Fiche de révision – Certification Microsoft DP-900 (Azure Data Fundamentals)

Cette fiche de révision couvre les **concepts essentiels** à maîtriser pour l'examen **DP-900 Azure Data Fundamentals**. La certification DP-900 valide votre connaissance des concepts de données fondamentaux et des services de données Azure, vous aidant à acquérir une solide compréhension des bases des services de données dans le cloud [1]. Vous trouverez ci-dessous un rappel structuré des principaux thèmes de l'examen, avec des explications détaillées, des exemples concrets et des schémas utiles pour faciliter vos révisions.

Concepts de base des données (Core Data Concepts)

Types de données : structurées, semi-structurées et non structurées

Comparaison entre données structurées et non structurées. Les **données structurées** sont organisées selon un schéma prédéfini (par exemple en tableaux de lignes et de colonnes) et souvent stockées dans des bases **relationnelles** [2]. Chaque ligne d'une table structurée suit la même structure de colonnes, ce qui rend les données faciles à interroger avec le langage SQL. Au contraire, les **données semi-structurées** ne s'intègrent pas dans un schéma tabulaire rigide mais possèdent tout de même une structure implicite (par exemple des documents JSON ou XML contenant des paires clé-valeur) [3]. Enfin, les **données non structurées** n'ont pas de modèle prédéfini – il peut s'agir de texte libre, de fichiers audio/vidéo, d'images, etc., dont le contenu n'est pas organisé en champs fixes [4]. Ces données brutes sont plus difficiles à analyser sans traitement préalable.

Les fichiers de données peuvent être stockés dans divers **formats** en fonction de leur structure. Par exemple, un fichier CSV (valeurs séparées par des virgules) convient bien aux données tabulaires structurées, tandis que des formats comme **JSON** ou **XML** sont utilisés pour des données semi-structurées en conservant la hiérarchie des éléments [5]. Pour les besoins de *Big Data*, on recourt également à des formats optimisés comme **Parquet** ou **Avro**, qui stockent les données de façon compressée et colonnaire afin d'en améliorer la compression et la vitesse de lecture.

Options de stockage : bases de données relationnelles vs NoSQL

En termes de **types de bases de données**, on distingue principalement les bases **relationnelles** et les bases **non relationnelles** (NoSQL). Les bases relationnelles (SQL) organisent les données en tables liées par des relations définies (clés primaires/étrangères). Elles imposent un schéma fixe (*schema-on-write*) : la structure des données est définie à l'avance et toute donnée insérée doit respecter ce schéma [6]. Elles garantissent l'intégrité des transactions selon le modèle **ACID** (Atomicité, Cohérence, Isolation, Durabilité) pour assurer des mises à jour fiables [6]. Les bases NoSQL, quant à elles, assouplissent le schéma (**schema-on-read** – la structure est interprétée à la lecture) et se déclinent en plusieurs modèles adaptés à des usages différents : **bases de documents** (JSON, XML), **bases clé-valeur** (stockage par paire clé/attribut, schéma très flexible), **bases en colonnes** (stockage colonne-famille optimisé pour les larges jeux de données distribués), et **bases de graphes** (gestion de relations complexes sous forme de nœuds/arêtes) [7]. Ces technologies NoSQL offrent une **haute scalabilité** et des performances accrues sur de gros volumes en sacrifiant certaines garanties des SGBD relationnels (par exemple, elles n'implémentent pas toujours les transactions ACID globales).

Note : Azure propose des services managés pour chaque type de base de données. Par exemple, *Azure SQL Database* est un SGBD relationnel managé, tandis qu'*Azure Cosmos DB* (voir plus loin) offre un service NoSQL multi-modèles, et *Azure Table Storage* fournit un stockage clé-valeur très simple pour des données semi-structurées.

Charges de travail : transactionnelles vs analytiques (OLTP vs OLAP)

En **workloads** (charges de travail) de données, on distingue les systèmes **transactionnels** et les systèmes **analytiques**. Les systèmes *transactionnels* correspondent au traitement de transaction en temps réel (**OLTP**, *Online Transaction Processing*) – ils gèrent de nombreuses petites opérations courantes de mise à jour ou consultation (par ex. enregistrement d'un paiement, d'une commande) avec un fort souci de fiabilité et de rapidité. Un système OLTP doit supporter un très grand nombre de transactions individuellement petites, souvent simultanées, tout en garantissant que chaque transaction aboutit correctement sans corruption de données [8]. À l'inverse, les systèmes *analytiques* (**OLAP**, *Online Analytical Processing*) sont conçus pour l'analyse de gros volumes de données, souvent historiques ou agrégées, afin d'en extraire des indicateurs et tendances. Un système OLAP réalise des requêtes complexes impliquant des agrégations sur des millions de lignes, ce qui prend plus de temps qu'une transaction OLTP typique, mais permet de **générer des rapports, tableaux de bord et analyses** pour la décision d'entreprise. En résumé, l'OLAP est optimisé pour le **reporting analytique sur des données agrégées**, tandis que l'OLTP est optimisé pour le **traitement transactionnel courant avec mises à jour en temps réel** [9]. La plupart des organisations utilisent une combinaison des deux : les données opérationnelles sont stockées et gérées par des systèmes OLTP (par ex. une base de données de vente en ligne), puis ces données sont périodiquement extraites vers un entrepôt de données ou un lac de données où des systèmes OLAP les analysent pour produire des insights (par ex. analyser les ventes mensuelles, le comportement client, etc.).

