

Faculté de foresterie et de géomatique

Département des sciences géomatiques



*Intégration et optimisation de données à référence spatiale
produites par le centre de recherche GEOTOP et de
données environnementales provenant
d'organismes gouvernementaux*

Dans le cadre du cours
Sujet spécial
SCG-62058

Présenté à
M. Yves van Chestein

Réalisé par
Mourad Atallah
Rodolphe Devillers
François Lemieux
Sylvain Vallières

8 septembre 1999

Table des matières

1	Introduction	3
2	Mise en contexte et problématique	3
3	But et objectif	4
4	Inventaire de la table NOAA	4
5	Approche stratégique	5
5.1	Restructuration de la table NOAA	5
5.2	Environnement matériel et logiciel	5
6	Déroulement du projet	6
6.1	Implantation de la table NOAA	6
6.2	Indexation de la table NOAA	6
6.3	Création de la table cible NOAN	7
6.4	Création de la table NOAG	9
6.5	Remise en question de la table NOAN	11
6.6	Création de vues	13
6.7	Tests de performance et résultats	13
7	Facteurs liés à la configuration matérielle	14
7.1	Vitesse et nombre de processeurs	14
7.2	Mémoire vive	15
7.3	Type de disque dur	15
7.4	Nombre de disques	15
8	Facteurs liés à l'environnement logiciel	15
8.1	Structure de la table	15
8.2	Type d'index	16
8.3	Taille du SGA	17
8.4	Version du logiciel Oracle	17
9	Discussion	17
9.1	Architecture informatique	18
9.2	Ressources humaines	18
9.3	Facteurs importants vers l'optimisation	19
9.3.1	<i>Temps nécessaire pour l'insertion, la mise à jour et la création d'index</i>	19
9.3.2	<i>Temps nécessaire pour le traitement de requêtes types versus les index</i>	20
10	Recommandations	21
Annexe 1	Annexe 1	23
	Tableau d'inventaire des tables NOAA, NOAP et GLACE	24
	Tableau synthèse des valeurs douteuses dans les tables NOAA et NOAP	24
	Tableau présentant les valeurs douteuses pour l'année 198 dans la table NOAA	25
	Tableau présentant les valeurs douteuses pour le mois 31 dans la table NOAA	25
	Tableau présentant les valeurs douteuses pour la longitude 763.883 dans la table NOAA	26
	Tableau présentant les valeurs douteuses pour le mois 31 dans la table NOAP	26
	Tableau présentant les valeurs douteuses pour la latitude -554.68 dans la table NOAP	26
Annexe 2	Annexe 2	27
	Tableau comparatif des composantes des ordinateurs Carpinus et Thor	28
Annexe 3	Annexe 3	29
	Liste des Commandes SQL utilisées pour évaluer l'espace mémoire et disque dur:	30
	Liste des Commandes SQL utilisées pour modifier l'espace alloué à des segments:	30
	Liste des Commandes SQL utilisées pour créer des tables, vues ou des index:	31
	Liste des Commandes SQL utilisées pour faire des insertions ou des mise à jour:	32

1 Introduction

Ce rapport présente les conclusions d'une étude portant sur l'optimisation d'un grand volume de données (environ 20 millions d'enregistrements) stockées dans un système de gestion de bases de données relationnelles (SGBDR). Nous nous sommes attardés particulièrement aux aspects touchant l'efficacité de l'architecture des données implantées sur des appareils comportant différentes configurations. Au moment des essais, ces appareils, de type "ordinateur personnel", supportaient les versions 7.3.2.3.13 (*Professional*) et 8.0.4 (*Enterprise*) du SGBDR d'Oracle Corporation. Des essais ont été réalisés en faisant varier différents facteurs tels que la structure de la table, des index et les facteurs se rapportant à la gestion des espaces de stockage.

2 Mise en contexte et problématique

Le projet d'étude s'inspire d'une problématique réelle à laquelle font face des chercheurs du Centre de Recherche en Géochimie Isotopique et en Géochronologie (GEOTOP) de l'Université du Québec à Montréal (UQAM). À des fins de consultations, ces chercheurs utilisent un jeu de données produit par des organismes gouvernementaux et commercialisé par le *National Oceanographic and Atmospheric Administration American* (NODC, 1994), branche du ministère du commerce des États-Unis. Il est en fait, le plus important jeu de données existant sur les paramètres physico-chimiques en milieu marin issu d'analyses effectuées dans les différents laboratoires mondiaux depuis la fin du siècle dernier. Au sein de l'UQAM, les données ont été extraites et chargées dans un système de gestion de bases de données Oracle. Étant donné le volume considérable qu'elles représentent, les données ont été regroupées en fonction de deux zones géographiques définies, la zone Atlantique et la zone Pacifique, et insérées dans les tables NOAA et NOAP dont la taille avoisine le Gigaoctet.

Actuellement à l'Université, ces tables résident sur un serveur de grande puissance (HP 9000 série 900). Elles sont administrées par le service informatique en l'absence d'un personnel qualifié au GEOTOP pour en faire la gestion. Néanmoins, ce service offre plusieurs avantages dont, entre autres, l'installation de nouveaux jeux de données, leurs entretiens et mises à jour, ainsi que les processus de sauvegarde journaliers. Par ailleurs, les utilisateurs sont assujettis au bon vouloir et à la disponibilité du gestionnaire des bases de données (*DBA*) qui, en période de restriction budgétaire et organisationnelle, ne favorise pas nécessairement l'aide au développement et à la recherche. Cette situation n'accélère en rien le travail devant être réalisé sur les tables mises en place. Par conséquent, les utilisateurs sont contraints d'effectuer des très longues requêtes qui peuvent nécessiter plusieurs heures de traitement de la part du serveur.

Dans ce contexte, les chercheurs du GEOTOP étaient intéressés à connaître les avantages et les inconvénients qu'engendrerait le rapatriement des tables sur un ordinateur personnel (*PC*) (figure 1). Ils se questionnèrent donc sur les effets d'une telle mutation en termes de vitesse de traitements et de formation du personnel tant pour l'exploitation que la gestion des données au centre de recherche.

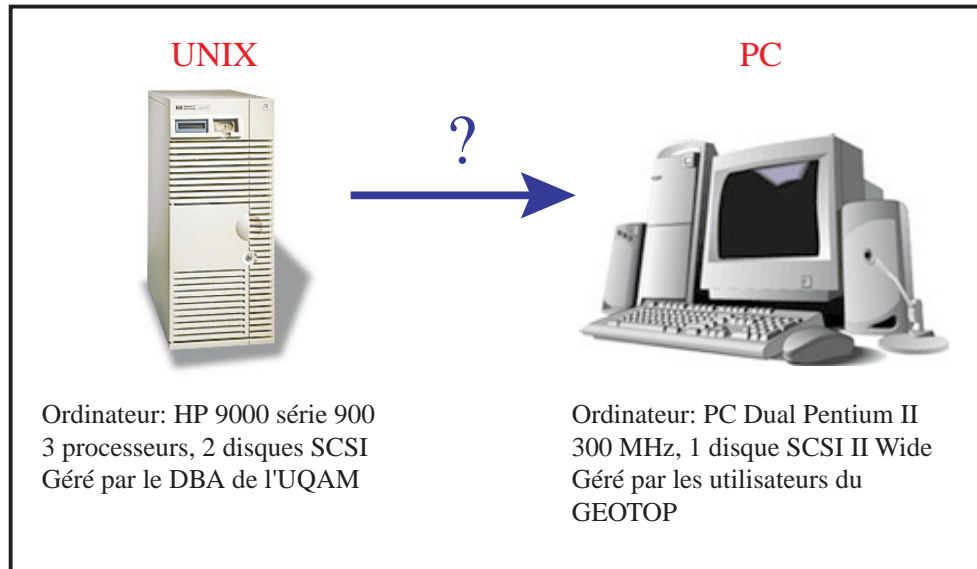


Figure 1 Choix de l'infrastructure matérielle supportant les bases de données NOAA et NOAP

3 But et objectif

Ce projet consiste en l'exploration des différentes possibilités permettant d'optimiser l'architecture des données dans le but de réduire de façon significative les temps de traitement. Le principal objectif de l'étude vise à déterminer les différents facteurs pouvant influencer les performances de certaines requêtes sur des tables Oracle de grande taille.

Les travaux ont porté exclusivement sur les données de la table NOAA en raison de sa taille, plus de 800 Mo, et de son taux d'utilisation élevé de la part des chercheurs du GEOTOP.

4 Inventaire de la table NOAA

Dans un premier temps, nous avons pris connaissance des caractéristiques initiales de la table NOAA en réalisant un inventaire de ses données (annexe 1). Cette étape jugée importante, nous a grandement servi à valider notre approche stratégique décrite à la section suivante.

C'est à partir d'une description de la table commandée dans l'environnement de communication SQL Plus d'Oracle, que nous avons obtenu d'abord les noms des champs et le type. Ensuite, à l'aide de certaines requêtes, des renseignements supplémentaires nous ont été fournis sur les champs. En effet, pour caractériser ces champs, nous avons utilisé les critères de recherche suivants: le nombre total d'enregistrements, le nombre d'enregistrements distincts, la valeur maximum et minimum ainsi que la valeur moyenne.

Durant l'exercice, nous avons décelé des erreurs se trouvant au préalable dans la table. D'après le tableau d'inventaire, certaines mesures auraient été effectuées en l'an 198 ou lors du mois 31 ou

encore à une longitude de 763,883. Après quelques recherches et ne sachant pas davantage la source exacte de ces erreurs, aucune correction n'a pu être appliquée. Les valeurs douteuses (moins de 50 enregistrements) ont donc été volontairement supprimées de la table NOAA.

Enfin, ce tableau synthèse nous sera d'une grande utilité au moment de faire la validation des opérations subséquentes qui serviront à la restructuration de la table.

