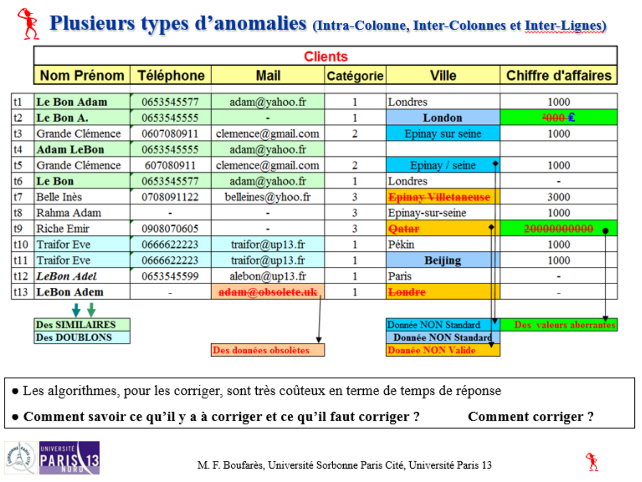
Intégration des données
&
Elimination des doubles et des similaires
Matching, Merging, and Deduplication

M. F. Boufarès
Université Sorbonne Paris Cité, Université Paris 13
Master 2 EID2, Informatique, Ingénieur

Think DIFFERENTLY, BIGGER and SMARTER !

Votre mission, si vous l'acceptez, est : Eliminer les doubles et les similaires !
Si vous échouez, nous nierons avoir eu connaissance de vos agissements !



Plusieurs types d'anomalies (Intra-Colonne, Inter-Colonnes et Inter-Lignes)

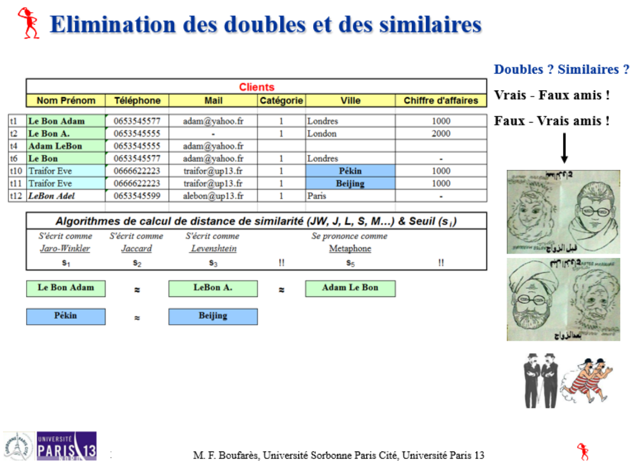
Clients

	Nom Prénom	Téléphone	Mail	Catégorie	Ville	Chiffre d'affaires
t1	**Le Bon Adam**	0653545577	adam@yahoo.fr	1	Londres	1000
t2	**Le Bon A.**	0653545555	-	1	**London**	~~2000 £~~
t3	Grande Clémence	0607080911	clemence@gmail.com	2	Epinay sur seine	1000
t4	**Adam LeBon**	0653545555	adam@yahoo.fr			
t5	Grande Clémence	607080911	clemence@gmail.com	2	Epinay / seine	1000
t6	**Le Bon**	0653545577	adam@yahoo.fr	1	Londres	-
t7	Belle Inès	0708091122	belleines@yhoo.fr	3	~~Epinay Villetaneuse~~	3000
t8	Rahma Adam	-	-	3	Epinay-sur-seine	1000
t9	Riche Emir	0908070605	-	3	~~Qatar~~	~~20000000000~~
t10	Traifor Eve	0666622223	traifor@up13.fr	1	Pékin	1000
t11	Traifor Eve	0666622223	traifor@up13.fr	1	**Beijing**	1000
t12	***LeBon Adel***	0653545599	alebon@up13.fr	1	Paris	-
t13	**LeBon Adem**	-	~~adam@obsolete.uk~~	1	~~Londre~~	-

🟩 Des SIMILAIRES
Des DOUBLONS
Des données obsolètes
Donnée NON Standard
Donnée NON Standard
Donnée NON Valide
Des valeurs aberrantes

- Les algorithmes, pour les corriger, sont très coûteux en terme de temps de réponse
- **Comment savoir ce qu'il y a à corriger et ce qu'il faut corriger ?** **Comment corriger ?**