Rôles et responsabilités autour des données

L'examen couvre également les principaux **rôles métiers** impliqués dans la gestion et l'exploitation des données :

- **Administrateur de base de données (DBA)** : il est responsable de la gestion des bases de données d'une organisation. Ses tâches incluent l'installation et la configuration des SGBD, la sécurité des accès, la surveillance des performances, l'optimisation des requêtes, la mise en place de sauvegardes et de plans de reprise, ainsi que le contrôle global de l'intégrité des données. En somme, le DBA assure que les bases de données sont **fiables, sécurisées et performantes** au quotidien.

- **Ingénieur de données (Data Engineer)** : il conçoit et met en place les **pipelines de données** et les infrastructures de traitement des données. Son rôle est d'orchestrer la collecte des données depuis diverses sources, leur transformation (nettoyage, agrégation…) et leur stockage dans des systèmes adaptés (entrepôts de données, data lakes, etc.) pour qu'elles puissent être exploitées. Le data engineer travaille donc sur l'aspect *architecture et intégration* des données, en veillant à la scalabilité des solutions. En d'autres termes, c'est lui qui « fabrique » les flux de données brutes en produits de données prêts à l'emploi [10].

- **Analyste de données (Data Analyst)** : il exploite les données pour en tirer des **indicateurs métiers et des insights** facilitant la prise de décision. En utilisant des outils d'analyse et de visualisation (tels que Power BI, voir plus loin), il explore les jeux de données afin d'identifier des tendances, produire des rapports, tableaux de bord et répondre aux questions des décideurs. Son travail consiste à extraire de la valeur des données existantes (*BI – Business Intelligence*),

souvent en collaboration avec les équipes métier. L'analyste doit bien comprendre les objectifs de l'entreprise pour formuler des analyses pertinentes, et il est un **intermédiaire entre la technique et le métier**, transformant des données en information compréhensible [11].

Ces trois rôles collaborent souvent étroitement (et avec d'autres comme les architectes de données ou data scientists) pour assurer une **gestion complète du cycle de vie des données** : depuis la création des infrastructures et flux (ingénieur), l'administration des systèmes de stockage (DBA), jusqu'à l'analyse et la valorisation des données (analyste).

Données relationnelles sur Azure (Relational data on Azure)

Concepts des bases de données relationnelles

Une **base de données relationnelle** stocke les informations dans des tables structurées reliées entre elles par des relations définies. Les données y sont hautement **normalisées** (réparties dans plusieurs tables de manière à éviter les duplications), et la cohérence est garantie par des contraintes (clés primaires, clés étrangères, contraintes d'unicité, etc.). Les SGBD relationnels adoptent généralement un modèle de schéma strict, et utilisent le **langage SQL** (*Structured Query Language*) pour interroger et manipuler les données [6].

- **Caractéristiques du modèle relationnel :** il impose un schéma fixe (toutes les lignes d'une table partagent les mêmes colonnes définies à l'avance) et supporte les transactions ACID pour assurer la fiabilité des mises à jour [6]. Les données sont organisées de façon à minimiser la redondance (**normalisation**). La **normalisation** est le processus consistant à structurer les données en plusieurs tables liées plutôt qu'en une seule table redondante, afin d'éliminer les duplicatas et anomalies de mise à jour. Par exemple, stocker séparément une table des Clients et une table des Commandes évite de répéter les informations client sur chaque commande. La normalisation améliore l'intégrité (chaque facteure est à un seul endroit) au prix de requêtes parfois plus complexes (nécessité de faire des jointures entre tables) [12].

- **Principales opérations SQL :** on distingue les instructions **DDL** (*Data Definition Language*) pour définir le schéma (ex : `CREATE TABLE` pour créer une table, `ALTER` pour modifier la structure, etc.), et les instructions **DML** (*Data Manipulation Language*) pour manipuler le contenu. Parmi les plus courantes, on trouve `SELECT` (lecture de données), `INSERT` (insertion de nouvelles données), `UPDATE` (mise à jour de données existantes) et `DELETE` (suppression de données). Ces commandes permettent de réaliser la majorité des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) au sein d'une base. D'autres éléments incluent les **jointures** (combiner des tables dans une requête), les **filtres** (`WHERE`), les agrégations (`GROUP BY`, fonctions SUM, COUNT, etc.) et les sous-requêtes, qui sont au cœur de l'interrogation relationnelle.

