Bases de données et carte à puce

Luc Bouganim
Luc.Bouganim@prism.uvsq.fr

Plan du cours

- **Applications bases de données**
- **PicoDBMS : Un système de base de données embarqué sur carte à puce**
- **Chip-Secured Data Access :Une solution logicielle/matérielle pour sécuriser des bases de données**

Applications traditionnelles des SGBD

- **OLTP** (On Line Transaction Processing)
  - Cible des SGBD depuis leur existence
  - Banques, réservation en ligne ...
  - **Très grand nombre de transactions** en parallèle
  - Transactions **simples**
- **OLAP** (On Line Analytical Processing)
  - Entrepôts de données, DataCube, Data Mining …
  - **Faible nombre de transactions**
  - Transactions très **complexes**

Applications émergentes des SGBD (1)

- **BD et WEB**
  - Serveurs Web dynamiques, sites marchands ...
  - Plusieurs profils (OLTP, publication d'informations en ligne, hébergement de données …)
- Challenges majeurs
  - Gestion de données XML
  - Fédération de sources de données hétérogènes
  - Grilles de données
  - **Sécurité des données en ligne**

Applications émergentes des SGBD (2)

- **BD personnelles ou PME**
  - Comptabilité
  - Agenda, comptes bancaires, carnet d'adresses, dossiers portables
  - BD embarquées sur calculateurs ultra-légers (PDA, téléphones cellulaires, **cartes à puce** …)
- Challenges majeurs
  - Gérer la mobilité
  - S'adapter aux **contraintes matérielles** du calculateur hôte
  - Assurer la **durabilité** des données
  - Assurer la **confidentialité** des données

Conclusion

- Les SGBD rendent de grands services
  - Centralisent les informations et les gèrent de façon cohérente, sûre et efficace
- Mais … ils sont une menace pour la confidentialité des données
  - Le regroupement de données non sensibles peut devenir très sensible (traquage des habitudes …)
- Intérêt des techniques de sécurisation des BD
  - **BD embarquées sur cartes à puce**
  - **Sécurité des BD en ligne**

PicoDBMS: Scaling down Database Techniques for the Smartcard

C. Bobineau, L. Bouganim, P. Pucheral, P. Valduriez

Université de Versailles & INRIA Rocquencourt

Historique de la carte à puce

- 70 : Premiers brevets [Moreno 74], [Bull CP8 77]
- 80 : Le système bancaire français adopte la carte à puce
- 90 : Une carte SIM dans chaque téléphone GSM
- 1997 : Les cartes deviennent multi-applications (JavaCard)
- 1999 : Smart Card Query Language (SCQL) [ISO 7816-7], Premier produit BD (SQL Java machine, ISOL), ➔ limités aux sélections monotables
- 2000 : PicoDBMS, un SGBD sur carte à puce !

SCQL et SQL Java Machine

- SCQL : SmartCard Query Langage :
  - Elaborée suite aux travaux du Laboratoire d'Informatique Fondamentale de Lille (**LIFL**)
- Selections monotables : **accès à base de curseur**
  - Similaire à des **fichiers** : open, next, close
  - Vues restreintes à des sélections/projections **monotables**
  - Droits **restreints** de la même manière
  - Possibilité **d'exécution atomique**
- SQL Java Machine (ISOL) : Implémentation JavaCard présentant les mêmes caractéristiques que SCQL
- Adaptés aux contraintes de l'époque : Prototype de **8Ko**

Problèmes de ces approches

- SCQL et ISOL transposent la problématique des BD au systèmes d'informations embarqués sur la carte
  - Cohérence de l'information
  - Partage entre plusieurs applications
  - «Facilité» d 'interrogation et d'administration
  - Simplification du code des applications embarquées
- Mais …
  - Quels sont les données **réellement partagées** dans une carte?
  - Quels sont les **applications cibles** ?
  - Design lié aux contraintes **antérieures** des cartes (qq Ko)
  - Requêtes simplistes => **droits d 'accès simplistes**

Pourquoi un SGBD sur carte ?

- BD sur carte ≠ SGBD sur carte
- Les SGBD lights existent déjà (Oracle-Lite, DB2 EveryWhere …) pour PDA et assimilés

	Portabilité	Sécurité	Capacité
Stockage portable	≈	✗	↘
PDA, Téléphone portable	≈	✗	↘
Carte à puce	↘	↘	≈

- La **sécurité** distingue les cartes à puce des autres calculateurs légers

Applications BD sur carte à puce ?

- La sécurité de la carte permet d'assurer une confidentialité maximale
- Le besoin d'un PicoDBMS existe si
  - Les données sont **confidentielles**
    - des données banales groupées deviennent confidentielles !
  - **Plusieurs acteurs** agissent sur ces données
    - Personnes ou logiciels
- PicoDBMS => droits d 'accès sophistiqués
- Droits d'accés sophistiqués => requêtes sophistiquées

Exemples d ’applications cibles

- BD téléchargeables (diplomates, militaires, commerciaux ...)
- **Environnement utilisateur** (bookmarks, configuration, agenda …)
- **Dossier personnel** (médical, scolaire, voiture, …)

- Besoins de ces applications:
  - requêtes complexes (jointures, agrégats, …)
  - droits d’accès et vues
  - transactions

L’exemple de la carte santé

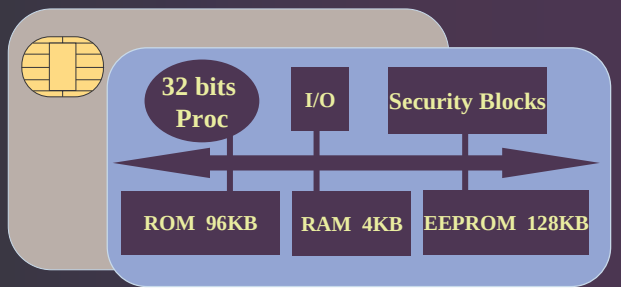
- Plusieurs pays ont lancé un programme de carte santé
- Volume de données «**pico-important**»
  - administratives: identification, assurance ...
  - urgence: groupe sanguin, allergies …
  - historique: prescriptions, opérations chirurgicales ...
- Requêtes «**pico-complexes**»
  - nombre d’antibiotiques prescrits au patient durant les trois derniers mois
- **Droits d’accès et vues**
  - Multi-utilisateur: porteur, médecins, organismes d’assurance …
  - La confidentialité du dossier médical est un problème majeur

L’environnement Virtuel

- Objectif : stocker dans la puce l’environnement du porteur :
  - Licences, infos de configurations, …
  - Bookmarks, cookies, mots de passes, souscriptions, …
  - Historique Web, statistiques, liens, préférences, …
- Volume de données «**pico-important**»
- Droits d’accès et vues
  - Multi-utilisateur: le porteur, **les différents logiciels**
  - Informations **hautement confidentielles (très personnelles)**
- Requêtes «pico-complexes» : **parcours d’arbres**

Contraintes liées à la carte à puce

- Sécurité
  - cryptographie → processeur RISC très puissant
  - puce minuscule (< 25mm^2) → capacité mémoire limitée
- Mémoire persistante = EEPROM
  - lectures rapides (≈ 60ns), écritures très lentes (> 1ms)
  - capacité en augmentation (1Mo bientôt ?)