M. F. Boufarès, Université Sorbonne Paris Cité, Université Paris 13



Elimination des doubles et des similaires

Clients

	Nom Prénom	Téléphone	Mail	Catégorie	Ville	Chiffre d'affaires
t1	**Le Bon Adam**	0653545577	adam@yahoo.fr	1	Londres	1000
t2	**Le Bon A.**	0653545555	-	1	London	2000
t4	**Adam LeBon**	0653545555	adam@yahoo.fr			
t6	**Le Bon**	0653545577	adam@yahoo.fr	1	Londres	-
t10	Traifor Eve	0666622223	traifor@up13.fr	1	**Pékin**	1000
t11	Traifor Eve	0666622223	traifor@up13.fr	1	**Beijing**	1000
t12	***LeBon Adel***	0653545599	alebon@up13.fr	1	Paris	-

Algorithmes de calcul de distance de similarité (JW, J, L, S, M…) & Seuil (s_i)

S'écrit comme Jaro-Winkler	S'écrit comme Jaccard	S'écrit comme Levenshtein		Se prononce comme Metaphone	
s_1	s_2	s_3	!!	s_5	!!

Le Bon Adam ≈ **LeBon A.** ≈ **Adam Le Bon**

Pékin ≈ **Beijing**

Doubles ? Similaires ?
Vrais - Faux amis !
Faux - Vrais amis !

M. F. Boufarès, Université Sorbonne Paris Cité, Université Paris 13

L'**élimination des doubles et des similaires**, appelée **déduplication des données** (**Record Linkage, Duplicates Data, Entity Resolution, Entity Matching**), consiste à détecter les lignes doubles ou similaires afin d'éliminer certaines redondances dans une source de données.

Les lignes similaires ou en double pourraient être fusionnées ou encore éliminées.

Cette problématique continue à faire l'objet de plusieurs travaux de recherche, d'une part, afin d'améliorer les performances des algorithmes pour de très gros volumes de données, et d'autre part, afin d'obtenir de meilleures précisions quant au calcul de similarité entre les lignes à cause des différentes anomalies qui existent dans les données.

En effet, la similarité entre les données est basée sur les différentes méthodes de mesures de distances de similarités qui existent dans la littérature telles que :

(i) les **mesures lexicographiques** (Levenshtein -Edit-Distance-, Jaro-Winkler, Q-Gram), ou
(ii) les **mesures phonétiques** (Soundex, Double Metaphone).

Il faut signaler que toutes ces méthodes ne prennent pas en considération la **sémantique contextuelle des données** à comparer.

Par exemple, la chaine de caractères « **Londres** » est égale à la chaine de caractères « **London** » si l'on sait qu'il s'agit de chaines de caractères qui représentent une **Ville** (**City**), elles sont tout simplement écrites dans deux langues différentes. Alors que, si ces deux chaines représentaient un **Nom** (**LastName**), elles seraient différentes.

Le choix des attributs clés pour la comparaison (les colonnes qui servent pour la comparaison) constitue aussi un grand problème.

Le résultat de l'élimination des doubles est fortement dépendant des étapes de nettoyage et de profilage des données qui précèdent la déduplication effective.

L'ordre des priorités des actions à mener influence grandement la qualité du résultat. L'homogénéisation, le traitement des valeurs nulles, les liens inter-colonnes et les liens inter-lignes sont des étapes dépendantes.

Différentes approches déterministes existent pour la comparaison des lignes dans une sources de données. Elles sont basées sur deux fonctions : **Match** (pour la comparaison) et **Merge** (pour la fusion).

Là encore, peu ou pas de sémantique est prise en compte dans le processus de dédoublonnage. Dans notre approche, la catégorisation sémantique a permis la recommandation des colonnes à prendre en considération pour l'élimination des doubles ce qui est fondamental pour la validité du résultat.

