Sécurisation des accès aux données

Nous verrons...

- Gestion des privilèges (droits)
- Vues
- Transactions
- Il existe bien d'autres pistes à explorer (les injections de code SQL par exemple)

Objets gérés par le SGBD

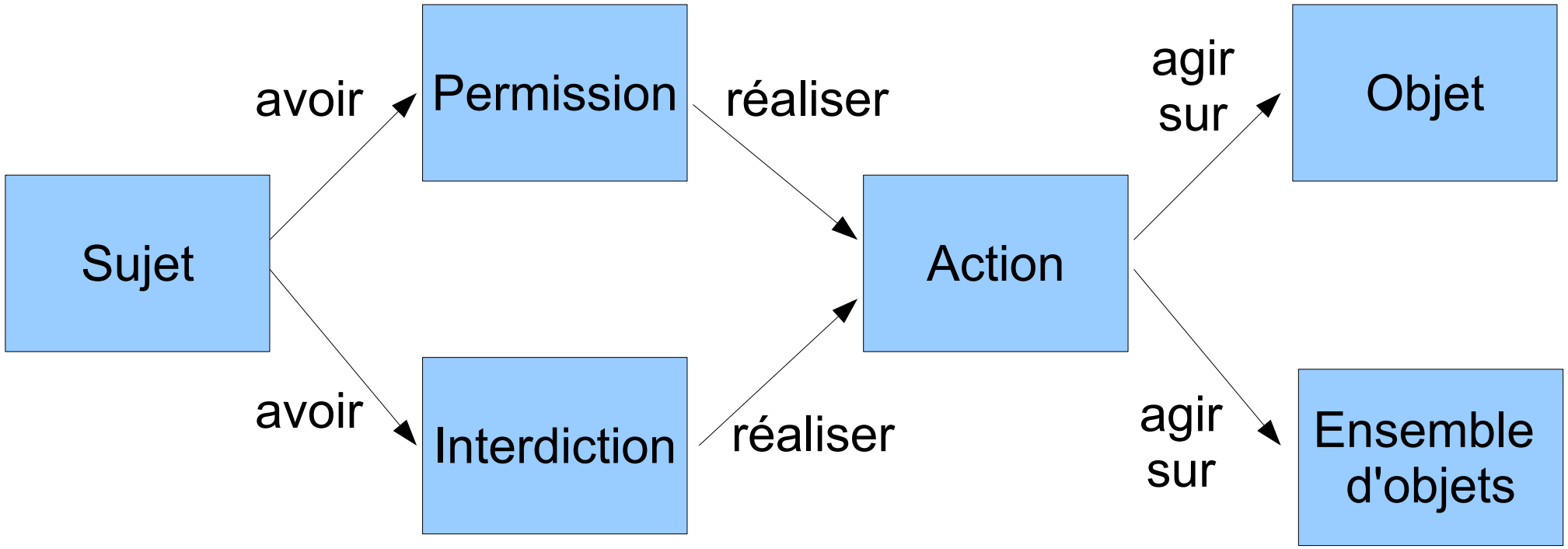
- De nombreux types d'objets :
  - Tables
  - Vues
  - Utilisateurs
  - Synonymes
  - ...
- Chaque objet créé a un **propriétaire** (celui qui l'a créé)
- A la création de l'objet, le propriétaire est le seul à pouvoir manipuler l'objet, mais il peut accorder des privilèges sur ces objets à d'autres utilisateurs

Privilèges

- Un **privilège** est un droit d'exécuter un type particulier de commande SQL, ou un droit d'accéder à un objet d'un autre utilisateur
- Un **rôle** est un ensemble de privilèges
- Privilèges systèmes : gérés par l'administrateur
- Privilèges vis à vis des objets : créés par les utilisateurs

Privilèges

Forme des règles



Inspiré de http://www-smis.inria.fr/~pucheral/

Ex : l'utilisateur compta a accès en lecture aux numéros des employés et à leurs salaires

Privilèges systèmes

Créer/modifier des objets essentiels de la base

```
GRANT {priv-syst | rôle}
   [, {priv-syst | rôle}]...
TO {utilisateur | rôle |PUBLIC}
   [, {utilisateur | rôle |PUBLIC}] ...
[WITH ADMIN OPTION] ;
```

