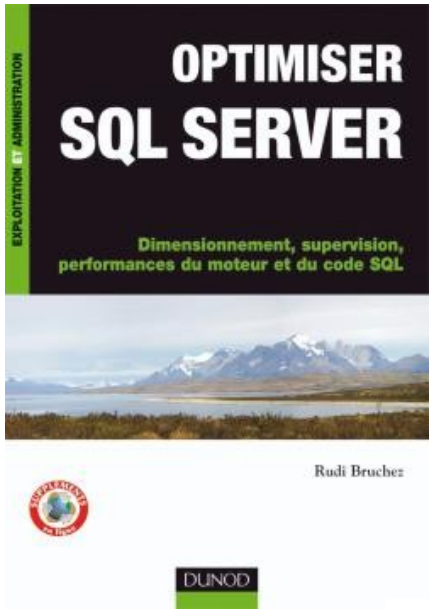
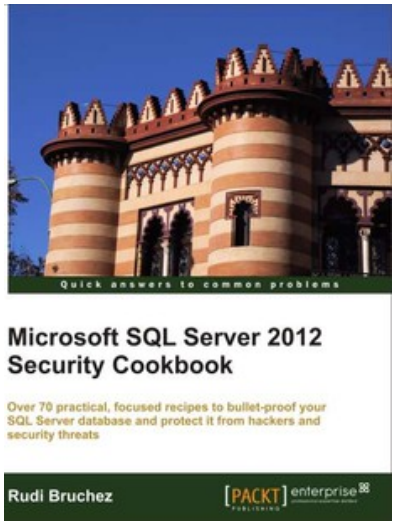
SQL Server Diagnostic

Cours
SQL Server 20XX diagnostic
et résolution de problèmes

Bienvenue !

Rudi Bruchez





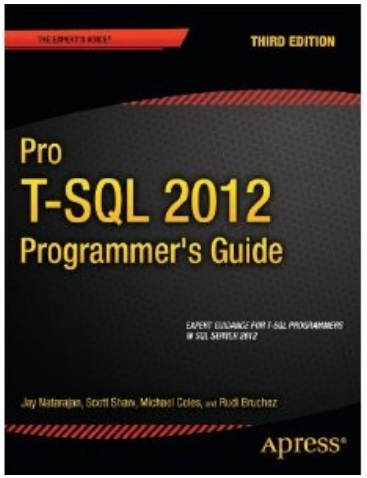
http://www.babaluga.com/

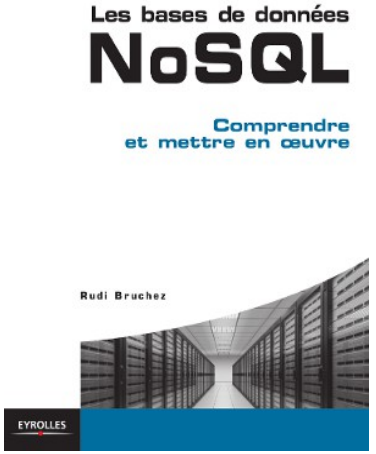
rudi@babaluga.com

http://rudi.developpez.com

http://www.video2brain.com/fr

http://www.developpez.net/forums/f49/bases-donnees/ms-sql-server/





Microsoft
CERTIFIED
Solutions Associate
SQL Server 2012

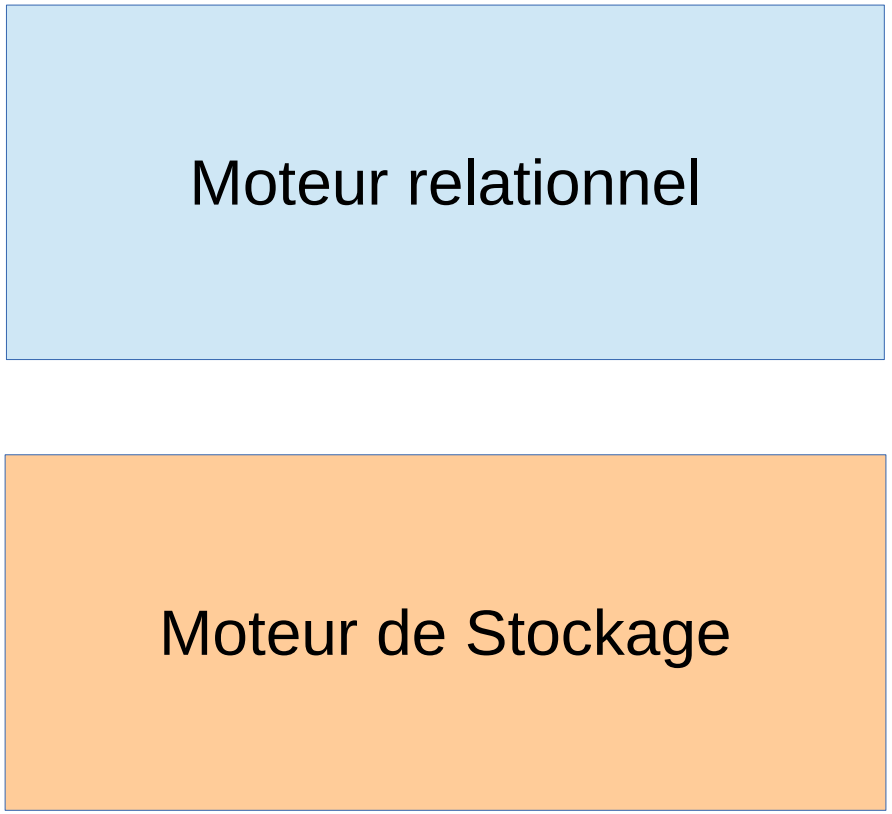
Programme

- 1. Comprendre le fonctionnement du moteur SQL Server
  - Fonctionnement interne de SQL Server
  - SQL OS
  - L'optimiseur
- 2. Les méthodologies et les étapes du diagnostic
  - Méthodologies
  - outils
- 3. Gérer l'exécution des requêtes
- 4. Les problématiques classiques et leur résolution

Chapitre 1
Comprendre le fonctionnement du moteur

- L'architecture générale de SQL Server
- SQLOS
- Le moteur relationnel et le moteur de stockage
- L'optimiseur de requête
- L'exécution des requêtes

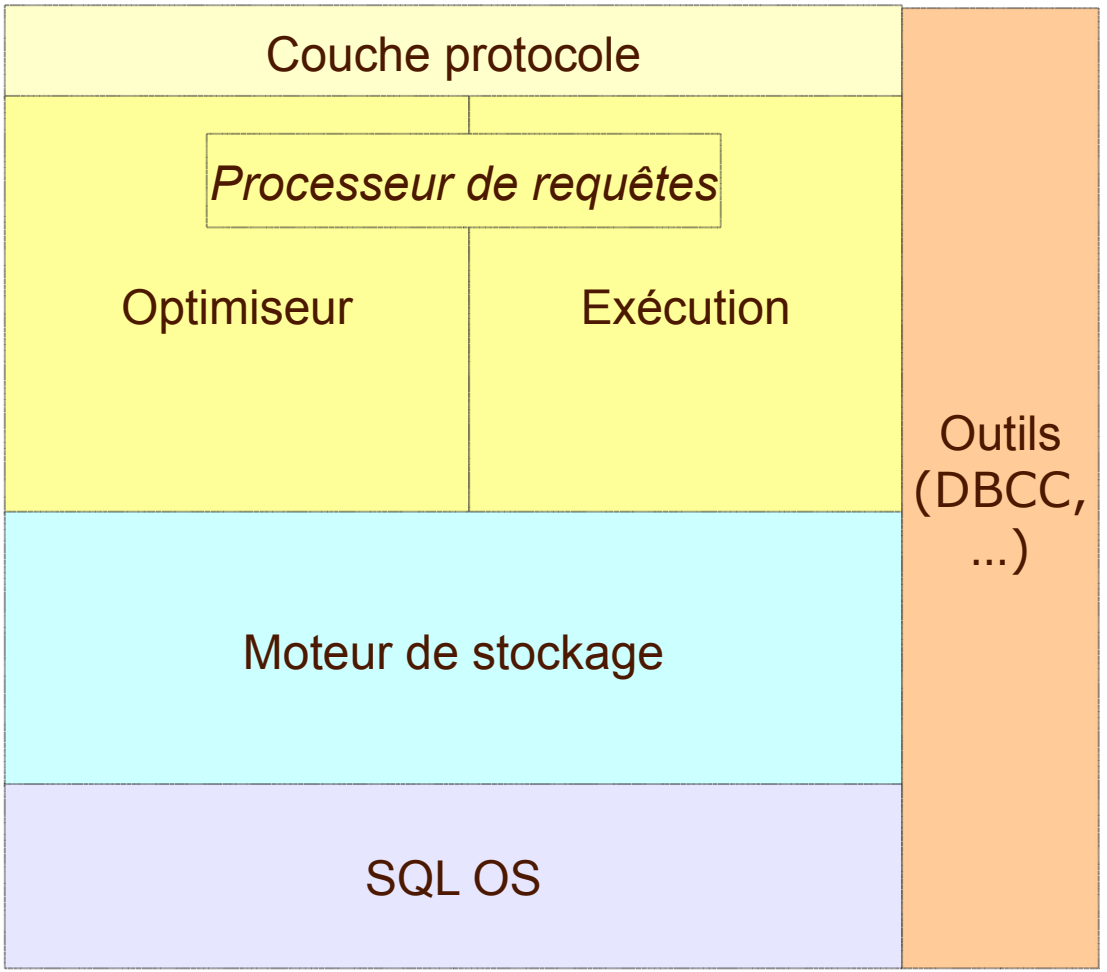
Comprendre le moteur
Moteur relationnel – dychotomie des moteurs



- Séparation classique permettant l'abstraction de la couche physique des données
- Le **moteur relationnel** interprète et exécute le code SQL
- Le **moteur de stockage** gère le stockage en RAM et sur disque, et les éléments transactionnels.

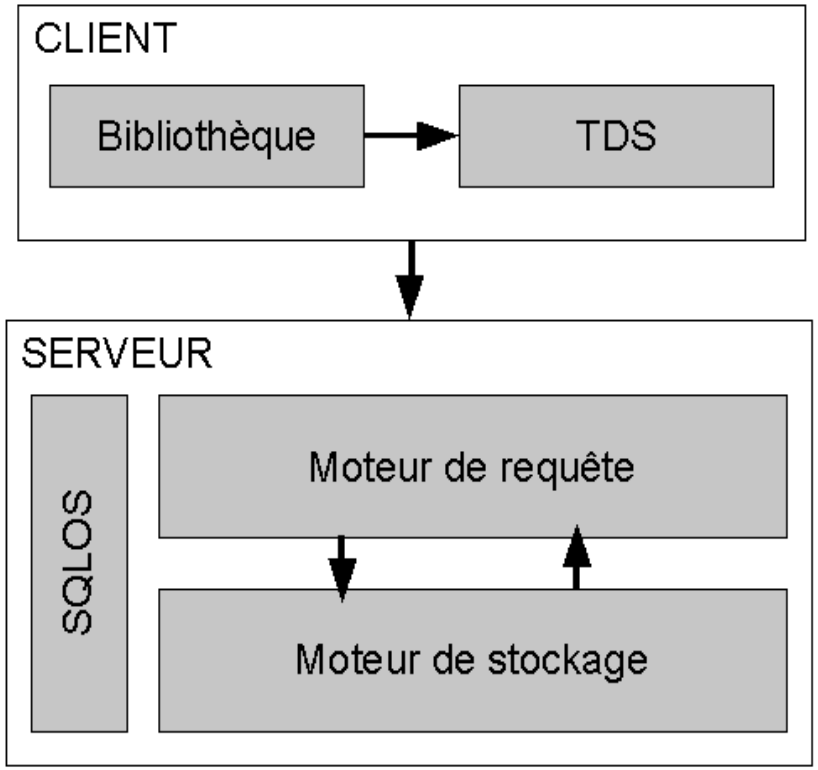
SQL OS

Comprendre le moteur
Architecture générale – Architecture MS SQL Server



- La couche de protocole échange des informations avec le client en TDS
- Le moteur relationnel s'occupe de commandes T-SQL
- Le moteur de stockage gère physiquement les données
- SQLOS adresse les ressources de la machine

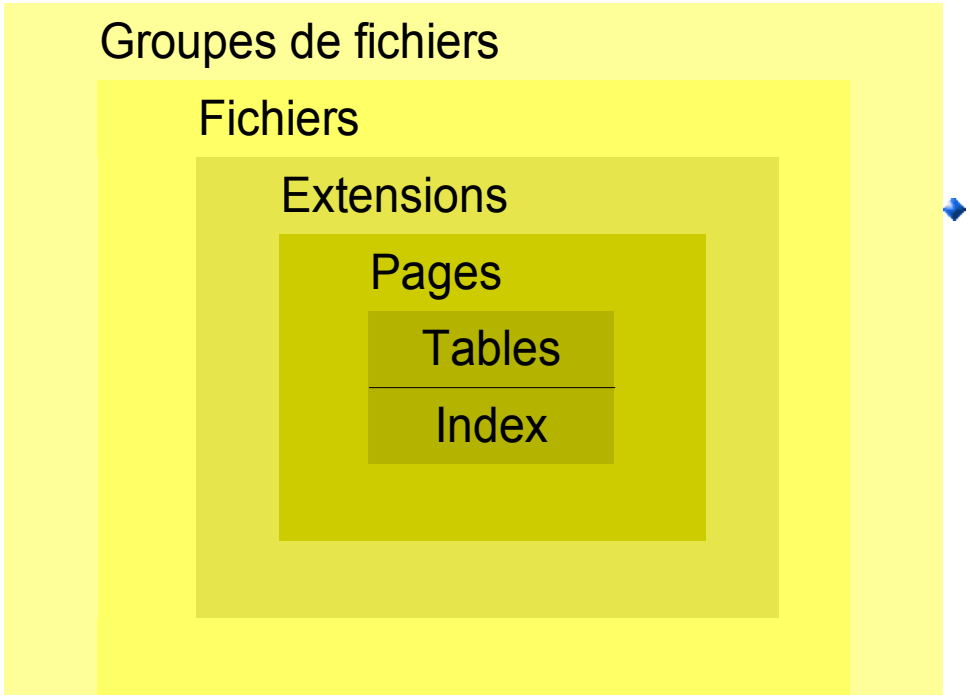
Comprendre le moteur
Architecture générale – protocole réseau



- Le protocole couche réseau (OSI) de SQL Server (et Sybase) est TDS
  - Tabular Data Stream

Internal error: The operation terminated unsuccessfully. OLE DB error: OLE DB or ODBC error: Protocol error in TDS stream; HY000; Protocol error in TDS stream; HY000; Protocol error in TDS stream; HY000; Communication link failure; 08S01; TCP Provider: An existing connection was forcibly closed by the remote host.

