

**ETL**

Extract, Transform, Load

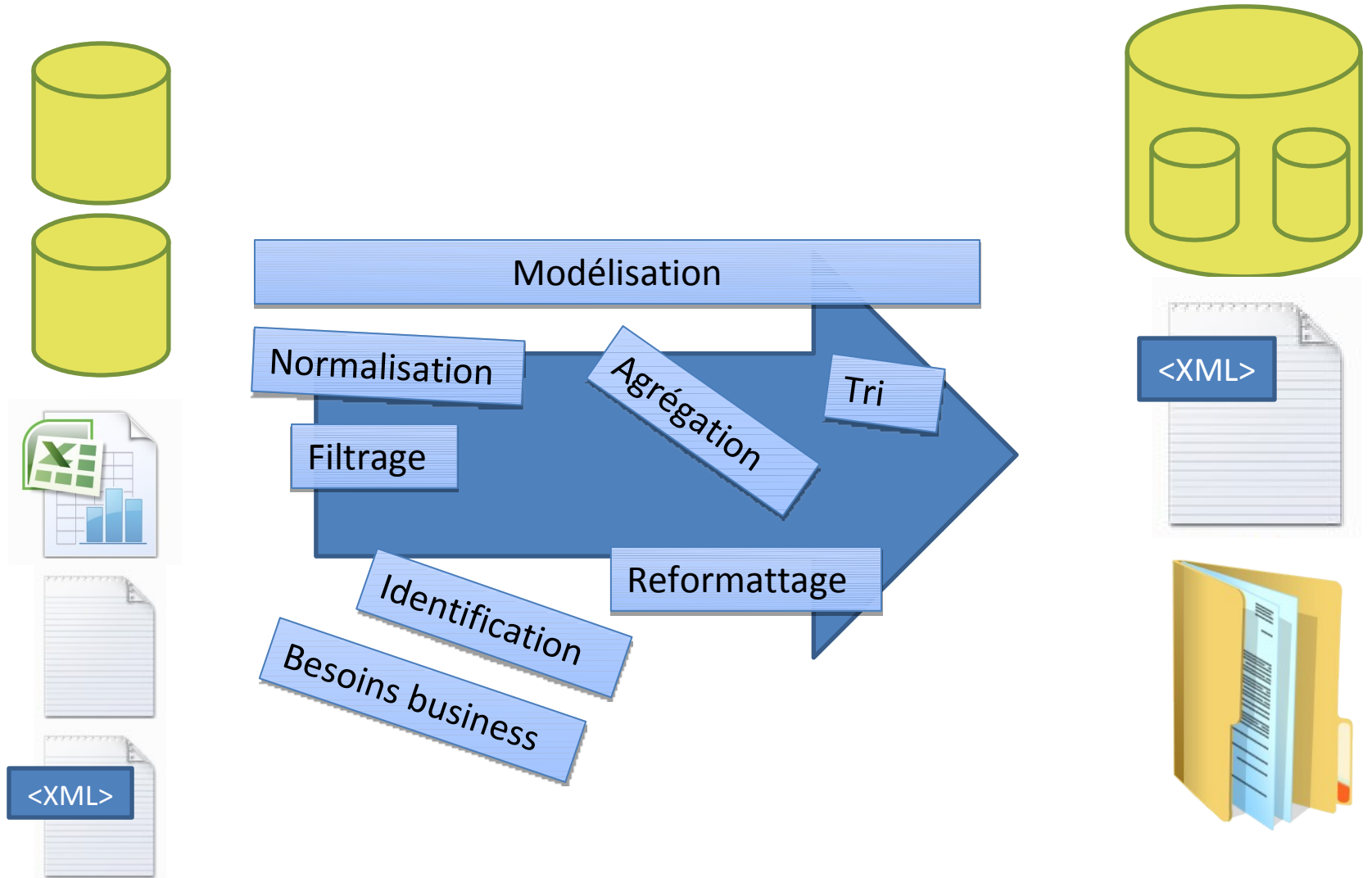
# Plan

- Introduction
- Extract, Transform, Load
