Le langage SQL

Introduction

SQL = Structured Query Language

Développé, durant les années 70, dans les laboratoires d'IBM pour le système relationnel expérimental SYSTEM-R .

Devenu un standard pour les SGBD relationnels.

SQL comporte quatre groupes d'instructions:

1. les instructions d'interrogation de données,

2. les instructions de définition de données pour la création de tables, d'introduction de nouveaux attributs dans la les tables existantes et la création des index,

3. les instructions de manipulation des données par l'insertion, la suppression et la modification des données,

4. les instructions de contrôles pour la mise en place et la suppression des autorisations d'accès aux données de la base.

Exemples d'instructions de définition de données

Création d'une table :

Create table Bateau(Nbat number(3) Primary key, Nombat char(40), Sponsor char(40)) ;

Ajout d'une colonne à une table :

Alter table Bateau Add (NbComp number(3)) ;

...

Exemples d'instructions de Manipulation des données

Insertion de tuple : **Insert into Bateau values (102, 'TASSILI', 'DJEZZY') ;**

Suppression de tupes : **Delete from Bateau where nbat=200 ;**

Modification de valeurs : **Update Bateau Set Sponsor='NEDJMA' Where Nbat=130 ;**

Exemple d'instruction de contrôle

Grant All privileges to user1 ;

Exemple d'instructions d'interrogation de données

SELECT Nbat From Bateau Where Sponsor='DJEZZY'

On s'intéresse dans ce chapitre aux instructions d'interrogation.

Syntaxe :

Une requête d'interrogation est représentée syntaxiquement comme un bloc de qualification :

Select <liste d'attributs>
From < nom-relation>
[**Where** <prédicat>]
[**Group By** <attribut>]
[**Having** <prédicat>]
[**Order By** <attribut>[Desc]] ;

Projection

La forme la plus simple du Select

Select <liste d'attributs>
From < nom-relation>;

Permet d'extraire de la table une sous-table obtenue par projection selon le (ou les) attribut(s) précisé(s).

Exemple

SELECT NomBat FROM Bateau; Donne les Noms de tous les Bateaux

BATEAU

Nbat	Nombat	Sponsor
102	TASSILI	DJEZZY
103	EL BAHDJA	BNA
104	LA COLOMBE	NEDJMA
105	HOGGAR	BNA

SELECT NomBat FROM Bateau;

Nombat
TASSILI
EL BAHDJA
LA COLOMBE
HOGGAR

Remarque :

L'option * (Astérix) à la place de <liste d'attributs> permet l'édition de la table dans sa totalité.

SELECT * FROM Bateau;

Nbat	Nombat	Sponsor
102	TASSILI	DJEZZY
103	EL BAHDJA	BNA
104	LA COLOMBE	NEDJMA
105	HOGGAR	BNA

Remarque

La projection dans SQL n'élimine pas les doubles dans le résultat. Si on désire le faire, on doit l'exprimer explicitement à l'aide du mot clé **UNIQUE** ou **DISTINCT**.

Exemple

Donner la liste des sponsors :

BATEAU

Nbat	Nombat	Sponsor
102	TASSILI	DJEZZY
103	EL BAHDJA	BNA
104	LA COLOMBE	NEDJMA
105	HOGGAR	BNA

SELECT Sponsor
FROM Bateau;

Sponsor
DJEZZY
BNA
NEDJMA
BNA

SELECT UNIQUE Sponsor
FROM Bateau;

Sponsor
DJEZZY
BNA
NEDJMA

Restriction

La clause WHERE du SELECT permet de définir une condition que doit vérifier les tuples à sélectionner

BATEAU

Nbat	Nombat	Sponsor
102	TASSILI	DJEZZY
103	EL BAHDJA	BNA
104	LA COLOMBE	NEDJMA
105	HOGGAR	BNA

SELECT * FROM Bateau
WHERE Sponsor = BNA ;

Nbat	Nombat	Sponsor
103	EL BAHDJA	BNA
105	HOGGAR	BNA

Exemple de Restriction + projection

1. Donner les numéros et les noms des bateaux sponsorisés par la BNA

Select Nbat, NomBat **From** Bateau **Where** Sponsor='BNA' ;

BATEAU

Nbat	Nombat	Sponsor
102	TASSILI	DJEZZY
103	EL BAHDJA	BNA
104	LA COLOMBE	NEDJMA
105	HOGGAR	BNA

SELECT Nbat, NomBat FROM Bateau
WHERE Sponsor = BNA ;

Nbat	Nombat
103	EL BAHDJA
105	HOGGAR

la clause Where

Le prédicat de la clause Where peut inclure les opérateurs de comparaison : **=, <>, <, >,>=, <=** et les opérateurs booléens **And, Or, Not**.

