25 Oct

Day 20

AZURE

① Create a Resource Group

↓

- A container that holds resources for any Azure solution like Virtual machines, databases, accounts.
- It helps managing & organising resources as a SINGLE UNIT,
  - ↳ allowing you to deploy, monitor, control access costs, lifecycle (create, manage, delete) together

Ex WebApp-RG
- A VM
- storage account
- network

If u delete RG → all 3 resources gets deleted automatically

∴ Shared lifecycle

② Create a resource now, from inside RG

1. Storage Account

- A container that holds all your storage data - blobs, files, tables

Azure Blobs → Cloud based object storage solution for unstructured data
- Can store vast amount of data like images, videos, audio
- highly scalable & cost effective

Azure Data Lake → Built on top of blob
- Optimized for big data analytics with a hierarchical namespace
  - ↳ organizes data like a file system not a blob

Normal Blob: → Virtual Hard drive in cloud.

- Path ↳ All files are stored as objects, just keys & paths. No true folders.

With HNS: Create folders within folders
↳ We can have real directories, subdirectories

∴ Data lake storage Gen 2
- file system + folder organization on top of the hard drive

→ Redundancy Options

LRS (Locally redundant storage)
- lowest cost
- Data is replicated 3 times within a single datacenter.

Why 3? → If 2: If 1 fails, only 1 remains so high risk of data loss
If 3+: Extra cost

∴ Many cloud providers use triple replication as a balance of cost, reliability, availability

Region: A geographic area, where Azure data centers are located Ex East US / Canada West
∪ choose region based on latency

↓

Availability Zone: A physically separated datacenters within a Region (collection of) designed to be isolated from failure of other zones.

↓

Data Centre: Actual facility where servers & storage & networking exists

LRS

If a server or disk fails
↳ other copies in datacentre keeps us data safe
But if datacentre fails (fire / power outage)
↳ All 3 copies get affected

ZRS → Zone Redundant storage
Data is replicated across 3 zones in a region
Limitation → regional disaster

GRS → Geo-redundant storage
Data is replicated to another region

GZRS → Geo Zone redundant storage
- Highest durability & availability
- data remains safe even if entire zone/region fails.

Redundancy → keeping extra copies of data or resources → so that if something fails, system continues to work

26 Oct

Day 21

(2) Data factory

↳ Cloud based ETL (Extract, Transform, Load) service used to move & transform data from various sources to destinations, for Analytics or storage

→ Used to build data pipelines → to collect/clean data automatically.

Pipeline : is a group of activities that together perform a specific data workflow, (ETL).

SHIR Self Hosted Integration Runtime

↳ Sometimes data is not in Azure, maybe its in your company's on-premises SQL server, or private Azure Network

ADF can't reach the data as its not public

∴ So u install SHIR app (a small software agent) on the computer/server inside ur network. This agent acts as a bridge b/w your data & ADF

↳ Data needs to connect to ADF in cloud, somehow

① Public Endpoint
- SHIR connects to ADF over public internet (but securely encrypted)
- simple setup, safe (outbound traffic only (SHIR → Azure)

② Private Endpoint
- SHIR connects to ADF privately using Azure Private links
- Traffic stays 100% private
- Used when company don't allow internet communication
- High security / sensitive data

Ex :- A company store its data in onprem SQL server
① behind a firewall
② ADF can't access it directly.
③ Install SHIR on the system
(which can access SQL server data locally & send data securely to ADF)

Ex Public Endpoint → How its secure/encrypted?
Ex :- Login to bank website
its public but encrypted (https)

Environments

- On prem :- means to company's own servers, databases, & storage & are physically hosted in their own building/datacentre not in cloud.
  Ex:- your official SQL server
- cloud : Azure SQL Database
- Hybrid : onprem DB + ADF pipelines

HTTPS →

- "S" stands for secure
- Uses encryption to keep data private & safe

Day 24

AZURE

3. Storage Resource
   - Containers
     - Add Container
       - (1) Bronze (2) Silver (3) Gold
       - (1 container for each layer)

ADF → Launch
→ Configure Git
↳ Create a branch (gurtej)

4. Azure SQL
   - ↳ Create server
   - ↳ Create database → Configure
     - Region (Latency)
     - Compute / Cores
       - (More cores More Processing Power
         - ↳ Faster, Expensive)
   - ↳ Networking → Public Endpoint
   - ↳ Security
   - ↳ Data Source → Start with Blank DB or Sample
   - Database collation → String data Rules
     - Case insenstive (A = a)
     - Accent sensitive (é ≠ e)

31 Oct

Day 25

Data lake → stores raw/unstructured data
Azure SQL DB → stores cleaned, structured data ~~for dashboards~~

Raw data → Data lake → Processed Data /Run Analytics → Azure SQL → Power BI

* Incremental load : means loading only "new/changed" data instead of loading entire dataset every time

Compare Last modified Date > Last Run Time

Backfilling → means loading historical data or /which u missed / need old data
→ often do backfilling first to load everything & then → incremental load.

