

Introduction à SQL

Cours
Version 2.4

E. Desmontils

Faculté des sciences et des techniques
LINA / Dpt-Info / MIAGE, Bureau 110
Emmanuel.Desmontils@univ-nantes.fr

4 novembre 2012



Première partie I

Introduction



Historique

Les langages de SQL

Utiliser SQL

La base du cours



Introduction

- ▶ SQL : “Structured Query Language”, langage d’interrogation structuré (définition et manipulation de BDR) ;
- ▶ Norme ANSI ;
- ▶ Histoire :
 - ▶ Années 70 : SEQUEL (IBM),
 - ▶ 1981 : 1er SGBD sous SQL,
 - ▶ 1986 : Normalisation SQL/86,
 - ▶ 1989 : SQL/89 (la plus courante),
 - ▶ 1992 : SQL/92 ou SQL2 (beaucoup d’améliorations),
 - ▶ Actuellement, SQL3 (couche objet) ;
- ▶ Cours : SQL2.

Historique

Les langages de SQL

Utiliser SQL

La base du cours



Introduction

- ▶ SQL = DDL + DCL + DML (dans l'ordre d'utilisation) ;
 - ▶ DDL (Data Definition Language) : langage de définition de données (LDD)
 - ▶ Créer, Modifier ou Supprimer les définitions des tables (Create, Drop, Alter, Comment, Describe, Rename...)
 - ▶ DCL (Data Control Language) : langage de gestion des protections d'accès aux tables (LCD) en particulier dans des environnements multi-utilisateurs (Grant, Revoke)
 - ▶ DML (Data Manipulation Language) : Langage de manipulation des données (LMD) avec Select, Insert, Update, Delete.

Historique

Les langages de SQL

Utiliser SQL

La base du cours



Introduction

- ▶ Deux utilisations possibles :
 - ▶ Interactive à partir d'une console,
 - ▶ Par programme avec langage hôte (Langage C, Java, PHP...) ou langage spécifique (PL/SQL d'Oracle),
- ▶ SQL : ensemble d'instructions, chaque instruction est terminée par " ; ", un commentaire " /* ... */ " ou " % ... "

NB : BDR = Manipulation d'un ou plusieurs ensembles de t-uples.

Historique

Les langages de SQL

Utiliser SQL

La base du cours



Exemple de BDR

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle

Exemple de BDR

Matières	<i>codemat</i>	titre	responsable	diplôme
1	MIAS2I5	I5	E238	Deug MIAS
2	MIAS2I6	I6	E426	Deug MIAS
3	IUP2MA	Automates	E238	Licence IUP-MIAGE
4	LIL6	Systèmes	E236	Licence Informatique
5	IUP2IS	Systèmes	'26	Licence IUP-MIAGE
6	MIAS2I3	Math-Info	E426	Deug MIAS
7	LIL5	Algo	E426	Licence Informatique

Exemple de BDR

Notes	<i>noe</i>	<i>codemat</i>	noteex	notecc
1	99628C	MIAS2I5	12	15.5
2	46283B	MIAS2I6	8	11
3	46283B	MIAS2I5	9.5	2
4	86719E	IUP2MA	12	5.5
5	99321C	LIL6	18	16.5
6	28936E	MIAS2I5	13.5	13.5
7	86719E	IUP2IS	8.5	10
8	99628C	MIAS2I6	3	6
9	99321C	LIL5	15	14.5
10	99322C	MIAS2I5	12	15.5
11	28936E	MIAS2I6	12	null

Deuxième partie II

DML (Data Manipulation Language)



Requêtes simples

Projection et sélection

Tri des lignes

Expression des jointures

Fonctions statistiques

Regroupements

Requêtes imbriquées

Différents types

Opérations valides

Requête corrélée

Exemples de requêtes imbriquées

Opérations ensemblistes

Mise à jour d'une table

Remarques

Cas de NULL

Notion de transaction



Recherche de base

- ▶ Projection, Sélection
- ▶ Manipulation individuelle des t-uples
- ▶ Forme générale :

```
Select [ $\epsilon$ |All|Distinct] Clause_projection  
From Table [[As]Synonyme]  
Where Clause_selection  
Order By Clause_tri;
```

Recherche de base

```
Select [ $\epsilon$ |All|Distinct] Clause_projection  
From Table [[As]Synonyme] ;
```

- ▶ `Clause_projection ::= '*'`
 | `exp [[As]nom] [,exp[[As]nom]]*`
- ▶ `nom ::=` Chaîne de 30 caractères max. (permet de remplacer l'entête de colonne par 'nom' au lieu de 'exp' par défaut);
- ▶ `exp ::=` Expressions
 - ▶ nom d'attribut de la forme 'table.attribut' ou 'attribut' s'il n'y pas d'ambiguïté,
 - ▶ Calculs, constantes... sur des attributs avec les opérateurs classiques + 'year', 'Month', 'day', 'substring'...
- ▶ '*' remplace l'énumération de tous les attributs de la table définie dans le 'from' ;



Recherche de base

```
Select [ $\epsilon$ |All|Distinct] Clause_projection  
From Table [[As]Synonyme] ;
```

- ▶ 'All' et ' ϵ ' : tous les t-uples sont affichés (même les doublons) ;
- ▶ 'Distinct' : suppression des doublons (projection de l'algèbre relationnelle) ;

Recherche de base

```
Select [ $\epsilon$ |All|Distinct] Clause_projection  
From Table [[As]Synonyme] ;
```

- ▶ **Table** ::= table définie dans la base ou table construite à l'aide d'une sélection (un 'Select' imbriqué) éventuellement préfixée par le créateur de la base 'nomcreateur.nomtable' ;
- ▶ **Synonyme** : simplifier la syntaxe ou lever les ambiguïtés.