5 Approche stratégique

Après avoir envisagé plusieurs scénarios, nous avons établi une approche ciblant deux volets d'exploration. Un premier volet consistait à rechercher une nouvelle structure de données pour la table NOAA afin que les requêtes soient résolues plus rapidement. Le deuxième volet visait à évaluer de quelle manière les appareils et les logiciels devraient être configurés pour avoir accès plus rapidement aux données.

5.1 Restructuration de la table NOAA

Nous avons constaté rapidement la présence de redondance, en rapport avec plusieurs valeurs, ce qui avait pour effet d'augmenter de manière significative la taille de la table NOAA. Par exemple, pour un site de mesures donné, la combinaison de valeurs latitude/longitude était répétée à chaque mois de l'année. Dès le début, la redondance a été considérée comme étant le facteur le plus limitatif sur les performances des traitements. Ainsi, dans une optique d'optimisation, notre première action a été de diminuer le nombre d'entrées (par rapport à un site) en faisant disparaître le champ mois de la table NOAA. De cette façon, l'aspect mensuel des données était relégué au niveau des six variables environnementales, c'est-à-dire en transposant les données d'une structure verticale à une structure plus horizontale. En d'autres mots, cette opération permettait de diviser par 12 le nombre d'entrées dans la table NOAA. Une fois la translation réalisée, différents types d'index ont été créés et testés afin d'en évaluer les variations de performance s'y rattachant.

5.2 Environnement matériel et logiciel

Bien que la gestion d'une telle table semble être plus aisée sur un serveur de type HP-UX, nous avons toutefois réalisé notre projet avec les ordinateurs personnels disponibles au Centre de recherche en géomatique (CRG) de l'Université Laval. Le profil matériel signifiait alors une limite quant aux facilités d'administration, rendant les manipulations plus difficiles à exécuter. Pour des raisons de logistique interne, cette étude a été menée successivement sur deux ordinateurs (annexe 2).

L'un d'eux, *Carpinus*, est équipé de 2 processeurs Pentium Pro à 180 MHz ayant 256 Mo de mémoire vive. L'instance Oracle a été installée sur un disque dur Barracuda de 9,3 Go de type SCSI 2 Ultra Wide. Par la suite, les données ont été transférées sur l'ordinateur *Thor* équipé d'un processeur à 300 MHz avec également 256 Mo de mémoire vive. Sur cet appareil, l'instance Oracle a été distribuée

sur 3 disques SCSI, de 2 Go chacun, et sur un disque de 3,72 Go de type EIDE. L'espace de stockage du système (SGA) était sur le même disque (type EIDE) qui supportait le système d'exploitation NT. Les espaces de stockage temporaire (TMP) et le segment d'annulation (RBS) ont été placés sur un disque tandis que les deux autres disques SCSI réservaient chacun un espace de stockage utilisateur (USR1 et USR2) incluant les tables et les index (NDX1 et NDX2). Ces dernières ont été alternées, le premier disque comportait USR1 et NDX2 et le second USR2 et NDX1, dans le but d'augmenter les performances d'accès aux données.

Concernant l'environnement logiciel, notre choix s'est porté sur le SGBD d'Oracle qui s'avère être le système en fonction à l'UQAM de même qu'au CRG. Au cours du projet, nous avons dû migrer de version, pour finalement travailler avec Oracle serveur 7.3.2.3.13 installé sur l'ordinateur *Carpinus* et sa version 8.0.4.0.0. opérable sur *Thor*.

6 Déroutement du projet

Au début du projet, les données de la table NOAA nous ont été fournies par le gestionnaire de la base de données. Elles étaient sauvegardées dans un fichier de type *Dump File* (noaa.dmp) généré à partir de la table originale du serveur de l'UQAM.

6.1 Implantation de la table NOAA

Lors de l'exportation de la table NOAA, nous n'avons transféré que les données en soi. La structure des index ainsi que les droits d'accès à la table par les utilisateurs n'ont pas été retenus. Avant de lancer l'opération, nous avons sélectionné l'option *Compress Extent for Imports* afin de régénérer la table dans une seule étendue (plus de 800 Mo d'espace disque). Ensuite, l'importation des 20 510 500 d'enregistrements s'est réalisée, via l'outil *Oracle Data Manager*, sans heurt au sein de l'instance Oracle de l'ordinateur *Carpinus*. Précisons toutefois, que l'opération a duré au total deux heures vingt-huit minutes. Une fois l'importation terminée, la commande suivante nous a permis de valider que la table tenait bel et bien dans une seule étendue.

```
SQL> select extent_id, bytes, blocks from user_extents where
       segment_name = 'NOAA';
```

6.2 Indexation de la table NOAA

L'étape suivante a consisté à créer un index composite sur les champs année, mois, latitude, longitude et profondeur (noaa_amlp). A priori, nous avons estimé que la taille de l'index serait approximativement du même ordre de grandeur que la table. Nous avons alors réservé une étendue de 800 Mo pour sa création.

```
SQL> create index noaa_amlp on noaa(annee, mois, latitude, longitude,
    profondeur)
    storage (initial 800m next 50m pctincrease 0);
```

La première tentative a échoué à cause d'un manque d'espace alloué au segment temporaire (TMP), celui-ci était alors de 400 Mo. Nous avons modifié la taille du TMP à plusieurs reprises avant de pouvoir créer l'index. Il nous a donc fallu un fichier TMP de 2 Go pour y arriver. Notre estimation initiale de la taille de l'index s'est avérée inexacte car celui-ci occupait dans plusieurs étendues. Comme l'objectif principal était l'optimisation des temps d'accès, nous avons détruit cet index et l'avons recréé, mais cette fois avec une étendue initiale plus importante. Le processus a finalement donné un index qui réside en une seule étendue d'environ 1,2 Go d'espace disque.

Quelques tests d'intégrité sur la table NOAA nous ont permis de constater que le temps requis pour effectuer les requêtes localement était sensiblement le même qu'avec le serveur HP 9000 série 900.

6.3 Création de la table cible NOAN

Suite à l'implantation de la table NOAA, nous avons entrepris la phase de restructuration du jeu de données en effectuant une translation de l'information associée à chaque variable par rapport à un site donné et suivant les douze mois de l'année (figure 2).

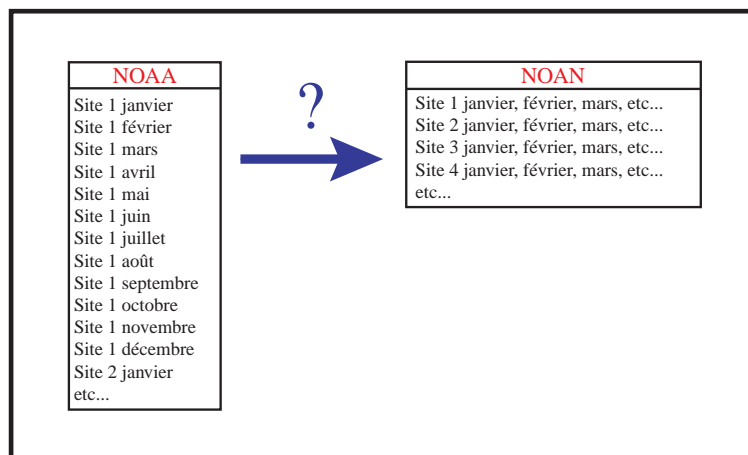


Figure 2 Restructuration de la table NOAA devenant la table cible NOAN

La table résultante NOAN devait par conséquent contenir 2 millions plutôt que 20 millions d'enregistrements en plus de remédier au problème de redondance que présentait la table NOAA. Nous avons également réajusté la longueur des champs afin de ne conserver que l'espace nécessaire au stockage des données. La translation de la table NOAA a demandé, au préalable, de créer la structure de la nouvelle table NOAN dans Oracle à partir de la commande SQL suivante:

```

SQL> create table noan(annee number(4),lat number(6,3),lon number(7,3),
prof number(6,1), temp01 number(4,2), temp02 number(4,2), temp03
number(4,2), temp04 number(4,2), temp05 number(4,2), temp06
number(4,2), temp07 number(4,2), temp08 number(4,2), temp09
number(4,2), temp10 number(4,2), temp11 number(4,2), temp12
number(4,2), salio1 number(4,2), salio2 number(4,2), salio3
number(4,2), salio4 number(4,2), salio5 number(4,2), salio6
number(4,2), salio7 number(4,2), salio8 number(4,2), salio9
number(4,2), salio10 number(4,2), salio11 number(4,2), salio12
number(4,2), doxy01 number(4,2), doxy02 number(4,2), doxy03
number(4,2), doxy04 number(4,2), doxy05 number(4,2), doxy06
number(4,2), doxy07 number(4,2), doxy08 number(4,2), doxy09
number(4,2), doxy10 number(4,2), doxy11 number(4,2), doxy12
number(4,2), phos01 number(4,3), phos02 number(4,3), phos03
number(4,3), phos04 number(4,3), phos05 number(4,3), phos06
number(4,3), phos07 number(4,3), phos08 number(4,3), phos09
number(4,3), phos10 number(4,3), phos11 number(4,3), phos12
number(4,3), sili01 number(6,3), sili02 number(6,3), sili03
number(6,3), sili04 number(6,3), sili05 number(6,3), sili06
number(6,3), sili07 number(6,3), sili08 number(6,3), sili09
number(6,3), sili10 number(6,3), sili11 number(6,3), sili12
number(6,3), ntra01 number(5,3), ntra02 number(5,3), ntra03
number(5,3), ntra04 number(5,3), ntra05 number(5,3), ntra06
number(5,3), ntra07 number(5,3), ntra08 number(5,3), ntra09
number(5,3), ntra10 number(5,3), ntra11 number(5,3), ntra12
number(5,3))
storage (initial 600M next 20M pctincrease 0);