Fonctionnalités du PicoDBMS

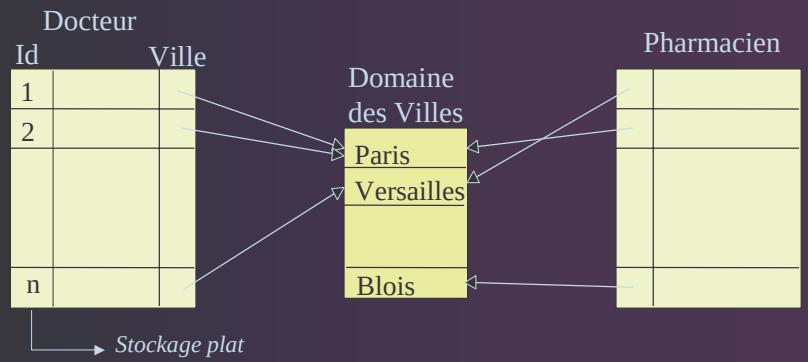
- **Impact de la sécurité**
  - la carte à puce est l'unique îlot sécurisé
  - la confidentialité des données **nous force à embarquer**:
    - Stockage, **Evaluation de Requêtes** (SPJ + Agrégats), Droits d'accès et Transactions
  - les autres fonctionnalités sont assurées par le terminal (e.g., analyse syntaxique, tri)
- **Impact de la portabilité**
  - La Durabilité ne peut être assurée dans la carte
  - L'Isolation est inutile

Principes de conception

- Compactness Rule
  - **minimiser la taille des données**, des index et du code du PicoDBMS
- RAM Rule
  - minimiser la RAM pour maximiser l'EEPROM
    - ⟹ exécuter toute requête **sans consommation de RAM**
- Write Rule
  - minimiser les écritures dans la mémoire persistante
- Read Rule
  - exploiter l'accès direct offert par l'EEPROM

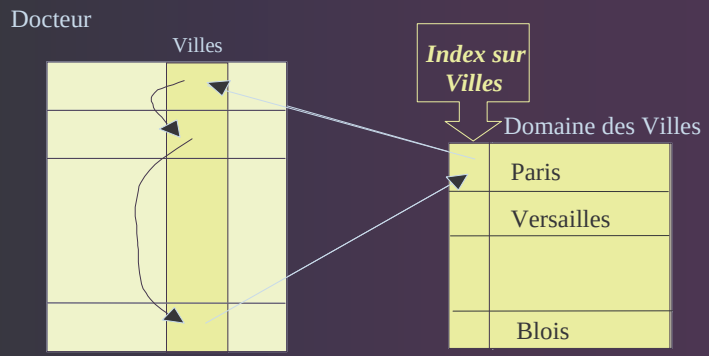
Modèle de stockage du PicoDBMS (1)

- Le PicoDBMS est un SGBD Mémoire
  - modèle de stockage basé sur les pointeurs
- Une combinaison de stockage **Plat** et en **Domaines**
  - Le stockage en domaines apporte la **compacité des données**



Modèle de stockage du PicoDBMS(2)

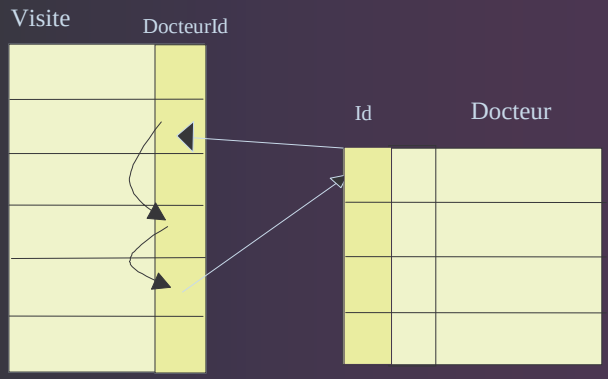
- Le Stockage Ring apporte la *compacité des index*
- Accélère la **Sélection** aux dépends de la **Projection**



Index Ring sur un attribut régulier

Modèle de stockage du PicoDBMS(3)

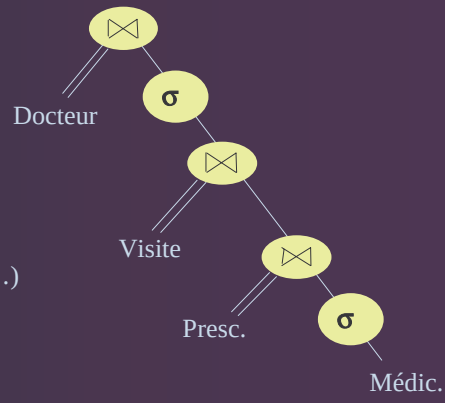
- Le Stockage Ring apporte la *compacité des index*
- Accélère la **Jointure** aux dépends de la **Projection**



Index Ring sur une clef étrangère

Exécution de requêtes (1)

- Ne **jamais matérialiser** de résultat intermédiaire
- Une nouvelle stratégie pipeline: *Arbre Droit Extrême*
- **Pas de consommation de RAM** pour SPJ

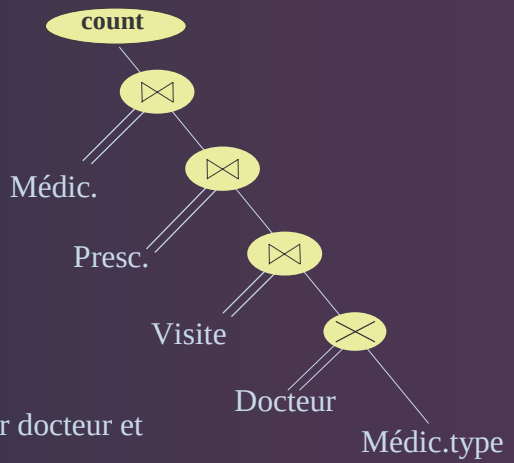


Docteur (IdDocteur, nom, spécialité …)
Visite (IdVisite, DocteurId, Date, Diagnostic …)
Prescription (IdVisite,IdMédicament, Qté …)
Médicament (IdMédicament, nom, type …)

Requête R1: Qui a prescrit des antibiotiques en 1999 ?