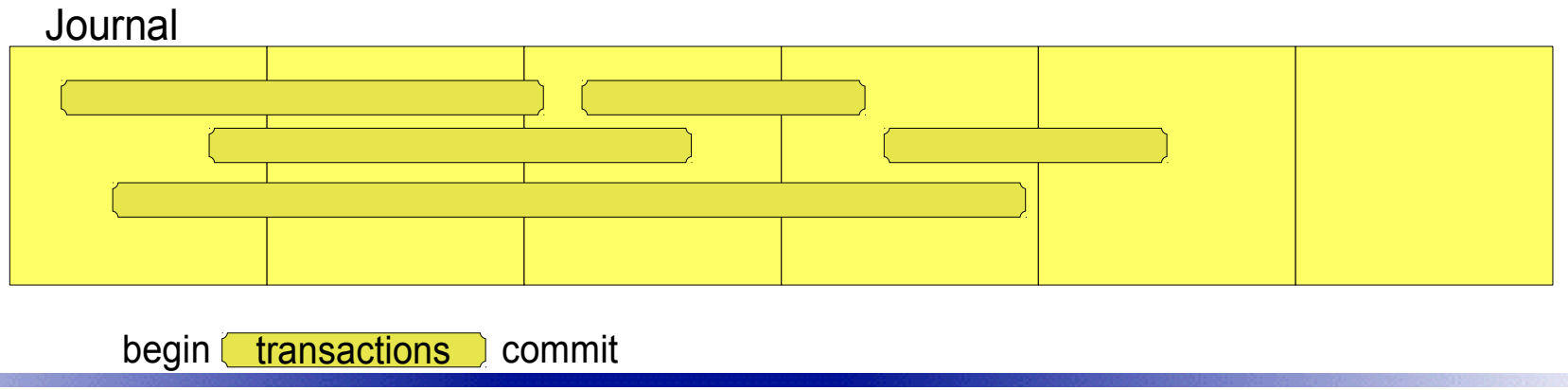
Comprendre le moteur SQLOS – gestion du disque



Les données sont stockées dans un fichier structuré

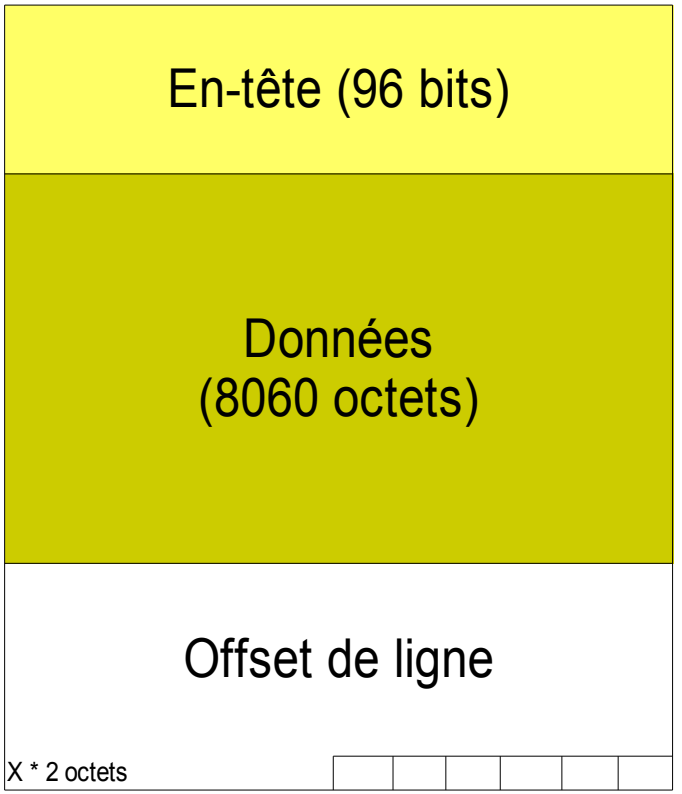
- L'allocation est effectué par pages de 8 ko

La durabilité de la transaction est assurée par un WAL (Write-ahead log)

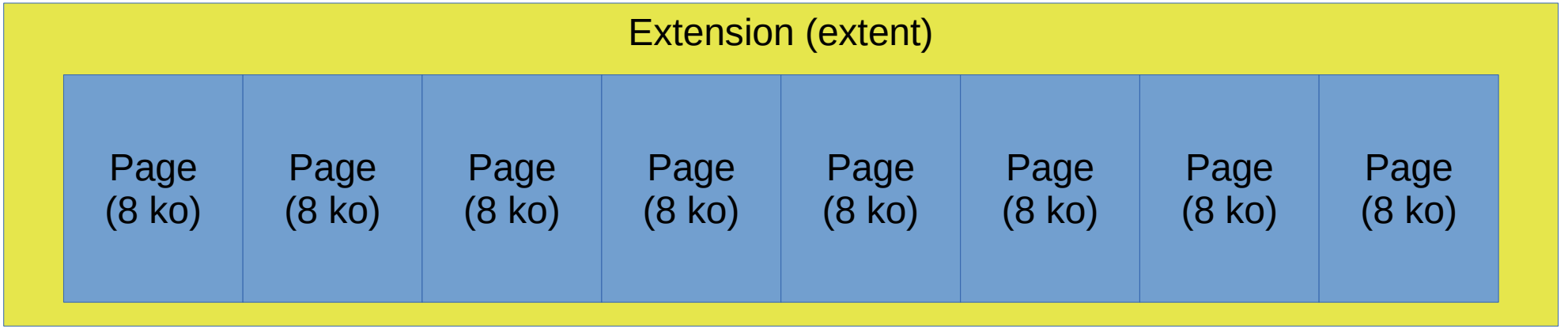


Comprendre le moteur
Moteur de stockage – les pages

- 8 ko max
- Unité minimale d'E/S
- DBCC PAGE pour voir son contenu
- Tout ce qui est dans le fichier de données, et en RAM, est dans des pages

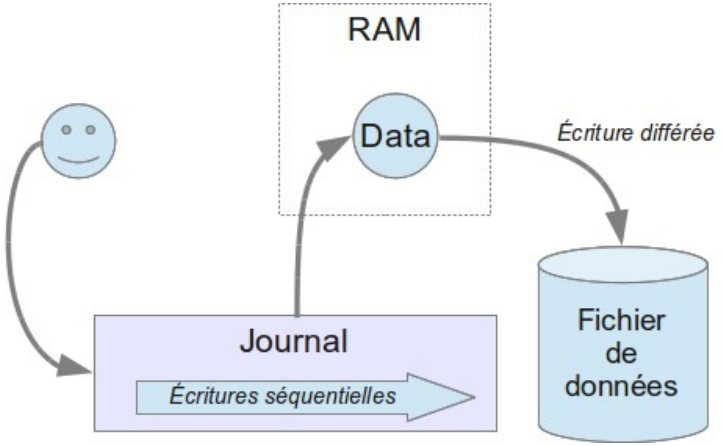


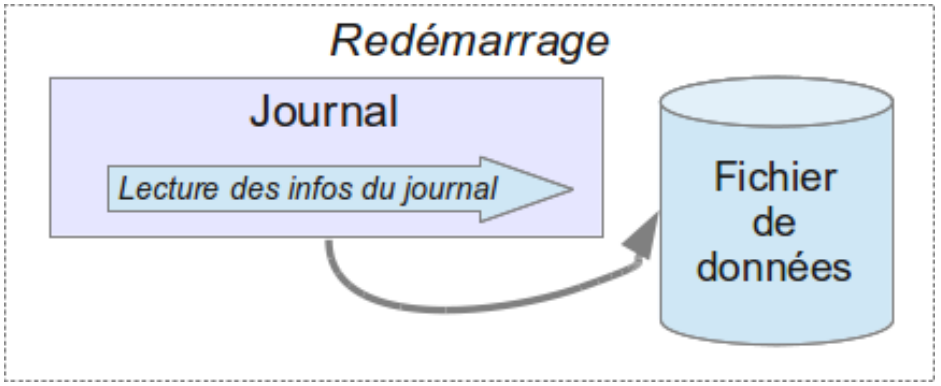
Comprendre le moteur
Moteur de stockage – les extensions



- L'extension est l'unité d'allocation du moteur de stockage
  - 8 pages de 8 ko = 64 ko
- L'extension peut être :
  - Mixte = contient des pages de d'objets différents
  - Uniforme = contient des pages du même objet

Comprendre le moteur
Moteur de stockage – le journal de transactions





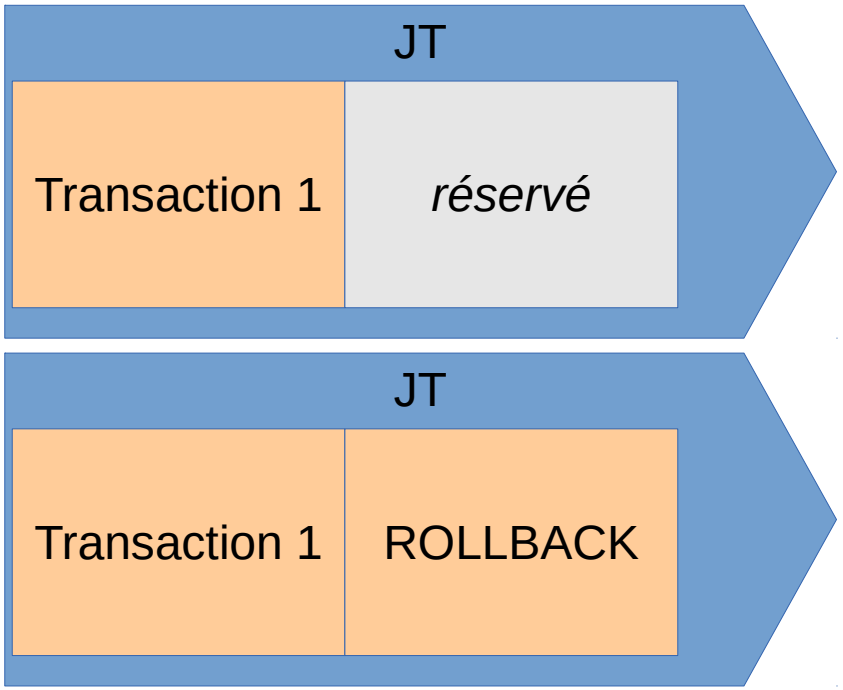
- Le journal de transaction est un WAL (write-ahead log)
- Il assure la durabilité de la transaction
- Il n'y a pas d'écriture directement sur le fichier de données, pour des raisons de performance
  - Écritures séquentielles vs écritures aléatoires

Comprendre le moteur
Moteur de stockage – structure du JT

- Le journal de transaction est un fichier à écritures séquentielles
- Il est segmenté en interne en VLFs (Virtual Log Files)
  - DBCC LOGINFO WITH no_infomsgs;
- Le nombre de VLF influe sur les performance du journal
  - À la création, le moteur de stockage dimensionne de 4 à 16 VLFs
- Pour corriger les JT qui comportent trop de VLFs, DBCC SHRINKFILE + ALTER DATABASE

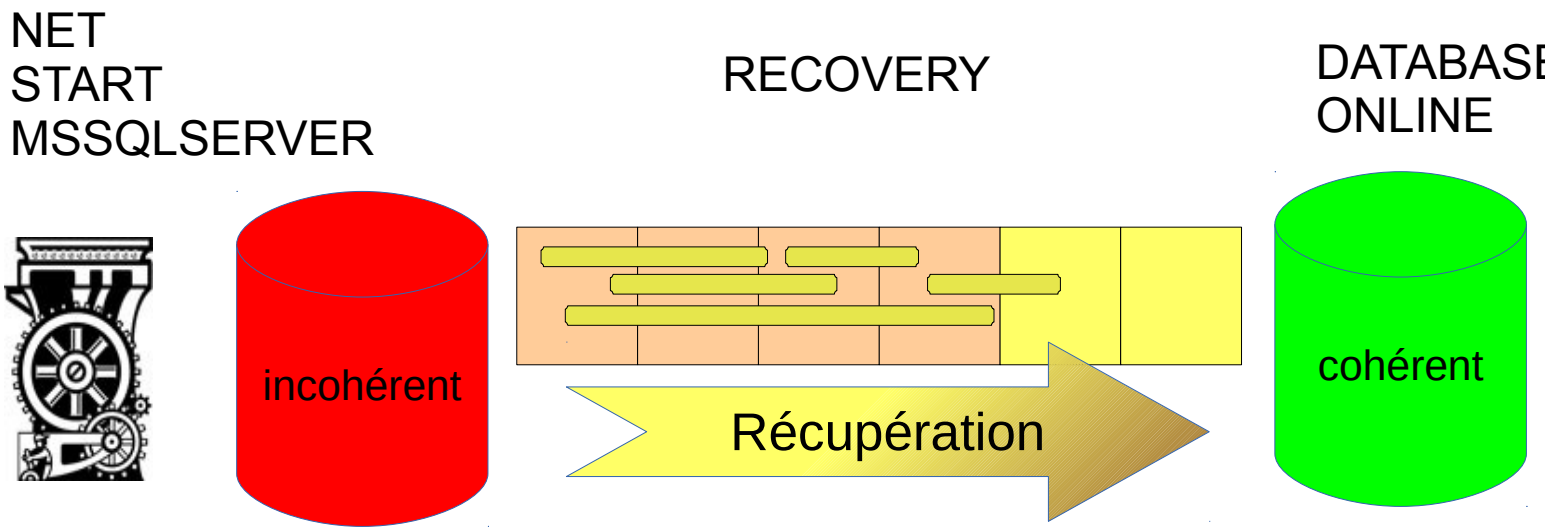
Comprendre le moteur
Moteur de stockage – le JT – rollback

- Le ROLLBACK consiste à défaire, instruction par instruction, ce qui a été fait dans la transaction
- Les instructions de ROLLBACK sont écrits dans le JT
- Il faut donc que le JT ai toujours assez d'espace libre pour effectuer le ROLLBACK des transactions en cours
- C'est possible car la transaction réserve de l'espace pour cela



