Krishna@24









Azure sql
Sp_tables---->all tables
Sp_columns <table_name>--->columns of ur table

Create new table and add content

```
select * into students2 from students
insert into students2 select * from students
```

Giving default value

```
create table students6(id int,
                       name varchar(30),
                       city varchar(30) default 'Mumbai')
```

Inserting values

```
insert into students6(id,name) values(1,'ram')
insert into students6(id,name) values(2,'sham')
insert into students6 values(3,'susil','chennai')
```

Ddl
Create
Alter
Drop
truncate

```
create table student7(id int identity(1,1),
                     name varchar(30));
```

Identity(1,1)-->start from 1 increment by 1

Cannot explicitly add values when identity is mentioned

```
insert into student7(name) values('ronaldo')
insert into student7(name) values('messi')
```

```
alter table student7 add city varchar(30)

alter table student7 drop column email
```

Increase size of name

```
alter table student7
alter column name varchar(100)
```

Sp_rename 'student7.name',student_name,'column'--->renaming name to student_name

Unique-->allows nukk

CREATE TABLE Employees (

Unique-->allows nukk
Primary key--->does not allow null
Check contraint
Foreign key

```
CREATE TABLE Employees (
   ID INT PRIMARY KEY,
   Name VARCHAR(100),
   Age INT CHECK (Age >= 18 AND Age <= 65)
);
```

```
create table customer( id int
                       name varchar(30) not null
                       city varchar(30) not null)
```

Multiple column values as primary key

```
CREATE TABLE EMPLOYEE1(ID INT
                       NAME VARCHAR(30),
                       SAL INT NOT NULL,
                       CONSTRAINT EMP1_PK PRIMARY KEY(ID,NAME))
```

Alter table employees drop constraint emp_pk

```
create table employees2(id int,
                        name varchar(30),
                        sal int constraint emp_ch
k
                        check(sal>5000))
```

Referencing foreign key

```
create table suppliers(supplier_id int primary key,
                       supplier_name varchar(30));
create table prod(product_id int primary key,
                  product_name varchar(30),
                  supplier_id int constraint prod_fk foreign key
                  references suppliers(supplier_id))
```

syntax

Thursday, August 31, 2023 11:47 AM

Ddl
Create
Alter
Drop
Truncate-deletes all rows,cannot be rollback
Delete-u can give condition and can have rollback

DCL
Grant
revoke

TCL
Commit
Rollback
savepoint

Dml
Insert
Update

Absolutely, let's walk through a practical example using a simple database table. We'll consider a scenario involving a "Customers" table where we're updating account balances. For simplicity, the table contains customer names and their account balances.

Assume the following initial state of the "Customers" table:

Customer ID	Customer Name	Account Balance
1	Alice	$100
2	Bob	$150
3	Carol	$200

Now, let's use the TCL commands in the context of a money transfer transaction:

1. COMMIT:
Suppose Alice wants to transfer $50 to Bob. Here's how the steps involving `COMMIT` might look:

1. Start Transaction:
   - Begin the transaction.

2. Deduct from Alice:
   - Update Alice's account balance: $100 - $50 = $50.

3. Add to Bob:
   - Update Bob's account balance: $150 + $50 = $200.

4. Commit Transaction:
   - Execute `COMMIT` to permanently save these changes.

After the `COMMIT` command, the "Customers" table will be updated:

Customer ID	Customer Name	Account Balance

1	Alice	$50
2	Bob	$200
3	Carol	$200

2. ROLLBACK:
Now, let's consider a situation where an error occurs before the `COMMIT` and we need to use `ROLLBACK`:

1. Start Transaction:
  - Begin the transaction.

2. Deduct from Alice:
  - Update Alice's account balance: $100 - $50 = $50.

3. ERROR Occurs:
  - An error prevents the addition to Bob's account.

4. Rollback Transaction:
  - Execute `ROLLBACK` to undo the changes made in the current transaction.

After the `ROLLBACK` command, the "Customers" table will remain in its original state, as the changes were canceled.

Remember, `COMMIT` makes changes permanent, while `ROLLBACK` undoes the changes made during the current transaction.

3. SAVEPOINT:
A `SAVEPOINT` allows you to set a point within a transaction to which you can later roll back. For example, let's say you want to perform multiple updates in a single transaction. You can set a `SAVEPOINT` before each update, and if an error occurs, you can roll back to the specific savepoint instead of undoing the entire transaction.

In this example, imagine you're updating both Alice's and Bob's accounts. You could set a `SAVEPOINT` after Alice's update, then proceed with Bob's update. If there's an issue with Bob's update, you can `ROLLBACK` to the savepoint after Alice's update, preserving the changes made to Alice's account.

