A la Recherche du Temps Perdu

- Bases de données temporelles -

Cédric du Mouza – CNAM – dumouza@cnam.fr
(d'après un support de Anne Doucet – LIP6)

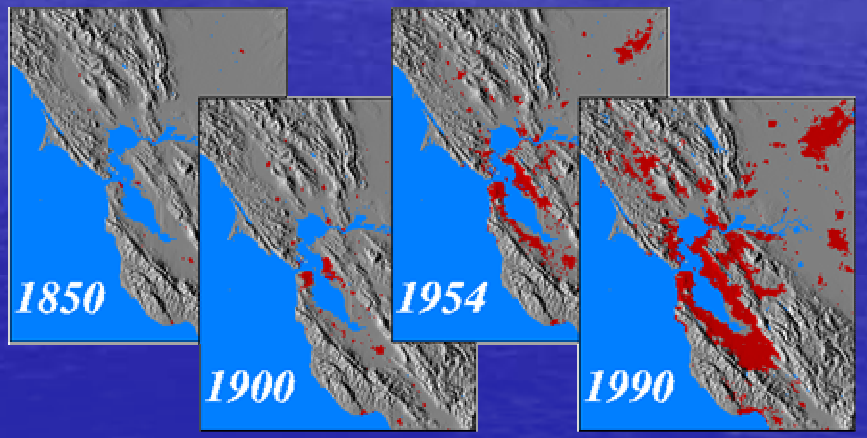
Bases de données temporelles

- Introduction et problématique
- Différentes approches de BD temporelles
- Modélisation et représentation du temps
- Langages de requêtes temporels
- Conception de BD temporelles

Introduction (1/6)

Beaucoup d'applications ont besoin de gérer le temps dans leur BD :

- gestion des enseignements d'une UFR
- comptabilité
- budgets
- entrepôts de données et olap
- gestion des stocks
- S.I.G
- assurances
- BD légales
- médecine
- réservations etc.



Introduction (2/6)

- Quasiment toutes les applications ont besoin du temps
- Intérêt de leur fournir des fonctionnalités temporelles *directement dans le SGBD* (intégrité, sécurité, perf.)

Exemple :

Nom	Salaire	Titre	DateNais	Start	Stop
Toto	60000	assistant	12 05 75	01 01 01	31 09 06
Toto	70000	assistant	12 05 75	01 10 06	31 09 07
Toto	70000	MCF	12 05 75	01 10 07	31 09 08
Toto	70000	Prof	12 05 75	01 10 08	01 01 10

- premiers pbs :
  - DateNais et Nom ne varient pas.
  - salaire de Toto au 10 01 2010 ?

Introduction (3/6)

- requêtes simples deviennent compliquées :
  - quel est le salaire actuel de Toto ?

```
select Salaire from Employés
where Nom = 'Toto'
and Start <= CURRENT_DATE
and Stop >= CURRENT_DATE
```

} devrait être implicite

  - quel est l'historique du salaire de Toto ?

Résultat

Nom	Salaire	Start	Stop
Toto	60000	01 01 01	31 09 06
Toto	70000	01 10 06	01 01 10

} Un seul tuple suffirait

Introduction (4/6)

- Il faut trouver tous les intervalles de temps "fusionnables" pour lesquels le salaire de Toto est le même : ***coalescence***
  - Très compliqué et coûteux en SQL directement (pire que division)
  - par programme : non-déclaratif
  - changer schéma en (Nom, Salaire, Start, Stop), (Nom, Titre, Start, Stop) ne marche que pour une requête

Introduction (5/6)

- Jointure temporelle : «historique des employés avec leur titre et salaire»