```

L'intégration des données dans NOAN s'est effectuée en deux étapes. La première consistait en une série de commandes *Insert* qui introduisaient pour chaque site l'année, la latitude, la longitude, la profondeur et la température d'un mois donné (par exemple janvier). Cependant, il n'a pas été possible de lancer cette commande pour l'ensemble de la table dû à des restrictions liées à la taille du segment d'annulation (RBS) dont nous disposions. Il a donc fallu partager cette commande au niveau du mois et imposer un *Commit* après chaque insertion afin de vider le segment RBS. Pour y parvenir, nous avons préparé un fichier de commandes SQL contenant l'ensemble des instructions d'insertion pour tous les mois et toutes les années. Voici un extrait pour le mois de janvier 1898.

```

SQL> insert into noan(annee, lat, lon, prof, temp01) select annee,
latitude, longitude, profondeur, temp from noaa where annee=1898
and mois=01;
SQL> commit;

```

La deuxième étape consistait à faire une mise à jour des champs vides de la table NOAN par le biais de la table NOAA. Pour la même raison qu'à l'étape précédente, il a fallu partager cette commande pour être en mesure d'effectuer la mise à jour.

```

SQL> update noan set temp02 = (select temp from noaa where noan.annee =
noaa.annee and noan.lat = noaa.latitude and noan.lon =
noaa.longitude and noan.prof = noaa.profondeur and noan.temp01 =
noaa.temp and noaa.mois = 02);
SQL> commit;

```

Au cours de cette seconde étape, nous nous sommes butés à un problème d'unicité qui a perturbé les opérations de mise à jour de la table NOAN. En se rapportant à l'exemple précédent, le critère de sélection température (temp) correspondait à plusieurs entrées de la table NOAA. Finalement, il s'est avéré que la combinaison des champs année, latitude, longitude, profondeur et température, pour un mois donné, n'était pas unique. À partir de ce moment, nous devons déterminer la pertinence de conserver ou non les duplicatas de la table NOAA. Compte tenu du type de requêtes, principalement des moyennes arithmétiques, et du faible nombre de duplicatas rencontrés pour un même site, nous avons choisi de regrouper ces valeurs. Ainsi, à l'aide de la commande SQL *Average*, il nous a été possible de faire la moyenne des valeurs associées à une même année, mois, latitude, longitude et profondeur.

En résumé, pour obtenir la table restructurée NOAN (figure 3), nous avons été dans l'obligation de créer une table intermédiaire NOAG ayant la même structure que la table NOAA, mais sans duplicata.

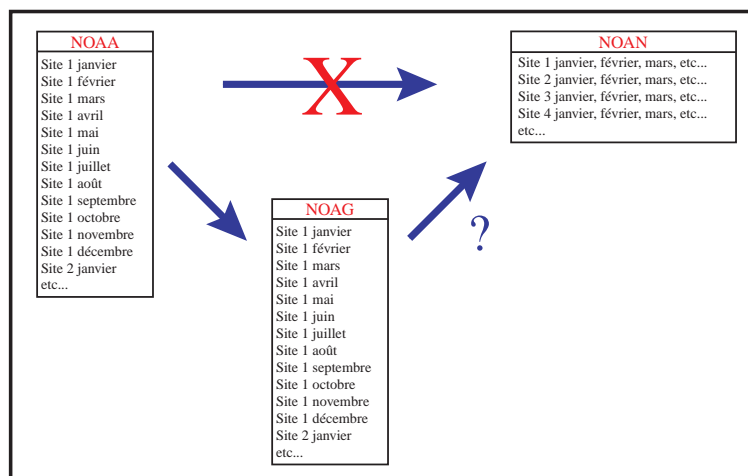


Figure 3 Changement d'approche, création de la table intermédiaire NOAG

6.4 Création de la table NOAG

À l'exception du nom de certains champs pour faciliter la syntaxe des requêtes SQL, les champs latitude, longitude et profondeur sont devenus lat, lon et prof, voici le syntaxe qui a servi à la création de la table NOAG:

```
SQL> create table noag(annee, mois, lat, lon, prof, temp, sali, doxy,
    phos, sili, ntra)
    storage (initial 600M next 10M pctincrease 0)
    as select annee, mois, latitude, longitude,
    profondeur, AVG(temp), AVG(sali),
    AVG(doxy), AVG(phos), AVG(sili), AVG(ntra)
    from noaa
    group by annee, latitude, longitude, profondeur, mois;
```

Cette commande, techniquement valide, n'a pu être réalisée en entier car le segment TMP ne pouvait contenir la totalité du résultat de la requête. Nous avons donc dû vider le contenu de la table NOAG (commande *Truncate*) et procéder par insertion (*Insert*) partielle découpée selon les années (exemple ci-dessous, de 1898 à 1900 exclusivement). La table NOAG résultante comptait 17 millions d'enregistrements, soit environ 3 millions de moins que NOAA. Par ailleurs, elle offre l'unicité pour la combinaison des champs année, mois, lat, lon et prof.

```
SQL> insert into noag select annee, mois, latitude, longitude,
    profondeur, AVG(temp), AVG(sali), AVG(doxy), AVG(phos), AVG(sili),
    AVG(ntra)
    from noaa where annee >= 1898 and annee < 1900
    group by annee, latitude, longitude, profondeur, mois;
SQL> commit;
```

Puisque nous avons désormais l'unicité des données, il nous a été possible de créer un index unique sur la table NOAG devant accélérer le processus de chargement de la table NOAN.

```
SQL> create unique index noag_amlp on noag(annee,mois,lat,lon,prof)
    storage (initial 600m next 10m pctincrease 0);
```

Ainsi, nous disposions d'une table sans duplicata avec un index unique tout à fait opérationnel sur *Carpinus*. Les temps obtenus pour une requête type étaient satisfaisants mais ne s'avéraient pas vraiment plus rapides que sur le serveur de l'UQAM. Nous avons alors estimé que les performances pourraient être améliorées si la table, son index ainsi que le segment temporaire résidaient sur des disques SCSI distincts accessibles en parallèle et non en série.

L'ordinateur *Thor* répondait à ces critères et possédait, entre autres, trois disques SCSI. Ces derniers sont toutefois d'une génération antérieure (type I) à celle de l'ordinateur *Carpinus* (type II, Ultra-Wide). Nous avons choisi d'exporter la table NOAG à l'aide de l'outil *Oracle Data Manager* générant par le fait même le fichier noag.dmp de 2,6 Go. Celui-ci fut réimporté dans l'instance Oracle de *Thor* sans problème, indépendamment de la version (8.0.4) installée sur celui-ci.

De la même façon qu'avec *Carpinus*, nous avons recréé un index unique sur les champs année, mois, lat, lon et prof, à la différence que cette fois l'index de la table NOAG ainsi que le segment d'annulation se trouvaient physiquement sur des disques distincts. Puisque nous disposions de disques SCSI dans le cas de *Thor*, et, en sachant qu'une table et son index peuvent être disposés sur deux disques différents, le système devait être capable d'accéder simultanément à la table et à son index. Nous espérions obtenir ainsi des gains de performance non négligeables. Au contraire, la table indexée NOAG a démontré des performances étant deux fois plus lentes que celles obtenues sur *Carpinus*. Il nous est apparu, à cette étape-ci, que le type de disque dur aurait une importance beaucoup plus grande que la répartition physique des tables et d'index sur des disques différents.

Il est possible de visualiser, à la figure 4, la répartition physique des différents segments de l'instance Oracle pour les environnements testés dans le projet.

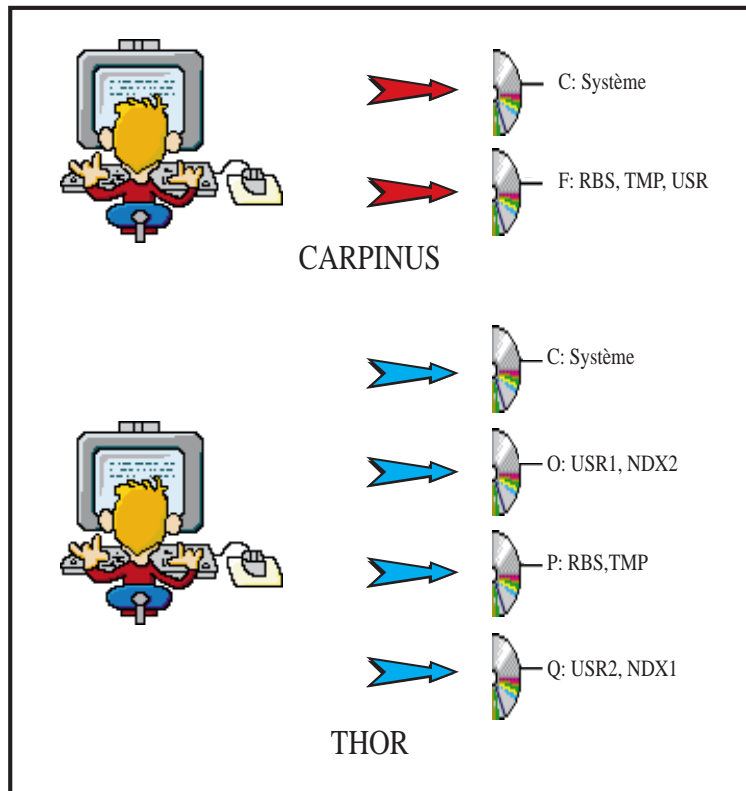


Figure 4 Répartition physique des segments (SYS, TPM, RBS, etc.) sur *Thor* et *Carpinus*

