haute école arc gestion
neuchâtel berne jura neuchâtel delémont

633-2.2
Dialogue avec la base de données

2. Transactions et verrous

Cedric.Baudet@he-arc.ch

Références

- Filière
  - Bachelor of Science en informatique de gestion
- Domaine
  - Technologies informatiques
- Module
  - 633 – Architectures distribuées
- Unité d'enseignement
  - 633-2.1 – Applications réseaux et services applicatifs
  - 633-2.2 – Dialogue avec la base de données

Objectifs

- Objectifs du module
  - comprendre et mettre en œuvre les services applicatifs fondamentaux;
  - développer, déployer et configurer des composants métiers sur un serveur d'applications.

- Objectifs de l'unité d'enseignement « Dialogue avec la base de données »
  - comprendre les différences entre l'architecture "serveur de fichiers" et "client/serveur";
  - maîtriser le principe du dialogue à sessions;
  - comprendre la notion de concurrence;
  - savoir manipuler des données à l'aide d'un langage hôte.

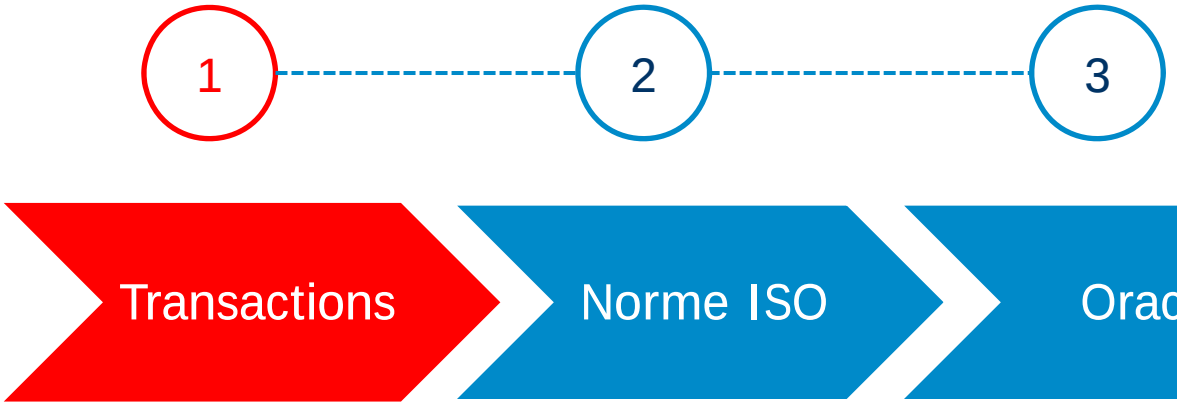
Découpage de l'unité d'enseignement

1. Architecture client/serveur

2. Transactions et verrous

3. Manipulation des données avec un langage hôte

Plan de la présentation



Transaction = Concept fondamental

Toute application manipulant un grand ensemble de données, et exploitées dans un environnement multi-utilisateurs se doit d'être transactionnelle.

- Sans système transactionnel, la base de données risque d'être **incohérente**
  - Rappel des missions du SII !!
    - Disponibilité de l'information
    - Pérennité de l'information
    - Intégrité de l'information
    - Confidentialité de l'information
    - Traçabilité des opérations

Définition d'une transaction

- Une transaction doit vérifier les propriétés
  - (A) Atomicité
  - (C) Consistance
  - (I) Isolation
  - (D) Durabilité

- Moyen mnémotechnique → ACID

ACID - Atomicité

- L'ensemble des opérations d'une transaction apparaît comme une seule opération atomique
- Soit toutes les opérations sont validées ou toutes annulées (tout ou rien)
- exemple : débit/crédit

1

Transactions

ACID Consistance (cohérence)

- L'exécution de la transaction fait passer la base de données d'un état consistant à un autre état consistant
  - ceci en accord avec l'ensemble de la base de données et de ses contraintes.

- Un état incohérent ne doit pas être possible !

1

Transactions

ACID Isolation

- Chaque transaction est indépendante des autres transactions concurrentes.
  - Semblable à une exécution en série des transactions. On parle de « sérialisation » des transactions.

- Les concurrences sont parfaitement contrôlées

1

Transactions

ACID Durabilité

- C'est la persistance des mises à jour d'une transaction validée.

- Les effets d'une transaction validée sont durables et permanents, quelques soient les problèmes logiciels ou matériels, notamment après la fin de la transaction.

Principales utilisations des transactions

- Traitement des opérations sémantiquement liées
  - Doit garantir le "tout ou rien"
    - Exemple : débit crédit,
    - Soit toutes les opérations sont validées soit annulées.

- Gestion des concurrences
  - Acquisition de verrous sur les enregistrements traités, empêchant une utilisation malencontreuse des données.

- Reprise sur pannes
  - Utilisation du système transactionnel pour la reconstitution d'un état cohérent de la base au redémarrage d'un système après une panne, quel que soit le type de panne.

Exemple de transaction comptable, état initial

Ecritures comptables					
Num	**Libellé**	**Cpt débit**	**Cpt crédit**	**Montant**	**Ecriture passée**
12345	**Bla bla**	**1010**	**2302**	**250.-**	**0**
….			**…**	**…**	

Compte 1010			
ecriture	**libelle**	**mouvement**	**Solde**
….	**….**	**…**	**…**
12340	**Bla bla**		**1000.-**

Compte 2302			
ecriture	**libelle**	**mouvement**	**Solde**
	….	**…**	**….**
2340	**Bla bla**		**600.-**

Exemple de transaction comptable, 3 écritures

3. Mise à jour du grand livre

Ecritures comptables					
Num	**Libellé**	**Cpt débit**	**Cpt crédit**	**Montant**	**Ecriture passée**
12345	***Bla bla***	***1010***	***2302***	***250.-***	***1***
….			**…**	**…**	

1. écriture dans compte débité

2. écriture dans compte crédité

Compte 1010			
ecriture	**libelle**	**mouvement**	**Solde**
….	**….**	**…**	**…**
12340	**Bla bla**		**1000.-**

Compte 2302			
ecriture	**libelle**	**mouvement**	**Solde**
	….	**…**	**….**
2340	**Bla bla**		**600.-**

Exemple de transaction comptable, 3 écritures

3. Mise à jour du grand livre

Ecritures comptables					
Num	**Libellé**	**Cpt débit**	**Cpt crédit**	**Montant**	**Ecriture passée**
12345	***Bla bla***	***1010***	***2302***	***250.-***	***1***
….			**…**	**…**	

1. écriture dans compte débité

2. écriture dans compte crédité

Compte 1010			
ecriture	**libelle**	**mouvement**	**Solde**
….	**….**	**…**	**…**
12340	**Bla bla**		**1000.-**
12345	**Bla bla**	**-250.-**	**750.-**

Compte 2302			
ecriture	**libelle**	**mouvement**	**Solde**
	….	**…**	**….**
2340	**Bla bla**		**600.-**
12345	**Bla bla**	**250.-**	**850.-**

Exemple de transaction comptable, avec panne

3. Mise à jour du grand livre

Ecritures comptables					
Num	**Libellé**	**Cpt débit**	**Cpt crédit**	**Montant**	**Ecriture passée**
12345	***Bla bla***	***1010***	***2302***	***250.-***	***1***
….			**…**	**…**	

1. écriture dans compte débité

BING

2. écriture dans compte crédité

Compte 1010			
ecriture	**libelle**	**mouvement**	**Solde**
….	**….**	**…**	**…**
12340	**Bla bla**		**1000.-**
12345	**Bla bla**	**-250.-**	**750.-**

Compte 2302			
ecriture	**libelle**	**mouvement**	**Solde**
	….	**…**	**….**
2340	**Bla bla**		**600.-**

Instructions de la transaction comptable

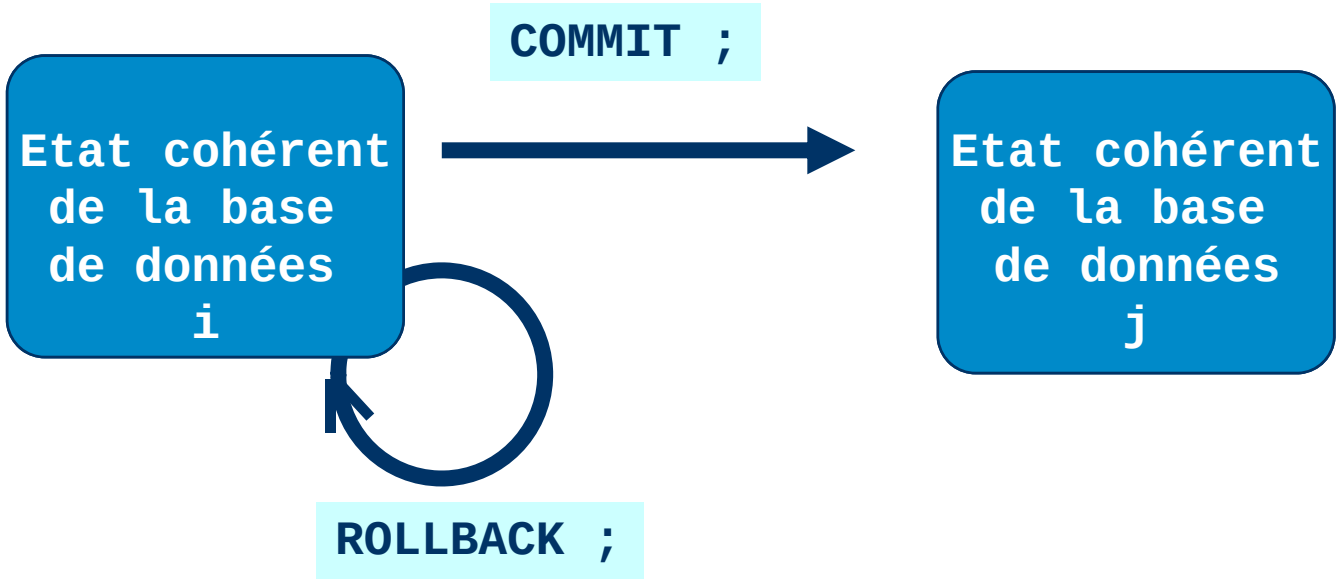
- L'écriture comptable complète nécessite 3 instructions SQL

```
INSERT INTO compte_1010(…) VALUES(…);

INSERT INTO compte_2302(…) VALUES(…);

UPDATE ecritures_comptables
SET ecriture_passee = 1
WHERE num=12345;
```

- Les 3 instructions constituent un ordre atomique, elles doivent être validées (ou annulées) ensemble !

Commandes SQL pour les transactions

- COMMIT
  - Validation de la transaction
- ROLLBACK
  - Annulation de la transaction



Instructions de la transaction comptable

- Les trois instructions sont une seule transaction

Transaction

```
INSERT INTO compte_1010(…) VALUES(…);

INSERT INTO compte_2302(…) VALUES(…);

UPDATE ecritures_comptables
SET ecriture_passee = 1
WHERE num=12345;

COMMIT ;
```

C'est à la fin de la transaction que la base de données est cohérente. A l'intérieur de la transaction elle ne l'est pas !!

Début d'une transaction

- Il n'y a pas de commande pour définir explicitement le début d'une transaction sous Oracle.

- Le début d'une transaction est défini :
  - par le début de la session;
  - par la fin de la transaction précédente.

Fin d'une transaction

- L'exécution d'un COMMIT ou d'un ROLLBACK définit le point de synchronisation où la base de données est dans un état consistant.

- En fin de session, un COMMIT est automatiquement exécuté si la déconnexion se fait sans erreurs, sinon un ROLLBACK est exécuté.

Les schémas d'Oracle <> CREATE SCHEMA

- Un schéma est un ensemble d'objets Oracle (tables, vues, triggers, etc...) regroupés sous un même nom.

- Sous Oracle la notion de schéma est équivalente à la notion d'utilisateur.
  - Pour les puristes, un utilisateur n'est pas un schéma mais possède un schéma avec son nom...

CREATE USER...

- Attention, la commande CREATE SCHEMA n'a pas du tout la même utilité...

1

Transactions

Création de plusieurs objets dans une transaction

- Oracle propose l'instruction CREATE SCHEMA pour créer plusieurs objets (LDD) dans une transaction.
- Si l'ensemble des instructions contenues dans CREATE SCHEMA sont correctes, elles seront toutes validées, sinon elles seront toutes annulées.
- L'instruction CREATE SCHEMA ne peut comporter que des créations de tables, de vues et d'affectation de privilèges sur des objets.