`WITH ADMIN OPTION` permet au bénéficiaire de transmettre ses privilèges

Rôle CONNECT

- se connecter à la base en appelant l'un des outils Oracle (SQL*Plus, par ex.)
- changer son mot de passe
- manipuler les objets (tables, vues) de la base si des droits (select, insert, ...) lui ont été accordés par les propriétaires des objets.
- transmettre des autorisations sur des objets s'il a obtenu le droit de retransmettre.
- créer des vues et des synonymes (nous en parlons d'ici peu) pour des objets autorisés.

Rôle RESOURCE

- créer des tables, index
- donner des droits de manipulation (ou les reprendre...) sur ses propres objets à d'autres utilisateurs.

Rôle DBA

- créer des utilisateurs
- donner (ou retirer) des privilèges aux utilisateurs
- accéder à tous les objets de la base
- créer des synonymes publics
- effectuer les opérations de maintenance de la base.

Exemples de privilèges systèmes

- CREATE TABLE
- CREATE USER
- CREATE VIEW
- CREATE PROCEDURE
- CREATE SYNONYM
- ...

Supprimer des privilèges

```
REVOKE {priv-syst | rôle} [, {priv-syst
| rôle}]...

FROM {utilisateur | rôle | PUBLIC}
   [,{utilisateur | rôle | PUBLIC}]... ;
```

Exemple

- Création (par un dba nécessairement) de l'utilisateur `rh` avec un mot de passe :

```
grant connect to rh
identified by mdprh ;
```

- L'utilisateur `dev` a, en plus, la possibilité de définir des structures de tables (et des index,...) :

```
grant connect, ressource dev
identified by mdpdev ;
```

- rh obtient les mêmes privilèges que dev :

```
grant ressource to rh ;
```

Exemples

- C'était une erreur...

```
revoke ressource from rh ;
```

- Finalement `rh` n'obtient que le droit de créer des tables, avec la permission de transmettre ce privilège...

```
grant create table to rh with admin option ;
```

- … et le transmet à `chef_equipe`

```
grant create table to chef_equipe ;
```

Privilèges sur les objets

Donnés par un utilisateur sur les objets qu'il a créés

```
GRANT {droit [, droit] ... | ALL } [(colonne
[, colonne]...)]
ON [utilisateur.]objet
TO {utilisateur | rôle [, utilisateur |
rôle]] ... | PUBLIC}
[ WITH GRANT OPTION ] ;
```

Exemples

- `rh` obtient des privilèges sur l'objet emp (donnés par le propriétaire ou une personne ayant le droit), et le droit de transmettre ces privilèges

```
grant select, update(sal) on emp to
rh with grant option ;
```

- `rh` permet à tous les employés de situer les autres employés (suppose que `rh` est le propriétaire de emp ou s'est vu attribuer les privilèges permettant d'exécuter cette commande)

```
grant select(ename, job, deptno) on
emp to public ;
```

Reprise de droits

```
REVOKE {droit [, droit ] ... | ALL}
ON [utilisateur.]objet
FROM {utilisateur | rôle [,
utilisateur | rôle] ... | PUBLIC} ;
```

NB : si la personne qui m'avait donné un droit le perd, je le perds aussi

Vues

- Perception de la base plus proche des besoins de l'utilisateur
- Définie par une requête
- Le SGBD stocke en général la définition, pas le résultat (voir USER_VIEWS ou ALL_VIEWS – colonne text)
- Sécurité : les données en dehors de la vue sont protégées
- Facilitent l'écriture de requêtes complexes

Interrogation de vues

- Comme pour une table normale
- Le SGBD enrichit la question avec la définition de vue qu'il trouve dans le dictionnaire de données

Création d'une vue

CREATE VIEW nomVue [(colonne [, colonne] ...)

