Fiche récapitulative des différences et similitudes SQL
SQL Server / Oracle

Cette fiche a pour but de vous aider dans la transition de SQL Server vers Oracle. Tous deux sont des moteurs de bases de données relationnelles. Ils implémentent donc le SQL, qui est normé. Il subsiste, malgré tout, quelques disparités, dues notamment aux différences de fonctionnement de ces deux moteurs.

DDL

<table>
<tr><th colspan="2">SQLSERVER</th><th colspan="2">ORACLE</th></tr>
<tr><th>2008</th><th>2012</th><th>11g</th><th>12c</th></tr>
<tr><th colspan="4">Création de table</th></tr>
<tr><td colspan="4">CREATE TABLE nom_table
(nom_colonne TYPECOLONNE [DEFAULT valeur_par_defaut] [NULL|NOT NULL] CONSTRAINT contrainte de colonne…
[,nom_colonne…]
[,CONSTRAINT contrainte_de_table…]) ;

Sous Oracle, NULL|NOT NULL peut être une contrainte nommée (exemple : CONSTRAINT nn_table_col NOT NULL)</td></tr>
<tr><th colspan="4">Spécificités</th></tr>
<tr><td></td><td></td><td></td><td>Lors d’un choix de valeur par défaut pour une colonne, il est possible de spécifier que cette valeur doit être affectée à la colonne même si c’est la valeur NULL qui est choisie explicitement :
DEFAULT ON NULL valeur</td></tr>
<tr><th colspan="4">Les schémas</th></tr>
<tr><td colspan="2">Un schéma est un ensemble d’objets, permettant de regrouper logiquement des tables, et est aussi utilisé pour optimiser la sécurité d’accès. Si l’utilisateur ne précise pas de schéma lorsqu’il crée une table, cette dernière sera créée dans son schéma par défaut, en général dbo</td><td colspan="2">Un schéma représente l’ensemble des objets d’un utilisateur. La table clients créée par toto, par exemple, est créée dans le schéma toto et s’appelle toto.clients</td></tr>
</table>

Modification de structure de table

```
ALTER TABLE  nom_table
      ALTER COLUMN nom_colonne … ;
    | ADD nom_colonne … ;
    | ADD CONSTRAINT nom_contrainte … ;
    | DROP CONSTRAINT nom_contrainte ;
    | DROP COLUMN nom_colonne ;
    | [ENABLE | DISABLE] TRIGGER [ALL | nom_trigger];
```

```
ALTER TABLE  nom_table
      MODIFY (nom_colonne …) ;
    | ADD (nom_colonne …) ;
    | ADD CONSTRAINT nom_contrainte … ;
    | DROP CONSTRAINT nom_contrainte ;
    | DROP COLUMN nom_colonne ;
    | [ENABLE | DISABLE] ALL TRIGGERS;
```

Suppression de table

```
DROP TABLE nomtable1 [,nomtable2…] ;
```

```
DROP TABLE nom_table [CASCADE CONSTRAINTS] [PURGE];
```

CASCADE CONSTRAINTS : supprime également les contraintes d'intégrité référentielle qui font référence à la table.
PURGE : si on ne précise pas cette option, la table est placée dans la corbeille. Elle est donc éventuellement récupérable (voir avec le DBA !) grâce à l'instruction FLASHBACK TABLE …

LES VUES

Créer la vue :

```
CREATE VIEW nomvue [(nom des colonnes)] AS requête
[WITH CHECK OPTION];
```

Modifier une vue existante :

```
ALTER VIEW nomvue [(nom des colonnes)] AS
nouvellerequête [WITH CHECK OPTION] ;
```

Créer la vue ou la modifier si elle existe déjà :

```
CREATE [OR REPLACE] [FORCE] VIEW [(nom des
colonnes)] nomvue AS requête WITH CHECK OPTION;
```

FORCE : permet de forcer la création de la vue, même si la requête porte sur une table qui n'existe pas encore. La vue ne sera alors utilisable qu'après sa compilation : `ALTER VIEW nomvue COMPILE ;`

Types de données

SQLSERVER	ORACLE
Caractères	

`CHAR(n)`
Avec n de 1 à 8000
`VARCHAR(n|max)`
Avec n de 1 à 8000 (ou max pour 3^{31} caractères)
`NCHAR(n)`
Avec n de 1 à 4000
`NVARCHAR(n|max)`
Avec n de 1 à 4000 (ou max pour 3^{31} caractères)
	`CHAR(n)`
	Avec n de 1 à 2000
	`VARCHAR2(n)` ou son synonyme `VARCHAR(n)`
	Avec n de 1 à 4000 (possibilité d'aller jusqu'à 32767 en 12c)
	`NCHAR(n)`
	Avec n de 1 à 2000
	`NVARCHAR2(n)` ou son synonyme `NVARCHAR(n)`
	Avec n de 1 à 4000 (possibilité d'aller jusqu'à 32767 en 12c)