1

Transactions

Création de plusieurs objets dans une transaction

- Syntaxe

```
CREATE SCHEMA AUTHORIZATION schema
 [ create_table_statement |
   create_view_statement |
   grant_statement ]...;
```

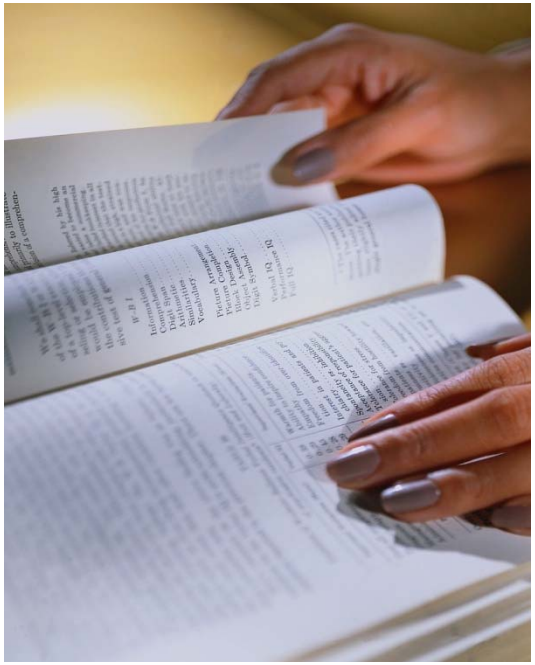
Questions?

1

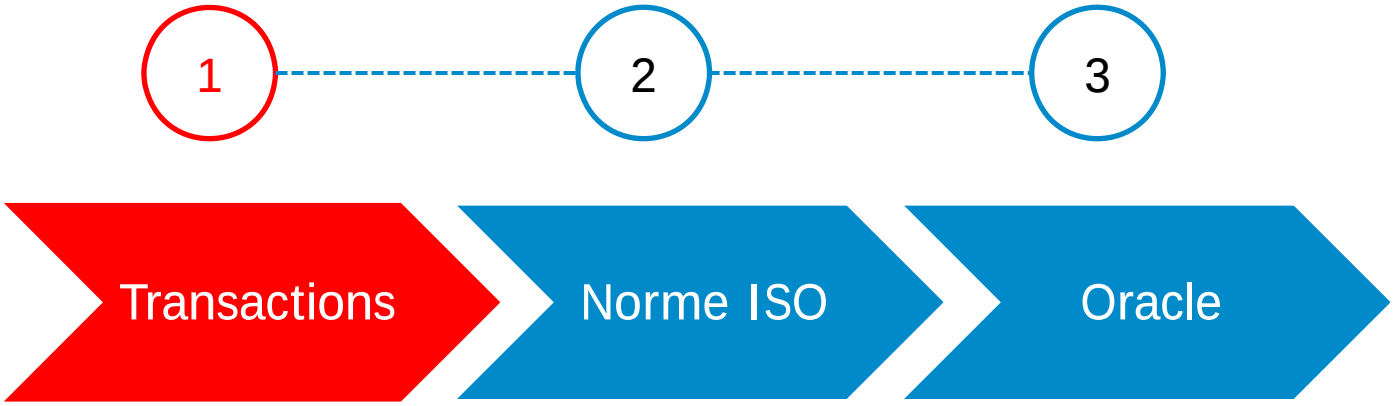
Transactions

Travaux pratiques

- Série 2, exercice 1
  - Sur Cyberlearn



Plan de la présentation



Gestion des concurrences

- L'accès concurrent par plusieurs sessions à un objet partagé doit impérativement être contrôlé.

- Si le système ne gère pas la concurrence des accès, les données risquent de devenir incohérentes
  - Missions du SII

→ quels sont les problèmes potentiels?

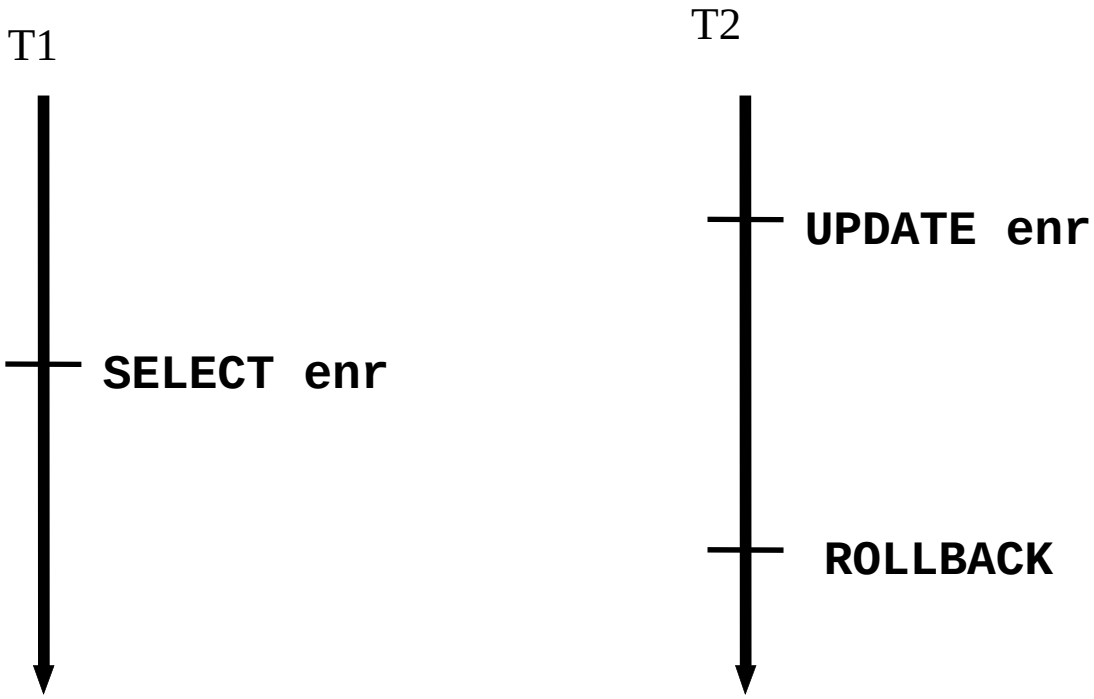
1

Transactions

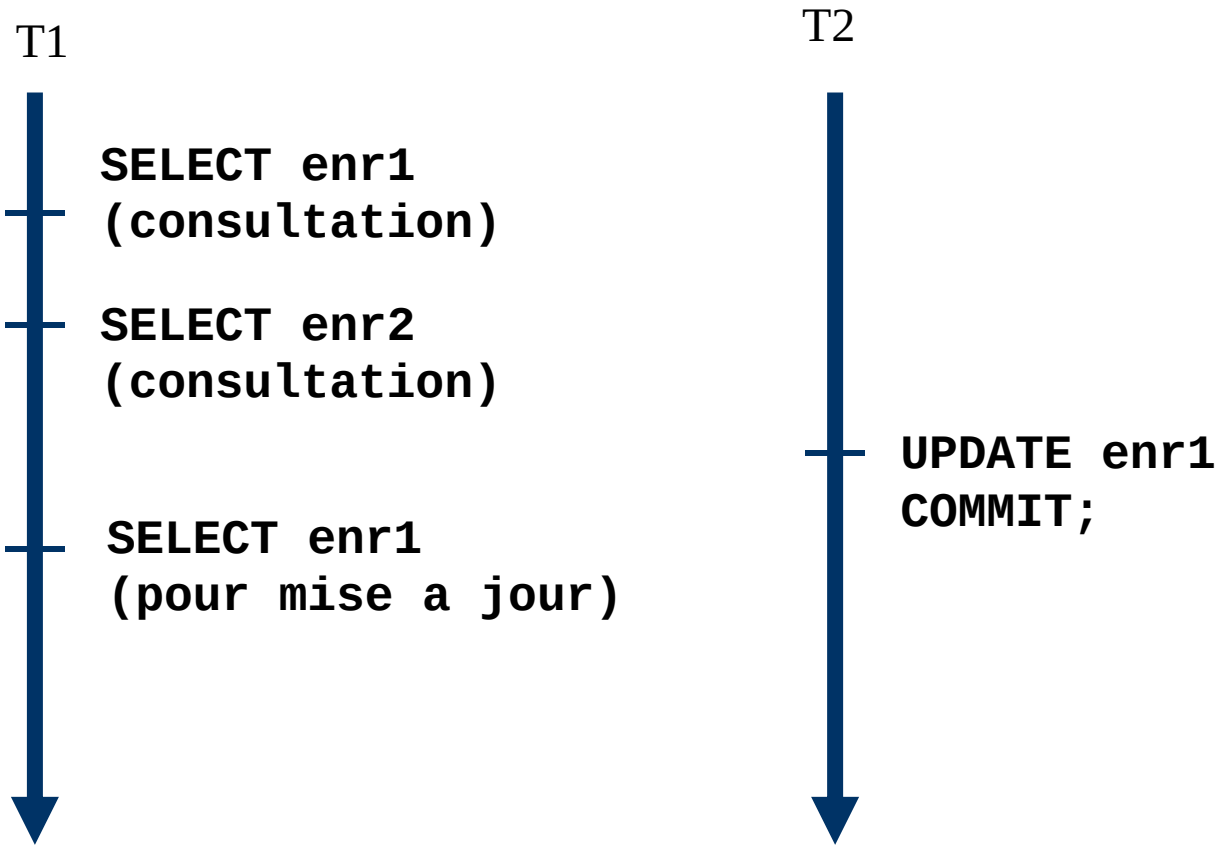
Terminologie des problématiques

- Lecture inconsistante *(dirty read)*
  - La lecture de données est fausse !!
- Lecture non répétitive *(nonrepeatable read)*
  - Deux lectures successives de la même donnée retournent des valeurs différentes
- Lecture de lignes fantômes *(phantom)*
  - Une lecture retourne une ligne qui n'existe pas !!

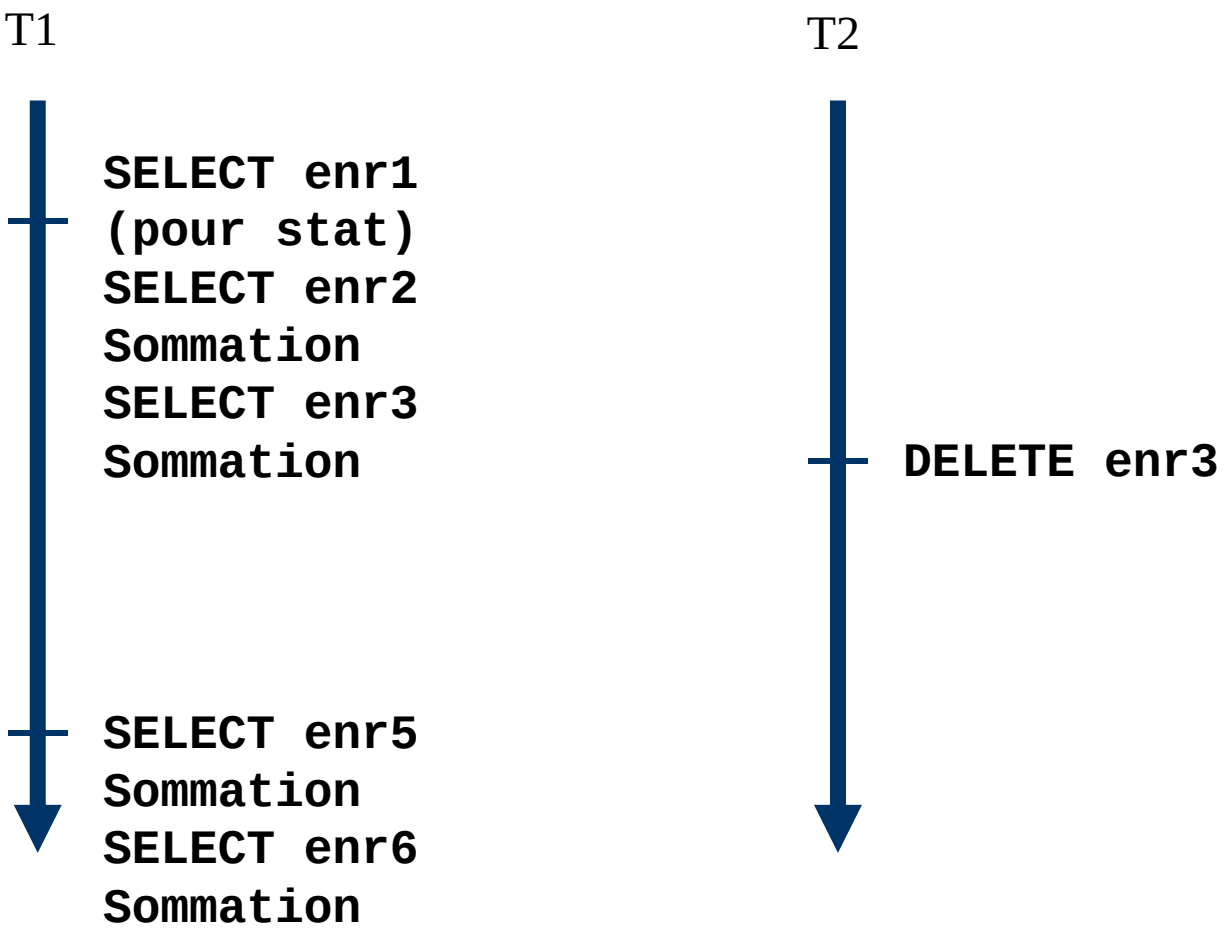
Exemple de lecture inconsistante



Exemple de lecture non répétitive



Exemple de lignes fantômes



Solution pour gérer correctement la concurrence

- Utiliser des verrous!