Exemple projection

► E_1 : Liste des matières

Select *

From Matières ;

Matières	codemat	titre	responsable	diplôme
1	MIAS2I5	I5	E238	Deug MIAS
2	MIAS2I6	I6	E426	Deug MIAS
3	IUP2MA	Automates	E238	Licence IUP-MIAGE
4	LIL6	Systèmes	E236	Licence Informatique
5	IUP2IS	Systèmes	E526	Licence IUP-MIAGE
6	MIAS2I3	Math-Info	E426	Deug MIAS
7	LIL5	Algo	E426	Licence Informatique

Exemple projection

- E_2 : Liste des numéros de responsable de matières

```
Select responsable
```

```
From Matières ;
```

responsable	
1	E238
2	E426
3	E238
4	E236
5	E526
6	E426
7	E426

- Attention aux doublons !

Exemple projection

- ▶ E'_2 : Liste des numéros de responsable de matières

```
Select Distinct responsable as "resp"
```

```
From Matières ;
```

resp	
1	E238
2	E426
3	E236
4	E526

- ▶ $\Pi_{responsable}(Matiere)$;
- ▶ Pour E_1 pas de "Distinct" nécessaire car la projection contient la clé (dont certainement pas de doublons).

Recherches de base

```
Select [ $\epsilon$ |All|Distinct] Clause_projection  
From Table [[As]Synonyme]  
Where Clause_selection;
```

- ▶ `Clause_selection ::= comparaison | Clause_selection op Clause_selection`
- ▶ `op ::= And | Or`
- ▶ `Comparaison ::= Not comparaison | exp cmp exp | exp Between exp And exp | exp Not Between exp And exp | exp Is null | exp Is Not null | exp In (exp, exp...)`
- ▶ `cmp ::= '=' | '!=' | ... | Like | Not Like`
- ▶ `exp ::= attribut ou expression numérique (avec les opérateurs classiques et d'autres).`



Recherches de base

- ▶ Sélection : Filtre les t-uples pour ne garder que ceux qui répondent à la sélection (ie à l'expression logique) ;
- ▶ “Like” : test l'égalité de deux chaînes en tenant compte de jokers dans la seconde expression :
 - ▶ “-” : remplace un caractère quelconque,
 - ▶ “%” : remplace 0 ou n caractères ;
- ▶ Possibilité d'utiliser des parenthèses pour clarifier ou pour modifier les propriétés ;
- ▶ Une expression peut être elle-même une table obtenue par un “Select...”.

Exemple sélection

- E_3 : Sélectionner les numéros d'étudiants ayant plus de 10 en CC de MIAS2I5

```
Select noe
```

```
From Notes
```

```
Where notecc >= 10 And codemat='MIAS2I5';
```

$\Pi_{noe}(\sigma_{notecc \geq 10 \wedge codemat='MIAS2I5'}(Notes))$	noe
1	99628C
2	28936E
3	99322C

Exemple sélection

- E_4 : Les étudiants dont le nom commence par un D et le numéro contient 93

```
Select *
```

```
From Etudiants
```

```
Where nom Like 'D%' And noetu Like '%93%';
```

	noetu	nom	prénom
1	28936E	Dupont	Franck
2	99321C	Denou	Michelle
3	99322C	Dupont	Isabelle

Recherches de base

```
Select [ $\epsilon$ |All|Distinct] Clause_projection  
From Table [[As]Synonyme]  
Where Clause_selection  
Order By Clause_tri;
```

- ▶ $\text{Clause_tri} ::= \text{colonne } [\epsilon|\text{ASC}|\text{DESC}]$
| $(, \text{colonne } [\epsilon|\text{ASC}|\text{DESC}])^*$;
- ▶ Permet d'ordonner les t-uples en fonction de la clause de tri.
Par défaut c'est "ASC";
- ▶ Sans la clause de tri : ordre des t-uples indéterminé ;
- ▶ A la place du nom de colonne, on peut donner son numéro.

Exemple tri

- E'_4 : Les étudiants dont le nom commence par un D et le numéro contient 93

```
Select *  
From Etudiants  
Where nom Like 'D%' And noetu Like  
'%93%'  
Order By nom;
```

```
Select *  
From Etudiants  
Where nom Like 'D%' And noetu Like  
'%93%'  
Order By 2;
```

	noetu	nom	prénom
1	99321C	Denou	Michelle
2	99322C	Dupont	Isabelle
3	28936E	Dupont	Franck

Exemple tri

- E'_4 : Les étudiants dont le nom commence par un D et le numéro contient 93

```
Select *  
  
From Etudiants  
  
Where nom Like 'D%' And noetu Like  
'%93%'  
  
Order By nom,prénom;
```

	noetu	nom	prénom
1	99321C	Denou	Michelle
2	28936E	Dupont	Franck
3	99322C	Dupont	Isabelle

```
Select *  
  
From Etudiants  
  
Where nom Like 'D%' And noetu Like  
'%93%'  
  
Order By 2,3;
```

