Trucs et astuces
proc sql (SAS) / bases de données (SGBD)

SQL explicite *Pass-Throught*

SQL Explicte vs Implicite

Implicite

```
LIBNAME oraloc1 ORACLE PATH=ORA10LOCHES
        SCHEMA=SCOTT USER=xxxxx PASSWORD=xxxxx;
```

NOTE: Libref ORALOC1 was successfully assigned as follows: Engine: ORACLE Physical Name: ORA10LOCHES

Caractéristiques de la méthode implicite:

- Les tables oracle sont gérées, dans le code, comme des tables SAS associées à une librairie,
- SAS pilote la soumission du code au SGBD : pas de flexibilité pour le programmeur

```
proc sql;
    create table visiteurs
    as
    select * from oraloc1.visiteurs;
quit;

data visiteurs;
     set oraloc1.visiteurs;
run;
```

Explicite

Le code SQL est envoyé tel quel à la base de données. Il faut donc soumettre un code spécifique selon le SGBD.

```
proc sql;
     connect to oracle
    (user=xxxx password=xxxx  path=ORA10LOCHES);
     create table visiteurs
     as
     select * from connection to oracle
        (select * from visiteurs);
quit;
```

Faciliter les développements

Savoir ce qui se passe côté SGBD

```
options sastrace=',,,d' sastraceloc=saslog;

proc sql;
  connect to oracle as CONBIL
    ( user="******" password="******"
path="*****");
  create table EXTRACT as
  select * from connection to CONBIL
    /* Requête SGBD */
    (select *
    from VENTES
    where ANNEE=&ANNEE.
    );
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

sastrace

Spécifie les informations du SGBD à envoyer à SAS.
Ici `sastrace=',,,d'` signifie que toutes les instructions envoyées au SGBD sont intégrées au journal

SELECT	DELETE
CREATE	SYSTEM CATALOG
DROP	COMMIT
INSERT	ROLLBACK
UPDATE	

sastraceloc

Spécifie l'emplacement de la log :
- log de SAS — saslog
- fichier — file 'emplacement'

```
ORACLE_1: Prepared: on connection 0 25 1884991235
Main 0 SQL (2)
select * from VENTES where ANNEE=2019
```

- Code SQL natif soumis au SGBD
- Visualisation de l'interprétation des macros variables

Faciliter les développements

Limiter les données à manipuler

```
proc sql inobs=4;
  connect to oracle as CONBIL
    ( user="******" password="******"
path="*****");
  create table EXTRACT as
  select * from connection to CONBIL
    /* Requête SGBD */
    (select *
    from COURSE
    );
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

prenom	temps	place
Naïma	3 min 15 s	1
Sarah	3 min 19 s	3
Sonia	3 min 16 s	2
Luc	10 min 39 s	4

Interactions SAS / SGBD

Paramétrer le sql

Caractères avec des guillemets :

- Utiliser les fonctions macros pour gérer les caractères spéciaux à la compilation.

```
%let COUREUR=%nrbquote('Luc');
```

Utiliser les fonctions macros pour l'exécution :

```
%unquote(&COUREUR.)
```

```
proc sql;
  connect to oracle as CONBIL
    ( user="******" password="******"  path="*****");
  create table EXTRACT as
  select * from connection to CONBIL
    /* Requête SGBD */
    (select *
    from COURSE
    where prenom= %unquote(&COUREUR.)    );
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

Optimiser les traitements

Gestion des formats

```
proc sql;
  connect to oracle as CONBIL
    ( user="******" password="******"
path="*****");
  create table FORMATS_IN as
  select   "C"           as TYPE
        ,"REGION"    as FMTNAME
        ,ID_REGION   as START
        ,CODE_REGION   as LABEL
  from connection to CONBIL
    /* Requête SGBD */
    (select ID_REGION
         ,CODE_REGION
    from DM_REGION
    );
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

Récupération des informations du SGBD et création de la table au format attendu par la proc format

- TYPE : numérique (N) ou caractère (C),
- FMTNAME : nom du format
- START : colonne des valeurs
- LABEL : colonne du format à appliquer

```
proc format cntlin=FORMATS_IN;
run;
```

Intérêts des formats

- limite les jointures (modèle en étoile),
- Optimise le temps de traitement
- Améliore la visibilité du code

Optimiser les traitements

Gestion des formats

```
proc sql;
  connect to oracle as CONBIL
    ( user="******" password="******"
path="*****");
  create table VENTES as
  select *
  from connection to CONBIL
    /* Requête SGBD */
    (select REG.CODE_REGION
          ,CLI.NOM_CLIENT
          ,VTS.NB_VENTES
    from VENTES VTS
    inner join DM_REGIONS REG
        on (VTS.ID_REGION=REG.ID_REGION)
    inner join DM_CLIENTS CLI
        on (VTS.ID_CLIENT=CLI.ID_CLIENT));
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

Jointures SGBD

Cette technique s'avère particulièrement efficace quand des jointures sont effectuées entre deux tables Oracle de volumétries différentes : une table avec peu d'enregistrements et une autre beaucoup plus importante.