D'autres opérateurs ont été introduits dans les dernières versions d'Oracles, nous citons : Between, IN, Like.

- **Between** : exp BETWEEN val1 et val2 : vrai si exp est comprise entre val1 et val2, faux sinon.

• **IN** : exp IN (val1, val2, val3...) vrai si exp est égale à l'une des valeurs de la liste entre parenthèses.

• **LIKE** : attribut LIKE <chaîne> où chaîne est une chaîne de caractère pouvant contenir un caractère joker '-' qui remplace un caractère quelconque ou bien '%' qui remplace une chaîne de longueur quelconque.

Exemple : Condition composée

Les numéros des compétitions de "Courses de la liberté" qui se sont déroulées après 2005.

COMPETITION

Ncomp	Nomcomp	Datcomp	Prixcomp
200	LE GRAND TOUR	21/03/2000	1000000
210	COURSE DE LA LIBERTE	05/05/2004	1000000
215	LE GRAND TOUR	20/03/2005	1100000
220	TROPHEE BARBEROUSSE	01/08/2005	1500000
240	COURSE DE LA LIBERTE	10/05/2007	1500000
260	TROPHEE BARBEROUSSE	01/08/2009	2000000
265	LE GRAND TOUR	21/03/2010	2000000
270	COURSE DE LA LIBERTE	08/05/2010	1800000

Requête :

```
SELECT Ncomp
FROM Compétition
WHERE NomComp = 'COURSES DE LA LIBERTE'
      AND DatComp > '31/12/2005' ;
```

Résultat de la requête :

Ncomp
240
270

Exemple avec BETWEEN

1. Donner les compétitions dont le prix est compris entre 1500000 et 2000000

COMPETITION

Ncomp	Nomcomp	Datcomp	Prixcomp
200	LE GRAND TOUR	21/03/2000	1000000
210	COURSE DE LA LIBERTE	05/05/2004	1000000
215	LE GRAND TOUR	20/03/2005	1100000
220	TROPHEE BARBEROUSSE	01/08/2005	1500000
240	COURSE DE LA LIBERTE	10/05/2007	1500000
260	TROPHEE BARBEROUSSE	01/08/2009	2000000
265	LE GRAND TOUR	21/03/2010	2000000
270	COURSE DE LA LIBERTE	08/05/2010	1800000

Requête :

```
SELECT *
FROM Compétition
WHERE Prixcomp BETWEEN 1500000 AND 2000000 ;
```

Résultat de la requête :

Ncomp	Nomcomp	Datcomp	Prixcomp
220	TROPHEE BARBEROUSSE	01/08/2005	1500000
240	COURSE DE LA LIBERTE	10/05/2007	1500000
260	TROPHEE BARBEROUSSE	01/08/2009	2000000
265	LE GRAND TOUR	21/03/2010	2000000
270	COURSE DE LA LIBERTE	08/05/2010	1800000

Exemple avec IN

Donner les bateaux qui ont occupés les 3 premières places :

COURSES

Nbat	Ncomp	Score
102	210	2
102	240	1
102	270	4
103	210	4
103	215	3
104	200	2
104	210	1
104	215	2
104	220	4
104	240	3
104	260	5
104	265	1
104	270	3
105	210	3
105	220	1

Requête :

```
SELECT *
FROM courses
WHERE Score IN ( 1 , 2 , 3 ) ;
```

Résultat de la requête

Nbat	Ncomp	Score
102	210	2
102	240	1
103	215	3
104	200	2
104	210	1
104	215	2
104	240	3
104	265	1
104	270	3
105	210	3
105	220	1

Exemple avec LIKE

Donner le nom des participants dont le nom commence par la lettre 'M' :

PARTICIPANTS

Npart	Nompart	Nbat
320	MOHAMMED	104
470	ALI	103
601	OMAR	102
720	MUSTAFA	105

Requête :

```
SELECT  *
FROM Participants
 WHERE  Nompart LIKE ' M % ' ;
```

Résultat de la requête :

