

# Qu'est-ce qu'une date ?

Philippe PRADOS

[pp@philippe.prados.name](mailto:pp@philippe.prados.name)

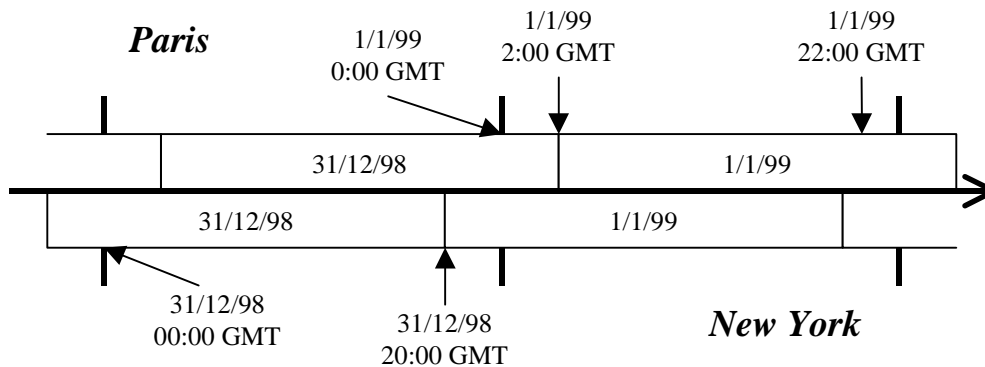
## Avant-propos

Ce document explique comment gérer les dates dans une application internationale. Les applications Internet doivent pouvoir être utilisées dans le monde entier. La notion de date est alors difficile à appréhender. Ce document explique ce qu'est vraiment une date, et comment la gérer dans une application. Les approches naïves classiques ne suffisent pas.

Le temps s'écoule inexorablement dans la même direction bien qu'aucune loi physique ne l'impose (voir "Une brève histoire du temps" de S. Hawking). Pour des raisons historiques et astronomiques, les humains ont inventé le calendrier avec les notions de jour, de mois et d'année.

Comme chacun sait, la terre possède 24 fuseaux horaires. Lorsqu'il est midi à New York, il est 18 heures à Paris. L'heure de référence est localisée sur le méridien de Greenwich. Il est alors possible d'avoir une heure absolue et de la convertir suivant la localisation de l'utilisateur.

Au même moment, deux personnes dans deux pays différents peuvent avoir une date différente. Il peut être lundi à New York et mardi à Paris. Cela complique beaucoup la notion de *date* dans les applications.



Par exemple, une application à Paris demande à un utilisateur d'indiquer une date. Le 1<sup>er</sup> janvier 1999 par exemple. Comment manipuler celle-ci à New York ? Par convention, on peut la sauvegarder comme étant la première seconde du jour indiqué pour la localisation de l'utilisateur. La date du 1/1/99 Paris sera mémorisée en 1/1/99 2:00 GMT. Si un utilisateur de New York indique la même date, le programme mémorisera le 31/12/98 20:00 GMT.

L'utilisateur de Paris désire trouver tous les dossiers du 1/1/99 Paris. Il faut sélectionner tous les dossiers compris entre le 1/1/99 2:00 GMT et le 2/1/99 2:00 GMT. En effet, en GMT, ces deux instants représentent la première et la dernière seconde de la journée pour Paris. Le dossier constitué à New York ne sera pas sélectionné alors qu'ils auront été construits avec la même date.

L'utilisateur de New York désire trouver tous les dossiers du 1/1/99 New York. Il faut trouver tous les dossiers compris entre le 31/12/98 20:00 GMT et le 1/1/99 20:00 GMT. Les deux dossiers de New York et de Paris seront sélectionnés.

Chaque dossier possède un attribut indiquant un instant de création en GMT. Il est automatiquement généré par le programme. Imaginons que les deux dossiers ont été créés le 1/1/99 0:00 GMT.