Exemples de données « dupliquées » suite au processus d'intégration de données (Doubles & Similaires) :

Exemple 1 :

BD ACM →→→ ID;Title;Authors;Venue;Year

564753;**A compact B-tree**;Peter Bumbulis, Ivan T. Bowman;International Conference on Management of Data;2002

872806;**A theory of redo recovery**;David Lomet, Mark Tuttle;International Conference on Management of Data;2003

BD DBLP →→→ ID;Title;Authors;Venue;Year

conf/sigmod/BumbulisB02;**A compact B-tree**;Ivan T. Bowman, Peter Bumbulis;SIGMOD Conference;2002

conf/sigmod/LometT03;**A Theory of Redo Recovery**;Mark R. Tuttle, David B. Lomet;
SIGMOD Conference;2003

conf/sigmod/DraperHW01;The Nimble Integration Engine;Daniel S. Weld, Alon Y. Halevy, Denise Draper;SIGMOD Conference ;2001

Exemple d'intégration de données dans le domaine de la bibliographie.
Traitement des doubles et des similaires

BD ACM →→→ ID;Title;Authors;Venue;Year

564753;**A compact B-tree**;Peter Bumbulis, Ivan T. Bowman;International Conference on Management of Data;2002

BD DBLP →→→ ID;Title;Authors;Venue;Year

conf/sigmod/BumbulisB02;**A compact B-tree**;Ivan T. Bowman, Peter Bumbulis;SIGMOD Conference;2002

	Title	**Authors**	**Venue**	**Year**
ID ID	Title Title	Authors Authors	Venue Venue	Year Year
564753	**A compact B-tree**	Peter Bumbulis, Ivan T. Bowman	International Conference on Management of Data	2002
conf/sigmod/ BumbulisB02	**A compact B-tree**	Ivan T. Bowman, Peter Bumbulis	SIGMOD Conference	2002

Title ?

A compact B-tree = or ≈ or ≠ A compact B-tree ?

??? **SimilarityDistanceAlg**(A compact B-tree ; A compact B-tree) **≤ Seuil1**

Authors ?

Peter Bumbulis, Ivan T. Bowman = or ≈ or ≠ Ivan T. Bowman, Peter Bumbulis ?

??? **SimilarityDistanceAlg**(Peter Bumbulis, Ivan T. Bowman ; Ivan T. Bowman, Peter Bumbulis) **≤ Seuil2**

Venue ?

International Conference on Management of Data = or ≈ or ≠ SIGMOD Conference ?

??? **SimilarityDistanceAlg**(International Conference on Management of Data ; SIGMOD Conference) **≤ Seuil3**

Year ?

2002 = or ≈ or ≠ 2002 ?

SimilarityDistanceAlg(2002; 2002) **≤ Seuil4**

Exemple 2 :

Traitement des doubles & des similaires : Record Linkage = Entity Resolution = Deduplication = Data Matching

===== Construction de la Base de Données mondiale des Hôtels ====================
===== Intégration de plusieurs sources de données ==============================

Best Western London Peckham Hotel
110 Peckham Road, Southwark, Londres, SE15 5EU, Royaume-Uni -
Téléphone : +44 20 7701 4222

Best Western London Peckham Hotel
110 Peckham Rd, London SE15 5EU, United Kingdom
Téléphone : +44 20 7701 4222

Hôtel Best Western London Peckham
110 Peckham Road, Southwark, Londres, RU
Téléphone : 00 44 20 7701 4222

==

Les fonctions Oracle, les plus connues, de calcul de similarité

Les fonctions Oracle, les plus connues, de calcul de **similarité** et de distance de similarité entre les **chaines de caractères** sont : **Levenshtein (Edit_Distance) distance, Jaro–Winkler distance, Soundex distance.**

NB : Les notes ci-dessous sont largement inspirées de la documentation Oracle sur le Web.

EDIT_DISTANCE Function ; EDIT_DISTANCE_SIMILARITY Function
JARO_WINKLER Function ; JARO_WINKLER_SIMILARITY Function
SOUNDEX Function

EDIT_DISTANCE Function

This function calculates the number of insertions, deletions or substitutions required to transform string-1 into string-2.

Syntax

```
UTL_MATCH.EDIT_DISTANCE (s1 IN VARCHAR2, s2 IN VARCHAR2)
RETURNS PLS_INTEGER;
```

Examples

```
SELECT UTL_MATCH.EDIT_DISTANCE('shackleford', 'shackelford') FROM
DUAL;
returns 2
```

EDIT_DISTANCE_SIMILARITY Function

This function calculates the number of insertions, deletions or substations required to transform string-1 into string-2, and returns the Normalized value of the Edit Distance between two Strings. The value is typically between 0 (no match) and 100 (perfect match).