Npart	Nompart	Nbat
320	MOHAMMED	104
720	MUSTAFA	105

Autre Exemple avec like

Donner le nom des participants dont le nom comprend la lettre M en deuxième position

PARTICIPANTS

Npart	Nompart	Nbat
320	MOHAMMED	104
470	ALI	103
601	OMAR	102
720	MUSTAFA	105

Requête :

```
SELECT  *
FROM Participants
 WHERE  Nompart LIKE  ' - M % ' ;
```

Résultat de la requête :

Npart	Nompart	Nbat
601	OMAR	102

Le tri du résultat d'un SELECT

Le résultat d'un select peut être trié dans l'ordre ascendant ou descendant en fonction d'une ou plusieurs colonnes.

Les critères du tri sont spécifiés par la clause **ORDER BY** dont la syntaxe est :

```
ORDER BY <attribut1> [DESC][,<attribut2> [DESC],...]
```

Le tri se fait d'abord selon la première colonne spécifiée dans l'order by, puis les lignes ayant la même valeur dans la première colonne sont triées selon la deuxième colonne de l'order by etc....

Le tri peut être ascendant ou descendant pour chaque colonne. Par défaut, il est ascendant.

Exemple

Donner la liste des compétions triée sur le nom et le numéro

```
SELECT *
FROM Compétition
ORDER BY Nomcomp, ncomp;
```

COMPETITION

Ncomp	Nomcomp	Datcomp	Prixcomp
200	LE GRAND TOUR	21/03/2000	1000000
210	COURSE DE LA LIBERTE	05/05/2004	1000000
215	LE GRAND TOUR	20/03/2005	1100000
220	TROPHEE BARBEROUSSE	01/08/2005	1500000
240	COURSE DE LA LIBERTE	10/05/2007	1500000
260	TROPHEE BARBEROUSSE	01/08/2009	2000000
265	LE GRAND TOUR	21/03/2010	2000000
270	COURSE DE LA LIBERTE	08/05/2010	1800000

SELECT * FROM Compétition
ORDER BY NomComp , Ncomp ;

Ncomp	Nomcomp	Datcomp	Prixcomp
210	COURSE DE LA LIBERTE	05/05/2004	1000000
240	COURSE DE LA LIBERTE	10/05/2007	1500000
270	COURSE DE LA LIBERTE	08/05/2010	1800000
200	LE GRAND TOUR	21/03/2000	1000000
215	LE GRAND TOUR	20/03/2005	1100000
265	LE GRAND TOUR	21/03/2010	2000000
220	TROPHEE BARBEROUSSE	01/08/2005	1500000
260	TROPHEE BARBEROUSSE	01/08/2009	2000000

Opération de jointure

Les noms des compétitions dans les quelles a participé le bateau n° 105

```
SELECT NomComp
FROM Compétition, Couses
WHERE Compétition.Ncomp = Couses.Ncomp AND Nbat = 105 ;
```

La jointure en SQL peut s'exprimer d'autres manières :

Jointure avec IN et NOT IN

s'exprime en imbriquant des blocs de sélection en utilisant des opérateurs d'appartenance (IN, NOT IN)

Exemple

Les noms des compétitions dans les quelles a participé le bateau n° 105 :

```
SELECT NomComp
FROM Compétition
WHERE Ncomp IN (SELECT Ncomp
                FROM Course
                WHERE Nbat = 105) ;
```

Avec NOT IN

Les numéros des bateaux qui ne participent pas aux compétitions LE GRAND TOUR

```
SELECT nbat
FROM courses
WHERE Ncomp NOT IN (SELECT Ncomp
                    FROM COMPETITION
                    WHERE NomComp = 'LE GRAND TOUR' ;
```

Jointure en utilisant EXISTS / NOT EXISTS

Le prédicat **EXISTS** contient une sous-interrogation qui, associée à la condition du EXISTS, peut être évaluée comme vraie ou fausse. Le prédicat EXISTS peut être utilisé à chaque fois qu'une interrogation employant le prédicat IN peut être utilisée. Il est également possible de formuler une interrogation avec **NOT EXISTS**.