Exécution de requêtes(2)

- **Pas de consommation** de RAM non plus pour les requêtes complexes (group by, distinct …)



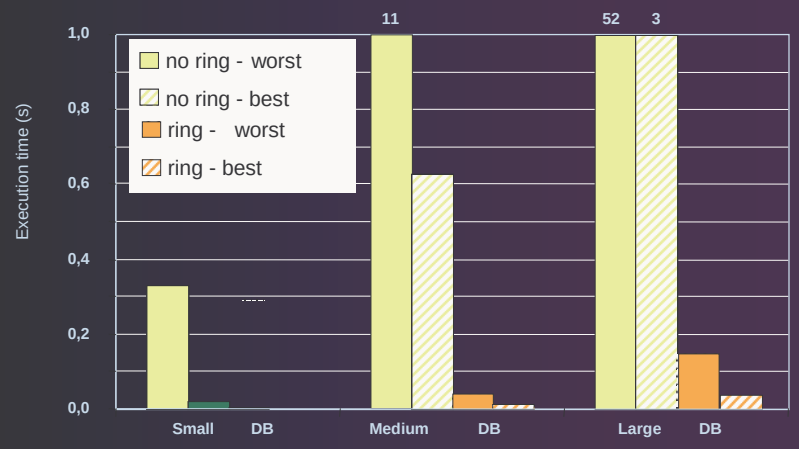
Requête R2:
Nombre de prescriptions par docteur et par type de médicament

Evaluation de performances

- Objectif : Evaluer si les performances du PicoDBMS sont **compatibles** avec les besoins des applications sur carte à puce
- Simulation d'un environnement carte à puce grâce à un ordinateur d'ancienne génération (PC 486 / 25 MHz)
- Bases de tests
  - *small : 10 doctors, 100 visits, 300 presc., 40 drugs*
  - *medium : 30 doctors, 500 visits, 2000 presc., 120 drugs*
  - *large : 50 doctors, 1000 visits, 5000 presc., 200 drugs*

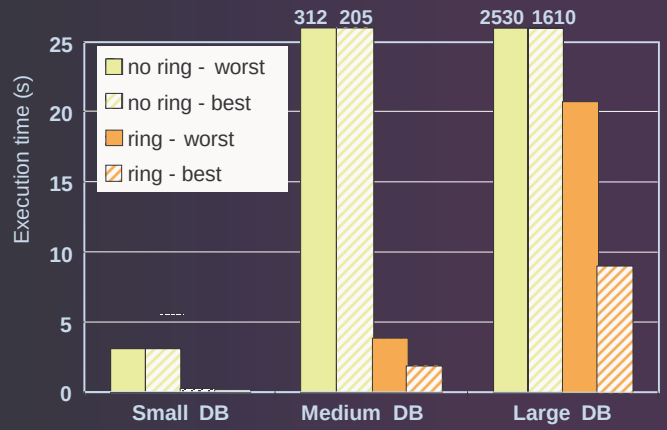
Evaluation de performances (2)

- les rings pour les jointures sont indispensables pour un volume important de données (large DB)
- l'optimiseur et les index de restriction (rings) ne se sont pas montrés nécessaires.



Evaluation de performances (3)

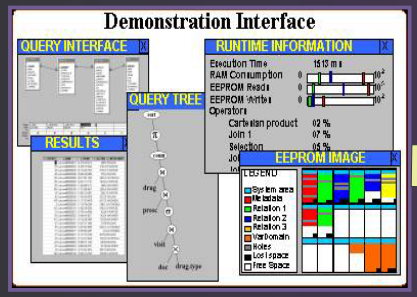
- Les résultats sur la requête R2 confirment ces résultats



Prototype PicoDBMS

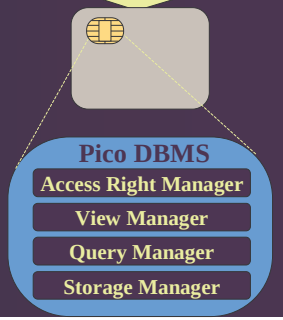
- Développé au PRiSM
- Fonctionnalités supportées
  - Création/destruction de tables
  - Gestion des utilisateurs
  - Gestion des **vues et des droits**
  - Interrogation : sélection, projection, **jointure, agrégats, group by**
  - Mises à jour : Insert, delete, update
- Ecrit en JavaCard 2.1 (footprint ≈ 35 Ko)
- Tourne sur un émulateur de cartes

Prototype (Démo. VLDB 2001)



JDBC Driver : Supported SQL

Create / Drop User
Create / Drop Table
Create / Drop View
Grant/Revoke
Update…set…where…
Delete from … where…
Insert into …values (…)
Select… from…where…group by…



Chip-Secured Data Access : une solution logicielle/matérielle pour sécuriser des bases de données

Luc Bouganim, Philippe Pucheral

Laboratoire PRiSM

Université de Versailles

Chip-Secured Data Access (C-SDA)

Sécuriser des bases de données externes par un logiciel embarqué dans un SOE

Plan

- Besoins et objectifs
- Applications visées
- Solutions existantes et leurs limites
- Solution proposée
  - Principe de la solution de base
  - Extension à tout type de requêtes
  - Performance des exécutions
  - Robustesse du chiffrement

Besoins : 2 facteurs

- Démocratisation des SGBD
  - grands comptes, banques de données, PME, données personnelles ...
  - toute base de données est par essence confidentielle !
    - Données nombreuses, organisées et "complètes"
- Mise en ligne des données sur l'Internet
  - sites marchands
  - hébergement de données personnelles ou professionnelles
  - accès distants à des serveurs d'entreprise par des utilisateurs mobiles
  - interconnexion de banques de données scientifiques, techniques, médicales ...

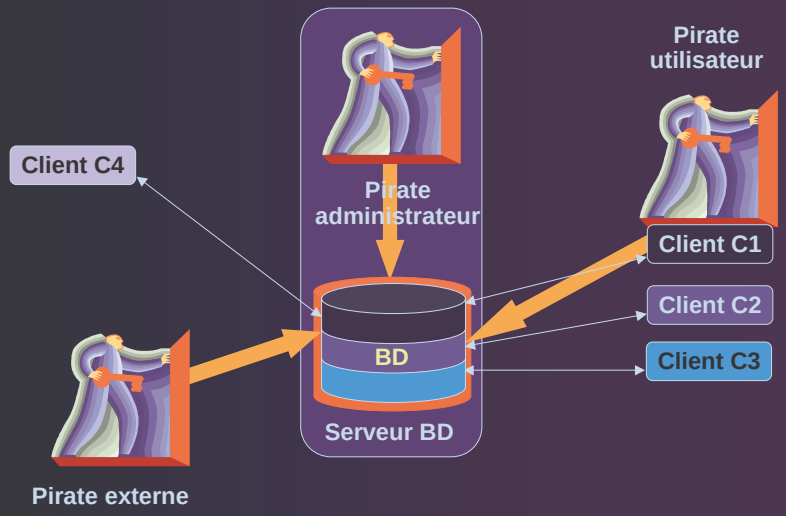
Le problème de la confidentialité : motivation des pirates

- Fraude (n° de cartes bancaires …)
- Atteinte à la vie privée des individus (enquêtes fiscales, financières, médicales, policières, politiques …)
- Guerre commerciale (violation de secrets industriels ou de stratégies commerciales)
- Guerre diplomatique ou militaire
- ...

Le problème de la confidentialité : quelques chiffres

- Rapport « Computer Crime and Security Survey » (Computer Security Institute et FBI)
- Coût des attaques aux USA (1998) : 137 milliards de US$
  - 37 % de plus qu'en 1997
  - dont vulnérabilité des BD : 103 milliards de US$
- 44% des attaques sont internes
  - 74% des pertes de propriété intellectuelle aux USA.

