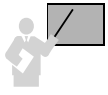
Intégrité référentielle

Les contraintes référentielles forment le cœur de la cohérence d'une base de données relationnelle. Ces contraintes sont fondées sur une relation entre clés étrangères et clés primaires et permettent de programmer des règles de gestion (exemple : l'affrètement d'un avion doit se faire par une compagnie existant dans la base de données). Ce faisant, les contrôles côté client (interface) sont ainsi déportés côté serveur.

C'est seulement dans sa version 7 en 1992, qu'Oracle a inclus dans son offre les contraintes référentielles.

Pour les règles de gestion trop complexes (exemple : l'affrètement d'un avion doit se faire par une compagnie qui a embauché au moins quinze pilotes dans les six derniers mois), il faudra programmer un déclencheur (voir le chapitre 7). Il faut savoir que les déclencheurs sont plus pénalisants que des contraintes dans un mode transactionnel (lectures consistantes).

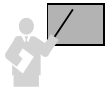


La contrainte référentielle concerne toujours deux tables – une table « père » aussi dite « maître » (*parent/referenced*) et une table « fils » (*child/dependent*) – possédant une ou plusieurs colonnes en commun. Pour la table « père », ces colonnes composent la clé primaire (ou candidate avec un index unique). Pour la table « fils », ces colonnes composent une clé étrangère.

Il est recommandé de créer un index par clé étrangère (Oracle ne le fait pas comme pour les clés primaires). La seule exception concerne les tables « pères » possédant des clés primaires (ou candidates) jamais modifiées ni supprimées dans le temps.

Cohérences

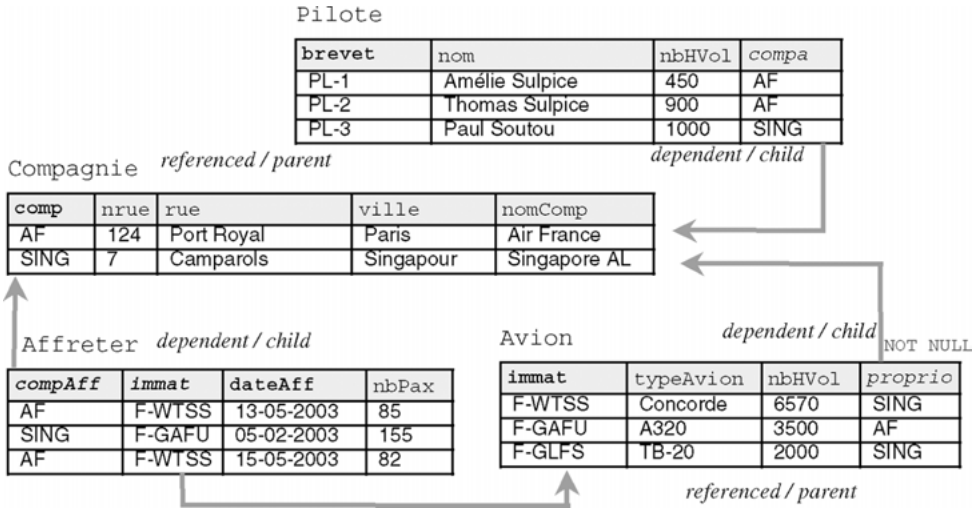
L'exemple suivant illustre quatre contraintes référentielles. Une table peut être « père » pour une contrainte et « fils » pour une autre (c'est le cas de la table `Avion`).



Deux types de problèmes sont automatiquement résolus par Oracle pour assurer l'intégrité référentielle :

- La cohérence du « fils » vers le « père » : on ne doit pas pouvoir insérer un enregistrement « fils » (ou modifier sa clé étrangère) rattaché à un enregistrement « père » inexistant. Il est cependant possible d'insérer un « fils » (ou de modifier sa clé étrangère) sans rattacher d'enregistrement « père » à la condition qu'il n'existe pas de contrainte `NOT NULL` au niveau de la clé étrangère.
- La cohérence du « père » vers le « fils » : on ne doit pas pouvoir supprimer un enregistrement « père » (ou modifier sa clé primaire) si un enregistrement « fils » y est encore rattaché. Il est possible de supprimer les « fils » associés (`DELETE CASCADE`) ou d'affecter la valeur nulle aux clés étrangères des « fils » associés (`DELETE SET NULL`). Oracle ne permet pas de propager une valeur par défaut (*set to default*) comme la norme SQL2 le propose.