Exemple

Les noms des compétitions dans les quelles a participé le bateau n° 105 :

```
SELECT NomComp
FROM Compétition
WHERE EXISTS (SELECT *
              FROM Course
              WHERE Nbat = 105 ) ;
```

Les numéros des bateaux qui ne participent pas aux compétitions LE GRAND TOUR

```
SELECT nbat
FROM courses
WHERE NOT EXISTS (SELECT *
                  FROM COMPETITION
                  WHERE (NomComp='LE GRAND TOUR')
                        and (courses.ncomp=competition.ncomp));
```

On peut utiliser des variables synonymes :

```
SELECT nbat
FROM courses x
WHERE NOT EXISTS (SELECT *  FROM COMPETITION y
        WHERE (NomComp='LE GRAND TOUR') and (x.ncomp=y.ncomp));
```

Expression de la division

Pour exprimer la division, SQL utilise une recherche avec NOT EXIST

Exemple

Noms des bateaux ayant participé à toutes les courses

```
SELECT NomB
FROM Bateau b
WHERE NOT EXISTS (SELECT *
FROM COMPETITION c
WHERE NOT EXISTS (SELECT *
FROM COURSES x
WHERE x.ncomp = c,ncomp  AND x.nbat = b.nbat) );
```

Cette requête peut être paraphrasée en : noms des bateaux tels qu'il n'existe pas de courses aux quelles ils n'ont pas participé.

UNION, INTERSECT et MINUS

SQL utilise les opérateurs ensemblistes **UNION** pour l'union, **INTERSECT** pour l'intersection et **MINUS** pour la différence ("Moins").

Une requête avec ces opérateurs s'écrit de la manière suivante :

```
SELECT .... FROM ……
    UNION / INTERSECT / MINUS
SELECT ... FROM ......
```

Exemple

Donner les numéros des bateaux qui ne participent à aucune compétition

```
(SELECT Nbat FROM BATEAU) MINUS (SELECT Nbat FROM COURSES);
```

Remarque

Dans une interrogation utilisant des opérateurs ensemblistes :

1. Les SELECT doivent avoir le même nombre de colonnes sélectionnées, et leurs types doivent être un à un identiques.
2. Les doublons sont éliminés, i.e., l'option UNIQUE/DISTINCT est implicite.
3. Les noms des colonnes titres sont ceux du premier Select
4. On peut combiner le résultat de plusieurs SELECT. P, ex. SELECT1 UNION SELECT2 INTERSECT SELECT 3 MINUS SELECT4 ; Dans ce cas l'expression sera évaluée en combinant les deux premiers SELECT à partir de la gauche puis le résultat avec le troisième SELECT etc.
5. L'ordre d'évaluation peut être précisé en utilisant les parenthèses

Exemple

SELECT1 UNION SELECT2 INTERSECT SELECT 3 MINUS SELECT4 ; s'exécute dans cet ordre :

(((SELECT1 UNION SELECT2) INTERSECT SELECT 3) MINUS SELECT4) ;

Opérateurs d'évaluation d'ensemble

SQL offre la possibilité d'utiliser sur un ensemble d'opérateurs de calcul:

COUNT : pour le dénombrement des éléments

SUM : pour la somme des éléments

AVG : pour la moyenne des éléments

MAX : pour la recherche du maximum

MIN : pour la recherche du minimum

Exemples

Donner le nombre total des bateaux s'écrit : SELECT **COUNT(*)** FROM Bateau ;

Donner le total des prix des compétitions TROPHEE BARBEROUSSE

```
SELECT  SUM(Prixcomp)
FROM Competition
WHERE Nomcomp = 'TROPHEE BARBEROUSSE' ;
```

Donner le plus grand prix donné dans une compétition

```
SELECT  MAX(Prixcomp)
FROM Competition ;
```

Donner le numéro, le nom des compétitions ayant donné le plus grand prix

```
SELECT  Ncomp, Nomcomp
FROM Competition
WHERE  Prixcomp =  (SELECT  MAX(Prixcomp)
                    FROM Competition) ;
```

Calcul sur plusieurs groupes : GROUP BY

Il est possible de subdiviser la table en groupe, chaque groupe étant l'ensemble des lignes ayant une valeur commune. Ceci s'effectue par :

```
GROUP BY exp1 [,exp2,...]
```

Qui définit les groupes comme les ensembles de lignes pour lesquelles exp. prend la même valeur.

