



SGBD et S.I. : Retour sur les applications

Mots-clés :

- Application persistante
- Architectures applicatives
- « Impedance mismatch »
- Schéma conceptuel / schéma relationnel
- S.I. et nouvelles applications bases de données

2



Résumons...

- **Systèmes de gestion de Bases de données**
 - Technologie incontournable pour les S.I.
 - Vise à
 - centraliser les données
 - Base de données : ensemble de données formant un ensemble logique cohérent
 - réduire le coût de maintenance du S.I.
 - Principe d'indépendance des données
 - Architecture ANSI
- Le **modèle relationnel** offre un cadre cohérent pour
 - Modéliser les données (redondance, abstraction et Performance)
 - Les interroger (efficacité, logique)
 - Les mettre à jour (transactions, cohérence, résistance aux pannes)
- Reste à développer des applications !

3



Évolution de l'informatique

- L'informatique ne se limite pas aux bases de données
 - Évolution des matériels et des réseaux
 - Terminal -> PC puissant -> souris -> interfaces évoluées -> ergonomie
 - Évolution des architectures
 - Généralisation du Client-serveur -> serveurs d'application
 - Évolution des standards
 - SQL standard pour les SGBD
 - XML standard pour l'EDI (Echange de données informatisées)
 - Évolution des méthodes
 - Unified Modeling Language
 - Évolution des métiers (Web)
 - Normalisation des interfaces (navigateur)
 - Déploiement -> maintenance, qualité de service, évolutivité, généricité

4



Plan de la présentation

- Introduction
- **SGBD et développement d'applications**
 - Interfaces programmatiques
 - Architectures de déploiement
 - Mapping langage / SGBD
- Méthodes de conception
- Conclusion du module

5



Interfaces programmatiques

- 1 application =
Gestion de données + traitement + interface
- Interaction SGBD / LP
- Principe : lecture / traitement / mise-à-jour
 - Le SGBD assure la persistance
 - Copie des données persistantes dans les variables du programme => transientes
 - Le programme décide des mises à jour à réaliser sur la base de données et les effectue.

Embedded SQL '86

```
EXEC SQL INCLUDE SQLCA;  
EXEC SQL BEGIN DECLARE SECTION;  
  char *dbaccess="bda10/bda@T:zazie:ol";  
  VARCHAR ename[50];  
  VARCHAR job[40];  
  VARCHAR dname[40];  
EXEC SQL END DECLARE SECTION;
```

```
char Buf[1024];
```

```
connection();
```

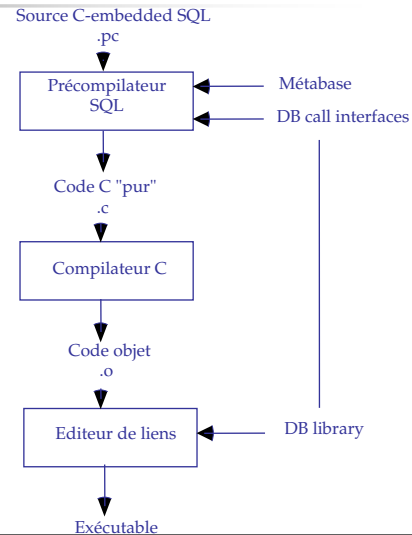
```
EXEC SQL DECLARE csa CURSOR FOR  
  SELECT e.ename, e.job, d.dname  
  FROM emp e, dept d,  
  WHERE e.deptno = d.deptno  
  ORDER BY d.dname, e.ename;
```

```
EXEC SQL OPEN csa ;  
val=0; /* compteur du nombre de tuples */  
while (1)  
{  
  EXEC SQL WHENEVER NOT FOUND DO  
    break;  
  EXEC SQL FETCH csa INTO  
    :ename, :job, :dname;  
  i=ename.len;  
  while(ename.arr[i-1]!=' ') i--;  
  ename.arr[i]='\0';  
  printf("%s # ", ename.arr);  
  i=job.len;  
  while(job.arr[i-1]!=' ') i--;  
  job.arr[i]='\0';  
  printf("%s # ", job.arr);  
  i=dname.len;  
  while(dname.arr[i-1]!=' ') i--;  
  dname.arr[i]='\0';  
  printf("%s \n ", dname.arr);  
  val++;  
}  
EXEC SQL CLOSE csa;  
printf("%d\n",val);  
  
deconnection();
```

