Projet Entrepôt de données et big data | HAI708I
Amazon Web Services (AWS)

Francesco-Pio BASILE Dorian REY Elliot DURAND

Faculté des Sciences
Université de Montpellier

2025

Table des matières

1 Analyses des besoins métiers

1.1 Étude de cas

Amazon Web Service (AWS) est une des filiales d'Amazon, lancée officiellement en 2006. Elle est née suite aux besoins croissant de serveurs et d'infrastructure pour le cloud. C'est aujourd'hui le leader mondial dans le domaine du cloud computing, aux côtés de Microsoft Azure et Google Cloud Platform. En 2022, 1.45 millions d'entreprises sont clients chez AWS. Ce dernier joue un rôle tellement majeur dans l'industrie que, lors de la panne du 20 octobre 2025, elle aurait causé des pertes estimées à plus d'un million de dollars par heure pour les entreprises clientes, avec un impact économique total estimé à 1,4 milliard de dollars. L'entreprise possède en conséquence une infrastructure gigantesque, avec plus de 120 zones de disponibilités, une zone de disponibilité est en endroit dans lequel il y a un ou plusieurs centres de données (l'entreprise a donc au moins 120 centres de données), nous l'appellerons par la suite AZ.

Le modèle économique d'AWS repose sur la vente de services, tel que la location de serveurs dédiés, l'hébergement de bases de données et la fourniture de serveurs spécialisés pour l'IA. La facturation est différente d'un abonnement mensuel classique, puisque les utilisateurs sont prélevés en fonction du taux d'utilisation des services (modèle "Pay as you go"). Par exemple, pour une base de données PostgreSQL, le tarif est actuellement de 0,026$ par Go. Chaque mois, l'utilisateur est prélevé par rapport à son utilisation global des services AWS. Il existe d'autres mesure de paiement, tels que le nombre de minutes / secondes de temps à utiliser le service ou bien de bande passante utilisé. Cette approche permet de maximiser la rentabilité d'un service car on ne paye pas mensuellement pour un service « illimité », mais on contrôle le prix à la source du service de manière bien plus précise. Cette approche est également efficace afin de convaincre les clients de payer notre service, car ils payent seulement ce qu'ils utilisent. Ainsi, que ce soit une très grosse entreprise ou un particulier, chacun peut profiter de la meilleur technologie possible sans être bloqué par un tarif de base très élevé. Ce modèle économique est très rentable, et AWS en démontre toute la puissance. La filiale est en effet la plus rentable de l'entreprise Amazon, avec 60% de marge net, soit 60% du chiffre d'affaire convertie en bénéfice.

1.2 Nos objectifs

Nos objectifs sont de vendre le plus de services possibles, tout en optimisant les coûts d'infrastructures, ces derniers étant quasiment la seule source de dépense de notre entreprise. Nous distinguerons trois types de dépenses : le monitoring, soit la mesure réel des coûts de nos services (en coût réel mais également en coût de calcul) ainsi que les incidents. De ce fait, les informations les plus utiles à la prise de décision au sein de l'entreprise seraient la vente de services, le coût réel de l'utilisation de nos services, les données associées à un incident (dates, coût estimé de la panne et estimation du nombre d'utilisateurs impactés) ainsi que le taux d'utilisation de nos services.

Nous tracerons en conséquence quatre types d'informations :

- La facturation des différents services proposés, représentant le prix mensuel à payer par les clients en fonction de leur consommation. Cette information est la plus importante, l'analyse de notre unique source de revenu est indispensable. Les traitements possible de cette information seraient :
  - Le service générant le plus de chiffre d'affaire.
  - Le pays avec les utilisateurs qui paye le plus.
  - Le mois est le plus rentable de l'année.
  - Le service le plus populaire (même s'il est pas le plus rentable).
  - Le type de service par secteur d'activité et le prix moyen dépense.
  - Évolution annuelle du prix unitaire moyen par service.

- Les coûts opérationnels, représentant le coût réel d'une opération, par exemple en coût d'électricité. Cette information est la plus importante après la facturation, elle représente le cœur de l'optimisation des coûts d'infrastructure. Les traitements possible de cette information seraient :
  - La localisation qui coûte le plus cher.
  - Le type de service qui pèse le plus au cours d'une année.
  - Le mois de l'année qui coûte le plus cher.

- Le traçage des différents incidents sous forme d'update records. Tout comme les coûts opérationnels, les incidents peuvent fortement alourdir les dépenses de notre entreprise. Leurs analyse est donc primordiale. Les traitements possible de cette information seraient :
  - Le type de la ressource la plus souvent en panne.
  - Le type de la ressource qui coûte le plus cher cher / impact le plus d'utilisateurs lors d'une panne.
  - Les agents ayant résolu le plus de pannes.
  - Classement des agents par performance (coût et volume).

- Le monitoring, représentant le taux d'utilisation des services mais également des coût de certaines opérations dans le service. Les traitements possible de cette information seraient :
  - Le service qui consomme le plus de CPU.
  - Quelles sont les utilisateurs les plus consommateur en ressource.
  - Le mois de l'année le plus solliciter.
  - Temps réel d'utilisation d'un service.

Le fichier SQL contenant les requêtes associées sont disponible sur le GitHub du projet dans : `./scripts/queries.sql`

2 Modèles en étoiles

Pour obtenir une vue complète et multidimensionnelle de l'activité sur la plateforme AWS, nous avons développé quatre modèles en étoile distincts. Chacun d'eux possède un modèle en étoile unique, une fonction spécifique et est optimisé pour un type d'étude ou d'analyse différent :

- Le modèle en étoile Facturation (Fig. 1) est le modèle principal pour l'analyse des coûts.
- Le modèle en étoile Actifs (Fig. 2) (si l'on considère les coûts totaux ou les états de ressources à un moment donné) est de type snapshot, car il capture une mesure à un instant T sans historique détaillé par transaction.
- Le modèle en étoile Incident (Fig. 3) est de type updated record, car les attributs sont mis à jour au fur et à mesure.
- Le modèle en étoile Monitoring (Fig. 4) est de type transactionnel, car il enregistre chaque événement ou mesure au fur et à mesure.

2.1 Facturation

Le modèle principal est celui de la facturation, AWS ne vendant que des services, l'entreprise ne peut pas gagner d'argent si elle n'en vend pas. L'analyse des entrées d'argent est crucial pour notre entreprise, ce modèle en étoile est le plus important car il répond à notre besoin de vendres le plus de services possibles :