Syntaxe

```
SELECT att. [, att., ...] FROM relation [, relation,...]
[WHERE prédicat]
GROUP BY exp. [, exp.,...]
```

Exemple

Requête donnant pour chaque bateau, le nombre de compétitions auxquels il a participé :

```
SELECT nbat, Count(*)
FROM Courses
GROUP BY nbat ;
```

Le nombre de bateaux ayant participé dans chaque compétition

```
SELECT ncomp, Count(*)
FROM Courses
GROUP BY ncomp ;
```

La figure suivante montre le regroupement de COURSES selon nbat et ncomp

GROUP BY Nbat

Nbat	Ncomp	Score
102	210	2
102	240	1
102	270	4
103	210	4
103	215	3
104	200	2
104	210	1
104	215	2
104	220	4
104	240	3
104	260	5
104	265	1
104	270	3
105	210	3
105	220	1

GOUP BY Ncomp

Ncomp	Nbat	Score
210	102	2
210	103	4
210	104	1
210	105	3
240	102	1
240	104	3
270	102	4
270	104	3
215	103	3
215	104	2
200	104	2
220	104	4
220	105	1
260	104	5
265	104	1

Un select de groupe avec une clause GROUP BY donnera une ligne résultat pour chaque groupe.

Ainsi la requête SELECT nbat, Count(*) FROM Courses GROUP BY nbat ; donne le résultat suivant :

```
102  3
103  2
104  8
105  2
```

Cohérence du résultat :

Dans la liste des colonnes résultats d'un SELECT de groupe ne peuvent figurer que des caractéristiques de groupe, i.e.,

1. des fonctions de groupe
2. des expressions figurant dans le GROUP BY

Sélection des groupes : clause HAVING

De la même façon qu'il est possible de sélectionner certaines lignes au moyen d'une clause WHERE, il est possible dans un SELECT de groupe de sélectionner certains groupes par la clause HAVING qui se place après le GROUP BY.

Syntaxe

```
SELECT att. [, att., ...] FROM relation [, relation,...]
[WHERE prédicat]
GROUP BY exp. [, exp.,...]
HAVING PREDICAT ;
```

Le prédicat figurant dans la clause HAVING suit les mêmes règles de syntaxe qu'un prédicat figurant dans la clause WHERE. Cependant, il ne peut porter que sur des caractéristiques de groupes : fonction de groupe ou expressions figurant dans la clause GROUP BY.

Exemple

Requête donnant les numéros des bateaux ayant participé à au moins 3 courses :

```
SELECT nbat, Count(*)
FROM Courses
GROUP BY nbat
HAVING Count(*) ≥ 3 ;
```

Cette requête donne le résultat suivant :

```
102  3
104  8
```

Langage Prédicatif

II

Le langage prédicatif est fondé sur le calcul des prédicats du premier ordre.

A. Rappels sur la logique du premier ordre.

La logique du premier ordre est avant tout un langage avec sa syntaxe et son interprétation.

La syntaxe du langage comporte :

1. La spécification d'un alphabet de symboles (vocabulaire de base),
2. La définition de termes et d'expressions utiles appelées formules bien formées (ou formules) qui peuvent être construites à partir de ces symboles.

Les symboles :

1. Les parenthèses (,)
2. Un ensemble de constantes notées : a, b, c, …
3. Un ensemble de variables notées : x, y, z, …
4. Un ensemble de prédicats à n arguments notés : P, Q, R, …
5. Un ensemble de fonctions à m arguments notées : f, g, h, …
6. Un ensemble de connecteurs logiques : $\rightarrow$ (implication), $\neg$ (négation), $\wedge$ ("et" logique), $\vee$ ("ou" logique)
7. Les quantificateurs : Existentiel $\exists$ et Quantitatif $\forall$

Définition d'un Terme

A l'aide de ce vocabulaire, on fabrique des termes de la façon suivante :

1. Les symboles de constantes et de variables sont des termes,
2. Si f est un symbole de fonction et t1, t2, t3, ..., tn sont des termes alors f (t1, t2, ..., tn) est un terme.

Définition d'un Prédicat

Un prédicat se définit en considérant un ensemble X et une variable x qui parcourt les éléments de X. On appelle prédicat P(x) une forme d'énoncé qui devient une proposition pour chaque valeur affectée à x, i.,e., que son évaluation conduit à lui attribuer la notion de vrai ou de faux.