Expression des jointures

```
Select *  
  
From Table1 [[As]Synonyme1] OpJoin  
Table2 [[As]Synonyme2]  
(On Cond_Join | Using (C1, ... ,Cn)) ;
```

- ▶ $OpJoin ::= (Left|Right|Full) \text{ Outer Join } | \text{ Cross Join } |$
 $[Natural] \text{ Join } | \text{ Inner Join}$
- ▶ $Cond_Join ::= att1 \text{ Op } att2 \text{ } | \text{ Cond_Join } (And \text{ } | \text{ Or})$
 $Cond_Join$

Expression des jointures

- ▶ Rappel : $R \bowtie_Q S = \sigma_Q(R \times S)$.

```
Select *  
From Table1 [[As]Synonyme1],  
Table2 [[As]Synonyme2]  
Where Cond_Join;
```

⇒

- ▶ “(+)” du côté de l'attribut où l'on garde les “null” pour la jointure externe.

Exemples de jointure

- ▶ E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure* entre "Etudiants" et "Notes" :
"noetu=noe"

→ Associer le bon nom à la bonne note

Exemples de jointure

- ▶ E_5 : Le **nom** des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure* entre "Etudiants" et "Notes" :
"noetu=noe"

→ Associer le bon nom à la bonne note

Exemples de jointure

- ▶ E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure entre "Etudiants" et "Notes"* :
"noetu=noe"

→ Associer le bon nom à la bonne note

Exemples de jointure

- ▶ E_5 : Le nom des étudiants ayant eu plus de 10 au CC du **module de code** MIAS2I5 par ordre décroissant des notes.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure entre "Etudiants" et "Notes"* :
"noetu=noe"

→ Associer le bon nom à la bonne note

Exemples de jointure

- ▶ E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par **ordre décroissant des notes**.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure entre "Etudiants" et "Notes"* :
"noetu=noe"

→ Associer le bon nom à la bonne note

Exemples de jointure

- ▶ E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure entre "Etudiants" et "Notes"* :
"noetu=noe"

→ Associer le bon nom à la bonne note

Exemples de jointure

- ▶ E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

⇒ nécessité d'utiliser la table "Etudiants" + Projection sur le nom

⇒ nécessité d'utiliser la table "Notes" + Sélection sur la valeur du CC

⇒ aussi dans la table "Notes" + Autre sélection sur le code

NB : attribut qui ne participe pas à la projection mais qui sert à trier les t-uples !

⇒ Nécessité d'une *jointure entre "Etudiants" et "Notes"* :
"noetu=noe"

→ Associer le bon nom à la bonne note



Exemples de jointure

- E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

```
Select nom  
  
From Etudiants Join Notes On noetu=noe  
  
Where notecc>=10 And codemat='MIAS2I5'  
  
Order By notecc Desc;
```

```
Select nom  
  
From Etudiants, Notes  
  
Where noetu=noe And notecc>=10  
  
And codemat='MIAS2I5'  
  
Order By notecc Desc;
```

nom	
1	Robert
2	Dupont
3	Dupont

Exemples de jointure

- E_5 : Le nom des étudiants ayant eu plus de 10 au CC du module de code MIAS2I5 par ordre décroissant des notes.

```
Select Distinct nom
From Etudiants Join Notes On noetu=noe
Where notecc>=10 And codemat='MIAS2I5'
Order By notecc Desc;
```

```
Select Distinct nom
From Etudiants, Notes
Where noetu=noe And notecc>=10
And codemat='MIAS2I5'
Order By notecc Desc;
```

nom	
1	Robert
2	Dupont

Exemples de jointure

- E_6 : Tous les étudiants (numéro) et la matière pour les étudiants ayant eu plus de 15 à un examen.

```
Select noe, titre  
From Notes as n Join Matieres as m  
On m.codemat=n.codemat  
Where noteex>=15;
```

```
Select noe, titre  
From Notes Natural Join Matieres  
Where noteex>=15;
```

	noe	titre
1	99321C	Système
2	99321C	Algo

Exemples de jointure

- E'_6 : Tous les étudiants (numéro, nom et prénom) ayant leur moyenne à l'examen de I5

```
Select noetu, nom, prenom
```

```
From Etudiants Join (Notes Natural Join Matieres) On noetu=noe
```

```
Where noteex>=10 And titre='I5';
```

	noetu	nom	prenom
1	99628C	Robert	Adrien
2	28936E	Dupont	Franck
3	99322C	Dupont	Isabelle

Les fonctions statistiques

- ▶ Effectuer des traitements statistiques sur les t-uples produits par l'expression SQL ;
- ▶ Forme : `f([ $\epsilon$ |All|Distinct] liste_attributs)`
- ▶ Avec comme fonctions possibles : AVG (moyennne), Count (nombre de t-uples), MAX (maximum), MIN (minimum), SUM (somme), STDDEV (écart-type), VARIANCE...
- ▶ Cas particulier du "Count" :
`Count(*)  $\equiv$  Count(All ...)`
- ▶ Les valeurs "null" ne sont pas prises en compte

Exemples

- E_7 : La moyenne au CC de I5.

```
Select AVG(notecc)
```

```
From Notes Natural Join Matieres
```

```
Where titre='I5';
```