① Load data → DB
↳ Query Editor

(Manually creating Sample DB,
in companies → DB already exists
& we simply connect to it)
& Build pipelines for extraction
& transformation

In Real Development

We use SQL Server Management Studio (SSMS)
↳ a microsoft tool to connect, manage, query SQL DBs

Azure SQL : stores Structured data, but transactional
not optimised for Analytics
As it could contain
↳ Redundant data, nulls
only optimised for OLTP data

① (Source) OLTP (operational data)

② Data lake (Landing/Staging) : ADF copies raw data from source to Datalake
Here we store raw data
Safe & cheap

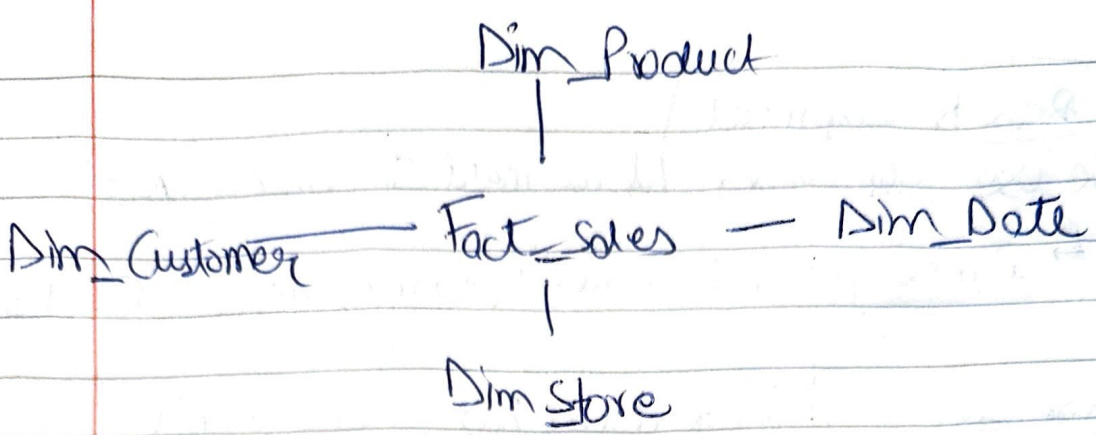
③ Databricks/ADF/Synapse Pipeline : Transformation (clean/join/Aggregate)

④ Datawarehouse (Synapse) : (load cleaned data into Fact and Dim Tables) STAR Schema
Ready for Reporting

⑤ PowerBI

- Azure Synapse Analytics : Platform for analytics AND Reporting
  Includes Data warehouse

STAR SCHEMA
↳ is a way to organise data in Data warehouse for analytics



Fact → Holds numeric data
Ex: Sale amount, quantity, ID

Dimension Table → Holds descriptive attributes that explain facts
Ex: Customer details → name, city
product info, brand
date info → day, month

Why? → faster joins, aggregations
→ makes Analytics fast & simple
→ used heavily in OLAP

In real companies, we use STAR Schema, a central fact table: holding measurable fields & surrounding Dimension Tables holding descriptive attributes

Fact Tables: Can have billion of Rows

→ most companies organise data into Fact & Dim. Tables

- ✗ OLTP : handles day to day transactions
  optimized for fast inserts/updates
  Small queries → Normalized schema
  Ex: Azure SQL stores transactional data
  Shopping cart updates / Banking deposits / Airline bookings

  OLAP : Analyzes historical data / aggregated data for insights, reporting
  - heavy queries / complex
  - Denormalized scheme (STAR)

  Ex: Synapse DW / DataBricks Analytics
  ADF pipelines / PowerBi
  Analyze sales trends, fraud detection, revenue trends, monthly transaction summary / flight revenue per route