Si l'utilisateur de Paris demande les dossiers créés le 1/1/99 Paris, il faut demander les dossiers dont la date de création est comprise entre le 1/1/99 2:00 GMT et le 2/1/99 2:00 GMT. Aucun dossier est trouvé.

Si l'utilisateur de New York demande les dossiers créés le 1/1/99 New York, il faut demander les dossiers dont la date de création est comprise entre le 31/12/98 20:00 GMT et le 1/1/99 20:00 GMT. Les deux dossiers sont trouvés.

Si les dossiers avaient été créés le 1/1/99 22:00 GMT, les réponses seraient différentes. L'utilisateur de Paris obtient les dossiers, l'utilisateur de New York n'obtient rien.

## 1. COMMENT DEVELOPPER ?

De ces exemples, on constate que la notion de date est difficile à appréhender. Une date correspond à une tranche de temps dont les bornes dépendent de la localisation de l'utilisateur. Au niveau du développement, différentes situations peuvent arriver. Il faut pouvoir comparer des instants et des dates.

- Un instant correspond à l'identification d'un moment précis sur la flèche du temps. Par convention, un instant est exprimé en date heure GMT.
- Une date correspond à la tranche de temps entre la première et la dernière seconde de la journée pour l'utilisateur. Par simplification, on mémorise la date en identifiant l'instant de la première seconde de la journée. Des calculs simples permettent de trouver la dernière seconde. Il suffit d'ajouter 86 400 (le nombre de seconde dans une journée). Attention, cela ne tient pas compte des jours particuliers de 23 et 25 heures lors du passage à l'heure d'été et l'heure d'hiver. Pour ces dates particulières, il faut ajouter 82 800 ou 90 000.

Comparer deux instants n'est pas difficile. Il suffit d'utiliser les opérateurs arithmétiques classiques.

```
instant1 == instant2
instant1 < instant2
instant1 > instant2
instant1 <= instant2
instant1 >= instant2
```

Cela permet de comparer les relations d'ordre sur la flèche du temps.

Comparer deux instants avec une sémantique de date est plus compliqué. Il faut convertir les instants pour les ajuster à la première et à la dernière seconde de la journée suivant la localisation de l'utilisateur.

Pour savoir si deux dates correspondent au même jour, il faut savoir si les deux espaces de temps se chevauchent. Pour répondre à la question "jusqu'au", il faut utiliser la dernière seconde de la journée. Pour répondre à la question "à partir de", il faut utiliser la première seconde.

<code>date1 == date2</code>	<code>minDay(date1) &lt; maxDay(date2) &amp;&amp; maxDay(d1) &gt; minDay(date2)</code>
<code>date1 &lt; date2</code>	<code>date1 &lt; minDay(date2)</code>
<code>date1 &lt;= date2</code>	<code>date1 &lt;= maxDay(date2)</code>
<code>date1 &gt; date2</code>	<code>date1 &gt; maxDay(date2)</code>
<code>date1 &gt;= date2</code>	<code>date1 &gt;= minDay(date2)</code>

`minDay()` permet d'obtenir la première seconde de la journée où se situe l'instant pour la localisation courante. `maxDay()` permet d'obtenir la dernière seconde. Il est à noter que le 31/12/98 et le 1/1/99 New York répondent positivement à l'égalité avec le 1/1/99 Paris.

Lorsqu'un utilisateur entre une date textuelle, il faut la convertir en instant, correspondant à la première ou à la dernière seconde de la journée locale. On présente un instant GMT à l'utilisateur en la convertissant en texte suivant sa localisation.

Avec cette approche, les dates ne sont plus des informations figées, mais le résultat d'un calcul. L'affichage du même dossier peut être différent à divers moments de la journée. Il n'est pas possible de convertir une date New York en date Paris. En effet, il existe dans ce cas deux solutions. Le 1/1/99 New York peut représenter le 31/12/98 ou le 1/1/99 Paris. Chaque affichage de date doit sélectionner l'instant le plus important : le début, le milieu, la fin de la journée... A partir de l'instant sélectionné, il est possible de le traduire en date locale. Par exemple, si le début de la journée du 1/1/99 New York est important, cela correspond au 31/12/98 Paris. Si la fin de la journée du 1/1/99 New York est importante, cela correspond au 1/1/99 Paris.