SQL-3

Friday, September 1, 2023 9:37 AM

INITIAL

```
DROP TABLE EMP
DROP TABLE DEPT
DROP TABLE BONUS
DROP TABLE SALGRADE
DROP TABLE DUMMY
CREATE TABLE EMP
(EMPNO NUMERIC(4) NOT NULL,
ENAME VARCHAR(10),
JOB VARCHAR(9),
MGR NUMERIC(4),
HIREDATE DATETIME,
SAL NUMERIC(7, 2),
COMM NUMERIC(7, 2),
DEPTNO NUMERIC(2))
INSERT INTO EMP VALUES
(7369, 'SMITH', 'CLERK', 7902, '17-DEC-1980', 800, NULL, 20)
INSERT INTO EMP VALUES
(7499, 'ALLEN', 'SALESMAN', 7698, '20-FEB-1981', 1600, 300, 30)
INSERT INTO EMP VALUES
(7521, 'WARD', 'SALESMAN', 7698, '22-FEB-1981', 1250, 500, 30)
INSERT INTO EMP VALUES
(7566, 'JONES', 'MANAGER', 7839, '2-APR-1981', 2975, NULL, 20)
INSERT INTO EMP VALUES
(7654, 'MARTIN', 'SALESMAN', 7698, '28-SEP-1981', 1250, 1400, 30)
INSERT INTO EMP VALUES
(7698, 'BLAKE', 'MANAGER', 7839, '1-MAY-1981', 2850, NULL, 30)
INSERT INTO EMP VALUES
(7782, 'CLARK', 'MANAGER', 7839, '9-JUN-1981', 2450, NULL, 10)
INSERT INTO EMP VALUES
(7788, 'SCOTT', 'ANALYST', 7566, '09-DEC-1982', 3000, NULL, 20)
INSERT INTO EMP VALUES
(7839, 'KING', 'PRESIDENT', NULL, '17-NOV-1981', 5000, NULL, 10)
INSERT INTO EMP VALUES
(7844, 'TURNER', 'SALESMAN', 7698, '8-SEP-1981', 1500, 0, 30)
INSERT INTO EMP VALUES
(7876, 'ADAMS', 'CLERK', 7788, '12-JAN-1983', 1100, NULL, 20)
INSERT INTO EMP VALUES
(7900, 'JAMES', 'CLERK', 7698, '3-DEC-1981', 950, NULL, 30)
INSERT INTO EMP VALUES
(7902, 'FORD', 'ANALYST', 7566, '3-DEC-1981', 3000, NULL, 20)
INSERT INTO EMP VALUES
(7934, 'MILLER', 'CLERK', 7782, '23-JAN-1982', 1300, NULL, 10)
CREATE TABLE DEPT
(DEPTNO NUMERIC(2),
DNAME VARCHAR(14),
LOC VARCHAR(13) )
INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK')
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS')
INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO')
```

```
INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON')
CREATE TABLE BONUS
(ENAME VARCHAR(10),
JOB VARCHAR(9),
SAL NUMERIC,
COMM NUMERIC)
CREATE TABLE SALGRADE
(GRADE NUMERIC,
LOSAL NUMERIC,
HISAL NUMERIC)
INSERT INTO SALGRADE VALUES (1, 700, 1200)
INSERT INTO SALGRADE VALUES (2, 1201, 1400)
INSERT INTO SALGRADE VALUES (3, 1401, 2000)
INSERT INTO SALGRADE VALUES (4, 2001, 3000)
INSERT INTO SALGRADE VALUES (5, 3001, 9999)
CREATE TABLE DUMMY
(DUMMY NUMERIC)
INSERT INTO DUMMY VALUES (0)
```

SQL-4

Friday, September 1, 2023 10:22 AM

ORDER

```
select * from emp order by ename

select * from emp order by ename DESC


select * from emp order by deptno,sal desc
```

COUNT

```
select count(*) as numofrecord from emp
```
NULL is not part of count

```
Select top 5

select top 5 * from emp
```

WHERE OPERATORS
=
!=
>
<
BETWEEN-->INCLUDES BOTH
NOT BETWEEN
IS NULL
IS NOT NULL
LIKE
NOT LIKE

Single row functions

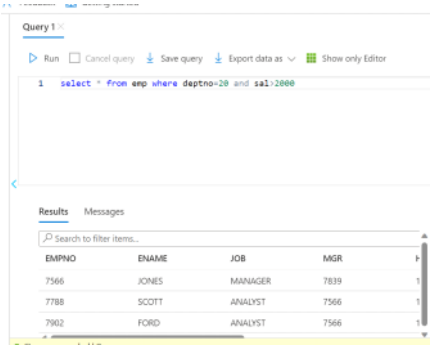
Applies for all rows
String function
Number function
Date function
Conversion function
Analytical function

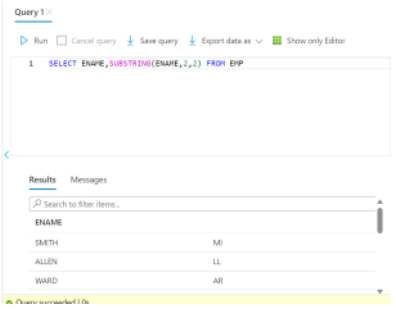
1)ASCII
Select ascii('B')
2)LEN(ENAME)

2)CONCAT

```
SELECT ENAME,CONCAT('HI',ENAME,'how are you')
 AS ENAME1 FROM EMP
```
Concat_ws('|','.')
Concat with seperator

```
SELECT ENAME,LEFT(ENAME,3) FROM EMP---->
```
EXTRACTS 3 CHARACTERS FROM LEFT

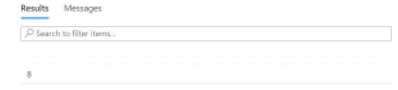





```
select NULLIF(ENAME,'SMITH') ENAME_O FROM EMP
```
REPLACES SMITH WITH NULL

```
SELECT SAL,IIF(SAL>3000,'GOOD SAL',IIF(SAL >2000,'AVG SAL','POOR')) SAL_DE
SC FROM EMP
```


