Systèmes de Gestion de Bases de Données

Chapitre

Contraintes d'intégrité et Confidentialité

Nacer Boudjlida

Confidentialité et Contraintes d'intégrité

- *Sécurité, protection, confidentialité* : empêcher l'accés à une base ou à une de ses parties par des personnes non autorisées
- *Intégrité* : maintenir la base dans un état cohérent vis-à-vis de contraintes données
- *Similitudes* :
  - Spécification de contraintes
  - Conservation (dictionnaire)
  - Préservation (SGBD ++)
- *Contenu du chapitre* :
  1. Confidentialité et administration
  2. Contraintes d'intégrité

Confidentialité et Protection de données

- *Aspects* : légaux, éthiques, techniques
- Assurées par un *sous-système d'autorisation*
- *Techniques* :
  - Numéros de compte, mots de passe
  - Allocation/Révocation de droits
  - Encryptage/Décryptage
  - "ad hoc" : BdD Statistiques

Sécurité et Administration

- *Utilisateur* : Numéro de compte, Mot de passe
- *Audit trail* : Fichier "espion", trace, par numéro de compte, de toute action effectuée ou tentée
- *Administrateur* : Utilisateur privilégié
  - Gestion des utilisateurs, de leurs droits
  - Gestion des espaces
  - Sécurité, performances, etc
- *Types de Droits* :
  - Compte : types de commandes/compte
  - "Objet" : relation, vue, procédure, etc.

Privilèges de niveau compte

- Création d'objets (*CREATE TABLE, VIEW, PROC, . . .*)
- Suppression d'objets (*DROP TABLE, VIEW, etc.*)
- Modification de schéma (*ALTER*)
- Consultation (*SELECT*), Insertion (*INSERT*)
- Modification (*UPDATE*), Suppression (*DELETE*)

Privilèges de niveau objet d'une base

- Contrôler accès, utilisation des objets
- Spécifier les types de commandes autorisées
- Compatibilté avec ceux de niveau compte
- Alloués par le *propriétaire*
- *Rappel* : Nommage des objets ([NomPropriétaire.]NomObjet)
- *Types* : *SELECT*, *UPDATE*, *ALTER*, *EXECUTE*, etc.
- *Privilèges parfois transmissibles* (Option GRANT)

Notion de rôle

- *Rôle* : Ensemble nommé de privilèges (assignable à un utilisateur ou à un rôle)
- Création, allocation, suppression de rôles :

Oracle : Gestion des utilisateurs

- Versions < 7 : gestion des utilisateurs et des privilèges : *grant* et *revoke* ;
- Versions ≥ 7 :
  - Notion de rôle ;
  - Commandes explicites de gestion des utilisateurs.
- La suite :
  1. Gestion des comptes et des utilisateurs ;
  2. Ressources et profils ;
  3. Gestion des rôles ;
  4. Gestion des privilèges.

2.1. Gestion des comptes et des utilisateurs

- Six types d'utilisateurs Oracle :
  1. Administrateurs (rôle *DBA*),
  2. Responsables sécurité (rôle *OPER*),
  3. Développeurs d'applications,
  4. Administrateurs d'applications,
  5. Utilisateurs d'une base,
  6. Administrateurs réseau.

2.1.1- Les administrateurs

- AU moins deux comptes, par défaut, créés lors de la création d'une base :
  1. *sys*/*change_on_install*
  2. *system*/*manager*.
- *dba* : rôle prédéfini
  - Créé lors de la création d'une base ;
  - Octroyable à d'autres utilisateurs par *sys* ou *system*.
- Authentification des utilisateurs bénéficiant de ces rôles :
  - Fichier de mots de passe ou
  - Système d'exploitation.

2.1.1- Les administrateurs : Exemple

- [redacted]
- Exemple :

2.1.2- Les utilisateurs d'une base

- *licence_max_sessions* (*init.ora*) : Nombre maximum de sessions simultanées
- *licence_max_users* : Nombre maximum d'utilisateurs qu'on peut créer dans une base ;
- Création utilisateur :
  - dans la base,
  - droit administrateur ou *create user*,
  - Nom, mot de passe, mode d'identification
- Autorisation de connexion : privilège *"connect"* ou *"create session"*
- Modification des caractéristiques d'un utilisateur : *"alter user"*

2.1.2- Les utilisateurs d'une base

- [redacted]
- Affectation d'espace par défaut :
  - *create user* ou *alter user* ;
  - *default tablespace* ;
  - *temporary tablespace*
- Suppression : *drop user*
- *drop user. . . "cascade"* : Suppression de l'utilisateur et de ses objets

2.1.2- Les utilisateurs d'une base : Exemples

2. Oracle : Gestion des utilisateurs

- La suite :
  1. Comptes et utilisateurs ;
  2. Ressources et profils ;
  3. Gestion des rôles ;
  4. Gestion des privilèges.