6.5 Remise en question de la table NOAN

L'approche retenue, visant l'optimisation de la structure des données de NOAA, était basée sur l'état répétitif des champs année, latitude et longitude que présentait chaque mois où il y avait de l'information. Nous espérons de cette manière être en mesure de réduire considérablement le nombre d'entrées dans la table, et avec l'index unique, d'augmenter potentiellement les vitesses de traitements. Suite à la création de la table NOAN, nous avons constaté qu'il n'y avait pas systématiquement des valeurs associées à chacun des mois (2 à 3 en moyenne) pour l'ensemble des sites. En réalité, la table NOAN contenait 13 millions d'enregistrements plutôt que 2 millions d'enregistrements anticipés au départ. Cette réduction est quand même appréciable en ce qui a trait à la taille du fichier, mais influence peu les performances pour le même type d'index. Par conséquent, nous avons remis en question la pertinence de créer la table NOAN compte tenu des efforts exigés pour la mettre en place ainsi que le risque élevé de corruption des données lors des nombreuses manipulations. De plus, avec la structure que présente NOAN, nous étions dans l'obligation de modifier la syntaxe de notre requête type. Tel que démontré à la figure 5, la requête adaptée est passablement plus longue que dans le cas de NOAA et elle retourne les résultats sous une forme peu conviviale à l'utilisateur. En contrepartie, la solution

**Requête type
sur la table
NOAA ou NOAG**

```
select mois, avg(sali) "moyennes
mensuelles", stddev(sali) "ecart
type", count(sali) "nombre de
mesures" from noaa where
profondeur = 0 and latitude
between (50 - (1)) and (50 + (1))
and longitude between ((-30) -
((1/cos(50/57.29578))/2)) and
((-30) + ((1/cos
(50/57.29578))/2))
group by mois
order by mois;
```



NOAA SUR NOBEL			
Mois	Moyennes mensuelles	Écart type	Nombre de mesures
1	35.42309	0.15940	11
2	35.33721	0.10634	14
3	35.28180	0.17059	30
4	35.43367	0.13235	15
5	35.43324	0.16828	17
6	35.17435	1.17000	23
7	35.38111	0.20694	19
8	35.28344	0.22151	27
9	35.27740	0.20567	20
10	35.08280	0.19388	15
11	35.14014	0.20820	28
12	35.13418	0.16992	22

**Requête type
sur la table
NOAN**

```
select avg(sali01) "janvier",
stddev(sali01) "ecart type",
count(sali01) "nombre de
mesures", avg(sali02) "fevrier",
stddev(sali02) "ecart type",
count(sali02) "nombre de
mesures", avg(sali03) "mars",
stddev(sali03) "ecart type",
count(sali03) "nombre de
mesures", avg(sali04) "avril",
stddev(sali04) "ecart type",
count(sali04) "nombre de
mesures", avg(sali05) "mai",
stddev(sali05) "ecart type",
count(sali05) "nombre de
mesures", avg(sali06) "juin",
stddev(sali06) "ecart type",
count(sali06) "nombre de
mesures", avg(sali07) "juillet",
stddev(sali07) "ecart type",
count(sali07) "nombre de
mesures", avg(sali08) "aout",
stddev(sali08) "ecart type",
count(sali08) "nombre de
mesures", avg(sali09)
"septembre", stddev(sali09)
"ecart type", count(sali09)
"nombre de mesures", avg(sali10)
"octobre", stddev(sali10) "ecart
type", count(sali10) "nombre de
mesures", avg(sali11)
"novembre", stddev(sali11)
"ecart type", count(sali11)
"nombre de mesures", avg(sali12)
"decembre", stddev(sali12)
"ecart type", count(sali12)
"nombre de mesures" from noan
where prof = 0 and lat between (50
- (1)) and (50 + (1)) and lon
between ((-30) - ((1/cos
(50/57.29578))/2)) and ((-
30+((1/cos(50/57.29578))/2));
```



NOAN SUR CARPINUS			
janvier	ecart type	nombre de	mesures
fevrier	ecart type	nombre de	mesures
mars	ecart type	nombre de	mesures
avril	ecart type	nombre de	mesures
mai	ecart type	nombre de	mesures
juin	ecart type	nombre de	mesures
juillet	ecart type	nombre de	mesures
aout	ecart type	nombre de	mesures
septembre	ecart type	nombre de	mesures
octobre	ecart type	nombre de	mesures
novembre	ecart type	nombre de	mesures
decembre	ecart type	nombre de	mesures
35.42309	0.15940	11	35.33721
0.10634	14	35.28180	0.17059
30	35.43367	0.13235	15
35.43324	0.16828	17	35.17435
1.17000	23	35.38111	0.20694
19	35.28344	0.22151	27
0.20567	20	35.08280	0.19388
15	35.14014	0.20820	28
35.13418	0.16992	22	

Figure 5 Comparaison entre les formes de la requête selon la table interrogée NOAA ou NOAN

intermédiaire proposant la création de la table NOAG, s'avère être beaucoup plus avantageuse en termes de temps, d'espace disque et de difficulté d'implantation. De plus, puisque les utilisateurs effectuent pour la majeure partie du temps leurs requêtes pour une profondeur de zéro mètre (allusion ici à la surface de l'eau), nous avons créé une table complémentaire répondant à ce besoin spécifique, soit la table SURFACE.

En résumé, la table NOAG permet de lancer tous les types de requêtes moyennant un temps de traitement passablement long tandis que la table SURFACE offre des temps de réponse presque instantanés.

6.6 Création de vues

Nous avons exploré la possibilité d'interroger des vues (*View*) dans Oracle. Si la création d'une vue est pratiquement instantanée, le temps pour résoudre une requête type semble être comparable, voire parfois un peu plus long, que celui requis en interrogeant directement une table. À titre d'exemple, nous avons testé une vue sur la table NOAG qui pointait vers les enregistrements présentant la surface de l'eau. En comparant le résultat d'une requête avec celui obtenu en questionnant la table indexée SURFACE, nous avons constaté que le temps de réponse était plus rapide avec la table qu'avec la vue, soit 2.75 et 10.97 secondes respectivement.

6.7 Tests de performance et résultats

Rappelons que cette étude avait comme objectif principal la recherche de solutions permettant de réduire le temps des traitements. Alors, nous avons effectué une série de tests de performance en faisant varier différents facteurs tels que la structure de la table et les index selon la configuration physique ou logicielle de l'ordinateur. À chaque test, nous devons lancer une requête dont la syntaxe s'apparentait en tous points à la requête la plus souvent employée par les chercheurs du GEOTOP. Plus précisément, cette requête permettait d'extraire les valeurs moyennes de salinité et cela, pour un site ayant une latitude et longitude donnée (voir l'exemple ci-dessous, table NOAG).

```
SQL> select mois,
      avg(sali) "moyennes mensuelles",
      stddev(sali) "ecart type",
      count(sali) "nombre de mesures"
      from noag
      where lat between (50 - (1)) and (50 + (1))
      and lon between ((-30) - ((1/cos(50/57.29578))/2)) and ((-30) +
      ((1/cos(50/57.29578))/2)) and prof = 0
      group by mois
      order by mois;
```

Afin de ne pas fausser les résultats, les règles suivantes ont dû être suivies au moment d'exécuter les tests: la requête devait être absente de la mémoire vive des ordinateurs pour ne pas sous-estimer les temps du traitement; dans le but d'éliminer l'influence que pourrait avoir une connexion réseau, la

requête a été lancée sur l'appareil possédant localement l'instance Oracle; le traitement devait être exécuté lorsque aucun autre logiciel n'était en fonction et qu'aucun utilisateur n'était connecté *via* une connexion réseau. Chaque test a été effectué à plusieurs reprises afin d'obtenir des résultats significatifs. Les résultats moyens des tests de performance sont présentés dans le tableau 1.

Nom de la table	Ordinateur	Type d'index	Temps (seconde)
NOAA	Carpinus	LLP	9.91
NOAA	Carpinus	AMLLP	171.02
NOAA	Carpinus	- - - - -	1 217.30
NOAA	Nobel	L + L	88.81
NOAG	Carpinus	- - - - -	143.19
NOAG	Carpinus	AMLLP	138.91
NOAG	Carpinus	LLP	5.56
NOAG	Thor	AMLLP	334.36
NOAG	Thor	LLP	11.24
NOAN	Thor	ALLP	388.35
NOAN	Thor	L (lat)	262.12
NOAN	Thor	LLP	10.33
NOAN	Thor	LLP	50.88

(A)nnée, (M)ois, (L)atitute, (L)ongitude, (P)rofondeur

Tableau 1 Tests de performance sur les tables avec ou sans index

À la lumière des résultats, nous constatons qu'il existe différents facteurs pouvant influencer les performances du traitement de la requête. Nous les avons regroupés selon deux catégories: les facteurs liés à la configuration physique de l'ordinateur (mécanique) et les facteurs associés à l'environnement logiciel (incluant sa configuration).

7 Facteurs liés à la configuration matérielle

7.1 Vitesse et nombre de processeurs

Lors de l'exécution des requêtes, les processeurs fonctionnaient à moins de 50 % de leur capacité (10 % la plupart du temps) tant sur l'ordinateur *Carpinus*, équipé de deux processeurs Pentium Pro à 180 MHz, que sur *Thor* qui a un seul processeur à 350 MHz. Cela laisse supposer que la vitesse et le nombre de processeurs ne sont pas des facteurs limitant dans cette étude.

7.2 Mémoire vive

Les deux ordinateurs utilisés possèdent chacun 256 Mo de mémoire vive. Au cours de l'exécution des requêtes, nous avons constaté que la mémoire vive utilisée n'augmentait pas d'après le *Task Manager* de Windows NT. En fait, Oracle utilise plutôt la mémoire allouée à son SGA qui, une fois les services d'Oracle fermés, est libérée et retournée en mémoire vive dans l'ordinateur. Sur *Thor*, la mémoire allouée au fonctionnement du SGA frôlait les 50 Mo. Par conséquent, la taille du SGA (allouée) est directement influencée par la quantité de mémoire vive installée sur l'appareil.

7.3 Type de disque dur

Des études préliminaires effectuées dans le cadre du cours Optimisation ont démontré que, de manière générale, les disques de type SCSI sont plus rapides que les disques de type EIDE. Toutefois, un disque EIDE récent peut être plus rapide qu'un disque SCSI plus ancien. Dans le cadre de cette étude, nous avons seulement utilisé des disques de type SCSI. Il a également été démontré que les vitesses de transfert de données entre deux disques sont contrôlées par la vitesse du disque où se fait l'écriture de l'information.