AS Requête

[WITH CHECK OPTION] ;

- Pas de ORDER BY dans la requête

Exemple

```
create view personnel
as select empno,ename,job,dname,loc
from emp JOIN dept ON emp.deptno =
dept.deptno ;

select * from personnel
where lower(job)='analyst';
```

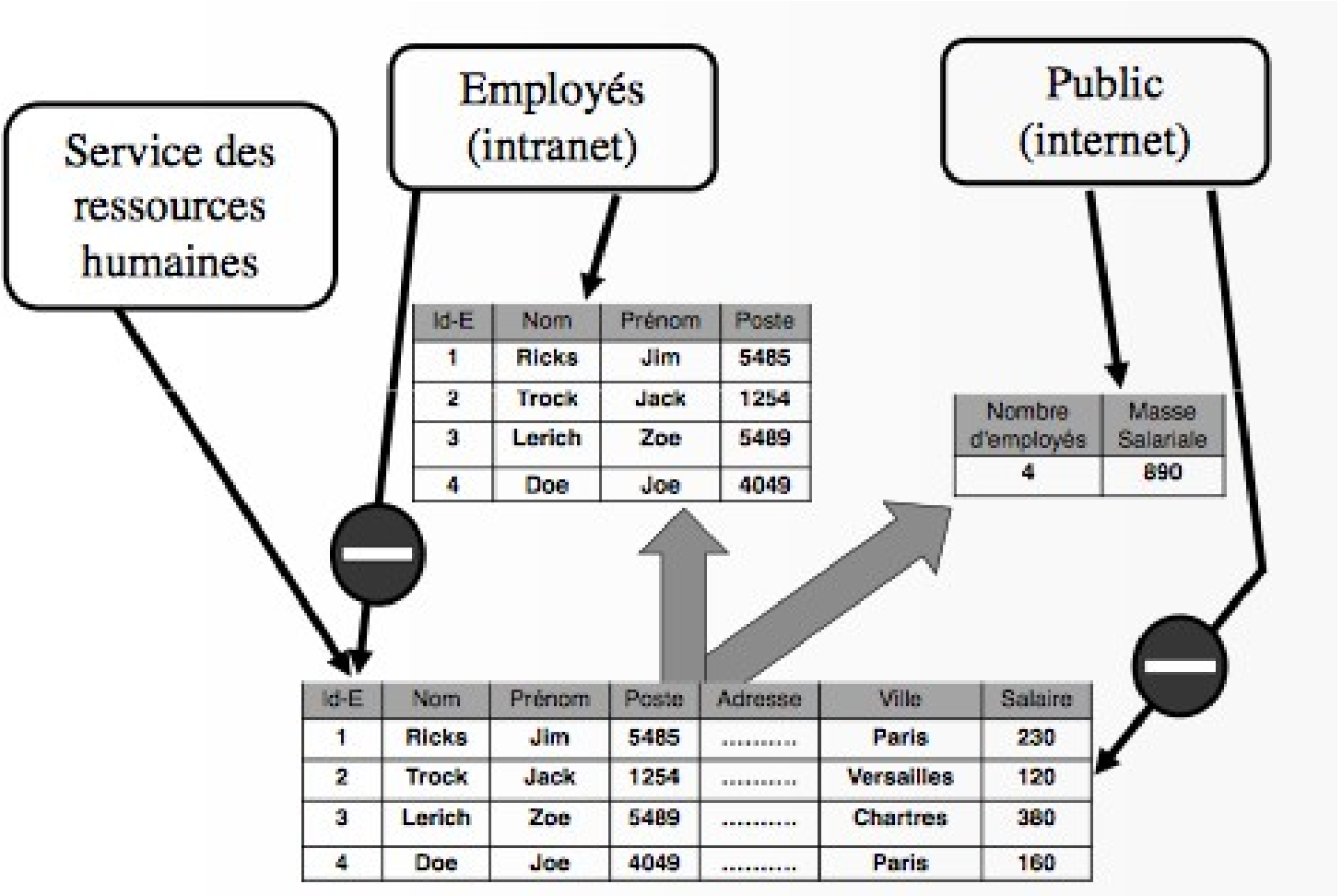
Requête effectivement exécutée :

```
select empno, ename, job, dname, loc
from emp JOIN dept ON emp.deptno =
dept.deptno
where lower(job)='analyst' ;
```

Suppression d'une vue

- `DROP VIEW nom_vue;`
- En principe une vue n'a pas d'existence physique
- La destruction ne détruit pas de tuples

Confidentialité via les vues



http://www-smis.inria.fr/~pucheral/

Transaction

Une transaction est une séquence atomique d'opérations (générées par des ordres SQL) qui fait passer la base d'un **état cohérent** à un autre état cohérent.