2.2. Limitations de ressources et notion de profil

- *Profil* : ensemble nommé de limites de ressources,
- Exemples de ressources :
  - Nombre de sessions (*sessions_per_user*),
  - Temps unité centrale (*cpu_per_session*, *cpu_per_call*),
  - Temps de connexion (*connect_time*) par session,
  - Nombre de lectures logiques (*logical_reads_per_session* ou *per_call*).
- Autres éléments possibles d'un *profil* :
  - durée de vie d'un mot de passe (*password_life_time*)
  - Nombre de tentatives de connexion infructueuses au delà duquel le compte est verrouillé (*failed_login_attempts*).

2.2. Limitations de ressources et notion de profil

- Permettre la limitation de ressources :
  - *resource_limit = true* (*init.ora*)
  - ou pendant la session :
    alter system set resource_limit = true
- *Profil par défaut* :
  - Dans chaque base,
  - Ressources illimitées (*unlimited*),
  - Hérité par tout utilisateur sans profil explicite.

  $\Longrightarrow$ définir des profils avec les "bonnes limites".

2.2. Limitations de ressources et notion de profil

- Opérations sur les profils :
  - Créer, modifier, supprimer : *create, alter, drop profile* ;
  - *Droits* : administrateur ou *create, alter, drop profile*.
- Affectation de profils :
  - {*create, alter*} *user . . . profile. . .*
  - Limites du profil remplacent celles du profil par défaut ;
  - Limites non spécifiées par le profil gardent leurs valeurs par défaut!
- [redacted]

2.2. Notion de profil : Exemple

1. Définition,
2. Restauration valeur par défaut,
3. Affectation à un utilisateur.

2. Oracle : Gestion des utilisateurs

- La suite :
  1. Comptes et utilisateurs ;
  2. Ressources et profils ;
  3. Gestion des rôles ;
  4. Gestion des privilèges.

2.3. Gestion des rôles

- Rôles prédéfinis : *sysdba* (*connect, dba, resource* : compatibilité versions $<$ V8)
- Autres rôles prédéfinis installables (Scripts fournis) :
  - Rôles *exp_full_database* et *imp_full_database* : Import/Export données
  - Script *catexp.sql*
- Création de rôles (*create role*) avec/sans mot de passe
- Affectation à des rôles et/ou des utilisateurs (*grant role*).

2.3. Gestion des rôles : Exemples

2.3. Gestion des rôles

- Affectation rôle par défaut : *alter user . . . default role . . .*
- A la connexion, privilèges = privilèges octroyés $\cup$ ceux du rôle par défaut
- Bénéficier des privilèges d'un rôle : *set role*
- Nombre maximum de rôles endossables pendant une session $\leq$ *max_enabled_roles* (*init.ora*).

2. Oracle : Gestion des utilisateurs

- La suite :
  1. Comptes et utilisateurs ;
  2. Ressources et profils ;
  3. Gestion des rôles ;
  4. Gestion des privilèges.

2.4. Gestion des privilèges sous Oracle

- Types de privilèges :
  1. Objets
  2. Système

2.4.1- Les privilèges sur des objets

- Qui : propriétaire objet
- Quoi :
  - *select*, *update, delete, insert* : vues ou relations,
  - Utiliser une relation dans une contrainte (*references*),
  - Exécuter (*execute*) des procédures, des fonctions ou des paquetages.
  - privilège *update* sur certaines colonnes.
- Droit de propager des privilèges acquis : *grant . . . with admin option*
- Révocation en cascade : *revoke . . . cascade constraints*.

2.4.1- Privilèges sur des objets : Exemple

2.4.2- Privilèges système

- Une centaine ! (cf. Manuel de référence)
- Créer/supprimer/modifier objets système ou base de données (session, user, profils, rôles, index, tables, vues, triggers, etc.)
- Agir sur :
  - Base : *alter database*
  - Système : *alter system*
  - *Audit* : *audit system*
- Egalement transmissibles.