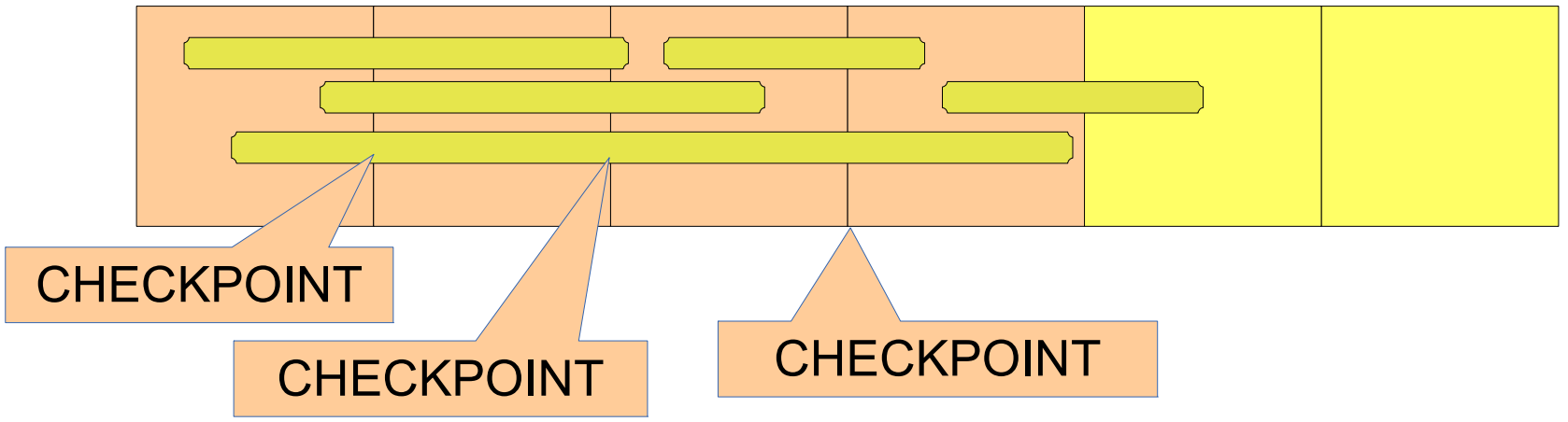
Comprendre le moteur
Moteur de stockage – le JT – recovery

- La récupération (recovery) est l'action d'utiliser le JT pour remettre la base de données dans un état transactionnellement cohérent, durant un restart ou une restauration de sauvegarde
- La portion du JT nécessaire au recovery est nommée la portion active du JT.



Comprendre le moteur
Moteur de stockage – le JT – Checkpoint

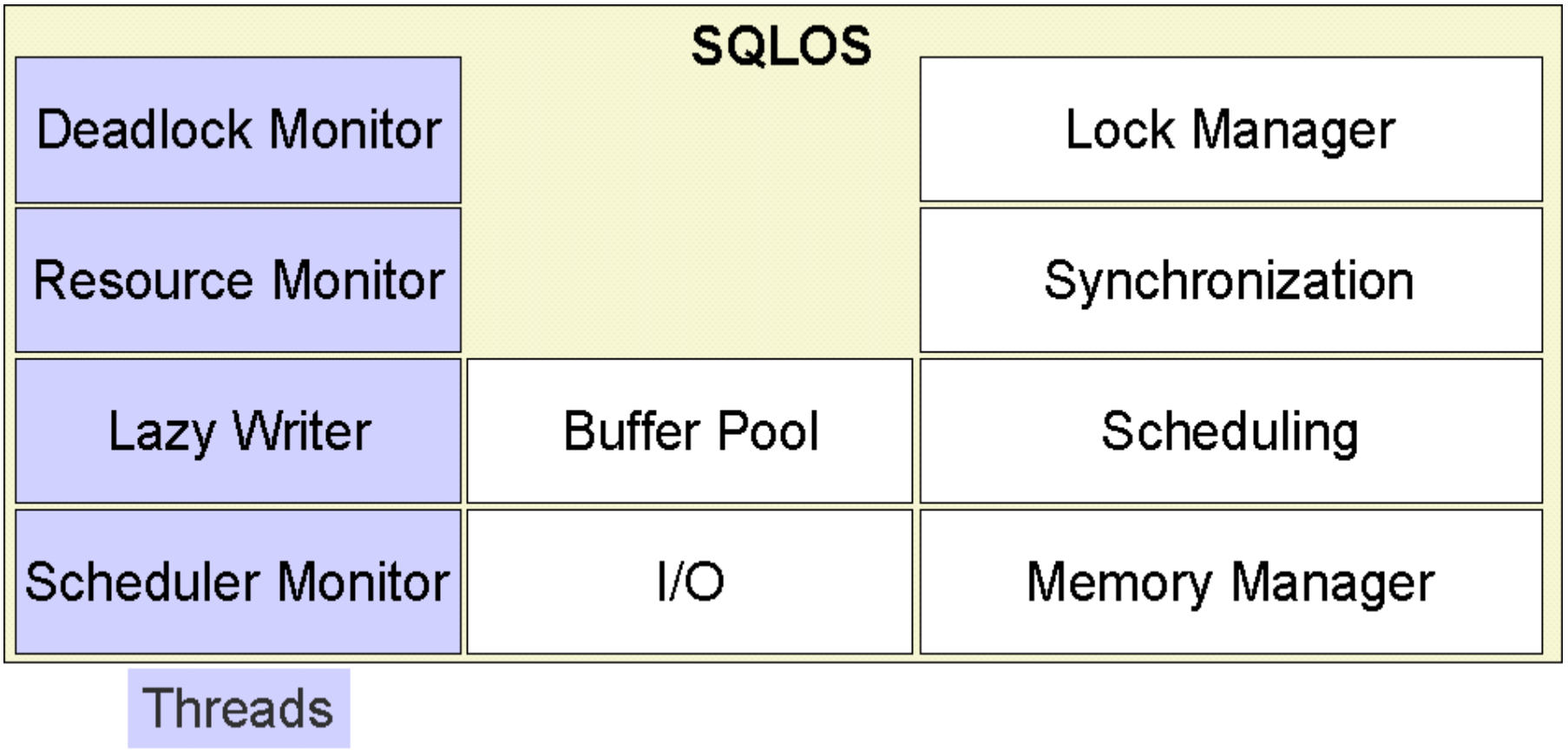
- Point de contrôle inscrit dans le journal de transactions
- Les sauvegardes vident le journal de transaction des transactions terminées avant le dernier checkpoint
- La portion vidée du journal devient disponible pour de nouvelles transactions
  - Cycle lorsque la fin du journal est atteinte



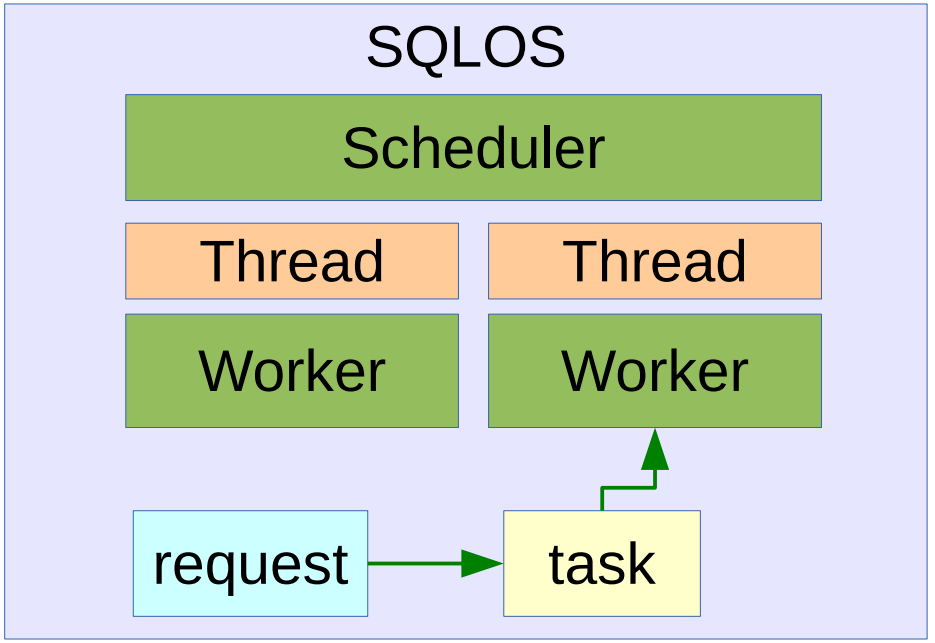
Comprendre le moteur SQLOS

- SQLOS est la couche d'accès aux périphériques.
- Situation optimale :
  - SQL Server est installé sur une machine dédiée
  - SQLOS récupère le temps processeur disponible, et la RAM allouée par Windows
  - Il distribue le temps processeur à l'aide de son ordonnanceur interne, coopératif
  - Il organise la RAM selon ses besoins, en piochant dans les pages du buffer, ou avec VirtualAlloc()

Comprendre le moteur SQLOS



Comprendre le moteur SQLOS – l'ordonnanceur (scheduler)



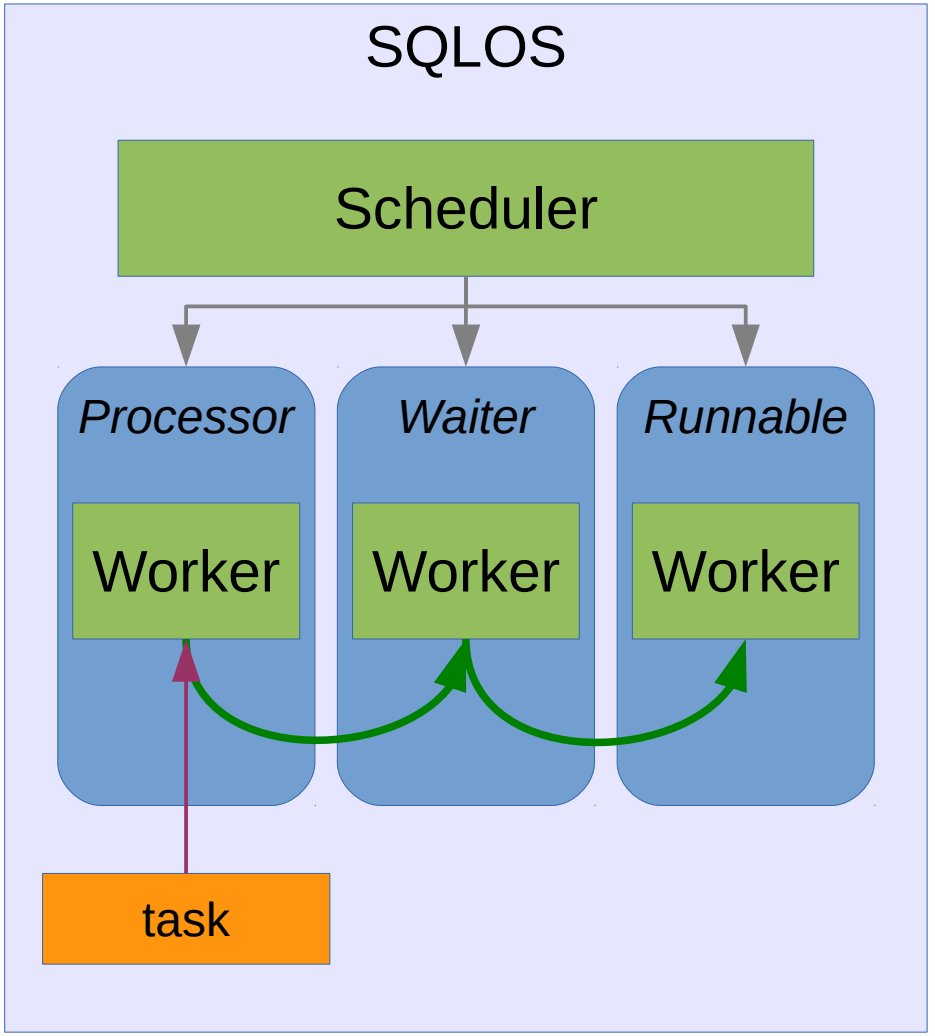
- Une requête SQL (request) est assignée à une tâche (task)
- La tâche est assignée à un worker
- Le scheduler exécute le worker sans interruption ou le le met dans une file d'attente

- Au démarrage, SQL Server crée un ordonnanceur (scheduler) par CPU
- Chaque ordonnanceur gère des workers
  - Le nombre de workers varie selon le nombre de sessions
  - Un seul worker peut être en exécution par ordonnanceur
  - SQLOS attribue une tâche à un worker
  - Un thread est la représentation physique du worker. Relation 1 – 1

Comprendre le moteur SQLOS – l'ordonnanceur (scheduler)

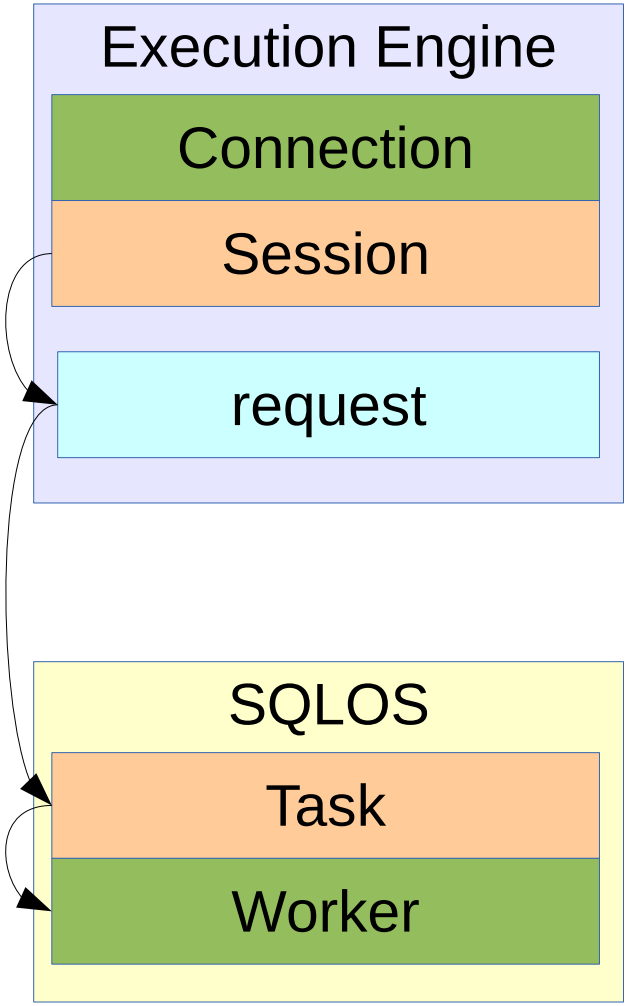
- L'ordonnanceur de Windows est préemptif
- L'ordonnanceur de SQLOS est non-préemptif
  - Il est coopératif
  - Il fonctionne en mode utilisateur (UMS)
- En édition entreprise, l'exécution est dépendante d'une gouvernance par le resource governor.
- Il gère des workers. Un worker en exécution est appelé un Worker thread = thread de Windows lié à un worker

