

BDC

Bases de données

**Langage de manipulation des
données : SQL**

Instruction de manipulation des données

- **SELECT** lecture des données
- **INSERT** insertion de nouveaux tuples
- **UPDATE** modification de tuples
- **DELETE** suppression de tuples



Lecture des données

Projection de relations

- Projection de toute la relation

```
SELECT *  
FROM relation;
```

- Projection de quelques colonnes

```
SELECT attribut1, attribut2, ...  
FROM relation;
```

- Projection avec suppression des doublons

```
SELECT DISTINCT attribut1  
FROM relation;
```

- Projection avec renommage

```
SELECT attribut1 AS nom  
FROM relation;
```

Exemple de schéma de relations

- **CHAMBRE** (NumChambre, Prix, NbrLits, NbrPers,
Confort, Équipement)
- **CLIENT** (NumClient, Nom, Prénom, Adresse, Tel)
- **RESERVATION** (NumClient, NumChambre,
DateArrivée, DateDépart)

Exemple de projection

- Tous les tuples de la relation CHAMBRE

```
SELECT *  
FROM CHAMBRE;
```

- Tous les numéros de chambre avec leur capacité

```
SELECT NumChambre, NbrPers  
FROM CHAMBRE;
```

Exemple de projection

- Les différents types de confort

```
SELECT DISTINCT Confort  
FROM CHAMBRE;
```

- Les prix par personne des différentes chambres

```
SELECT NumChambre, Prix/NbrPers AS PrixParPersonne  
FROM CHAMBRE;
```

Projection avec tri du résultat

SELECT attribut 1, attribut 2, attribut3, etc
FROM relation
ORDER BY attribut1, attribut3;



SELECT attribut 1, attribut 2, attribut3
FROM relation
ORDER BY attribut1 **ASC**, attribut3 **ASC**;

SELECT attribut 1, attribut 2, attribut3
FROM relation
ORDER BY attribut1 **DESC**, attribut3 **ASC**;



SELECT attribut 1, attribut 2, attribut3
FROM relation
ORDER BY 1 **DESC**, 3 **ASC**;

Exemple de projection avec tri du résultat

- Lister les clients par ordre croissant des noms et prénoms

```
SELECT NumClient, Nom, Prenom, Adresse, Tel  
FROM Client  
ORDER BY Nom, Prenom ;
```

ou

```
SELECT NumClient, Nom, Prenom, Adresse, Tel  
FROM Client  
ORDER BY Nom ASC, Prenom ASC ;
```

ou

```
SELECT NumClient, Nom, Prenom, Adresse, Tel  
FROM Client  
ORDER BY 2, 3 ;
```

Restriction de relations

SELECT *
FROM relation
WHERE prédicat;

- On peut utiliser les opérateurs de comparaison : =, >, <, >=, <=, <>, !=
- On peut utiliser les opérateurs logiques : **AND**, **OR**, **NOT**
- On peut tester la ressemblance de chaînes de caractères à un modèle :
Attribut **LIKE** ModèleChaîne
- On peut tester l'appartenance d'un attribut à un domaine de valeurs :
BETWEEN Constante1 **AND** Constante2
- On peut tester l'appartenance ou non appartenance d'un attribut à un ensemble
IS IN (Const1, Const2, ...)
IS NOT IN (Const1, Const2, ...)
- On peut tester la nullité ou non nullité d'un attribut :
Attribut **IS NULL** ou Attribut **IS NOT NULL**

Restriction de relations

```
SELECT *  
FROM relation  
WHERE attribut1 = 'chaîne de caractères';
```

```
SELECT *  
FROM relation  
WHERE attribut2 BETWEEN valeur1 AND valeur2;
```

```
SELECT *  
FROM relation  
WHERE NOT (Prédicat1 AND Prédicat2);
```

Exemple de restriction

```
SELECT *  
FROM relation  
WHERE attribut3 LIKE '% B'  
OR    attribut3 LIKE '%CCC%'  
OR    attribut4 LIKE '75- - -';
```

```
SELECT *  
FROM relation  
WHERE attribut3 IS NULL  
OR attribut5 IS IN (valeur1, valeur2, valeur3);
```