Définition d'une formule atomique

Une formule atomique est définie comme un prédicat à m arguments, dont les arguments sont des termes : si P est un prédicat et t1, t2, ..., tm sont des termes alors P(t1, t2, ..., tm) est une formule atomique.

Définition d'une Formule bien formée :

-Une formule atomique est une formule bien formée (fbf).

Si f1 et f2 sont des fbfs alors

$f1 \wedge f2,\quad f1 \vee f2,\quad \neg f1,\quad f1 \rightarrow f2$
$(f1),\quad \exists x f1,\quad \forall x f1$

} sont des fbfs

Définition d'une Variable libre :

Une variable est **libre** dans une formule si elle n'est soumise à aucun quantificateur.

Définition d'une Formule fermée/ouverte :

Une formule est **fermée** si toutes les variables sont quantifiées sinon elle est **ouverte**.

Quelques équivalences logiques.

- $\neg \neg P \equiv P$
- $P \vee Q \equiv \neg(\neg P \wedge \neg Q)$
- $P \wedge Q \equiv \neg(\neg P \vee \neg Q)$
- $P \rightarrow Q \equiv \neg P \vee Q$ (se lit si P alors Q)
- $\forall x\, P(x) \equiv \neg \exists x \neg P(x)$
- $\forall x\, P(x)$ signifie (tout élément a la propriété P)
- $\exists x\, P(x)$ signifie (il y a au moins un élément qui a la propriété P)

B. Interprétation

La syntaxe du langage étant définie, il s'agit de donner une signification aux formules. Cela revient à les interpréter comme des assertions sur le domaine de discours. Dans les bases de données relationnelles, le domaine de discours est l'ensemble des occurrences de la base, i. e, l'ensemble des domaines des attributs des relations.

Deux grandes classes de langages prédicatifs ont été développées qui se distinguent essentiellement par l'interprétation des variables.

- **Langages prédicatifs à variables n-uplets** dont les variables sont interprétées comme des n-uplets de la BD
- **Langages prédicatifs à variables domaines** dont les variables sont interprétées comme des domaines des attributs de la BD.

Définition : Le calcul relationnel à variables n-uplets

Le calcul relationnel à variables n-uplets se déduit du calcul des prédicats en interprétant les formules bien formées comme suit :

1. Les variables sont associées à des n-uplets

2. Les seuls termes considérés sont :

-les constantes associées aux valeurs des domaines des attributs, par exemple 'rouge', et

-les fonctions génératrices d'attributs notées v.A où v est une variable et A un attribut (par exemple v.couleur).

3. Les prédicats utilisés sont ceux permettant les comparaisons logiques.

Syntaxe

On défini la syntaxe du calcul relationnel à variables n-uplets sous la Forme Normal de Backus (B. N. F.) comme suit

Requête ::= { Projection / Formule }

Formule ::= Defvar / Defvar Λ Qualification

Defvar ::= Relation(variable) / Defvar Λ Relation(variable)

Qualification ::= Terme θ Terme / Terme θ Constante / Qualification V Qualification / ¬ Qualification / ∃ Variable Qualification / ∀ Variable Qualification

Terme ::= Variable . Attribut

Projection :: = Terme / projection , Terme

θ ::= = / ≠ / < / > / ≤ / ≥.

Variable ::= a / b / ... / z

Exemples :

1. Donner les informations sur les fournisseurs :

   { f / Fournisseur(f) }

2. Quels sont les fournisseurs localisés à Alger?

   { f / Fournisseur(f) ∧ f.Ville = "Alger" }

3. Liste des noms et des matériaux de toutes les pièces :

   { p.nom, p.matériau / Pièce(p) }

Remarque

Toute variable apparaissant dans le critère de projection doit être libre dans la formule (non qualifiée).

Autres Exemples

1. Donner les informations sur les bateaux
2. Donner le n° du bateau de ALI
3. Donner le n° et le nom du bateau de ALI.
4. Donner les n° des Bateaux qui ne participent pas aux compétitions n° 210" LE GRAND TOUR"

5. Donner les n° des Bateaux qui ne participent pas aux compétitions LE GRAND TOUR"

6. Donner le nom des compétitions dans lesquelles ont participé tous les bateaux.