1

Transactions

Verrous

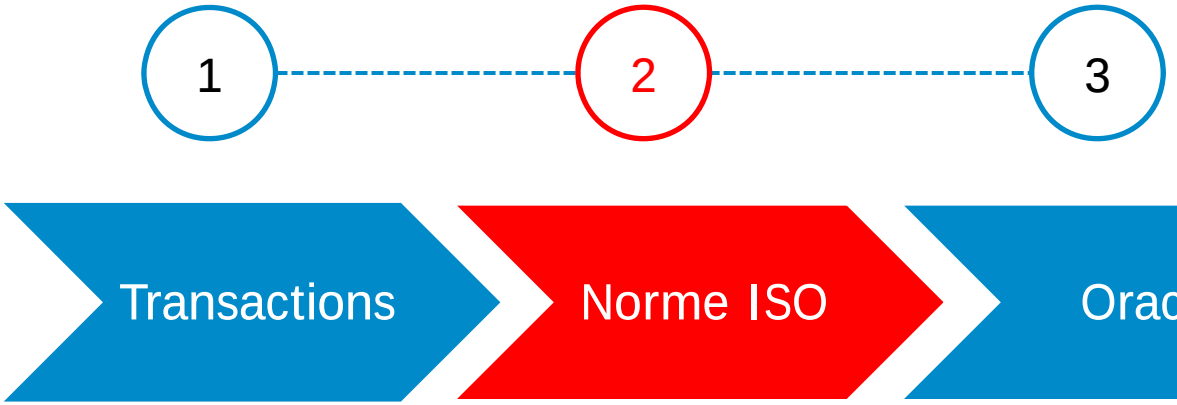
- Protection active des données pendant la transaction, par acquisition de verrous sur les enregistrements traités.

- Dans la norme, les verrous sont automatiquement réalisés par le système, donc le programmeur ne doit pas bloquer et débloquer lui-même les enregistrements.
  - C'est une bonne nouvelle non ?!?

Eléments clés des étudiants

- Une transaction doit respecter ACID !
- Si on est en multi-utilisateur et qu'il n'y a pas de transaction, il y a risque d'incohérence.
- Selon la norme, le développeur n'a rien à faire ...
- Une transaction se termine soit par un commit, soit par un rollback.

Plan de la présentation



Typologie des verrous SQL2

- Selon la norme...
- Verrou partagé, shared lock
  - utilisé pour lire 1 ou n enregistrements
- Verrou exclusif, exclusif lock
  - utilisé pour modifier 1 ou n enregistrements
- Verrou global, global lock
  - utilisé pour bloquer les enregistrements d'une table
- http://www.iso.org/iso/fr/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38646

Verrou partagé (shared lock)

- Il est utilisé pour **lire** un enregistrement et est placé par un SELECT.
- Il permet un autre accès en verrou partagé sur le même enregistrement, mais pas d'acquérir un autre type de verrou.
- Oracle ne pose pas de verrous partagés par défaut !!!

Verrou exclusif (exclusif lock)

- Il est utilisé pour **modifier** un enregistrement et est placé par les commandes INSERT,UPDATE et DELETE.

- Une fois le verrou posé, il interdit tout autre accès à l'enregistrement.
  - Pas même en lecture. (verrou partagé)

Verrou global (global lock)

- Il est utilisé pour modifier la structure d'une table par les instruction CREATE, ALTER et DROP.

- Bloque la table entière !!
- L'ensemble de la table n'est plus accessible pour les autres transactions.

Résumé des verrous implicites SQL 2

- La norme SQL 2 spécifie les verrous implicites suivants:
  - l'instruction SELECT pose implicitement un verrou partagé;
  - les instructions INSERT, UPDATE et DELETE posent implicitement un verrou exclusif.

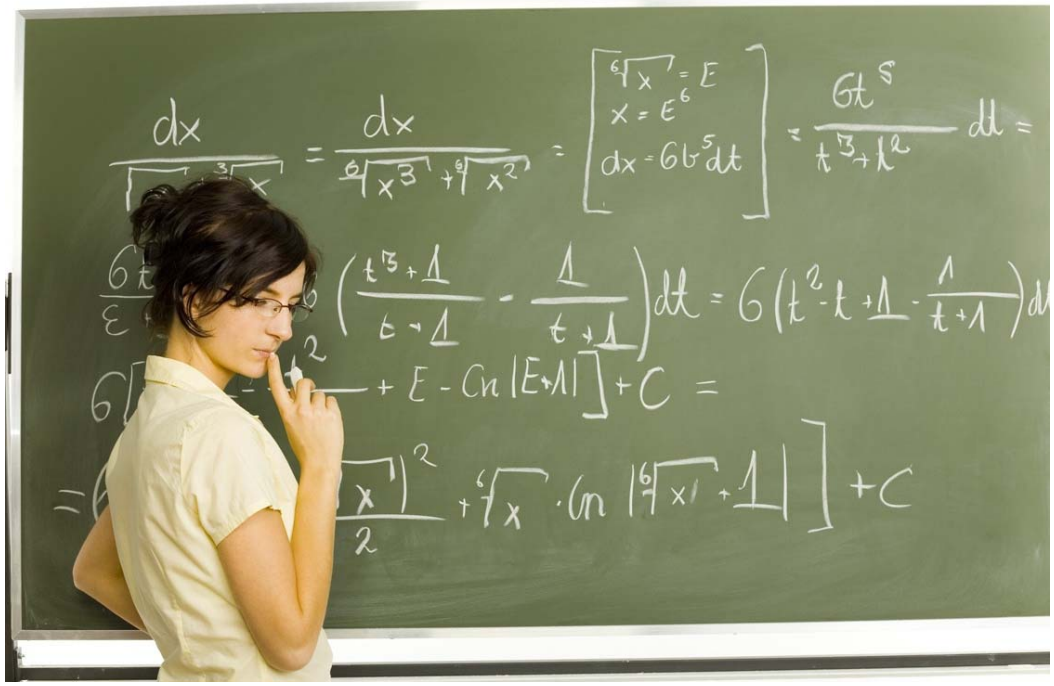
- Le programmeur ne doit en principe pas verrouiller manuellement les tuples. C'est néanmoins possible, particulièrement avec un verrou global.

Relâchement des verrous

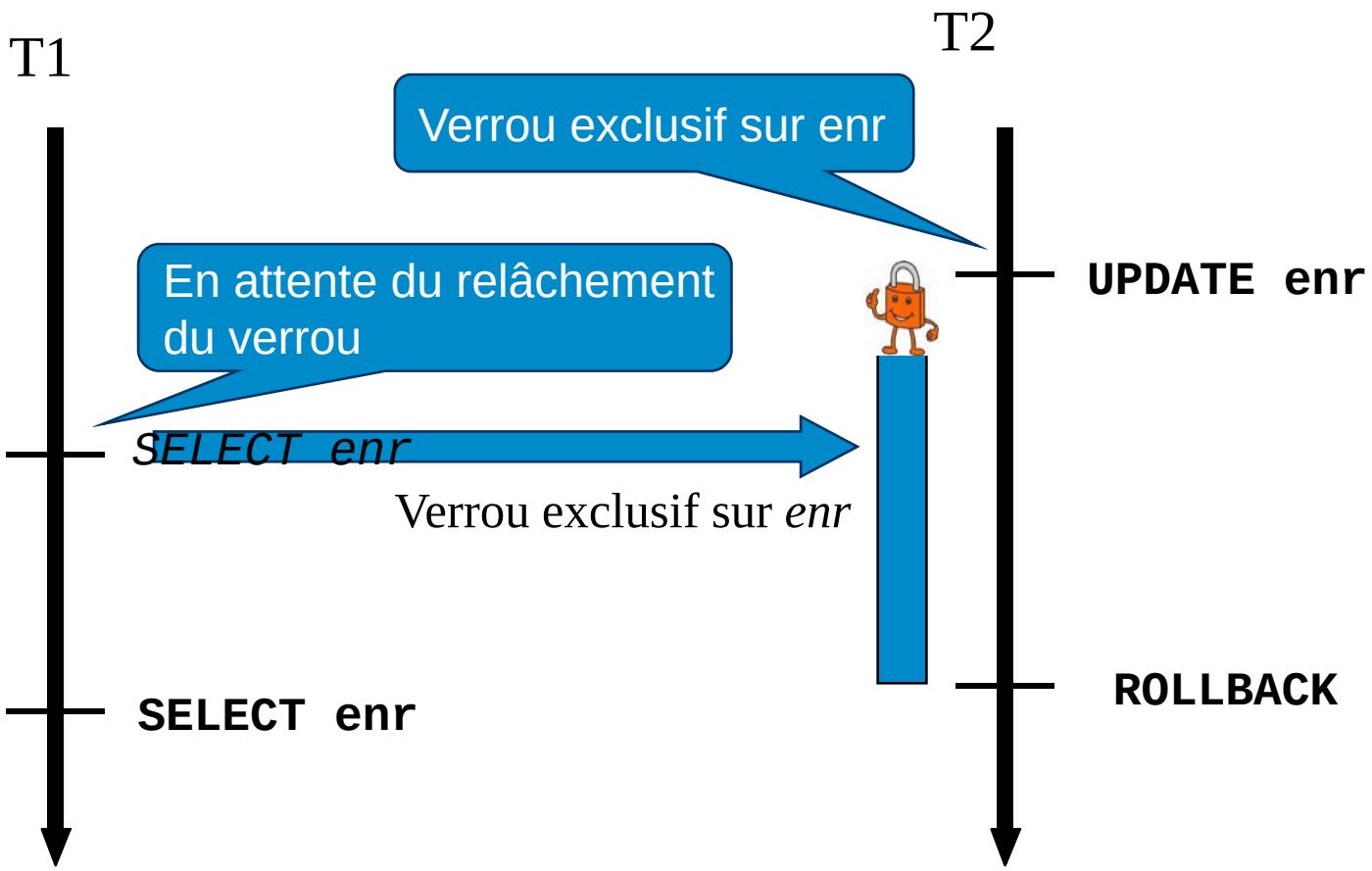
- C'est les instructions COMMIT et ROLLBACK qui relâchent les verrous → donc la fin de la transaction !
- L'ensemble des verrous posé par les instructions d'une transaction sont relâchés à la fin de la transaction.
- Il n'existe pas d'instruction explicite *UNLOCK !!*

Des solutions à nos problèmes ?