2.1.1 Modèle en étoile

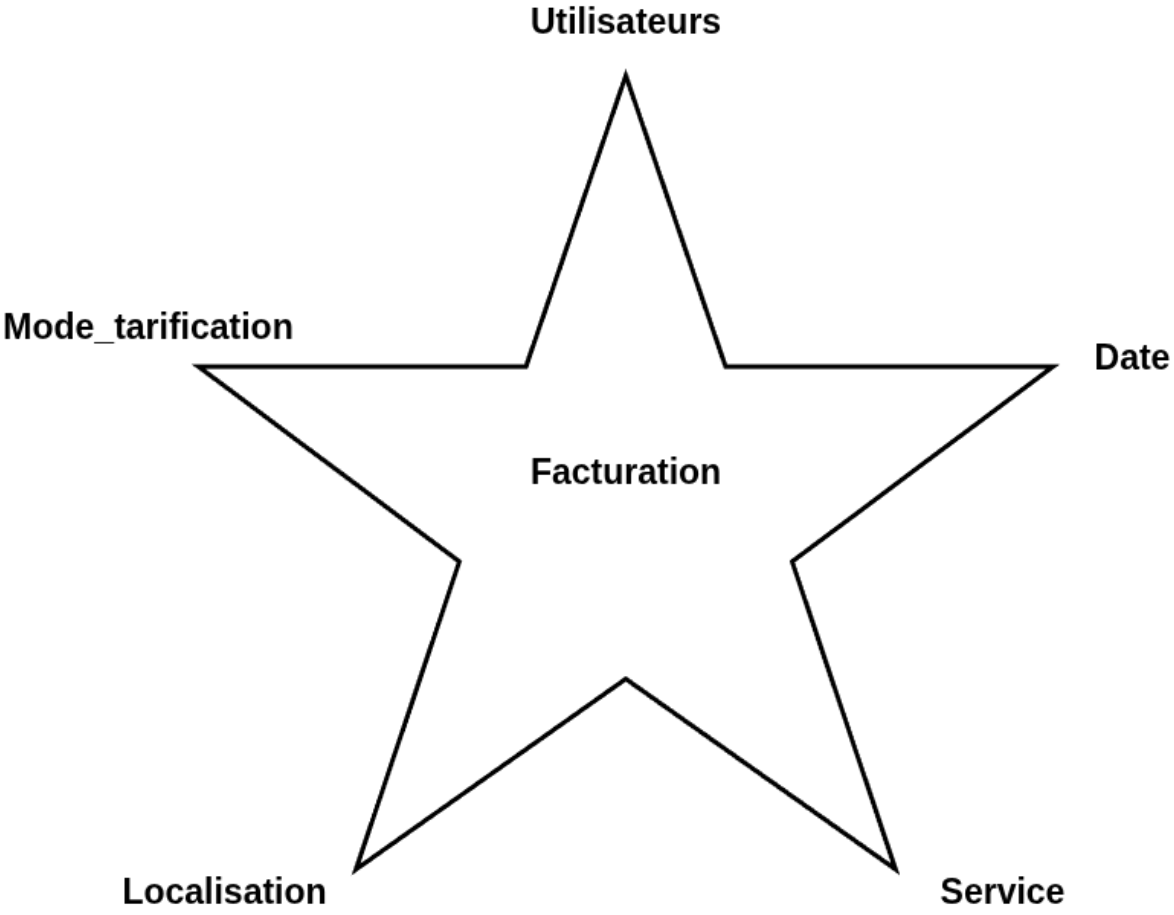

Fig. 1. – Modèle en étoile pour la facturation

Utilisateurs	Service	Localisation	Date	Mode_facturation
id_utilisateur	id_service	id_localisation	id_date	id_mode_facturation
type_utilisateur	code_service	ville	date_complete	code_mode_facturation
nom_famille	nom_service	code_ville	numero_jour	unite_mesure
prenom	categorie	departement_province	jour	categorie_usage
sexe	famille_service	pays	numero_mois	methode_paiement
date_de_naissance	statut_service	continent	mois	
nom_legal		fuseau_horraire	trimestre	
numero_identification		code_az	annee	
secteur_activite		nom_az	jour_semaine	
email			est_weekend	
telephone			jour_ferie	
pays_utilisateur			numero_semaine_annee	
date_inscription				

Dimensions

Facturation
id_utilisateur
id_service
id_localisation
id_date
id_mode_facturation
total_brute
total_taxe
remise
total_net
quantite_consommee
prix_unitaire

Table de fait

2.1.2 Listes des mesures

Les différentes mesures sont :

- total_brute : mesure additive représentant le revenue total hors taxe et remise, ce que gagne vraiment notre entreprise.
- total_taxe : mesure additive représentant le total de taxe à payer.
- remise : mesure additive représentant une possible réduction pour l'utilisateur.
- total_net : mesure additive représentant le prix final payé par l'utilisateur.
- quantité_consommé : mesure non additive représentant la quantité consommée par l'utilisateur, indépendamment de l'échelle (par exemple 10 pour 10go ou 14400 minutes, soit 10 jours).
- prix_unitaire : mesure non additive représentant le prix unitaire du service payé (par exemple 0.2 pour 0.2$).

Les mesures étant un prix à payé sont additives, c'est car peu importe la dimension, additionner un prix aura toujours un sens, sauf s'il est un prix unitaire ou bien une quantité consommée. On peut retrouver le prix brute en calculant la quantité consommée * le prix unitaire.

2.1.3 Exemple d'instance

Voici un exemple de ligne que l'on pourrait retrouver sans join dans Facturation :

id_utilisateurs	id_service	id_localisation	id_date	id_mode_facturation
12345	23	1	74854	8

Clés etrangères

total_brute	total_taxe	remise	total_net	quantite_consommee	prix_unitaire
1 000	57.7	40	1017.7	10 000	0.1

Mesures

- L'utilisateur 12345 à payé 1017.7$, soit notre total brute de 1000$ + 57.7$ de taxe (5.77%, en supposant que nous sommes en Virginie, soit l'AZ la plus grande de AWS), moins une remise de 40$.
- Notre utilisateur à payé le service 23 (Amazon RDS, location de base de données relationnels, avec MySQL) situé à la localisation 1 (AZ en Virginie) à la date 74854, correspondant au 01/01/2026.
- Il utilisera le mode de facturation 8, en payant le service en nombre de gigas octets, il aura dans notre cas payé pour $10,000$ go avec un prix unitaire de 0.1$ par go.

2.1.4 Estimation de la taille

Le nombre de ligne de cet entrepôt de donnée sera proportionnel au nombres d'utilisateurs, admettons que l'entreprise a $250,000$ utilisateurs actif qui ont au moins un service, on a alors une facturation minimum par an par client donc : $12 * 250,000 = 3,000,000$.
Sachant que $250,000$ est assez loin de la réalité (4.8m d'utilisateurs actif estimé en 2024) et que la limite de nombre d'entrée dans un fichier Excel (stable) est a peu près a 1.2 million, il est donc tout a fait justifié et pertinent de créer un entrepôts de données pour l'analyse de la facturation.

2.2 Actifs

AWS étant est une plateforme qui facture l'utilisation de ses ressources, il est fondamental de pouvoir analyser ses coûts (ou « actifs »). Ce schéma en étoile répond précisément à ce besoin d'analyse financière. Notre modèle étant un snapshot, il capture les résultats agrégés ou l'état cumulé des coûts à des points temporels définis, ce qui est l'objectif de base du reporting financier. Ce modèle est le premier et le plus important dans l'analyse des coûts d'infrastructure :