AVG(notecc)	
1	11,625

Exemples

- E_8 : La moyenne de l'étudiant 99628C en MIAS2I5.

```
Select (noteex+notecc)/2 as moyenne
```

```
From Notes
```

```
Where noe='99628C' And codemat='MIAS2I5';
```

moyenne	
1	13,75

Exemples

- E_9 : Les nom, prénom et moyenne des étudiants reçus en I5.

```
Select nom, prenom, (noteex+notecc)/2 as moyenneI5
```

```
From Etudiants Join (Notes Natural Join Matieres) On noetu=noe
```

```
Where titre='I5' And (noteex+notecc)/2>=10;
```

	nom	prenom	moyenneI5
1	Robert	Adrien	13,75
2	Dupont	Franck	13,5
3	Dupont	Isabelle	13,75

Exemples

- E'_9 : Les nom, prénom et moyenne des étudiants en I6.

```
Select nom, prenom, (noteex+notecc)/2 as moyenneI6
```

```
From Etudiants Join (Notes Natural Join Matieres) On noetu=noe
```

```
Where titre='I6';
```

	nom	prenom	moyennel6
1	Dupont	Isabelle	9,5
2	Robert	Adrien	4,5
3	Dupont	Franck	null

Les regroupements

- ▶ Jusqu'ici : calculs sur l'ensemble des t-uples sélectionnés mais pas sur des groupes de t-uples ;
- ▶ Exemple : Comment traiter “La moyenne des notes d'examen pour chaque matière” ?
- ▶ Solution : les regroupements

```
Select ...  
  
From ...  
  
Group By Clause_regroupement  
  
Having Cond_groupes;
```

- ▶ Les t-uples ayant les attributs de la clause de regroupement identiques sont regroupé dans une “sous-table”. Les sous-tables sont filtrées par la condition sur les groupes (Cf. where).

Exemple

- ▶ E_{10} : La moyenne des notes d'examen pour chaque matière.

```
Select codemat, AVG(noteex)
From Notes
Group By codemat;
```

Exemple

- E_{10} : La moyenne des notes d'examen pour chaque matière.

```
Select codemat, AVG(noteex)
```

```
From Notes
```

```
Group By codemat;
```

Notes	<i>noe</i>	<i>codemat</i>	<i>noteex</i>	<i>notecc</i>
1	99628C	MIAS2I5	12	15.5
2	46283B	MIAS2I6	8	11
3	46283B	MIAS2I5	9.5	2
4	86719E	IUP2MA	12	5.5
5	99321C	LIL6	18	16.5
6	28936E	MIAS2I5	13.5	13.5
7	86719E	IUP2IS	8.5	10
8	99628C	MIAS2I6	3	6
9	99321C	LIL5	15	14.5
10	99322C	MIAS2I5	12	15.5
11	28936E	MIAS2I6	12	null

Exemple

- E_{10} : La moyenne des notes d'examen pour chaque matière.

```
Select codemat, AVG(noteex)
```

```
From Notes
```

```
Group By codemat;
```

Notes	<i>noe</i>	<i>codemat</i>	noteex	notecc
1	99628C	MIAS2I5	12	15.5
3	46283B	MIAS2I5	9.5	2
6	28936E	MIAS2I5	13.5	13.5
10	99322C	MIAS2I5	12	15.5
2	46283B	MIAS2I6	8	11
8	99628C	MIAS2I6	3	6
11	28936E	MIAS2I6	12	null
4	86719E	IUP2MA	12	5.5
5	99321C	LIL6	18	16.5
7	86719E	IUP2IS	8.5	10
9	99321C	LIL5	15	14.5

Exemple

- E_{10} : La moyenne des notes d'examen pour chaque matière.

```
Select codemat, AVG(noteex)  
From Notes  
Group By codemat;
```

	<i>codemat</i>	AVG(noteex)
1	MIAS2I5	11,75
2	MIAS2I6	7,67
3	IUP2MA	12
4	LIL6	18
5	IUP2IS	8.5
6	LIL5	15

Exemple

- E_{10} : La moyenne des notes d'examen pour chaque matière seulement pour les matières ayant plus de 10.

```
Select codemat, AVG(noteex)
From Notes
Group By codemat
Having AVG(noteex)>=10;
```

	<i>codemat</i>	AVG(<i>noteex</i>)
1	MIAS2I5	11,75
2	MIAS2I6	7,67
3	IUP2MA	12
4	LIL6	18
5	IUP2IS	8.5
6	LIL5	15

Exemple

- E_{10} : La moyenne des notes d'examen pour chaque matière seulement pour les matières ayant plus de 10.

	<i>codemat</i>	AVG(noteex)
1	MIAS2I5	11,75
3	IUP2MA	12
4	LIL6	18
6	LIL5	15

```
Select codemat, AVG(noteex)
From Notes
Group By codemat
Having AVG(noteex)>=10;
```

Exemple

- ▶ E_{11} : Les étudiants ayant leur année par ordre alphabétique des noms puis des prénoms.

```
Select noe, nom, prenom  
From Notes Join Etudiants On noetu=noe  
Group By noe, nom, prenom  
Having AVG((notecc+noteex)/2)>=10  
Order By nom, prenom;
```

- ▶ Généralement les attributs du “Group By” sont augmentés de ceux participant à la projection mais pas aux calculs.
- ▶ Ici, “noe” suffit pour le regroupement mais on y ajout “nom” et “prenom”.