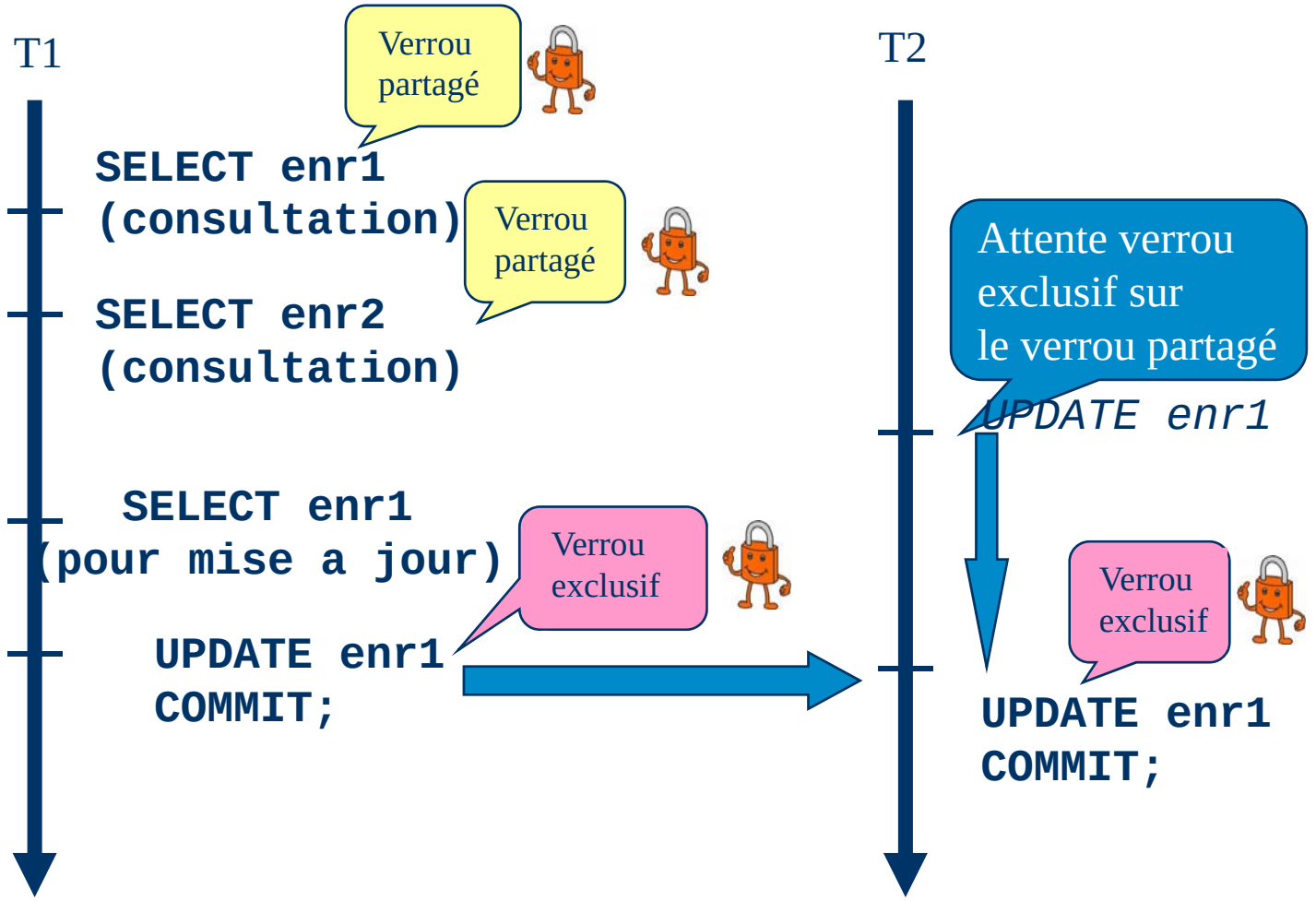
- Alors ces verrous donnent des solutions à nos problèmes ?!?
  - Lecture inconsistante
  - Lecture non répétitive
  - Lignes fantômes



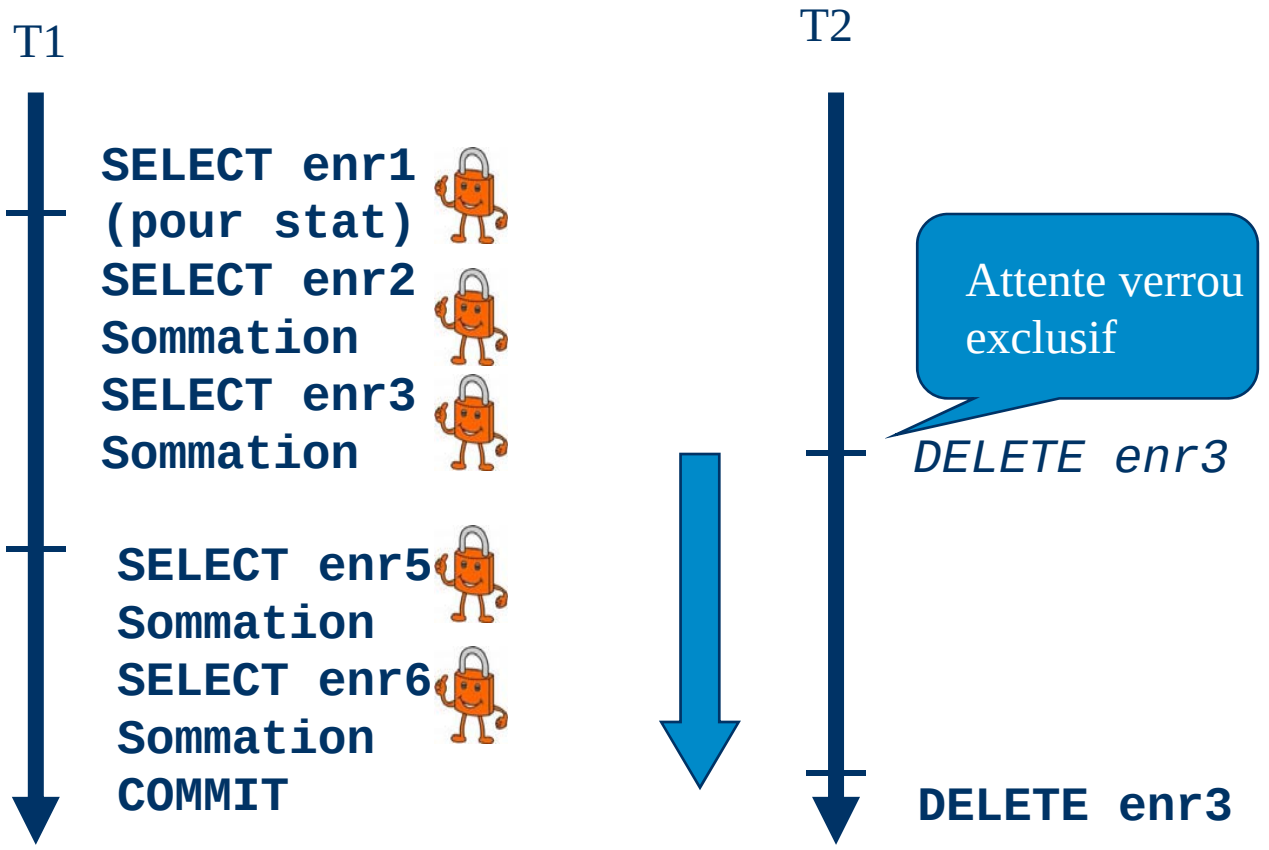
Solution de la lecture inconsistante



Solution de la lecture non répétitive



Solution évitant les lignes fantômes



Problématiques engendrés par les verrous

- La pose de verrous résout les problèmes décrits, mais ont deux conséquences négatives:
  - sérialisation des transactions provoque une attente pour la libération de la ressource;
    - La contention (maintient) engendre une dégradation de performance globale du système.
    - Le non possibilité de poser un verrou exclusif (pour modif) se nomme Famine ou live lock.
  - le verrouillage de plusieurs ressources peut être bloquant
    - Inter-verrouillage bloquant ou dead lock

Dead lock

T1

Verrou sur T1

UPDATE table 1

Attente sur verrou pour UPDATE table2;

UPDATE table 2

T2

Verrou sur T2

UPDATE table 2

Attente sur verrou pour UPDATE table1;

UPDATE table 1

Traitement du Dead lock

- Le Dead lock doit être détecté par le SGBD, car il ne peut l'être par le développeur.

- Sur détection du blocage, le système doit avertir l'utilisateur de la non possibilité de terminer la transaction.
- Le comportement du système en cas de dead lock est dépendant du produit utilisé.
  - Annulation d'une transaction ou des deux transactions
  - Message à l'utilisateur
  - ....

Verrous explicites

- Certains cas de figures (dead lock) peuvent être évités par l ’utilisation de verrous explicites.

- La pose de verrous globaux risque d’augmenter la contention sur les données et de provoque une forte dégradation des performances.

Questions?

Caractéristiques d'une transaction

- Une transaction est bornée
  - Début implicite
  - Fin explicite (COMMIT, ROLLBACK)
- Le mode de la transaction détermine les opérations possibles
  - READ ONLY
  - READ WRITE
- Niveau d'isolation de la transaction

Niveaux d'isolation

- Le niveau d'isolation indique le comportement de la transaction par rapport aux autres transactions.

- Plus le niveau d'isolation est faible, plus les autres transactions peuvent agir sur les données concernées par la première.

Niveaux d'isolation et comportement des verrous par défaut

- Selon la norme !

- READ UNCOMITED
  - Les verrous ne sont pas maintenus entre les instructions.

  → aucune isolation, toutes les erreurs possibles.

- READ COMMITTED
  - Verrou exclusif maintenus jusqu'à la fin de la transaction. Les verrous partagés ne sont pas maintenus.

  → prévient l'apparition de l'erreur de lecture inconsistante, autres erreurs possibles

Niveaux d'isolations et comportement des verrous par défaut

- REPETABLE READ
  - Les verrous partagés sont maintenus.
  - Les verrous exclusifs ne sont pas maintenus

  → erreur de lecture inconsistante et non répétitive impossible, erreur de lignes fantômes possibles.

- SERIALIZABLE
  - Tous les verrous sont maintenus

  → pas de problème possible

Erreurs possibles avec niveaux d'isolation

Niveau d'isolation	Lecture inconsistante	Lecture non répétitive	Ligne fantôme
Read uncomitted	Possible	Possible	Possible
Read committed	Impossible	Possible	Possible
Repeatable read	Impossible	Impossible	Possible
Serializable	impossible	Impossible	Impossible

Mode de transaction sécurisé

- Mode : Read/Write
- Niveau : Serializable

- Toutes opérations en toute sécurité, mais **performances diminuées**.

- Possibilité d'abaisser explicitement le niveau d'isolation.
  - Performance accrue
  - Danger, problèmes possibles

Modification des caractéristiques

- Au début de la transaction, on peut modifier ses caractéristiques, soit son mode ou son niveau d'isolation.

- Exemple
  - SET TRANSACTION

    READ ONLY

    ISOLATION LEVEL READ UNCOMMITTED ;

- Les nouvelles caractéristiques seront actives lors de toute la transaction et les valeurs par défaut seront reprises en fin de transaction.

Contraintes et transaction

- Les contraintes déclaratives du schéma seront contrôlées soit après chaque ordre SQL soit en fin de transaction, à l exécution du COMMIT.

- Une contrainte peut donc être dans deux « états »
  - IMMEDIATE
    - Contrôlée après chaque ordre SQL
  - DEFERRED
    - Contrôlée en fin de transaction, à l exécution du COMMIT

Déclaration de la contraintes

- Lors de sa déclaration, on donne à la contraintes deux attributs
  - Si elle est « deferrable »,
  - Son état par défaut

- Syntaxe :

  <constraint_attributes> ::= [NOT]DEFERRABLE

  [INITIALLY DEFERRED | INITIALLY IMMEDIATE]

- Exemple

```
… CONSTRAINT pk_table PRIMARY KEY
        DEFERRABLE INITIALLY IMMEDIATE ;
```

Modification de la contrainte

- Au début d'une transaction, on peut modifier l 'état de la contraintes si elle est définie DEFERRABLE.

- Syntaxe

```
SET CONSTRAINTS {liste_contrainte | ALL}
                {DEFERRED | IMMEDIATE}
```

- Exemple

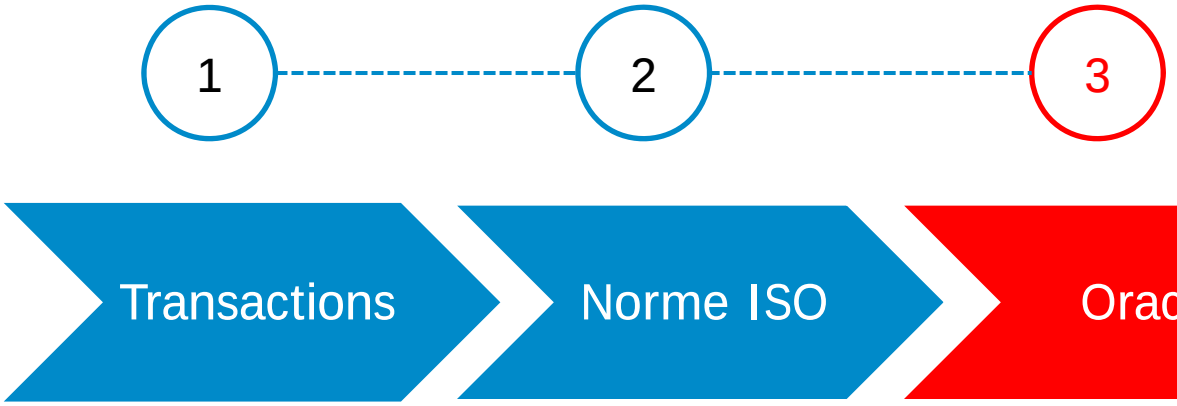
```
SET CONSTRAINT pk_table DEFERRED ;
```

- L'état défini par la clause INITIALY sera repris en fin de transaction (COMMIT)

Questions?