Exemple de restriction

- Toutes les chambres ayant un bain et une télévision

SELECT *

FROM Chambre

WHERE Confort = 'Bain' **AND** UPPER (Equipment) = 'TV ';

- Toutes les chambres dont le prix est en dessous de 120 €

SELECT *

FROM Chambre

WHERE Prix <120;

- Toutes les chambres disposant d'un moyen pour se laver

SELECT *

FROM Chambre

WHERE Confort **IS IN** ('Bain', 'Douche') ;



La comparaison de chaînes de caractères tient compte de la casse

Exemple de restriction

- Tous les clients dont le nom commence par 'DU'

```
SELECT *  
FROM Client  
WHERE Nom like 'DU%';
```

- Toutes les chambres dont le prix est compris entre 85 € et 120 €

```
SELECT *  
FROM Chambre  
WHERE Prix BETWEEN 85 AND 120;
```

- Lister les chambres pour lesquels les clients n'ont pas annoncé une date de départ

```
SELECT NumChambre  
FROM Reservation  
WHERE DateDepart IS NULL;
```

Tables de vérité pour le calcul de prédicats

NOT	
F	V
I	I
V	F

NOT	
0	1
1/2	1/2
1	0

1-X

Tables de vérité pour le calcul de prédicats

OR	V	I	F
V	V	V	V
I	V	I	I
F	V	I	F

OR	1	1/2	0
1	1	1	1
1/2	1	1/2	1/2
0	1	1/2	0

Max (X, Y)

Tables de vérité pour le calcul de prédicats

AND	V	I	F
V	V	I	F
I	I	I	F
F	F	F	F

AND	1	1/2	0
1	1	1/2	0
1/2	1/2	1/2	0
0	0	0	0

Min (X, Y)

Valeur « Null » et évaluation de prédicats

A	condition	résultat
10	A is null	F
null	A is null	V
null	A = null	I
null	a!=10	I

ne pas utiliser

Evaluation de prédicats

- Le système utilise la table de vérité à trois
- Mais une condition d'un WHERE évaluée à I (à la fin)
agit comme un F

Remarque sur l'évaluation d'expressions

- Toute expression arithmétique contenant une valeur nulle est évaluée à null
 - $\text{exp} : \text{null} + 10 \rightarrow \text{null}$
- La plupart des fonctions manipulant une donnée à la fois renvoient une valeur nulle si cette donnée est nulle
 - $\text{exp} : \text{ABS}(n)$
- La plupart des fonctions manipulant plusieurs données à la fois ignorent les données nulles.
 - $\text{exp.} : \text{AVG}(\text{colonne})$

Exemple de schéma de base de données d'un organisme de voyage

- STATION (NomStation, Capacité, Lieu, Région, Tarif)
- ACTIVITE (NomStation, Libellé, Prix)
- CLIENT (IdClient, Nom, Prénom, Ville, Région, Solde)
- SEJOUR (IdClient, NomStation, DateSéjour, NbPLaces)

Exemple de schéma de base de données d'un organisme de voyage

STATION

NomStation	Capacité	Lieu	Région	Tarif
Venusa	350	Guadeloupe	Antilles	1200
Farniente	200	Seychelles	Océan Indien	1500
Santalba	150	Martinique	Antilles	2000
Passac	400	Alpes	Europe	1000

ACTIVITE

NomStation	Libellé	Prix
Venusa	Voile	150
Venusa	Plongee	120
Farniente	Plongee	130
Passac	Ski	200
Passac	Piscine	20
Santalba	Kayac	50

Exemple de schéma de base de données d'un organisme de voyage

CLIENT

IdClient	Nom	Prénom	Ville	Région	Solde
10	Fogg	Phileas	Londres	Europe	12465
20	Pascal	Blaise	Paris	Europe	6763
30	Kerouac	Jack	NewYork	Amerique	9812

SEJOUR

IdClient	NomStation	DateSéjour	NbPlaces
10	Passac	1998-07-01	2
30	Santalba	1996-08-14	5
20	Santalba	1998-08-03	4
30	Passac	1998-08-15	3
30	Venusa	1998-08-03	3
20	Venusa	1998-08-03	6
30	Farniente	1999-06-24	5
10	Farniente	1998-09-05	3