- **Objets de base de données courants :** en environnement SQL, les données sont stockées dans des **tables**. Un ensemble de tables reliées constitue un schéma de base de données. On utilise des **index** pour accélérer les recherches sur de larges tables (similaires à l'index d'un livre), au prix d'un surcoût à la mise à jour. Des **vues** peuvent être créées pour représenter des requêtes enregistrées comme des tables virtuelles, facilitant l'accès aux données sans dupliquer celles-ci. Les **procédures stockées** et fonctions sont des blocs de code SQL enregistrés dans la base, utilisés pour encapsuler la logique métier ou les manipulations fréquentes côté serveur. Enfin, on trouve des éléments tels que les **séquences** (générateurs d'identifiants auto-incrémentés), les **déclencheurs** (*triggers* qui exécutent du code en réaction à un événement sur la base), etc. Ces objets forment l'écosystème complet d'une base relationnelle.

Services Azure de bases relationnelles

Microsoft Azure propose plusieurs services managés pour héberger des bases de données relationnelles sans avoir à gérer l'infrastructure sous-jacente :

- **Azure SQL Database** : base de données relationnelle *as-a-service* entièrement managée, basée sur le moteur Microsoft SQL Server. Ce service offre l'**évolutivité** (ajout/suppression de ressources à la demande), la haute disponibilité intégrée, et dispense l'utilisateur des tâches d'administration lourdes (installation de serveur, mises à jour, sauvegardes – celles-ci sont automatisées). Azure SQL Database est idéal pour les nouvelles applications cloud nécessitant une base SQL Server sans l'overhead de gestion. Il peut être déployé en mode base de données unique, pool élastique (plusieurs bases partageant des ressources) ou instance managée limitée (voir Azure SQL MI ci-dessous) [13].

- **Azure SQL Managed Instance (MI)** : instance managée de SQL Server dans Azure, offrant quasiment 100% de compatibilité avec un SQL Server sur site. Azure SQL MI est conçue pour faciliter la migration des applications existantes : elle supporte les fonctionnalités SQL Server (Agent SQL, requêtes cross-database, etc.) non disponibles dans Azure SQL Database mono-base. C'est un **PaaS** qui combine les avantages du managé (patches, backups gérés) et la compatibilité d'une instance locale, simplifiant le *lift-and-shift* vers Azure.

- **SQL Server sur Machine Virtuelle Azure** : Azure permet également d'héberger SQL Server sur une VM Windows ou Linux dans le cloud (mode IaaS). Cette option équivaut à installer SQL Server sur son propre serveur, avec la liberté de configurer le système à sa guise, mais implique de gérer soi-même les mises à jour, backups, etc. On l'utilise surtout pour des besoins spécifiques non couverts par les offres PaaS (versions particulières, besoins d'extensions tierces, ou compatibilité à 100% requise).

- **Bases de données open-source managées** : Azure propose des services managés pour les principaux SGBD open-source : **Azure Database for MySQL**, **Azure Database for PostgreSQL** et **Azure Database for MariaDB**. Ces services offrent l'expérience familière de ces SGBD tout en déléguant à Azure la gestion de l'infrastructure (haute dispo, sauvegardes, patches). Par exemple, *Azure Database for MySQL* fournit un serveur MySQL dans le cloud sans que l'utilisateur ait à administrer le système sous-jacent [14]. *Azure Database for PostgreSQL* existe en mode **Serveur unique** (instance classique) ou **Hyperscale (Citus)** permettant de distribuer une base PostgreSQL sur plusieurs nœuds pour supporter des charges très lourdes [15]. Ces offres facilitent la migration d'applications utilisant MySQL/PostgreSQL/MariaDB vers Azure, ou le développement de nouvelles applications en bénéficiant des atouts du cloud (mise à l'échelle flexible, service managé).

En résumé, Azure couvre tous les besoins relationnels : que ce soit via son **Azure SQL** (famille SQL Server) ou via ses services pour SGBD **open-source**, il est possible d'héberger dans le cloud tout type de base relationnelle avec le niveau de contrôle ou de délégation désiré.

Données non relationnelles sur Azure (Non-relational data on Azure)

Stockage Azure (Blob, File, Table)

Azure Storage fournit plusieurs services de stockage non relationnel pour des données semi ou non structurées, chacun adapté à un cas d'usage :

- **Azure Blob Storage** : le service de stockage d'objets (*blobs*) dans Azure, conçu pour stocker d'énormes volumes de données non structurées (fichiers, données binaires, etc.) de manière durable et économique [16]. Un *blob* peut être n'importe quel fichier – document, image, vidéo, sauvegarde, logs – et Blob Storage permet d'y accéder via des protocoles HTTP/HTTPS mondialement. On peut s'en servir pour diffuser du contenu (par exemple servir des images ou vidéos à une application web), stocker des sauvegardes et archives, ou alimenter des traitements big data [17]. Blob Storage est **hautement scalable** : il peut gérer des pétaoctets de données et des milliers de requêtes par seconde. Azure Blob propose en outre différents *tiers* de stockage (Hot, Cool, Archive) pour optimiser le coût en fonction de la fréquence d'accès aux données. À noter qu'Azure Data Lake Storage Gen2 est une extension d'Azure Blob qui fournit une couche de système de fichiers hiérarchique pour les besoins analytiques big data [18] (il s'agit du même service sous le capot, avec des fonctionnalités additionnelles pour les lacs de données).