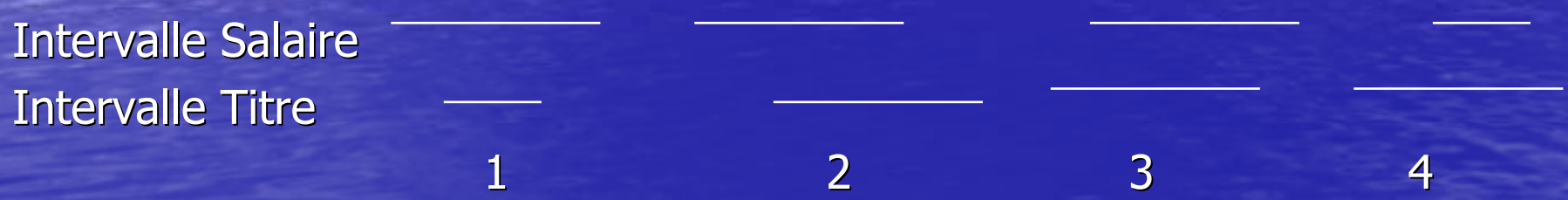
SalaireEMP

Nom	Salaire	Start	Stop
Toto	60000	01 01 01	31 09 06
Toto	70000	01 10 06	01 01 10

FonctionEMP

Nom	Titre	Start	Stop
Toto	assistant	01 01 01	31 09 07
Toto	MCF	01 10 07	31 09 08
Toto	Prof	01 10 08	01 01 10

Il faut exprimer la condition de jointure pour tous les cas possibles (4) entre l'intervalle pour le salaire et l'intervalle pour le titre



Introduction (6/6)

- L'utilisation d'un SGBD temporel permet
  - des schémas plus simples
  - des requêtes plus simples
    - select Salaire from Employes where Nom = 'Toto'
    - select SalaireEMP.Nom, Salaire, Titre from SalaireEMP, FonctionEMP where SalaireEMP.Nom = FonctionEMP.Nom
  - moins de code procédural
- Bénéfices :
  - code des applis. moins complexe
  - plus facile à produire, à corriger, à maintenir
  - meilleures performances car sous contrôle du SGBD

SGBD temporels : différentes approches (1/2)

- Sans changer la technologie existante :
  - **utiliser un type date et programmer tous les traitements**
    - requêtes complexes, traitements coûteux, erreurs possibles
    - maintenance compliquée: modif. dans les programmes
  - **utiliser un TAD pour le temps**
    - il faut que le SGBD offre des TAD
    - Simplifie les requêtes et les traitements
    - uniformité si le TAD est en bibliothèque
    - pas d'optimisation pour des opérations temporelles

SGBD temporels : différentes approches (2/2)

- En changeant la technologie
  - **Extension d'un modèle existant (la plus utilisée)**
    - utiliser les concepts existants pour définir des concepts temporels (si possible)
    - ajout d'opérateurs à l'algèbre (ex. jointure temporelle)
    - avantage : ne pas redéfinir tout le SGBD
    - inconvénient : manque de généralité et d'extensibilité (seuls certains concepts sont rendus temporels)
  - **Généralisation d'un modèle existant**
    - tous les concepts et opérations sont rendus temporels (schéma, données, ...)
    - plus compliqué et coûteux à mettre en oeuvre

Modélisation du temps : concepts (1/2)

- Instant :
  - point temporel dans le monde réel
- Structure du temps :
  - linéaire, avec alternative, périodique
- Limites du temps:
  - Illimité, avec origine (limite inf.), limité (limites inf. et sup.)
  - limites connues ou inconnues
- Densité du temps:
  - Discrète (isomorphe à $\mathbb{N}$), dense (isom. $\mathbb{Q}$), continue (isom. $\mathbb{R}$)
  - le plus raisonnable a représenter : temps discret limité (ou origine)
- Chronon (temps discret) :
  - plus petite unité de temps représentable
  - un événement a lieu en t si il a lieu en un instant appartenant au chronon t.