2.2.1 Modèle en étoile

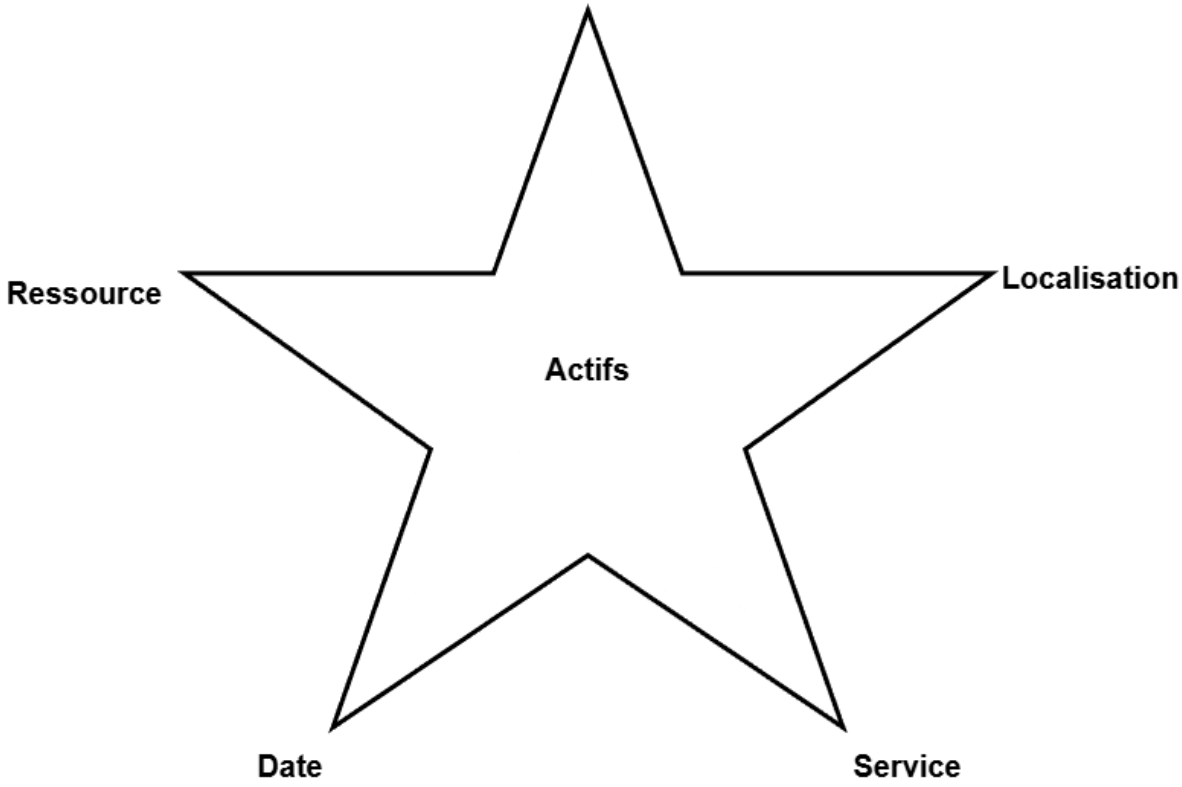

Fig. 2. – Modèle en étoile pour les actifs

Service	Localisation	Ressource	Date
id_service	id_localisation	id_ressource	id_date
code_service	ville	nom_ressource	date_complete
nom_service	code_ville	type_ressource	numero_jour
categorie	dpt_province	statut_ressource	jour
famille_service	pays	DT_mise_en_route	numero_mois
statut_service	continent	DT_derniere_maintenance	mois
	fuseau_horraire	DT_fin_service	trimestre
	code_az	IP	annee
	nom_az	OS	jour_semaine
		data_center	est_weekend
			jour_ferie
			numero_semaine_annee

Dimensions

Actifs
id_service
id_localisation
id_ressource
id_date
cout_total
cout_electricite
cout_maintenance
cout_materiel
cout_personnel
cout_theorique_max

Table de fait

2.2.2 Listes des mesures

Les différentes mesures sont :

- cout_total : mesure additive qui représente le coût total de la ressource.
- cout_electricite : mesure additive qui représente le coût électrique estimé de la ressource.
- cout_maintenance : mesure additive qui représente les frais de maintenance associés à la ressource. C'est l'addition de `cout_materiel` et `cout_personnel`.
- cout_theorique_max : mesure additive qui représente le coût théorique maximal pour une machine (en fonction de l'utilisation) atteint pendant la période de l'instantané.
- cout_materiel : mesure additive qui représente le coût du matériel utiliser. Elle inclue l'amortissement du matériel.
- cout_personnel : mesure additive qui représente le coût du personnel. Les employés, la maintenances et les frais de main d'œuvre.

On remarque que la majorité des montants sont additifs, car additionner les coûts (ex: 100 + 200) a un sens direct pour l'analyse financière.

2.2.3 Exemple d'instance

Voici un exemple de ligne que l'on pourrait retrouver sans join dans Actifs :

id_service	id_localisation	id_ressource	id_date
23	1	12345	74854

Clés étrangères

cout_total	cout_electricite	cout_maintenance	cout_materiel	cout_personnel	cout_theorique_max
330	180	150	100	50	500

Mesures

- cout_total (330) : C'est le coût total facturé pour la ressource qui a pour id 12345 à la date (qui a pour id) 74854. C'est l'addition de cout_electricite + cout_maintenance = $180 + 150 = 330$
- cout_electricite (180) et cout_maintenance (150) : Ce sont des sous-coûts, des mesures spécifiques qui sont incluses dans le coût total, mais qui sont suivies séparément. Si on prend l'exemple d'une data base a Paris, on sait que le coût d'électricité s'élève à 0,1952€ / kwH ce qui veut dire que l'utilisateur a consommer un total de 922.13KwH pour 180€ à payer si l'utilisateur habite dans une zone euro.
- cout_theorique_max (500) : Ce coût est logiquement plus élevé que le cout_total (330). Il représente ce qu'aurait coûté la ressource si elle avait été utilisée à 100 % de sa capacité pendant la période. Avec un nombre maximum de personnel, matériel et électricité utilisé.

2.2.4 Estimation de la taille

Le nombre de ligne de cet entrepôt de donnée sera principalement par rapport au nombre d'instantanés (snapshots) pris par jour ou par mois. Admettons que l'entreprise prend $250,000$ snapshots (ce qui est très peu pour une compagnie aussi grande que AWS) de ressources par mois, on a alors $12 * 250,000 = 3,000,000$ enregistrements par an au minimum, ce qui justifie la création d'un entrepôts de données pour l'analyse des actifs.

2.3 Incident

AWS loue des services, elle est donc confronté à toutes sortes d'incident, qu'il soit interne ou externe. Ce modèle en étoile permettra d'analyser les coûts réel de ces derniers. Ce dernier est le deuxième plus important dans l'analyse des coûts d'infrastructure :