Le problème de la confidentialité : types d'attaques



Le problème de la confidentialité : types d'attaques

- Pirate externe
  - capable de s'infiltrer sur le serveur BD et de lire ses fichiers
  - capable de casser une clé de chiffrement avec un texte connu
- Pirate utilisateur
  - est reconnu par le SGBD et a accès à une partie des données
  - suivant le mode de chiffrement, il a accès à certaines clés
- Pirate administrateur (DBA)
  - employé peu scrupuleux ou pirate s'étant octroyé ces droits
  - a accès à des données inaccessibles aux autres pirates (journal)
  - peut espionner le SGBD pendant l'exécution

Objectifs

- Garantir la confidentialité des données stockées vis à vis de pirates externes, utilisateurs, et surtout administrateurs!
- Offrir un accès sécurisé en interrogation / modification à partir de tout terminal
- Permettre un partage strictement contrôlé entre de multiples utilisateurs

Contraintes technologiques

- Les données sont gérées par un SGBD traditionnel
- Support matériel de technologie courante
  e.g., une carte à puce conventionnelle
- Algorithmes de chiffrement connus

La technologie peut être mise en œuvre aujourd'hui, pour des systèmes d'information existants

Applications visées

Applications Personnelles

- Dossier personnel
  - Dossier médical,
  - Données bancaires,
  - …
- Virtual home
  - Agenda, carnet d'adresses
  - Mots de passes, bookmarks, cookies
  - courrier électronique
  - ...

- Données accessibles de n'importe où et à tout moment => données hébergées
- Hautement confidentielles
- Partagées par plusieurs acteurs
- Interrogations 'complexes' : like, >, <, agrégats …
- faible volume de données

Applications PME

- Données comptables
- Procédés de fabrication
- Dossiers clients
- Données bancaires (CB)

- Hébergement des données motivé par la tolérance aux pannes et aux virus et par des accès itinérants
- mêmes besoins que appli. perso.
- grand volume de données

Applications BtoB

- Échanges donneur d'ordre/ fournisseur sur Internet
- Données gérées par des intermédiaires
- Données d'équipementiers hébergées par un constructeur

- Mêmes besoins que préc.
- Droits très sélectifs
- Interrogations très complexes

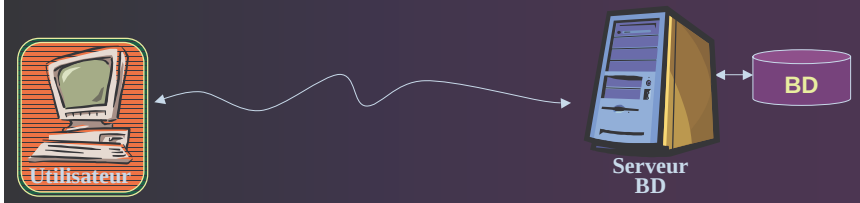
Applications scientifiques

- Procédés en cours de brevetabilité (ex: Anvar)
- Banques de données scientifiques (ex: génome)

- Mêmes besoins que préc.
- Traçabilité des accès

Techniques existantes

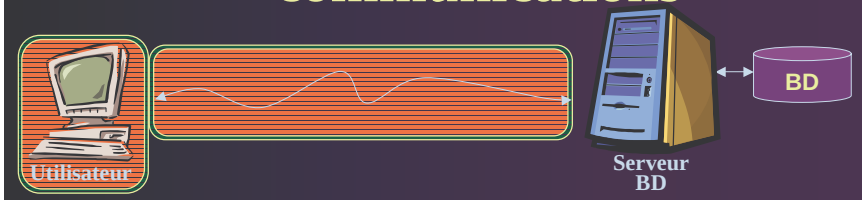
T1 : Identification/authentification



- Base : login + password
- Nombreux protocoles et systèmes matériels (carte à puce!)

➔ Nécessaire mais clairement insuffisant !!

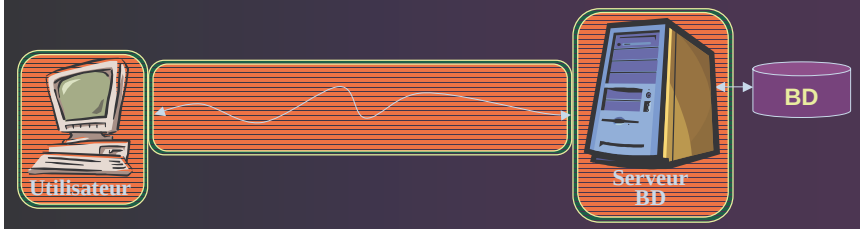
T2 : Chiffrement des communications



- Technologie éprouvée (y compris avec des cartes à puce)
- Identification/Authentification du client, confidentialité et intégrité des messages, non répudiation des transactions

➔ Nécessaire mais clairement insuffisant !!

T3 : Mécanismes de droits BD



- Affectés à des utilisateurs (ou groupes)
- Portent sur des tables, vues ou procédures stockées
- Très sophistiqués

- Ne résiste pas à une attaque sur les fichiers du serveur ou à une attaque du DBA !

➔ Nécessaire mais insuffisant !!

T3 : Rappel sur les vues et droits (1)

Vue «annuaire»

Id-E	Nom	Prénom	Poste
1	Ricks	Jim	5485
2	Trock	Jack	1254
3	Lerich	Zoe	5489
4	Doe	Joe	4049

Vue «stats»

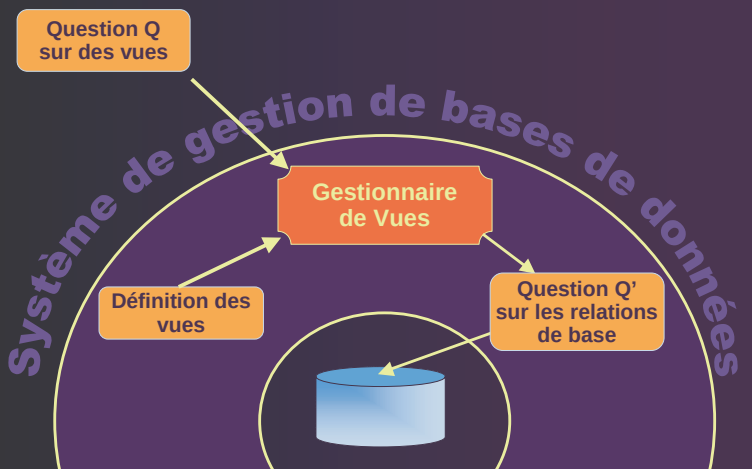
Nombre d'employés	Masse Salariale
4	890

Table : «employés»

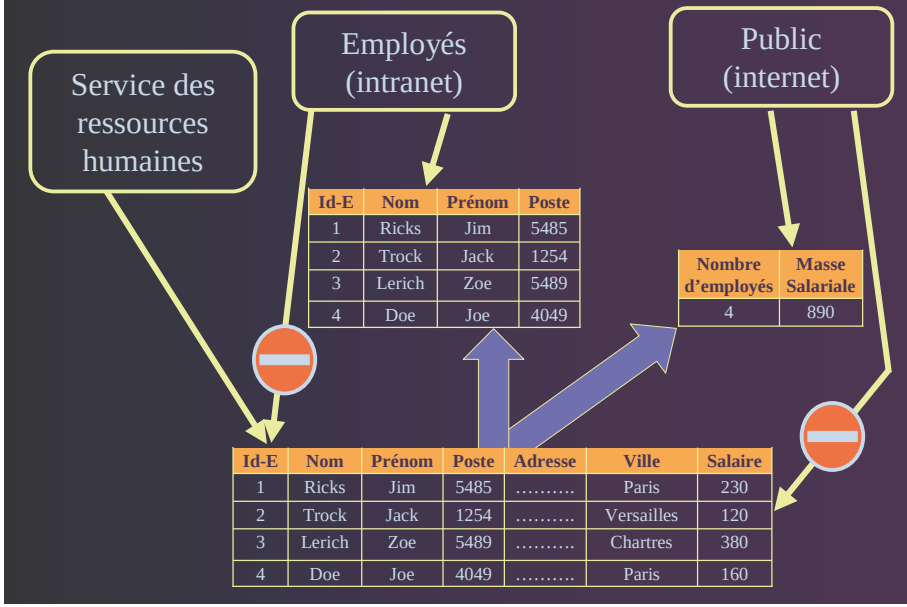
Id-E	Nom	Prénom	Poste	Adresse	Ville	Salaire
1	Ricks	Jim	5485	…….	Paris	230
2	Trock	Jack	1254	…….	Versailles	120
3	Lerich	Zoe	5489	…….	Chartres	380
4	Doe	Joe	4049	…….	Paris	160