- **Azure Files** : ce service offre des partages de fichiers entièrement managés dans le cloud, accessibles via les protocoles standards SMB (Windows) ou NFS (Linux). En clair, Azure Files permet de créer un **partage réseau dans Azure** que l'on peut monter depuis des machines virtuelles ou des postes clients, comme on le ferait d'un NAS classique. L'intérêt est de pouvoir migrer facilement des applications existantes qui s'appuient sur des lecteurs réseaux ou stockages de fichiers, vers Azure, sans changement applicatif. Les partages Azure Files sont accessibles en mode cloud (via internet) ou en montage local sur un réseau d'entreprise (via Azure File Sync). *Azure Files* est un service **100% managé** : Microsoft gère l'infrastructure sous-jacente et assure la redondance des données. Les utilisateurs bénéficient d'un stockage de fichiers simple d'utilisation (« lift & shift » d'un serveur de fichiers vers Azure) avec la possibilité d'intégrer aux annuaires AD pour la gestion des droits. En résumé, Azure Files fournit des **partages de fichiers cloud via SMB/NFS**, évitant d'avoir à maintenir un serveur de fichiers sur VM [19].

- **Azure Table Storage** : un service de stockage NoSQL de type **clé/attribut** (Key-Value) offrant un schéma très flexible pour les données semi-structurées. Azure Table permet de stocker des entités (similaires à des lignes) identifiées par une clé partition et une clé de ligne, chaque entité pouvant avoir des propriétés arbitraires sans schéma strict. C'est un stockage « NoSQL » simple, idéal pour des applications qui ont besoin d'une base scalable, à faible coût, pour stocker des enregistrements simples sans relations complexes. **Chaque table Azure** peut contenir des milliards d'entités, et le service répartit automatiquement les données pour en assurer la montée en charge (*scale-out* horizontal). Les accès se font principalement par clé (PartitionKey + RowKey) et sont très rapides. En somme, Azure Table Storage fournit un **store NoSQL schéma-libre hautement disponible**, souvent utilisé pour des scénarios comme stocker des profils utilisateurs, des configurations, des logs structurés, ou tout dataset où la simplicité et l'échelle priment sur les fonctionnalités avancées. Microsoft le décrit comme un stockage NoSQL clé-valeur pour un développement rapide sur des jeux massifs de données semi-structurées [20]. (Notons qu'Azure Table, en tant que composant « historique » d'Azure Storage, a désormais un

successeur plus puissant via *Cosmos DB Table API* ; cependant Azure Table Storage reste utilisé pour sa simplicité et son coût faible dans certains cas.)

Azure Cosmos DB

Azure Cosmos DB est le service de base de données NoSQL phare d'Azure, offrant une distribution globale et une performance à grande échelle. Cosmos DB se distingue par son caractère *multi-modèle* : il prend en charge différents types de données et API selon les besoins. Concrètement, Cosmos DB stocke les données sous forme de documents JSON atomiques, mais expose plusieurs interfaces : l'**API SQL (Core)** propre à Cosmos pour des requêtes type SQL sur JSON, l'**API MongoDB** (compatibilité avec les pilotes et requêtes Mongo), l'**API Cassandra** (pour utiliser Cosmos comme un magasin de colonnes style Cassandra), l'**API Gremlin** (base de **graphes** pour données fortement liées) et l'**API Table** (mode clé-valeur compatible avec Azure Table Storage). Toutes ces API interagissent avec le même moteur sous-jacent, qui offre des garanties de performance et de disponibilité élevées. Cosmos DB est conçu pour des applications **mondiales** nécessitant une latence de l'ordre de la milliseconde et un débit de requêtes très élevé, avec la possibilité de répartir les données sur plusieurs régions Azure dans le monde entier (*multi-master replication*). Il s'accompagne de *SLA* stricts garantissant une disponibilité 99.999% en lecture dans les configurations multi-régions, ainsi que des performances (débit, latence) configurables par l'allocation d'unités de requête (RU/s) [21] [22] .

Cas d'usage : Cosmos DB convient particulièrement aux applications **web, mobiles, IoT ou gaming** qui ont besoin de distribuer les données au plus près des utilisateurs pour minimiser la latence, et d'absorber des pics de charge massifs. Par exemple, un réseau social mondial, un service e-commerce multi-continents, ou un système de capteurs IoT enregistrant des télémétries en temps réel bénéficieront de Cosmos DB. Il est également utilisé pour la gestion de catalogues produits, les systèmes de personnalisation en ligne, les données de profils utilisateurs, etc., bref tout scénario nécessitant un stockage *NoSQL* **évolutif globalement**. Azure Cosmos DB fournit cinq niveaux de **consistance** au choix (éventuelle, préfixe cohérent, session, obsolescence bornée, stricte) afin de permettre de régler le compromis cohérence/latence selon les besoins de l'application.