SELECT UPPER('BABJEE')--->CONVERTS TO UPPERCASE
LTRIM(' BABJEE')
RTRIM('BABJEE ')
SELECT TRIM(' BABJEE ')
SELECT ABS(-1)
SELECT CEILING(25.75)
SELECT FLOOR (25.75)

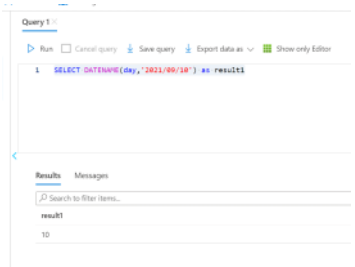



```
select empno,ename,sal, RANK() over(ORDER BY SAL) as sal_rank from emp
```

SELECT GETDATE()

DENSE_RANK()--->1 1 2 INSTEAD OF 11 3

```
SELECT HIREDATE,YEAR(HIREDATE) FROM EMP WHERE YEAR(HIREDATE)=1981
```
YEAR
MONTH
DAY




```
select empno,ename,sal,e.deptno,dname,loc from emp e
inner join dept d
on e.deptno=d.deptno
```

SUBQUERY

```
select * from emp o where sal > (select avg(sal) from emp i where o.deptno=i.dept
no )
```

Employees having salary>avg salary of their own dept

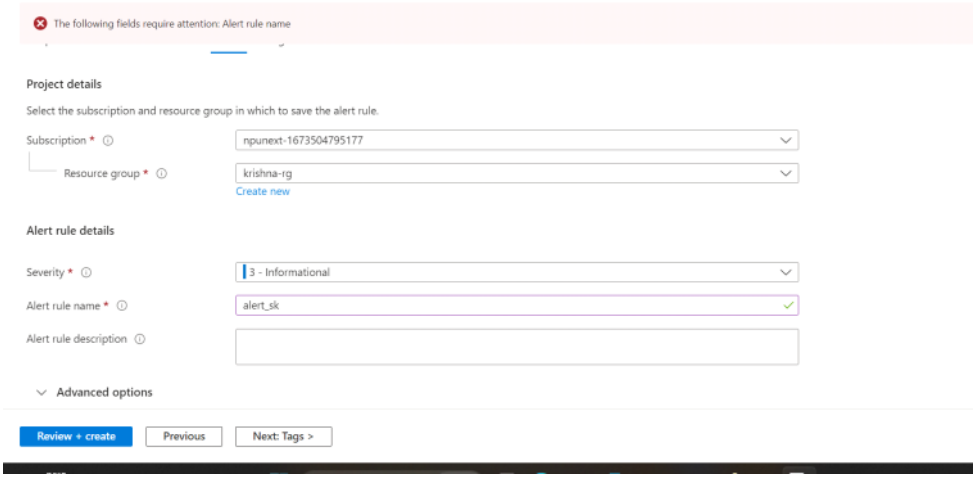
Non aggregated column in select list should be in group by clause

Begin transaction

rollback

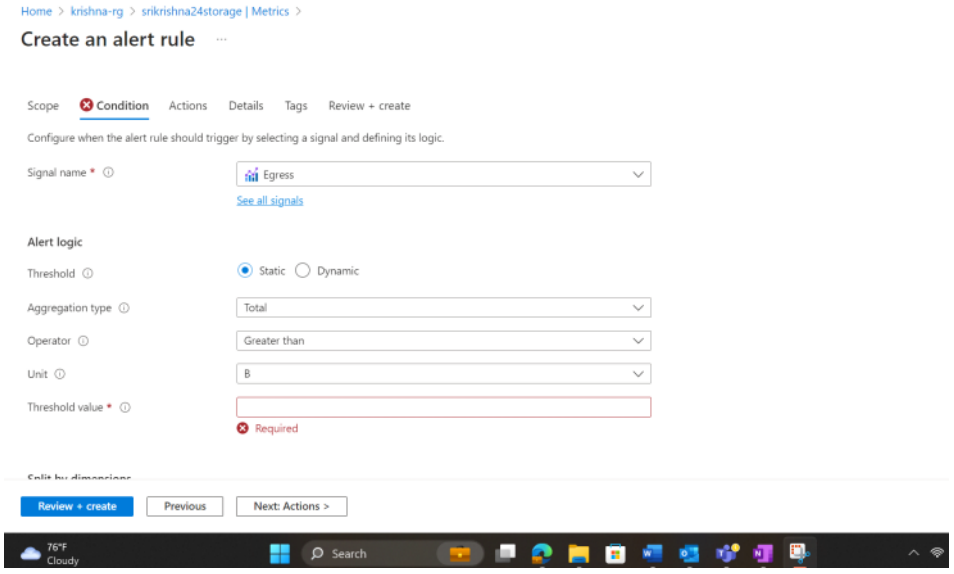
Azure storage-2

Tuesday, September 5, 2023 9:21 AM



Smb server--->client(AZURE STORAGE)

Remoting-->rest-->nlo



Vm+storage+rg

Tuesday, September 5, 2023 12:06 PM

Delete rg
Create new rg
Virtual machine create
Image-windows os,username pass
Put all ports enabled
Licensing click ok
Create
Win+r-->mstsc