Pour mieux connaître l'influence de cette composante physique, nous avons implanté la table NOAG sur les deux ordinateurs personnels et y avons créé deux types d'index différents soit AMLLP et LLP (tableau 1). Nous avons constaté qu'il faut, en moyenne, 2,2 fois plus de temps à résoudre une requête type sur *Thor* que sur *Carpinus*.

7.4 Nombre de disques

Dans le cas de *Carpinus*, les espaces de stockage étaient situés sur le même disque tandis qu'avec *Thor*, nous avons réparti le tout sur trois disques différents. Les résultats obtenus indiquent que la répartition des espaces de stockage sur différents disques n'engendre pas nécessairement un gain de performance appréciable en comparaison avec les résultats issus des tests sur la génération des disques.

8 Facteurs liés à l'environnement logiciel

8.1 Structure de la table

Afin de quantifier l'influence que pourrait avoir la structure de la table par rapport au traitement d'une requête type, nous avons d'abord, à partir de *Carpinus*, comparé les temps provenant d'essais sur les tables NOAA et NOAG. Pour que la comparaison soit valable, il fallait qu'elles aient le même type d'index. Dans ce cas, les résultats les plus significatifs ont été obtenus avec un index de type LLP, soit des temps de 9.91 et 5.56 secondes pour les tables NOAA et NOAG respectivement. Avec le même type d'index, une autre série de tests effectués sur les tables NOAG et NOAN ont donné des temps de 11.24 et 10.33 secondes sur l'ordinateur personnel *Thor*.

Les différences de performance observées entre la structure des tables NOAA et NOAG peuvent s'expliquer par une diminution du nombre d'enregistrements. En revanche, les performances liées à la table restructurée NOAN montrent une légère baisse de la performance, bien que cette variation ne soit pas significative.

8.2 Type d'index

La table NOAA implantée sur le serveur universitaire *Nobel* de l'UQAM était indexée sur les champs latitude et longitude (deux index distincts situés sur un même disque). Dans notre étude, plusieurs types d'index ont été expérimentés en vue de déterminer l'index le plus performant.

Tel que schématisé à la figure 6, nous avons d'abord utilisé les index uniques AMLLP dans le cas de la table NOAG et ALLP pour la table NOAN. À cause de la présence de duplicatas dans la table NOAA, aucun index unique n'a été testé. Par contre, nous avons testé plusieurs types d'index non uniques.

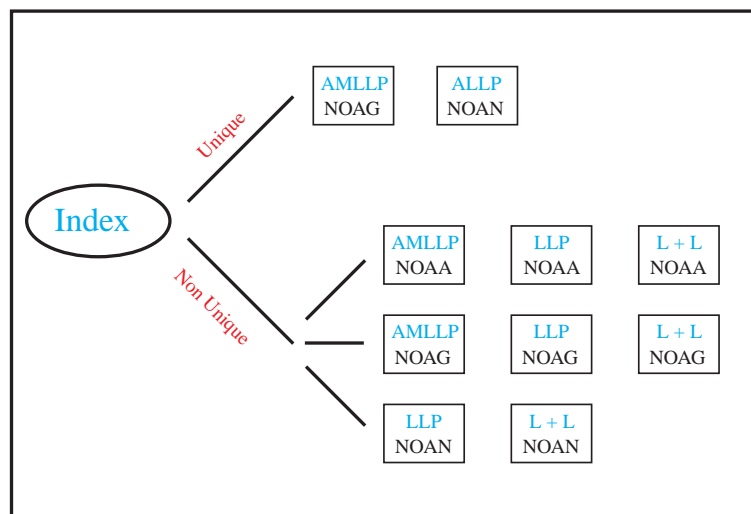


Figure 6 Type d'indexation expérimentée sur les tables NOAA, NOAG et NOAN

Les tests effectués montrent que les temps les plus longs ont été observés sur les tables sans index atteignant 1 217 secondes dans le cas d'une requête exécutée sur la table NOAA (tableau 1, ordinateur *Carpinus*). Concernant les temps sur les tables indexées, nous avons obtenu une diminution d'un facteur 17 pour la table NOAA et d'un facteur 25 pour la table NOAG, toutes deux sur *Carpinus*.

Pour des raisons pratiques, des essais plus nombreux ont été effectués sur la table SURFACE qui inclut toutes les données présentes dans la table NOAG et où le champ profondeur a une valeur de zéro. Les temps obtenus sans index ont donné dans ce cas près de 20 secondes. Il est intéressant de noter que certains index, dont l'ordre des champs ne présente aucune logique face aux champs appelés dans la requête, peuvent causer des temps de requêtes plus longs qu'avec une table sans index. Ainsi,

l'index PMALL a occasionné un temps maximum de 30 secondes, ce qui est une fois et demie supérieure au temps obtenu sans index. Cependant, le choix d'un index combinant les mêmes champs mais dans un ordre se rapprochant de la requête (AMLLP), fournit des temps de l'ordre des 20 secondes (identiques avec les temps obtenus sans index).

Les temps les plus courts ont été réalisés en utilisant des index de type LLP ou LL. Parmi les index LL, le simple fait d'inverser les champs latitude et longitude de l'index a engendré une variation des performances. Ainsi, le résultat inférieur a été de 2.34 secondes avec un index de type LL dont la séquence était longitude/latitude. De plus, les index de type LLP et LL demandaient en mémoire SGA près de la moitié de celle octroyée pour créer les index de type AMLLP.

8.3 Taille du SGA

Ce paramètre de configuration spécifique à Oracle, le SGA (*System Global Area*), peut influencer significativement les performances lors de requêtes. Le SGA représente l'équivalent de la mémoire vive pour le logiciel Oracle. Ainsi, plus la taille du SGA est élevée, plus Oracle sera capable de garder en mémoire les données à traiter, sans constamment avoir à les recharger. Dans notre cas, les tables avaient une taille supérieure à 600 Mo. Avec les appareils dont nous disposions, il s'avérait impossible d'allouer autant au SGA pour conserver, selon chaque cas, les tables en mémoire. En contrepartie, nous avons varié sa taille et observé les performances s'y rattachant.

8.4 Version du logiciel Oracle

La compagnie Oracle affirme que les versions 8.x sont plus performantes que les 7.x. Nous n'avons pas été à même de vérifier cette affirmation car il nous aurait fallu installer successivement les deux versions du logiciel sur le même ordinateur, ce qui n'était pas réellement possible.

9 Discussion

La majeure partie du projet consistait en une phase exploratoire ayant trait principalement à la manipulation d'un grand volume de données sur des ordinateurs de type personnel sous Oracle. Il était initialement prévu que, suite à l'implantation du jeu de données et son optimisation, certains aspects d'intégration et d'environnement multi-plateformes seraient étudiés. Notre estimation des efforts consacrés à la restructuration des données a été grandement sous-estimée puisque nous avons eu le temps de couvrir exclusivement ce point. Ce fut cependant extrêmement enrichissant car nous avons été confrontés à une problématique très particulière où les difficultés associées à la taille du jeu de données ne croissaient pas de façon linéaire mais bien de façon exponentielle. Nous avons donc été amenés à revoir notre conception de ce qu'est la performance. En effet, un ordinateur considéré très performant pour des tâches usuelles de bureautique peut, pour des besoins différents, être considéré comme très peu performant voire pratiquement inutilisable.

Comme l'approche était essentiellement de type exploratoire, il est difficile de rapporter tous les essais et tentatives qui furent effectués sur le jeu de données. Dans ce qui suit, nous présentons les facteurs qui jouent un rôle important.

9.1 Architecture informatique

Dans cette étude, nous avons opposé les performances d'un ordinateur de type serveur à une station de travail sur ordinateur personnel (*PC*). Après l'optimisation des données, les performances réalisées à partir d'une requête type dans l'environnement *PC* sont près de 16 fois supérieures à celles obtenues sur le serveur NOBEL. Il faut cependant préciser que les données du serveur ont été intégrées par le gestionnaire de l'UQAM, sans optimisation particulière considérant les besoins des utilisateurs. À titre d'exemple, la table NOAA sur NOBEL réside dans plus de 20 étendues.

Les performances intéressantes issues des *PC* testés sont en étroite relation avec les composantes matérielles qui les constituent. Ceux-ci présentaient des caractéristiques de haute gamme quant à la mémoire vive, les processeurs et les unités de disques. Une instance Oracle, des tables et des index pour un tel jeu de données monopolisent très rapidement leur usage physique. En effet, dans le cas de l'ordinateur *Carpinus*, plus de 50 Mo de mémoire vive sont utilisés en permanence par Oracle et environ 10 Go d'espace sur les disques durs. Si nous ajoutons à cela les besoins en mémoire vive et en espace de stockage du système d'exploitation ainsi que les divers services (ex. réseau, serveur primaire, etc.), nous atteignons rapidement la pleine capacité du système. En revanche, le serveur de l'UQAM s'acquitte assez bien des tâches puisqu'il est conçu spécifiquement pour ces fins tout en gérant une grande quantité d'accès concurrentiels.

9.2 Ressources humaines

Comme les deux plateformes (serveur unix vs station *PC-NT*) sont fondamentalement différentes en ce qui a trait au système d'exploitation et aux besoins auxquels elles répondent généralement, il en va de même pour les connaissances requises pour les piloter. Du côté utilisateur, au centre de recherche de l'UQAM, la gestion d'une plateforme *PC-NT* s'avère être une option beaucoup moins complexe car ces utilisateurs n'oeuvrent pas directement dans un domaine lié aux technologies de l'informatique et surtout celui des SGBD.