T3 : Rappel sur les vues et droits (2)

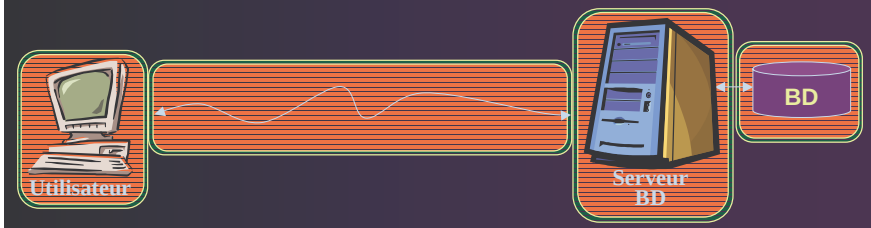
Le SGBD **transforme** la question sur les vues en question sur les relations de base



T3 : Rappel sur les vues et droits (3)



T4 : Chiffrement de la BD



- Principe : chiffrer l'empreinte disque de la BD
- Seule solution pour résister aux attaques sur les fichiers

➔ Oracle Obfuscation Toolkit

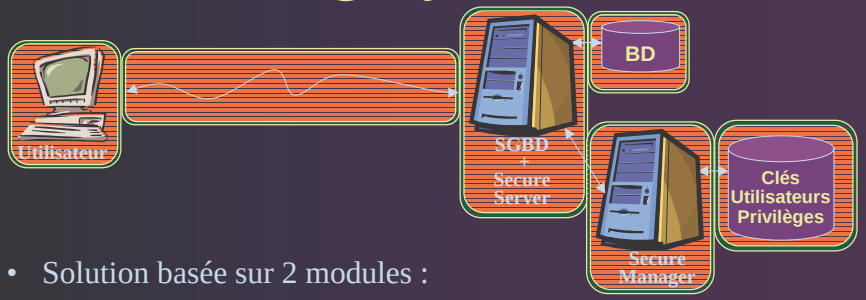
➔ Protegrity Secure.Data
  - Oracle, Informix, Sybase, SQL Server, IBM DB2

➔ Proposition d'IBM (recherche – même principe que Protegrity)

T4 : Oracle Obfuscation Toolkit

- Fourniture d'un « package » permettant le chiffrement / déchiffrement de données
- Problèmes :
  - Gestion et partage de clés à la charge de l'application
  - Système non résistant à un pirate administrateur
    - Les packages peuvent être substitués,
    - Les données apparaissent en clair lors de l'exécution des requêtes
- ORACLE : « DBA has all privileges » !!

T4 : Protegrity Secure.Data



- Solution basée sur 2 modules :
  - Secure.Manager : gestion des utilisateurs, droits et clés
  - Secure.Server : module de chiffrement intégré au noyau SGBD
- … et 2 personnes physiques différentes …
  - Database Administrator (DBA) / Security Administrator (SA)
- Isolation DBA/SA ?
- Données toujours en clair à un moment de l'exécution

Définition du problème

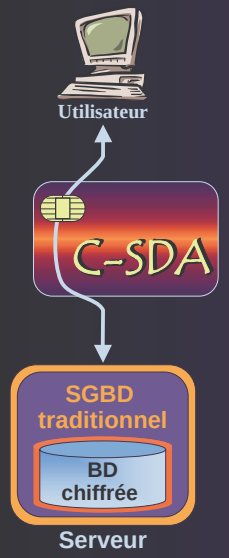
- La sécurité repose sur 2 points :
  - Un **logiciel de sécurité inattaquable** ➔ pas d'administrateur
  - Un **environnement d'exécution sûr**
    - **SOE** : Secure Operating Environment
    - Hardware / OS physiquement inattaquable
    - Exemple : carte à puce

Un SGBD n'est pas auto-administrable et ne s'exécute pas dans un SOE

Solution proposée :

C-SDA

Solution proposée : C-SDA



Chip-Secured Data Access

- chiffrement des données,
- gestion des requêtes,
- déchiffrement des résultats,
- gestion des droits BD des utilisateurs,

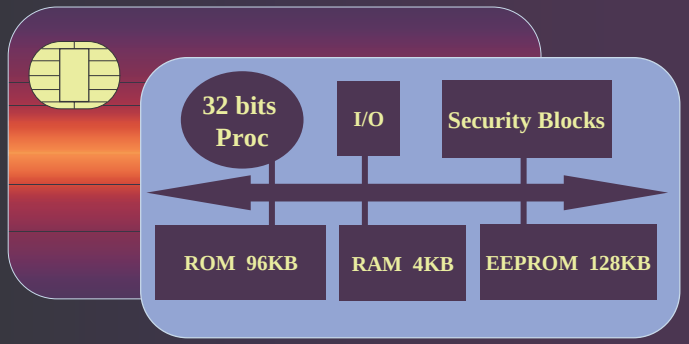
et

- auto-administrable,
- plateforme sécurisée (SOE).

Tout manquement à l'une de ces règles introduit un trou de sécurité

Caractéristiques de la carte

- Sécurité extrême des données et du code embarqué
- Processeur puissant adapté aux algorithmes de chiffrement
- RAM de très faible taille
- Taille croissante de la mémoire stable mais écritures très lentes



Chiffrement de la BD

- Données et méta-données chiffrées (➔ empreinte disque illisible)
- C-SDA intercepte toutes les commandes de création/modification/destruction de données
- Chiffrement partiel possible

produits

ref	nom	type	prix
d300	Dell	Pentium3	9800
I260	IBM	Pentium2	6400
...	...	...	...

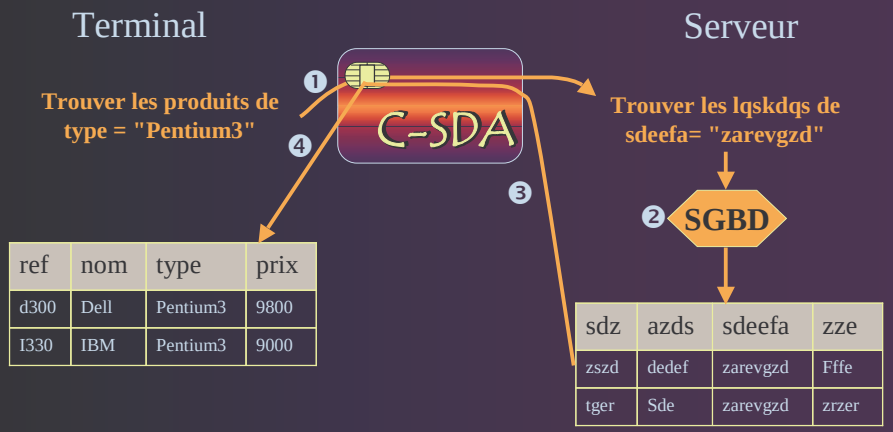
lqskdqs

sdz	azds	sdeefa	zze
zszd	dedef	zarevgzd	Fffe
df'g	Sde	iukèefsa	dgss
...	...	...	...