Modélisation du temps : concepts (2/2)

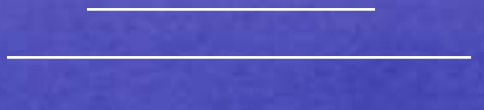
- Intervalle temporel :
  - période de temps ayant un début et une fin
- Elément temporel :
  - ensemble d'intervalles temporels (ex. les vacances)
- Manipulation des intervalles :
  - vu comme des ensembles d'instants : par union, différence....
    Le résultat peut être un élément temporel
  - comparaison par les prédicats d'Allen
- Période de vie :
  - période pendant laquelle un objet existe

Prédicats d'Allen [1983]

Jeu complet de prédicats pour manipuler les intervalles de temps

- I1 before (after) I2
  «précède»

- I1 during (contains) I2
  «pendant»



- I1 overlaps (overlapped_by) I2
  «chevauche»

- I1 meets (met_by) I2
  «jouxte»

- I1 starts (started_by) I2
  «commence»

- I1 finishes (finished_by) I2
  «finit»

- I1 equal I2

Modélisation : notions de temps

- Temps utilisateur :
  - non-interprété par le système, uniquement par l'utilisateur
  - ex : date de naissance
- Temps de validité :
  - indique quand un fait est valide dans le monde réel
  - passé et futur. Alternatives possibles.
  - utilisé dans les requêtes et les contraintes.
  - now : présent dans le monde réel
- Temps de transaction :
  - indique quand un fait est enregistré dans la BD
  - utilisation : tracer les erreurs et corrections, croyance...
  - uniquement passé et linéaire
  - until-change : état courant de la BD
- Autres : temps de décision, etc...

Différents type de BD temporelles

- BD snapshot : non-temporelle
- BD historique :
  - type le plus répandu
  - temps de validité = histoire du monde réel
  - besoin de langage de requête sophistiqué (raisonnement temporel)
- BD rollback :
  - temps de transaction : stocke les états successifs de la BD
  - peut être utilisé pour du versionnement
- BD bi-temporelle :
  - temps de validité + temps de transaction
  - propriétés des BD historiques et de BD rollback

Estampillage temporel des données (1/6)

- Association du temps aux items de la BD
- Dépend de :
  - la granularité de ce qui est estampillé
    - tuples
    - objets
    - la BD entière
    - attributs
  - la nature des estampilles temporelles
    - instants
    - intervalles
    - éléments temporels
    - avec ou sans alternatives

Estampillage temporel des données (2/6)

- Estampillage de tuples :
  - chaque dimension temporelle correspond à 1 (ou +) attributs spéciaux
  - approche par extension de technologie
  - pb :
    - fragmentation des données d'un même item sur plusieurs tuples
    - redondance pour les attributs qui varient peu ou pas (ex. DateNais)

Nom	Salaire	Titre	DateNais	Start	Stop
Toto	60000	assistant	12 05 75	01 01 01	31 09 06
Toto	70000	assistant	12 05 75	01 10 06	31 09 07
Toto	70000	MCF	12 05 75	01 10 07	31 09 08
Toto	70000	Prof	12 05 75	01 10 08	01 01 10

Estampillage temporel des données (3/6)

- Estampillage d'objet
  - même principe que tuple
  - selon la granularité des sous-objets
    - augmentation de la fragmentation horizontale
    - diminution de la redondance : seuls les oid sont dupliqués

oid	Nom	Salaire	Titre	Naissance	Start	Stop
1	Toto	60000	assistant	< 2 >	01 01 01	31 09 06
1	Toto	70000	assistant	< 2 >	01 10 06	31 09 07
1	Toto	70000	MCF	< 2 >	01 10 07	31 09 08
1	Toto	70000	Prof	< 2 >	01 10 08	01 01 10

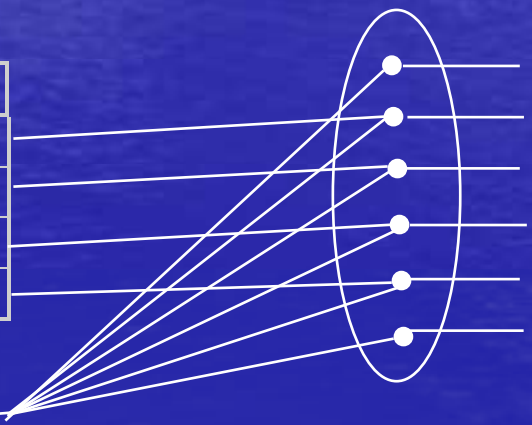
oid	DateNais	LieuNais	Start	Stop
2	12 05 75	strasbourg	12 05 75	forever