Azure Data Factory

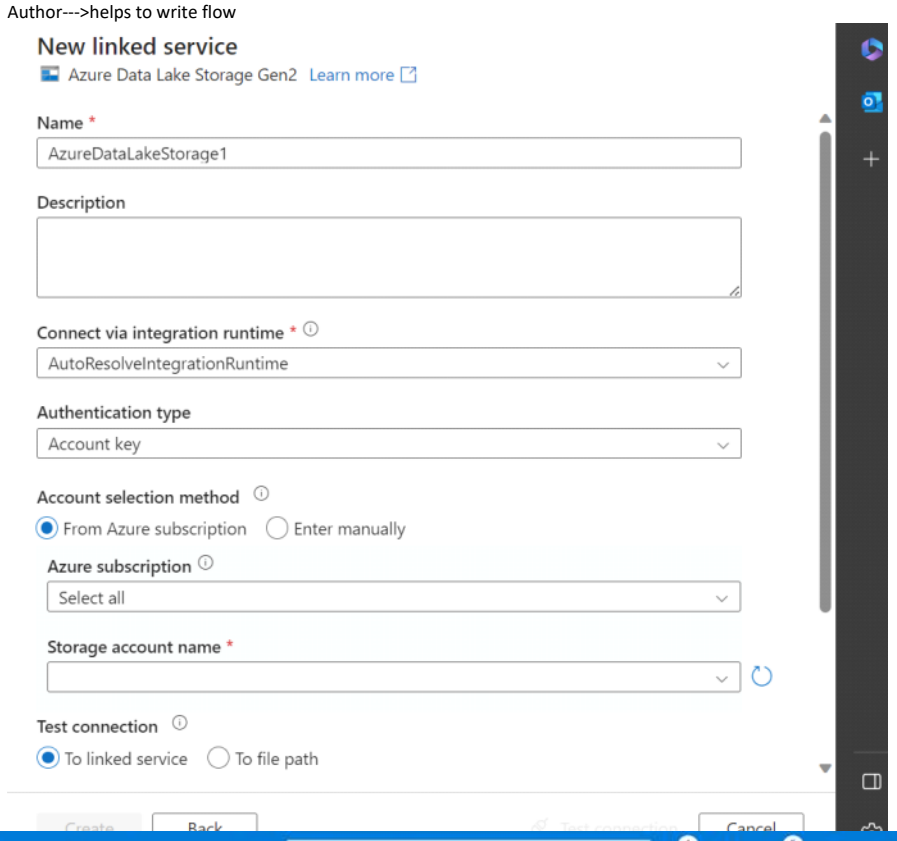
Tuesday, September 5, 2023 2:08 PM

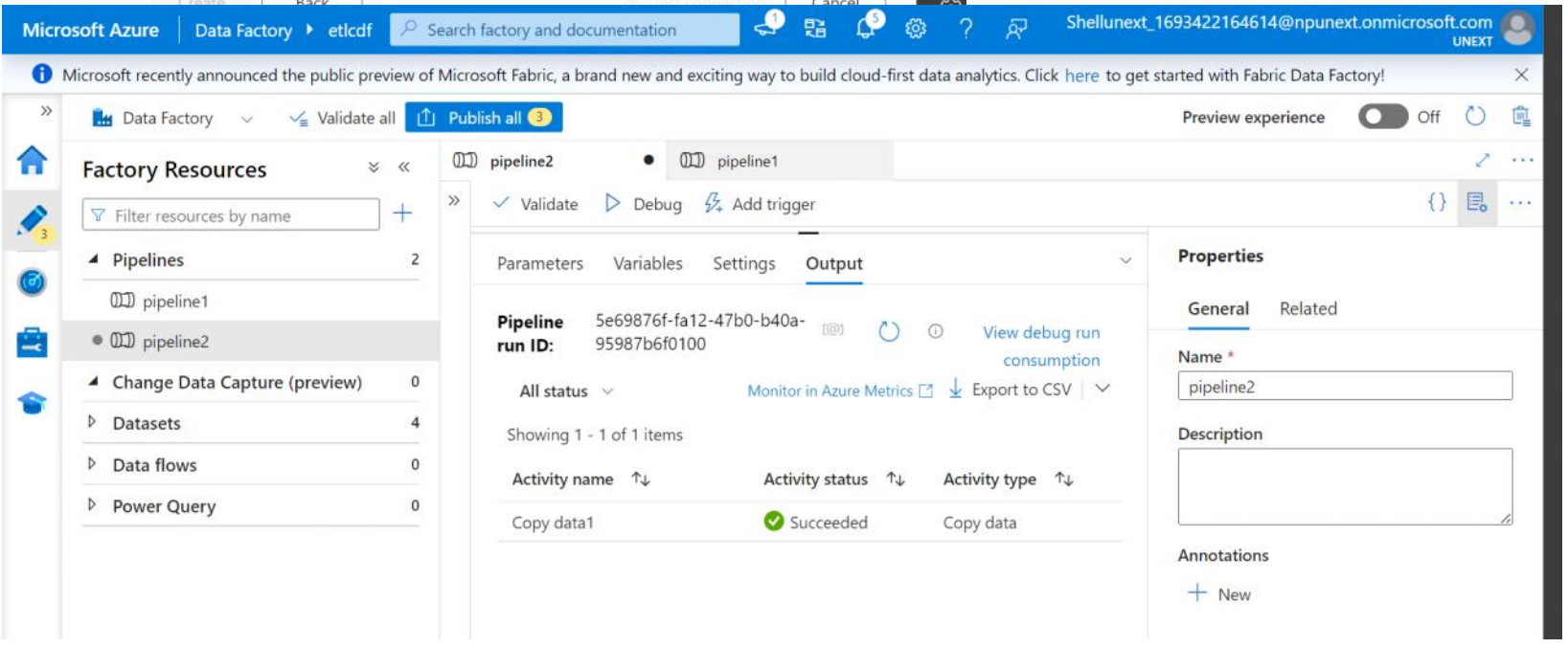
Source---->data integration----->sink
Serverless
cloud