Syntax

```
UTL_MATCH.EDIT_DISTANCE_SIMILARITY(s1 IN VARCHAR2, s2 IN VARCHAR2)
RETURNS PLS_INTEGER;
```

Examples

```
SELECT UTL_MATCH.EDIT_DISTANCE_SIMILARITY(‘shackleford’,
‘shackelford’) FROM DUAL;
returns 82
```

JARO_WINKLER Function

This function calculates the measure of agreement between two strings.

Syntax

```
UTL_MATCH.JARO_WINKLER (s1 IN VARCHAR2, s2 IN  VARCHAR2)
RETURNS BINARY_DOUBLE;
```

Examples

```
SELECT UTL_MATCH.JARO_WINKLER('shackleford', 'shackelford') FROM
DUAL;
returns 9.818E-001
```

JARO_WINKLER_SIMILARITY Function

This function calculates the measure of agreement between two strings, and returns a score between 0 (no match) and 100 (perfect match).

Syntax

```
UTL_MATCH.JARO_WINKLER (s1 IN VARCHAR2, s2 IN VARCHAR2)
RETURNS PLS_INTEGER;
```

Examples

```
SELECT UTL_MATCH.JARO_WINKLER_SIMILARITY('shackleford',
'shackelford')
FROM DUAL;
returns 98
```

EXAMPLES

```
DROP TABLE match_tab;

CREATE TABLE match_tab (
  id   NUMBER,
  col1 VARCHAR2(15),
  col2 VARCHAR2(15),
  CONSTRAINT match_tab_pk PRIMARY  KEY (id)
);

INSERT INTO match_tab VALUES (1, 'Peter Parker', 'Pete Parker');
INSERT INTO match_tab VALUES (2, 'Peter Parker', 'peter parker');
INSERT INTO match_tab VALUES (3, 'Clark Kent', 'Claire Kent');
INSERT INTO match_tab VALUES (4, 'Wonder Woman', 'Ponder Woman');
INSERT INTO match_tab VALUES (5, 'Superman', 'Superman');
INSERT INTO match_tab VALUES (6, 'The Hulk', 'Iron Man');
COMMIT;
```

EDIT_DISTANCE

The "Edit Distance", or "Levenshtein Distance", test measures the similarity between two strings by counting the number of character changes (inserts, updates, deletes) required to transform the first string into the second. The number of changes required is known as the distance.

```
SELECT col1, col2, UTL_MATCH.edit_distance(col1, col2) AS ed
FROM   match_tab ORDER BY id;
```

```
COL1            COL2                    ED
--------------- --------------- ----------
Peter Parker    Pete Parker              1
Peter Parker    peter parker             2
Clark Kent      Claire Kent              2
Wonder Woman    Ponder Woman             1
Superman        Superman                 0
The Hulk        Iron Man                 8
```

EDIT_DISTANCE_SIMILARITY

The EDIT_DISTANCE_SIMILARITY function uses the same method as the EDIT_DISTANCE function to determine the similarity of the strings, but it returns a normalized result ranging from 0 (no match) to 100 (complete match).

```
SELECT col1, col2, UTL_MATCH.edit_distance_similarity(col1, col2) AS eds
FROM   match_tab ORDER BY id;
```

```
COL1            COL2                   EDS
--------------- --------------- ----------
Peter Parker    Pete Parker             92
Peter Parker    peter parker            84
Clark Kent      Claire Kent             82
Wonder Woman    Ponder Woman            92
Superman        Superman               100
The Hulk        Iron Man                 0
```

```
SELECT id, col1, col2,
       UTL_MATCH.edit_distance_similarity(col1, col2) AS eds
FROM   match_tab
WHERE  UTL_MATCH.edit_distance_similarity(col1, col2) > 90
ORDER BY id;
```

```
        ID COL1            COL2                   EDS
---------- --------------- --------------- ----------
         1 Peter Parker    Pete Parker             92
         4 Wonder Woman    Ponder Woman            92
         5 Superman        Superman               100
```