Figure 2-9 *Tables et contraintes référentielles*



Déclarons à présent ces contraintes sous SQL.

Contraintes côté « père »

La table « père » contient soit une contrainte de clé primaire soit une contrainte de clé candidate qui s'exprime par un index unique. Le tableau suivant illustre ces deux possibilités dans le cas de la table `Compagnie`. Notons que la table possédant une clé candidate aurait pu aussi contenir une clé primaire.

Tableau 2-14 Écritures des contraintes de la table « père »

Clé primaire	Clé candidate

`CREATE TABLE Compagnie`
`(comp CHAR(4), nrue NUMBER(3),`
`rue CHAR(20), ville CHAR(15),`
`nomComp CHAR(15),`
`CONSTRAINT pk_Compagnie`
`PRIMARY KEY(comp));`
	`CREATE TABLE Compagnie`
	`(comp CHAR(4), nrue NUMBER(3),`
	`rue CHAR(20), ville CHAR(15),`
	`nomComp CHAR(15),`
	`CONSTRAINT un_Compagnie UNIQUE(comp));`

Contraintes côté « fils »

Indépendamment de l'écriture de la table « père », deux écritures sont possibles au niveau de la table « fils ». La première définit la contrainte en même temps que la colonne. Ainsi elle ne convient qu'aux clés composées d'une seule colonne. La deuxième écriture détermine la

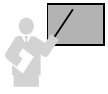
contrainte après la définition de la colonne. Cette écriture est préférable car elle convient aussi aux clés composées de plusieurs colonnes de par sa lisibilité.

Tableau 2-15 Écritures des contraintes de la table « fils »

Colonne et contrainte	Contrainte et colonne

`CREATE TABLE Pilote`
`(brevet CHAR(6) CONSTRAINT`
`pk_Pilote PRIMARY KEY,`
`nom CHAR(15), nbHVol NUMBER(7,2),`
`compa CHAR(4) CONSTRAINT fk_Pil_compa_Comp`
`REFERENCES Compagnie(comp));`
	`CREATE TABLE Pilote`
	`(brevet CHAR(6), nom CHAR(15),`
	`nbHVol NUMBER(7,2), compa CHAR(4),`
	`CONSTRAINT pk_Pilote`
	`PRIMARY KEY(brevet),`
	`CONSTRAINT fk_Pil_compa_Comp`
	`FOREIGN KEY(compa)`
	`REFERENCES Compagnie(comp));`

Clés composites et nulles



Les clés étrangères ou primaires peuvent être définies sur trente-deux colonnes au maximum (*composite keys*).

Les clés étrangères peuvent être nulles si aucune contrainte `NOT NULL` n'est déclarée.

Décrivons à présent le script SQL qui convient à notre exemple (la syntaxe de création des deux premières tables a été discutée plus haut) et étudions ensuite les mécanismes programmés par ces contraintes.

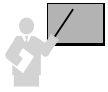
```
CREATE TABLE Compagnie …

CREATE TABLE Pilote …

CREATE TABLE Avion
   (immat CHAR(6), typeAvion CHAR(15), nbhVol NUMBER(10,2),
   proprio CHAR(4),
   CONSTRAINT pk_Avion PRIMARY KEY(immat),
   CONSTRAINT nn_proprio CHECK (proprio IS NOT NULL),
   CONSTRAINT fk_Avion_comp_Compag FOREIGN KEY(proprio)
      REFERENCES Compagnie(comp));

CREATE TABLE Affreter
   (compAff CHAR(4), immat CHAR(6), dateAff DATE, nbPax NUMBER(3),
   CONSTRAINT pk_Affreter PRIMARY KEY (compAff, immat, dateAff),
   CONSTRAINT fk_Aff_na_Avion FOREIGN KEY(immat) REFERENCES
   Avion(immat),
   CONSTRAINT fk_Aff_comp_Compag FOREIGN KEY(compAff)
      REFERENCES Compagnie(comp));
```

Cohérence du fils vers le père



Si la clé étrangère est déclarée `NOT NULL`, l'insertion d'un enregistrement « fils » n'est possible que s'il est rattaché à un enregistrement « père » existant. Dans le cas inverse, l'insertion d'un enregistrement « fils » rattaché à aucun « père » est possible.

Le tableau suivant décrit des insertions correctes et une insertion incorrecte. Le message d'erreur est ici en anglais (en français : `violation de contrainte d'intégrité - touche parent introuvable`).