Comprendre le moteur SQLOS – l'ordonnanceur (scheduler)



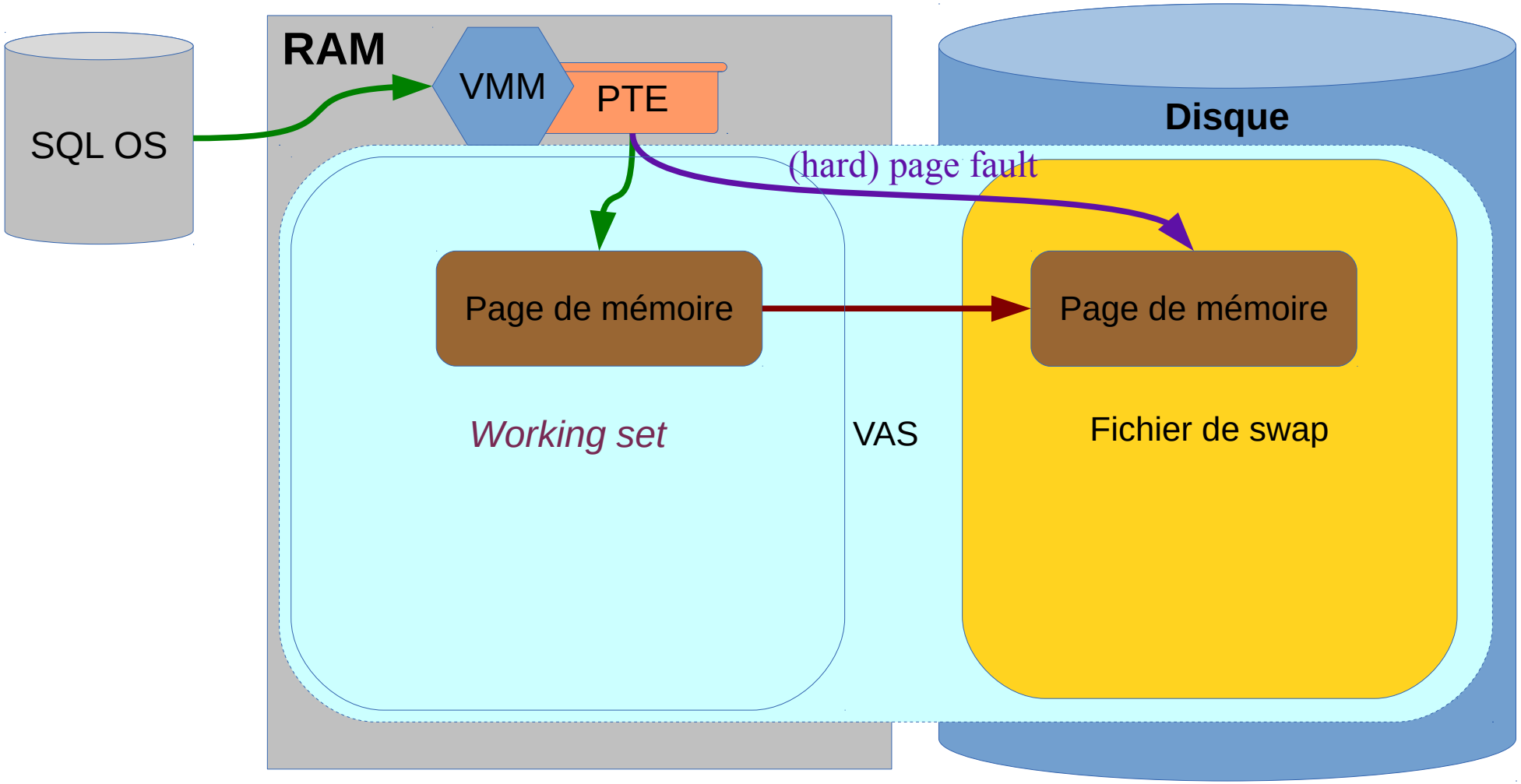
- Un ordonnanceur maintient trois composants importants :
  - Processor – l'exécuteur du worker thread
  - Waiter – liste des workers en attente sur une ressource
  - Runnable – liste des workers prêts à l'exécution
- La tâche d'un worker n'est pas interrompue par l'ordonnanceur, pour éviter les basculements de contexte, sauf en cas d'attente, nous verrons ça plus loin.

Comprendre le moteur
Le moteur relationnel – sessions et exécution



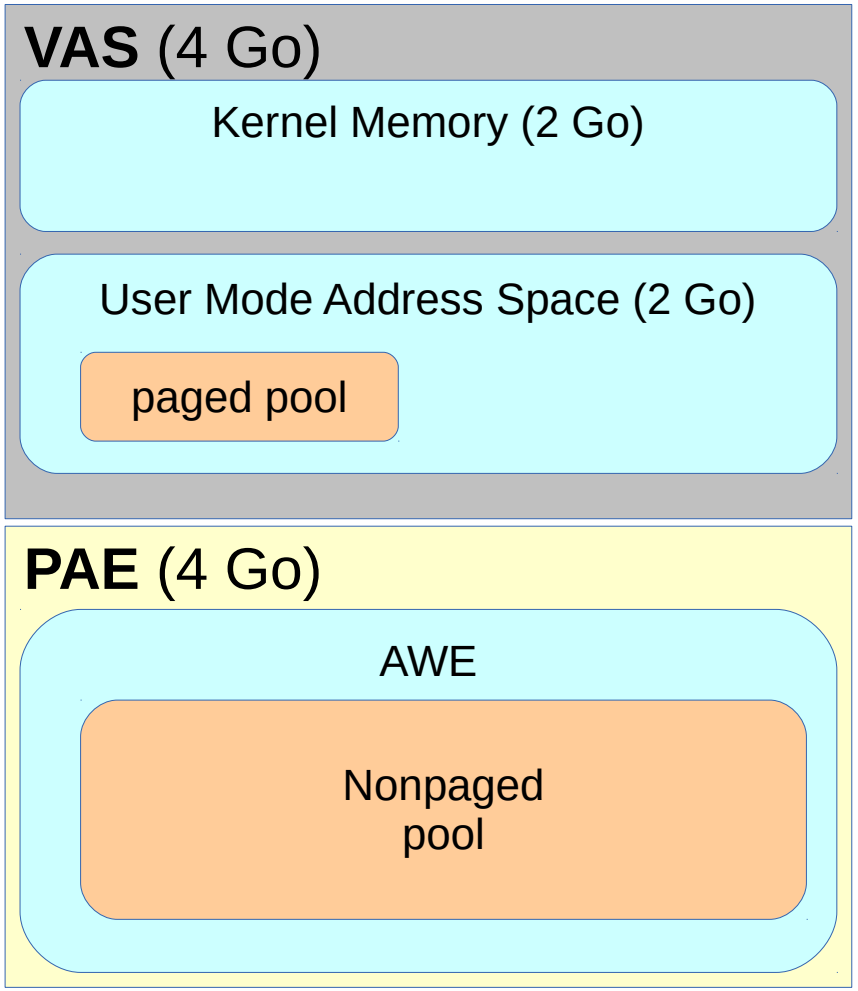
- Le connexion est le lien physique entre un client et le serveur
- La session est la représentation logique de la connexion, après succès de la phase de login
- Quand le client envoie une requête, elle est assignée à une tâche dans SQLOS, et la tâche est assignée à un worker