Eléments clés des étudiants

Plan de la présentation



Gestion de concurrence par Oracle

ORACLE

- Les niveaux supportés sont
  - Read committed
  - Serializable
- Pour des raisons de performance, le niveau d'isolation par défaut d 'Oracle est *Read Committed*.
- La transaction peut-être mise *serializable*

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ;
```

Verrouillage avec Oracle

ORACLE

- Pour préserver la cohérence des données, Oracle implémente la notion de verrou.
- Verrouillage implicite
  - Le moteur effectue automatiquement des verrous prédéfinis.
- Verrouillage explicite
  - Contrôle manuel des mécanismes de verrouillages.
    - Au niveau de la transaction
    - Au niveau de l'instance (DBA)

Verrouillage des données

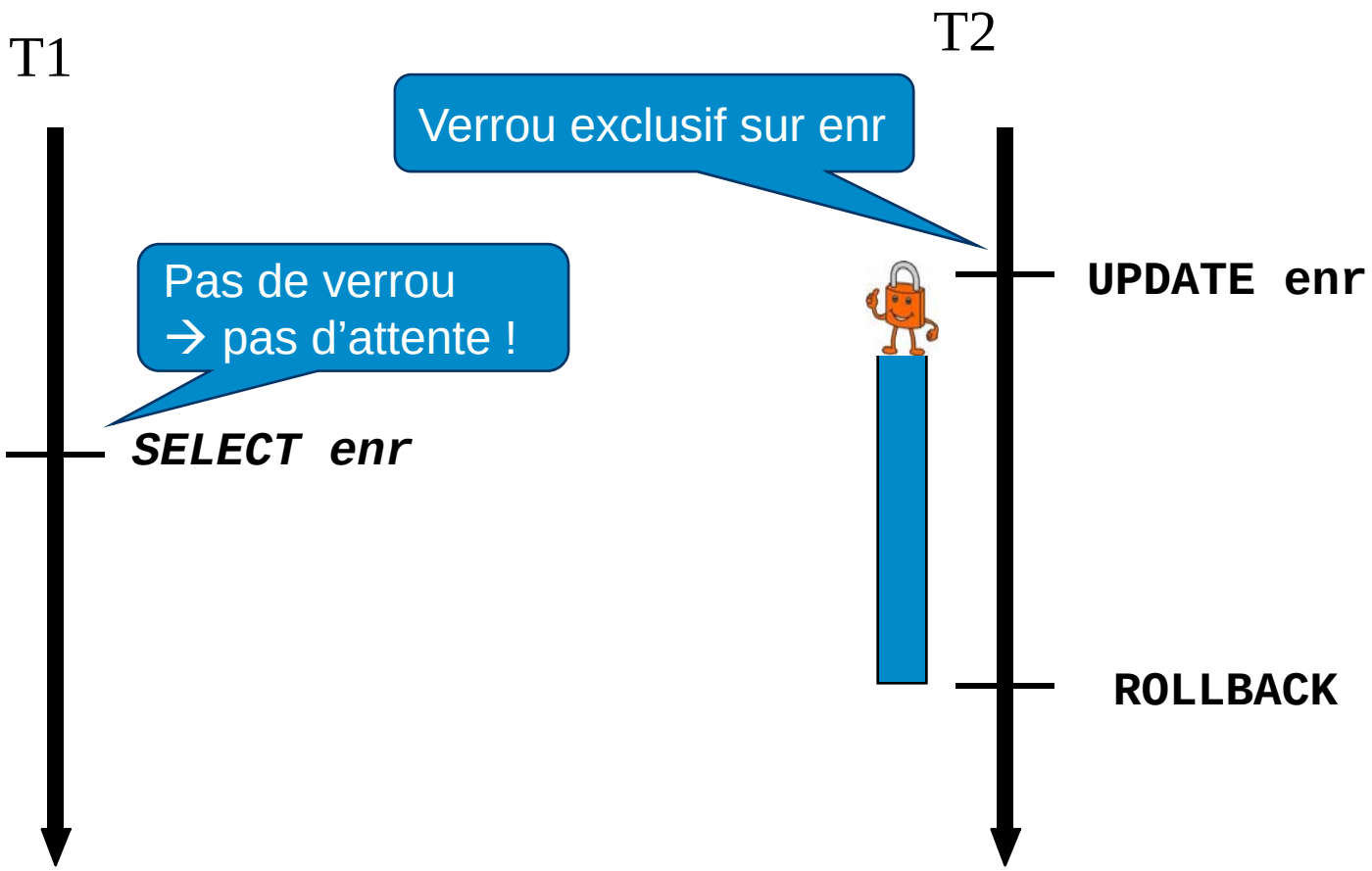
ORACLE

- Oracle ne suit pas vraiment la norme mot à mot...
  - Ce n'est pas le seul.
- Beaucoup utilisé :
  - Mode lignes partagées (Row share)
  - Mode lignes exclusives (Row exclusive)
  - Mode table exclusive (Exclusive)
- Moins utilisé :
  - Mode table partagée (Share)
  - Mode partage exclusif de ligne (Share row exclusive)

Verrouillage des données en mode read committed sur Oracle

- SELECT → pas de verrou
- INSERT, UPDATE et DELEVE
  - Les lignes manipulées par une commande d'écriture seront verrouillées, mode lignes exclusives (Row Exclusive), pendant la transaction.
- LDD → Mode table exclusive (Exclusive)

La lecture inconsistante sur Oracle



Verrouillage partagé de lignes

ORACLE

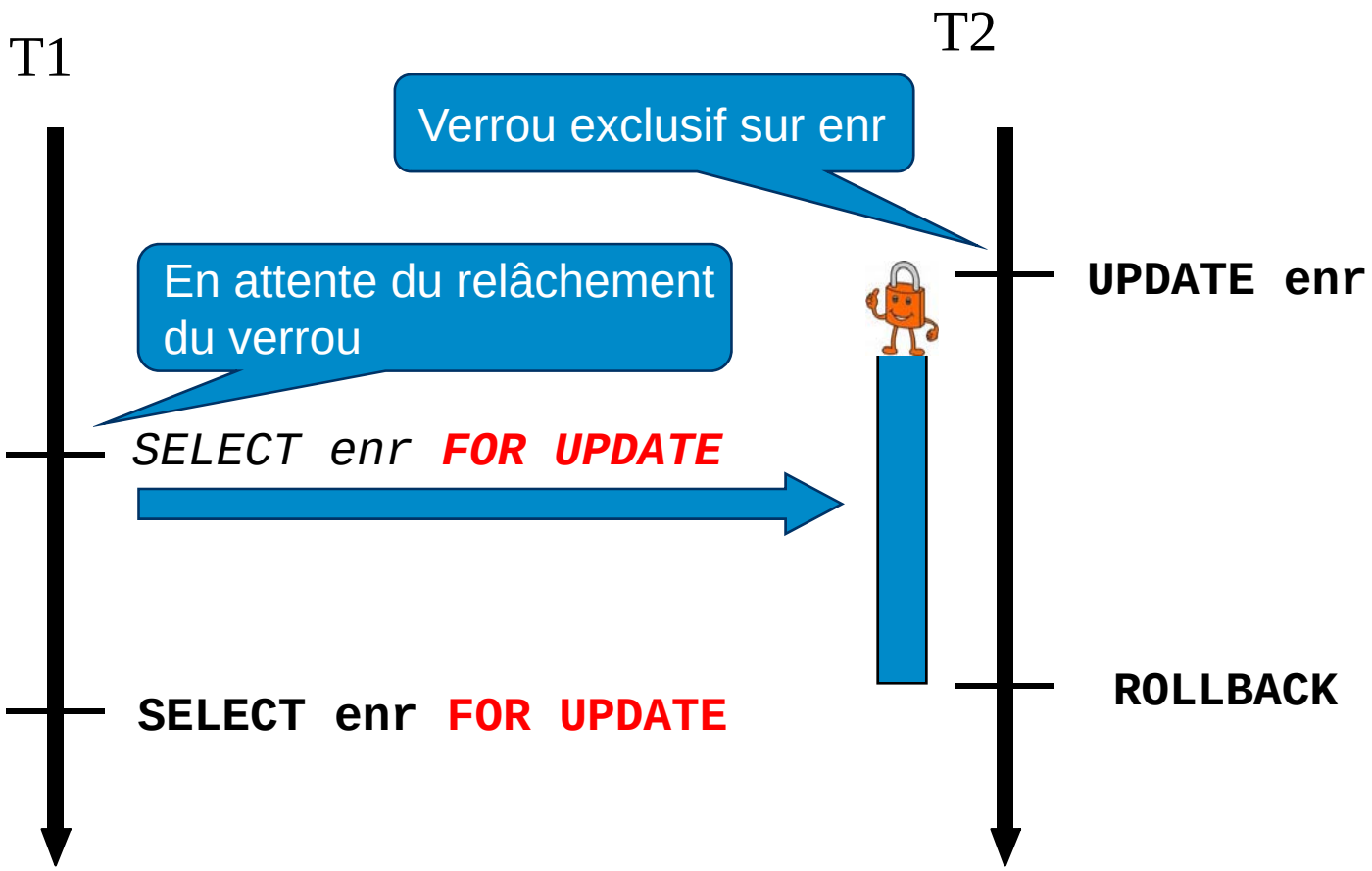
- Sous Oracle, l'instruction SELECT associé au niveau d'isolation READ COMMITTED ne garde pas les verrous.
  - Lecture inconsistante possible !!!

- Dans ce mode d'isolation la clause FOR UPDATE permet de poser des verrous de lignes partagées (Row share) sur tous les tuples qui satisfont la clause WHERE.

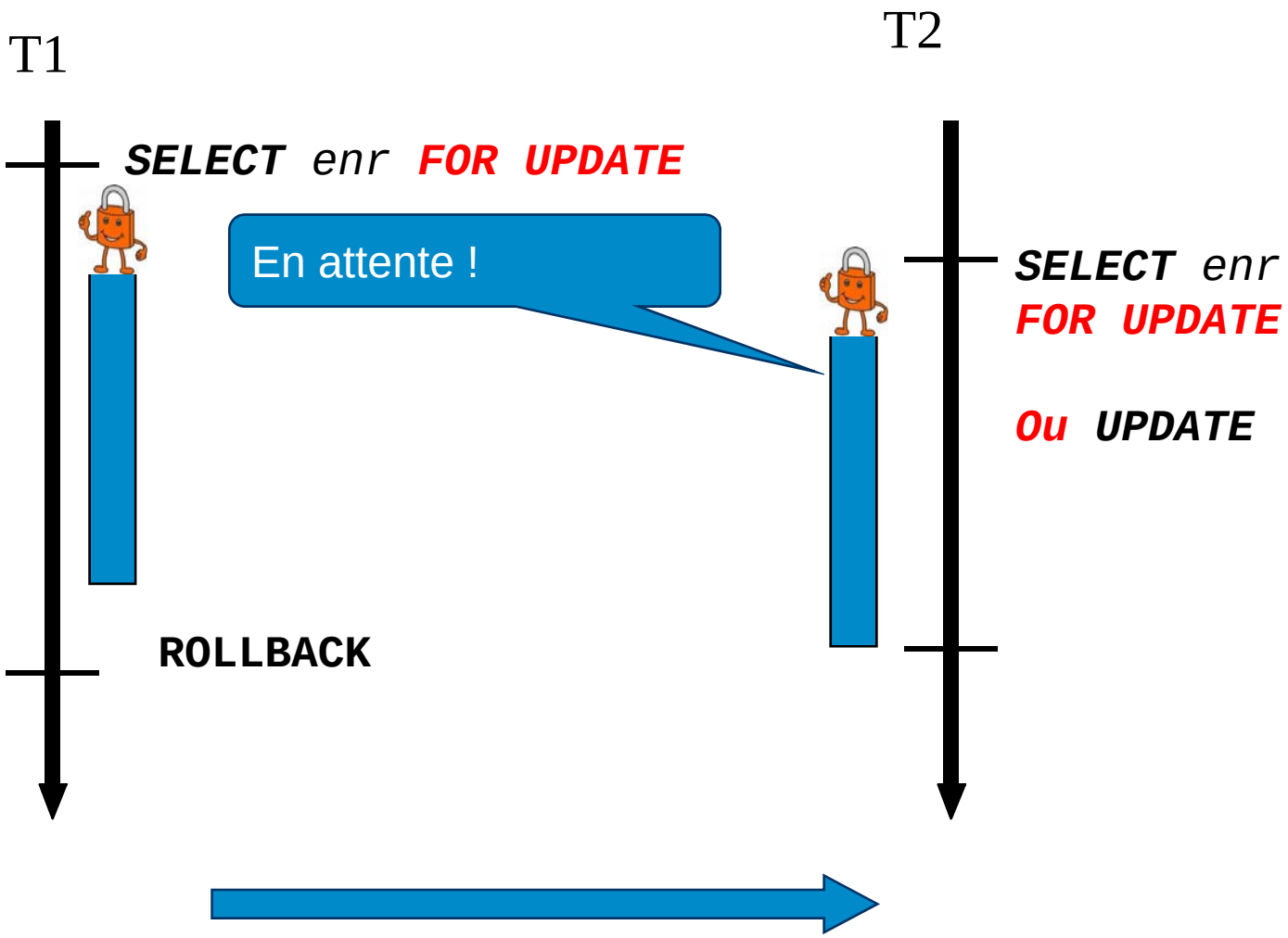
```
SELECT liste_selection
   FROM nom_table
   WHERE condition
   FOR UPDATE ;
```

La lecture inconsistante sur Oracle

ORACLE



La lecture inconsistante sur Oracle ou plus intelligent…



Lecture consistante avec transaction en lecture seule

ORACLE

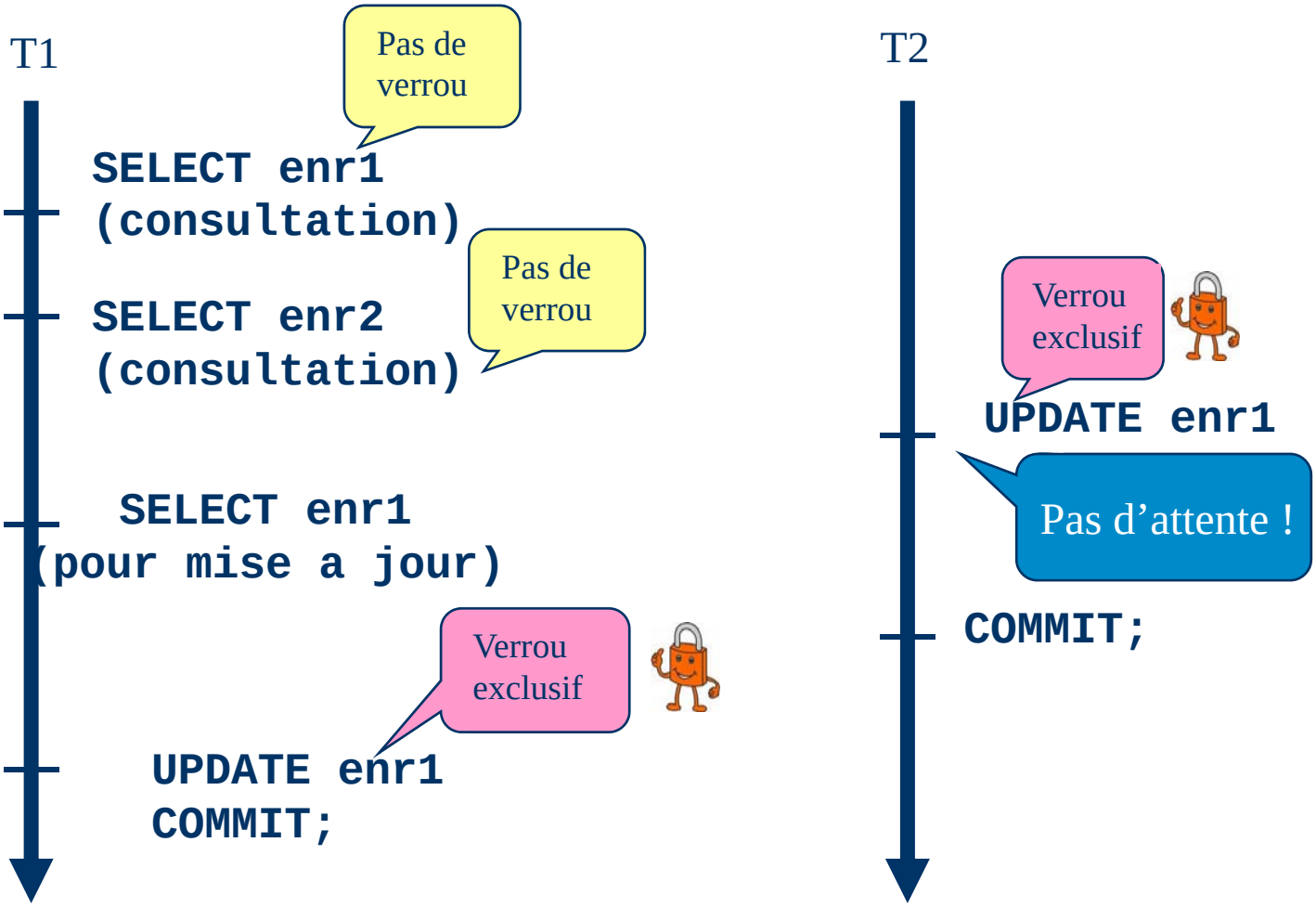
- De part le niveau d'isolation par défaut, l'instruction SELECT ne pose pas de verrous implicites :
  - !! Lecture inconsistante
- Possibilité de transaction en lecture seule