2.3.1 Modèle en étoile

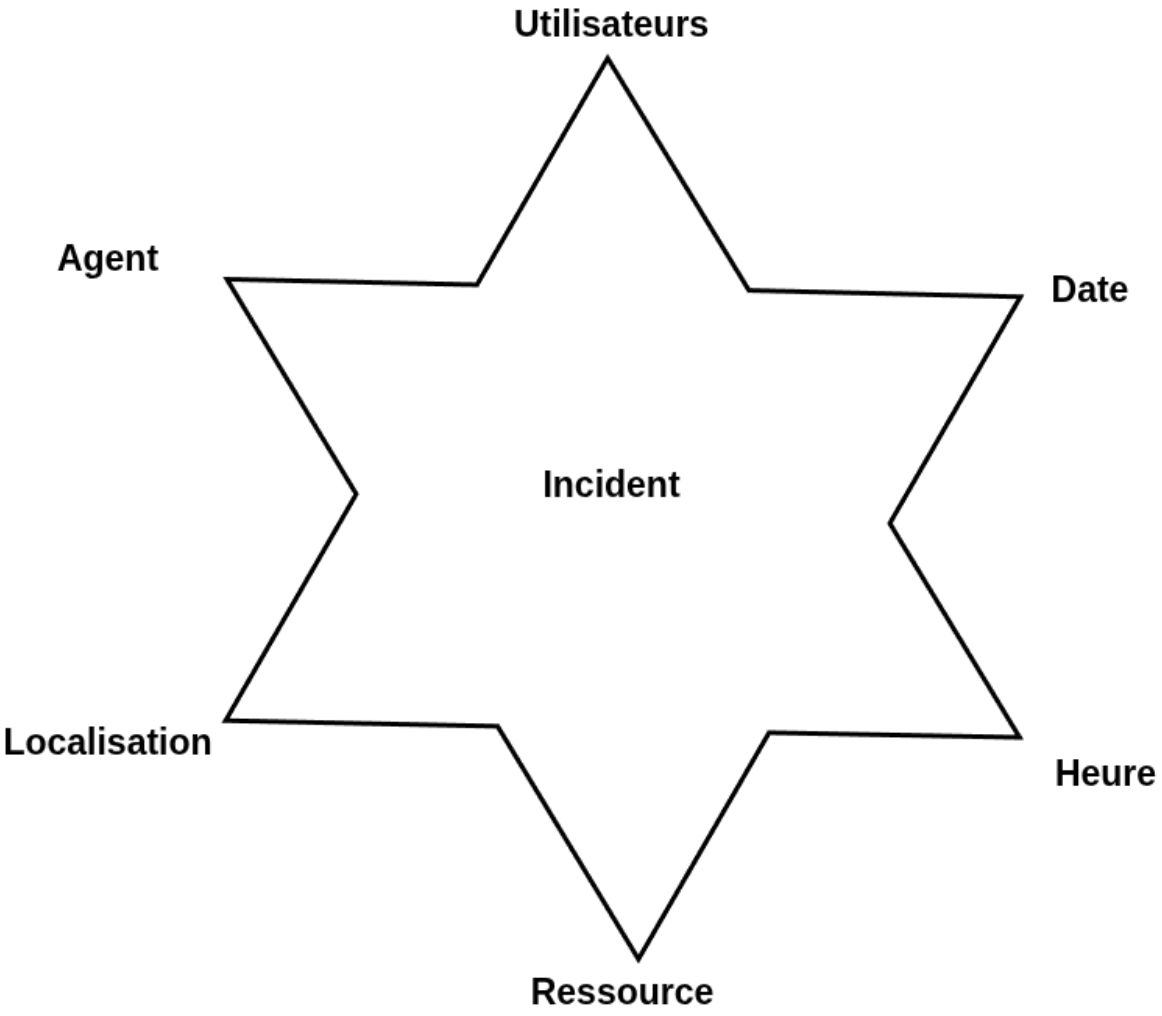

Fig. 3. – Modèle en étoile pour l'incident

Utilisateurs	Agent	Localisation	Ressource	Date	Heure	Statut
id_utilisateur_responsable	id_agent	id_localisation	id_ressource	id_date	id_heure	id_statut
type_utilisateur	matricule	ville	nom_ressource	date_complete	heure_complete	nom_statut
nom_famille	nom_agent	code_ville	type_ressource	numero_jour	heure_minute	
prenom	prenom_agent	departement_province	statut_ressource	jour	heure	
sexe	equipe	pays	DT_mise_en_route	numero_mois	minute	
date_de_naissance	niveau_support	continent	DT_derniere_maintenance	mois	seconde	
nom_legal	sexe	fuseau_horraire	DT_fin_service	trimestre	periode_journee	
numero_identification	date_de_naissance	code_az	IP	annee	est_heure_ouvrable	
secteur_activite	email	nom_az	OS	jour_semaine		
email	telephone		data_center	est_weekend		
telephone	date_entree			jour_ferie		
pays_utilisateur	nb_pause_toilette			numero_semaine_annee		
date_inscription						

Dimensions

Incident	
id_incident	
id_date_creation	*id_heure_creation*
id_date_estimation_resolution	*id_heure_estimation_resolution*
id_date_debut_traitement	*id_heure_debut_traitement*
id_date_mise_en_attente	*id_heure_mise_en_attente*
id_date_reprise	*id_heure_reprise*
id_date_resolution	*id_heure_resolution*
id_date_confirmation_utilisateur	*id_heure_confirmation_utilisateur*
id_date_cloture	*id_heure_cloture*
id_utilisateur_responsable	
id_agent	
id_localisation	
id_ressource	
cout_agent	
cout_remboursement_estime	
cout_client_estime	
nb_utilisateurs_impactes	

Table de faits

Les clés étrangères dans la table de fait apparaisse en italique. Les clés étrangères dans la table de fait commençant par id_date sont des clés étrangères de la dimension Date. De la même manière, les clés étrangère de la table de fait commençant par id_heure sont des clés étrangères de la dimension Heure.

2.3.2 Listes des mesures

- cout_agent : Mesure additive représentant le coût total estimé des salaires à payer a nos employés pour avoir réparer l'incident.
- cout_remboursement_estime : Mesure additive représentant le coût a rembourser a l'utilisateur pour l'incident.
- cout_client_estime : Mesure additive représentant le coût total estimé de l'incident pour l'utilisateur.
- nb_utilisateurs_impactes : Mesure additive représentant le nombre total estimé d'utilisateurs impacté par l'incident.

2.3.3 Exemple d'instance

Nous aurons ici deux instances pour décrires les deux types de pannes que nous aurons :
- Les pannes externes, provoqués par les client et non remboursé par AWS et attribué à un utilisateur.
- Les pannes internes, soumis au remboursement et et attribué à l'utilisateur NULL (id : 0).

Voici un exemple de ligne que l'on pourrait retrouver sans join dans Incidents pour une panne externe :

id_utilisateur_responsable	id_agent	id_localisation	id_ressource	satut
54321	73	1	65432	2

Clés étrangères

id_date_creation id_heure_creation	id_date_estimation_resolution id_date_estimation_resolution	id_date_debut_traitement id_heure_debut_traitement	id_date_mise_en_attente id_heure_mise_en_attente
12345 32400	12345 61200	12345 32700	12345 43200
id_date_reprise id_heure_reprise	**id_date_resolution id_heure_resolution**	**id_date_confirmation_utilisateur id_heure_confirmation_utilisateur**	**id_date_cloture id_heure_cloture**
12346 32400	12346 39600	NULL NULL	NULL NULL

Clés étrangères des dates et heures

cout_agent	cout_remboursement_estime	cout_client_estime	nb_utilisateurs_impactes
450	0	150000	4

Mesures

- L'utilisateur 54321, par exemple une entreprise de e-commerce, provoque une panne, par exemple une mauvaise mise à jour, à la localisation 1 (AZ en Virginie) à la ressource 65432 (par exemple la ressource hébergent l'API de paiement). L'agent 73 est responsable de la résolution de la panne. L'incident est encore en statut 2 (résolue, mais pas clôture).
- La panne à été signalé le 01/01/2026 à 09h00, avec une estimation de résolution le jour même à 17h00. Le début du traitement à commencer 5 minutes après, à 9h05. Elle à été mise en attente à 12h00 et repris le lendemain à 09h00. L'incident à été marqué comme résolue le 02/01/2026 à 11h00, le client doit encore confirmé qu'il peut réutiliser le service et ainsi pouvoir fermer administrativement l'incident.
- L'incident aurait coûté 450$ à notre entreprise, la faute étant au client nous comptons seulement le salaire de nos agents ayant réparé la panne dans `cout_agent`. Le coût estimé de la perte pour le client sera de 150 000$, soit 1 jour de chiffre d'affaire. Le nombres d'utilisateurs estimé serait de 4, avec 4 de nos clients utilisant cette API de paiement. L'attribut `cout_client_estime` pourra être mis a jour en fonction de la perte potentielle des autres utilisateurs subissant la panne de la ressource.

Nous aurions pu, dans ce scénario, imaginer que la panne vienne d'une mise à jour de notre système interne. L'utilisateur serait alors 0 (un utilisateur représentant notre propre entreprise) et le coût de remboursement estimé dépendrait du contrat entre le client et AWS. Par exemple, si nous devons rembourser 10$ par minute de panne, cela fait 14 400$ pour une panne de 24h. Nous aurions en conséquence à la place des 0 de cout_remboursement_estime : $450 + 14,400 = 14,850$$ de dédommagement.

2.3.4 Estimation de la taille

Nous avons estimé le nombre de client à $4,5M$, il n'existe pas de chiffre pour estimé le nombre d'incidents possible, mais en prenant une fourchette basse, par exemple 0.1 incident par utilisateur par mois : $(4,800,000 * 0.1) * 12 = 5,760,000$ enregistrements par an au minimum, ce qui justifie la création d'un entrepôts de données pour l'analyse des incidents.