En résumé, Cosmos DB est une **base NoSQL distribuée globalement, multi-modèle**, offrant des temps de réponse très faibles et une élasticité quasi-infinie. Il est **multi-API** : un même service gère des documents, des graphes, des tables clé-valeur, etc., via les API familières correspondantes [23] . C'est un service de choix pour toutes les applications nécessitant **faible latence, haute disponibilité et passage à l'échelle mondiale** des données [24] .

(À noter : la mise en œuvre de Cosmos DB, bien que transparente pour l'utilisateur, repose sur des partitions horizontales automatiques des données et un indexage systématique des documents JSON, permettant des requêtes riches. Le modèle de facturation est basé sur le débit réservé en RU/s et la taille stockée.)

Charges de travail analytiques sur Azure (Analytics workloads on Azure)

Cette section recouvre les solutions Azure destinées à **ingérer, stocker et analyser** de larges volumes de données, ainsi qu'à produire des visualisations destinées aux utilisateurs finaux (décideurs, analystes).

Architecture analytique à grande échelle : ingestion, stockage et traitement des données

Dans une solution analytique typique (par exemple, implémentation d'un *data warehouse* ou d'une *data lakehouse*), on distingue plusieurs étapes : **l'ingestion des données**, **le stockage analytique** et **le traitement/analyse** des données.

- **Ingestion de données** : c'est le processus de collecte des données depuis leurs sources vers une plateforme centralisée. Les données peuvent provenir de bases transactionnelles (on extrait des copies, souvent via des mécanismes d'ETL/ELT), de fichiers, de flux IoT ou logs, etc. Sur Azure, un service clé pour l'ingestion par lots est **Azure Data Factory** (ou les pipelines d'Azure Synapse), qui orchestre des copies de données à intervalle régulier depuis diverses sources vers des cibles (par exemple, copier des fichiers CSV quotidiens vers un Data Lake, ou extraire des tables d'une base on-premise vers Azure Blob). Pour des **flux en temps réel**, Azure propose des services d'ingestion d'événements comme **Azure Event Hubs** ou **Azure IoT Hub**, capables d'absorber des millions d'événements par seconde en entrée et de les remettre aux moteurs de stream processing. L'objectif de la couche d'ingestion est d'assurer un **débit élevé** et une **scalabilité** pour acheminer la donnée brute vers les zones de traitement, tout en gérant la fiabilité (tolérance aux pannes, files d'attente tampon, etc.) et la diversité des formats.

- **Stockage analytique** : une fois les données collectées, il faut les stocker de manière à pouvoir les analyser efficacement. Deux approches principales sont le **Data Warehouse** et le **Data Lake** (et de plus en plus des architectures hybrides *lakehouse*).

- Un **entrepôt de données (data warehouse)** stocke les données de façon relationnelle mais dénormalisée (schéma en étoile ou en flocon, optimisé pour les requêtes d'analyse). Azure propose **Azure Synapse Analytics (anciennement Azure SQL Data Warehouse)** comme entrepôt de données *cloud* massivement parallèle. Synapse permet de stocker des téraoctets de données relationnelles et d'exécuter des requêtes SQL analytiques distribuées sur de multiples nœuds pour obtenir des résultats en quelques secondes ou minutes.

- Un **lac de données (data lake)** stocke, lui, les données dans leur format brut (par ex fichiers JSON, CSV, Parquet) sur un système distribué (typiquement Azure Data Lake Storage sur Blob) et on applique un schéma à la lecture (*schema-on-read*). Le data lake est idéal pour conserver *toutes* les données (structurées ou non) à coût modique, et les exploiter via des moteurs big data (Spark, Hive…) sans avoir à les importer dans un SGBD. Dans la pratique, on voit des architectures combinant les deux : données brutes historisées dans le *lake*, données raffinées et agrégées dans l'entrepôt. Azure encourage désormais les solutions **Synapse Lakehouse** (notamment via Microsoft Fabric, voir plus loin) qui unissent ces mondes.

- **Traitement et analyse des données** : c'est l'étape où l'on transforme les données et où l'on en extrait de la valeur. Azure propose plusieurs moteurs pour traiter les données à grande échelle :

- **Azure Databricks** : plateforme d'analyse big data basée sur **Apache Spark**, entièrement managée sur Azure. Databricks fournit un environnement de *notebooks* collaboratifs (en Python, SQL, Scala, R, etc.) où data engineers et data scientists peuvent développer des pipelines de données, des entraînements de modèles ML, etc. Spark permet de traiter des volumes massifs en mémoire ou sur disque en distribuant le calcul sur des clusters. Databricks s'intègre nativement avec Azure Storage, Data Lake et Synapse. C'est idéal pour du *batch processing* avancé, du machine learning à l'échelle ou des transformations complexes sur des données volumineuses.