```
SET TRANSACTION READ ONLY
```

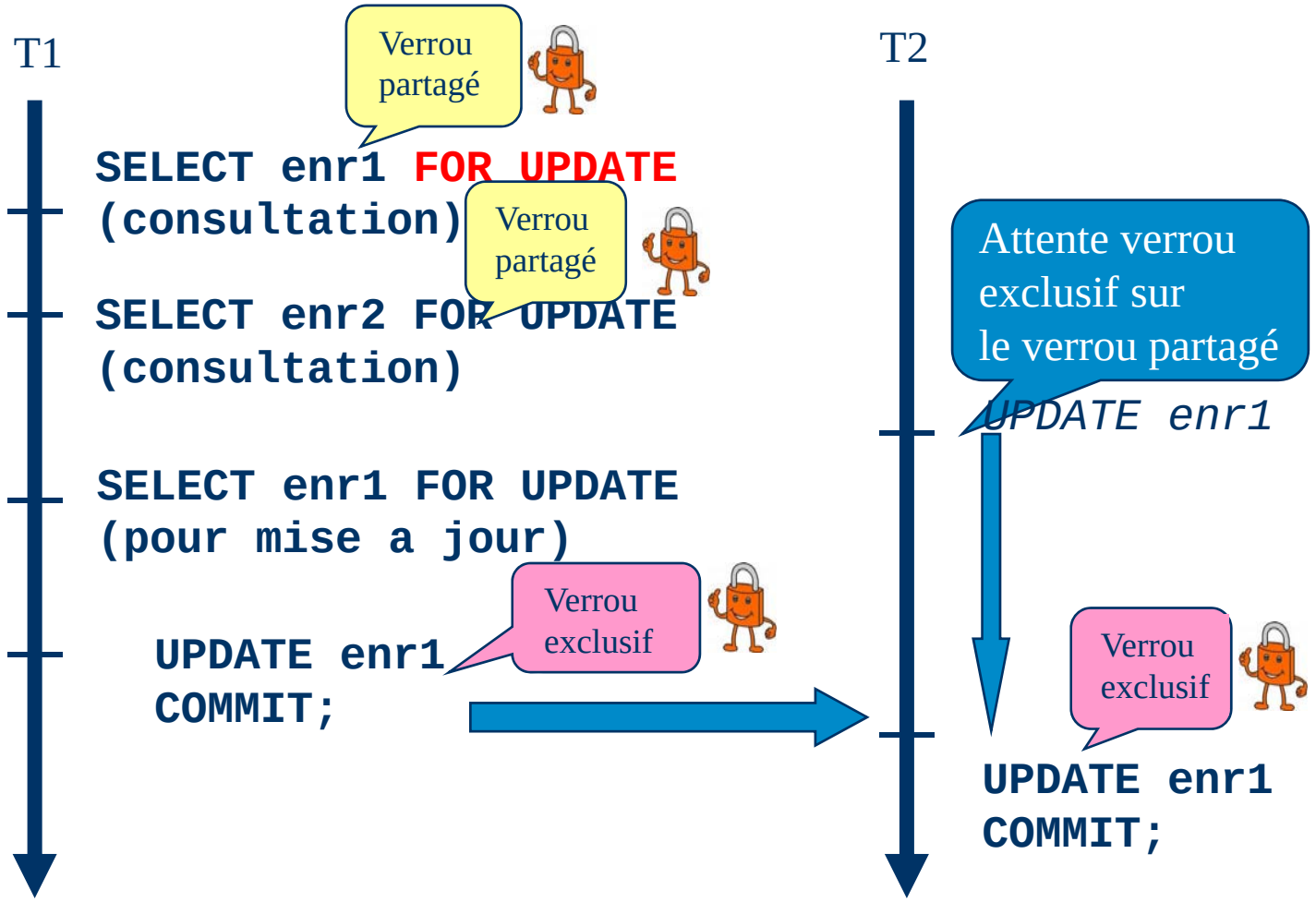
  - C 'est une lecture consistante sans verrous
- A utiliser pour les états et les calculs puisque cette transaction ne mettra pas les autres en attente.

Lecture non répétitive sur Oracle

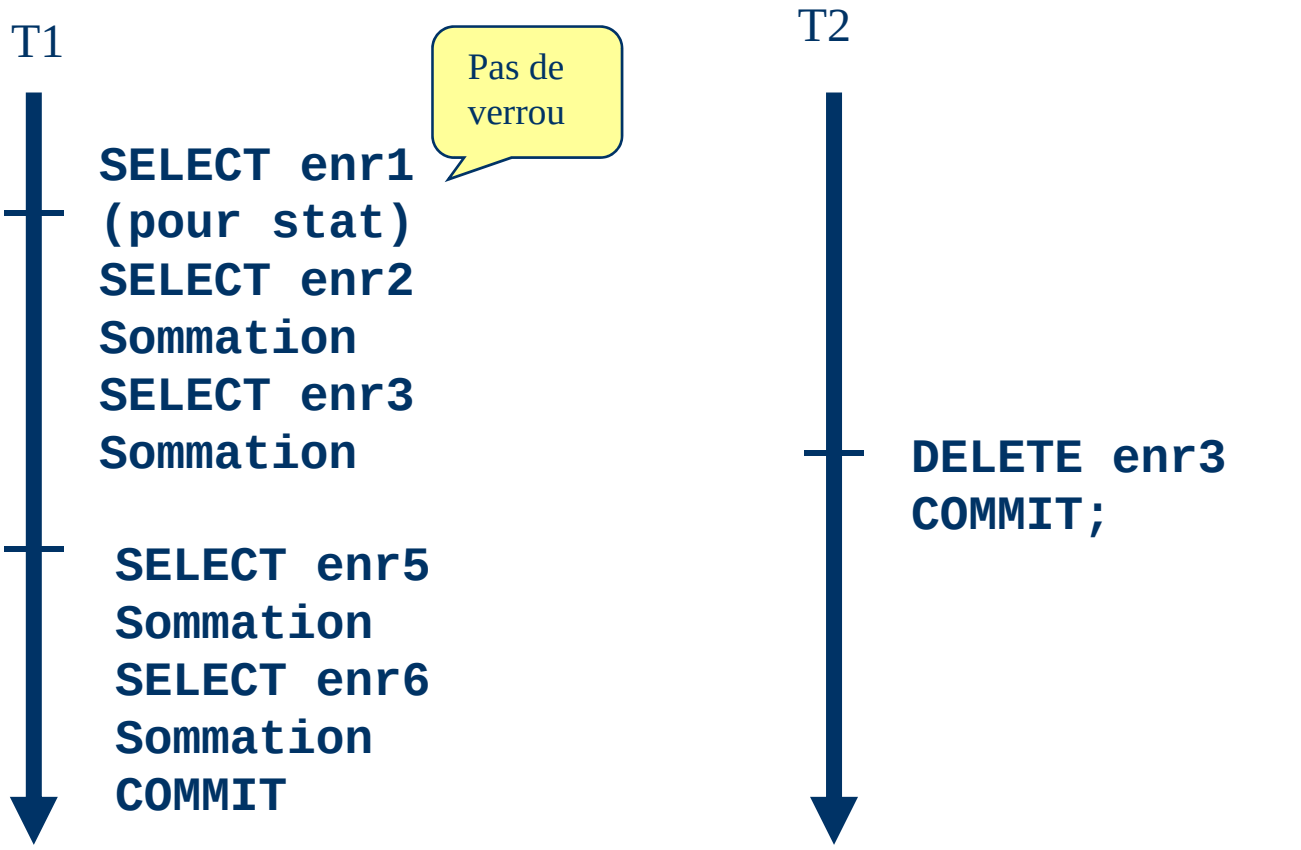
ORACLE



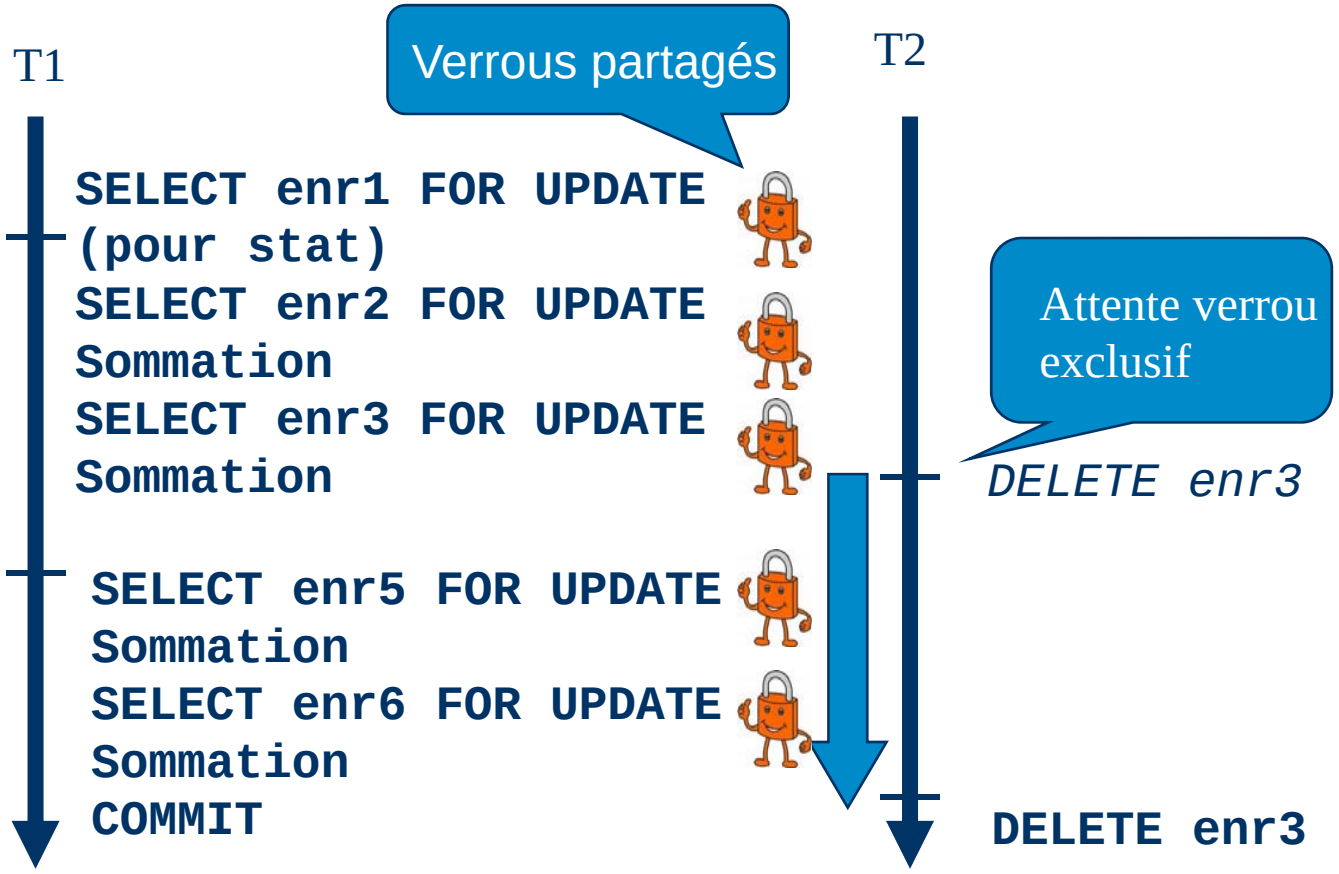
Lecture non répétitive sur Oracle



Lignes fantômes sur Oracle



Lignes fantômes sur Oracle



Verrouillage explicite

ORACLE

- Avec Oracle, seul le verrouillage explicite sur une table pour la transaction est possible.
- Tous les tuples de la tables seront verrouillés
- Syntaxe :