- Seuls les possesseurs de la clé peuvent accéder aux données
- Attention, chiffrement et gestion des droits ne doivent pas interférer

Traitement d'une requête

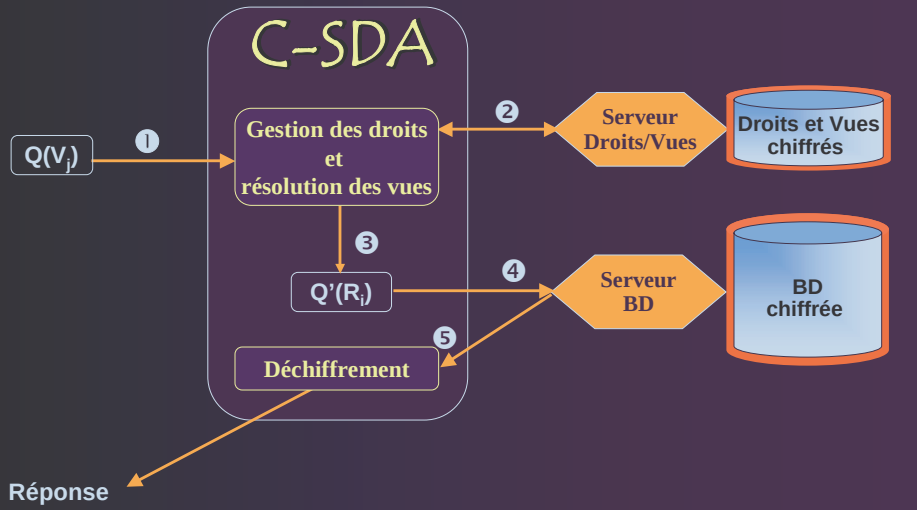
- C-SDA intercepte les requêtes de Select et les traduit



Gestion des droits

- Droits BD / droits systèmes
  - Droits BD : consulter, modifier, supprimer des données
  - Droits système : créer une partition, etc…
- Les droits BD font partie du noyau de sécurité …
  - **➔ Les droits BD doivent donc être vérifiés par C-SDA !**
- Les droits BD sont basés sur les vues …
  - **➔Les vues doivent donc être gérées par C-SDA !**
- Mais les droits et les vues sont des données partagées …
  - **➔droits et vues (définitions) doivent être stockés sur le serveur!**

Droits et vues



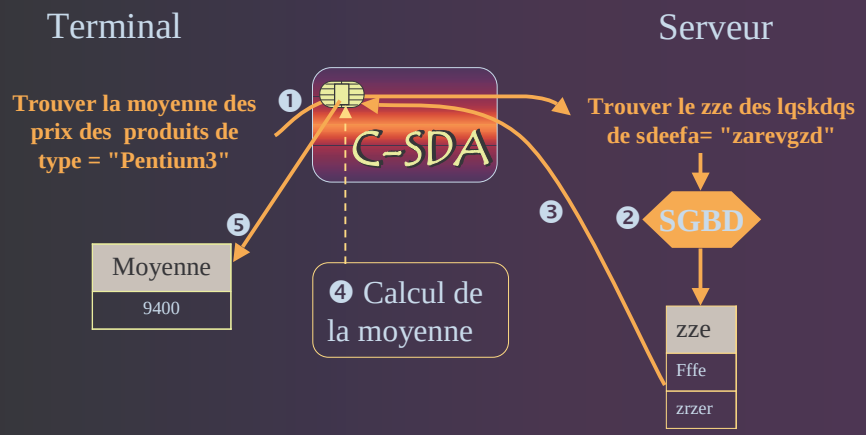
Bilan de la solution de base

- Avantages
  - L'accès est sécurisé à partir de tout terminal
  - Les données sont chiffrées, seules les cartes des propriétaires possèdent les clés
- Inconvénients
  - Requêtes limitées aux prédicats d'égalité
  - Droits limités de façon similaire

Solution étendue

- Gestion de tout type de requêtes
- Impact sur la performance
- Impact sur la robustesse du chiffrement

Supprimer les limitations sur les requêtes



La partie des requêtes non évaluable sur les données chiffrées est évaluée sur la carte (prédicats <, >, fonctions de calcul, etc..)

Requêtes : Sémantique de SQL (1)

- Exemple de requête SQL:
  - Pour les clients ayant fait plus de 10 commandes depuis 1996, somme des montants de ces commandes

```
SELECT      C.Code, C.nom, sum(Q.montant)
FROM        Client C, Commandes Q
WHERE       C.Code = Q.Code and Q.date > 1996
GROUP BY    C.Code, C.nom
HAVING      count(*) >= 10
ORDER BY    C.nom
```


Requêtes : Sémantique de SQL (2)

1. Produit cartésien des relations du FROM : Client X Commande
2. Prédicats du WHERE : C.code = Q.Code and Q.date > 1996
3. Groupement (GROUP BY) : C.Code, C.Nom
4. Calcul des agrégats : Count(*) , sum(Q.montant)
5. Prédicats du HAVING : Count(*) >= 10
6. Projection (SELECT) : C.Code, C.nom, sum(Q.montant)
7. Tri final (ORDER BY) : C.Nom

Requêtes : sur le serveur (Qs)

Possibilité de faire, sur les données chiffrées les traitements mettant en jeu des égalités (= ou ≠)

1. Produit cartésien des relations du FROM : Client X Commande
2. Prédicats du WHERE : C.code = Q.Code and Q.date > 1996
3. Groupement (GROUP BY) : C.Code, C.Nom
4. Calcul des agrégats : Count(*) , sum(Q.montant)
5. Prédicats du HAVING : Count(*) >= 10
6. Projection (SELECT) : C.Code, C.nom, sum(Q.montant)
7. Tri final (ORDER BY) : C.Nom

Requêtes : sur C-SDA (Qc)

Possibilité d'exécuter en PIPELINE le reste des opérations

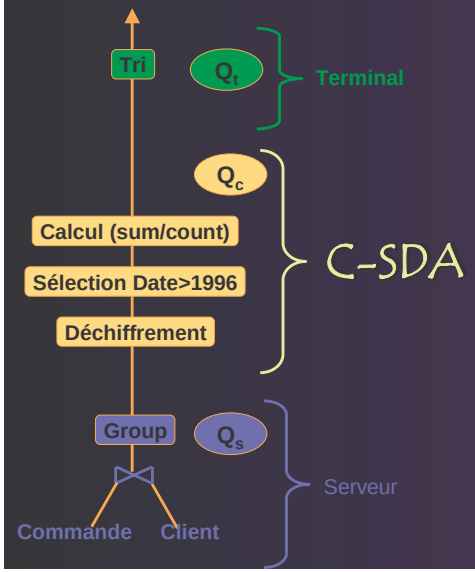
1. Produit cartésien des relations du FROM : Client X Commande
2. Prédicats du WHERE : C.code = Q.Code and Q.date > 1996
3. Groupement (GROUP BY) : C.Code, C.Nom
4. Calcul des agrégats : Count(*) , sum(Q.montant)
5. Prédicats du HAVING : Count(*) >= 10
6. Projection (SELECT) : C.Code, C.nom, sum(Q.montant)
7. Tri final (ORDER BY) : C.Nom