La question importante soulevée dans ce travail, concerne la faisabilité de rapatrier les bases de données du serveur central vers un poste local afin d'accroître l'autonomie des utilisateurs. Une des premières constatations est l'écart qui existe entre les connaissances nécessaires pour l'utilisation versus l'optimisation et la gestion de la base de données. Dans l'optique de rapatrier les bases de données localement, les chercheurs du GEOTOP auront probablement besoin d'assistance pour implanter la nouvelle instance.

9.3 Facteurs importants vers l'optimisation

9.3.1 Temps nécessaire pour l'insertion, la mise à jour et la création d'index

Les premières opérations effectuées consistaient à insérer des données extraites de la table NOAA dans la table cible NOAN. Avant d'entreprendre cette étape de restructuration, nous anticipions que les temps de traitements, avec un tel jeu de données, n'excéderaient pas la dizaine de minutes. Au terme du projet, nous affirmons que le facteur temps est celui que nous avons le plus sous-estimé car notre cadre de référence changea très rapidement, passant de l'heure à la dizaine d'heures. Cette nouvelle dimension temporelle fut corroborée dès les premiers essais de mise à jour des champs de la table NOAN. En fait, les problèmes associés à la taille des segments d'annulation et temporaire ont obligé la fragmentation de toutes les commandes d'insertion et de mise à jour au niveau du mois et ce, pour les 97 années présentes dans la table NOAA. La plus part du temps, nous devions planifier, de nuit, la séquence d'exécution des fichiers de commandes afin de ne pas monopoliser l'ordinateur où résidait l'instance Oracle. A titre d'exemple, la mise à jour de la table NOAN indexée à partir de la table NOAG indexée a nécessité environ 22 heures de traitement pour les champs salinité, oxygène dissout, phosphate, silicate et nitrate, donc un grand total de 4 jours et demi de temps processeur! De même, le temps requis pour la création d'un index sur les champs année, mois, longitude, latitude et profondeur de la table NOAG prenait en moyenne 3 heures et demie.

La figure 7 présente les résultats obtenus pour la création de différents index sur une même table. Pour des fins pratiques, nous avons utilisé la table SURFACE pour illustrer l'évolution du facteur temporel.

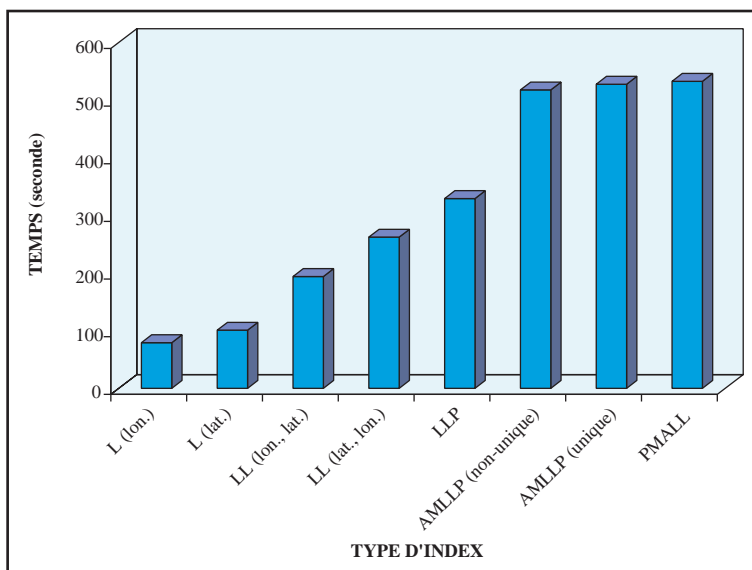


Figure 7 Temps de création des différents types d'index sur la table SURFACE

9.3.2 Temps nécessaire pour le traitement de requêtes types versus les index

Le comparatif que nous avons effectué sur le temps de résolution d'une requête type, a fait ressortir que l'utilisation d'un index est absolument nécessaire pour obtenir des performances acceptables. Cependant, il faut prendre grand soin de choisir la bonne combinaison d'index. En effet, en fonction de la combinaison, il est possible d'améliorer grandement les performances ou bien d'obtenir des performances moindres.

La figure 8 montre les résultats obtenus pour l'exécution d'une requête type sur la table SURFACE avec différents index et sans index. Il en ressort que le type d'index est évidemment très étroitement lié à la structure de la requête. Les premiers champs appelés dans la requête, plus spécifiquement après la clause *Where*, doivent se retrouver dans l'index et idéalement dans le même ordre. Dans le cas où les utilisateurs effectuent pratiquement toujours le même type d'interrogation, il est facile de créer un index adapté à leur besoin. Dans ce cas, c'est la combinaison latitude/longitude/profondeur (LLP) qui s'est avérée être la plus performante. A priori, nous pouvons être tentés de construire un index incluant la totalité des champs appelés par la requête dans le but de pousser les performances au maximum. Il s'avère que ce type d'index composite (année/mois/latitude/longitude/profondeur; AMLLP sur SURFACE), par exemple, devient très long à balayer et offre des temps de traitement similaire à la table SURFACE sans index. La figure 8 présente également le cas où un index, formulé dans un ordre ne respectant pas la séquence d'appel de la requête, donne des temps 40 % plus longs que si cette même requête était lancée sur la table sans index.

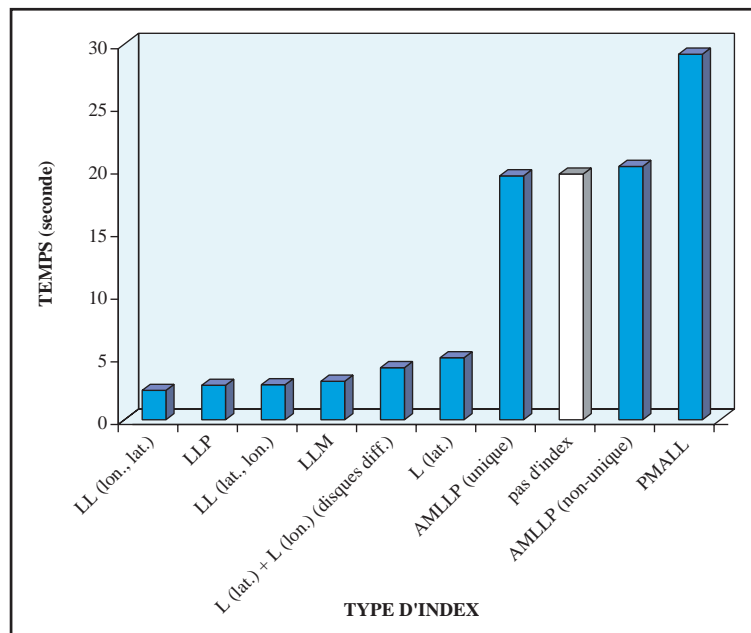


Figure 8 Temps pris pour l'exécution d'une requête type sur la table SURFACE avec différents index

Par ailleurs, un des aspects qui nous intéressait beaucoup concernait la distribution de la table et de son (ses) index sur des disques SCSI différents afin d'optimiser les temps de lecture. Malheureusement, le comparatif dont nous disposons ne permettait pas de façon univoque de conclure sur ce sujet puisque nous n'avons pas pu comparer des environnements informatiques identiques. L'ordinateur *Carpinus* présente, en ce qui a trait au stockage des tables, d'index et des segments, un seul disque très performant (SCSI-2 Ultra-Wide) alors que l'autre ordinateur, *Thor*, présente plusieurs disques mais d'une génération plus ancienne (SCSI). Les résultats que nous avons obtenus ne permettent donc pas de faire la part exacte entre ces deux variables. Cependant, le gain de temps que nous espérons obtenir en répartissant la table, l'index et le segment TMP ne s'est pas concrétisé puisque les meilleurs temps sont demeurés ceux de *Carpinus*. Nous en concluons que le type de disque, c'est-à-dire les variabilités de performance entre les différentes technologies SCSI, peut être plus déterminant que la distribution des tables, index et divers segments dans le cas où il y a d'importantes différences technologiques entre les disques (voir figure 9).

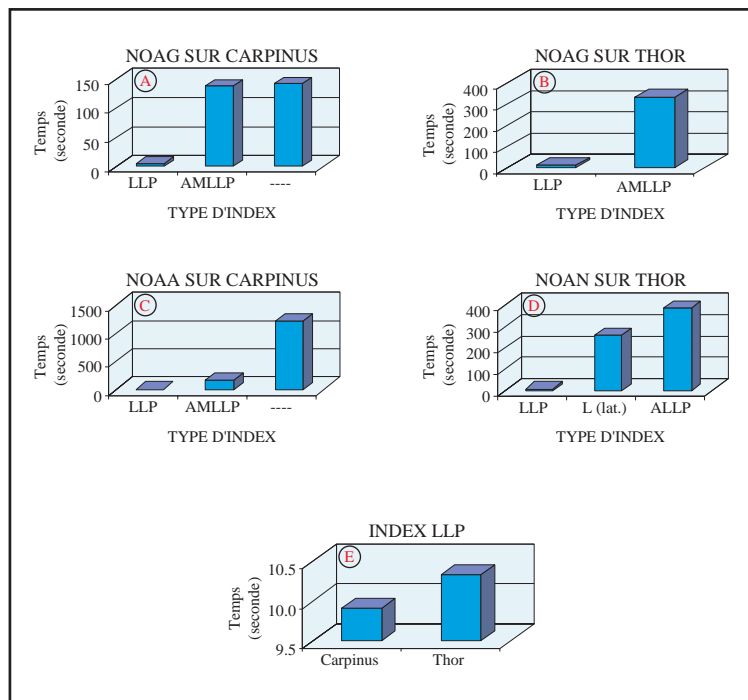


Figure 9 Temps pris pour l'exécution d'une requête type avec différents index

10 Recommandations

En guise de conclusion générale à ce travail nous proposons un diagramme qui présente, selon les paramètres étudiés, les facteurs les plus déterminants dans l'optimisation d'un jeu de données de grande envergure (figure 10). Ce diagramme a été construit à partir de données quantitatives et également qualitatives dans certains cas. En ordonné l'échelle du diagramme présente une évaluation qualitative de l'importance de chacun des paramètres étudiés selon une cote de 0 à 10 (10 étant considéré comme le degré le plus élevé).