Tableau 2-16 Insertions correctes et incorrectes

Insertions correctes	Insertion incorrecte

`-- fils avec père`
`INSERT INTO Pilote VALUES`
`('PL-3', 'Paul Soutou', 1000, 'SING');`
`-- fils sans père`
`INSERT INTO Pilote VALUES`
`('PL-4', 'Un Connu', 0, NULL);`
`-- fils avec pères`
`INSERT INTO Avion VALUES`
`('F-WTSS', 'Concorde', 6570, 'SING');`
`INSERT INTO Affreter VALUES`
`('AF', 'F-WTSS', '15-05-2003', 82)`
	`-- avec père inconnu`
	`INSERT INTO Pilote VALUES`
	`('PL-5', 'Pb de Compagnie', 0, '?');`
	
	`ORA-02291: integrity constraint`
	`(SOUTOU.FK_PIL_COMPA_COMP) violated -`
	`parent key not found`

Pour insérer un affrètement, il faut donc avoir ajouté au préalable au moins une compagnie et un avion.

Le chargement de la base de données est conditionné par la hiérarchie des contraintes référentielles. Ici, il faut insérer d'abord les compagnies, puis les pilotes (ou les avions), enfin les affrètements.

Il suffit de relire le script de création de vos tables pour en déduire l'ordre d'insertion des enregistrements.

Cohérence du père vers le fils

Trois alternatives sont possibles pour assurer la cohérence de la table « père » vers la table « fils » via une clé étrangère :

- Prévenir la modification ou la suppression d'une clé primaire (ou candidate) de la table « père ». Cette alternative est celle par défaut. Dans notre exemple, toutes les clés étrangères sont ainsi composées. La suppression d'un avion n'est donc pas possible si ce dernier est référencé dans un affrètement.

- Propager la suppression des enregistrements « fils » associés à l'enregistrement « père » supprimé. Ce mécanisme est réalisé par la directive `ON DELETE CASCADE`. Dans notre exemple, nous pourrions ainsi décider de supprimer tous les affrètements dès qu'on retire un avion.
- Propager l'affectation de la valeur nulle aux clés étrangères des enregistrements « fils » associés à l'enregistrement « père » supprimé. Ce mécanisme est réalisé par la directive `ON DELETE SET NULL`. Il ne faut pas de contrainte `NOT NULL` sur la clé étrangère. Dans notre exemple, nous pourrions ainsi décider de mettre `NULL` dans la colonne `compa` de la table `Pilote` pour chaque pilote d'une compagnie supprimée. Nous ne pourrions pas appliquer ce mécanisme à la table `Affreter` qui dispose de contraintes `NOT NULL` sur ses clés étrangères (car composant la clé primaire).

Tableau 2-17 Cohérence du « père » vers le « fils »

Alternative	Exemple de syntaxe
Prévenir la modification ou la suppression d'une clé primaire	`CONSTRAINT fk_Aff_na_Avion FOREIGN KEY(immat) REFERENCES Avion(immat)`
Propager la suppression des enregistrements	`CONSTRAINT fk_Aff_na_Avion FOREIGN KEY(immat) REFERENCES Avion(immat) ON DELETE CASCADE`
Propager l'affectation de la valeur nulle aux clés étrangères	`CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY(compa) REFERENCES Compagnie(comp) ON DELETE SET NULL`

L'extension de la modification d'une clé primaire vers les tables référencées n'est pas automatique (il faut la programmer si nécessaire par un déclencheur).

En résumé

Le tableau suivant résume les conditions requises pour modifier l'état de la base de données en respectant l'intégrité référentielle.

Tableau 2-18 Instructions SQL sur les clés

Instruction	Table « parent »	Table « fils»
`INSERT`	Correcte si la clé primaire (ou candidate) est unique.	Correcte si la clé étrangère est référencée dans la table « père » ou est nulle (partiellement ou en totalité).
`UPDATE`	Correcte si l'instruction ne laisse pas d'enregistrements dans la table « fils » ayant une clé étrangère non référencée.	Correcte si la nouvelle clé étrangère référence un enregistrement « père » existant.
`DELETE`	Correcte si aucun enregistrement de la table « fils » ne référence le ou les enregistrements détruits.	Correcte sans condition.
`DELETE CASCADE`	Correcte sans condition.	Correcte sans condition.
`DELETE SET NULL`	Correcte sans condition.	Correcte sans condition.