- Démonstration
- Conclusion

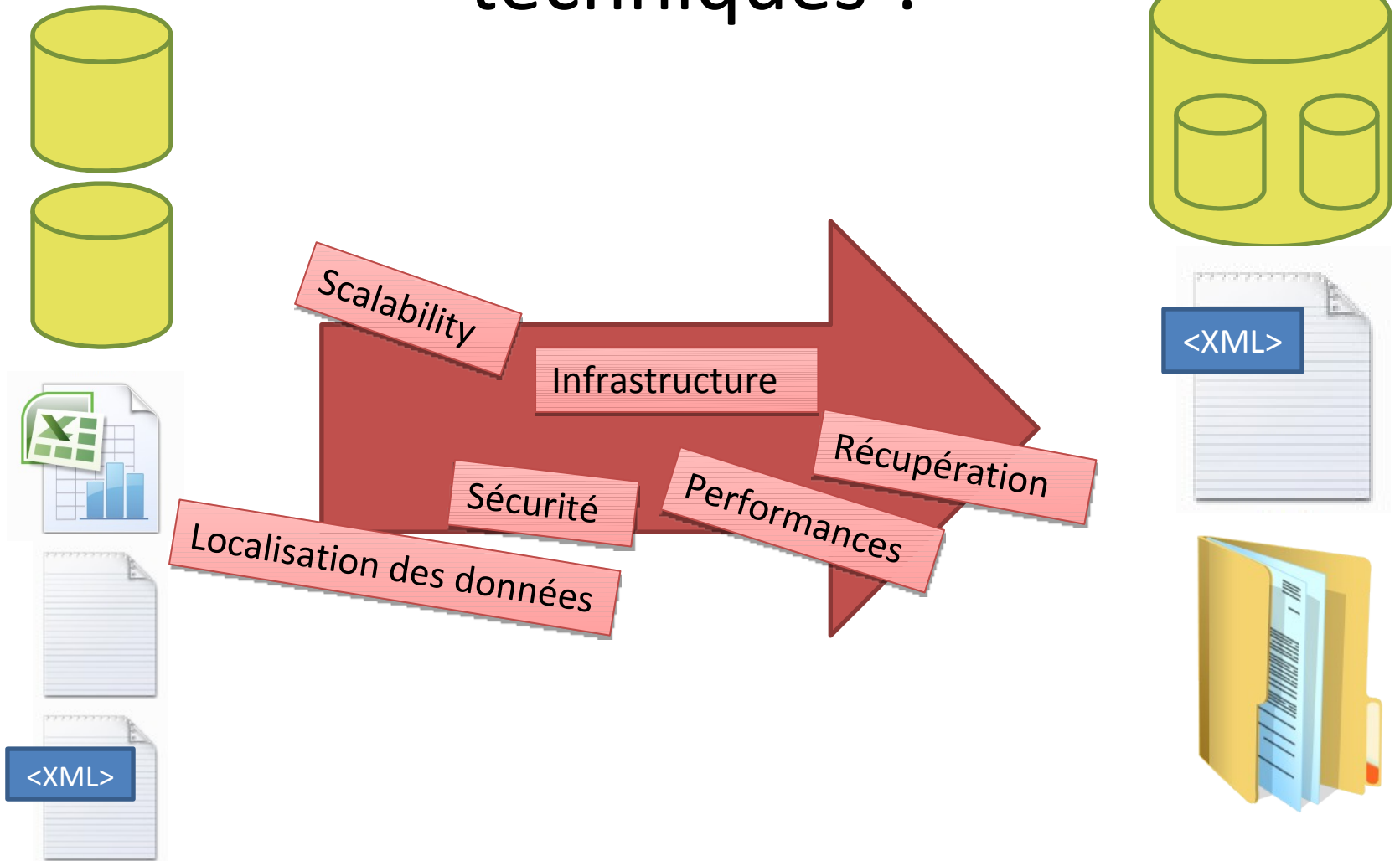
# Plan

- Introduction