JARO_WINKLER

The "Jaro-Winkler Algorithm" provides a different method for finding the distance between two strings.

```
SELECT col1, col2, UTL_MATCH.jaro_winkler(col1, col2) AS jw
FROM   match_tab ORDER BY id;
```

```
COL1            COL2                    JW
--------------- --------------- ----------
Peter Parker    Pete Parker     9.288E-001
Peter Parker    peter parker    8.889E-001
Clark Kent      Claire Kent     9.083E-001
Wonder Woman    Ponder Woman    9.444E-001
Superman        Superman          1.0E+000
The Hulk        Iron Man        4.167E-001
```

JARO_WINKLER_SIMILARITY

The JARO_WINKLER_SIMILARITY function uses the same method as the JARO_WINKLER function to determine the similarity of the strings, but it returns a normalized result ranging from 0 (no match) to 100 (complete match).

```
SELECT col1, col2, UTL_MATCH.jaro_winkler_similarity(col1, col2) AS jws
FROM   match_tab ORDER BY id;
```

```
COL1            COL2                   JWS
--------------- --------------- ----------
Peter Parker    Pete Parker             92
Peter Parker    peter parker            88
Clark Kent      Claire Kent             90
Wonder Woman    Ponder Woman            94
Superman        Superman               100
The Hulk        Iron Man                41
```

```
SELECT col1, col2, UTL_MATCH.jaro_winkler_similarity(col1, col2) AS jws
FROM   match_tab
WHERE  UTL_MATCH.jaro_winkler_similarity(col1, col2) > 90
ORDER BY id;
```

```
COL1            COL2                   JWS
--------------- --------------- ----------
Peter Parker    Pete Parker             92
Wonder Woman    Ponder Woman            94
Superman        Superman               100
```

Levenshtein distance

In information theory and computer science, the **Levenshtein distance** is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who considered this distance in 1965.

Levenshtein distance may also be referred to as **edit distance**, although that may also denote a larger family of distance metrics. It is closely related to pairwise string alignments.

Jaro–Winkler distance

In computer science and statistics, the **Jaro–Winkler distance** (Winkler, 1990) is a measure of similarity between two strings. It is a variant of the **Jaro distance** metric (Jaro, 1989, 1995), a type of string edit distance, and was developed in the area of record linkage (duplicate detection) (Winkler, 1990). The higher the Jaro–Winkler distance for two strings is, the more similar the strings are. The Jaro-Winkler similarity (for equation see below) is given by 1 - Jaro Winkler distance. The Jaro–Winkler distance metric is designed and best suited for short strings such as person names. The score is normalized such that 0 equates to no similarity and 1 is an exact match.

SOUNDEX

SOUNDEX is a build in function used to compare data items with their audiable or spoken values. It is a phonetic algorithm for indexing names by sound, as pronounced in English. For example, REIN, REIGN, and RAIN are all spelled differently but sound the same when spoken aloud. These "sound-alike" words are referred to as "homophones" or "heterographs". The National Archives and Records Administration (NARA) maintains the current rule set for the official implementation of Soundex used by the U.S. Government.

SOUNDEX returns a character string containing the phonetic representation of char. This function lets you compare words that are spelled differently, but sound alike in English.

```
SELECT SOUNDEX('REIN'),SOUNDEX('REIGN'),SOUNDEX('RAIN') FROM DUAL;

SOUN SOUN SOUN
---- ---- ----
T500 T500 T500
```

Examples

The following example returns the employees whose last names are a phonetic representation of "Smyth":

```
SELECT last_name, first_name FROM hr.employees
WHERE SOUNDEX(last_name)= SOUNDEX('SMYTHE');

LAST_NAME  FIRST_NAME
---------- ----------
Smith      Lindsey
Smith      William
```

Fonction soundex francais !

A tester sur Oracle ! soundexfr, soundex_fr, soundex2, phonex ???

```
SELECT  soundexfr('durand'),
        soundexfr('durhand'),
        soundexfr('durond'),
        soundexfr('dupond')
FROM  DUAL;
```

Metaphone / Double Metaphone

The Metaphone processor converts the values for a String attribute into a code which represents the phonetic pronunciation of the original string, using the Double Metaphone algorithm.

The Double Metaphone algorithm is a more general phonetic technique than Soundex (which is specifically designed for people’s names), and is more sophisticated and context-sensitive than the original Metaphone algorithm.