Exemple de schéma de base de données d'un organisme de voyage

SELECT Libelle
FROM Activite;



Libelle
Voile
Plongee
Plongee
Ski
Piscine
Kayac

SELECT DISTINCT Libelle
FROM Activite;



Libelle
Voile
Plongee
Ski
Piscine
Kayac

SELECT *
FROM Station;



Toute la relation Station

Exemple de schéma de base de données d'un organisme de voyage

```
SELECT NomStation  
FROM Station  
WHERE Region = 'ANTILLES';
```



NomStation

```
SELECT NomStation  
FROM Station  
WHERE Region = 'Antilles';
```



NomStation
Venusa Santalba

```
SELECT NomStation  
FROM Station  
WHERE UPPER(Region) = 'ANTILLES';
```



Exemple de schéma de base de données d'un organisme de voyage

```
SELECT Libelle, prix / 6.56, 'Cours de l'euro =', 6.56  
FROM Activite  
WHERE NomStation = 'Santalba';
```



Libellé	prix / 6.56	'Cours de l'euro ='	6.56
Kayac	7.62	'Cours de l'euro ='	6.56

```
SELECT Libelle, prix / 6.56 AS prixEnEuros  
FROM Activite  
WHERE NomStation = 'Santalba';
```



Libellé	prixEnEuros
Kayac	7.62

Produit cartésien et jointure

■ Produit cartésien

```
SELECT *  
FROM relation1, relation2;
```

■ Jointures

```
SELECT *  
FROM relation1, relation2  
WHERE critère de jointure;
```

[NOT] Prédicat1 [AND/OR Prédicat2]

- attributRelation1 <OpérateurComparaison> attributRelation2
- **relation1**.attribut <OpérateurComparaison> **relation2**.attribut

```
SELECT *  
FROM relation1 R1, relation2 R2  
WHERE critère de jointure;
```

[NOT] Prédicat1 [AND/OR Prédicat2]

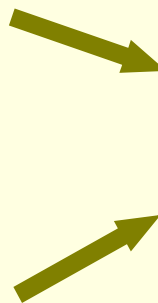
- attributRelation1 <OpérateurComparaison> attributRelation2
- **R1**.attribut <OpérateurComparaison> **R2**.attribut

Exemple de jointure avec restriction

- Donnez pour chaque activité de chaque station, le lieu où elle peut être pratiquée

```
SELECT NomStation, Libelle, Lieu  
FROM Station, Activite  
WHERE Station. NomStation = Activite. NomStation ;
```

```
SELECT NomStation, Libelle, Lieu  
FROM Station S, Activite A  
WHERE S. NomStation = A. NomStation ;
```



NomStation	Libelle	Lieu
Venusa	Voile	Guadeloupe
Venusa	Plongee	Guadeloupe
Farniente	Plongee	Seychelles
Passac	Ski	Alpes
Passac	Piscine	Alpes
Santalba	Kayac	Martinique

Exemple de jointure avec restriction

- Les noms et prénoms de tous les parisiens ayant effectué un séjour dans une station
- Mauvaise réponses :

```
SELECT Nom, Prénom  
FROM Client C, Sejour S  
WHERE S. IdClient = C. IdClient  
        AND Ville = 'Paris' ;
```



Un client parisien qui a séjourné +ieurs fois apparaît autant de fois que de séjours

```
SELECT DISTINCT Nom, Prénom  
FROM Client C, Sejour S  
WHERE S. IdClient = C. IdClient  
        AND Ville = 'Paris' ;
```



2 clients parisiens qui portent le même nom et même prénom sont confondus dans le résultat

- Bonne réponse :

```
SELECT DISTINCT IdClient, Nom, Prénom  
FROM Client C, Sejour S  
WHERE S. IdClient = C. IdClient  
        AND Ville = 'Paris' ;
```

Exemple de jointure avec restriction

- Le nom des clients n'ayant pas annoncé de date de départ

```
SELECT DISTINCT NumClient, Nom  
FROM Client, Reservation  
WHERE Client.NumClient= Reservation.NumClient  
        AND DateDepart IS NULL ;
```

```
SELECT DISTINCT NumClient, Nom  
FROM Client C, Reservation R  
WHERE C.NumClient=R.NumClient  
        AND DateDepart IS NULL ;
```

Remarque



Ne pas faire de jointures si pas nécessaire.