```
proc sql;
  connect to oracle as CONBIL
    ( user="******" password="******"
path="*****");
  create table VENTES as
  select put(ID_REGION,$REGION.)  as CODE_REGION
     ,put(ID_CLIENT,$NOM.)  as NOM_CLIENT
     ,NB_VENTES
  from connection to CONBIL
    /* Requête SGBD */
    (select ID_REGION
          ,ID_CLIENT
          ,NB_VENTES
    from VENTES
    );
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

Formats SAS

Suppression des jointures : identifiants de la table de faits

Optimiser les traitements

Informer SAS des tris effectués

```
proc sql;
  connect to oracle as CONBIL
    ( user="******" password="******"  path="*****");
  create table REGIONS as
  select  *
  from connection to CONBIL
    /* Requête SGBD */
    (select ID_REGION
          ,CODE_REGION
     from DM_REGION
     order by ID_REGION);
    /* Requête SGBD */
  disconnect from CONBIL;
 quit;
```

Par défaut, la table SAS résultante n'est pas considérée comme triée, le tri ayant été effectué par le SGBD.
L'option de table SORTEDBY= permet de lister les colonnes de tri et le sens du tri (ascending par défaut).
Cela optimise, la création des index, l'utilisation des jointures.

The CONTENTS Procedure

Data Set Name		Observations	27
Member Type	DATA	Variables	2
Engine	V9	Indexes	0
Created	mardi 24 septembre 2019	Observation Length	28
Last Modified	mardi 24 septembre 2019	Deleted Observations	0
Protection		Compressed	BINARY
Data Set Type		Reuse Space	NO
Label		Point to Observations	YES
Data Representation	WINDOWS_64	Sorted	NO
Encoding	wlatin1 Western (Windows)		

```
create table REGIONS (sortedby=ID_REGION)
```

The CONTENTS Procedure

Data Set Name		Observations	27
Member Type	DATA	Variables	2
Engine	V9	Indexes	0
Created	mardi 24 septembre 2019	Observation Length	28
Last Modified	mardi 24 septembre 2019	Deleted Observations	0
Protection		Compressed	BINARY
Data Set Type		Reuse Space	NO
Label		Point to Observations	YES
Data Representation	WINDOWS_64	Sorted	YES
Encoding	wlatin1 Western (Windows)		

Sort Information	
Sortedby	ID_REGION
Validated	NO
Character Set	ANSI

Optimiser les traitements

Laisser le SGBD agir

Utiliser efficacement les spécificités des SGBD.
Exemples d'utilisation :

Pour Oracle
- Utiliser les fonctions analytiques
- Les clauses `with'

Pour teradata :
- Créer des tables temporaires indexées

Oracle : Clause with

```
WITH dept_count AS (
  SELECT deptno, COUNT(*) AS dept_count
  FROM   emp
  GROUP BY deptno)
SELECT e.ename AS employee_name,
      dc.dept_count AS emp_dept_count
FROM   emp e
      JOIN dept_count dc ON e.deptno = dc.deptno;
```

Avantages :
- Peut améliorer la visibilité du code
- Oracle peut créer une table temporaire globale

Optimiser les traitements

Laisser le SGBD agir

Fonction d'agrégation *(sum, max, min,…)*

```
SELECT id_inter,
    jurisdiction,
    cnt_entities,
    sum(cnt_entities) OVER (PARTITION BY id_inter) AS
entities_by_inter
FROM nb_entities
```

id_inter	jurisdiction	cnt_entities	entities_by_inter
4000	SAM	10	16
4000	PMA	1	16
4000	CYP	5	16
4001	SAM	3	5
4001	NEV	2	5

Fonction de rang

```
SELECT    prenom,
          temps,
    rank() OVER (ORDER BY temps)
FROM course
```

prenom	temps	rank()
Naïma	3 min 15 s	1
Sarah	3 min 19 s	3
Sonia	3 min 16 s	2
Luc	10 min 39 s	4