Estampillage temporel des données (4/6)

- Estampillage de toute la BD
  - gestion des estampilles plus simple
  - nécessite gestion de version par identification globale

oid	Nom	Salaire	Titre	Naissance
1	Toto	60000	assistant	< 2 >
1	Toto	70000	assistant	< 2 >
1	Toto	70000	MCF	< 2 >
1	Toto	70000	Prof	< 2 >

oid	DateNais	LieuNais
2	12 05 75	strasbourg



Start	Stop
12 05 75	31 09 00
01 01 01	31 09 06
01 10 06	31 09 07
01 10 07	31 09 08
01 10 08	01 01 10
02 01 10	forever

Estampillage temporel des données (5/6)

- Estampillage d'attributs
  - nécessite un modèle de données NF2 (non first normal form)
  - granularité la plus fine donc redondance minimum
  - gestion et requêtes plus compliquées

Nom	**Salaire**	**Titre**	**DateNais**
Toto	(60000, 01 01 01, 31 09 06), (70000, 01 10 06, 01 01 10)	(assistant, 01 01 01, 31 09 07), (MCF, 01 10 07, 31 09 08), (Prof, 01 10 08, 01 01 10)	12 05 75

Estampillage temporel des données (6/6)

- Estampillage par des instants
  - instant associé à un événement du monde réel
  - Simple mais ne représente que des événements discrets
- Estampillage par intervalle
  - début et fin d'un fait.
  - valeur spéciales: now, until-change, forever
  - redondance si même valeur pour des intervalles disjoints
- Estampillage par élément temporel
  - pas de redondance
  - beaucoup plus compliqué (parcours de liste)
- Autres (intervalles chevauchants...)

Nom	Visité	quand?
Toto	Londres	01 01 01
Toto	Berlin	01 10 06
Toto	Lisbonne	01 10 07

Nom	Salaire	Start	Stop
Toto	60000	01 01 01	31 09 06
Toto	70000	01 10 06	31 09 08
Toto	60000	01 10 08	now

Nom	Salaire	élément tempo.
Toto	60000	(01 01 01, 31 09 06) , (01 10 08, now)
Toto	70000	(01 10 06, 31 09 08)

Langage de requêtes temporels Logique temporelle (1/2)

- Logique du premier ordre / états + connecteurs temporels
  - considéré par rapport à un état courant
  - previous A (•A): A était vrai dans l'état précédent
  - next A (o A): A sera vrai dans l'état suivant
  - A since B : A a été vrai depuis que B a été vrai
  - A until B : A sera vrai jusqu'à ce que B sera vrai
  - on peut construire d'autres connecteurs
    - ¨A : true since A (A a été vrai au moins une fois dans le passé)
    - ◊ A : true until A (A sera vrai au moins une fois dans le futur)
    - ■A : ¬¨¬A (A a toujours été vrai dans le passé)
    - □A : ¬◊¬A (A sera toujours vrai dans le futur)

Langages de requêtes temporels
Logique temporelle (2/2)

- On suppose une base de données qui donne pour chaque année ***indep(pays,ville,S)*** si pour cette année là, notée S, le ***pays*** était indépendant et avait ***ville*** pour capitale
- En quelles années la Pologne était indépendante et pas la Slovaquie (les états pour lesquels la formule est vraie)?