2.4 Monitoring

AWS étant avant tout une compagnie qui loue de l'infrastructure, notamment du stockage, il est essentiel d'avoir une table qui « monitore » les machines louées. Notre modèle étant transactionnel, il capture chaque mesure de performance sans agrégation, ce qui est l'objectif de base du suivi technique. Ce modèle est le troisième plus important dans l'analyse des coûts d'infrastructure :

2.4.1 Modèle en étoile

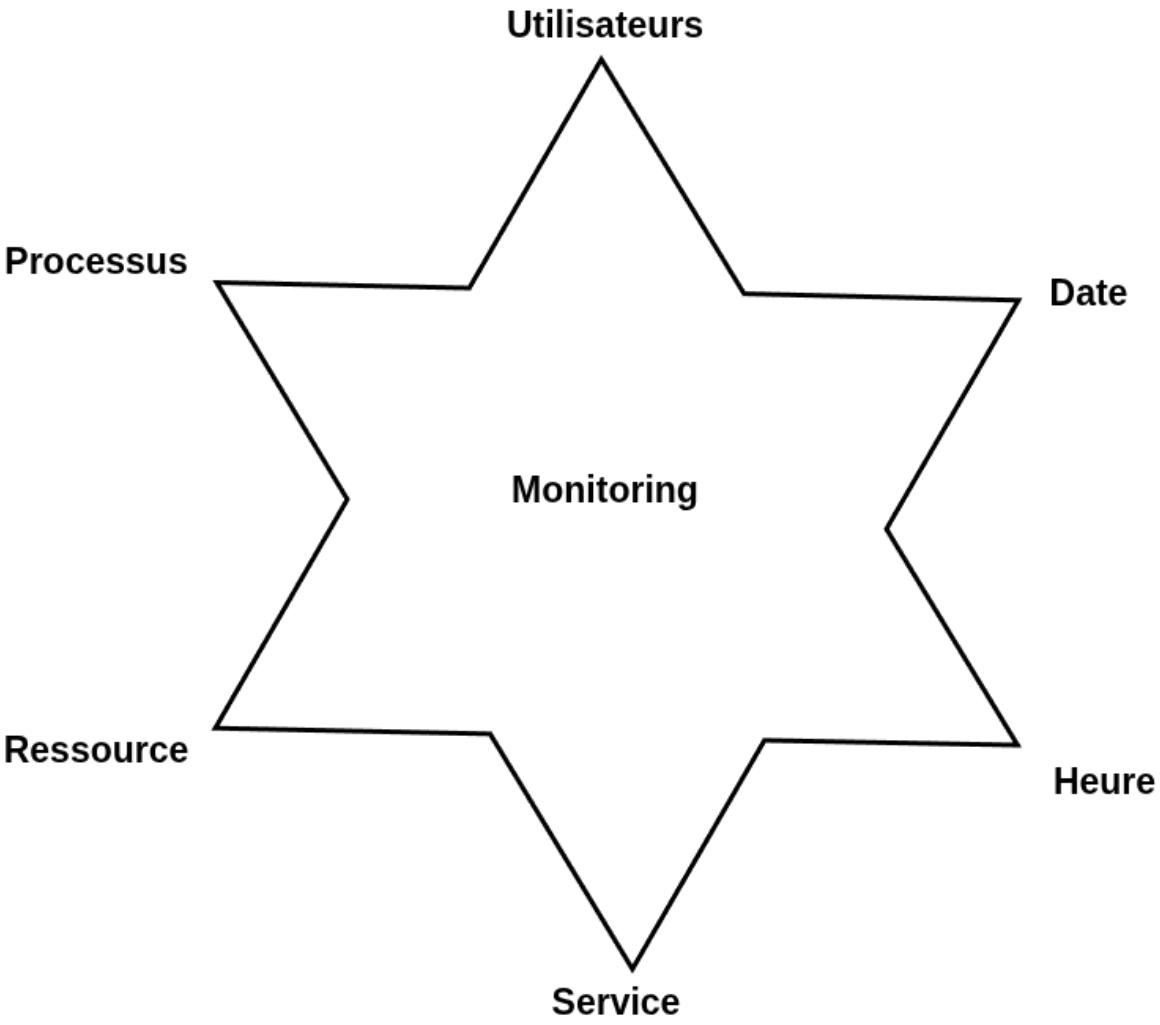

Fig. 4. – Modèle en étoile pour le monitoring

Utilisateurs	Ressource	Processus	Service	Date	Heure
id_utilisateur	id_ressource	id_processus	id_service	id_date	id_heure
type_utilisateur	nom_ressource	nom_processus	code_service	date_complete	heure_complete
nom_famille	type_ressource	type_processus	nom_service	jour	heure_minute
prenom	status_ressource	version	categorie	mois	heure
sexe	DT_mise_en_route	criticite	famille_service	trimestre	minute
date_de_naissance	DT_derniere_maintenance		statut_service	annee	seconde
nom_legal	DT_fin_service			jour_ferie	periode_journee
numero_identification	ip			jour_semaine	est_heure_ouvrable
secteur_activite	OS			est_weekend	
email	data_center				
telephone					
pays_utilisateur					
date_inscription					

Dimensions

Monitoring
id_utilisateur
id_ressource
id_processus
id_service
id_date
id_heure
CPU_utilisation
GPU_utilisation
RPM_ventilateur
RPM_GPU_ventilateur
est_actif
temps_total_actif
temperature
nb_core_utilises
nb_threads_utilises

Table de fait

2.4.2 Listes des mesures

Les différentes mesures sont :

- CPU_utilisation : mesure non additive représentant le taux d'utilisation du processeur.
- GPU_utilisation : mesure non additive représentant le taux d'utilisation de la carte graphique.
- RPM_ventilateur : mesure non additive représentant la rotation par minute des ventilateurs et donc signifiant le taux d'utilisation de la ressource.
- RPM_GPU_ventilateur : mesure non additive représentant la rotation par minute des ventilateurs du processeur graphique.

- est_actif : mesure non additive représentant si la ressource est actif ou non.
- temperature : mesure non additive représentant la chaleur du matériel.
- temps_total_actif : mesure non additive représentant depuis combien de temps la ressource est utilisée en minutes.
- nb_core_utilises : mesure non additive représentant le nombre de cœurs utilisés par l'application/processus.
- nb_thread_utilises : mesure additive représentant le nombre de thread utilisés par l'application/processus.

On remarque que les taux d'utilisation et la température ne sont pas additives, c'est car additionner deux pourcentages ou deux températures (ex: $10\% + 20\%$) n'a pas de sens physique. Seules les durées et les comptes peuvent être facilement additionnés.

2.4.3 Exemple d'instance

Voici un exemple de ligne que l'on pourrait retrouver sans join dans Monitoring :

id_utilisateur	id_ressource	id_processus	id_service	id_date	id_heure
23	12345	400	4	4567	32700

Clés étrangères

CPU_utilisation	GPU_utilisation	RPM_ventilateur	RPM_GPU_ventil	est_actif
97	30	1500	650	oui
temps_total_actif	**temperature**	**nb_core_utilisateur**	**nb_threads_utilises**	
480	80	6	15	

Mesures

- La transaction à été prise à la date 4567 (par exemple le 02/01/2026) à l'heure 32700 (soit 09h05) sur le service 4 (instance E2C, location de serveurs) dans la ressource 12345 (par exemple, la ressource se situe dans un data center à Paris) et concerne le processus 400 (par exemple, la compilation d'un gros programme).
- L'utilisateur 23 utilise le CPU à 97 %, avec une température de 80°C (relativement élevée) justifie la vitesse des ventilateurs, avec 1500 RPM pour le système principal et 650 RPM pour le GPU. Cela indique que le CPU est davantage sollicité que le GPU (67% de plus), ce qui suggère que l'utilisateur exécute une tâche peu graphique.
- Le nombre de cœurs utilisés (6) et de threads actifs (15) montre plus en détail l'intensité du processus en cours. Ces données peuvent être utiles pour l'analyse de la charge utilisateur et l'optimisation des performances du système.
- La ressource est active et elle est utilisé depuis 480 minutes soit 8h.