- **Azure Synapse Analytics** : c'est une plateforme unifiée d'analytique. Non seulement Synapse inclut le moteur d'entrepôt de données (SQL pools), mais il propose aussi des **pools Spark** intégrés pour du traitement big data, des **pipelines d'orchestration** (similaires à Data Factory) et un **studio** unifié pour développer tout cela. Synapse vise à fournir un environnement tout-en-un pour construire des solutions de *Business Intelligence* modernes, allant du raw data jusqu'aux tableaux de bord. En 2023, Microsoft a introduit **Microsoft Fabric**, qui pousse encore plus loin cette unification (intégrant Synapse, Power BI et d'autres services dans une expérience SaaS unique).
- **Azure HDInsight** : service plus ancien d'Azure permettant de déployer des clusters Hadoop/Spark/Kafka gérés. HDInsight offre des clusters préconfigurés pour différents composants de l'écosystème Hadoop (MapReduce, Spark, Hive, HBase, Storm, Kafka…). C'est une option flexible pour ceux qui ont besoin d'une solution open-source un peu plus *DIY*, mais elle est moins utilisée aujourd'hui au profit de Databricks ou Synapse qui apportent plus de facilité d'utilisation.
- **Azure Data Factory** : mentionné pour l'ingestion, il propose aussi les *Data Flow* pour faire des transformations de données à l'échelle de façon visuelle (il génère du code Spark sous le capot). Toutefois, Data Factory est surtout un orchestrateur/ETL, pas un moteur d'analyse en soi. Souvent, il déclenche des activités sur Databricks, Synapse ou d'autres services pour effectuer les traitements.

En résumé, Azure dispose d'un **écosystème complet de services analytiques**. Pour les besoins d'**entreposage de données**, on utilise Synapse (SQL pools) ou d'autres bases analytiques. Pour le **traitement big data**, on a Databricks, Synapse Spark, HDInsight. Pour l'**ingestion et l'orchestration**, Data Factory (ou Synapse pipelines). Et pour l'**intégration totale** de ces composants, Microsoft propose depuis peu **Microsoft Fabric**, qui intègre data factory, un lakehouse, un entrepôt, des moteurs Spark et SQL, et Power BI dans une expérience unifiée.

> **Services analytiques Azure** – *Récapitulatif* : **Azure Synapse Analytics**, **Azure Databricks**, **Azure HDInsight** et **Azure Data Factory** sont quatre services clés souvent utilisés conjointement pour construire des solutions analytiques dans Azure [25]. Synapse fournit un moteur d'entrepôt de données et du big data Spark intégré, Databricks excelle pour le big data collaboratif, HDInsight permet d'exploiter l'écosystème Hadoop open source, et Data Factory assure l'orchestration des flux de données entre ces systèmes et les sources/cibles externes.

Analytique en temps réel (réaltime analytics)

L'analytique en temps réel consiste à traiter et analyser les données **en continu**, à mesure qu'elles sont produites, afin de générer instantanément des insights ou actions. Cela se distingue du traitement par lots (batch) où les données sont accumulées puis traitées de façon différée. La principale différence est la **latence** : le *batch processing* traite de gros volumes en une fois, avec une latence élevée (ex : rapports quotidiens ou mensuels), alors que le *stream processing* traite événement par événement (ou micro-lots fréquents) avec une latence de quelques secondes tout au plus [26] [27]. En somme, le streaming sacrifie l'efficacité globale sur volume pour obtenir une réponse quasi-immédiate sur chaque donnée entrante.

Batch vs Streaming – Un procédé *batch* classique pourrait consister à agriger toutes les transactions d'une journée et calculer des statistiques la nuit pour un rapport le lendemain. Au contraire, en *stream*, on analyserait chaque transaction ou petite fenêtre de transactions à la volée pour mettre à jour en temps réel les indicateurs (par exemple afficher en direct le chiffre d'affaires du jour). Le streaming est crucial pour les applications où le temps réel apporte de la valeur : détection de fraudes instantanée, monitoring de machines industrielles pour alerter en cas d'anomalie, fil d'actualités en live, etc. Le batch

reste pertinent pour les traitements lourds non temps-réel (agrégations historiques, calculs complexes non urgents).

Services Azure pour l'analytique en temps réel : Azure propose plusieurs outils pour implémenter des pipelines de données streaming :

- **Azure Stream Analytics** : un moteur de traitement d'événements entièrement managé, qui permet d'écrire des requêtes SQL pour analyser des flux de données en entrée (provenant par exemple d'Event Hubs, IoT Hub ou du flux de mises à jour de Cosmos DB). Stream Analytics applique en continu les requêtes aux événements reçus, avec des fenêtres temporelles, des agrégations et des détections de motifs, puis peut écrire les résultats vers diverses sorties (bases de données, tableaux de bord Power BI en temps réel, stockage, etc.). L'intérêt majeur est la simplicité : il s'agit d'une solution *serverless* où l'on n'écrit quasi que du SQL, le service se charge de l'infrastructure. **Azure Stream Analytics** est conçu pour le *mission-critical* en streaming, permettant de bâtir un pipeline de streaming bout-en-bout en quelques clics ou quelques lignes de SQL [28] . C'est idéal pour traiter des données de capteurs IoT, des logs d'activité, des clics web en temps réel, et générer des alertes ou dashboards en direct.