Par atomique, on entend que toutes les opérations de mise à jour incluses dans la transaction doivent être faites, ou aucune (c'est tout ou rien).

Transaction : exemple classique

Un transfert de n euros d'un compte A à un compte B

- Retirer n de A
- Mettre n sur B

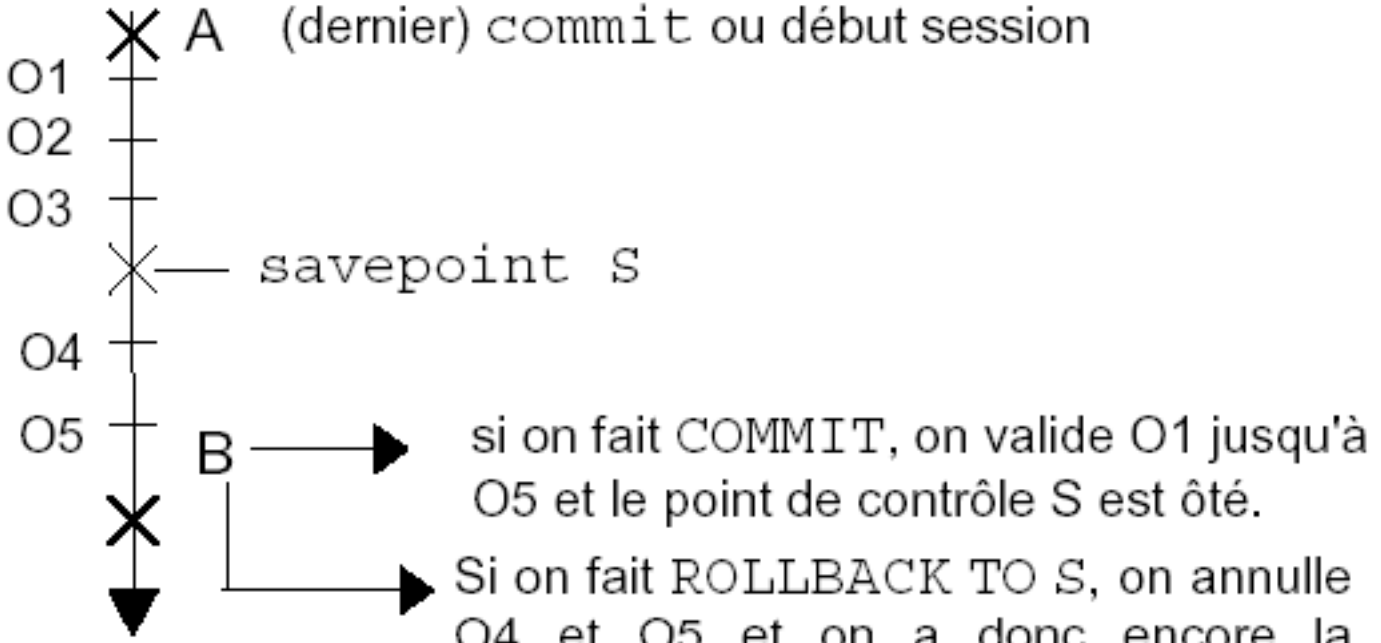
Propriétés ACID d'une transaction

- Atomicité : une transaction doit s'exécuter en tout ou rien
- Cohérence : une transaction doit respecter l'ensemble des contraintes d'intégrité
- Isolation : une transaction ne voit pas les effets des transactions concurrentes
- Durabilité : les effets d'une transaction validée ne sont jamais perdus, quel que soit le type de panne