Exemple

- E_{11} : Les étudiants ayant leur année par ordre alphabétique des noms puis des prénoms.

```
Select noe, nom, prenom
From Notes Join Etudiants On noetu=noe
Group By noe, nom, prenom
Having AVG((notecc+noteex)/2)>=10
Order By nom, prenom;
```

	noetu	nom	prenom
1	99321C	Denou	Michelle
2	28936E	Dupont	Franck
3	99322C	Dupont	Isabelle (Uniquement le I5!)

Autres Exemples

...



Requêtes simples

Projection et sélection

Tri des lignes

Expression des jointures

Fonctions statistiques

Regroupements

Requêtes imbriquées

Différents types

Opérations valides

Requête corrélée

Exemples de requêtes imbriquées

Opérations ensemblistes

Mise à jour d'une table

Remarques

Cas de NULL

Notion de transaction



Requêtes imbriquées

- ▶ Possibilité de remplacer une table (“From”) ou une expression (“Where” et/ou “Having”) par une requête (appelée *sous-requête*)
- ▶ Les opérateurs autorisés dépendent du type de réponse de la sous-requête :
 1. Un t-uple d'un attribut (une valeur),
 2. n t-uples d'un attribut (un ensemble de valeurs),
 3. Un t-uple de n attributs
 4. n t-uples de n attributs
 5. 0 ou n t-uples
- ▶ Attention, les blocs internes sont évalués en premier

Opérations valides

- ▶ 1 t-uple / 1 attribut : $\text{exp op (select ...)}$
avec op un opérateur de comparaison classique;
- ▶ n t-uples / 1 attribut : $\text{exp op' (select ...)}$
avec $\text{op'} \in \{ \text{In, Not In, op Any (à au moins un élément), op All (à tous les éléments)} \}$;
- ▶ $\text{In} = \text{Any}$
- ▶ $\text{Not In} = \text{!All}$
- ▶ 1 t-uple / n attributs :
 $(\text{exp1, exp2, ...}) \text{ op (select ...)}$
avec $\text{op} \in \{ =, != \}$
- ▶ n t-uples / n -attributs :
 $(\text{exp1, exp2, ...}) \text{ op' (select ...)}$

Opérations valides

- ▶ o-n t-uples :
Exists (select ...) (vrai si le select retourne au moins un t-uple)
et Not exists (select ...)
- ▶ $x \text{ op Any (select } y \text{ From } t \text{ Where } p) =$
Exists (Select * From t Where p and $x \text{ op } t.y$)
- ▶ $x \text{ op All (select } y \text{ From } t \text{ Where } p) =$
Not Exists (Select * From t Where p and Not($x \text{ op } t.y$))

Requête corrélée

- ▶ *Requête corrélée* ou *requête synchronisée* : dans la sous-requête intervient un ou plusieurs attributs de tables de la requête qui l'utilise.
- ▶ Très souvent utilisée avec l'opérateur "Exists"
- ▶ Attention : ce bloc est évalué pour chaque t-uple (ou groupe) $\Rightarrow$ très coûteux !

Exemples

cas 1 : Un t-uple d'un attribut

- ▶ E_{12} : Numéros des étudiants ayant eu plus que l'étudiant 99628C en examen de MIAS2I5.



Exemples

cas 1 : Un t-uple d'un attribut

- E_{12} : Numéros des étudiants ayant eu plus que l'étudiant 99628C en examen de MIAS2I5.

```
Select noe
```

```
From notes
```

```
Where noteex > (Select noteex From notes Where noe='99628C' And code-  
mat='MIAS2I5') And codemat='MIAS2I5';
```

Exemples

cas 1 : Un t-uple d'un attribut

- E_{12} : Numéros des étudiants ayant eu plus que l'étudiant 99628C en examen de MIAS2I5.

```
Select noe
```

```
From notes
```

```
Where noteex > (Select noteex From notes Where noe='99628C' And code-  
mat='MIAS2I5') And codemat='MIAS2I5';
```

noteex

1	12
---	----

Exemples

cas 1 : Un t-uple d'un attribut

- E_{12} : Numéros des étudiants ayant eu plus que l'étudiant 99628C en examen de MIAS2I5.

```
Select noe
```

```
From notes
```

```
Where noteex > (Select noteex From notes Where noe='99628C' And code-  
mat='MIAS2I5') And codemat='MIAS2I5';
```

noteex	
1	12

noe	
1	28936E



Exemples

cas 1 : Un t-uple d'un attribut (suite)

- ▶ E_{12} : Numéros des étudiants ayant eu plus que l'étudiant 99628C en examen de MIAS2I5.
- ▶ Cette requête peut aussi être calculée par une auto-jointure !

