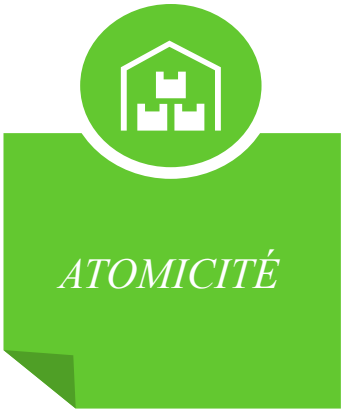


Propriétés ACID d'une base de données et les TRANSACTIONS

...

ACID

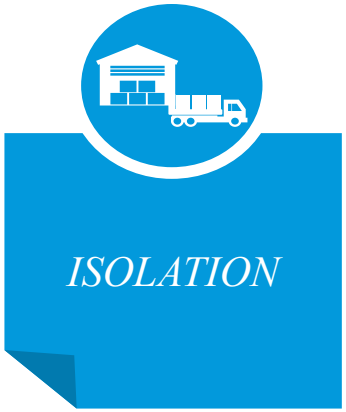
ACID



La transaction est effectuée ENTIÈRE ou pas du tout



C'est SEULEMENT les données valides qui sont insérées.



Les transactions n'affectent pas les AUTRES.



Les changement PERSISTENT



ACID

Ces 4 propriétés nous assurent que la base de données reste intègre.

Une des menace de l'intégrité est constamment la concurrence.

La concurrence c'est lorsque deux opérations tentent de s'exécuter en même temps et se retrouvent planifiées en parallèle par l'ordonnanceur et que le processeur alterne son travail entre les deux tâches laissant les choses en plan pour chacune. Si les deux tâches agissent sur les mêmes données et que les états ne sont pas consistants en milieu d'opération et que l'autre opération n'en tient pas compte.

INTÉGRITÉ

#ACID Properties in SQL - **Atomicité**

Les transactions comportent différentes instructions SQL ensemble. L'atomicité s'assure que chaque transaction est traitée comme un ensemble.

Dans l'**Atomicité**, les **transactions** peuvent soit exécuter, soit échouer. L'atomicité empêche les mises à jour de la base de données qui ne se produisent que partiellement

Si l'une des instructions de la transaction échoue, toutes les autres instructions échouent même celles déjà passées. Si toutes les instructions sont exécutées avec succès, la transaction est réussie.

En TRANSACTION : deux conclusions sont possibles :

- ROLLBACK : si une transaction est abandonnée, les modifications apportées à la base de données ne sont pas répercutées sur la base de données
- COMMIT : Si une transaction réussit, les données sont effectuées dans la base de données.

#ACID Properties in SQL - **Consistence**

La **consistence** ou **cohérence** consiste à maintenir une base de données afin que les données d'une base de données soient transférées d'un état valide à un autre état valide. Cela fait référence à l'exactitude des données.

Le principe de cohérence est de s'assurer que les données modifiées dans la base de données ne sont pas corrompues

Par exemple, dans une application qui transfère des fonds d'un compte à un autre, la propriété de cohérence garantit que la valeur totale des fonds dans les deux comptes est la même au début et à la fin de chaque transaction.

#ACID Properties in SQL - **Isolation**

L'**isolation** est le principe garantissant que plusieurs transactions qui se produisent simultanément ne laissent pas la base de données de manière incohérente.

La transaction en cours n'interfère pas avec les autres et s'exécute indépendamment

Les modifications qui se produisent dans une transaction particulière ne seront visibles pour aucune autre transaction jusqu'à ce que la transaction soit terminée (que tout soit mémorisé et validé par la base de données).

Ce principe garantit que l'exécution simultanée de transactions aboutira à un état identique à si elles avaient été exécutées consécutivement, l'une après l'autre.

Voir https://theaws.blog/acid-cheat-sheet/

#ACID Properties in SQL - **Durabilité**

La **durabilité (persistence)** est utilisée pour garantir qu'une fois les données validées, elles resteront les mêmes et ne seront pas affectées par une défaillance du système.

Cela signifie généralement que les transactions terminées (ou leurs effets) sont enregistrées en mémoire

Par exemple, dans une application qui transfère des fonds d'un compte à un autre, la propriété de durabilité garantit que les modifications apportées à chaque compte ne seront pas annulées.