  $(\exists Ville)\ (indep(\text{`Pologne'},Ville,S) \wedge \neg(\exists Ville2)indep(\text{`Slovaquie'},Ville2,S)\ )$
- Quelle ville a supplanté Cracovie à la tête de la

Langages de requêtes temporels
TSQL2 (1)

- Langage de requêtes pour le modèle relationnel bitemporel. Estampillage des tuples par éléments bitemporels
- Projet démarré en juillet 93 pour trouver un consensus après 15 ans et plus de 650 articles et 24 propositions de langages
- Un essai a été fait pour incorporer des parties de TSQL2 dans le standard SQL 1999 (SQL3) -> sous-standard SQL/Temporal, très critiqué et support abandonné fin 2001
- Fin 2011, dans SQL 2011 est ré-introduit la notion de temps (validité, transaction, bitemporel ainsi qu'un certain nombre d'opérateurs)

Langages de requêtes temporels TSQL2 (2)

- Création d'une table temporelle en ajoutant en fin de create table :
  - –as valid <granularité> — table historique
  - –as transaction — table rollback
  - –as valid <granularité> and transaction — table bitemporelle

create table Magasins(NomMag, budget, responsable) as valid state day

create table Employés(NomEmp, mag, salaire) as valid state day

create table Ventes(codeMag, ventes) as valid event day

(«state» : ensembles d'instants, «event» : instant)

Langages de requêtes temporels TSQL2 (3)

Problèmes : Le nom peut changer avec le temps. Échelle de temps homogène.

create table Magasin (NomMag, Budget, Resp_ID) as valid state

create table Employés (ID surrogate, Nom, Mag, Salaire) as valid state

NomMag	Budget	Resp_ID	Validité
Jouets	150	DI	{[1/1/02 – 7/31/04]}
Jouets	200	DI	{[8/1/04 – 12/31/06]}
Jouets	100	DI	{[1/1/07 –*now*]}
Livres	150	ED	{[4/1/07 – *now*]}

Now = 1/1/10

ID	Nom	NomMag	Salaire	Validité
ED	Ed	Jouets	20	{[2/1/02 – 5/31/02]}
ED	Ed	Jouets	30	{[6/1/02 – 1/31/05]}
ED	Ed	Jouets	40	{[2/1/05 – 1/31/07]}
ED	Ed	Livres	40	{[4/1/07 – 12/31/07]}
ED	Edward	Livres	40	{[1/1/08 – *now*]}
DI	Di	Jouets	30	{[1/1/02 – 7/31/04]}
DI	Di	Jouets	40	{[8/1/04 – 8/31/06]}
DI	Di	Jouets	50	{[9/1/06 – *now*]}

Langages de requêtes temporels TSQL2 (4) : requêtes « snapshot »

- «Qui a travaillé au magasin de jouets le 2 mars 03 ?»

*select **snapshot** Nom from Employés as e*
*where NomMag = 'Jouets' and **valid**(e)*
***contains date** '3/2/03'*

  - snapshot : renvoie un résultat non historisé
  - valid(e) : retourne l'intervalle de temps correspondant à e

Langages de requêtes temporels TSQL2 (5) : requêtes « snapshot »

- «Quels employés ont eu le même salaire pendant une période continue > 3ans ?»

select snapshot e.Nom
*from Employés(ID, Salaire) (**period**) as e*
*where **cast**(valid(e) **as interval** year) >= interval '3' year*

  - period : force la coalescence
  - cast(x as …) : transformation
  - interval : génère une durée
  - Résultat : {('Ed'), ('Di'),('Edward ')}

Langages de requêtes temporels TSQL2 (6) : requêtes « time-varying »

«Quelles sont les périodes au magasin de jouets pour des employés qui sont resté plus de 8 ans ?»

select valid(e) from Employés(ID) (period) as e
where NomMag = 'Jouets' and cast(valid(e) as interval year) > interval '8' year