- Extract, Transform, Load
- Démonstration
- Conclusion

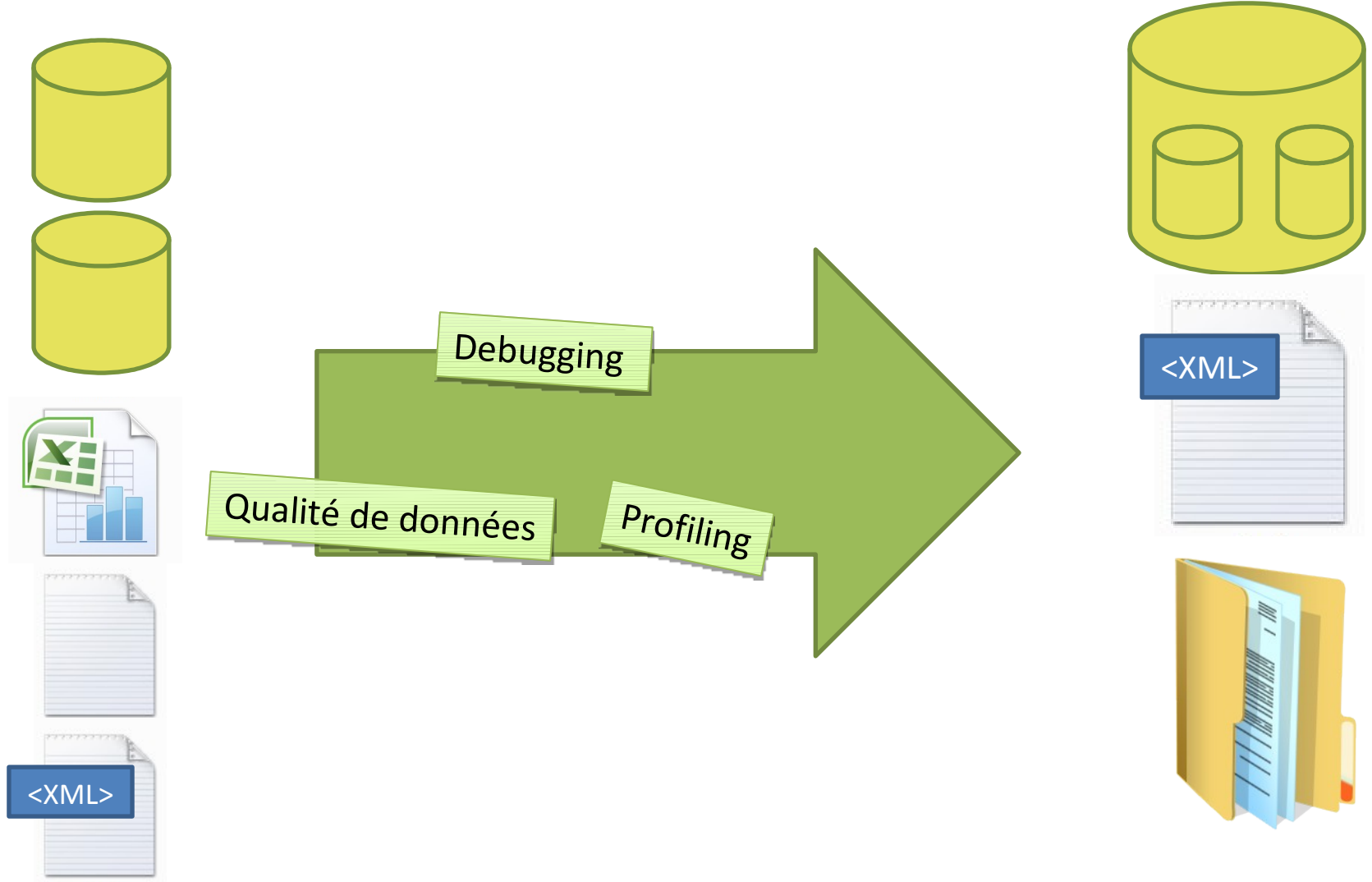
# Problématique: Quoi ?