- Exemple : La liste des activités de chaque station
- Requête à retenir

```
SELECT NomStation, Libelle  
FROM Activite ;
```

- Requête à éviter

```
SELECT NomStation, Libelle  
FROM Activite A, Station S  
WHERE S. NomStation = A. NomStation ;
```

Union et intersection

■ Union

```
SELECT Attribut1, Attribut2, ..... }  
FROM .....  
UNION  
SELECT Attribut1, Attribut2, ..... }  
FROM ..... ;
```

Les deux relations
ont le même schéma

■ Intersection

```
SELECT Attribut1, Attribut2, ..... }  
FROM .....  
INTERSECT  
SELECT Attribut1, Attribut2, ..... }  
FROM ..... ;
```

Les deux relations
ont le même schéma



La suppression des doublons dépend des SGBDs

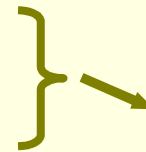
Différence

■ Différence

SELECT Attribut1, Attribut2,
FROM relation1

EXCEPT

SELECT Attribut1, Attribut2,
FROM relation2 ;

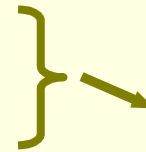


Les deux relations
ont le même schéma

SELECT Attribut1, Attribut2,
FROM relation1

MINUS

SELECT Attribut1, Attribut2,
FROM relation2 ;



Les deux relations
ont le même schéma

Exemple d'union, d'intersection et de différence

- Donnez tous les noms de région existant dans la base

```
SELECT Region FROM Station  
UNION  
SELECT Region FROM Client
```

- Donnez les régions où l'on trouve à la fois des clients et des stations

```
SELECT Region FROM Station  
INTERSECT  
SELECT Region FROM Client
```

- Donnez les régions où l'on trouve des stations mais pas des clients

```
SELECT Region FROM Station  
EXCEPT  
SELECT Region FROM Client
```

Les requêtes imbriquées

```
SELECT R1.attribut1, R1.Attribut2  
FROM relation1 R1  
WHERE R1.attribut3 OPERATEUR (SELECT .....);
```



IN ou **NOT IN**

ALL ou **ANY** précédé de =, <>, >, <, >= ou <=

- **IN** ou **NOT IN** : est ou n'est pas dans l'ensemble renvoyé par la sous requête
- **ANY** : la comparaison se fait avec au moins une des valeurs renvoyée par la sous requête (=ANY équivaut à IN)
- **ALL** : la comparaison se fait avec toutes les valeurs renvoyée par la sous requête (<> ALL équivaut à NOT IN)

Exemple de requêtes imbriquées

- Lister les numéros de chambre dont le prix est le plus élevé.

```
SELECT NumChambre  
FROM Chambre  
WHERE Prix >= ALL (SELECT DISTINCT Prix  
                        FROM Chambre);
```

- Les noms des stations où ont séjourné des clients parisiens

```
SELECT NomStation  
FROM Sejour  
WHERE IdClient IN (SELECT IdClient  
                    FROM Client  
                    WHERE ville = 'Paris');
```



IN ne peut être remplacé par le signe = car on n'est pas sûr que la sous requête renvoie un et un seul tuple

Les requêtes imbriquées versus jointures

- Certaines requêtes imbriquées peuvent s'écrire avec des jointures

```
SELECT DISTINCT NumClient, Nom  
FROM Client C, Reservation R  
WHERE C.NumClient=R.NumClient  
        AND DateDepart IS NULL ;
```

ou

```
SELECT NumClient, Nom  
FROM Client  
WHERE NumClient IN (SELECT NumClient  
                    FROM Reservation  
                    WHERE DateDepart IS NULL)
```

Un autre type de requêtes imbriquées

(test d'ensemble non vide)

```
SELECT listeAttributs  
FROM relation1 R1, .....  
WHERE OpérateurExistentiel (SELECT *  
FROM relation2 R2, .....  
WHERE R1.attribut = R2.attribut  
AND AutresConditions);
```