Exemples

cas 1 : Un t-uple d'un attribut (suite)

- ▶ E_{12} : Numéros des étudiants ayant eu plus que l'étudiant 99628C en examen de MIAS2I5.
- ▶ Cette requête peut aussi être calculée par une auto-jointure !

```
Select noe  
  
From notes as n1 Join notes as n2 on n1.no != n2.no  
  
Where n2.no='99628C' And codemat='MIAS2I5' and n1.noteex > n2.noteex;
```

noe	
1	28936E

Exemples

cas 2 : n t-uples d'un attribut

- ▶ E_{13} : La meilleure moyenne à l'examen pour les matières

Exemples

cas 2 : n t-uples d'un attribut

- E_{13} : La meilleure moyenne à l'examen pour les matières

```
Select AVG(noteex)
From notes
Group By codemat
Having AVG(noteex)>=All (Select AVG(noteex) From notes Group By code-
mat);
```

AVG(noteex)	
1	18

Exemples

cas 2 : n t-uples d'un attribut

- E_{13} : La meilleure moyenne à l'examen pour les matières

```
Select MAX(moyenne)
```

```
From (Select AVG(noteex) From notes Group By codemat) as x(moyenne) ;
```

x	max
1	18

Exemples (cas 2 : n t-uples d'un attribut) : suite

- ▶ E_{13} : Le numéro de l'étudiant ayant la meilleure moyenne générale

Exemples (cas 2 : n t-uples d'un attribut) : suite

- E_{13} : Le numéro de l'étudiant ayant la meilleur moyenne générale

```
Select noe,AVG((noteex+notecc)/2)
From notes
Group By noe
Having AVG((noteex+notecc)/2)>=All (Select AVG((noteex+notecc)/2) From
notes Group By noe);
```

Notes	<i>noe</i>	AVG((noteex+notecc)/2)
1	99321C	16

Exemples

cas 2 : n t-uples d'un attribut & cas 5 : 0 ou n t-uples

- ▶ E_{14} : Le nom et prénom des étudiants ayant passé les examens en I5

Exemples

cas 2 : n t-uples d'un attribut & cas 5 : 0 ou n t-uples

- E_{14} : Le nom et prénom des étudiants ayant passé les examens en I5

```
Select nom, prenom
```

```
From Etudiants
```

```
Where noetu In (Select noe From notes Where codemat='MIAS2I5' And noteex  
Is Not null);
```

Exemples

cas 2 : n t-uples d'un attribut & cas 5 : 0 ou n t-uples

- ▶ E_{14} : Le nom et prénom des étudiants ayant passé les examens en I5

```
Select [Distinct] nom, prenom  
From Etudiants Join Notes On noe=noetu  
Where codemat='MIAS2I5' And noteex Is Not null;
```

Exemples

cas 2 : n t-uples d'un attribut & cas 5 : 0 ou n t-uples

- ▶ E_{14} : Le nom et prénom des étudiants ayant passé les examens en I5

```
Select nom, prenom
```

```
From Etudiants e
```

```
Where Exists (Select * From notes Where e.noetu=noe And code-  
mat='MIAS2I5' And noteex Is Not null);
```

- ▶ NB : Deux nouvelles formulations de jointure !

Exemples

cas 2 : n t-uples d'un attribut & cas 5 : 0 ou n t-uples

- E_{14} : Le nom et prénom des étudiants ayant passé les examens en I5

	nom	prénom
1	Dupont	Franck
2	Dupont	Isabelle
3	Robert	Adrien
4	Dupont	Isabelle

Exemples

cas 3 : Un t-uple de n attributs

- E_{15} : Les étudiants (numéro) ayant les mêmes notes en examen et en cc que '99628C' en 'MIAS2I5'.

Exemples

cas 3 : Un t-uple de n attributs

- E_{15} : Les étudiants (numéro) ayant les mêmes notes en examen et en cc que '99628C' en 'MIAS2I5'.

```
Select noe  
  
From notes  
  
Where noe!='99628C' And (noteex, notecc) = (Select noteex, notecc From  
notes Where noe='99628C' And codemat='MIAS2I5');
```

Exemples

cas 3 : Un t-uple de n attributs

- E_{15} : Les étudiants (numéro) ayant les mêmes notes en examen et en cc que '99628C' en 'MIAS2I5'.

```
Select n1.noe  
  
From notes n1 Join notes n2 On n1.noteex=n2.noteex And  
n1.notecc=n2.notecc And n1.noe!=n2.noe  
  
Where n2.noe='99628C' And n2.codemat='MIAS2I5';
```

Exemples

cas 3 : Un t-uple de n attributs

- E_{15} : Les étudiants (numéro) ayant les mêmes notes en examen et en cc que '99628C' en 'MIAS2I5'.

noe	
1	99322C

Exemples

cas 4 : n t-uples de n attributs

- E_{15} : Les étudiants (numéro) ayant les mêmes notes en examen et en cc pour une même matière qu'un autre étudiant.

Exemples

cas 4 : n t-uples de n attributs

- E_{15} : Les étudiants (numéro) ayant les mêmes notes en examen et en cc pour une même matière qu'un autre étudiant.

```
Select noe  
From notes n  
Where (noteex, notecc, codemat) In (Select noteex, notecc, codemat From  
notes Where noe!=n.noe);
```

noe	
1	99322C

Requêtes simples

- Projection et sélection

- Tri des lignes

- Expression des jointures

- Fonctions statistiques

- Regroupements

Requêtes imbriquées

- Différents types

- Opérations valides

- Requête corrélée

- Exemples de requêtes imbriquées

Opérations ensemblistes

- Mise à jour d'une table

Remarques

- Cas de NULL

- Notion de transaction



Opérations ensemblistes

- ▶ Certains opérateurs de l'algèbre relationnelle peuvent être implémentés par des sous-requêtes ;
- ▶ La différence : Except ou

```
Select A1 as R1, A2 as R2...
```

```
From R
```

```
Where Not Exists (Select A1 as S1, A2 as S2... From S Where R1=S1 And  
R2=S2 And ...);
```