7

SQL'86 – embedded SQL

- Embedded SQL
 - Le code source est normalisé
 - Fonctions de la librairie non normalisées
 - Précompilateur propriétaire
- Compilation en présence du SGBD



8

Une longue suite de standards

- Standard ANSI (1986), ISO (1987)
 - Embedded SQL, SQLCODE
- 1989 : SQL 89 (X/Open, SAG)
 - SQL dynamique
 - Dictionnaire de données
- 1992 : SQL2 (ISO & ANSI)
 - SQLSTATE
 - SQL procédural 95
 - SQL CLI 95
- 1999: SQL3 (ADT)

9



SQL Procédural

- SQL'99: Persistent stored module (SQL/PSM)
 - SQL-invoked routines
 - Regroupement de routines en modules
- SQL-invoked routines
 - Procédure, fonction, méthode
 - Ajout / SQL
 - Blocs de controles
 - If-then-else, boucles, exception, etc.
 - Déclaration et affectation de variables
 - Gestion de zones de diagnostics
 - Infos sur l'état du processus

10



Exemple de SQL procédural

- Création d'une procédure

```
create or replace procedure insert_emp(  
    ename      varchar2,  
    job        varchar2,  
    deptName   varchar2)  
is  
    num number;  
begin  
    select deptno into num from dept where dname=deptName;  
    insert into emp values  
        (num_pers_seq.nextval, ename, job, num);  
end;
```
- Appel de la procédure insert_emp

```
exec insert_emp ('Ward', 'SALESMAN', 'RESEARCH');
```