2.4.4 Estimation de la taille

Le nombre de ligne de cet entrepôt de donnée sera principalement par rapport au nombre d'événements, admettons que l'entreprise a $250,000$ ressources actives qui génèrent des données toutes les 5 minutes, on a alors $12 * 24 * 365 * 250,000 = 26,280,000,000$ enregistrements par an au minimum, on a donc tout fait justifier de créer un entrepôts de données l'analyse du monitoring.

3 Conception - Techniques Avancées de Modélisation

3.1 Tables Pont

Notre table pont sera nommé intervention et elle lie agent, ressource et incident.

L'objectif de cette table est d'identifier quels agents interviennent sur une ressource lors d'un incident. Nous ne connaissons pas à l'avance le nombre d'agents mobilisés pour une ressource lors d'une panne, nous savons seulement que l'agent « X » est intervenu sur la panne « Y ».

Ainsi, la création d'une table pont permettrait de lister tous les agents ayant travaillé sur la ressource associée à la panne « Y ». En plus, la relation Ressource – Agent est de type many-to-many, et cette structure offre des informations analytiques pertinentes.

La table aura donc pour clé primaire composite : *id_agent, id_incident et id_ressource*. Elle servira à identifier précisément les agents affectés à chaque ressource lors d'un incident.

Intervention
id_agent
id_incident
id_ressource

Un exemple de requête analytique sur cette table serait :

```SQL
SELECT a.prenom_agent, a.nom_agent, COUNT(DISTINCT iv.id_incident) AS nombre_incidents_resolus
FROM intervention iv
JOIN agent a ON iv.id_agent = a.id_agent
JOIN incident i ON iv.id_incident = i.id_incident
JOIN statut s ON i.id_statut = s.id_statut
WHERE s.nom_statut = 'closed'
GROUP BY a.id_agent, a.prenom_agent, a.nom_agent
ORDER BY nombre_incidents_resolus DESC;
```

Cette requête nous permet d'obtenir le nom et prénom des agents ayant réparé le plus grand nombre de pannes.

3.2 Dimension la plus volumineuse

La dimension la plus volumineuse est la dimension Utilisateurs, qui est estimé à 4.8 millions de lignes. Les attributs statiques et dynamiques de celle ci sont :

Attributs Statiques	
type_utilisateur	nom_famille
prenom	sexe
date_de_naissance	nom_legal
numero_identification	date_inscription

Attributs Dynamiques
secteur_activite
email
téléphone
pays_utilisateurs

3.3 Possibilitées de mise à jour des attributs dynamiques

- secteur_activite : Une entreprise peut changer de secteur d'activité, que ce soit en évoluant ou en régraissant. Par exemple Netflix avant son succès en ligne aurait pu être au début : `RETAIL` à : `MEDIA`. Cette attribue pouvant avoir de l'interet analytique, nous devrions en garder un historique, la stratégie de mise à jour sera donc de type 2, soit le row versionning. Il y aura donc une nouvelle entrée à chaque fois qu'une entreprise change de secteur d'activité. Il faudrat également rajouter trois attributs précisant la date de début, de fin et est_actif (Yes / No) pour préciser si c'est le secteur d'activité le plus récent de l'entreprise.
- email : Une entreprise ou bien un particulier peut mettre à jour son email, par exemple : `pio.basile@aws.com` à : `pio.basile@gmail.com`. Pour cette attribut, qui est une information de contact, nous pouvons la mettre à jour avec la stratégie de type 1, soit la reécriture sur l'attribut. Garder l'historique de cet attribut est négligeable.
- telephone : Une entreprise ou bien un particulier peut mettre à jour son numéro de téléphone, par exemple : `+33601020304` à : `+1 731 678 7253` (il déménage aux Etats Unis). Comme pour l'email, la stratégie de mise à jour sera de type 1.
- pays_utilisateurs : Une entreprise ou bien un particulier peut changer de pays, par exemple une entreprise française connaît une forte croissance et décide de s'exporter : `FRANCE` à : `IRELAND` (pour faire de l'optimisation fiscal). Comme pour le secteur d'activité, la stratégie de mise à jour sera de type 2.

3.4 Partitionnement

Nous allons partitionner notre table par ligne et par colonne :

- Pour le partitionnement par colonne, nous allons séparer par attributs statiques et attributs dynamiques.
- Pour le partitionnement par ligne, nous segmenterons les utilisateurs par pays et par secteurs d'activités.