Pour éviter ces ambiguïtés, il ne faut plus afficher de date mais des couples dates/heures. Ainsi, l'utilisateur aperçoit toujours un instant sur la flèche du temps. Il peut entrer uniquement la date, cela correspond implicitement à une heure à zéro ou à 23:59 pour sa localisation. Il peut également entrer une date et une heure. A lui de tenir compte des décalages horaire éventuel.

Par exemple, un utilisateur de Paris manipule une date d'échéance sur un dossier du Japon. S'agissant d'une date d'échéance, s'il n'indique pas d'horaire, la dernière seconde de la journée est prise en compte. Il désire informer le système d'une date d'échéance du 1/1/99 Japon (2/1/99 8:59 GMT). Il peut entrer le 2/1/99 10:59 Paris (2/1/99 8:59 GMT). Pour simplifier, il entre la veille pour avoir une marge suffisante. Il indique le 1/1/99 Paris (2/1/99 1:59 GMT). L'utilisateur de Tokyo, en consultant le dossier, verra une échéance au 2/1/99 10:59 Tokyo (2/1/99 1:59 GMT).

Contextuellement, une zone horaire peut être associée à un dossier. Ainsi, toutes les dates de celui-ci l'utiliseront pour les saisies et les affichages.

Pour réaliser cela, il faut un composant logiciel permettant à l'utilisateur d'entrer un instant au format date/heure. Si l'heure n'est pas renseignée, un paramètre permet d'indiquer s'il faut ajouter une heure à zéro ou à 23:59.

Dans une approche client/serveur, le client doit parfois indiquer sa zone horaire au serveur pour que celui-ci puisse effectuer les ajustements nécessaires lors des requêtes.

### 1.1 Date absolue

Une approche consiste à sélectionner la tranche de temps la plus importante pour une date. Il faut choisir une tranche de temps comprenant la première et la dernière seconde où la date existe dans le monde.

Pour la date du 1/1/99, il faut sélectionner la tranche de temps allant du 31/12/98 12:00 GMT au 2/1/99 11:59:59. Avec ces limites, la journée dure 48 heures !

Dans cette optique, il faut modifier certains algorithmes. Pour répondre à la question "à partir de", il faut utiliser la dernière seconde de la journée de 48 heures ; pour "jusqu'au", il faut utiliser la première seconde. Les réponses seront en avance ou en retard, mais toujours dans l'espace considéré.

Par exemple, pour répondre à la question "jusqu'au 1/1/99" à Paris, le test s'effectue sur le moment 31/12/99 12:00 GMT (31/12/99 14:00 Paris). L'utilisateur possède une marge de manœuvre, toujours valide dans le monde.

### 1.2 Date sémantique

Suivant les applications, la date peut être *sémantique*. C'est-à-dire qu'elle ne représente pas un espace dans la flèche du temps, mais représente la vision de l'utilisateur suivant le contexte.

Par exemple, à Paris, si vous regardez l'agenda d'une personne de New York, vous adaptez mentalement les horaires pour tenir compte du décalage. Vous savez, en regardant l'agenda, qu'il faut interpréter les dates avec la vision de New York. Le 1/1/99 est interprété suivant son contexte par l'utilisateur.

Cette approche permet de simplifier la programmation. Les problèmes de fuseaux horaires ne sont plus traités par l'application, mais par l'utilisateur. A lui de prendre les marges suffisantes pour adapter les dates suivant le contexte.

L'application décide de fonctionner avec une heure de référence. En générale, l'heure des utilisateurs. L'application cliente force la zone horaire à la valeur de référence. S'il n'est pas possible de modifier la zone horaire, sur une applet Java par exemple, les routines d'affichages ou d'introduction d'une date suppriment l'effet de la zone horaire pour obtenir l'heure de référence.