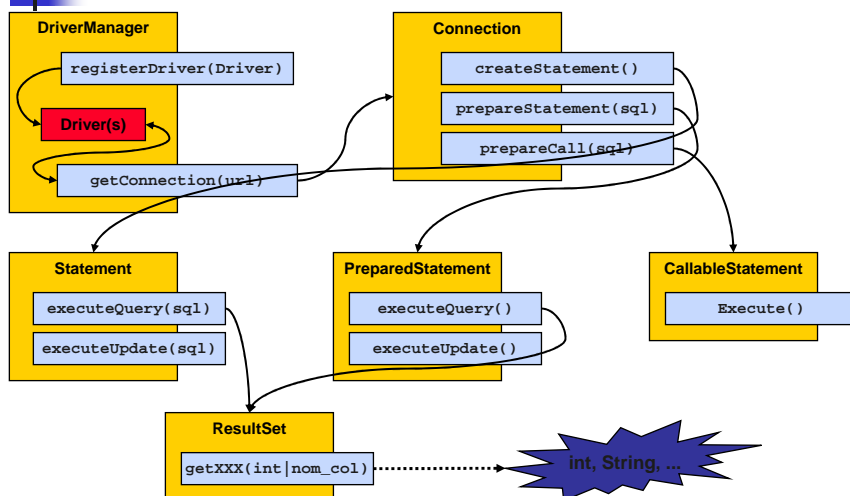
11

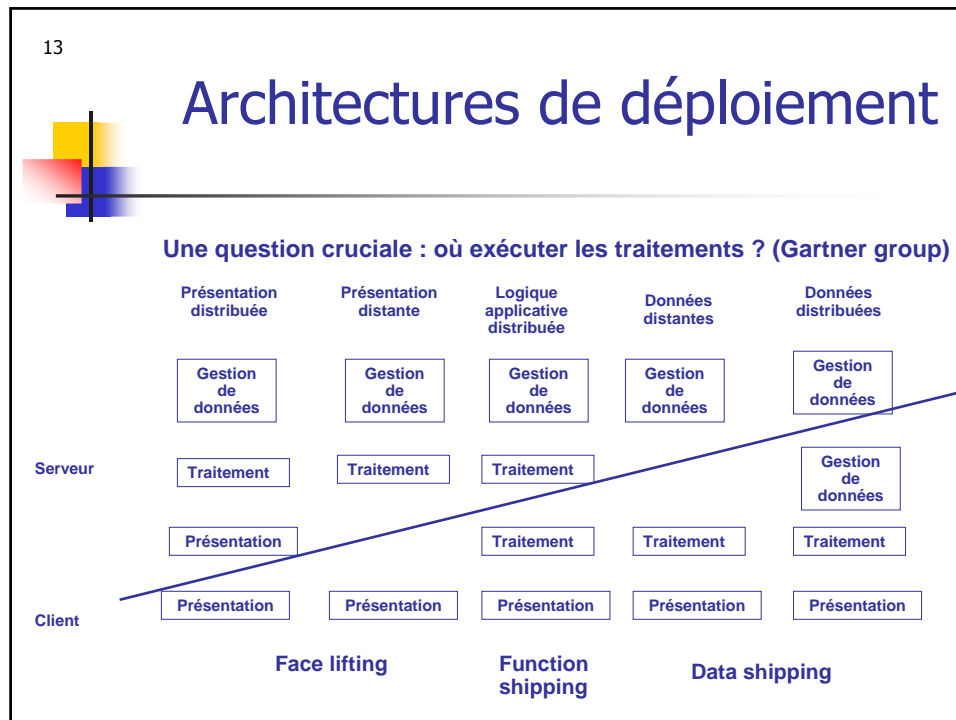
SQL'99 : SQL-Call Level interface (SQL/CLI)

- Spécification d'une bibliothèque de fonctions permettant l'insertion de commandes SQL comme paramètres
 - On standardise les prototypes de la bibliothèque du SGBD (édition de lien)
 - On évite la pré-compilation.
 - Le SGBD est sélectionné à l'exécution à partir d'un fichier de configuration
 - Chargement dynamique de la librairie du SGBD accédé
- Exemple : JDBC
 - on peut travailler avec plusieurs SGBD à la fois

12

JDBC : résumé



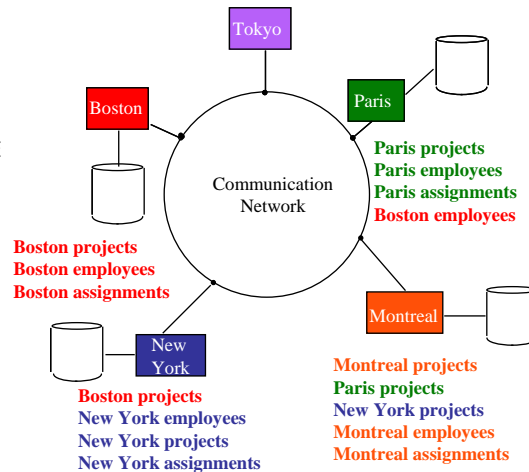


- 14
- ## Une architecture exemple MySQL - PHP
- Intérêt du TP
 - Application facile à développer
 - PHP : Langage simple (typage dynamique, etc)
 - Accès à la BD
 - API SQL
 - Architecture C/S + Application « distante »
 - Imbrication avec HTML/ protocole HTTP
 - Révélateur
 - Point fort : architecture
 - Point faible : MySQL n'implante pas toutes les fonctionnalités SGBD