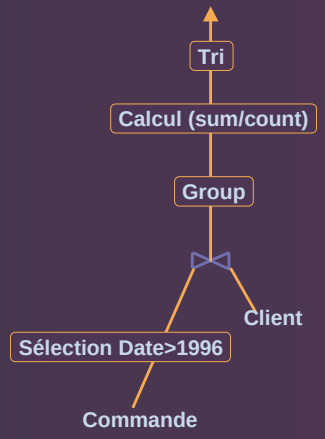
Requêtes : sur le terminal (Qt)

Tri final…

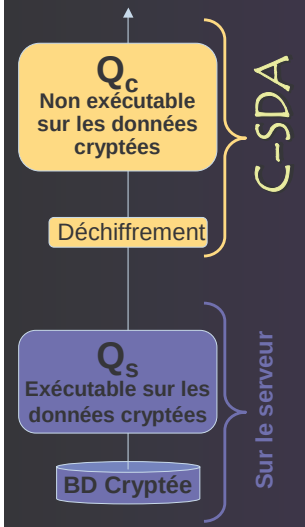
1. Produit cartésien des relations du FROM : Client X Commande
2. Prédicats du WHERE : C.code = Q.Code and Q.date > 1996
3. Groupement (GROUP BY) : C.Code, C.Nom
4. Calcul des agrégats : Count(*) , sum(Q.montant)
5. Prédicats du HAVING : Count(*) >= 10
6. Projection (SELECT) : C.Code, C.nom, sum(Q.montant)
7. Tri final (ORDER BY) : C.Nom

Requêtes : Arbres d'exécution





Requêtes : Bilan

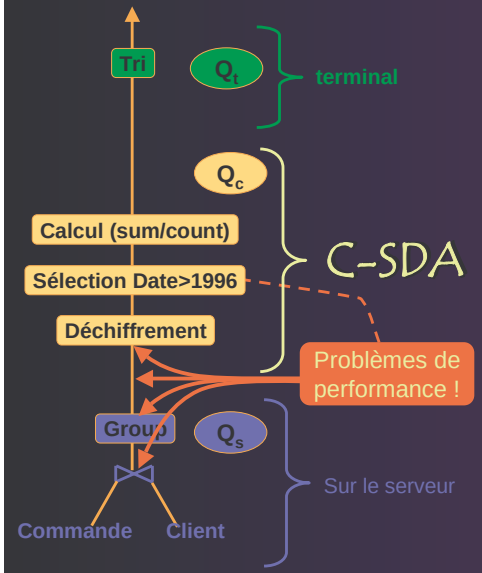


Contributions
- décomposition $Q = Q_t(Q_c(Q_s))$
- algorithme pipeline d'évaluation de Q_c

Avantages :
- plus de limitation pour les requêtes
- et donc plus de limitations pour les droits
- compatible avec les cartes actuelles

Inconvénient :
- Performance des requêtes complexes

Performance : les problèmes



- Prédicats d'inégalité
  - restrictions
  - inéqui-jointures
- Problèmes de performance
  - transfert réseau
  - exécution SGBD
  - exécution C-SDA

Performance : Solution

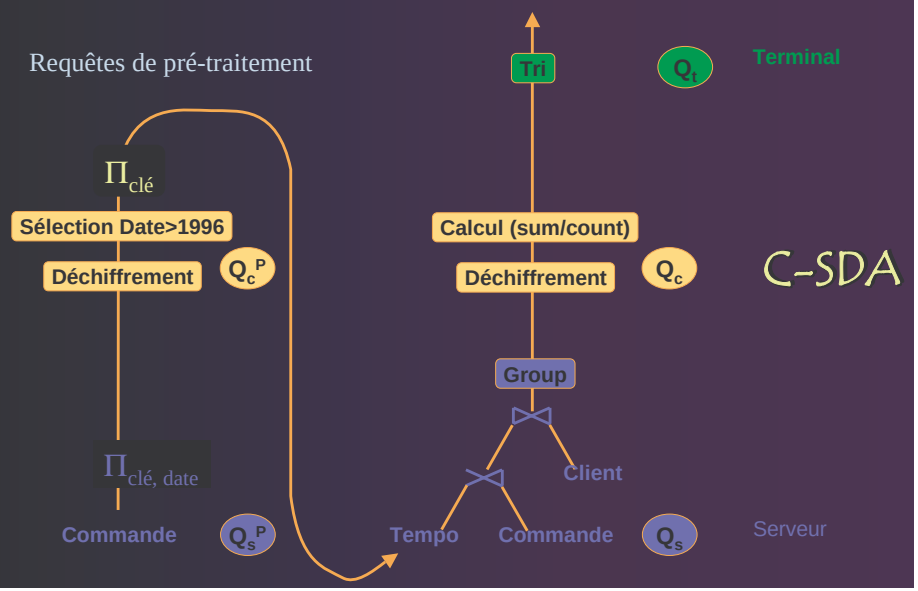
- Principe : appliquer les prédicats au + tôt

➔Utilisation de requêtes de prétraitements

- Prétraitement : Qc^P (Qs^P)
  - se répète pour chaque prédicat d'inégalité.
  - Qs^P renvoie le(s) attribut(s) à tester + 1 clé
  - Qc^P déchiffre le(s) attribut(s), évalue le prédicat et renvoie les clés satisfaisant le prédicat sous la forme d'une table temporaire
  - cette table temporaire est intégrée dans la requête initiale Q par un prédicat de semi-jointure

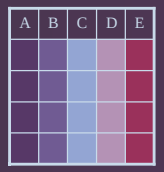
$$Q = Qt\ (Qc\ (Qs\ (*[Qc^P\ (Qs^P)])))$$

Performance : Exemple



Robustesse du chiffrement

- Idée : Utiliser plusieurs clés (ou algos) de chiffrement
- Contrainte : a = b ⇔ chiffre (a) =chiffre (b)
- Fragmentation verticale
  - utiliser des clés différentes pour des attributs non comparables



- Fragmentation horizontale
  - Plusieurs clés pour différentes valeurs du même attribut
    - Appliquer une fonction de hachage h sur la valeur a.
    - Utiliser la clé Key(h(a)) comme clé de chiffrement
    - Le couple (h(a), $Chiffre_{key(h(a))}(a)$) est stocké

'*Isoler*' les données les plus *sensibles*

- Isoler les données sensibles sur la carte pour les rendre définitivement inaccessibles au pirate
- La taille de ces données est limitée par l'EEPROM
  - Exemple : données d'identifications, …
  - Permet par exemple de dépersonnaliser une BD
- 3 problèmes
  - Complexité de l'évaluation de requêtes
  - Durabilité de ces données
  - Partage de ces données