EXISTS ou Not EXISTS



```
SELECT listeAttributs  
FROM relation1 R1, .....  
WHERE R1.attribut3 OpérateurExistentiel (SELECT 'x'  
FROM relation2 R2, .....  
WHERE R1.attribut = R2.attribut  
AND AutresConditions);
```

Exemple de requêtes imbriquées

- Les noms et prénoms des clients arrivés aujourd'hui.

```
SELECT Nom, Prenom
FROM Client C
WHERE EXISTS (SELECT *
                FROM RESERVATION R
                WHERE DateArrivee = SYSDATE
                AND C.NumClient = R.NumClient;
```

```
SELECT Nom, Prénom
FROM Client C
WHERE EXISTS (SELECT ' x '
                FROM RESERVATION R
                WHERE Date Arrivée = SYSDATE
                AND C.NumClient = R.NumClient;
```

Exemple de requêtes imbriquées

- Les noms et prénoms des clients qui ont séjournés à Santalba

```
SELECT Nom, Prenom
FROM Client C
WHERE EXISTS (SELECT *
               FROM Sejour S
               WHERE NomStation = 'Santalba'
               AND C.IdClient = S.IdClient;
```

```
SELECT Nom, Prenom
FROM Client C
WHERE EXISTS (SELECT ' x '
               FROM Sejour S
               WHERE NomStation = 'Santalba'
               AND C.IdClient = S.IdClient;
```


Execution de la requête

Etape 0 : récupérer des tables intervenant dans la requête

CLIENT

IdClient	Nom	Prénom	Ville	Région	Solde
10	Fogg	Phileas	Londres	Europe	12465
20	Pascal	Blaise	Paris	Europe	6763
30	Kerouac	Jack	NewYork	Amérique	9812

SEJOUR

IdClient	NomStation	DateSéjour	NbPlaces
10	Passac	1998-07-01	2
30	Santalba	1996-08-14	5
20	Santalba	1998-08-03	4
30	Passac	1998-08-15	3
30	Venusa	1998-08-03	3
20	Venusa	1998-08-03	6
30	Farniente	1999-06-24	5
10	Farniente	1998-09-05	3

Execution de la requête

Etape 1 : exécuter les restrictions de la sous-requête qui sont indépendantes de la sur-requête

SEJOUR

IdClient	NomStation	DateSéjour	NbPlaces
10	Passac	1998-07-01	2
30	Santalba	1996-08-14	5
20	Santalba	1998-08-03	4
30	Passac	1998-08-15	3
30	Venusa	1998-08-03	3
20	Venusa	1998-08-03	6
30	Farniente	1999-06-24	5
10	Farniente	1998-09-05	3

Execution de la requête

Etape 2 : exécuter la ou les conditions liant la sous-requête à la sur-requête

CLIENT

IdClient	Nom	Prénom	Ville	Région	Solde
10	Fogg	Phileas	Londres	Europe	12465
20	Pascal	Blaise	Paris	Europe	6763
30	Kerouac	Jack	NewYork	Amérique	9812

SEJOUR

IdClient	NomStation	DateSéjour	NbPlaces
10	Passac	1998-07-01	2
30	Santalba	1996-08-14	5
20	Santalba	1998-08-03	4
30	Passac	1998-08-15	3
30	Venusa	1998-08-03	3
20	Venusa	1998-08-03	6
30	Farniente	1999-06-24	5
10	Farniente	1998-09-05	3

Execution de la requête

Etape 3 : exécuter la projection de la sur-requête

Nom	Prénom
Pascal	Blaise
Kerouac	Jack

Quelques fonctions de calcul

- **SUM**(attribut) ou **SUM(DISTINCT** attribut)
- **AVG**(attribut) ou **AVG(DISTINCT** attribut)
- **MAX**(attribut)
- **MIN**(attribut)
- **COUNT**(attribut) ou **COUNT(DISTINCT** attribut)

Exemple

- La capacité théorique d'accueil de l'hôtel

```
SELECT SUM(NbrPers)  
FROM CHAMBRE;
```