15

SGBD distribué

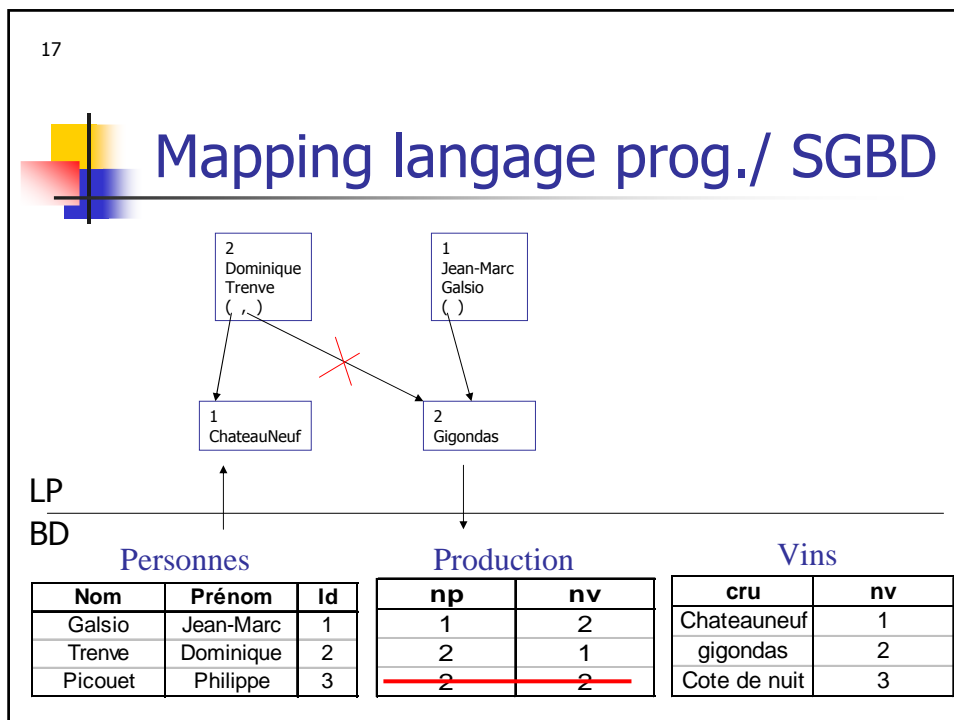
```
Select ENAME, SAL  
FROM E, A, P  
WHERE DUR > 12  
AND A.ENO = E.ENO  
AND A.TITLE = P.TITLE
```