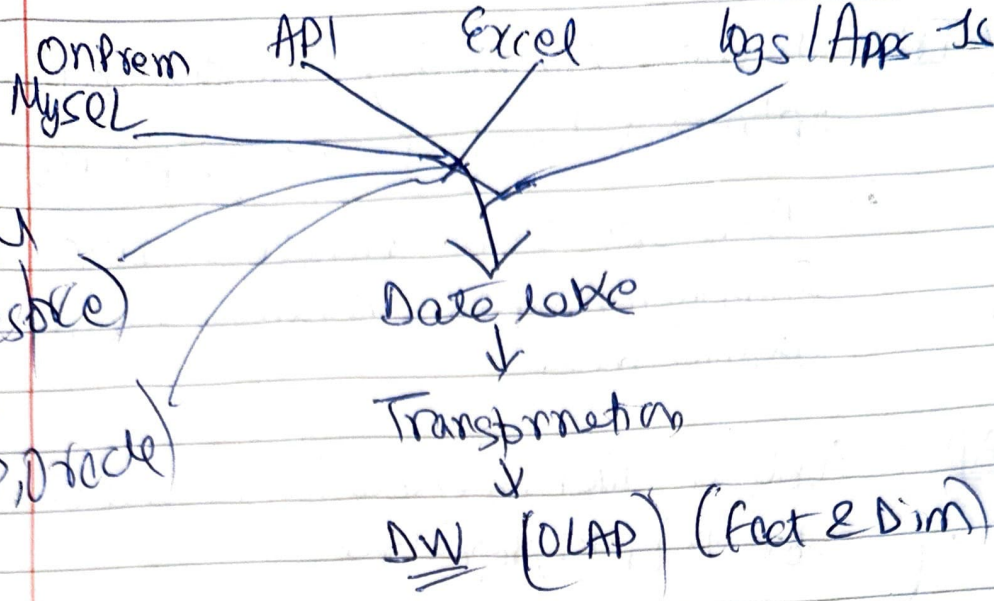
- Data comes from OLTP / or multiple sources

✗ Transactional Data → Records of business ~~state~~ events that happen in real time
→ Bank deposited 500$, Order 23 placed by cust ~~added to cart~~ or

Orders

order ID	customer ID	Product ID	Amount	Date

Each row → 1 transaction



Normalization

1. Start by analyzing source data / study it
   - Identify — Entities, Attributes, ~~Relationships~~ — using ER Model
2. Distinct entities become seperate tables — Split into multiple Tables
3. Assign unique identifiers to link tables
4. Ensure each table stores only 1 type of information (Removing Redundancy)
5. Check for Integrity
   - ↳ validate relationships b/w tables

Why denormalize for Analytics

↳ As OLAP requires joining many Tables
But joining will be very slow for million of rows
∴ Denormalized Dim/fact tables
~~reducing~~ → reduces number of joins
making query much faster

↳ Denormalization → is not messy
it is purposefully ~~star~~ flattened for Analytics
→ you control what is included
↳ for reporting only what needed

→ Combine multiple related tables into a single table
in a controlled & intentional way.
So that analytics, reporting become faster

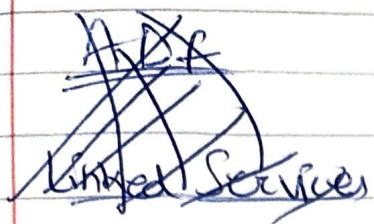
→ widen the tables, instead of joining
for fast Reading

✳ In OLTP, you heavily normalize data → Many small tables
↳ Customers, Orders, Products, Payments, Categories, Details
∴ To Analyze we need '6' Joins (:( )

∴ In OLAP/Data warehousing → we denormalize into simpler models → STAR SCHEMA
Dim
Dim — Fact — Dim → Here OLAP, joins (less) but with
Dim
fewer & larger tables → which is better for larger data

Nov 1

Day 26



Normalization Levels Hierarchy

1NF : Requirement

↓

All columns have atomic values (no repeating grps)

Atomic : one fact per column

not a list, array

Eg Alice 102, 103

↳ Split :

Alice	102
Alice	103

2NF : A table is in 2NF, if it is in 1NF AND all non-key columns fully Depends on Primary key.

Key → a col'n / multiple col'ns which uniquely identify a row

Non-key → everything else col'n : descriptive info

3NF : in 2NF AND, every nonkey column only depends on a key column, no any other non-key col'n.

This means no transitive dependency

3NF removes ↗

Ex: City depends on Region but both are Non Key so we split

BCNF : if it is in 3NF AND, every determinant must be a candidate key.

(Boyce-Codd Normal Form)

Candidate key : A col'n / set of col'n that uniquely identify each row

determinant : Any col'n / set of col'n that determine another col'n value

If a non key col'n determines another col'n
↳ ~~fails~~ its NOT BCNF

Here also to fix → we split into multiple tables so every determinant is a key in its own table

Course Table	
Course	Instructor

Instructor Table	
Instructor	Room

4NF, 5NF ... Removes further dependencies

we choose 1. Primary key from multiple Candidate keys