- Nombre de chambres de l'hôtel

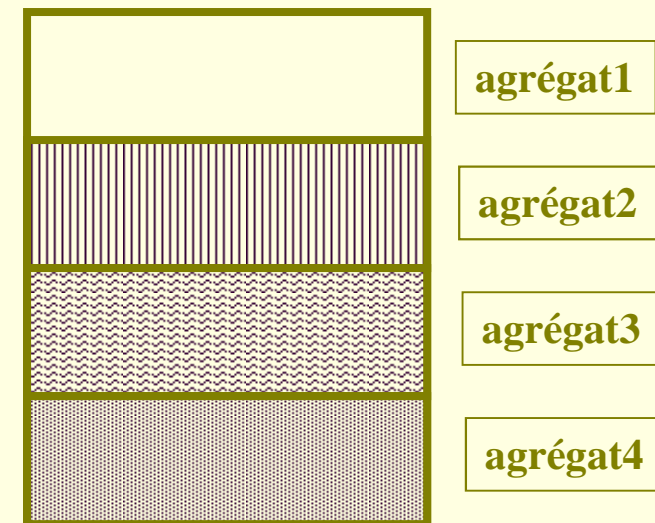
```
SELECT COUNT(*)  
FROM CHAMBRE;
```

- Nombre de stations, la moyenne des tarifs ainsi que le tarif max et min

```
SELECT COUNT(Nom Station), AVG(Tarif), MIN(Tarif), MAX(Tarif)  
FROM Station
```

Partitionnement d'une relation

- Possibilité de partitionner une relation.
- Le partitionnement se fait selon un critère de partitionnement
- Chaque partition est un agrégat



- Possibilité de sélectionner des agrégats répondant à un critère de sélection

Partitionnement d'une relation

Fait partie des attributs du critère de partitionnement

Appliquées sur les agrégats sélectionnés

SELECT liste d'attributs et/ou fcts de calcul

FROM relation1, relation2,

WHERE prédicats

Décrit la relation à partitionner. Le partitionnement sera celui de la relation issue du produit cartésien (si pas de WHERE) ou de la relation issue de la jointure et/ou restriction.

GROUP BY critère de partitionnement

Prédicat portant sur une liste d'attributs

HAVING critère de sélection de partitions;

Prédicat portant sur une liste d'attributs

Exemple

- Lister par type de confort les chambres dont le nombre de lits dépasse 1



Ens.Chambres

Décrit dans le FROM



Ens.Chambres
à plus de 1 lit

Décrit dans le WHERE



Ens.Chambres à plus
de 1 lit ayant confort a

.....

Ens.Chambres à plus
de 1 lit ayant confort z

Décrit dans le GROUP BY

Exemple

- Lister par type de confort les chambres dont le nombre de lits dépasse 1

```
SELECT Confort, NumChambre  
FROM CHAMBRE  
WHERE Nblits>1  
GROUP BY Confort, NumChambre
```

```
SELECT *  
FROM CHAMBRE  
WHERE Nblits>1  
GROUP BY Confort
```

Exemple

- Prix min et max des chambres par type de confort

```
SELECT Confort, MIN(Prix), MAX(Prix)  
FROM CHAMBRE  
GROUP BY Confort;
```

Il s'agit dans cette requête de calculer le minimum et le maximum des prix dans chaque partition; une partition étant un ensemble de chambre de même confort

Exemples

- Le nombre de places réservées par client

```
SELECT C.IdClient , Nom, SUM(NbPlaces)
FROM Client C, Sejour S
WHERE C.IdClient = S.IdClient
GROUP BY C.IdClient, Nom;
```

- Le nombre de places réservées par client. **On intéresse uniquement aux clients ayant réservés plus de 10 places**

```
SELECT C.IdClient , Nom, SUM(NbPlaces)
FROM Client C, Sejour S
WHERE C.IdClient = S.IdClient
GROUP BY C.IdClient, Nom
HAVING SUM(NbPlaces) > = 10;
```

RESUME

- **SELECT** attributs résultats [avec fonctions] 6
- **FROM** tables utilisées 1
- [**WHERE** critère de jointure et/ou critère de restriction] 2
- [**GROUP BY** critère de partitionnement] 3
- [**HAVING** critère de sélection portant sur les agrégats] 4
- [**ORDER BY** critère de tri du résultat]; 5



Insertion des données

Insertion de tuples

- Insertion tuple par tuple

INSERT INTO relation [(attribut [,attribut] ...)]