2.4.2- Privilèges système : Exemple

2.4. Gestion des utilisateurs : Eléments du dictionnaire

- *all_users*, *user_users*, *dba_users*,
- *user_ts_quotas*, *dba_ts_quotas*,
- *user_password_limits*, *user_resource_limits*,
- *dba_profiles*,
- *v$session*, *v$sesstat*, *session_roles*,
- *session_privs*,
- *all_tab_privs_recd*, *user_tab_privs_recd* (privilèges reçus),
- *all* et *user_tab_privs_made* (privilèges octroyés),
- *all* et *user_col_privs_recd* (ou *made*),
- *user* et *dba_role_privs*, . . .

2.4. Gestion des utilisateurs : Eléments du dictionnaire

- *"select * from session_privs"* : Privilèges en vigueur pour la session courante ;
- *select table_name, grantee, grantor, privilege from user_tab_privs_made*
  - noms des objets (*table_name*),
  - nom du bénéficiaire (*grantee*),
  - désignation du privilège (*privilege*)
  - nom de l'utilisateur qui l'a octroyé (*grantor*).

Les Contraintes d'Intégrité (CI)

- Schéma : description des données ET des contraintes
- *Contraintes d'Intégrité* : Assertions devant être vraies durant la vie de la base ou en des instants déterminés
- *Cohérence d'une base* : satisfaction de ses contraintes
- *Structure de la présentation* :
  1. Typologie des contraintes
  2. Spécification et stratégies d'implantation

Une typologie des contraintes d'intégrité

1. *CI Individuelles* : sur un type de donnée
   - Domaine (entier, réel, intervalle ou liste de valeurs, ...),
   - Valeur obligatoire (NOT NULL), etc
2. *CI Intra-relation* : sur un ensemble de tuples ou sur les valeurs des attributs d'un tuple au sein d'une relation
   - Unicité de valeur, clés,
   - Cardinalité : *dépôt numéro 3 ne stocke que les produits P1, P2 et P3*
   - Dépendances : *Produits P4 et P5 ne doivent pas être stockés dans le même dépôt*

Typologie des CI (suite)

3. *CI Inter-relations* : dépendances référentielles (existentielles ou d'inclusion)
   $\Pi_{No\_Prod}(stock) \subseteq \Pi_{No\_Prod}(Produit)$
4. *Contraintes Dynamiques* : Valeurs dans des états différents
   - Pas de diminution de salaires
   - Situation matrimoniale : pas de retour à célibataire
5. *Contraintes Temporelles* : commandes de la semaine courante à traiter avant le premier jour ouvrable de la semaine suivante.

CI : les Problèmes

1. *Spécification* : parfois dans le LDD
2. *Implantation* : SGBD ou programmation
3. *En cas de violation* ?
   - Annuler des actions (cf. gestion des transactions)
   - Rétablissement par inférence
4. *Cohérence* : Non contradiction (techniques de preuve)

CI : Spécification

1. *Spécification procédurale* : au sein des programmes
2. *Spécification déclarative* : Langage de type CP1
   - Stockage dans le dictionnaire
   - Préservation : *Sous-système de contrôle de l'intégrité*
   - Exemple (System-R, 1976) :

```
ASSERT contrainte_sur_salaire
ON personne p c , equipe e :
            p.salaire < c.salaire
     AND    p.#equ = e.#equ
     AND    e.#resp_equ = c.#id
```

CI : Spécification

3. *Utilisation des triggers*
   - Partie événement déclencheur
   - Partie Condition
   - Partie Action

   - Exemple (System-R, 1976) :

```
DEFINE TRIGGER contrainte_sur_salaire
ON personne p c , equipe e :
            p.salaire > c.salaire
      AND   p.#equ = e.#equ
      AND   e.#resp_equ = c.#id
ACTION Crime-de-Lese-Majeste(p.#id, c.#id)
```


CI : Exemples

1. *CI exprimables dans le LDD* (Create table)

   - Unicité : UNIQUE, PRIMARY KEY
   - Inclusion : REFERENCES
   - Valeur Obligatoire : [NOT] NULL
   - Valeur par Défaut : DEFAULT *Expression_Constante*
   - Autres : CHECK *(Expression logique)*
   - Remarques :

   (a) Contraintes nommées
   (b) Modification pendant la durée de vie de la base
   (c) "Partage" de contraintes
   (d) Activation/Désactivation (ENABLE/DISABLE CONSTRAINT)