Infinite program--->server
Physical server---->workstation
rack
Blade(bear metal)
Logical server----->host
Vm

Adf
1)linked service--->helps to connect diff service
2)data sources

Author--->helps to write flow





Annotations

+ New

Adf-3

Wednesday, September 6, 2023 9:49 AM

Bash

az login

CloudVendorCode(az)
ServiceName(ResGrp) \
FeatureName[] \
OperationName(Create/Delete/Upload)\
Properties(--Key Value)

Az group create\
--name <name of rg>\
--location eastus

Az \
Group \
Create

Copy-Ctrl
+Ins

Paste-->Shift+Ins

##create storage
Az storage account create \
--resource-group krishnabashrg \
--name bashstoragekrishna \
--location eastus

##create storage container

az storage container create \
-- resource-group krishnabashrg \
--account-name bashstoragekrishna \
--name container1 \
--auth-mode key

##create file

Cat > emp.txt

Ctrl+d to save

##upload storage container
Az storage blob upload \
--resource group \
--account-name \
--container-name \
--name input/emp.txt
--file emp.txt\
--auth-mopde key

```
  "homeTenantId": "dce87315-8ffa-4a01-ab40-0de5a7214b2f",
  "id": "64430eab-e6d8-4fa5-b988-5eab3a8f97cf",
  "isDefault": true,
  "managedByTenants": [],
  "name": "npunext-1673504795177",
  "state": "Enabled",
  "tenantId": "dce87315-8ffa-4a01-ab40-0de5a7214b2f",
  "user": {
    "name": "Shellunext_1693422164614@npunext.onmicrosoft.com",
    "type": "user"
  }
}
shellunext [ ~ ]$ az group create\--name krishnabashrg\--location eastus
'create--name' is misspelled or not recognized by the system.

Examples from AI knowledge base:
https://aka.ms/cli_ref
Read more about the command in reference docs
shellunext [ ~ ]$ az group create --name krishnabashrg --location eastus
{
  "id": "/subscriptions/64430eab-e6d8-4fa5-b988-5eab3a8f97cf/resourceGroups/krishnabashrg",
  "location": "eastus",
  "managedBy": null,
  "name": "krishnabashrg",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
shellunext [ ~ ]$
```

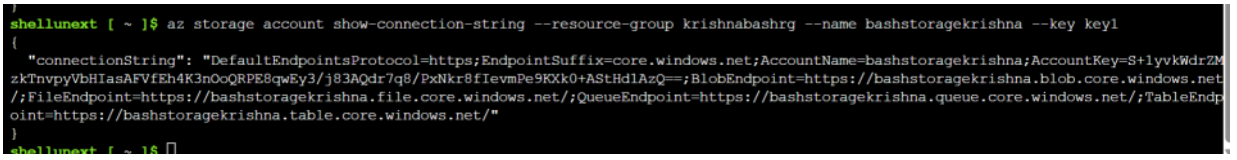
```
clouddrive  emp.txt
shellunext [ ~ ]$ az storage blob upload --account-name bashstoragekrishna --container-name container1 --name input/emp.txt --file emp.txt
--auth-mode key

There are no credentials provided in your command and environment, we will query for account key for your storage account.
It is recommended to provide --connection-string, --account-key or --sas-token in your command as credentials.

You also can add `--auth-mode login` in your command to use Azure Active Directory (Azure AD) for authorization if your login account is as
signed required RBAC roles.
For more information about RBAC roles in storage, visit https://docs.microsoft.com/azure/storage/common/storage-auth-aad-rbac-cli.

In addition, setting the corresponding environment variables can avoid inputting credentials in your command. Please use --help to get more
 information about environment variable usage.
Finished[#############################################################]  100.0000%
{
  "client_request_id": "cc7f8f0c-4c72-11ee-b460-0a4f4dd248ea",
  "content_md5": "n03+3iN4YlY8nZpbf77Lyw==",
  "date": "2023-09-06T05:04:03+00:00",
  "encryption_key_sha256": null,
  "encryption_scope": null,
  "etag": "\"0x8DBAE96B0BD9086\"",
  "lastModified": "2023-09-06T05:04:03+00:00",
  "request_id": "c6d843c2-001e-0033-397f-e03555000000",
  "request_server_encrypted": true,
  "version": "2022-11-02",
  "version_id": null
}
shellunext [ ~ ]$
```

##create adf workspace

Az datafactory create \
--resource-group

```
shellunext [ ~ ]$ az datafactory create --resource-group krishnabashrg  --name adf06septkrishna
{
  "additionalProperties": null,
  "createTime": "2023-09-06T05:12:03.898936+00:00",
  "eTag": "\"8500d119-0000-0100-0000-64f80a230000\"",
  "encryption": {
    "identity": null,
    "keyName": null,
    "keyVersion": null,
    "vaultBaseUrl": null
```