Numériques	

`DECIMAL(p,s)` ou `NUMERIC(p,s)`
P chiffres en tout dont s chiffres après la virgule.
Avec P entre 1 et 38 et s entre 1 et p.
`BIGINT`
Entier signé sur 8 octets
`INT ou INTEGER`
Entier signé sur 4 octets
`SMALLINT`
Entier signé sur 2 octets
`TINYINT`
Entier sur 1 octet, compris entre 0 et 255
	`NUMBER(p,s)`
	P chiffres en tout dont s chiffres après la virgule.
	Avec P entre 1 et 38 et s entre -84 et 127.
	
	On peut aussi utiliser les synonymes de `NUMBER` :
	`DECIMAL`, `NUMERIC`, `INTEGER`, `INT`….

dates	

`DATETIME`
Date et heure (8 octets)
`DATE`
Date seulement (4 octets)
`TIME`
Heure seulement (4 octets)
	`DATE`
	Date et heure

Spéciaux	

`BIT`
0 ou 1 (ou NULL)
`UNIQUEIDENTIFIER`
Identifiant unique si généré par le système (`NEWID()`)	

Auto incrémentation

<table>
<tr><th colspan="2">SQLSERVER</th><th colspan="2">ORACLE</th></tr>
<tr><th>2008</th><th>2012</th><th>11g</th><th>12c</th></tr>
<tr><td colspan="4">IDENTITY</td></tr>
<tr><td colspan="2">IDENTITY est une propriété qu'on peut affecter à une seule colonne par table, à condition que cette colonne soit de type entier :

…nomcolonne TYPEENTIER IDENTITY(valeur de départ, pas d'incrémentation)
Par défaut 1 et 1.</td><td></td><td>…nomcolonne TYPEENTIER GENERATED [ALWAYS|BY DEFAUT] AS IDENTITY(options)

Oracle utilise en réalité une séquence pour générer la valeur ! Les options sont donc les mêmes que celles de la séquence</td></tr>
<tr><td colspan="4">LES SEQUENCES</td></tr>
<tr><td></td><td>CREATE SEQUENCE nom AS TYPEENTIER START WITH n INCREMENT BY n… ;

Pour générer une nouvelle valeur :
NEXT VALUES FOR nomseq</td><td colspan="2">Une séquence est un objet de la base, indépendant de toute table, contrairement à IDENTITY qui est associé à une colonne.

CREATE SEQUENCE nom START WITH n INCREMENT BY n… ;
Pour générer une nouvelle valeur :
Nomseq.NEXTVAL
Pour connaitre la dernière valeur générée :
Nomseq.CURRVAL</td></tr>
</table>

Quelques fonctions

SQLSERVER	ORACLE
Caractères	
Passer une chaine en majuscules, en minuscules : `UPPER(chaine), LOWER(chaine)`	Passer une chaine en majuscules, en minuscules : `UPPER(chaine), LOWER(chaine)`

Supprimer les espaces à droite, à gauche dans une chaine :
`RTRIM(chaine), LTRIM(chaine)`
	Première lettre en majuscule et les autres en minuscules
	`INITCAP(chaine)`
	Supprimer les chaines spécifiées `c` (par défaut l'espace ' ') à droite, à gauche, les deux dans une chaine :
	`RTRIM(chaine[,c]), LTRIM(chaine[,c]), TRIM(chaine[,c])`

Renvoyer les n caractères les plus à droite, à gauche :
`RIGHT(chaine,n) , LEFT(chaine,n)`
Renvoyer une partie d'une chaine :
`SUBSTRING(chaine,taille [,position])`
Connaitre le nombre de caractères d'une chaine :
LEN(chaine)
	Renvoyer une partie d'une chaine :
	`SUBSTR(chaine,taille [,position])`
	Connaitre le nombre de caractères d'une chaine :
	`LENGTH(chaine)`