CI et LDD : Exemples

```
CREATE TABLE produit
     (prod# int PRIMARY KEY CONSTRAINT C_NoProd,
      libelle VARCHAR(30)
     )
CREATE TABLE stock
     (prod# int,
      dep# int CHECK (dep# BETWEEN 22 AND 500)
      qte int DEFAULT 0 CONSTRAINT C_qte
      PRIMARY KEY (prod#, dep#),
      FOREIGN KEY (prod#) REFERENCES produit(prod#)
      CHECK (qte >= 0) CONSTRAINT C_stock)
```


CI et LDD : Exemples (suite)

```
CREATE TABLE depot
     (no_dep int NOT NULL,
      adr_dep VARCHAR(50),
      capacite int),
      PRIMARY KEY no_dep CONSTRAINT C_NoDep
      CHECK (capacite > 0))
```


CI Sybase : Triggers

2. *Déclencheurs ou Triggers*
   - Evénement : *insert, update, delete*

   - Objet : *une* relation

   - Action : Programme Transact-SQL (≃ PL/SQL Oracle) sur relations de base et *relations logiques*
     - *inserted* : tuples modifiés (update) ou insérés
     - *deleted* : tuples avant modification (update) ou supprimés

   - Un trigger au plus par type d'événement par relation

CI Sybase : Triggers (suite)

(a) *Création* : propriétaire de relation ou ayant droit

```
CREATE TRIGGER delprod
ON produit FOR DELETE
AS  IF ( SELECT COUNT(*) FROM deleted, stock
            WHERE deleted.#prod = stock.#prod) > 0
    BEGIN
        ROLLBACK TRAN
        PRINT "– Produit existe en stock: Suppression refusee"
    END
    ELSE PRINT "– Suppression effectuee"
```

(b) *Suppression* : DROP TRIGGER delprod

Triggers sous Oracle

- Trigger $\simeq$ Sorte de règle ECA :

  Si l'événement E survient
  Alors Si la condition C est vraie
  Alors exécuter l'action A

- **Définition d'un trigger sous ORACLE :** CREATE OR REPLACE TRIGGER

Triggers sous Oracle : Composants

1.
2.
3.
4.
5.
6. .
   -
   -

Triggers Oracle

- Exemple :

  AFTER UPDATE OF qte_en_stock ON Inventaire
  — Evénement (UPDATE), Instant (AFTER) et Objet (Inventaire)
  WHEN (new.qte_en_stock < new.seuil) — Condition
  FOR EACH ROW — Nombre de déclenchements
  <Bloc PL/SQL> — Action

- Action d'un trigger v.s. Procédure ou Fonction
  - Procédure ou Fonction : sur invocation explicite
  - Trigger : sur arrivée de l'événement et si condition satisfaite

Triggers : Cas d'utilisation (exemples)

1. Générer des valeurs dérivées (calculées)
2. Empêcher l'exécution de transactions (programmes) non valides
3. Implanter des contraintes d'intégrité dynamiques
4. Implanter des fonctions de surveillance (audit)
5. Synchroniser des tables répliquées
6. Implanter des contraintes référentielles dans une base de données distribuées
7. etc.

Triggers Oracle : Types d'événements

1.
2.
3.

- La partie action peut :
  1. Contenir du SQL, PL/SQL, Java, C
  2. Définir des "objets" PL/SQL (Variables, constantes, curseurs, etc.)
  3. Appeler des procédures stockées

Triggers Oracle

- **Notes :**
  1. Corrélation (ancienne/nouvelle valeur)
     NEW.NomDeColonne, **OLD**.NomDeColonne
  2. Renommage de **NEW** et de **OLD** : cf. clause **REFERENCING OLD/NEW AS** dans CREATE TRIGGER
  3. "Cascade" de triggers (et terminaison)
- Typologie des triggers (et plan de la suite) :
  1. Sur instruction, sur ligne (FOR EACH)
  2. Avant (BEFORE)/Après (AFTER) exécution de l'événement déclenchant
  3. "Au lieu de" l'instruction déclenchante (INSTEAD OF)
  4. Sur événement système ou utilisateur

Typologie des triggers : 1) Sur instruction, sur ligne

- Lors de la déclaration du trigger, indication du nombre de fois où il doit être exécuté
  1. Autant de fois que de lignes concernées (FOR EACH ROW)
  2. Une seule fois (par défaut)

Typologie des triggers : 1.1) Trigger sur ligne

- Déclenché chaque fois qu'une table est concernée
- Exemple : update . . . where . . .
- Cas d'utilisation : Quand l'action du trigger **dépend** de la valeur des données de l'instruction déclenchante ou de celle des tuples concernés