```
LOCK TABLE nom_table
  IN mode_verrouilage MODE ;
```

Verrouillage explicite

- Le SELECT FOR UPDATE est un verrouillage explicite. Il en existe d'autres types.
- Le verrouillage explicite sur une table pour la transaction est possible.
- Tous les tuples de la tables seront verrouillés
- Syntaxe :

```
LOCK TABLE nom_table
  IN mode_verrouillage MODE ;
```

  - Mode_verrouillage [ROW SHARE | ROW EXCLUSIVE | SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE]

Verrouillage explicite

- ROW SHARE, RS
  - Idem SELECT FOR UPDATE mais sur toute une table.
- ROW EXCLUSIVE, RX
  - Idem INSERT, UPDATE et DELETE.
- SHARE, S
  - Permet aux autres transactions exclusivement les SELECT, SELECT FOR UPDATE et les LOCK TABLE ... IN SHARE MODE et SHARE ROW MODE.
- SHARE ROW EXCLUSIVE, SRX
  - Permet aux autres transactions exclusivement les SELECT, SELECT FOR UPDATE et les LOCK TABLE ... IN SHARE MODE.
- EXCLUSIVE, X
  - Permet uniquement la lecture par les autres transactions.

Récapitulation verrou sous Oracle

ORACLE

Instruction	Verrou	Comptabilité				
		RS	RX	S	SRX	X
SELECT	Aucun	O	O	O	O	O
INSERT	RX	O	O	N	N	N
UPDATE	RX	O*	O*	N	N	N
DELETE	RX	O*	O*	N	N	N
SELECT FOR UPDATE	RS	O*	O*	O*	O*	N
LOCK TABLE ... IN ROW SHARE MODE	RS	O	O	O	O	N
LOCK TABLE ... IN ROW EXCLUSIVE MODE	RX	O	O	N	N	N
LOCK TABLE ... IN SHARE MODE	S	O	N	O	N	N
LOCK TABLE ... IN SHARE ROW EXCLUSIVE MODE	SRX	O	N	N	N	N
LOCK TABLE ... IN EXCLUSIVE MODE	X	N	N	N	N	N

O* : Oui sauf si verrou incompatible → attente

Source : DEMAL P. SQL2-SQL3

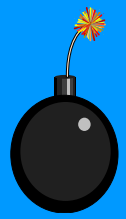
Traitement du Dead Lock

ORACLE

- Le SGBD Oracle détecte bien les Dead Lock et son comportement est le suivant:
  1. Détection du verrouillage bloquant
  2. Avertissement par message de la transaction qui s 'est faite bloquée.
     - ORA-00060: deadlock detected while waiting for resource
  3. Le SGBD ne fait pas de ROLLBACK !!

- C'est au client d'annuler lui même la transaction.
  - Ce comportement est très bien adapté au mode client-serveur, puisque il est aisé de réessayer la transaction une fois que l 'autre transaction aura relâché ses verrous.

Verrouillage du dictionnaire



- Chaque instruction est une transaction.
  - Chaque instruction termine la transaction en cours.
  - !! Attention aux transactions comportant des instructions du LMD.

- Les verrous sur le dictionnaire sont implicites et peuvent êtres posés dans deux modes:
  - Exclusif
    - Suppression, modification d'une structure de table.
  - Partagé
    - Création de tables, vues, procédures et synonymes.

Contraintes et transactions sur Oracle

ORACLE

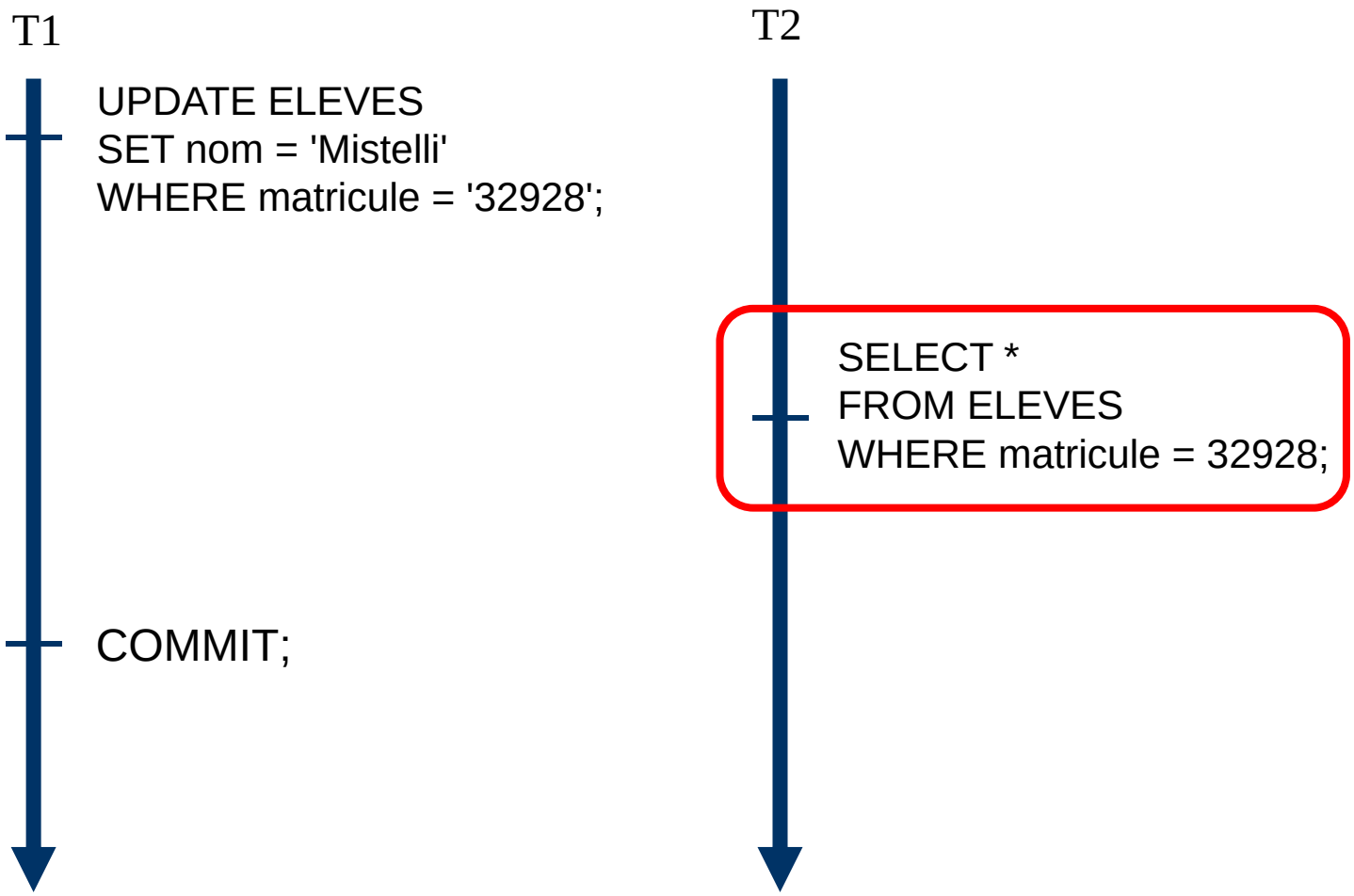
- Les contraintes du schéma peuvent être contrôlées selon la norme.
  - `IMMEDIATE`
    - Contrôlée après chaque ordre SQL
  - `DEFERRED`
    - Contrôlée en fin de transaction, à l exécution du COMMIT

- Commande en début de transaction