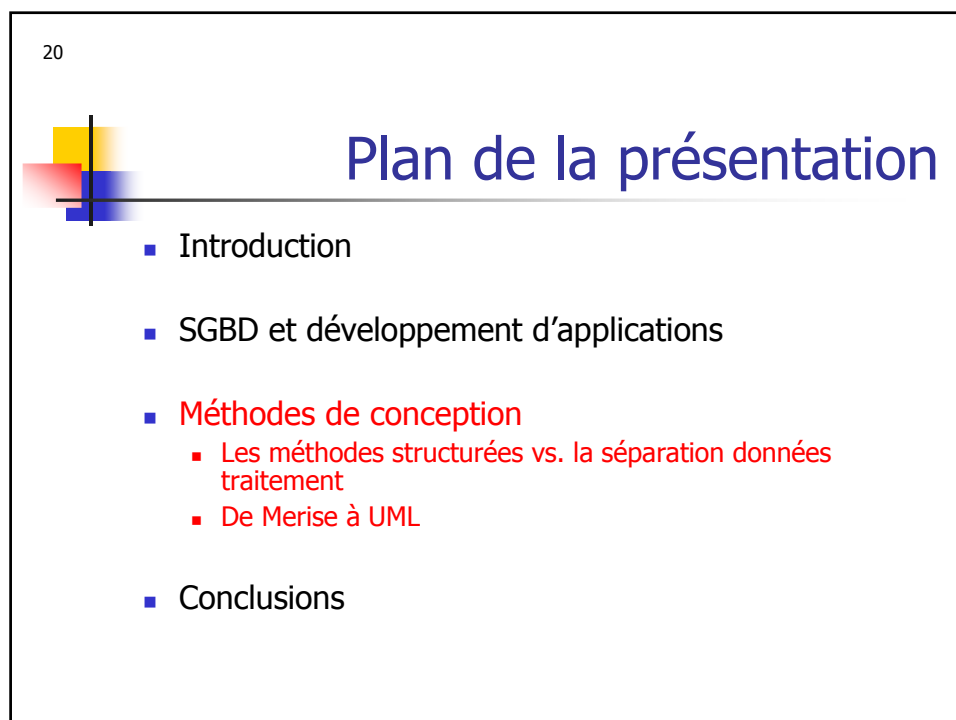
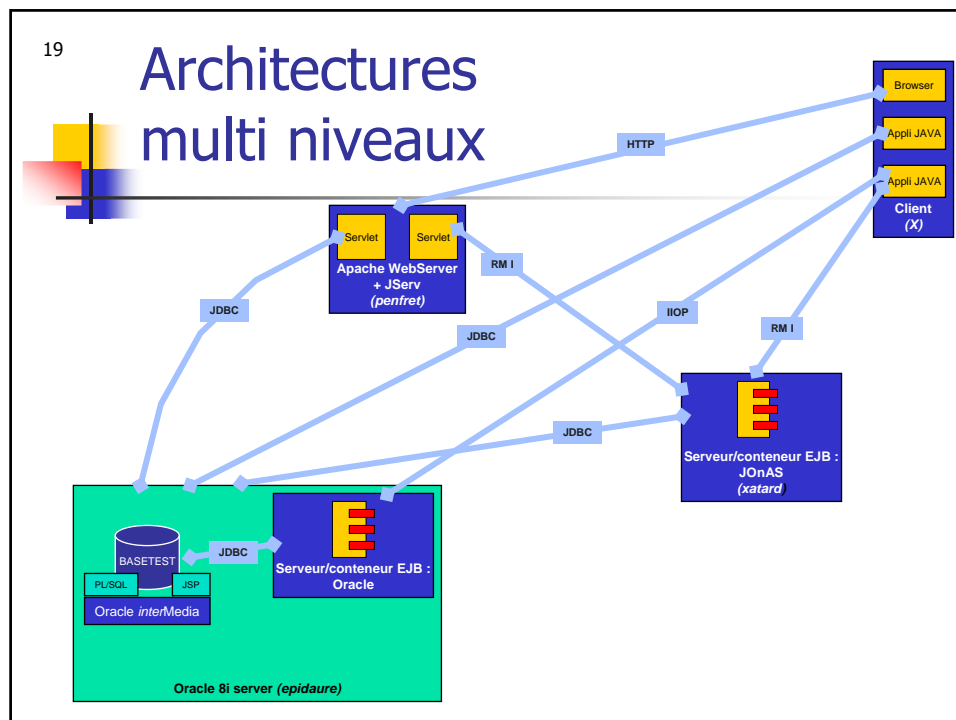
16

Objectif atteint ?

- Le développement au dessus d'un SGBD reste difficile
- « Impedance mismatch »
 - Types de données différents
 - Types relationnels simples
 - Types LP structurés et imbriqués
 - Moteur d'exécution différents
 - Ensembliste pour les SGBD relationnels
 - Tuple à tuple pour les LP