# Problématique: Contraintes techniques ?



# Problématique: Et après ?



# Plan

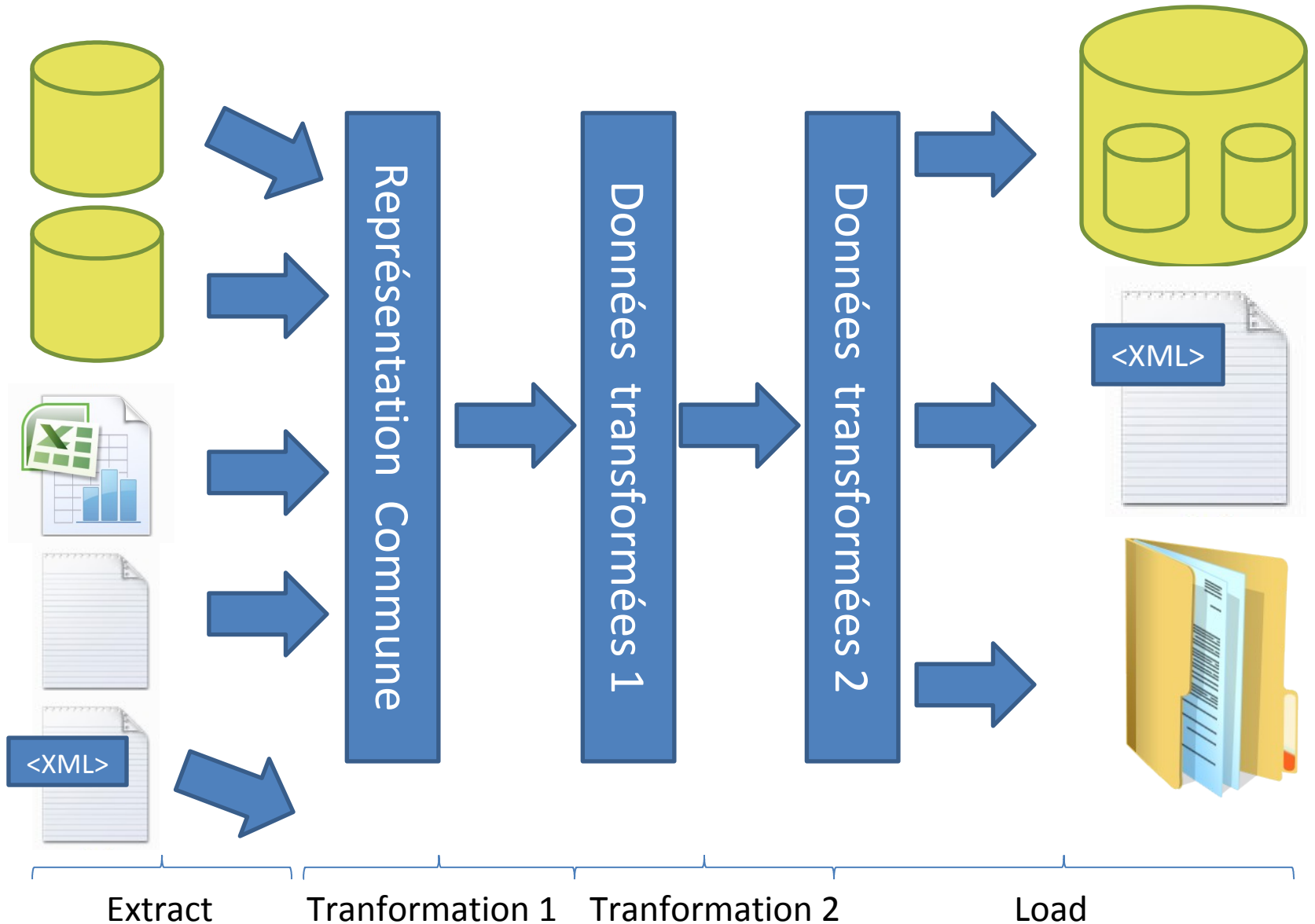
- Introduction
- Extract, Transform, Load
  - Définition
    - Détail E – T – L
    - Concepts proches
  - Historique
  - Complexité des solutions
  - Design
- Démonstration
- Conclusion

# ETL

- **Extract** : Extraction des données depuis les différentes sources.
- **Transform** : Point fort d'ETL. Partie de l'architecture qui trie, filtre, agrège, dérive, regroupe, etc. différentes données.
- **Load** : Stockage des résultats de transformation (typiquement, chargement de ces résultats dans un data warehouse).