La structure de l'index utilisé est de loin le facteur qui a le plus affecté les temps de traitements pour les raisons décrites précédemment. Vient ensuite les disques durs qui contiennent les tables et les index. En outre, nous avons observé que la composante matérielle est celle la plus sollicitée lors des requêtes de création des index et des mises à jour. La quantité de mémoire vive dont dispose l'ordinateur est un facteur important pour autant qu'il en soit attribuée une quantité proportionnelle au SGA de l'instance Oracle. Dans le cas présent, étant donné la taille de la table NOAG (800 Mo), celle-ci ne pouvait évidemment pas tenir en mémoire. Cependant, pour les tables de plus petite taille comme SURFACE, une partie non négligeable de celle-ci résidait en mémoire et offrait un gain de performance.

Curieusement, la structure de la table n'a pas joué un rôle aussi important que nous l'avions cru, les différences observées entre NOAA, NOAG et NOAN sont respectivement 9.91, 11.24, 10.33 secondes, ce qui représente un gain de 12% au mieux. Dernier point digne d'intérêt, la vitesse d'horloge du processeur ne joue finalement qu'un rôle mineur puisqu'il présente un taux d'utilisation (*Task Manager*) variant généralement entre 10 et 15 %. Ce qui représente pour un processeur à 200 et 400 MHz un gain total de 30 MHz pour effectuer le travail.

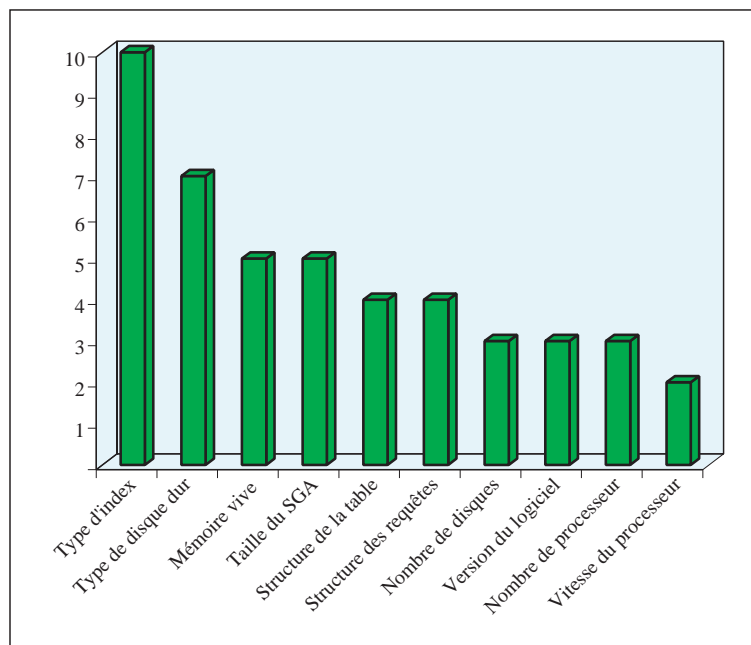


Figure 11 Facteurs qui influencent le plus l'optimisation d'un jeu de données important

Les facteurs identifiés dans cette étude ne sont pas les seuls à pouvoir influencer la performance du traitement des requêtes. Cependant, nous estimons que ces paramètres permettent d'obtenir plus de 90% en gain de performance visant l'optimisation d'un tel système. L'ordre d'importance des facteurs peut varier en fonction des types de bases de données ainsi que du nombre d'utilisateurs connectés. L'expérimentation demeure par conséquent le meilleur moyen d'optimiser les performances de son système de gestion.

Annexe 1

Tables	Champs	Type	Nb	Distinct	Min	Max	Moy	Description
NOAA	Annee	NUMBER	20 510 500	97	198	1993	1969.17	Année d'échantillonnage
	Mois	NUMBER		13	1	31	6.52	Mois d'échantillonnage
	Latitude	NUMBER (8,3)			-78	90	29.38	Latitude d'échantillonnage
	Longitude	NUMBER (8,3)			-99.983	763.883	-26.17	Longitude d'échantillonnage
	Profondeur	NUMBER (8,1)			0	8500	141.1	Profondeur d'échantillonnage
	Temp	NUMBER (8,3)			-3	34.9	14.2	Température (en °C)
	Sali	NUMBER (8,3)			0	43.792	33.45	Salinité (psu)
	Doxy	NUMBER (8,3)			0.01	11.97	5.31	Concentrations en oxygène dissout
	Phos	NUMBER (8,3)			0	4	0.8	Concentrations en phosphates (PO42-) dans l'eau
	Sili	NUMBER (8,3)			0	224.686	16.27	Concentrations en silice (SiO4) dans l'eau
Ntra	NUMBER (8,3)			0	52.078	10.34	Concentrations en nitrates (NO3-) dans l'eau	
NOAAP	Annee	NUMBER	20 168 338		1900	1993	1971.76	Année d'échantillonnage
	Mois	NUMBER			0	31	6.48	Mois d'échantillonnage
	Latitude	NUMBER (8,3)			-85.5	89.55	23.96	Latitude d'échantillonnage
	Longitude	NUMBER (8,3)			-554.68	180.825	14.05	Longitude d'échantillonnage
	Profondeur	NUMBER (8,1)			0	9000	134.16	Profondeur d'échantillonnage
	Temp	NUMBER (8,3)			-3	34.8	15.43	Température (en °C)
	Sali	NUMBER (8,3)			0	38.71	33.94	Salinité (psu)
	Doxy	NUMBER (8,3)			0.01	11.99	4.33	Concentrations en oxygène dissout
	Phos	NUMBER (8,3)			0	5.75	1.2	Concentrations en phosphates (PO42-) dans l'eau
	Sili	NUMBER (8,3)			0	200	28.4	Concentrations en silice (SiO4) dans l'eau
Ntra	NUMBER (8,3)			0	54	16.74	Concentrations en nitrates (NO3-) dans l'eau	
GLACE	Annee	NUMBER	1 278 080		1953	1990	1971.5	Année de la mesure
	Mois	NUMBER			1	12	6.5	Mois de la mesure
	Latitude	NUMBER (9,4)			33.6	90	62.26	Latitude de la mesure
	Longitude	NUMBER (9,4)			-179.86	179.98	-29.28	Longitude de la mesure
	Valeurs	NUMBER			0	10	3.36	Présence de glace (échelle arbitraire)

Tableau d'inventaire des tables NOAA, NOAP et GLACE

Table	Variable	Valeur obtenue	Valeur Réelle	Nombre d'enregistrements incorrects	Tableau
NOAA	Année Minimum	198	1898	Nb d'enregistrements ayant année = 198 : 16	A
	Mois maximum	31	12	Nb d'enregistrements ayant mois>12 : 5	B
	Longitude maximum	763.883	150	Nb d'enregistrements ayant longitude > 181 : 8	C
NOAAP	Mois maximum	31	12	Nb d'enregistrements ayant mois>12 : 5	D
	Longitude minimum	-554.68	-180.68	Nb d'enregistrements ayant longitude < -181 : 5	E

Tableau synthèse des valeurs douteuses dans les tables NOAA et NOAP

Annee	Mois	Latitude	Longitude	Profondeur	Temp	Sali	Doxy	Phos	Sili	Ntra
198	12	40.067	19.1	0	15.19
198	12	40.067	19.1	10	15.42	38.23
198	12	40.067	19.1	20	15.41	38.25
198	12	40.067	19.1	30	15.36	38.25
198	12	40.067	19.1	50	15.197	38.27
198	12	40.067	19.1	75	14.973	38.274
198	12	40.067	19.1	100	14.641	38.574
198	12	40.067	19.1	125	14.405	38.578
198	12	40.067	19.1	150	14.258	38.591
198	12	40.067	19.1	250	14.092	38.66
198	12	40.067	19.1	300	14.006	38.66
198	12	40.067	19.1	400	13.856	38.66
198	12	40.067	19.1	500	13.76	38.65
198	12	40.067	19.1	600	13.769	38.65
198	12	40.067	19.1	700	13.758	38.65
198	12	40.067	19.1	800	13.713	38.65

Tableau présentant les valeurs douteuses pour l'année 198 dans la table NOAA

Annee	Mois	Latitude	Longitude	Profondeur	Temp	Sali	Doxy	Phos	Sili	Ntra
1949	31	20	-96	0	23.3
1949	31	20	-96	10	23.3
1949	31	20	-96	20	23.3
1949	31	20	-96	30	23.4
1949	31	20	-96	50	23.4

Tableau présentant les valeurs douteuses pour le mois 31 dans la table NOAA

Annee	Mois	Latitude	Longitude	Profondeur	Temp	Sali	Doxy	Phos	Sili	Ntra
1976	8	-35.855	763.883	0	.	35.56
1976	8	-35.855	763.883	10	.	35.59
1976	8	-35.855	763.883	150	17.036	35.578
1976	8	-35.855	763.883	250	14.942	35.38
1976	8	-35.855	763.883	300	14.001	35.281
1976	8	-35.855	763.883	400	12.17	35.192
1976	8	-35.855	763.883	500	10.551	34.925
1976	8	-35.855	763.883	600	9.413	34.781

Tableau présentant les valeurs douteuses pour la longitude 763.883 dans la table NOAA

Annee	Mois	Latitude	Longitude	Profondeur	Temp	Sali	Doxy	Phos	Sili	Ntra
1949	31	20	-96	0	23.3
1949	31	20	-96	10	23.3
1949	31	20	-96	20	23.3
1949	31	20	-96	30	23.4
1949	31	20	-96	50	23.4