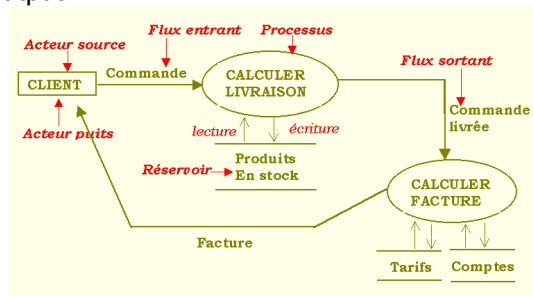
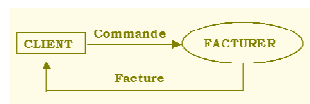
- 18
- ## Des solutions existent
- **Transparence de la programmation OO au dessus d'un SGBD Relationnel?**
 - Quand on supprime une association entre deux objets en mémoire, on veut faire la même chose dans la BD.
 - **Mapping encapsulé**
 - Les composants permettent d'encapsuler les traductions entre les classes du programme et leur stockage relationnel.
 - SGBDR/SQL + Archi/composants + LPOO
 - Solution opérationnelle (JDO / EJB / J2EE)
 - Lourde à mettre en œuvre / pertes en performances



21

Années 70 : Méthodes structurées

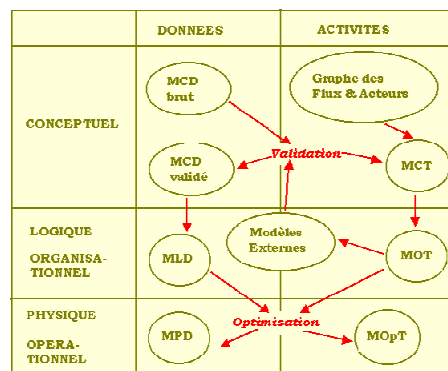
- Division d'une fonction en une myriade de sous fonctions plus simples (analyse structurée)
 - Jusqu'à obtention du niveau de facilité humainement gérable
- Data Flow Diagram :
 - process, input, output
 - Actor, datastore

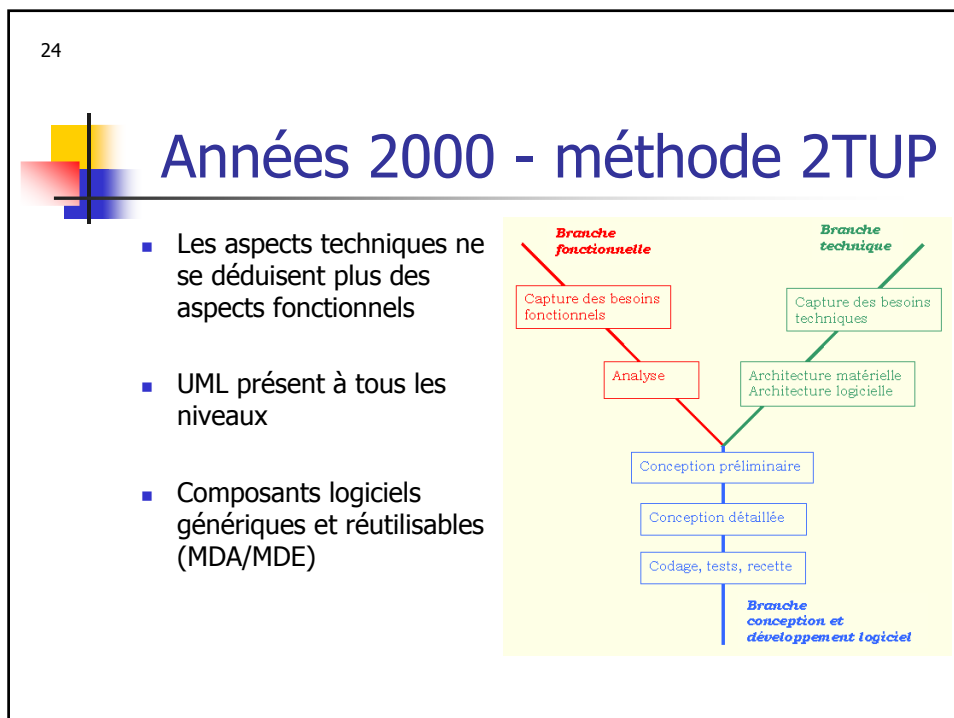
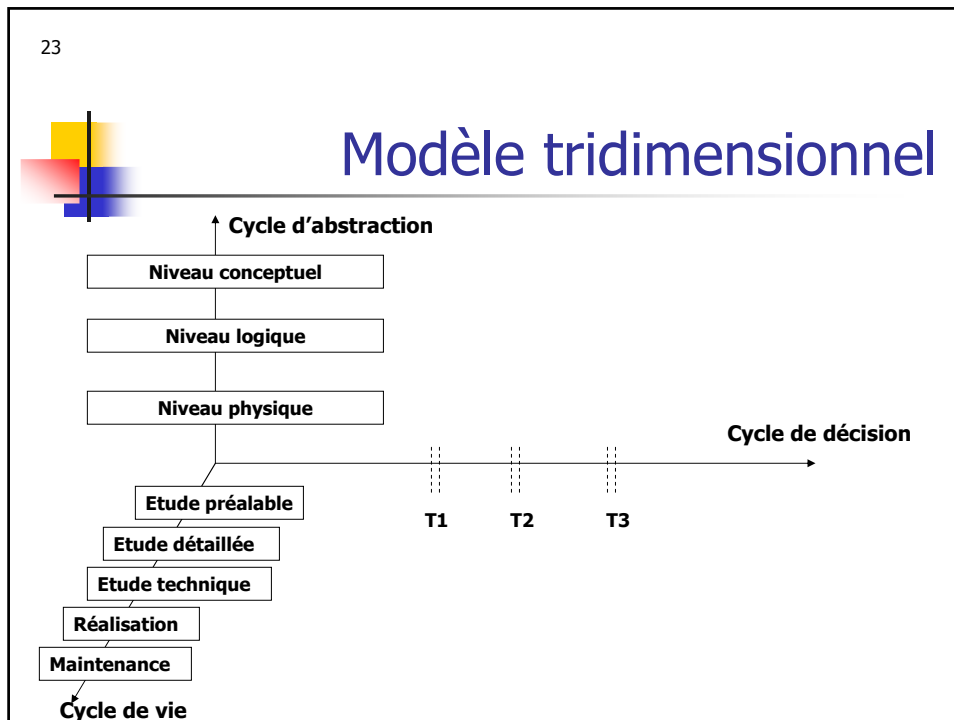