- **Azure Data Explorer / Synapse Data Explorer** : Azure Data Explorer (ADX) est un service spécialement conçu pour les analyses exploratoires de **données chronologiques et logs** à grande échelle, quasiment en temps réel. Intégré maintenant dans Synapse (Synapse Data Explorer), il permet d'ingérer des flux de données (par ex des télémétries IoT, logs applicatifs, métriques temps réel) et de les rendre requêtables quasi instantanément via un langage KQL (Kusto Query Language) très puissant. ADX excelle dans les requêtes agrégatives complexes sur des fenêtres de temps courtes, le tout sur des volumétries gigantesques, avec une optimisation pour les séries temporelles. Microsoft le recommande pour construire des solutions de supervision, d'analytique IoT ou d'analyse de logs qui nécessitent d'interroger des données fraîchement arrivées en quelques secondes [29] . Par exemple, couplé à Stream Analytics ou Event Hubs pour l'ingestion, Data Explorer peut permettre de détecter en quasi-direct des anomalies dans des données de capteurs, ou explorer des journaux d'événements juste après leur émission.

- **Spark Structured Streaming (Databricks/Synapse)** : pour des besoins plus sur-mesure, les moteurs Apache Spark supportent aussi le traitement de flux via *Structured Streaming*. Sur Azure Databricks ou Synapse Spark, on peut écrire des programmes en Python/Scala utilisant Spark pour lire en continu depuis une source (comme Event Hub ou Kafka) et appliquer des transformations arbitraires, puis écrire les résultats vers des sinks. Cette approche donne plus de flexibilité (logique personnalisée, usage de bibliothèques ML en temps réel, etc.) au prix d'une complexité accrue par rapport à Stream Analytics. C'est approprié lorsque l'on veut intégrer du *machine learning temps réel* ou des étapes de calcul complexes dans le pipeline de streaming.

En résumé, pour de l'**analytics temps réel**, Azure fournit à la fois des services gérés clé-en-main (Stream Analytics) et la possibilité d'utiliser des frameworks custom (Spark streaming, ou même Storm via HDInsight). On utilisera aussi souvent **Event Hubs** comme **ingestion** pour ces scénarios : c'est un bus d'événements haute performance qui alimente les moteurs de streaming en données, assurant le **découplage** entre la production des événements et leur traitement en temps réel.

Visualisation des données avec Power BI

La dernière étape d'une solution analytique consiste souvent à **visualiser** les données de façon intelligible pour les utilisateurs finaux. Microsoft **Power BI** est l'outil principal dans l'écosystème Azure/ Microsoft pour la *Business Intelligence* et la data visualization.

Présentation de Power BI : Power BI est un service d'analyse et de visualisation de données qui permet de créer des rapports interactifs et des tableaux de bord dynamiques à partir de diverses sources de données. Il s'agit d'un outil de *self-service BI*, c'est-à-dire que des utilisateurs métiers (non développeurs) peuvent, via une interface conviviale, concevoir leurs propres analyses. Power BI se compose d'une application de bureau (*Power BI Desktop*) pour la création de rapports, d'un service cloud (*Power BI Service*) pour le partage et la consultation en ligne, et de applications mobiles pour consulter les tableaux de bord. Microsoft le décrit comme *« un service d'analyse métier qui délivre des insights permettant des décisions rapides et éclairées »*, offrant des visualisations interactives et des capacités analytiques via une interface simple pour que les utilisateurs puissent créer eux-mêmes rapports et dashboards [30].

Capacités et modèle de données : Power BI se connecte à un très large éventail de sources de données (fichiers Excel/CSV, bases SQL, données Azure, services SaaS type Salesforce, etc.). Il intègre un moteur de transformation de données appelé **Power Query** qui permet de nettoyer et façonner les données avant analyse (par exemple fusionner deux tables, calculer de nouvelles colonnes, pivoter des données – le tout de manière graphique, les étapes étant enregistrées et rejouées à chaque actualisation). Une fois les données importées et transformées, Power BI les charge dans un **modèle analytique en mémoire** (basé sur la technologie Microsoft Analysis Services). Ce modèle prend la forme de tables reliées entre elles – l'analyste peut définir des **relations** (par ex. une table Ventes liée à une table Calendrier et une table Produits) pour créer un schéma en étoile. Le modèle permet aussi de créer des **mesures** calculées (metrics) via le langage DAX, pour les indicateurs plus complexes (ex : somme des ventes, moyenne mobile sur 30 jours, % du total, etc.). Le moteur en mémoire de Power BI est très optimisé et permet d'interroger des millions de lignes en quelques fractions de seconde une fois les données chargées.