La transaction terminée devient irréversible. Alors que lorsqu'elle était en cours, n'importe quel problème causait un ROLLBACK de tout, quand elle est COMMIT le rollback n'est plus possible.

Cela est différent de décider de faire la transaction inverse pour par exemple REMBOURSER un client ayant effectué un PAIEMENT.

#ACID Properties in SQL - Exemples

https://stackoverflow.com/a/999447

- **Atomicity** - Quentin transfère 1000$ vers le compte de son déneigeur Steve. Si l'électricité coupe en plein milieu, les deux opérations (retrait et dépot) sont annulées et non pas une seule.
- **Consistency** - Après un transfert, le total des deux comptes reste le même.
- **Isolation** - Un autre client transfère un paiement au déneigeur en même temps que Quentin. Cette opération se déroule séparément (après ou avant) et n'a pas d'effet de bord sur le résultat de l'opération de Quentin.
- **Durability** - Si la transaction de Quentin se termine, elle persiste pour toujours. Cette propriété implique que les 3 autres propriétés ont été respectées et que la transaction a terminé

Pour l'intégrité



- PRINCIPES
- ATOMICITÉ
- CONSISTENCE - COHÉRENCE
- ISOLATION
- DURABILITÉ
- EXEMPLES

#ACID Properties in SQL

https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions

#ACID Properties in SQL - Exemples

https://www.educative.io/edpresso/what-are-acid-properties-in-a-database

https://web.archive.org/web/20110823114953/http://www.archwing.com/technet/technet_ACID.html

https://stackoverflow.com/a/999447

https://www.csestack.org/acid-properties-dbms-examples/

#ACID Properties in SQL - Exemples

https://stackoverflow.com/a/999447

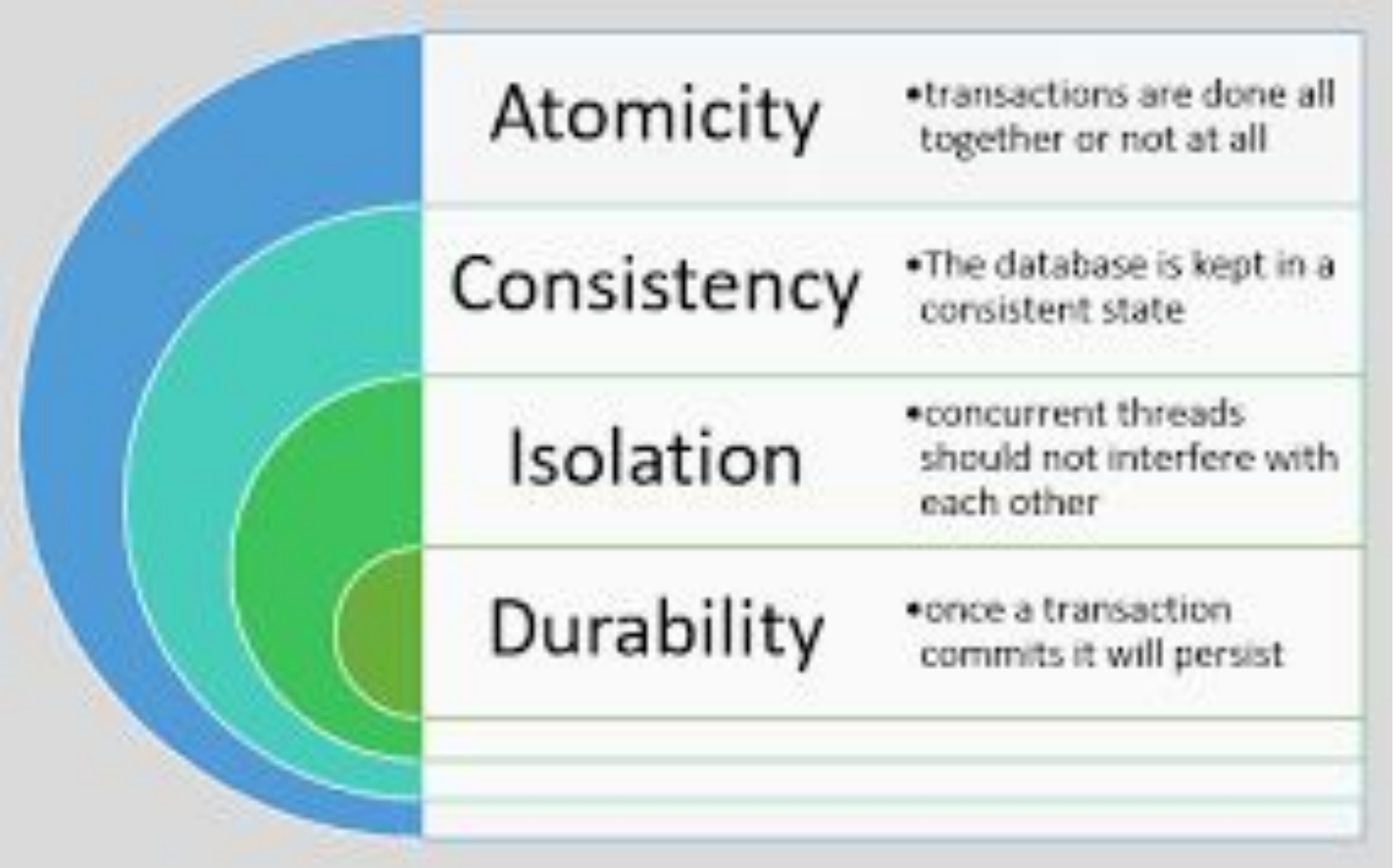
- **Atomicity** - Quentin transfère 1000$ dans le compte pendant que Alice retire 1000$ - Une transaction doit terminer avant que l'autre commence.
- **Consistency** - une base de données qui suit un compte chèque ne peut autoriser que des numéros de chèque uniques pour chaque transaction.
- **Isolation** - un caissier qui consulte un solde doit être isolé d'une transaction concurrente impliquant un retrait sur le même compte. Ce n'est que lorsque la transaction de retrait est validée avec succès et que le caissier consulte à nouveau le solde que le nouveau solde est affiché.
- **Durability** - Une panne du système ou toute autre défaillance ne doit pas faire perdre les résultats d'une transaction ni le contenu de la base de données. La durabilité est souvent assurée par des journaux de transactions séparés pouvant "recréer" toutes les transactions (backup).