22

Années 80 : méthodes systémiques

- Merise
 - Cycle d'abstraction
 - Isoler l'invariant du SI
 - Cycle de vie
 - Gestion d'un projet d'informatisation
 - Modèle en cascade
 - Cycle de décision
 - insertion des S.I. dans une organisation





25



Conclusion

- Intégration aux nouveaux langages de programmation
 - Garantir la cohérence entre les objets du programme et les tuples de la base
- Intégration aux standards architecturaux
 - SGBD, client serveur et architectures multi-couches
- Intégration aux nouveaux standards de conception
 - UML standard pour différentes méthodes

26



Conclusion (2)

- 1^{ère} génération de S.I.
 - OLTP: On Line Transaction Processing
- Caractéristiques
 - Transactions simples et nombreuses (k x 100 /s)
 - Architectures centralisées
 - Peu d'interfaces (formulaire de saisie)
 - Langage COBOL ou C
 - TPC A et B (<http://www.tpc.org/>)
- Le SGBD est un acteur majeur
 - 1 Formulaire = +/- 1 requête
 - Traitements lourds en batch
 - Générateur de rapports

27

Conclusion (3)

- Progrès réalisés : vers des SI pilotant
 - Orientés processus
 - ERP (PGI) : Enterprise Resource Planning
 - Orientés documents
 - Systèmes de recherche d'information
 - Orientés décision
 - Datawarehouse : données historisées
 - OLAP: On Line Analysis Processing
 - Orientés connaissances
 - ECD : Extraction de Connaissances à partir des Données
 - Intégration de bases de données, d'analyse de données, statistiques et apprentissage (data mining)

28

Vers les SI décisionnels