En pratique, l'utilisateur construit dans Power BI Desktop des pages de rapport en faisant glisser des champs (dimensions, mesures) sur des **visualisations** (graphiques, tableaux, cartes géographiques, etc.). **Power BI offre une large bibliothèque de visualisations** standard : histogrammes, graphes en courbes, secteurs (camemberts), cartes, jauges, nuages de points, matrices, arbres de décomposition, etc. Chaque type de visuel a ses usages recommandés. Par exemple, un **graphique à barres** est adapté pour comparer des valeurs entre différentes catégories (par ex. ventes par région) [31], un **graphique linéaire ou en aire** met en avant l'évolution d'une valeur dans le temps (tendance chronologique) [32], tandis qu'un **diagramme circulaire (camembert ou son équivalent anneau)** illustre la part de chaque composant dans un total (répartition en pourcentage) [33]. L'outil propose également des éléments comme les **cartes géographiques** (pour visualiser des données spatiales), les **indicateurs KPI/cartes** (pour afficher un seul chiffre clé avec son écart à un objectif) ou des **graphs croisés dynamiques** (tables/matrices permettant de détailler des chiffres par différentes dimensions).

Une fois le rapport conçu et publié sur le **Power BI Service**, il peut être partagé avec d'autres utilisateurs de l'organisation. Ceux-ci pourront interagir avec : filtrer les données, cliquer sur un élément d'un graphique pour voir les détails associés (grâce au *croisement interactif* des visuels), exporter des données sous-jacentes si autorisé, etc. Power BI permet aussi de **programmer l'actualisation automatique** des tableaux de bord (par ex. tous les jours ou toutes les heures, selon la connectivité des sources) pour que les chiffres soient à jour. On peut également créer des **tableaux de bord** en épinglant des visuels clés de différents rapports sur une page synthétique.

En résumé, **Power BI** fournit les **fonctionnalités de visualisation et d'analyse interactives** nécessaires pour exploiter les données une fois qu'elles ont été préparées. Ses **points forts** incluent : - la possibilité de se connecter à un grand nombre de sources hétérogènes et de préparer les données sans code, - un **moteur de modélisation** riche permettant de construire un modèle analytique cohérent (similaire à un petit entrepôt de données tabulaire en mémoire), - une panoplie de **visuels interactifs** pour représenter les données de manière compréhensible et percutante, - des fonctions de **partage et collaboration** via le service en ligne (avec sécurité gérée via Azure AD, etc.).

Pour l'examen DP-900, il faut connaître à un niveau fondamental ces capacités : savoir qu'on peut importer et transformer des données, que Power BI permet de créer des modèles de données avec des relations, et choisir le bon type de visuel en fonction de la nature de l'information à communiquer (comparaison, tendance, proportion, localisation géographique, etc.) [34] .

Références : *Microsoft Learn et documentation officielle Azure, Blog TestPrepTraining – DP-900 Cheat Sheet, K21 Academy – Articles DP-900, Amazon Tech Blog sur batch vs stream, Documentation AWS/Azure sur OLTP vs OLAP, Documentation produit Azure (Azure Storage, Cosmos DB, Stream Analytics, Power BI), etc.*

---

[1] [24] [25] [34] Microsoft Azure Data Fundamentals: DP-900 Cheat Sheet - Blog
https://www.testpreptraining.com/blog/dp-900-microsoft-azure-data-fundamentals-cheat-sheet/

[2] [3] [4] Structured, Unstructured & Semi-Structured Data: Key Differences 2025
https://k21academy.com/microsoft-azure/dp-900/structured-data-vs-unstructured-data-vs-semi-structured-data/

[5] [8] [12] SKILLCERTPRO
https://skillcertpro.com/wp-content/uploads/2020/09/DP-900-Master-Cheat-Sheet.pdf

[6] Understand data store models - Azure Architecture Center | Microsoft Learn
https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-overview

[7] [13] [14] [15] Relational And Non-Relational Datastores In Azure
https://k21academy.com/microsoft-azure/dp-900/relational-and-non-relational-datastores/

[9] OLTP vs OLAP - Difference Between Data Processing Systems - AWS
https://aws.amazon.com/compare/the-difference-between-olap-and-oltp/

[10] [11] Data Engineer vs Data Analyst vs Data Scientist vs DBA
https://k21academy.com/microsoft-azure/data-engineer/azure-data-science-and-data-engineering-certifications-dp-900-vs-dp-100-vs-dp-200-dp-201/

[16] [17] [18] Introduction to Blob (object) Storage - Azure Storage | Microsoft Learn
https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

[19] Introduction to Azure Files | Microsoft Learn
https://learn.microsoft.com/en-us/azure/storage/files/storage-files-introduction

[20] Table storage | Microsoft Azure
https://azure.microsoft.com/en-us/products/storage/tables

[21] [22] [23] Common Use Cases and Scenarios for Azure Cosmos DB | Microsoft Learn
https://learn.microsoft.com/en-us/azure/cosmos-db/use-cases

[26] [27] Batch vs stream data processing: a comprehensive guide for developers - About Amazon India
https://www.aboutamazon.in/news/tech-blog/a-guide-to-batch-vs-stream-data-processing-for-developers

28 Azure Stream Analytics | Microsoft Azure
https://azure.microsoft.com/en-us/products/stream-analytics

29 Data Explorer – Exploration and Analytics Service - Microsoft Azure
https://azure.microsoft.com/en-us/products/data-explorer

30 MotherDuck + Microsoft Power BI
https://motherduck.com/ecosystem/power-bi/

31 32 33 Visualization types in Power BI - Power BI | Microsoft Learn
https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a