Show connection string to create json file



{
"connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=bashstoragekrishna;
AccountKey=S+
1yvkWdrZMzkTnvpyVbHIasAFVfEh4K3nOoQRPE8qwEy3/j83AQdr7q8/PxNkr8fIevmPe9KXk0
+AStHd1AzQ==;BlobEndpoint=https://bashstoragekrishna.blob.core.windows.net/;FileEndpoint=https://
bashstoragekrishna.file.core.windows.net/;QueueEndpoint=https://bashstoragekrishna.queue.core.win
dows.net/;TableEndpoint=https://bashstoragekrishna.table.core.windows.net/"
}

Create Linked Service from json file:
Az datafactory linked-service create
--resource-group
--factory name
--linked-service-name LSStorageAccountGen2
--properties @Azure.....json

Add dynamic content below using any comb

```
@not(equals(item(),'ccc'))
```

conditions

items

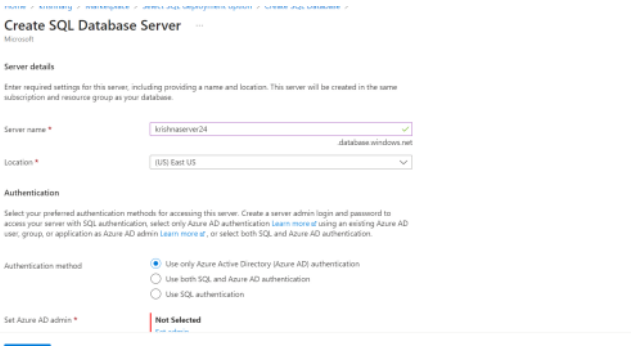
@variables('DataArray')

```
{
    "ItemsCount": 4,
    "FilteredItemsCount": 3,
    "Value": [
        "aaa",
        "bbb",
        "ddd"
    ]
}
```

Data flow in azure

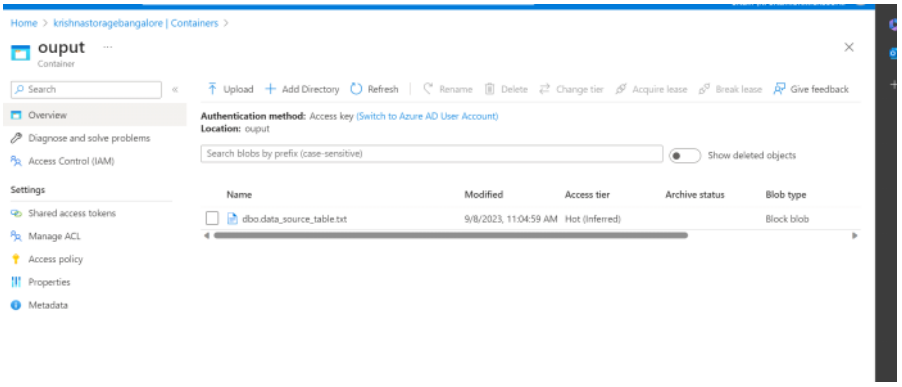
Thursday, September 7, 2023 10:18 AM

Custom activity(instead of copy filter etc)--->data flow



Krishna
krishpappi@24

krishnadatabasefriday



Change Data Capture (CDC) and watermark procedures are commonly used in data warehousing and ETL (Extract, Transform, Load) processes to track and manage changes in source data.

CDC (Change Data Capture):

Purpose: CDC is a technique used to identify and capture changes (inserts, updates, deletes) made to source data over time. It helps to maintain a historical record of changes for analysis and reporting.

Implementation: CDC is typically implemented at the source database level. Databases that support CDC maintain change tables or logs that record modifications to the data.

Usage: It is commonly used in scenarios where you need to keep a track record of changes, such as in data warehousing, data synchronization, and auditing.

Process: CDC processes periodically poll or subscribe to the change logs or tables in the source database to identify and capture changes. These changes are then propagated to a destination database or system.

Granularity: CDC captures individual changes, allowing you to reconstruct the history of each modified row.

Example: Suppose you want to maintain a historical record of all customer orders in an e-commerce system. CDC would capture changes to order data, including new orders, updates to existing orders, and order cancellations.

Watermark Procedure:

Purpose: A watermark procedure is used to keep track of the most recent or highest timestamp (or some other identifier) of data that has been processed. It helps to avoid reprocessing the same data during ETL or data integration tasks.

Implementation: Watermarks are typically implemented as a timestamp, version number, or some other indicator that signifies the point up to which data has been processed.

Usage: Watermark procedures are commonly used in ETL processes to ensure that only new or changed data is processed and loaded into a data warehouse or another target system.

Process: When new data arrives, the watermark procedure checks its timestamp or identifier against the watermark value. Data with a higher timestamp or identifier is considered new and is processed, while older data is skipped.

Granularity: Watermark procedures are typically used to track data at a batch or block level, indicating the point up to which a batch has been processed.

Example: In a daily ETL process, a watermark procedure might keep track of the highest date in the source data that has been processed. The next day, it would only process data with dates later than the watermark value from the previous day.

In summary, CDC is used to capture and track individual changes in source data, while a watermark procedure is used to track the highest or most recent point of data processed to avoid reprocessing the same data during ETL or data integration tasks. Both are essential techniques in data integration and data warehousing to ensure data accuracy and efficiency.