Typologie des triggers : 1.2) Sur instruction

- Activé **une seule fois**, quel que soit le nombre de tuples concernés
- Cas d'utilisation : Quand l'action du trigger **ne dépend pas** de la valeur des données de l'instruction déclenchante ou de celle des tuples concernés
- Exemples d'usage :
  - Surveillance (*auditing*)
  - Tests complexes sur le temps ou sur un utilisateur

Typologie des triggers : 2) AVANT/APRES

- S'appliquent pour les deux types précédents
- Si trigger sur instruction du LMD :
  - S'applique sur des tables, **pas sur des vues**
- Si trigger sur instruction du LMD :
  - S'applique sur la base ou sur un schéma

Typologie des triggers : 2) AVANT/APRES

1. Trigger AVANT
   - Exécuté avant l'exécution de l'instruction (update, delete)
   - Exemple 1 : Vérifier que l'instruction est autorisée
   - Exemple 2 : Dériver des valeurs de colonnes avant modification
2. Trigger APRES : Exécuté après l'exécution de l'instruction

Triggers Sur instruction, Sur ligne : Conclusion

- 4 compositions possibles
- Possiblité de définir **plusieurs triggers d'un même type** pour toute instruction du LMD (insert, update, delete)
- 4 combinaisons possibles :

1. **Trigger de type AVANT instruction** :
   - Attention : Lignes verrouillées (cf. accès concurrents)
2. **Trigger de type AVANT chaque ligne** :

   Si la condition gardant le trigger est vraie
   alors exécuter l'action du trigger AVANT modification des lignes
   ET AVANT vérification des contraintes d'intégrité

Triggers Sur instruction, Sur ligne : Conclusion

3. **Trigger de type APRES instruction** :
   Exécuter l'action du trigger APRES modification des lignes
   ET APRES vérification des contraintes d'intégrité
4. **Trigger de type APRES chaque ligne** :
   Même modèle d'exécution qu'un trigger AVANT
   mais réalisé APRES l'instruction ET pour chaque ligne

Typologie des triggers : 3) De substitution (INSTEAD OF)

- Exécution de l'action du trigger à la place de l'instruction
- Cas d'utilisation : Mise à jour **"à travers une vue"**
  - Pour mettre à jour les tables sous-jacentes
  - **Quand la mise à jour à travers la vue** n'est pas possible
- Vues non modifiables : Leur définition comporte :
  - des opérateurs ensemblistes, des jointures
  - des fonctions d'agrégation (somme, moyenne, etc.)
  - GROUP BY, START WITH, CONNECT, clause DISTINCT
- Vues modifiables
  - Celles non modifiables
  - **et quand** la mise à jour n'est pas ambigüe

Typologie des triggers : 4) Sur événement système ou utilisateur

- Pour publication d'information sur la base à l'intention "d'abonnés"
- Abonnement d'applications à des événements sur la base (package *DBMS_AQ*)
- Evénements Système : sur base ou schéma
  1. Lancement/Arrêt (startup/shutdown)
  2. Guardes sur les transitions de rôle
  3. Messages d'erreurs du serveur de données

Typologie : 4) Sur événement système ou utilisateur

- Evénements Utilisateurs :
  1. Connexion/Déconnexion (logon/logoff) : sur base ou schéma
  2. Instructions du LDD (CREATE, ALTER, DROP) : sur base ou schéma
  3. Instructions du LMD (INSERT, UPDATE, DELETE) : sur table ou vue
- Publication/Abonnement (*publish/subscribe*) à des événements : Mécanismes de *Oracle Streams Advanced Queuing*. Voir :
  - *Oracle Streams Advanced Queuing User's Guide and Reference*
  - *Oracle Database PL/SQL Packages and Types Reference*

Typologie : 4) Sur événement système ou utilisateur

- Exemple :

```
CREATE TRIGGER Si_Arret
  ON DATABASE SHUTDOWN
  BEGIN
    ...
    DBMS_AQ.ENQUEUE(...);
    ...
  END;
```


Triggers : Modèles d'exécution

1. Cas triggers multiples sur la même instruction : dans l'ordre,
   (a) Triggers de niveaux instruction
   (b) Triggers de niveaux ligne
2. Cas triggers multiples d'un même type : Ordre aléatoire
3. Triggers et Test de contraintes : une instruction SQL peut déclencher 4 types de triggers
   (a) Before row
   (b) Before instruction
   (c) After row
   (d) After instruction
   - L'instruction déclenchante et l'action du trigger peuvent mettre en cause des contraintes d'intégrité : quel modèle d'exécution?