Tableau présentant les valeurs douteuses pour le mois 31 dans la table NOAP

Annee	Mois	Latitude	Longitude	Profondeur	Temp	Sali	Doxy	Phos	Sili	Ntra
1979	3	-61.28	-544.68	0	1.27	34.36
1979	3	-61.28	-544.68	10	1.28	34.37
1979	3	-61.28	-544.68	20	1.281	34.371
1979	3	-61.28	-544.68	30	1.289	34.379
1979	3	-61.28	-544.68	50	1.312	34.382

Tableau présentant les valeurs douteuses pour la latitude -544.68 dans la table NOAP

Annexe 2

Configuration des ordinateurs Carpinus et Thor		
	Carpinus	Thor
Processeur	(2) Intel Pentium Pro @180 MHz	(1) Intel Pentium II MMX @350 MHz
Mémoire vive (RAM)	256 Mo	256 Mo
Mémoire virtuelle	512 Mo	512 Mo
Carte graphique	Matrox Graphics MGA Millennium 8 Mo	Matrox Graphics Mystique G200 AGP 8 Mo
Disque(s) dur(s)	Quantum FireBall SE3 IDE	WDC AC34 000L IDE
	Seagate Barracuda Ultra Wide SCSI II	Quantum VP32210 SCSI
		Quantum FireBall TM21105 SCSI
		HP C3725S SCSI
Système d'exploitation (OS)	Windows NT 4.0 SP 4	Windows NT 4.0 SP 5

Tableau comparatif des composantes des ordinateurs Carpinus et Thor

Annexe 3

Liste des Commandes SQL utilisées pour évaluer l'espace mémoire et disque dur:

Visualiser les tables, vues, synonymes et séquences créées par l'utilisateur:

```
SQL> select * from cat;
```

Visualiser les privilèges de l'utilisateur connecté:

```
SQL> select * from user_role_privs;
```

Visualiser le nombre de tables qu'un utilisateur possède (SYS dans cet exemple):

```
SQL> select owner, count(*) from dba_tables where owner!='SYS' group by  
owner;
```

Visualiser le nombre d'octets et de blocs qu'occupent les segments d'une table donnée:

```
SQL> select extent_id, bytes, blocks from user_extents where  
segment_name='NOAG';
```

Visualiser les segments de tables et d'index pour le compte courant:

```
SQL> select segment_name from user_extents;
```

Visualiser le nombre total d'octets pour une table:

```
SQL> select segment_name, extent_ID, bytes  
from user_extents  
where segment_name='NOAA'
```

Visualiser le nombre total d'octets et de blocs occupés par les espaces de stockage (utilisateur, système, segments d'annulation et fichiers temporaires) pour l'instance:

```
SQL> select tablespace_name, sum(bytes), sum(blocks) from  
sys.dba_free_space  
group by tablespace_name;
```

Liste des Commandes SQL utilisées pour modifier l'espace alloué à des segments:

Exemple de commande qui permet de modifier la taille physique des fichiers qui contiennent les segments de l'instance:

```
SQL> alter database datafile 'f:\orant\database\usrlorcl.ora' resize  
3000M;
```

```
SQL> alter database datafile 'f:\orant\database\tmp1orcl.ora' resize  
640M;
```

Exemple de commande qui permet de modifier les paramètres d'un segment de l'instance:

```
SQL> alter tablespace tmp default storage (pctincrease 10);
```

Commande qui assigne les espaces de stockage permanent et temporaires allant être utilisés par défaut par l'utilisateur:

```
SQL> alter user geotop default tablespace usr2 temporary tablespace tmp;
```

Commande de création d'un espace de stockage USR2 avec spécification de sa taille:

```
SQL> create tablespace usr2 datafile 'q:\orant\database\usr2orc1.ora'  
size 900M;
```

Séquence de commandes qui permet de détruire un segment d'annulation et de le reconstruire:

```
SQL> alter rollback segment rbs1 offline;  
SQL> drop rollback segment rbs1;  
SQL> create rollback segment rbs1 tablespace rbs1;  
SQL> alter rollback segment rbs1 online;
```

Liste des Commandes SQL utilisées pour créer des tables, vues ou des index:

Exemple de commande de création d'une table avec des paramètres de dimension (exemple de la table NOAN):

```
SQL> create table noan(annee number(4),lat number(6,3),lon number(7,3),  
prof number(6,1), temp01 number(4,2), temp02 number(4,2), temp03  
number(4,2), temp04 number(4,2), temp05 number(4,2), temp06  
number(4,2), temp07 number(4,2), temp08 number(4,2), temp09  
number(4,2), temp10 number(4,2), temp11 number(4,2), temp12  
number(4,2), sali01 number(4,2), sali02 number(4,2), sali03  
number(4,2), sali04 number(4,2), sali05 number(4,2), sali06  
number(4,2), sali07 number(4,2), sali08 number(4,2), sali09  
number(4,2), sali10 number(4,2), sali11 number(4,2), sali12  
number(4,2), doxy01 number(4,2), doxy02 number(4,2), doxy03  
number(4,2), doxy04 number(4,2), doxy05 number(4,2), doxy06  
number(4,2), doxy07 number(4,2), doxy08 number(4,2), doxy09  
number(4,2), doxy10 number(4,2), doxy11 number(4,2), doxy12  
number(4,2), phos01 number(4,3), phos02 number(4,3), phos03  
number(4,3), phos04 number(4,3), phos05 number(4,3), phos06  
number(4,3), phos07 number(4,3), phos08 number(4,3), phos09  
number(4,3), phos10 number(4,3), phos11 number(4,3), phos12  
number(4,3), sili01 number(6,3), sili02 number(6,3), sili03  
number(6,3), sili04 number(6,3), sili05 number(6,3), sili06  
number(6,3), sili07 number(6,3), sili08 number(6,3), sili09  
number(6,3), sili10 number(6,3), sili11 number(6,3), sili12  
number(6,3), ntra01 number(5,3), ntra02 number(5,3), ntra03  
number(5,3), ntra04 number(5,3), ntra05 number(5,3), ntra06  
number(5,3), ntra07 number(5,3), ntra08 number(5,3), ntra09  
number(5,3), ntra10 number(5,3), ntra11 number(5,3), ntra12  
number(5,3))  
storage (initial 600M next 20M pctincrease 0);
```

Exemple d'enregistrement d'un résultat de requête dans une nouvelle table en spécifiant les paramètres de dimension:

```
SQL> create table mininoag(annee, mois, lat, lon, prof, temp, sali,
doxy, phos, sili, ntra)
storage (initial 20M next 5M pctincrease 0)
as select annee, mois, lat, lon, prof, temp, sali, doxy, phos,
sili, ntra
from noag where annee <1931;
```

Exemple d'enregistrement d'un résultat de requête dans une nouvelle table en spécifiant l'espace de stockage (USR2) et les paramètres de dimension:

```
SQL> create table noan tablespace usr2
storage (initial 600M next 10M pctincrease 0)
as select * from noan@carpinus where annee=1878;
```

Exemple d'enregistrement d'un résultat de requête dans une nouvelle vue:

```
SQL> create view sv50(annee, lat, lon, prof, temp01, sali01, doxy01,
phos01, sili01, ntra01)
as select annee, latitude, longitude, profondeur, temp, sali, doxy,
phos, sili, ntra
from noaa where annee=1950 and mois=01;
```

Exemple de commande de création d'un index unique multiple avec des paramètres de dimension:

```
SQL> create unique index noag_amlp on noag(annee, mois, lat, lon, prof)
storage (initial 600M next 20M pctincrease 0);
```

Exemple de commande de création d'un index multiple en spécifiant l'espace de stockage (NDX2) et les paramètres de dimension:

```
SQL> create index noan_allp on noan(annee, lat, lon, prof)
tablespace ndx2
storage (initial 500M next 20M pctincrease 0);
```

Séquence de commande qui peut être enregistrée dans un fichier .SQL et qui permet, en plus de créer un index, de connaître le temps pris pour réaliser l'opération et de le conserver dans un fichier texte:

```
SQL> set timing on
SQL> spool g:\nobel\sql\ndx2.dat
SQL> create unique index noag_amlp on noag(annee, mois, lat, lon, prof)
tablespace ndx1 storage (initial 600m next 10m pctincrease 0);
SQL> spool off
```

Liste des Commandes SQL utilisées pour faire des insertions ou des mise à jour:

Commande d'insertion des sites de la table NOAA dans la table NOAG en utilisant l'option *Average* (AVG) qui a permis d'éliminer les valeurs dupliquées en en faisant des moyennes:

```
SQL> insert into noag select annee, mois, latitude, longitude,
profondeur, AVG(temp), AVG(sali),
AVG(doxy), AVG(phos), AVG(sili), AVG(ntra)
```

```
from noaa where annee = 1963 and mois > 4 and mois <=6
group by annee, latitude, longitude, profondeur, mois;
```

Commande d'insertion des sites de la table NOAG dans la table NOAN en utilisant l'option *Distinct* qui permet d'insérer une seule occurrence pour un même site:

```
SQL> insert into noan(annee,lat,lon,prof)
select distinct annee, lat, lon, prof from noag where annee > 1897
and annee <= 1898 ;
```

Commande de mise à jour des données dans la table NOAN à partir de NOAG par équivalence des champs qui compose l'identifiant unique des sites:

```
SQL> update noan set temp01=(select temp from noag where noag.annee =
1898 and noan.lat = noag.lat and noan.lon = noag.lon and noan.prof
= noag.prof and noag.mois = 01) where noan.annee = 1898;
```

Exemple de script permettant de sauvegarder le contenu d'une requête pour fin d'exportation. Les champs seront concaténés et séparés par un séparateur vertical *Pipe* (|), le fichier résultant pourra être rechargé dans Oracle à l'aide du module *SQL Loader*:

```
set echo off
set heading off
set pagesize 0
set termout off
set trimspool on
spool g:\nobel\noan2.unl
select annee||' '||lat||' '||lon||' '||prof||' '||temp08||' ' from
noan2;
spool off
set trimspool off
set echo on
set heading on
set pagesize 24
set termout on
```