Complexité de l'évaluation

- Isoler des données sans compliquer l'évaluation
  - Remplacer les données sensibles de la base par des indices dans un "domaine sensible" stocké sur la carte à puce
- Peut être vu comme un mode de chiffrement *incassable*
- la propriété a = b ⇔ chiffre (a) = chiffre (b) est préservée
  => même stratégie d'évaluation

N°	Nom	Code CB
4	Dupond	0454255782
8	Durand	0450500609
9	Dutronc	0456589413
13	Duval	0454547898
15	Dussol	0455121236

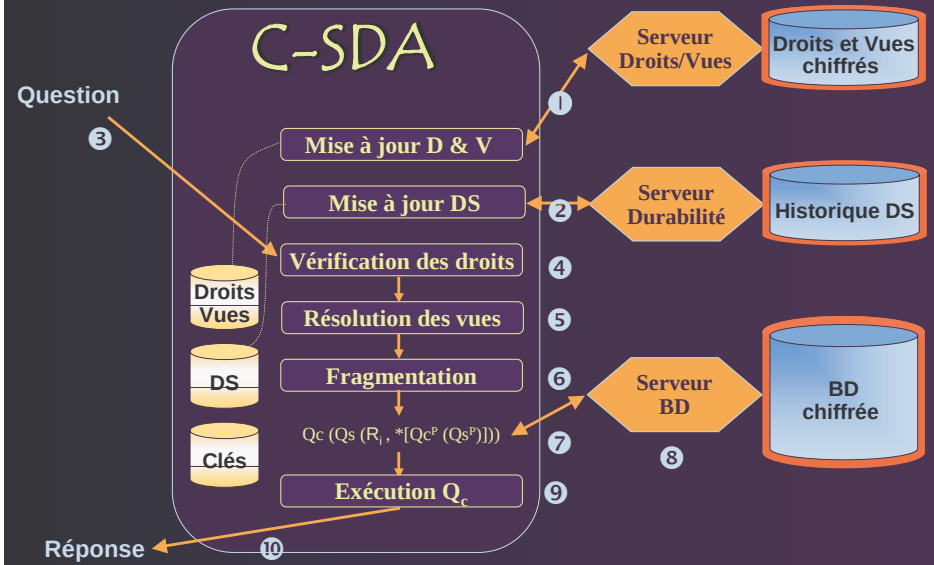
N°	Nom	Code CB
4	Dupond	1
8	Durand	2
9	Dutronc	3
13	Duval	4
15	Dussol	5

Code CB
0454255782
0450500609
0456589413
0454547898
0455121236

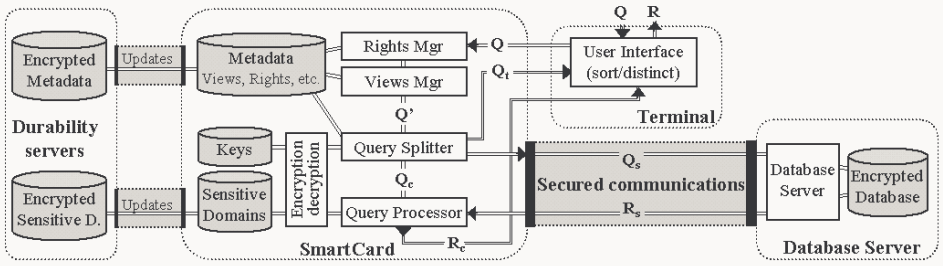
Durabilité et partage

- Données statiques : Stockage redondant
  - sur une *carte backup* utilisé en cas de perte, vol ou destruction de la carte C-SDA
- Données dynamiques : Stockage sur un serveur de durabilité
  - Si possible distinct du serveur de données
  - Enregistre l'historique des mises à jour
  - Pas de contrainte sur le chiffrement
  - Résoud aussi le problème du partage

Bilan



Architecture finale

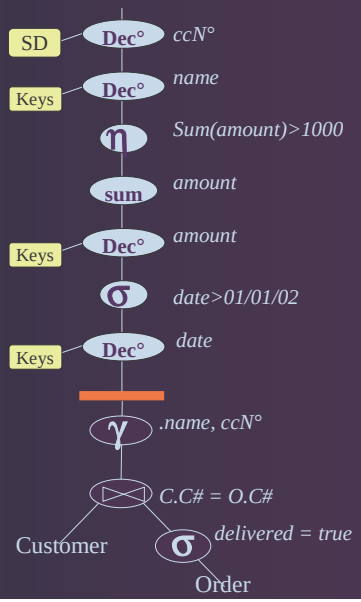



```
Select   name, ccN°, total
From     Invoice
Where    F.total>1000;
```
Q

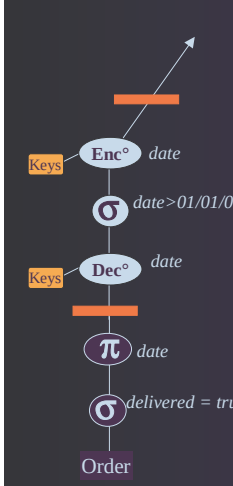
```
Select   name, ccN°, sum(amount) total
From     Customer C, Order O
Where    C.C# = O.C# and delivered = true
         and date>#01/01/02#
Groupby  C.name, C.ccN°
Having   sum(O.amount) >1000;
```
Q'

```
Select   E(name), E(ccN°),
         E(amount), E(date)
From     E(Customer) C, E(Order) O
Where    C.E(C#)=O.E(C#) and
         E(delivered)=E(true)
Orderby  E(name), E(ccN°)
```
Qs

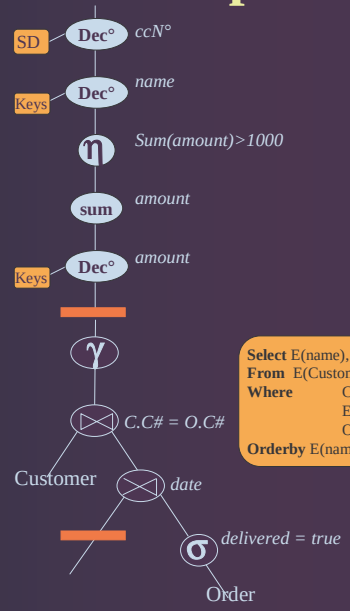
Exemple final



Exemple Final Optimisé



Select E(date)
From E(Order)
Where E(delivered)=E(true)



Select E(name), E(ccN°), E(amount)
From E(Customer) C, E(Order) O, Temp T
Where C.E(C#)=O.E(C#) **and** E(delivered)=E(true) **and** O.E(date)=T.E(date)
Orderby E(name), E(ccN°)

Conclusion

- Les atouts techniques
  - sécurité des données contre « tout » type d'attaque
  - « 100% » compatible serveurs BD actuels
  - « 100% » compatible cartes à puce actuelles
- Calendrier
  - prototype en cours (solution JavaCard / Oracle)
  - Brevet déposé en Août 2001

Conclusion générale

	BD d'entreprise	BD personnelles	BD 'light' (PDA / Tél.)	PicoDBMS carte à puce
Capacité				
Prix				
Nombre				