Comprendre le moteur SQLOS – la RAM sous Windows

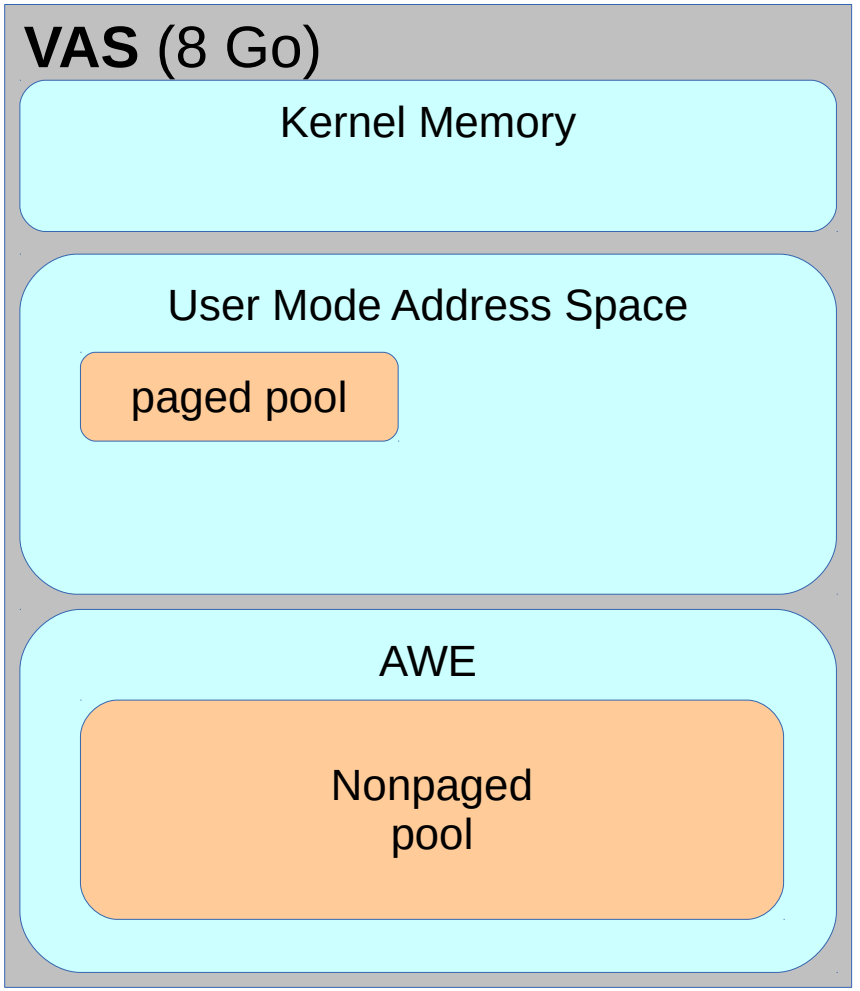


Comprendre le moteur SQLOS – la RAM sous Windows

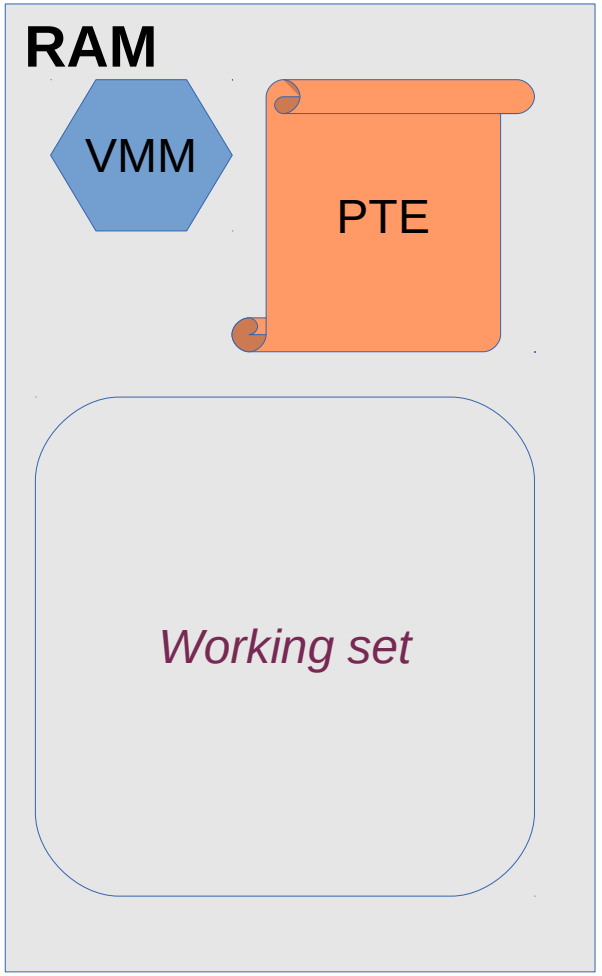
- 32 bit – 8 Go de RAM



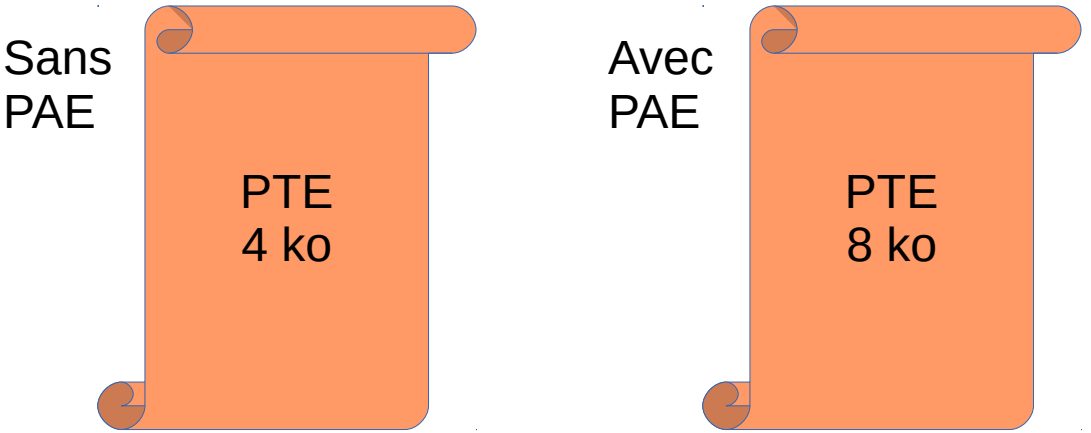
- 64 bit – 8 Go de RAM



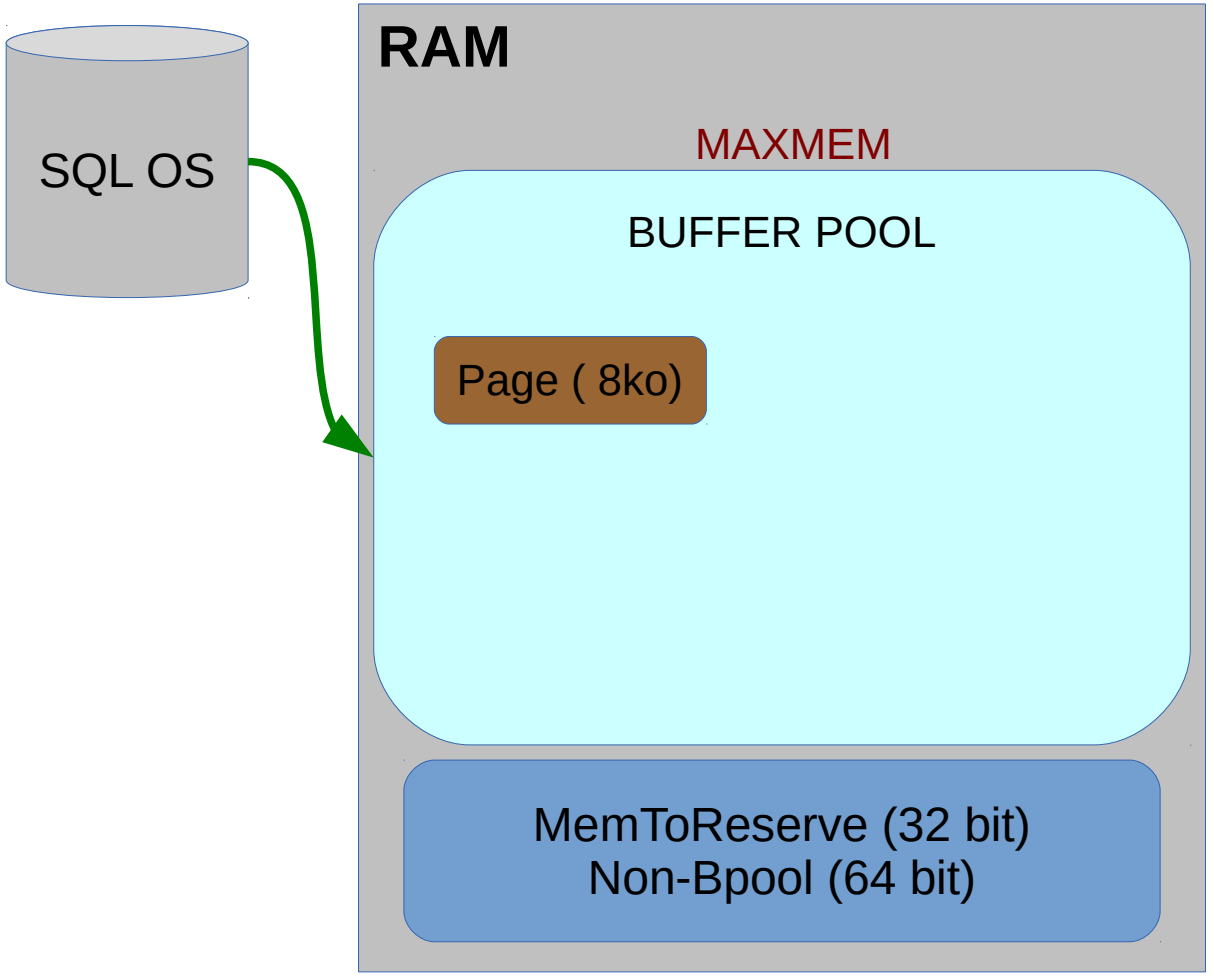
Comprendre le moteur
SQLOS – 32 bit – 3GB + PAE ou non ?



- 3GB est utile pour augmenter l'user-mode address space en 32 bit
- PAE est utile pour adresser la mémoire > 4 Go
- Mais attention aux PTE

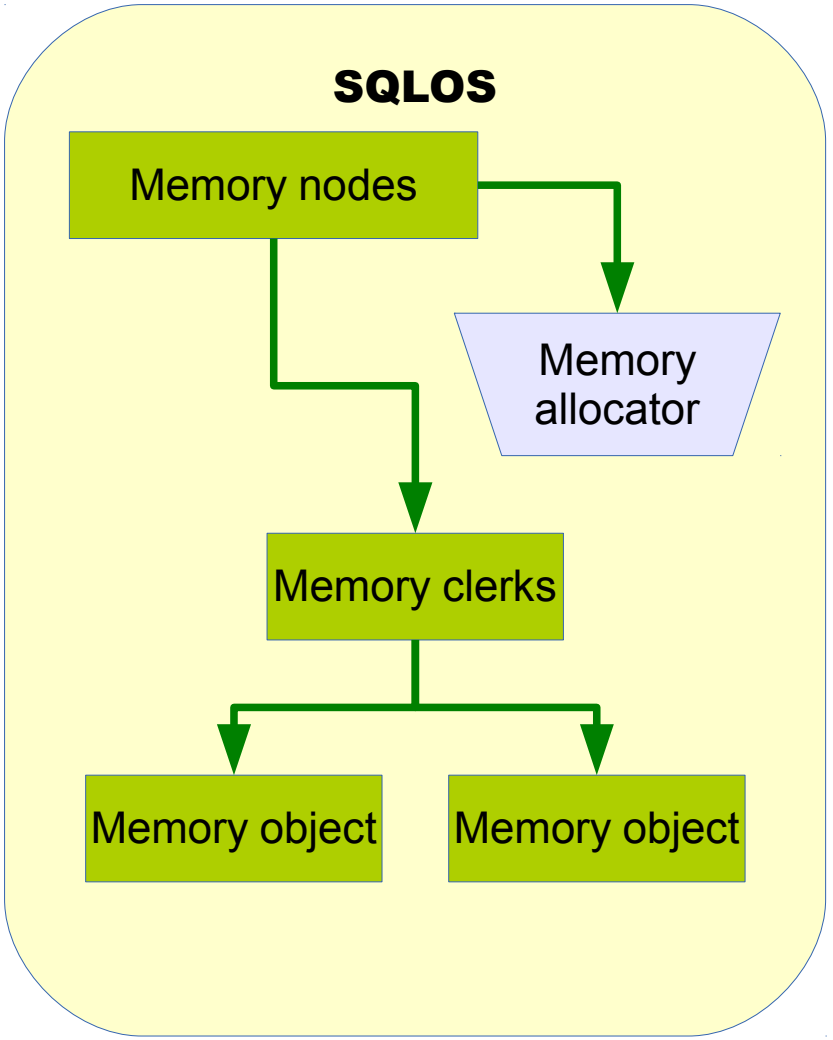


Comprendre le moteur
SQLOS – Allocation de la mémoire



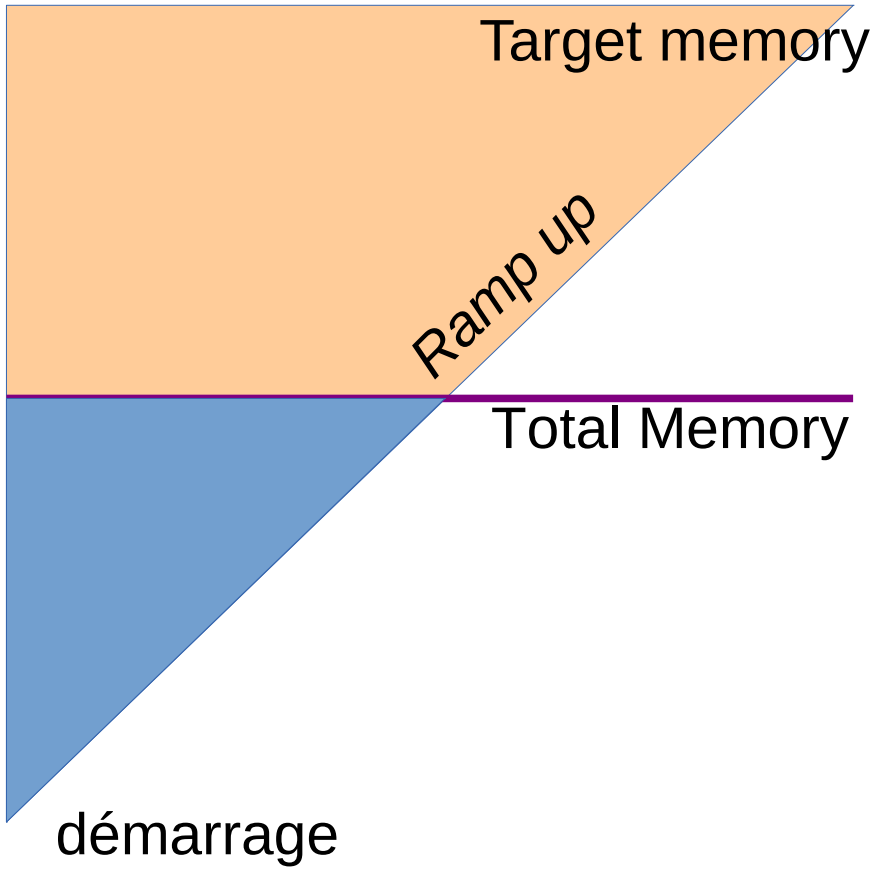
- La mémoire utilisée par SQLOS est divisée en deux parties :
  - Bpool
  - MTL ou NonBpool
- La partie NonBPool peut remplir la mémoire par fuite

Comprendre le moteur
SQLOS – Gestionnaire de mémoire



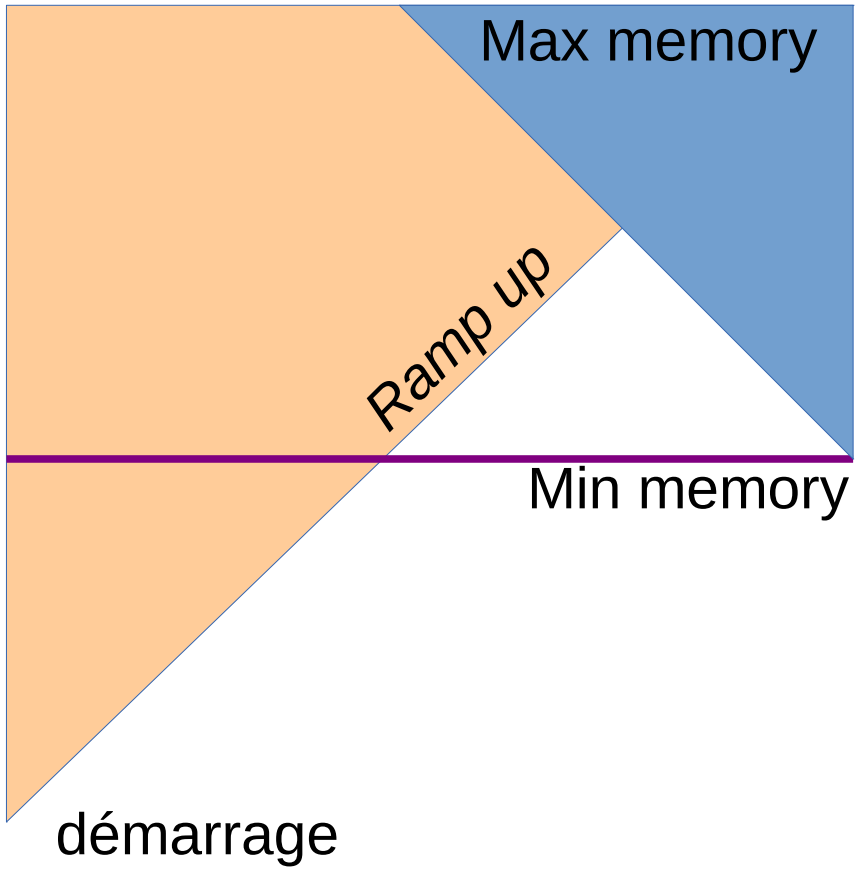
- La mémoire est séparée en Noeuds, un par nœud NUMA
  - Memory nodes
  - Il y a toujours au moins un node 0
  - Il y a un node en plus pour la DAC : 64
- Des routines allouent la mémoire à partir des nœuds, en utilisant des clerks, qui font des appels aux API Windows (VirtualAlloc(), AWE)

Comprendre le moteur
SQLOS – Occupation de la mémoire



- SQL Server n'occupe pas directement la mémoire
- Il occupe la mémoire alloué par Windows au fur et à mesure de ses besoins
- La mémoire actuellement occupée est indiquée par la mesure **Total Memory**
- La mémoire accordée par Windows est indiquée par la mesure **Target Memory**

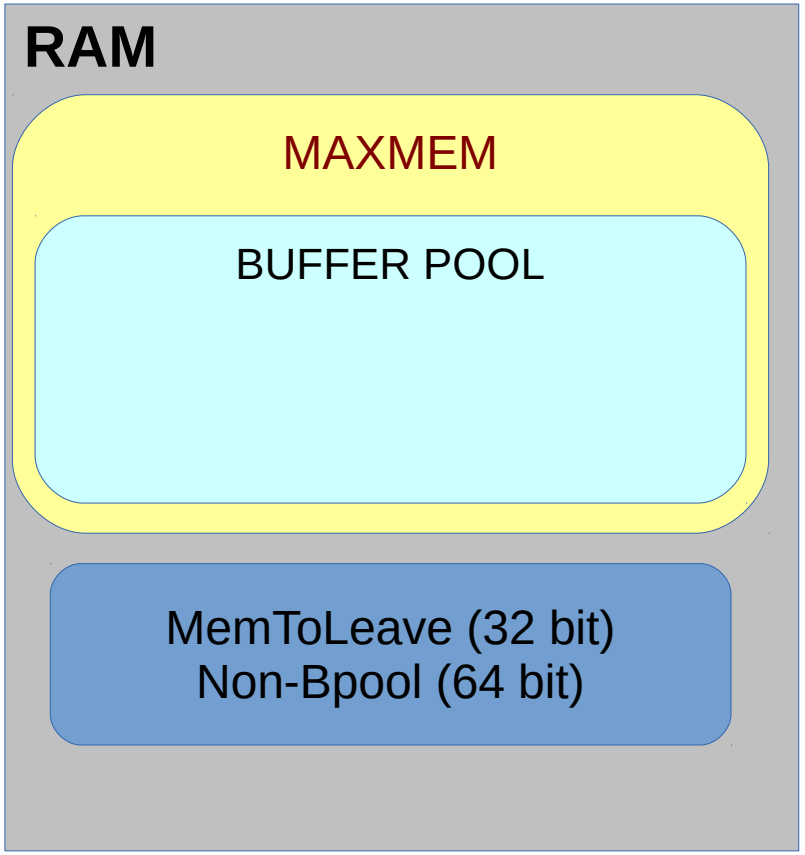
Comprendre le moteur
SQLOS – Min Server Memory et Max Server Memory