Azure synapse

Tuesday, September 12, 2023 12:12 PM

Data

- Live
- Dead
  - Traditional
    - Sql,db,dwh
  - big data
  - new data

Localhost@123
sqladminuser

When to use a synapse

When u need a managed service
When u have a large dataset and complex queries
Manage structure and unstructured ds
Data pipeline orchestration
Analytics on real time data

Serverless sql pool
Pay as you go model

Dedicated compute resources which allow you to control and optimize
Performance using dwu(data warehousing unit)

Powerbi

Wednesday, September 13, 2023 9:50 AM

Data Centre
Storage
Compute
N/W
Security

PBI Variations
1)DESKTOP
No knowledge req easy to implement

2)SERVICE
Requires base knowledge
Allows sdk and api integration
Allows CLI

3)MOBILE
Lightweight s/w

MDF
Media descriptor file
First option--->Rows data in a new database creation

pyspark

Wednesday, September 20, 2023 9:32 AM

Aws-elastic map reduce
Databricks supports all this

JOB	JOB2
STAGE	STAGE
STAGE	STAGE

Lifecycle of **spark**
1)submit appl
2)initialize-when appl is submitted spark program launch on a clustered node
Driver program initializing spark context--->for coordinating the job executions
3)job and stage creation

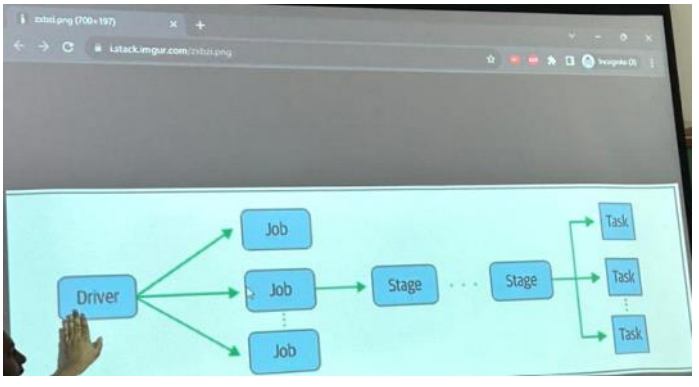
Spark appl--->divided into one more jobs
--->Each job consists of stages--->
stage is created for each RDD transformation that req data shuffling--->
DAG generation (directed acyclic graph)--->spark makes a DAG that represents logical execution plan of ur appl----->Dag is basically plan rep/workflow--->
Dag contains info about dependencies between stages and task----->
Task scheduling---->
schedular takes(input(Dag)--->schedules task for exec)
-->execution--->
tasks are executed on worker node in parallel/distributed node
---->shuffling/data exchange
task completion
-->as task completes they produce immediate results
--->these results are cached/persisted in mem for stages to use which reduces the need for recompilation
-->stages are complete when all task is successful
-->job completion

For each rdd--->job

Stage--->map function and filter

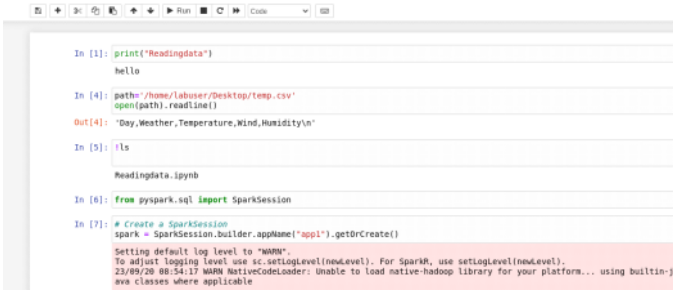
Task
Thread+executor
A single operation applied to a single partition
Its executed as a single thread in an executor

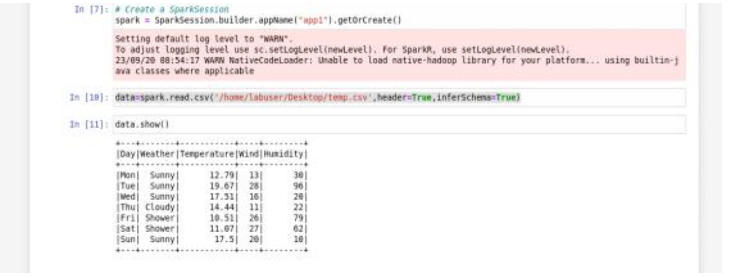
Batch parallel
unified analytics
Real time
Streaming
Graph orientated operations



Spark-Read

Wednesday, September 20, 2023 2:24 PM



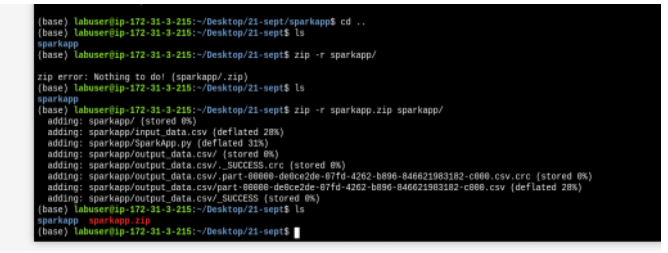


Spark is object

```
from pyspark.sql import SparkSession

spark=SparkSession.builder.appName("MySparkApp").getOrCreate()

df=spark.read.csv("/home/labuser/Desktop/temp.csv",header=True,inferSchema=True)

df.write.csv("output_data.csv",header=True)

spark.stop()
```

Write in this file
>>--append

Cp command to copy



Submit

Spark-submit \
--master spark://MasterIP:7077
(submit to master node)

```
from pyspark.sql import SparkSession

spark=SparkSession.builder.appName("RDDMapExample").getOrCreate()
data=[1,2,3,4,5]
rdd=spark.sparkContext.parallelize(data)

def double(x):
        return x*2

result_rdd=rdd.map(double)
result=result_rdd.collect()
print(result)
spark.stop
```