Etudier le comportement des algorithmes de calcul de distance de similarité entre les valeurs dans une BD selon leurs catégories (leurs sémantiques) !

```
-- =====================================================
-- =====================================================
REM -- Elimination des doubles et des similaires
REM -- Matching, Merging, and Deduplication
-- =====================================================
-- =====================================================

REM - Professeur : Mr M. F. Boufarès
REM - Big Data, Bases de Données Avancées, Entrepôts de données, Lacs de données

-- =====================================================

REM -- Binôme Numéro : xy
REM -- NOM1 Prénom1 & NOM2 Prénom2

-- =====================================================

-- =====================================================
REM – Re-Création de la table des différentes valeurs à comparer
-- =====================================================

DROP TABLE matchval;

-- =====================================================
REM -- Création de la table des différentes valeurs à comparer
-- =====================================================

CREATE TABLE matchval
(
  Idval         VARCHAR2(7),
  categorieval  VARCHAR2(15),
  valeur1       VARCHAR2(15),
  valeur2       VARCHAR2(15),
  CONSTRAINT matchval_pk PRIMARY KEY (Idval)
);

-- =====================================================
REM – Insertion dans la table des différentes valeurs à comparer
-- =====================================================

INSERT INTO matchval VALUES (‘A00-001’, ‘FIRSTNAME’, 'Adam', ‘ADAM');
INSERT INTO matchval VALUES (‘A00-002’, ‘FIRSTNAME’, 'Adam', ‘Adem');
INSERT INTO matchval VALUES (‘A00-003’, ‘FIRSTNAME’, 'Adam', ‘Adams');
INSERT INTO matchval VALUES (‘A00-004’, ‘FIRSTNAME’, 'Rahma', ‘Rama');
INSERT INTO matchval VALUES (‘A00-005’, ‘FIRSTNAME’, 'Marie-Noel', ‘Marie Noel');
INSERT INTO matchval VALUES (‘A00-006’, ‘FIRSTNAME’, 'Franc', ‘Frank');
INSERT INTO matchval VALUES (‘A00-007’, ‘FIRSTNAME’, 'Mbarak', ‘Moubarak');
INSERT INTO matchval VALUES (‘A00-008’, ‘FIRSTNAME’, 'Inès', ‘Ines');
INSERT INTO matchval VALUES (‘A00-009’, ‘FIRSTNAME’, 'Inès', ‘Iness');
INSERT INTO matchval VALUES (‘A00-010’, ‘FIRSTNAME’, 'Inès', ‘Yneès');
INSERT INTO matchval VALUES (‘A00-011’, ‘FIRSTNAME’, 'Inès', ‘Agnès');


COMMIT;

INSERT INTO matchval VALUES (‘A00-021’, ‘FIRSTLASTNAME’, 'Peter Parker', 'Pete Parker');
INSERT INTO matchval VALUES (‘A00-022’, ‘FIRSTLASTNAME’, 'Peter Parker', 'peter parker');
INSERT INTO matchval VALUES (‘A00-033’, ‘FIRSTLASTNAME’, 'Clark Kent', 'Claire Kent');
INSERT INTO matchval VALUES (‘A00-024’, ‘FIRSTLASTNAME’, 'Wonder Woman', 'Ponder Woman');
INSERT INTO matchval VALUES (‘A00-025’, ‘FIRSTLASTNAME’, 'Superman', 'Superman');
INSERT INTO matchval VALUES (‘A00-026’, ‘FIRSTLASTNAME’, 'The Hulk', 'Iron Man');
INSERT INTO matchval VALUES (‘A00-027’, ‘FIRSTLASTNAME’, 'Harissa FORD', ‘Harisson Ford');
INSERT INTO matchval VALUES (‘A00-028’, ‘FIRSTLASTNAME’, 'Bus WILLY', 'Bruce Willy');
INSERT INTO matchval VALUES (‘A00-029’, ‘FIRSTLASTNAME’, 'Brigitte Bardo', ‘Brigitte Fardo');