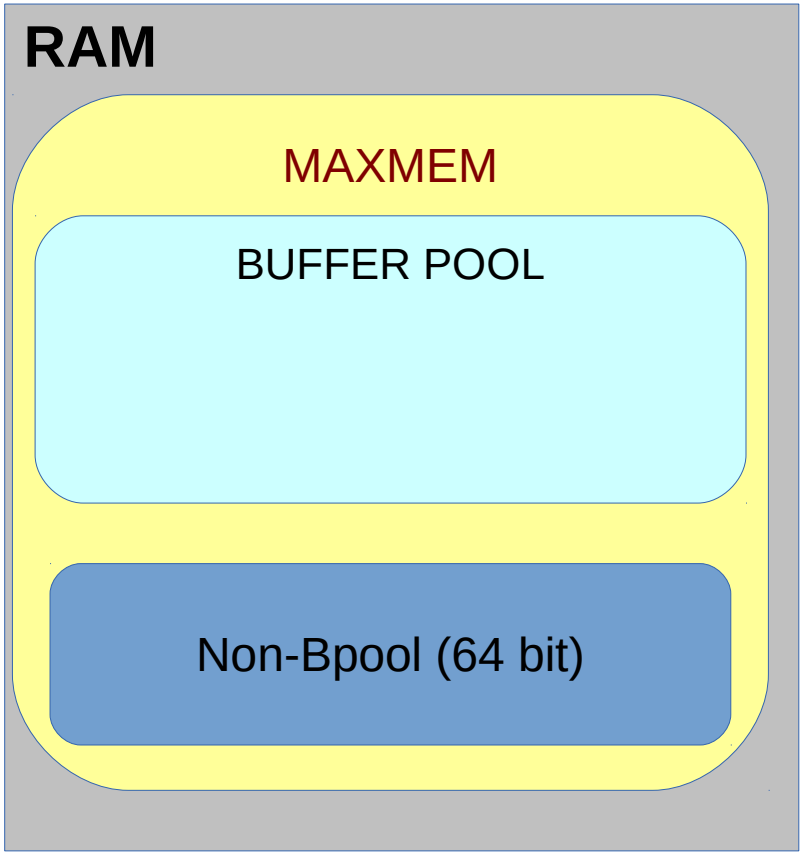
- SQL Server n'occupe pas directement la mémoire
- Le paramètre Max Server Memory indique la limite haute à ne pas dépasser
- Le paramètre Min Server Memory indique la limite basse en deçà de laquelle SQL Server ne rendra pas de mémoire à Windows en cas de pression sur la mémoire

Comprendre le moteur SQLOS – Max Server Memory

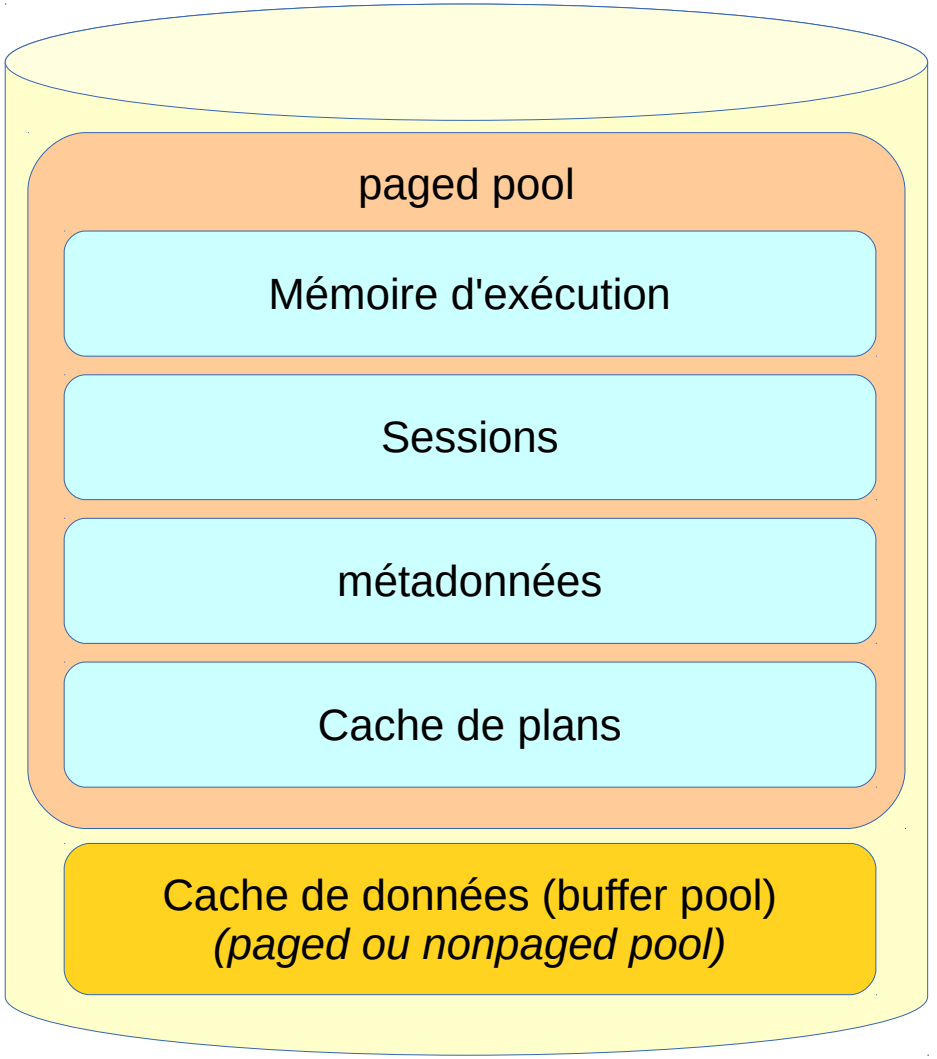
- < SQL SERVER 2012



- >= SQL Server 2012

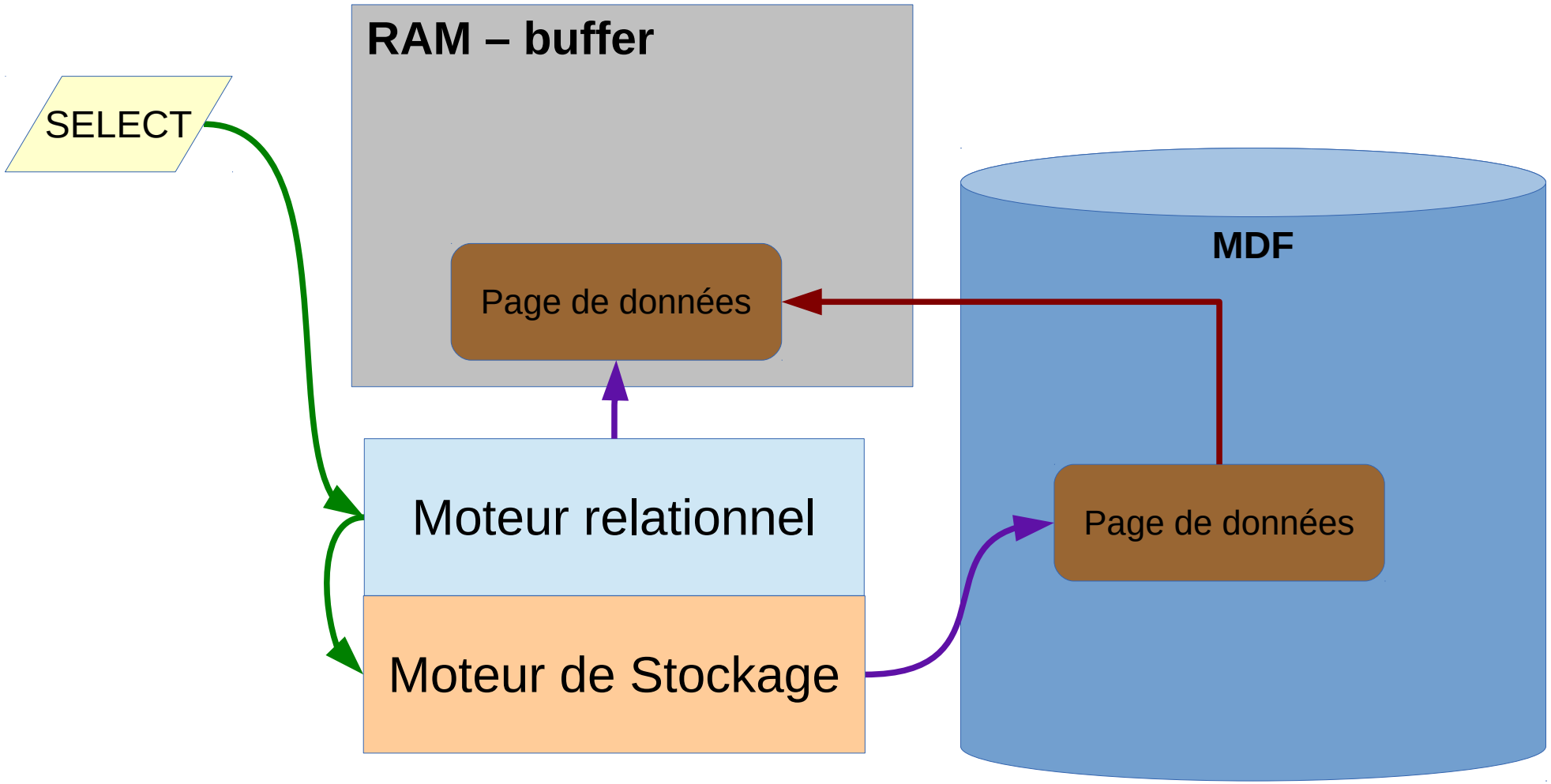


Comprendre le moteur SQLOS – Utilisation de la RAM



- La mémoire Bpool est divisé en :
  - Mémoire d'exécution, de sessions, de verrous, …
  - Cache de plans
  - Cache de données (buffer pool)
- Seul le buffer pool peut être placé dans le nonpaged pool.
  - En utilisant le mécanisme AWE, même en 64 bit

Comprendre le moteur SQLOS – le buffer pool



Comprendre le moteur SQLOS – Configuration de la RAM

- 32 bit – 8 Go de RAM

VAS (4 Go)

/3GB sur <= Win 2003

/UserVA sur <= Win 2003

bcdedit /set increaseuserva >= Win 2008

PAE (4 Go)

/PAE sur <= Win 2003

AWE

AWE enabled = 1

- 64 bit – 8 Go de RAM

VAS (8 Go)

AWE

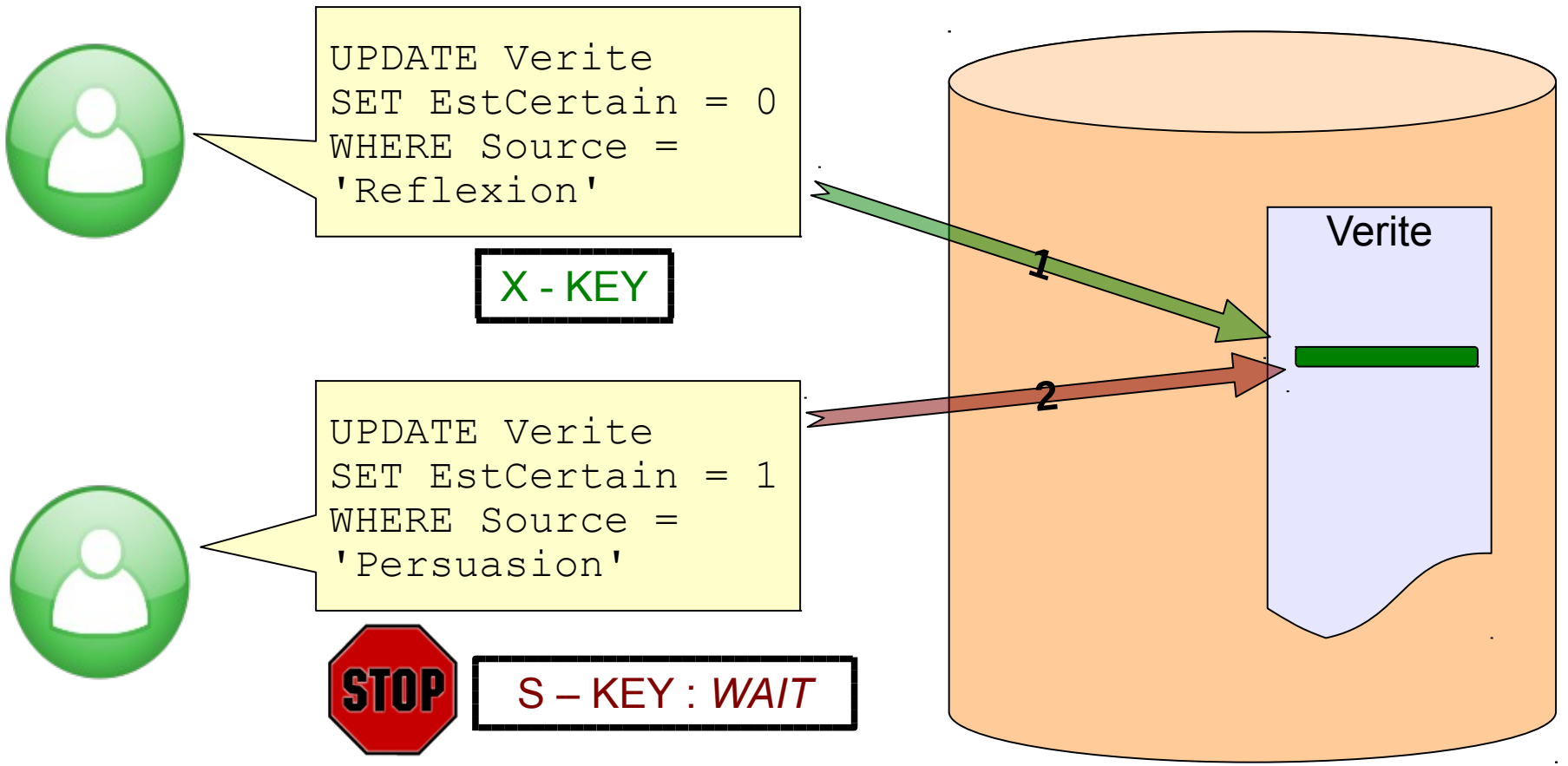
Nonpaged pool

Lock page in memory
Sur le compte de service

Problématiques classiques
Transactions – rappel sur la transaction

- Une transaction est dite ACID
  - Atomique : tout est validé ou rien
    - C'est le boulot du journal de transactions
  - Cohérente : la base respecte des règles de cohérence
  - **Isolée** : on ne peut accéder aux données lorsqu'elles sont dans un état transitoire (incohérent)
    - SQL Server procède à un verrouillage pessimiste et protège les ressources en cours de modification
  - Durable : une fois validée, c'est pour la vie
    - C'est le boulot du journal de transactions