Rdd operations

ACTIONS		TRANSFORMATIONS
Collect		Map
First		Reduce
Reduce		reducebykey
Count saveastext		JOIN

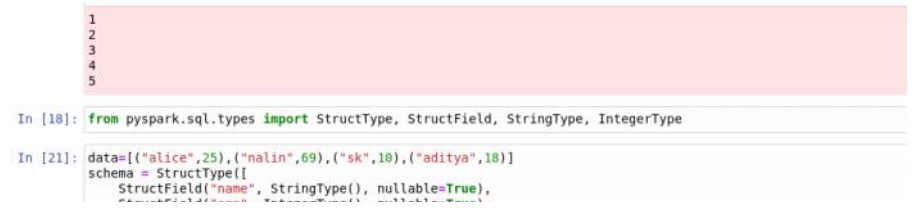
Actions: are options on RDD that triggers
Execution of spark computation
Therefore the evaluation of transformations and produce as a result(perform an action)
Eagerly executed
Used to retrieve resultws,save data and triggers side effects in sparks

Transformation: are opn on RDD that create new RDD result
Are lazy evaluated
Define data pipeline sequences

Why rdd
Perf optimisation
Fault tolerant
In memory processing

Spark-context

- Main entry point
- Consists of all basic functionality of spark
- Spark driver contains DAG scheduler, task scheduler, backend scheduler, Block managers, which are responsible for translating the user written code into jobs That are actually executed on the cluster



Why rdd
Perf optimisation
Fault tolerant
In memory processing
Data sharing
Compatibility
Integrate with external storage

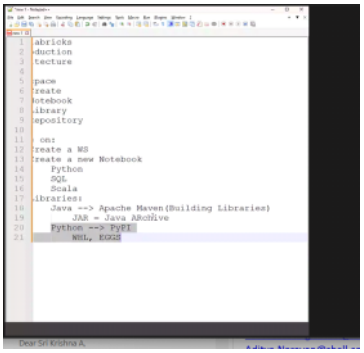
```
In [18]: from pyspark.sql.types import StructType, StructField, StringType, IntegerType

In [21]: data=[("alice",25),("nalin",69),("sk",10),("aditya",18)]
         schema = StructType([
             StructField("name", StringType(), nullable=True),
             StructField("age", IntegerType(), nullable=True),
         ])

In [22]: from pyspark.sql import SparkSession

In [23]: spark=SparkSession.builder.appName('a').getOrCreate()

In [ ]: rdd=spark.sparkContext.parallelize(data)
```

Creating session and rdd

Transformation
1)Narrow transf

Databricks

Tuesday, September 26, 2023 9:19 AM



Azure Databricks
Analytical platform
You can work
Building
Deploying
Sharing
Enterprise data maintain

Supports multiple cloud vendors like aws,azure,gcp+community

Used for
Ml+ai at scale
Big data
Data processing(sql,nosql,iot)

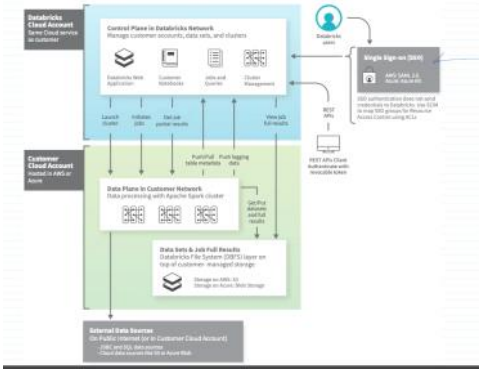
For DE:

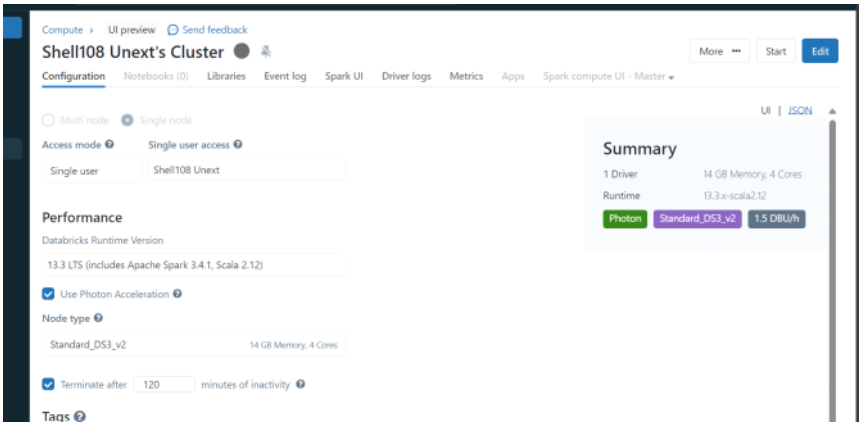
```
* Data Engineers:
    > Data Ingestion
    > ETL
    > Compute Management
    > SQL, PySpark, etc.
    > Data Discovery
    > Data Annotation
```

HOW TO USE DB:
Conceptual-->web ui
Integration--->rest api
Automation--->cli

```
1) Data Engineers
    Ingestion
    Data Management
```

Architecture





Wednesday, August 30, 2023 10:02 AM

Krishna
Pappi@24