Triggers et CI : Modèles d'exécution

1.
2. .
   (a)
   (b)
   (c)
   (d)
   (e)
3.
4.

Triggers et CI : Modèles d'exécution

- **Modèle d'exécution récursif** :
- Exemple de récursion : Scenario
  1. Instruction déclenche un trigger "BEFORE ROW" **T-before-R** + Test CI
  2. **T-before-R** fait une mise à jour qui :
     (a) Nécessite un test de CI
     (b) Déclenche un trigger "APRES instruction" **T-after-I**

Triggers et CI : Exemple d'exécution récursive

Triggers : Remarques

1. Privilèges requis pour la création :
   (a) Sur des objets possédés : CREATE TRIGGER
   (b) Sur ceux non possédés : CREATE ANY TRIGGER, ALTER ANY TABLE
   (c) Sur une base : ADMINISTER DATABASE TRIGGER
2. Autoriser (ENABLE)/Inhiber (DISABLE) le déclenchement :
   - cf. ALTER TABLE . . . ENABLE/DISABLE ALL TRIGGERS ou ALTER TRIGGER . . . ENABLE/DISABLE
3. Accès concurrents aux données : Mêmes règles de verrouillage que pour une transaction

Triggers : Remarques

4. Atomicité : Tout ou rien, effets de l'instruction déclenchante compris
5. Exécution : $\simeq$ procédures stockées
6. Stockage : Sous forme PseudoCode compilé (Pas de stockage du source)
7. Information : cf. USER_TRIGGERS, ALL_TRIGGERS et DBA_TRIGGERS dans le dictionnaire
8. Cas où un trigger est gardé par plusieurs instructions : prédicats INSERTING, DELETING et UPDATING

Triggers : Remarques

Triggers : Remarques

9. Cas où un UPDATE ne concerne que certaines colonnes :

Triggers : Remarques

10. Arrêt de l'exécution d'un trigger : Lever une erreur
    - Procédure *raise_application_error* *(No-Erreur, Message-Erreur)* :
      - $No - Erreur \in [-20999.. -20000]$
      - Message-Erreur : chaîne d'au plus 500 caractères
    (a) Arrêt de l'action du trigger
    (b) Annulation des effets de la transaction
    (c) Emission du No Erreur et du message vers l'application
11. Triggers et Exceptions :
    - Si exception non "programmée" : Annulation des effets du trigger et de ceux de l'instruction déclenchante
    - Exceptions "programmées" : cf. PL/SQL

Triggers : Remarques

- Quelques exceptions prédéfinies (Rappels) :
  NO_DATA_FOUND : levée quand un SELECT INTO ne retourne pas de lignes
  DUP_VAL_ON_INDEX : tentative d'insertion d'une ligne "en double" dans un index UNIQUE
  ZERO_DIVIDE : division par zéro

Exemples

Triggers Oracle : Conclusions

- Utiliser des triggers pour garantir que, lorsqu'une opération spécifique est effectuée, les actions liées ont aussi réalisées.
- N'utiliser des triggers que s'il n'y a pas d'autre possibilité ((exemple des contraintes déclaratives).
- Eviter des triggers récursifs (un trigger AFTER UPDATE faisant un UPDATE sur la même table).

CI et Triggers : les problèmes

1. *Déclenchement* : Immédiat, différé, etc
   - Sybase : Immédiat
   - Oracle : Avant/Après/Pour chaque tuple
2. *Propagation et terminaison* (Cascading)
   - Oracle : à la charge de l'utilisateur
   - Sybase :
     - Paramètre de configuration (profondeur 16, sa)
       sp_configure "allow nested triggers", 0 | 1
     - Récursion : Action d'un trigger T déclenche T
       set self_recursion ON | OFF
3. *Atomicité* : cf. transactions imbriquées

Conclusion

- *Protection, Confidentialité* : Vues, privilèges
- *Contrainte d'Intégrité* :
  - Pas de typologie universelle
  - Utilisables pour :
    1. Qualité de l'information
    2. Déduction (BdD Déductives)
    3. Optimisation sémantique
- *Problèmes non résolus* (BdD Actives) :
  - Evénements "user-defined"
  - Evénements composites
  - Terminaison, etc.