Problématiques classiques
Verrouillage



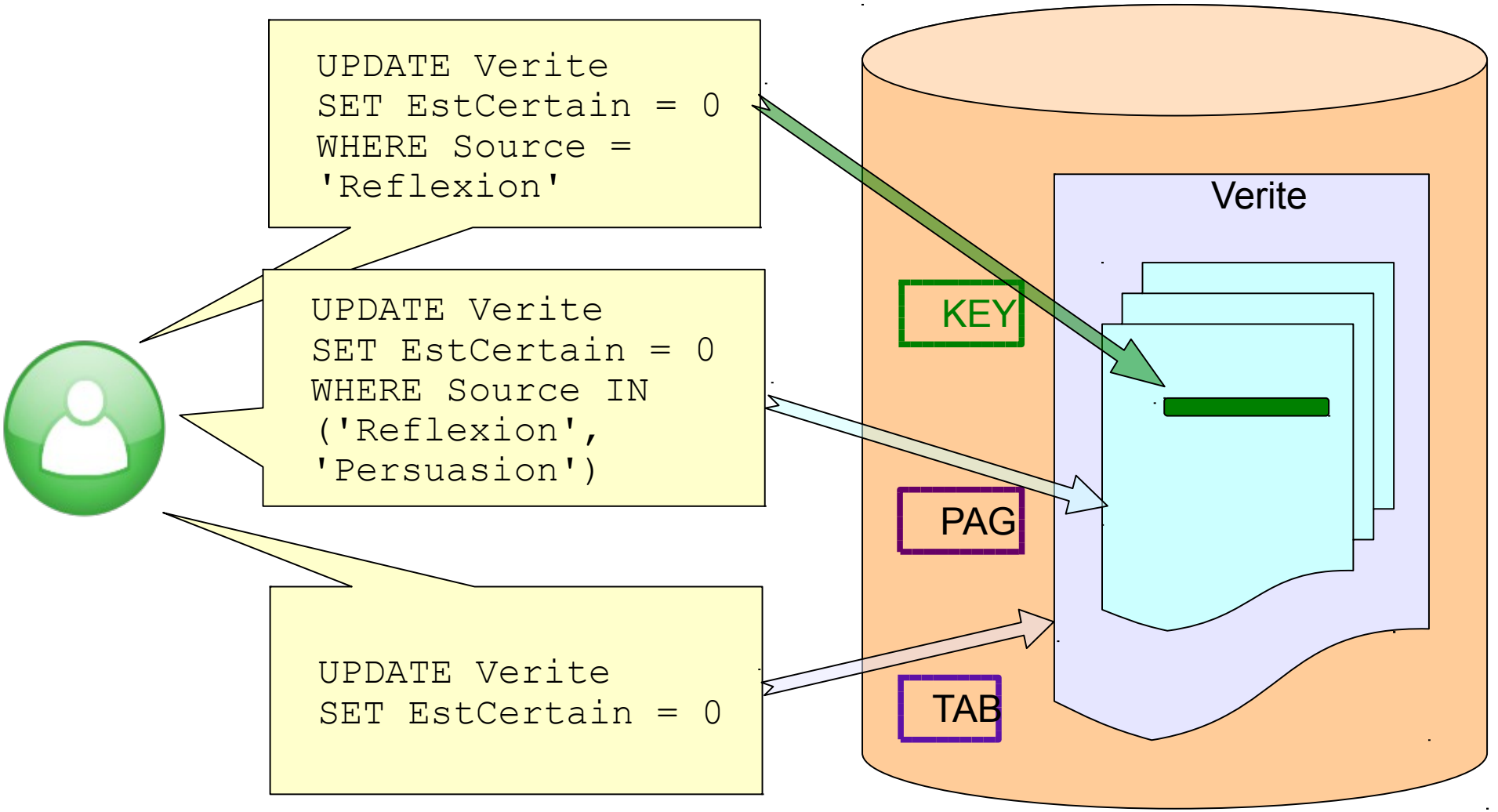
Problématiques classiques
Verrouillage – types de verrous

Type	Description
RID	Sur une ligne, heap
KEY	Sur une ligne, Index ordonné
PAG	Sur une page
EXT	Sur une extension
TAB	Sur une table
DB	Sur des métadonnées (collections XML)
FIL	Sur un fichier
MD	Sur des métadonnées (collections XML)
AU	Sur une unité d'allocation

Problématiques classiques
Verrouillage – modes de verrouillage

Mode	Description
Sch-S	Schéma (structure), partagé
Sch-M	Schéma (structure), modification : exclusif
S	Partagé, lecture
U	Modification éventuelle, prévention de deadlock
X	Exclusif, modification
IS	Intent Shared lock : interdit le verrou exclusif à partir du niveau supérieur de verrouillage
IU	Intent Modification lock : interdit le verrou exclusif à partir du niveau supérieur de verrouillage
IX	Intent Exclusive lock : interdit le verrou exclusif à partir du niveau supérieur de verrouillage

Problématiques classiques
Verrouillage – escalade de verrous



Problématiques classiques
Verrouillage – compatibilité des verrous

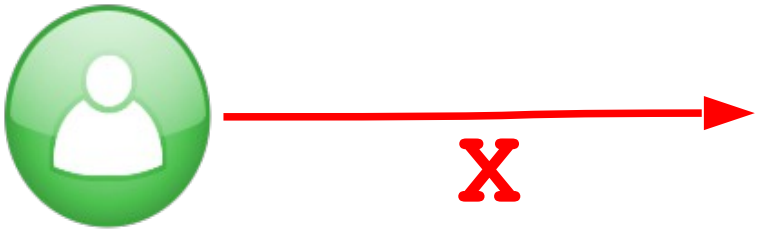
Tout n'est pas permis !

Ce mode est il possible ?

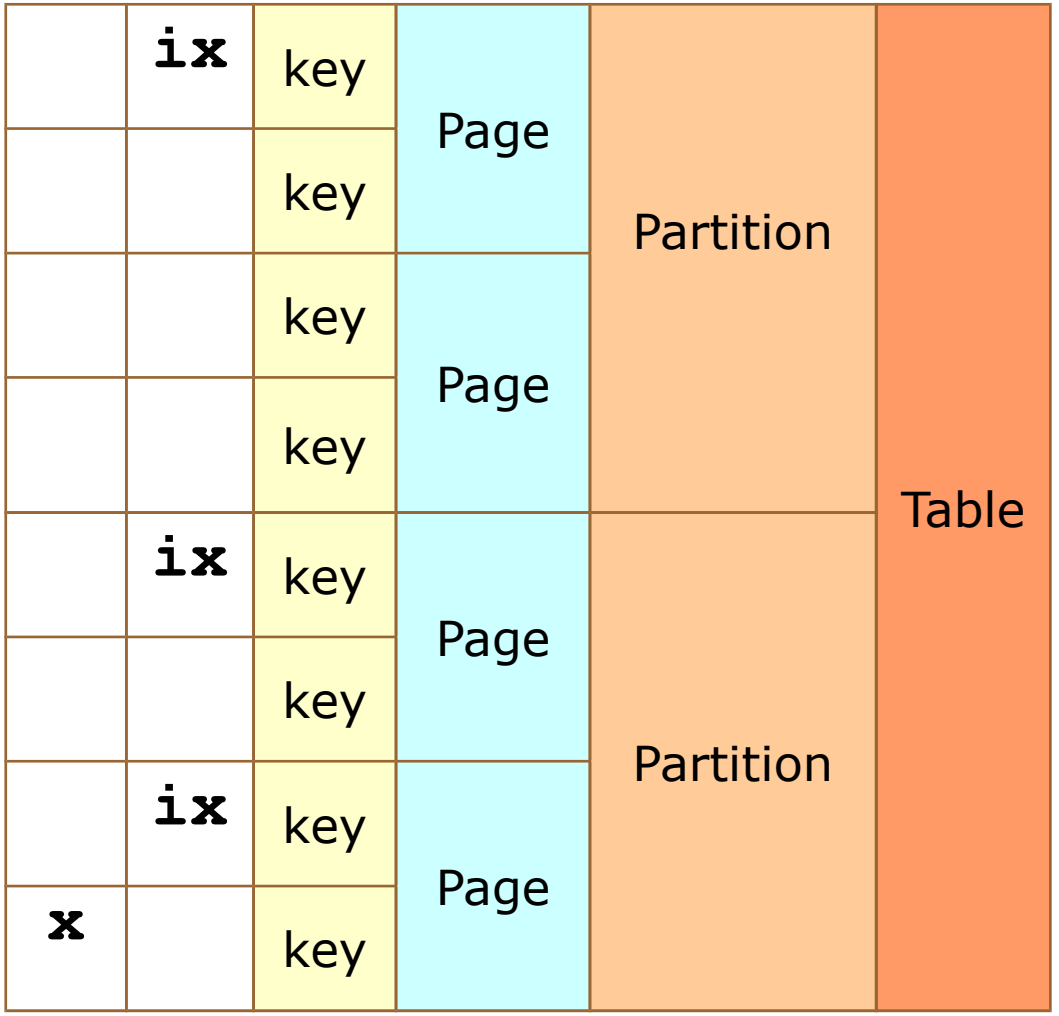
Mode posé

	S	IS	U	X	IX
S	Oui	Oui	Oui	Non	Non
U	Oui	Oui	Non	Non	Non
X	Non	Non	Non	Non	Non
IS	Oui	Oui	Oui	Non	Oui
IX	Non	Oui	Non	Non	Oui

Problématiques classiques
Verrouillage – les verrous d'intention (I_)

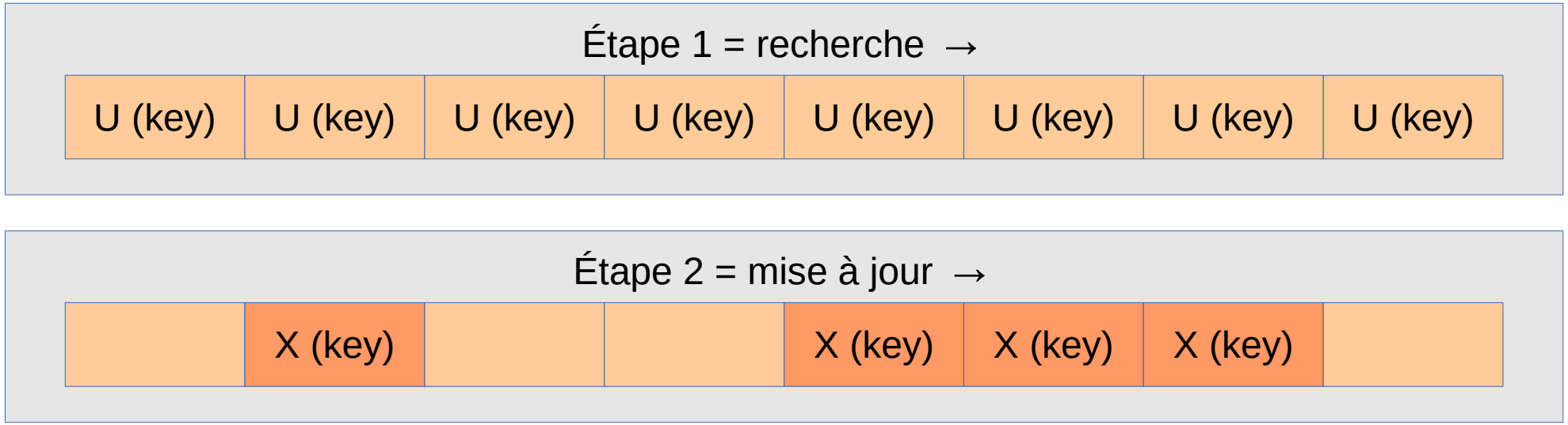


Le verrou d'intention permet de protéger les ressources contenantes d'un verrou, pour éviter un conflit dû aux différences de niveau



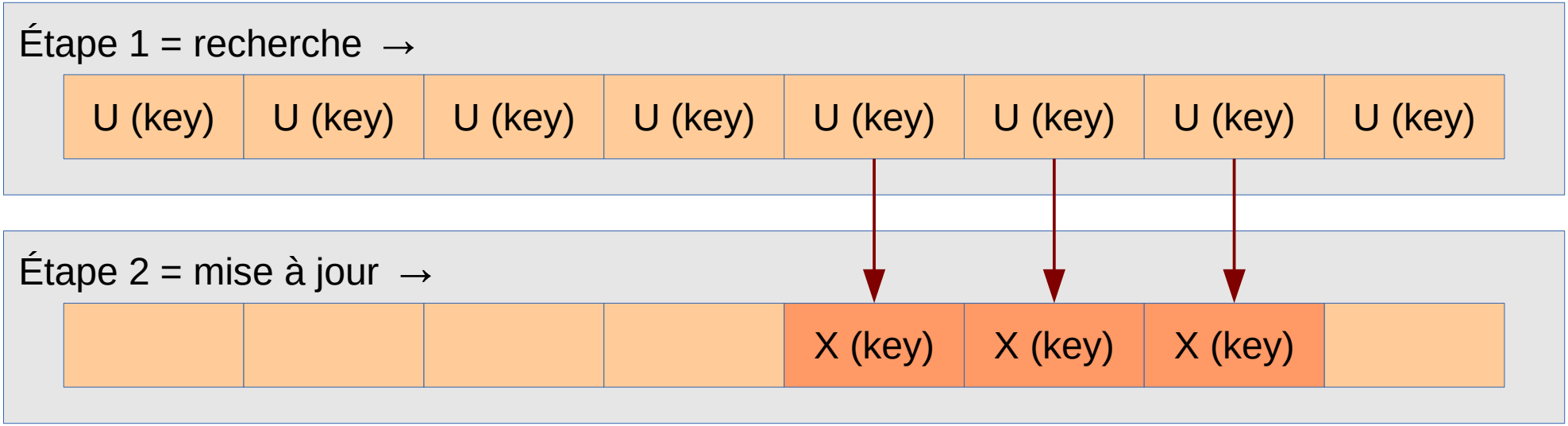
Problématiques classiques
Verrouillage – les verrous d'update (U)

```
UPDATE Contact.Contact
SET Nom = UPPER(LEFT(Nom, 1)) + SUBSTRING(Nom, 2, 50)
WHERE LEFT(Nom, 1) = LOWER(LEFT(Nom, 1)) COLLATE French_CS_AS;
```



- Le verrou U est un verrou de lecture dans une requête d'UPDATE, posé durant la phase de recherche.
- Compatible avec un S, incompatible avec un U

Problématiques classiques
Verrouillage – Conversion des verrous



- Selon les besoins de la requête, un verrou peut être converti dans un autre mode
  - En cas de verrou U : U en X
  - En cas d'escalade : IS en S, IX en X, …
  - …

Problématiques classiques
Verrouillage – Comment voir/surveiller les verrous ?