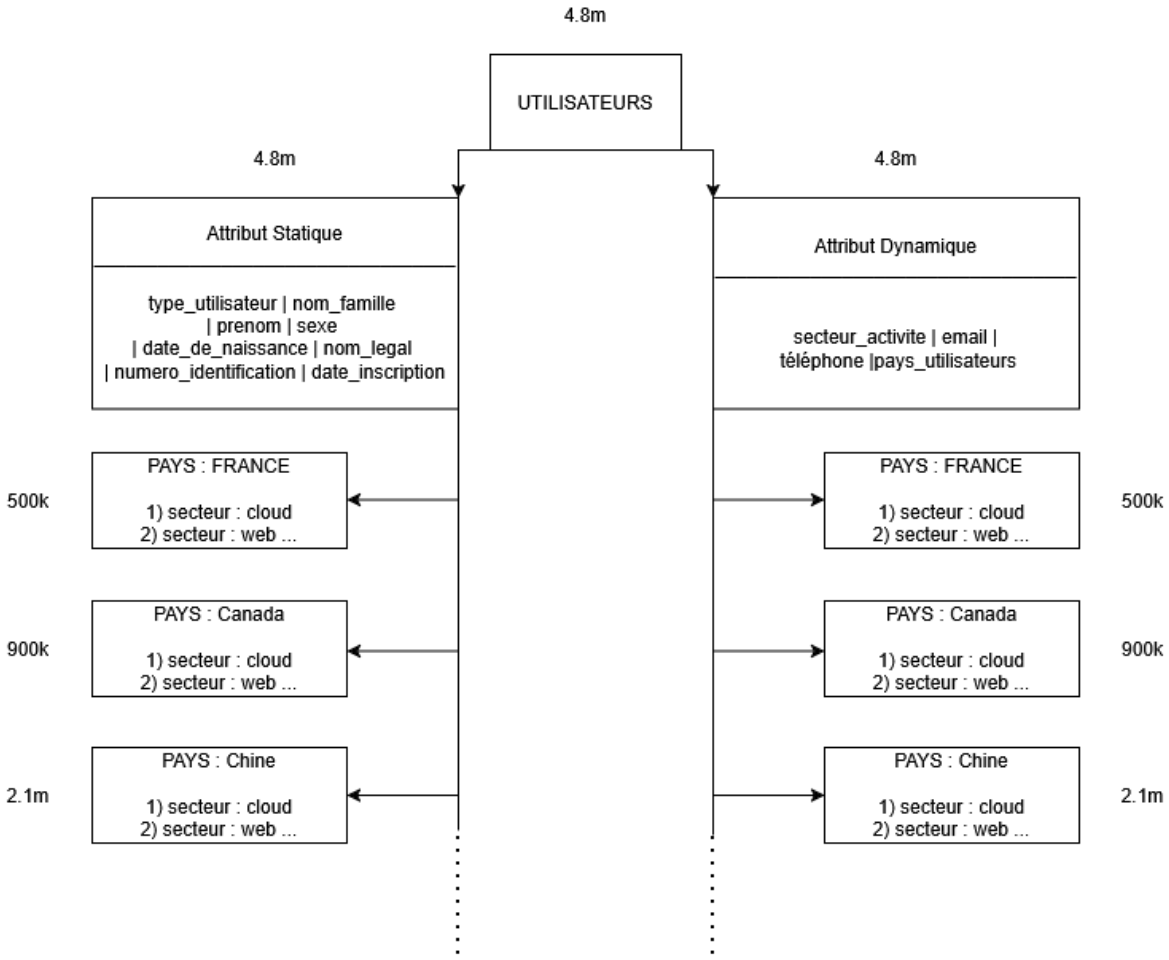

Fig. 5. – Partitionnement de la table utilisateurs 1

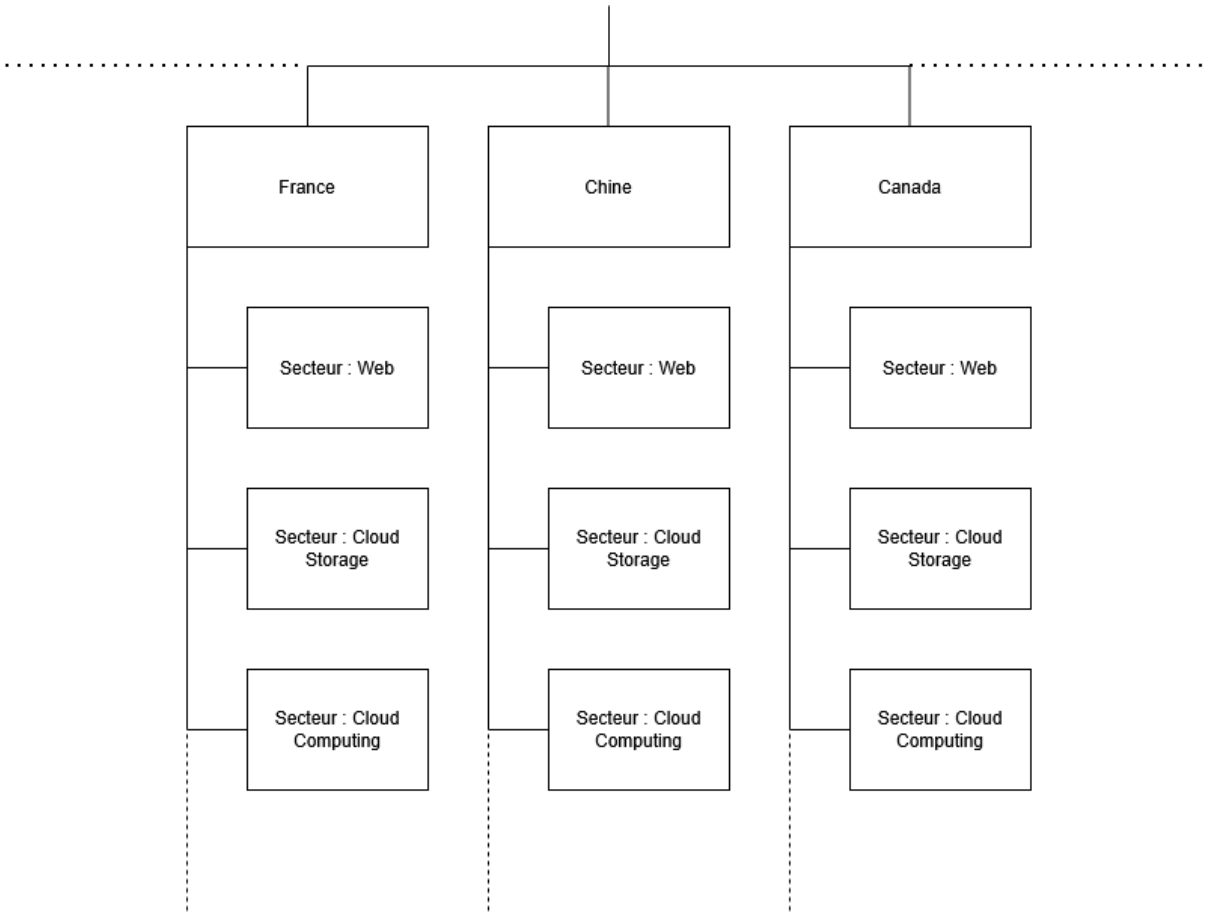

Fig. 6. – Partitionnement de la table utilisateurs 2

Grâce à ce modèle de partitionnement hybride, il devient possible de réduire considérablement les coûts d'exécution pour toutes les requêtes qui ne nécessitent qu'un seul type d'attribut. Les requêtes basées uniquement sur les attributs statiques ou dynamiques lisent uniquement la portion de table concernée, ce qui diminue fortement les volumes scannés et améliore donc les performances globales.

De plus, lorsqu'un attribut dynamique est modifié par exemple un email, un numéro de téléphone ou un secteur d'activité seule la partition dynamique est impactée. Cela limite fortement les opérations d'E/S et réduit la charge sur le système, puisque les attributs statiques, rarement modifiés, restent intacts. Cette séparation contribue directement à de meilleures performances et à une gestion plus efficace des mises à jour.

Enfin, le découpage horizontal des utilisateurs par pays puis par secteur d'activité offre un niveau supplémentaire d'optimisation : les requêtes ciblées (par région, par marché ou par domaine) ne traitent que des segments de données beaucoup plus restreints. Ce double niveau de partitionnement vertical et horizontal garantit à la fois une scalabilité optimale, une meilleure distribution de la charge de travail et une réduction significative des temps d'exécution.

4 Implémentation, requêtage, vues matérialisées et indexes bitmap

4.1 Implémentation Oracle

Le fichier SQL contenant la création des dimensions et des tables de faits est disponible sur le GitHub du projet dans : `./scripts/create_tables.sql`. Le fichier SQL contenant la création de la table pont est disponible dans : `./scripts/create_bridge_table.sql`.

4.2 Vues Virtuelles

Le fichier SQL contenant la création de la table pont est disponible dans : `./scripts/create_virtual_view.sql`. Ci dessous un extrait des vues virtuelles :

```sql
CREATE VIEW vue_utilisateurs_statique AS
    SELECT
        id_utilisateur,
        type_utilisateur,
        nom_legal,
        date_inscription
    FROM
        utilisateurs_statique;

CREATE VIEW vue_utilisateurs_dynamique AS
    SELECT
        id_utilisateur,
        secteur_activite,
        pays_utilisateur,
        est_actif
    FROM
        utilisateurs_dynamique;

CREATE VIEW vue_service AS
    SELECT
        id_service,
        nom_service,
        categorie,
        famille_service,
        statut_service
    FROM
        service;
```

4.3 Requêtes analytiques

On a proposer plusieurs requêtes analytiques qui sont tous présents sur le Github du projet dans : `./scripts/queries.sql`. Voici les requêtes analytiques présenté dans l'introduction :

```sql

