DP900 Final Notes – part 1

Describe core data concepts (25—30%)

Describe ways to represent data

• Describe features of structured data

Normalised data
Tables, Views, Primary Key, Foreign Keys, Joins, Schema cannot change
Adheres to ACID. (Atomicity, Consistency, Isolation, Durability)
Used for OLTP and OLAP
SQL

• Describe features of semi-structured

Most do not adhere to ACID
No complex structures, relations or indexes
Easily scale up horizontally
Fast to store (Iot)
Slow to query
Can replicate information geographically very fast
Can get data loss in replicated databases
No good at good for managing distributed transactions
No standard interface for data manipulation

• Describe features of unstructured data

Collection of unrelated files that have no key or fields that can be searched (apart from meta data)

Identify options for data storage

• Describe common formats for data files

Delimited text files
JSON (JavaScript Object Notation)
XML
BLOB (Binary Large Object)
Optimised File Format 9Columnar like Orc, Avro and Parquet)

• Describe types of databases

Relational
Good old SQL
Non-Relational
Key-value databases in which each record consists of a unique key and an associated value, which can be in any format.

Products	
Key	**Value**
123	"Hammer ($2.99)"
162	"Screwdriver ($3.49)"
201	"Wrench ($4.25)"

Document databases, which are a specific form of key-value database in which the value is a JSON document (which the system is optimized to parse and query)

Customers

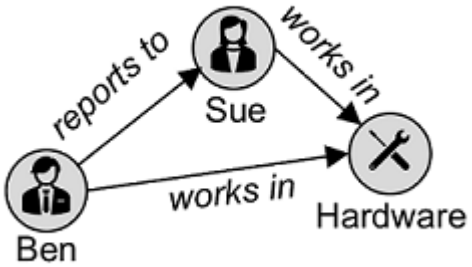
Key	Document
1	{ "name": "Joe Jones", "email": "joe@litware.com" }
2	{ "name": "Samir Nadoy", "email": "Samir@northwind.com" }

Column family databases, which store tabular data comprising rows and columns, but you can divide the columns into groups known as column-families. Each column family holds a set of columns that are logically related together.

Orders

Key	Customer		Product	
	Name	Address	Name	Price
1000	Joe Jones	1 Main St.	Hammer	2.99
1001	Samir Nadoy	123 Elm Pl.	Wrench	4.25

Graph databases, which store entities as nodes with links to define relationships between them.



Describe common data workloads

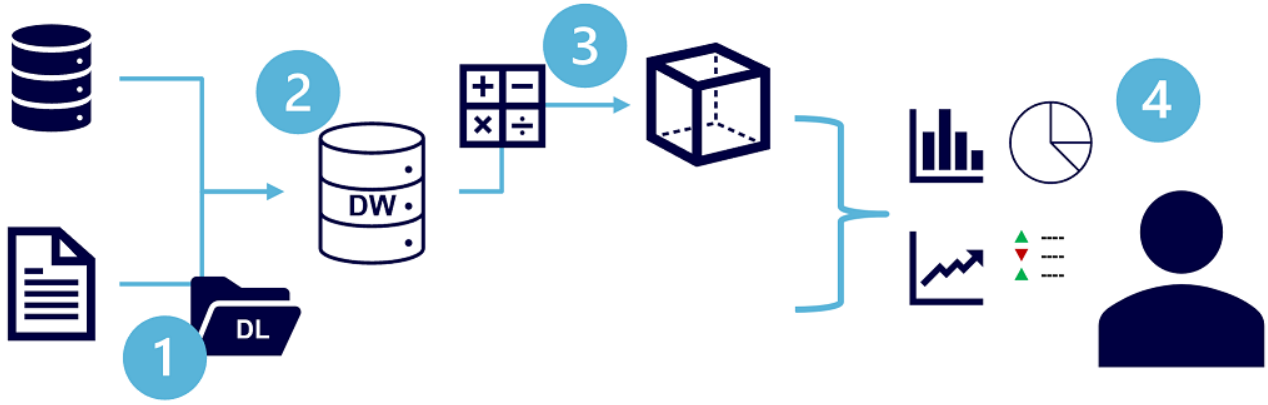
• Describe features of transactional workloads