INSERT INTO matchval VALUES (‘A00-030’, ‘FIRSTLASTNAME’, 'Hedi Mufti', 'Eddy Murfi');
INSERT INTO matchval VALUES (‘A00-031’, ‘FIRSTLASTNAME’, 'Alain DE LOiN', 'Alain DELON');

COMMIT;

INSERT INTO matchval VALUES (‘A00-032’, ‘CITY’, 'Paris', ‘PArisss');
INSERT INTO matchval VALUES (‘A00-033’, ‘CITY’, 'Paris', ‘Pari');
INSERT INTO matchval VALUES (‘A00-034’, ‘CITY’, 'Pékin', ‘Beijing');
INSERT INTO matchval VALUES (‘A00-035’, ‘CITY’, 'Londres', ‘Londre');
INSERT INTO matchval VALUES (‘A00-036’, ‘CITY’, 'Londres', ‘London');

COMMIT;

INSERT INTO matchval VALUES (‘A00-041’, ‘DATE’, '10-11-2016', ’10-NOV-16');
INSERT INTO matchval VALUES (‘A00-042’, ‘DATE’, '10-11-2016', ’JEUDI 10 NOVEMBER 2016');
INSERT INTO matchval VALUES (‘A00-043’, ‘DATE’, '11-11-2016', ‘11-11-2016’);
INSERT INTO matchval VALUES (‘A00-044’, ‘DATE’, '11-11-2016', ‘11-12-2016’);
INSERT INTO matchval VALUES (‘A00-045’, ‘DATE’, '11-11-2016', ‘12-11-2016’);
INSERT INTO matchval VALUES (‘A00-045’, ‘DATE’, '11-11-2016', ‘11-11-2017’);

COMMIT;

INSERT INTO matchval VALUES (‘A00-051’, ‘EMAIL’, 'fb@lipn.univ-paris13.fr', ‘fb@lipn.univ-
paris13.fr’);
INSERT INTO matchval VALUES (‘A00-052’, ‘EMAIL’, 'fb@lipn.univ-paris13.fr', ‘yb@lipn.univ-
paris13.fr’);
INSERT INTO matchval VALUES (‘A00-053’, ‘EMAIL’, 'fb@lipn.univ-paris13.fr', ‘fb@iutv.univ-
paris13.fr’);

COMMIT;


INSERT INTO matchval VALUES (‘A00-061’, ‘URL’, 'www.univ-paris13.fr', ‘www.univ-paris13.fr’);
INSERT INTO matchval VALUES (‘A00-062’, ‘URL’, 'www.univ-paris13.fr', ‘www.univ-paris1.fr’);
INSERT INTO matchval VALUES (‘A00-063’, ‘URL’, 'www.univ-paris13.fr', ‘www.univ-paris3.fr’);
INSERT INTO matchval VALUES (‘A00-064’, ‘URL’, 'www.univ-paris13.fr', ‘www.univ-pari13.fr’);
INSERT INTO matchval VALUES (‘A00-065’, ‘URL’, 'www.univ-paris13.fr', ‘www.univparis13.fr’);

COMMIT;

INSERT INTO matchval VALUES (‘A00-071’, ‘NUMBER’, '17091958', ‘17091958’);
INSERT INTO matchval VALUES (‘A00-072’, ‘NUMBER’, '17091958', ‘27091958’);
INSERT INTO matchval VALUES (‘A00-073’, ‘NUMBER’, '12345679', ‘12345678’);
INSERT INTO matchval VALUES (‘A00-073’, ‘NUMBER’, '0.000012345679', ‘0.000012345678’);

COMMIT;
```

```
-- ======================================================
REM -- Clacul des distances de similarités
-- ======================================================
```

```
SELECT Idval,
       categorieval,
       valeur1,
       valeur2,
       UTL_MATCH.edit_distance_similarity(valeur1, valeur2) AS EDS
       UTL_MATCH.jaro_winkler_similarity(valeur1, valeur2) AS JWS
FROM   matchval
ORDER BY Idval;
```

```
SELECT Idval,
       categorieval,
       UPPER(valeur1),
       UPPER(valeur2),
       UTL_MATCH.edit_distance_similarity(UPPER(valeur1), UPPER(valeur2)) AS EDS
       UTL_MATCH.jaro_winkler_similarity(UPPER(valeur1), UPPER(valeur2)) AS JWS
FROM   matchval
ORDER BY Idval;
```

```
SELECT Idval,
       categorieval,
       LOWER(valeur1),
       LOWER(valeur2),
       UTL_MATCH.edit_distance_similarity(LOWER(valeur1), LOWER(valeur2)) AS EDS
       UTL_MATCH.jaro_winkler_similarity(LOWER(valeur1), LOWER(valeur2)) AS JWS
FROM   matchval
ORDER BY Idval;
```