Dates	

Date du jour :
`GETDATE( )`
Ajouter n format à une date :
`DATEADD(format,n,ladate)`
Renvoyer la différence en format entre deux dates :
`DATEDIFF(format,datepart,datefin)`
Renvoyer une partie de date sous forme numérique :
`DATEPART(format, ladate)`
Renvoyer une partie de date sous forme textuelle :
`DATENAME(format, ladate)`
Renvoyer le jour, le mois, l'année d'une date :
`DAY(ladate), MONTH(ladate),YEAR(ladate)`
	Date du jour :
	`SYSDATE( )`
	Ajouter n mois à une date :
	`ADD_MONTHS(ladate,n)`
	Renvoyer la différence en mois entre deux dates :
	`MONTHS_BETWEEN(date1,date2)`
	Renvoyer une partie de date sous forme numérique :
	`EXTRACT(format FROM ladate)`
	Tronquer une date au début de l'élément fourni en tant que format :
	`TRUNC(ladate,format)`

<table>
<tr><th colspan="2">Conversions</th></tr>
<tr>
<td>Convertir une valeur dans le type spécifié :
CAST(exp as TYPEDONNEE)
Convertir une valeur dans le type spécifié, avec un format s'il s'agit d'une conversion vers une date ou une heure :
CONVERT(TYPEDONNEE,exp [,format])

Nouveautés sqlserver 2012 :
Renvoyer NULL si la conversion échoue :
TRY_CONVERT(TYPEDONNEE,exp [,format])
Transformer une chaine en précisant par exemple quel est le séparateur des milliers ou celui des décimales… :
PARSE(chaine AS TYPEDONNEE [USING localisation])</td>
<td>Convertir un numérique en caractères selon un format et éventuellement les conventions d'un pays :
TO_CHAR(n [,format [,conventionsPays]])
Convertir une date en caractères, selon un format :
TO_CHAR(ladate [,format])
Convertir une chaine en date, selon un format :
TO_DATE (chaine [, format])
Convertir une chaine en numérique
TO_NUMBER(chaine)</td>
</tr>
<tr><th colspan="2">Système</th></tr>
<tr>
<td>Renvoyer la première expression non NULL de la liste :
COALESCE (exp1,exp2…)
Renvoyer une valeur si l'expression est NULL :
ISNULL(exp, valeur)
Savoir si une expression est d'un format date valide, renvoie 1 dans ce cas :
ISDATE(exp)
Savoir si une expression est d'un format numérique valide, renvoie 1 dans ce cas :
ISNUMERIC(exp)</td>
<td>Renvoyer la première expression non NULL de la liste :
COALESCE (exp1,exp2…)
Renvoyer une valeur si l'expression est NULL :
NVL(exp, valeur)

Renvoyer un résultat dépendant de la valeur d'une colonne :
DECODE(colonne, val1,result1,val2,result2…)</td>
</tr>
<tr><th colspan="2">Operateur de concaténation</th></tr>
<tr><td>+</td><td>||</td></tr>
</table>

DIVERS

SQLSERVER	ORACLE
Requêtes sans clause « FROM »	
`SELECT GETDATE() ;`	`SELECT SYSDATE() FROM DUAL ;`
Pseudo colonnes	
	Deux colonnes existent dans chaque table, sans qu'on ait besoin de les créer : `ROWID` : l'identifiant de la ligne dans la base. `ROWNUM` : le numéro de la ligne dans la requête, avant le tri. `SELECT ROWID, ROWNUM FROM table ;`
Récupérer les n premières lignes	
`SELECT TOP n … ;`	`Select …. WHERE rownum <= n ;`
Sauvegarder le résultat d'une requête	
`SELECT … INTO nomtable … ;`	`CREATE TABLE nomtable … AS SELECT … ;`
Tables temporaires	

Table temporaire locale :
`SELECT … INTO #nomtable … ;`
Table temporaire globale :
`SELECT … INTO ##nomtable … ;`
	Les tables temporaires sous Oracle ne sont pas temporaires ! C'est leur contenu qui l'est. Les tables temporaires sont vidées, soit lors du `COMMIT`, soit lors de la fin de session.
	`CREATE GLOBAL TEMPORARY TABLE nomtable … [ON COMMIT PRESERVE ROWS] ;`

Les transactions	

SQLServer est par défaut en auto-commit.
Il faut donc demander explicitement à démarrer une transaction :
`BEGIN TRAN nomtransaction ;`
Valider une transaction :
`COMMIT TRAN nomtransaction ;`
Annuler une transaction :
`ROLLBACK TRAN nomtransaction ;`
	Sous Oracle, les instructions s'exécutent toujours dans le cadre d'une transaction. Il faut donc penser à :
	
	Valider la transaction :
	`COMMIT;`
	Annuler la transaction :
	`ROLLBACK ;`