OLTP solutions rely on a database system in which data storage is optimized for both read and write operations in order to support transactional workloads in which data records are created, retrieved, updated, and deleted (often referred to as *CRUD* operations). These operations are applied transactionally, in a way that ensures the integrity of the data stored in the database. To accomplish this, OLTP systems enforce transactions that support so-called ACID semantics:

Atomicity – each transaction is treated as a single unit, which succeeds completely or fails completely. For example, a transaction that involved debiting funds from one account and crediting

the same amount to another account must complete both actions. If either action can't be completed, then the other action must fail.
Consistency – transactions can only take the data in the database from one valid state to another. To continue the debit and credit example above, the completed state of the transaction must reflect the transfer of funds from one account to the other.
Isolation – concurrent transactions cannot interfere with one another, and must result in a consistent database state. For example, while the transaction to transfer funds from one account to another is in-process, another transaction that checks the balance of these accounts must return consistent results - the balance-checking transaction can't retrieve a value for one account that reflects the balance *before* the transfer, and a value for the other account that reflects the balance *after* the transfer.
Durability – when a transaction has been committed, it will remain committed. After the account transfer transaction has completed, the revised account balances are persisted so that even if the database system were to be switched off, the committed transaction would be reflected when it is switched on again.

• Describe features of analytical workloads
Analytical data processing typically uses read-only (or read-*mostly*) systems that store vast volumes of historical data or business metrics. Analytics can be based on a snapshot of the data at a given point in time, or a series of snapshots.
The specific details for an analytical processing system can vary between solutions, but a common architecture for enterprise-scale analytics looks like this:



Data files may be stored in a central data lake for analysis.
An extract, transform, and load (ETL) process copies data from files and OLTP databases into a data warehouse that is optimized for read activity. Commonly, a data warehouse schema is based on *fact* tables that contain numeric values you want to analyse (for example, sales amounts), with related *dimension* tables that represent the entities by which you want to measure them (for example, customer or product),
Data in the data warehouse may be aggregated and loaded into an online analytical processing (OLAP) model, or *cube*. Aggregated numeric values (*measures*) from fact tables are calculated for intersections of *dimensions* from dimension tables. For example, sales revenue might be totaled by date, customer, and product.
The data in the data lake, data warehouse, and analytical model can be queried to produce reports, visualizations, and dashboards.
Data lakes are common in large-scale data analytical processing scenarios, where a large volume of file-based data must be collected and analysed.
Data warehouses are an established way to store data in a relational schema that is optimized for read operations – primarily queries to support reporting and data visualization. The data warehouse

schema may require some denormalization of data in an OLTP data source (introducing some duplication to make queries perform faster).
An OLAP model is an aggregated type of data storage that is optimized for analytical workloads. Data aggregations are across dimensions at different levels, enabling you to *drill up/down* to view aggregations at multiple hierarchical levels; for example to find total sales by region, by city, or for an individual address. Because OLAP data is pre-aggregated, queries to return the summaries it contains can be run quickly.
Different types of user might perform data analytical work at different stages of the overall architecture. For example:
Data scientists might work directly with data files in a data lake to explore and model data.
Data Analysts might query tables directly in the data warehouse to produce complex reports and visualizations.
Business users might consume pre-aggregated data in an analytical model in the form of reports or dashboards.

Identify roles and responsibilities for data workloads

• Describe responsibilities for database administrators
Design, Implementation, Maintenance, Backups, Availability, Performance and Optimisation.
Policies, Backup and Recovery plans
Granting and Denying access to the database

• Describe responsibilities for data engineers
Design and Implement assets to ingest data via pipelines, cleansing and transformations. Wide range of data platforms for relation and non relational data. Responsible for data and privacy for all of the data.

• Describe responsibilities for data analysts
Maximise value of the data assets and builds models for analytical reports and visualisations for Insights