Avec *sp_lock*

sp_lock [[@spid1 =] *IdSession1*] [, [@spid2 =] *IdSession2*] [;]

```
exec sp_lock 55
```

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	55	14	0	0	DB		S	GRANT
2	55	14	37575172	0	TAB		X	GRANT

Avec la DMV *sys.dm_tran_locks*

Cette DMV renvoie de nombreuses informations utiles sur les verrous en cours

On la combinera efficacement avec la vue sys.partitions pour connaître les objets verrouillés

Problématiques classiques
Verrouillage – exemple de requête

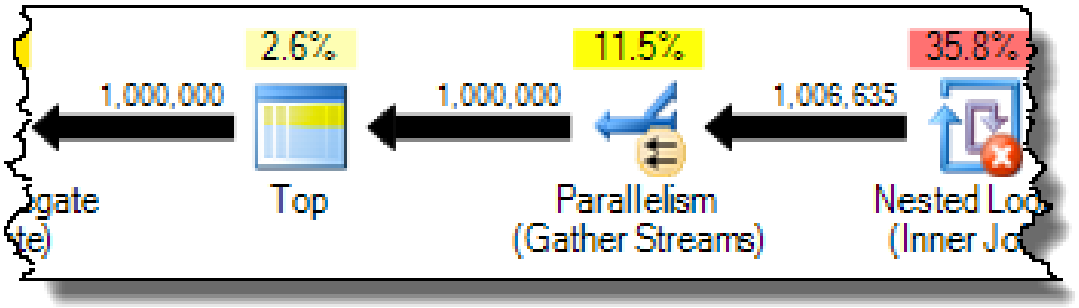
- Verrous maintenus sur un objet

```
SELECT
    l.resource_type,
    l.resource_subtype,
    l.resource_lock_partition,
    l.request_mode,
    l.request_type,
    l.request_session_id,
    t.text
FROM sys.dm_tran_locks l
JOIN sys.dm_exec_requests r
    ON l.request_request_id = r.request_id
AND l.request_session_id = r.session_id
CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) t
WHERE resource_database_id = DB_ID('MaBase')
AND resource_associated_entity_id = OBJECT_ID('MaTable');
```

Problématiques classiques
Parallélisme

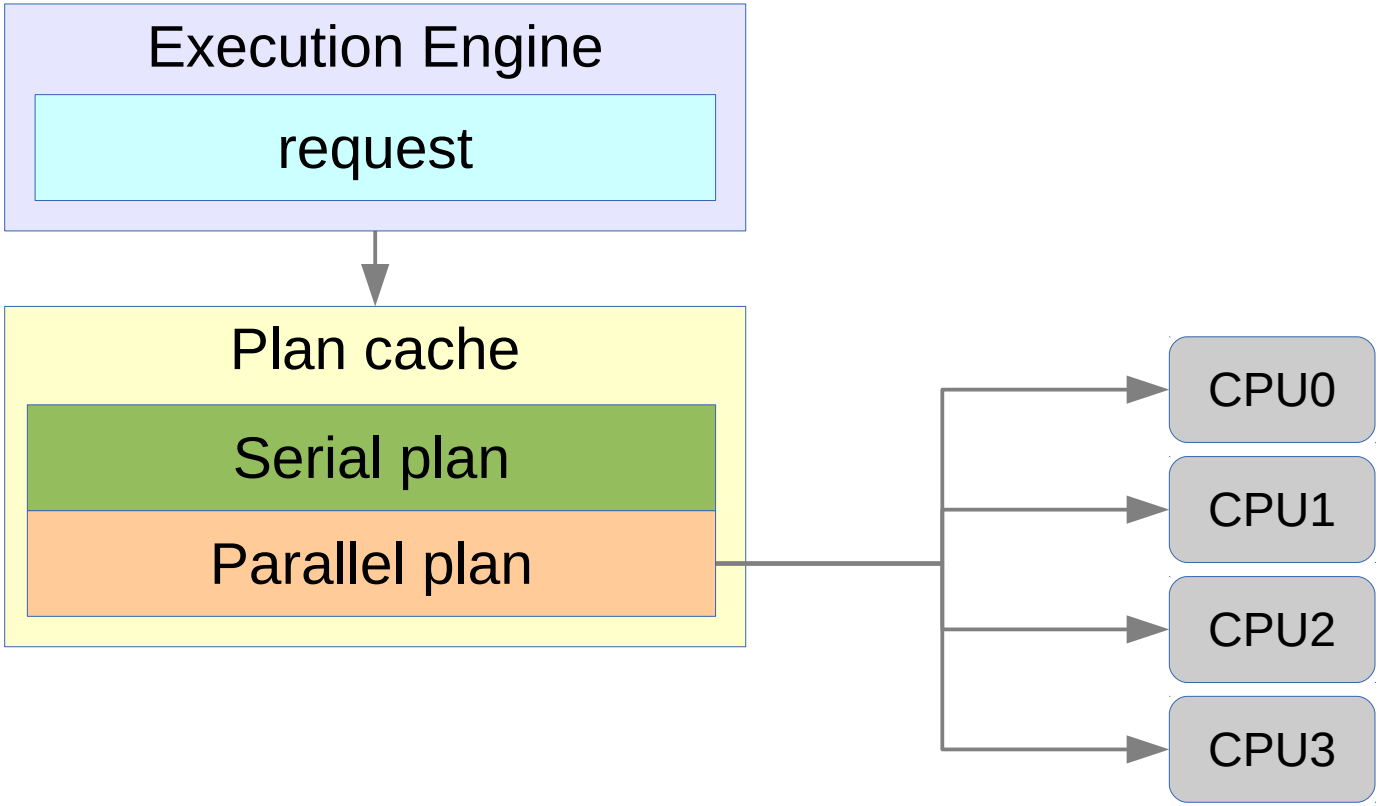
- Deux plans peuvent être générés pour une requête :
  - Exécution parallélisable (sans garantie)
  - Exécution sérialisée

Parallelism	
Attente de la requête	-1
Degré maximum de parallélisme	1
Seuil de coût pour le parallélisme	5
Verrous	0

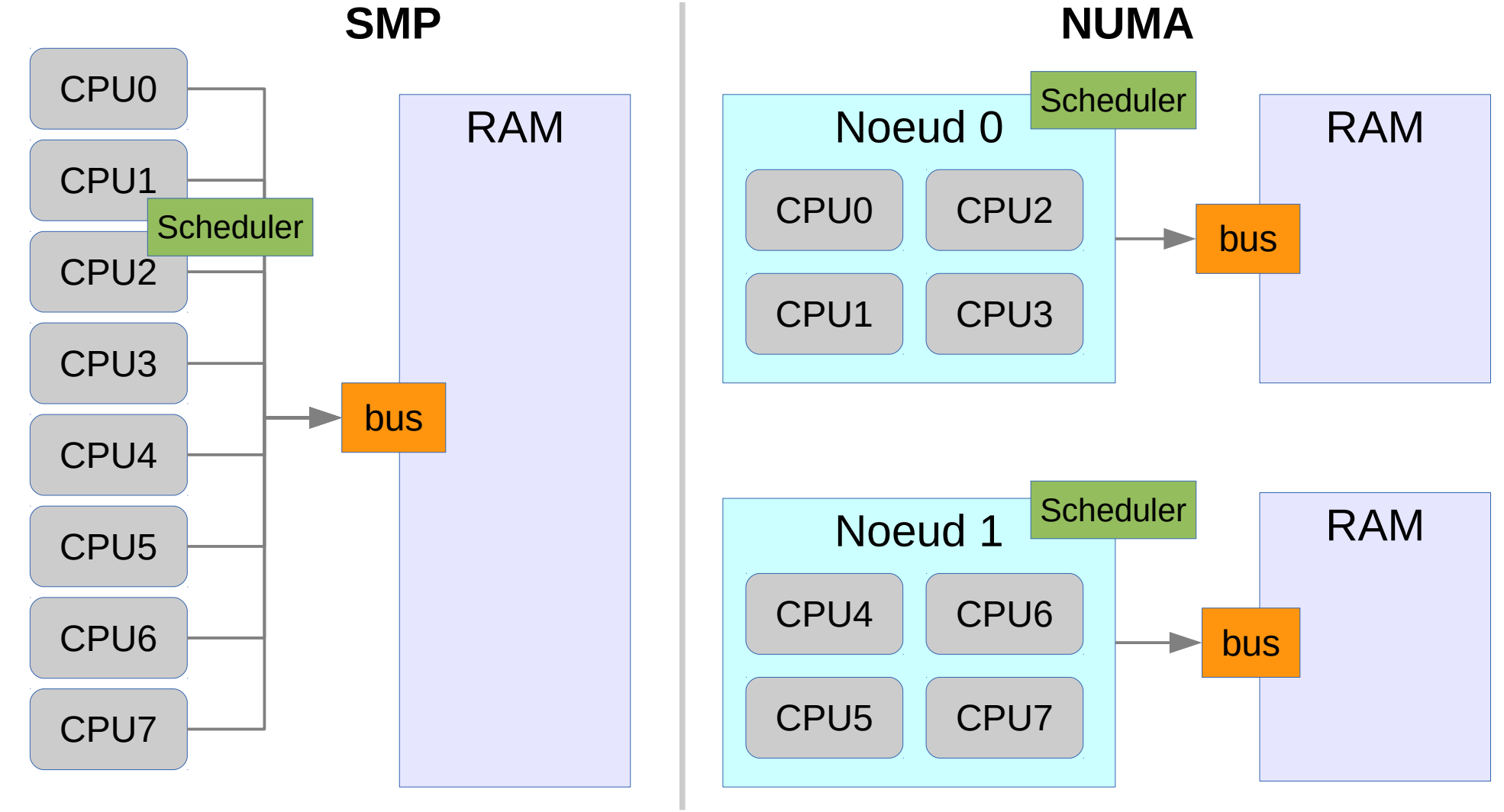


- Le moteur choisit de paralléliser durant l'exécution si :
  - La durée estimée dépasse le seuil de coût
  - L'activité des processeurs est faible
- Le parallélisme s'effectue sur le nombre de processeurs défini dans le degré maximum (maxdop)

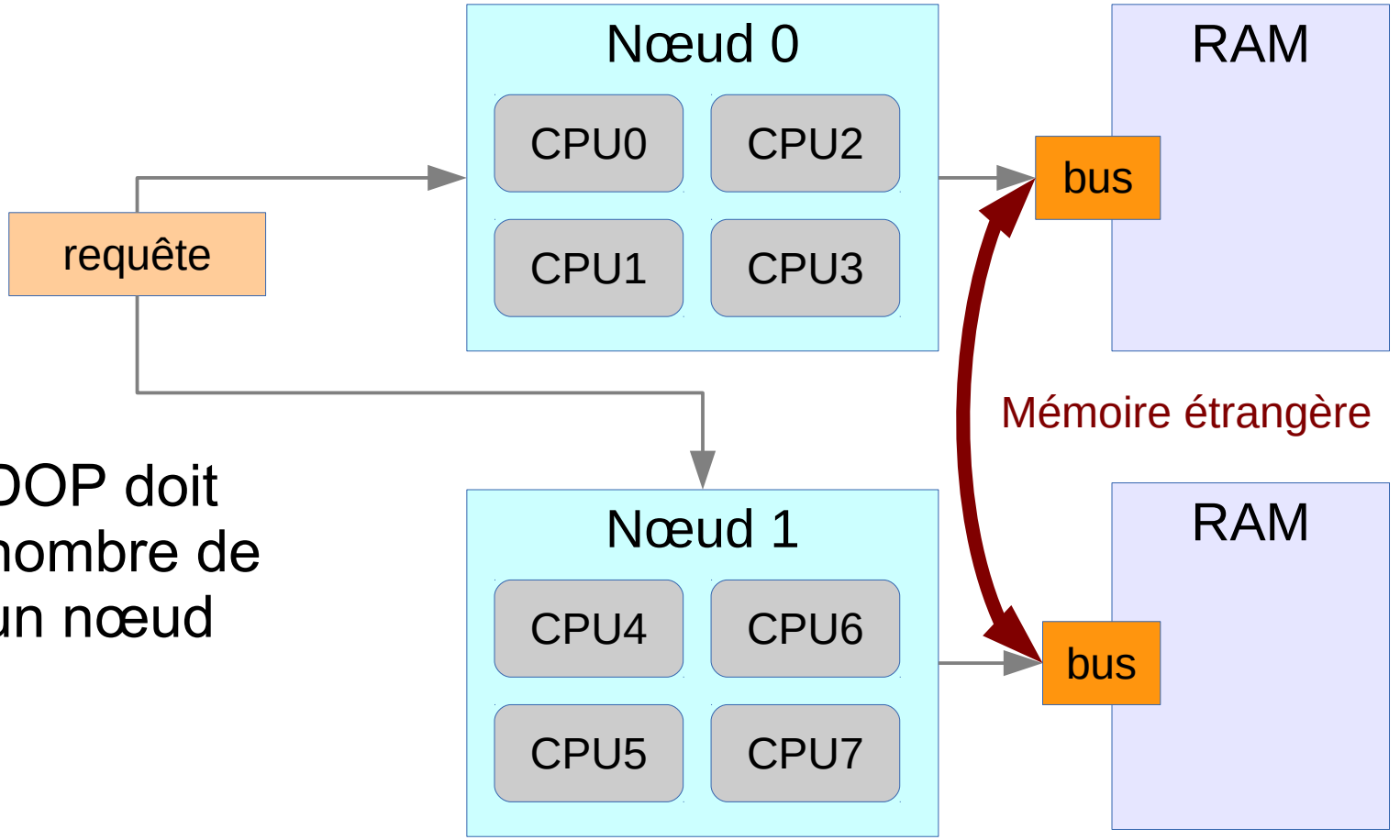
Problématiques classiques
Parallélisme



Problématiques classiques
Parallélisme – architecture NUMA



Problématiques classiques
Parallélisme – NUMA et parallélisme



- Votre MAXDOP doit être <= au nombre de CPU dans un nœud NUMA

Merci et bon courage !