VALUES valeur [, valeur] ...;

- Insertion d'un ensemble de tuples 'calculés' à partir des données de la base

INSERT INTO relation [(attribut [,attribut] ...)]

Requête;

Pas de clause **ORDER BY** dans Requête



Seules les insertions respectant les contraintes sont exécutés par le SGBD

Exemples

- Hypothèse : la table client a été créée de cette façon :

```
CREATE TABLE CLIENT  
  ( NumClient NUMBER (4) PRIMARY KEY,  
    Nom VARCHAR2 (15) ,  
    Prenom VARCHAR2 (15),  
    Ville VARCHAR2 (30)  
    Tel NUMBER(10));
```

- On souhaite inserer le client Pierre Paul. Il habite Paris. Son Numéro est 6000 :

```
INSERT INTO Client
```

```
VALUES 6000, 'Pierre ', 'Paul', 'Paris', 0160708940;
```

Ou encore

```
INSERT INTO Client (NumClient, Prenom, Nom, Ville, Tel)
```

```
VALUES 6000, 'Paul', 'Pierre ', 'Paris', 0160708940;
```

La $j^{\text{ème}}$ donnée correspond au $j^{\text{ème}}$ attribut dans la déclaration du schéma de la table

La $i^{\text{ème}}$ donnée correspond au $i^{\text{ème}}$ attribut dans le INSERT

Exemples

- Mais si j'insère le tuple suivant :

INSERT INTO Client

VALUES 6000, 'Pierre ', 'Paul';

- Alors la ville et le téléphone du client 6000 seront égaux à NULL car dans la déclaration de la table CLIENT, les attributs Ville et Tel sont facultatifs.

Exemples

- Si comme hypothèse la table client a été créée de cette façon :

```
CREATE TABLE CLIENT  
( NumClient NUMBER (4) PRIMARY KEY,  
  Nom VARCHAR2 (15) ,  
  Prenom VARCHAR2 (15),  
  Ville VARCHAR2 (30) NOT NULL,  
  Tel NUMBER(10));
```

- Et que j'essaye d'insérer le tuple suivant :

```
INSERT INTO Client  
VALUES 6000, 'Pierre ', 'Paul';
```

- Alors le système rejettera mon insertion car dans la déclaration du schéma de la table client, l'attribut Ville est obligatoire.



Modification des données

Modification des données

UPDATE relation



La relation sur laquelle porte la modification des données

SET attribut = {expression |
NULL |requête}
[attribut = {valeur | NULL
|requête}] ...

La ou les colonnes à
modifier avec la modification
correspondante

[**WHERE** condition de
sélection];



Si la clause WHERE est présente,
la modification s'appliquera à tous
les tuples vérifiant la condition de
sélection.

Dans le cas contraire, la
modification s'appliquera à tous
les tuples de la relation



Seules les modifications respectant
les contraintes sont exécutées

Exemples

UPDATE Chambre
SET Prix=Prix*1.003;



L'augmentation du prix est appliquée à toutes les chambres

UPDATE client
SET Ville='Nimes'
WHERE NumClient=6000;



L'adresse est modifiée pour le client 6000

UPDATE client
SET tel=
(**SELECT** tel-pr
FROM Conferencier
WHERE idConferencier = 6000)
WHERE NumClient=6000;



Le numéro de téléphone du conférencier 6000 est attribué au client 6000



suppression des données

Suppression des données

DELETE FROM relation



La relation concernée par la suppression des données

[**WHERE** condition de sélection];



Si la clause **WHERE** est présente, tous les tuples vérifiant la condition de sélection seront supprimés.
Dans le cas contraire, tous les tuples de la relation seront supprimés.



Seules les suppressions respectant les contraintes sont exécutées

Exemples

DELETE FROM Reservation
WHERE NumClient **IN** (**SELECT** NumClient
 FROM client
 WHERE ville = 'Paris');

Toutes les réservations
des clients habitant Paris
seront supprimés.

DELETE FROM client
WHERE ville = ' Paris ';

Tous les clients habitant
Paris seront supprimés.

DELETE FROM client;

La relation Client est vidée de son
contenu. Si Client est référencée la
suppression est rejetée