- ▶ L'intersection : Intersect ou

```
Select A1, A2...
```

```
From R
```

```
Where Exists (Select A1, A2... From S Where R.A1=S.A1 And ...);
```

- ▶ L'union : Union ou Union All (autorise les doublons)



Requêtes simples

Projection et sélection

Tri des lignes

Expression des jointures

Fonctions statistiques

Regroupements

Requêtes imbriquées

Différents types

Opérations valides

Requête corrélée

Exemples de requêtes imbriquées

Opérations ensemblistes

Mise à jour d'une table

Remarques

Cas de NULL

Notion de transaction



Mise à jour

- Insertion :

Insert Into Table (Col1, ..., Colp) Values (V1, ..., Vp);
OU Insert Into Table (Col1, ..., Colp) Select...;

- Impossible de référencer Table dans le Select...

- Modification :

Update Table Set Col1=Exp1 ... [Where cond_selection];
OU Update Table Set (Col1, ..., Colp) = (Select...) [Where
cond_selection];

- Suppression :

Delete From Table [Where cond_selection];



Exemples

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle

Exemples

- Insert Into Etudiants Values ('99226B', Lepauvre, Alban);

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle

Exemples

- Insert Into Etudiants Values ('99226B', Lepauvre, Alban);

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle
7	99226B	Lepauvre	Alban

Exemples

- ▶ Insert Into Etudiants Values ('99226B', Lepauvre, Alban);
- ▶ Update Etudiant Set noetu='99227B' Where noetu='99226B';

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle
7	99226B	Lepauvre	Alban

Exemples

- ▶ Insert Into Etudiants Values ('99226B', Lepauvre, Alban);
- ▶ Update Etudiant Set noetu='99227B' Where noetu='99226B';

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle
7	99227B	Lepauvre	Alban

Exemples

- ▶ Insert Into Etudiants Values ('99226B', Lepauvre, Alban);
- ▶ Update Etudiant Set noetu='99227B' Where noetu='99226B';
- ▶ Delete From Etudiant Where noetu='99227B'

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle
7	99227B	Lepauvre	Alban

Exemples

- ▶ Insert Into Etudiants Values ('99226B', Lepauvre, Alban);
- ▶ Update Etudiant Set noetu='99227B' Where noetu='99226B';
- ▶ Delete From Etudiant Where noetu='99227B'

Étudiants	<i>noetu</i>	nom	prénom
1	28936E	Dupont	Franck
2	46283B	Dupont	Isabelle
3	86719E	Martin	Adrien
4	99628C	Robert	Adrien
5	99321C	Denou	Michelle
6	99322C	Dupont	Isabelle

Requêtes simples

Projection et sélection

Tri des lignes

Expression des jointures

Fonctions statistiques

Regroupements

Requêtes imbriquées

Différents types

Opérations valides

Requête corrélée

Exemples de requêtes imbriquées

Opérations ensemblistes

Mise à jour d'une table

Remarques

Cas de NULL

Notion de transaction



Influence des valeurs indéfinies

- ▶ Valeur indéfinie, inconnue ou impossible : “NULL”
- ▶ Le résultat de la comparaison entre “null” et toute autre valeur (y compris “null”) n’est ni vrai, ni faux mais “null”.
- ▶ Conditionnelle : logique trivalente

Influence des valeurs indéfinies

- ▶ Valeur indéfinie, inconnue ou impossible : “NULL”
- ▶ Le résultat de la comparaison entre “null” et toute autre valeur (y compris “null”) n’est ni vrai, ni faux mais “null”.
- ▶ Conditionnelle : logique trivalente

And	V	I	F
V	V	I	F
I	I	I	F
F	F	F	F

Influence des valeurs indéfinies

- ▶ Valeur indéfinie, inconnue ou impossible : “NULL”
- ▶ Le résultat de la comparaison entre “null” et toute autre valeur (y compris “null”) n’est ni vrai, ni faux mais “null”.
- ▶ Conditionnelle : logique trivalente

Or	V	I	F
V	V	V	V
I	V	I	I
F	V	I	F

Influence des valeurs indéfinies

- ▶ Valeur indéfinie, inconnue ou impossible : “NULL”
- ▶ Le résultat de la comparaison entre “null” et toute autre valeur (y compris “null”) n’est ni vrai, ni faux mais “null”.
- ▶ Conditionnelle : logique trivalente

	Not
V	F
I	I
F	V

Notion de transaction

- ▶ *Transaction* : un ensemble cohérent et sensible de modifications de la base
 - ▶ contrôle des opérations lors d'accès concurrents,
 - ▶ risque de rupture de la connexion
- ▶ Niveau d'isolation Set Transaction... ;
 - ▶ Par exemple, est-ce que l'on autorise des consultations pendant la transaction ? etc.

Notion de transaction

- ▶ Transactions sans "Auto Commit" :
 - ▶ Le système ne fait rien de définitif si l'ordre n'est pas donné.
 - ▶ Valider une transaction : `Commit`
les modifications sont alors définitives et visibles des autres ;
 - ▶ Annuler une transaction : `Rollback`.