SQL Serveur de Microsoft ne tient pas compte des fuseaux horaires. Les dates sont mémorisées avec l'approche sémantique. Demander tous les enregistrements correspondant à une date particulière, permet de retrouver les enregistrements pour l'heure de référence. Il est possible

d'effectuer les ajustements nécessaires dans l'application pour tenir compte de la localisation précise de l'utilisateur, mais dans ce cas, il faut modifier les requêtes pour demander les enregistrements compris entre deux instants.

Avec les dates sémantiques, il est possible de tenir un peu compte des fuseaux horaires. Pour cela, il faut choisir comme date de référence : Greenwich. Suivant les requêtes, il faut utiliser la borne inférieure ou supérieure de la *date absolue*. Il faut se placer dans tous les fuseaux horaires et regarder comment interpréter les requêtes. Deux stratégies sont possibles : optimiste et pessimiste.

Pour attendre la fin d'une journée, l'approche optimiste attend le dernier moment du dernier fuseau horaire. L'approche pessimiste attend le dernier moment du premier fuseau horaire.

Avec l'approche optimiste, la date peut être dépassée pour le premier fuseau (GMT-12) mais pas encore pour le dernier (GMT+12). Donc, la date est encore valide quelque part.

Avec l'approche pessimiste, si la date est dépassée quelque part, on considère qu'elle est dépassée partout, même si localement ce n'est pas le cas. Par exemple, lorsque à GMT-12, la journée est terminée, elle ne fait que commencer à GMT+12.

L'approche optimiste attend la dernière seconde où une date existe dans le monde ; l'approche pessimiste attend la dernière seconde du premier fuseau horaire. On peut proposer la table de vérité suivante :

	Classique	Optimiste	Pessimiste
Avant le 1/1/99	< 1/1/99	< 1/1/99 +12:00	< 1/1/99 -12:00
Jusqu'au 1/1/99	<= 1/1/99	< 1/1/99 +36:00	< 1/1/99 +12:00
Depuis le 1/1/99	> 1/1/99	> 1/1/99 +36:00	> 1/1/99 +12:00
A partir du 1/1/99	>= 1/1/99	> 1/1/99 -12:00	> 1/1/99 +12:00

Pour chaque question concernant les dates sémantique, l'approche classique utilise les opérateurs traditionnels. Les autres ajustent le moment suivant le premier ou le dernier fuseau horaire à prendre en compte pour répondre à la question. Tant que la réponse à la question peut être valide quelque part, l'approche optimise en tient compte. Si la réponse est invalide quelque part, l'approche pessimiste rejette la question.

## 2. CONCLUSION

Pourquoi ces difficultés ? Les applications Internet sont mondiales. Le même serveur alimente des clients du monde entier. Chacun utilise une zone horaire spécifique, différente de la zone horaire du serveur. Cela entraîne les difficultés évoquées dans ce document.

Auparavant, les applications fonctionnaient dans un cadre restreint. Les utilisateurs étaient géographiquement proches du serveur. Ils utilisaient le même fuseau horaire que le serveur. Dans cette situation particulière, il n'est pas nécessaire de tenir compte des difficultés évoquées dans ce document.

S'il est nécessaire d'être rigide vis-à-vis des instants sur la flèche du temps, il faut manipuler les dates comme décrites dans le chapitre "Comment développer ?". Si l'utilisateur est capable de tenir compte du contexte, et que l'application ne doit pas avoir de traitement dépendant de moment précis sur la flèche du temps, il est possible d'utiliser l'approche décrite dans le chapitre "Date sémantique".

On imagine souvent, à tort, que l'utilisation de moment GMT suffit à régler tous les problèmes. Nous avons démontré que ce n'est pas le cas. La notion de date est floue. Vous devez en tenir compte dès le début du développement pour ne pas avoir de surprise lorsqu'un client japonais voudra consulter un dossier américain.