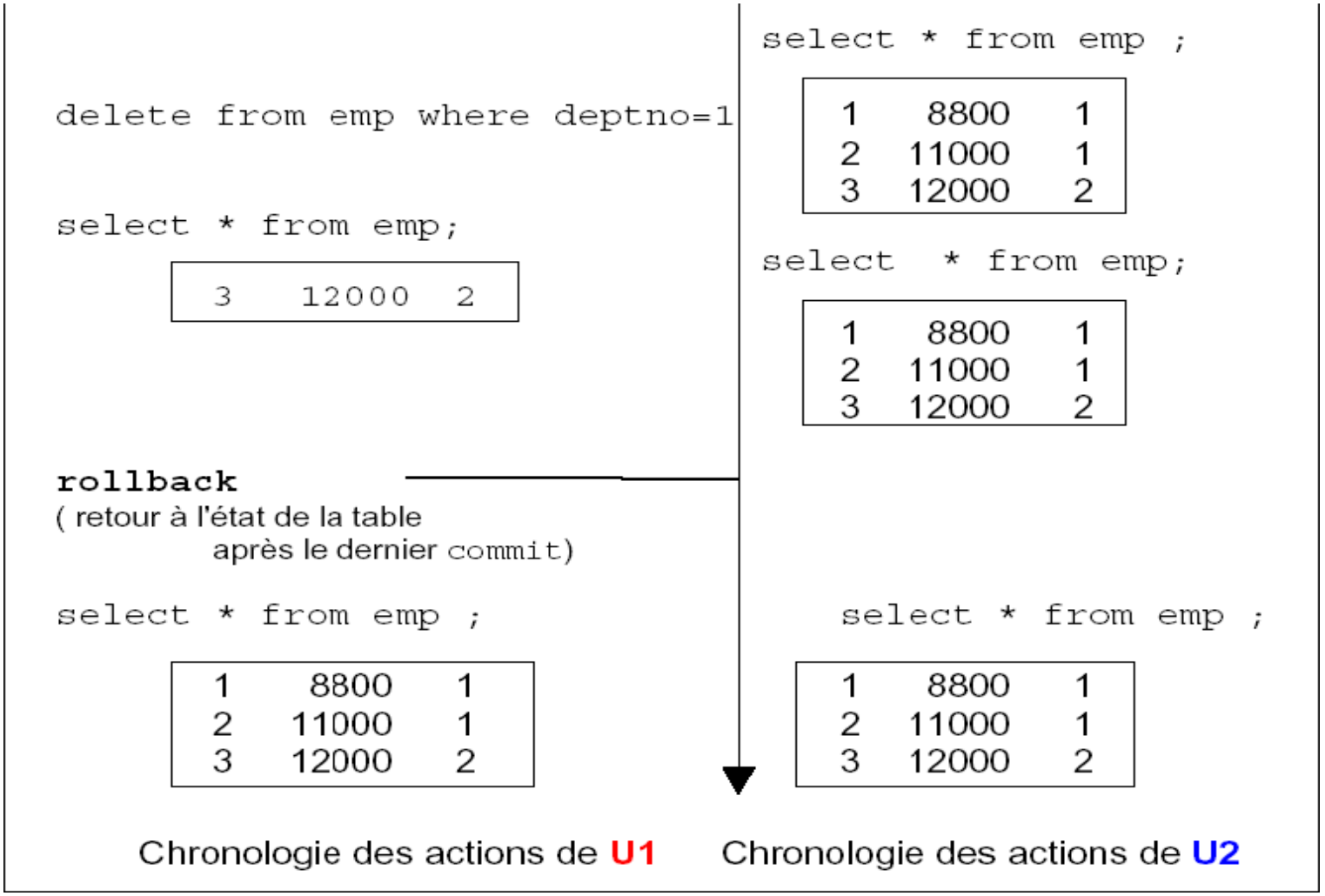
En pratique

- Les transactions sont délimitées par des primitives COMMIT et ROLLBACK ou la fermeture/ouverture de session
- Points de sauvegarde possibles avec SAVEPOINT
- Les commandes CREATE, DROP, ALTER, GRANT, … provoquent un COMMIT automatique

Déroulement d'une transaction



Déroulement d'une transaction



Contributeurs

- Agnès Braud
- Julien Schnell