– Résultat : { ([1/1/02 - *now*]) }

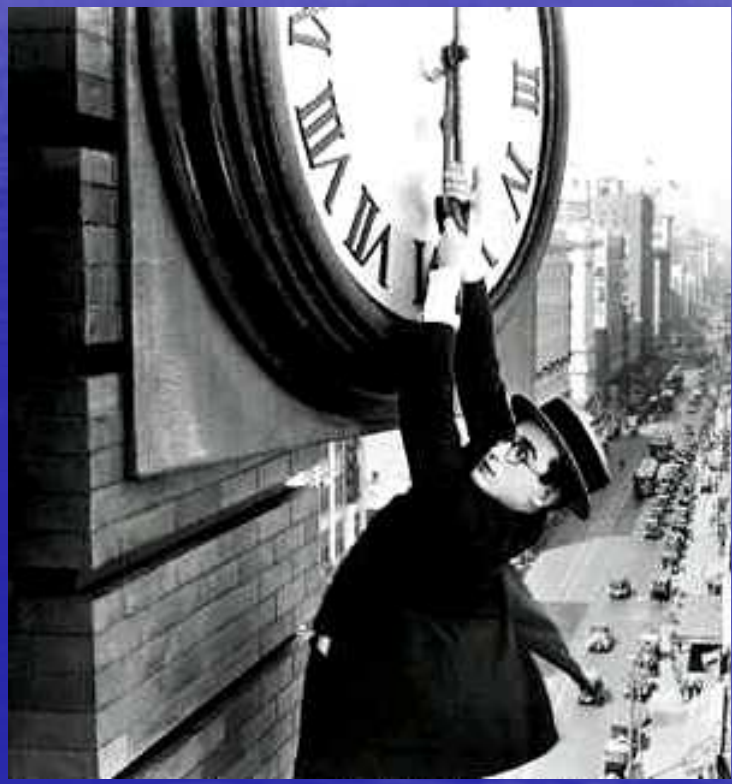
Langages de requêtes temporels TSQL2 (7) : requêtes « time-varying »

- «A quelles dates a-t-on réembauché quelqu'un après plus d'un mois ?»

```
select begin( valid(e2) ) from Employés(ID) (period)
   as e1, e2
   where (e1.id = e2.id) and valid(e1) precedes
   valid(e2)
   and ( end( valid (e2)) – begin(valid( e1 ))  month )
   > interval '1' month
```

Quels SGBD pour gérer le temps?

- ORACLE: Oracle Workspace Manager, permet de gérer le temps de validité et les versions (basé sur SQL:2011)
- PostgreSQL: un package (opensource) développé pour supporter TSQL2
- IBM DB2 (10): validité, transaction, bitemporel, etc (basé sur SQL:2011)
- Teradata (basé sur TSQL2)



Exemple: PostgreSQL (1/4)

2 types temporels: *timestamptz* et *period*

Dizaines de fonctions:

- D'estampilles:
  - timestamptz first(period p)
  - timestamptz next(period p), …
- Booléennes:
  - boolean contains(period p, timestamptz ts)
  - boolean overlaps(period p1, period p2)
  - boolean before(period p1, period p2) …

Exemple: PostgreSQL (2/4)

- Période:
  - period period(timestamptz ts1, timestamptz ts2)
  - period period_intersect(period p1, period p2)
  - period minus(period p1, period p2) …
- Opérateurs:
  - Toutes ces fonctions ont des opérateurs associés utilisables dans les requêtes sql: +, @>, &&, <<, etc

Exemple: PostgreSQL (3/4)

```
CREATE TABLE meeting (
    name TEXT PRIMARY KEY,
    location TEXT,
    during PERIOD );

-- create a special for fast lookups
CREATE INDEX meet_during_idx ON meeting USING gist (during);

 INSERT INTO meeting VALUES('Project Planning', 'Room 173',
    period('2008-01-01 14:00:00', '2008-01-01 16:00:00'));
INSERT INTO meeting VALUES('Yearly Budget', 'Room 212',
    period('2008-01-01 10:00:00', '2008-01-01 12:00:00'));
INSERT INTO meeting VALUES('Code Review', 'Room 212',
    period('2008-01-01 9:00:00', '2008-01-01 11:00:00'));
```

Exemple: PostgreSQL (4/4)

```
-- Find conflicting events
SELECT m1.name, m2.name
FROM meeting m1, meeting m2
WHERE m1.name < m2.name AND
   overlaps(m1.during, m2.during);

-- Where am I supposed to be right now?
SELECT location FROM meeting
WHERE during @> now();
```

Merci de votre attention!