# Architecture E – T – L



# Extract

- Exemple de sources de données
  - Base de données relationnelles
  - Fichiers plats ou semi-structurés (XML)
  - Web spidering
  - Capture de données à l'écran (screen scraping)
  - Communication avec d'autres applications  
(notamment pour l'intégration de legacy systems)
- Vérification de la validité des données
- Génération d'une représentation uniforme pour la transformation

# Transform

- Exprimées sous forme de règles
- Nettoyage des données
  - Normalisation, éclatement, filtrage, sélection, suppression des doublons
- Transformations générales
  - Dérivée des valeurs, reformatage, transposition
- Transformation d'agrégation
  - Jointure, agrégation statistiques, calcul de totaux
- Répond à des besoins techniques et business

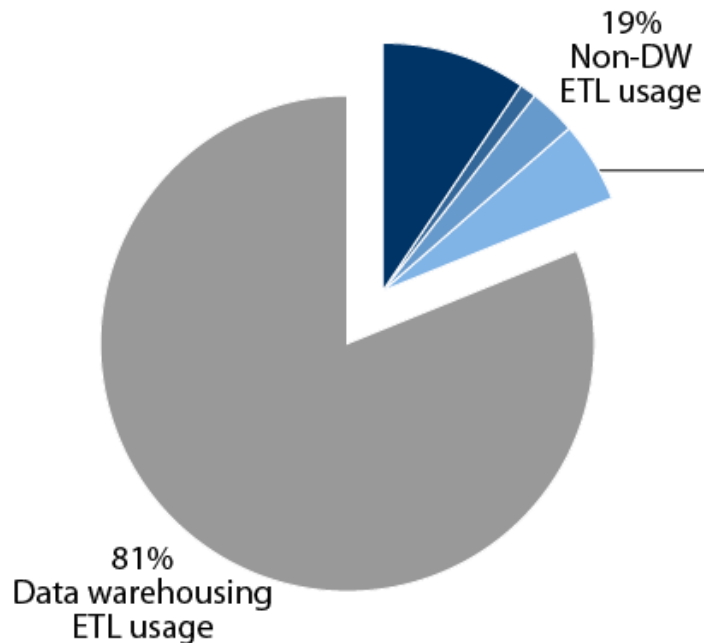
# Load

- Typiquement, cible des data warehouse, data marts
- Intègre les données dans une staging table (récupération, stabilité), puis les publie
- Crée les rapports adéquats
- 
-

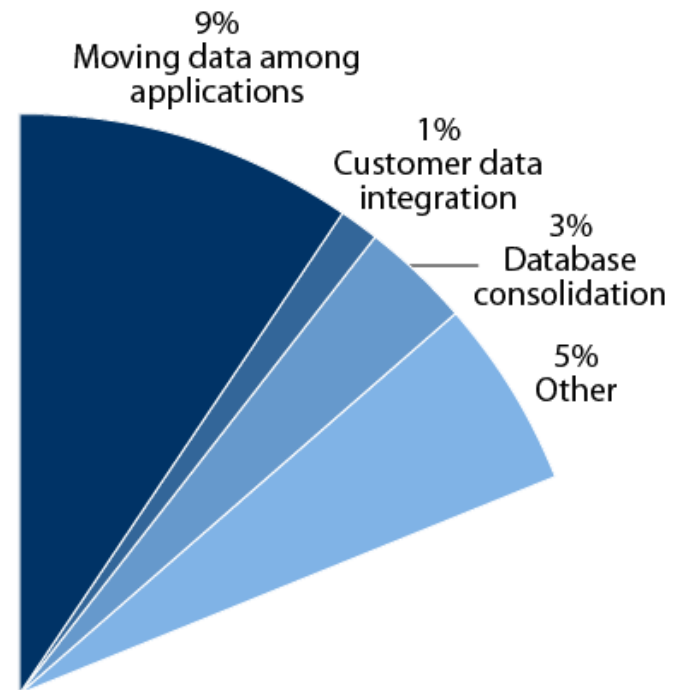
# Autres utilisations

- Utilisation (2005)

“What is the split between your ETL usage for data warehousing (DW) tasks and for non-DW tasks?”