Notion de transaction

- ▶ Transactions avec "Auto Commit" :
 - ▶ Par défaut, chaque requête est automatiquement effective définitivement ("commit" implicite et systématique)
Cas d'Oracle, de SQL Serveur, de PostgreSQL...
 - ▶ Débuter une transaction : `Begin Transaction`
`nom_transaction,`
 - ▶ Valider une transaction : `Commit nom_transaction`
les modifications sont alors définitives et visibles des autres,
 - ▶ Annuler une transaction : `Rollback nom_transaction;`



Troisième partie III

DDL (Data Definition Language)



Création d'une table

- Définition de la table

- Définition d'un domaine

Contraintes d'intégrité

- Contraintes sur les attributs

- Contraintes liées aux clés

- Contraintes inter-tables

Autres commandes



Création d'une table

- ▶ Creation : `Create Table latable (col1 type1 [Cont1], col2 type2 [Cont2]...) [As Select ...];` ou `Create Table latable [(col1 type1 [Cont1], col2 type2 [Cont2]...)] As Select ...;`
- ▶ Les types :
 - ▶ Types numériques : Integer, Smallint, Numeric, Decimal, Real, Double, Float... Parfois Number, Number(taille_maxi) ou Number(taille_maxi,Decimales) sous Oracle,
 - ▶ Chaîne de caractères : Char(long) qui complète par des espaces, Varchar(long) pour une longueur variable,
 - ▶ Date : XX/YY/ZZ (fr) ou XX-YYY-ZZ (en)
 - ▶ Time, Timestamp



Création d'une table

- ▶ Domaine $\sim$ type
- ▶ Créer un domaine : `Create Domain nom type [Default ...];`
- ▶ Supprimer un domaine : `Drop Domain nom;`
- ▶ Utilisation : `Domain(nom)`

Création d'une table

- Définition de la table

- Définition d'un domaine

Contraintes d'intégrité

- Contraintes sur les attributs

- Contraintes liées aux clés

- Contraintes inter-tables

Autres commandes



Les contraintes d'intégrité

- ▶ Contrainte sur un attribut : après le type ;
 - ▶ Primary key : clé primaire de la table,
 - ▶ Not Null : interdiction d'avoir la valeur Null pour cet attribut,
 - ▶ Default Val avec Val comme une constante, User, Null, current_date, current_time...
 - ▶ Unique
 - ▶ Check(cond) avec la condition vérifiée si l'attribut n'est pas null, cette condition pouvant contenir un Select....
- ▶ Contrainte sur plusieurs attributs : après la déclaration des attributs, de la forme
Constraint Nom Contrainte

Les contraintes liées aux clés

- ▶ `Check(cond)`
- ▶ `Primary Key (col1, ...,coln)` : clé primaire de la table (les colonnes ne doivent pas autoriser `null`) ;
- ▶ `Unique (col1, ...,coln)` : interdit qu'une colonne ou la concaténation des colonnes) contienne 2 valeurs identiques ;
- ▶ `Foreign Key (col1, ...,coln) References tableref [(col1, ...,coln)] [on relation]` : détermine une clé étrangère qui est clé primaire (ou Unique) dans `tableref` (aux noms près) : on insère un t-uple que si la concaténation des colonnes est une clé d'un t-uple de la table de référence.

Les contraintes définissant les dépendances entre tables

- ▶ `relation` : Gestion des mises à jours de la table (ajout et suppression de t-uples).
`On (Delete|Update) (Restrict|Cascade|Set Null|Set Default V)`
- ▶ Par exemple : `On Delete Cascade`, ie si on supprime dans `tableref` alors on le fait aussi ici...
- ▶ Possibilité de contraintes plus générales : `Create Assertion lacontrainte check(...)`

Création d'une table

- Définition de la table

- Définition d'un domaine

Contraintes d'intégrité

- Contraintes sur les attributs

- Contraintes liées aux clés

- Contraintes inter-tables

Autres commandes



Autres

- ▶ Consulter la définition d'une table : `Describe latable;`
- ▶ Modifier la table :
`Alter Table latable (Add|Modify) (col1 type1, ...,coln typen);`
ou `Alter Table latable Drop Col;`
- Attention : Not Null impossible si la table contient des t-uples.
- Permet aussi d'ajouter ou supprimer des contraintes
- ▶ Supprimer une table : `Drop Table latable;`
- ▶ Commenter une table : `Comment on Table (latable|latable.col) is ...;`
- ▶ Notions de Vues et d'index



Quatrième partie IV

DCL (Data Control Language)



DCL : gérer le “multi”

Droits sur les object d'une base



- ▶ Environnement multi-utilisateur
- ▶ Contrôles de l'utilisation de la base
 - ▶ Confidentialité
 - ▶ Cohérence
 - ▶ Concurrency
 - ▶ Simplicité...
- ▶ Gestion des accès concurrents...
- ▶ Changement du mot de passe :
Grant Connect To utilisateur By motdepasse;

DCL : gérer le “multi”

Droits sur les object d'une base



- ▶ Accorder des droits :
`Grant Liste_privilèges On liste_objets To
(public|liste_utilisateurs) [With Grant Option]`
- ▶ Privilèges : `Select, Insert, Update [(col1,...,coln)],
Delete, Alter, Index, All`
- ▶ `With Grant Option` : peut transmettre ses privilèges à d'autres
- ▶ Retirer les droits : `Revoke [Grant Option For] privileges
On objets From utilisateurs;`
- ▶ Retirer un privilège = retirer ce privilège à tous les utilisateurs
qui l'ont obtenu par son intermédiaire.