```

```sql
-- Le service génère le plus de chiffre d'affaire
SELECT s.nom_service, SUM(f.total_brut) AS chiffre_affaire_total
FROM facturation f
JOIN vue_service s ON f.id_service = s.id_service
GROUP BY s.nom_service
ORDER BY chiffre_affaire_total DESC

-- Le pays avec les utilisateurs qui paye le plus
SELECT u.pays_utilisateur, SUM(f.total_brut) AS chiffre_affaire_total
FROM facturation f
JOIN vue_utilisateurs_dynamique u ON f.id_utilisateur = u.id_utilisateur
WHERE u.est_actif = 'Y'
GROUP BY u.pays_utilisateur
ORDER BY chiffre_affaire_total DESC

-- Le mois est le plus rentable de l'année
SELECT d.annee, d.mois, SUM(f.total_brut) AS revenu_mensuel
FROM facturation f
JOIN vue_date d ON f.id_date = d.id_date
-- WHERE d.annee = 2025 | Pour année spécifique
GROUP BY d.annee, d.mois
ORDER BY revenu_mensuel DESC

-- Le service le plus populaire (même s'il est pas le plus rentable)
SELECT s.nom_service, COUNT(DISTINCT f.id_utilisateur) AS nombre_utilisateurs
FROM facturation f
JOIN vue_service s ON f.id_service = s.id_service
GROUP BY s.nom_service
ORDER BY nombre_utilisateurs DESC

-- Le type de service par secteur d'activité et le prix moyen dépense
SELECT s.categorie, u.secteur_activite, AVG(f.total_brut) AS prix_moyen_depense
FROM facturation f
JOIN vue_service s ON f.id_service = s.id_service
JOIN vue_utilisateurs_dynamique u ON f.id_utilisateur = u.id_utilisateur
WHERE u.est_actif = 'Y'
GROUP BY s.categorie, u.secteur_activite

-- Évolution annuelle du prix unitaire moyen par service
SELECT d.annee, s.nom_service, AVG(f.prix_unitaire) AS prix_unitaire_moyen
FROM facturation f
JOIN vue_date d ON f.id_date = d.id_date
JOIN vue_service s ON f.id_service = s.id_service
GROUP BY d.annee, s.nom_service
ORDER BY d.annee, s.nom_service
```

4.4 Vues matérielles

Pour répondre efficacement à toutes tes requêtes analytiques en optimisant le stockage et la performance, il faut analyser les dimensions utilisées dans les clauses GROUP BY et JOIN.

On utilise donc la méthode treillis d'agrégation sur les dimensions de la facturation. Le résultat est la figure ci-dessous :

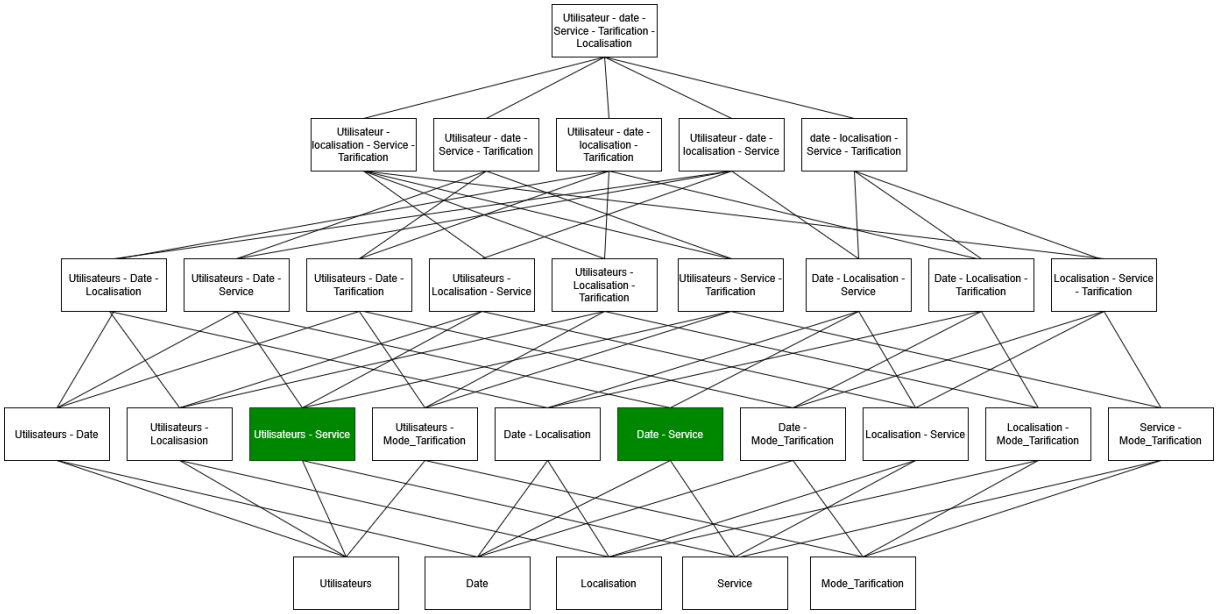

Fig. 7. – Lattice des dimensions de facturation

Nous avons d'abord écarté les dimensions « Localisation » et « Tarification » qui ne sont pas interrogées directement. L'attention s'est portée sur les trois axes restants : Utilisateurs, Service et Date.

Nous avons constaté une séparation évidentes dans les besoins d'analyse : certaines requêtes étudient le comportement client (croisement Utilisateurs / Service), tandis que d'autres suivent l'évolution temporelle (croisement Date / Service). Puisqu'aucune requête ne demande de lier simultanément un utilisateur spécifique à une date précise, il est inutile de maintenir un niveau de détail combinant les trois.

C'est pourquoi nous avons choisi de matérialiser les deux nœuds distincts en vert : [Utilisateurs - Service] et [Date - Service]. Cette approche évite de stocker la combinaison volumineuse de tous les utilisateurs sur toutes les dates, garantissant ainsi un volume de données plus léger et des performances de lecture supérieures.

Le fichier SQL contenant la création des vues matérialisées est disponible dans : `./scripts/create_materialized_view.sql`. Ainsi grâce à ces vues, lorsque l'on fera cette requête par exemple :

```sql
SELECT s.categorie, u.secteur_activite, AVG(f.total_brut) AS prix_moyen_depense
FROM facturation f
JOIN service s ON f.id_service = s.id_service
JOIN utilisateurs_dynamique u ON f.id_utilisateur = u.id_utilisateur
WHERE u.est_actif = 'Y'
GROUP BY s.categorie, u.secteur_activite
```

Oracle ira automatiquement chercher le montant moyen de total brut dans la vue matérialisé au lieu de le recalculer à chaque fois, optimisant ainsi grandement le temps d'execution de la requête.

4.5 Index de type bitmap

Cet index est conçu pour optimiser les requêtes suivantes :

```sql
-- Le service génère le plus de chiffre d'affaire
SELECT s.nom_service, SUM(f.total_brut) AS chiffre_affaire_total
FROM facturation f
JOIN vue_service s ON f.id_service = s.id_service
GROUP BY s.nom_service
ORDER BY chiffre_affaire_total DESC

-- Le service le plus populaire (même s'il est pas le plus rentable)
SELECT s.nom_service, COUNT(DISTINCT f.id_utilisateur) AS nombre_utilisateurs
FROM facturation f
JOIN vue_service s ON f.id_service = s.id_service
GROUP BY s.nom_service
ORDER BY nombre_utilisateurs DESC
```

L'index projette le nom_service (issu de la table Service) directement sur la table de faits facturation.

```sql
CREATE BITMAP INDEX BJI_CA_PAR_SERVICE
ON facturation (s.nom_service) -- Colonne de la dimension que l'on indexe sur la table de faits
FROM facturation f, service s
WHERE f.id_service = s.id_service; -- Condition de jointure entre les deux tables
```

De même, cet index permet d'éliminer la jointure entre facturation et Service. Toutes les requêtes de regroupement ou de comptage par nom de service (y compris le COUNT(DISTINCT id_utilisateur) pour la popularité) verront leurs performances considérablement améliorées, car l'accès à la dimension Service est rendu immédiat par l'index bitmap.

5 Annexe

Bibliographie

[1] D. M. S. Greg Bensinger Shubham Kalia, « Amazon says AWS cloud service back to normal after outage disrupts businesses worldwide », *Reuters*, 2025.

[2] J. King, « AWS Outage: How Virginia Data Centers Brought Down the Internet Worldwide », *NewsWeek*, 2025.

[3] https://aws.amazon.com/what-is-aws/, « What is AWS », *Amazon*.

[4] « Understanding How AWS Works », *Lotops*, 2024.

[5] Amazon, « How AWS Pricing Works », *https://docs.aws.amazon.com/whitepapers/latest/how-aws-pricing-works/abstract-and-introduction.html*, 2024.