#ACID Properties in SQL - Exemples EN

https://stackoverflow.com/a/999447

- **Atomicity** - a transaction to transfer funds from one account to another involves making a withdrawal operation from the first account and a deposit operation on the second. If the deposit operation failed, you don’t want the withdrawal operation to happen either.
- **Consistency** - a database tracking a checking account may only allow unique check numbers to exist for each transaction
- **Isolation** - a teller looking up a balance must be isolated from a concurrent transaction involving a withdrawal from the same account. Only when the withdrawal transaction commits successfully and the teller looks at the balance again will the new balance be reported.
- **Durability** - A system crash or any other failure must not be allowed to lose the results of a transaction or the contents of the database. Durability is often achieved through separate transaction logs that can "re-create" all transactions from some picked point in time (like a backup).

#ACID Properties in SQL



Et en NoSQL ?

Les bases de données relationnelles suivent fortement les propriétés ACID (Atomicity, Consistency, Isolation, Durability) tandis que les bases NoSQL suivent les principes BASE (Basically Available, Soft State, Eventual consistency).

Basically **A**vailable	• Prioriser la disponibilité des données.
Soft state	• La propagation des données a la priorité, tandis que les vérifications d'incohérence sont déléguées à l'externe.
Eventual consistency	• On suppose que les incohérences temporaires finiront par se résorber dans un état final stable.

Et en NoSQL ? - EN

The relational databases strongly follow the ACID (Atomicity, Consistency, Isolation, and Durability) properties while the NoSQL databases follow BASE (Basically Available, Soft State, Eventual consistency) principles.

Basically **A**vailable	• Prioritizing the availability of data.
Soft state	• The propagation of data is given priority, while checks on inconsistency are delegate to external elements.
Eventual consistency	• It is assumed that temporary inconsistencies will in the end be sorted out in a stable final state.

MERCI