```
SET CONSTRAINTS {liste_contrainte | ALL}
 {DEFERRED | IMMEDIATE} ;
```

Questions?

Quizz 1
En read committed sur Oracle



Quizz 1
En read committed sur Oracle

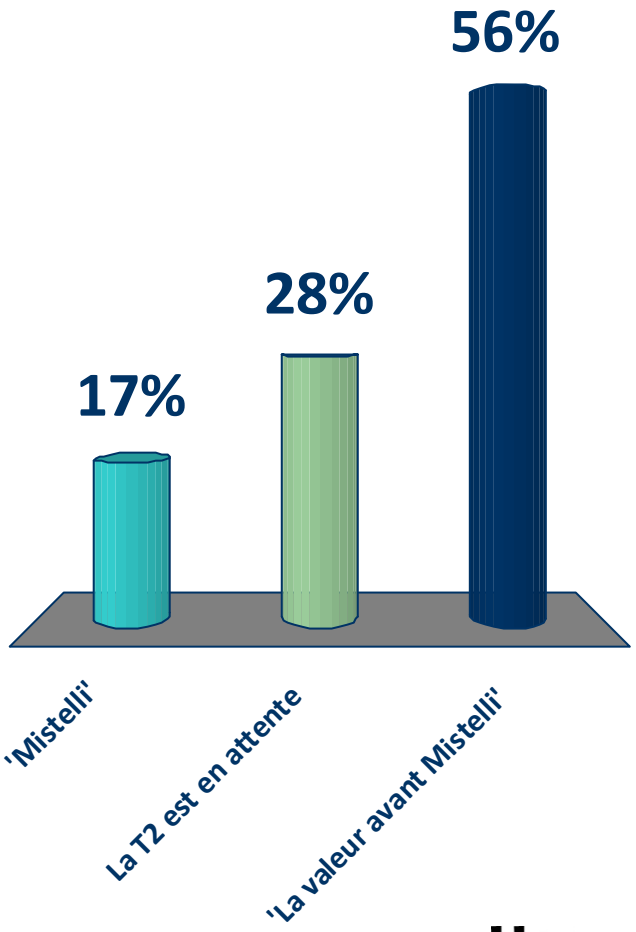
Quelle est la valeur que voit l'utilisateur de T2 ?

SELECT * FROM ELEVES WHERE matricule = 32928;

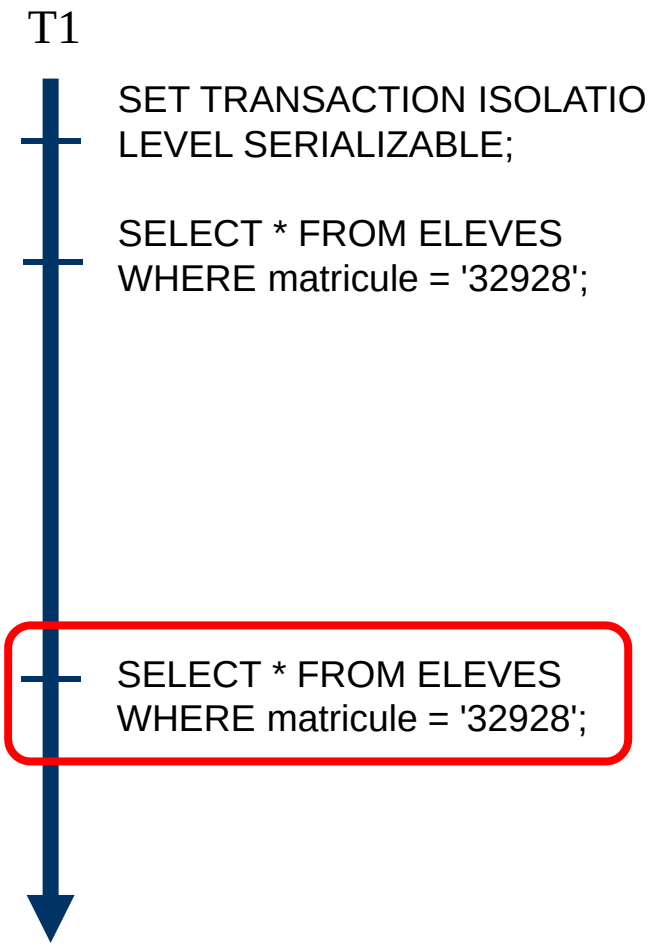
A. 'Mistelli'

B. La T2 est en attente

C. 'La valeur avant Mistelli'



Quizz 2
Sur Oracle



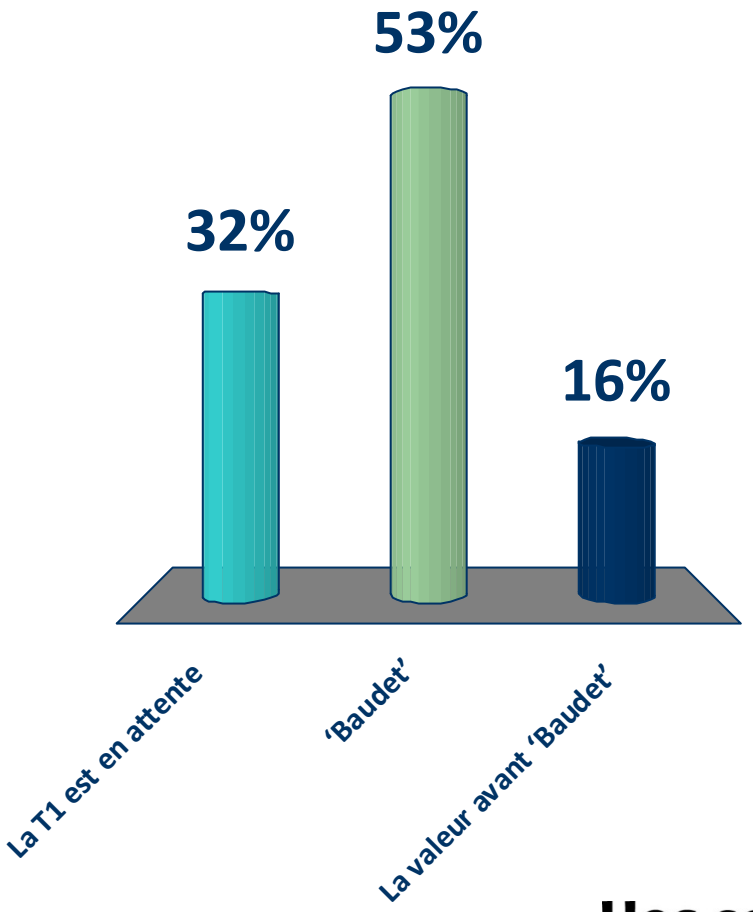
Quizz 2
Sur Oracle

Quelle est la valeur sur T1 dans le dernier SELECT ?

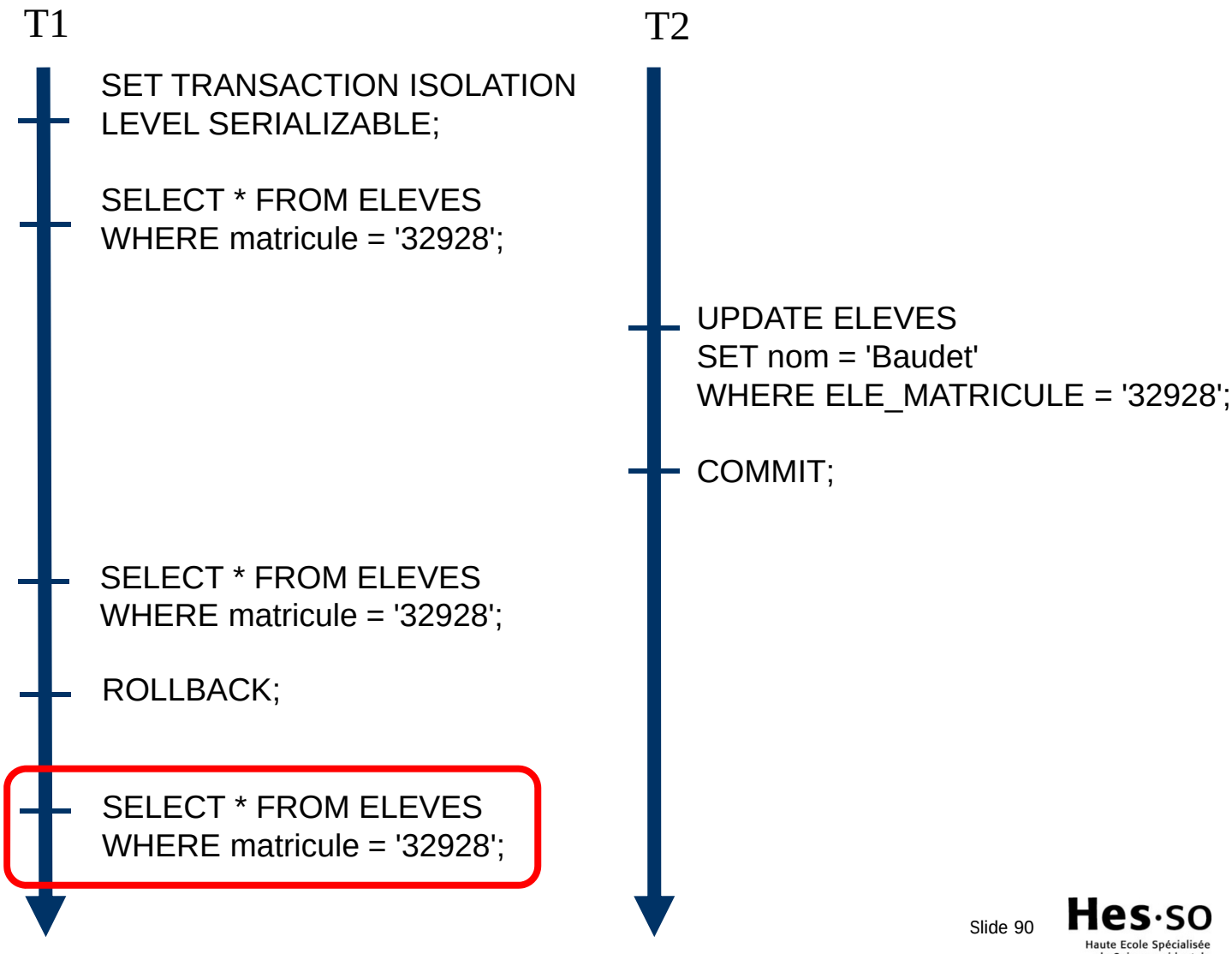
A. La T1 est en attente

B. ‘Baudet’

C. La valeur avant ‘Baudet’



Quizz 3
Sur Oracle



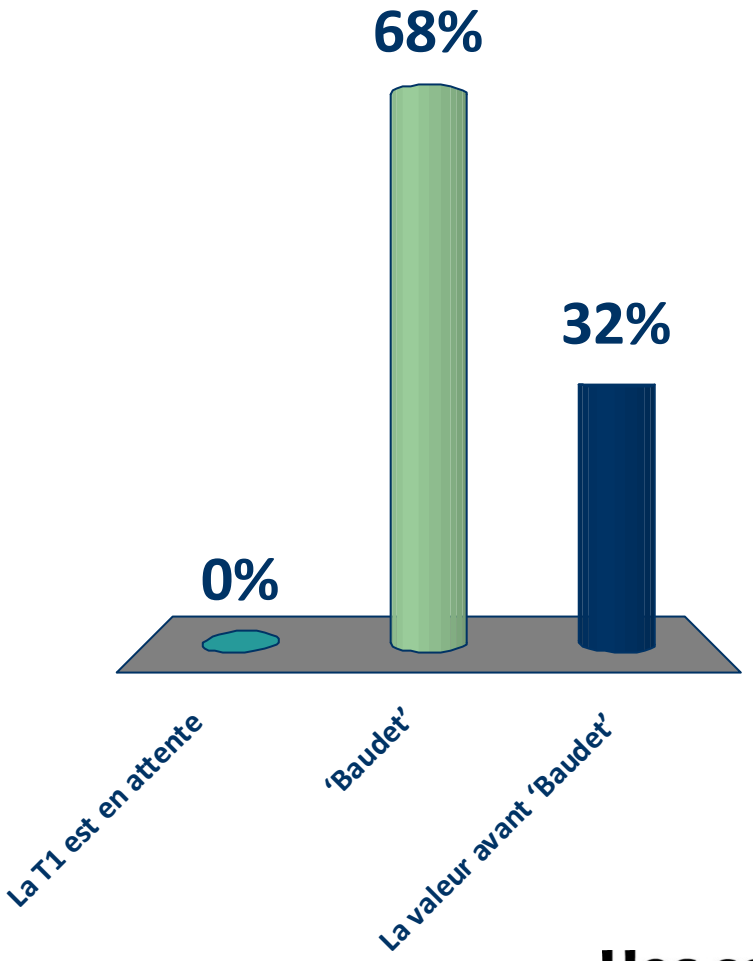
Quizz 3
Sur Oracle

Quelle est la valeur sur T1 dans le dernier SELECT ?

A. La T1 est en attente

B. ‘Baudet’

C. La valeur avant ‘Baudet’



Eléments clés des étudiants

Travaux pratiques

ORACLE

- Série 2, exercice 2
  - Sur Cyberlearn