“What is your primary non-DW ETL usage?”



Base: 28 vendor-provided ETL reference customers (percentages do not total 100 due to rounding)

Source: Forrester, 2005

# Attentes associées

- **Fiabilité** : le processus ETL doit toujours fonctionner identiquement, de sorte à fournir à intervalle régulier les données attendues.
- **Disponibilité** : le data warehouse doit atteindre le service attendu, notamment en terme de disponibilité.
- **Flexibilité** : un data warehouse de qualité doit évoluer avec le business qu'il dessert. Les processus ETL correspondant doivent donc pouvoir s'adapter aisément.

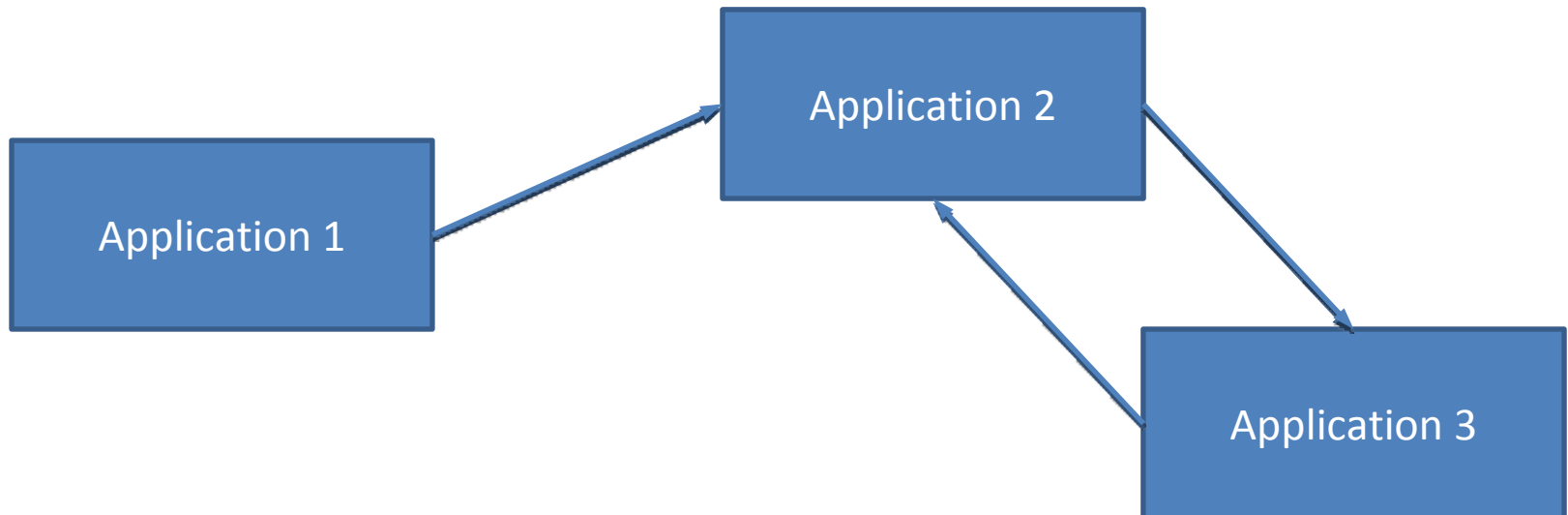
# Concept associés

- EII (Enterprise Information Integration)
  - Vise à fournir une vue sur des données
  - Différentes sources, mais déjà agrégées
    - Pas de transformation



# Concept associés

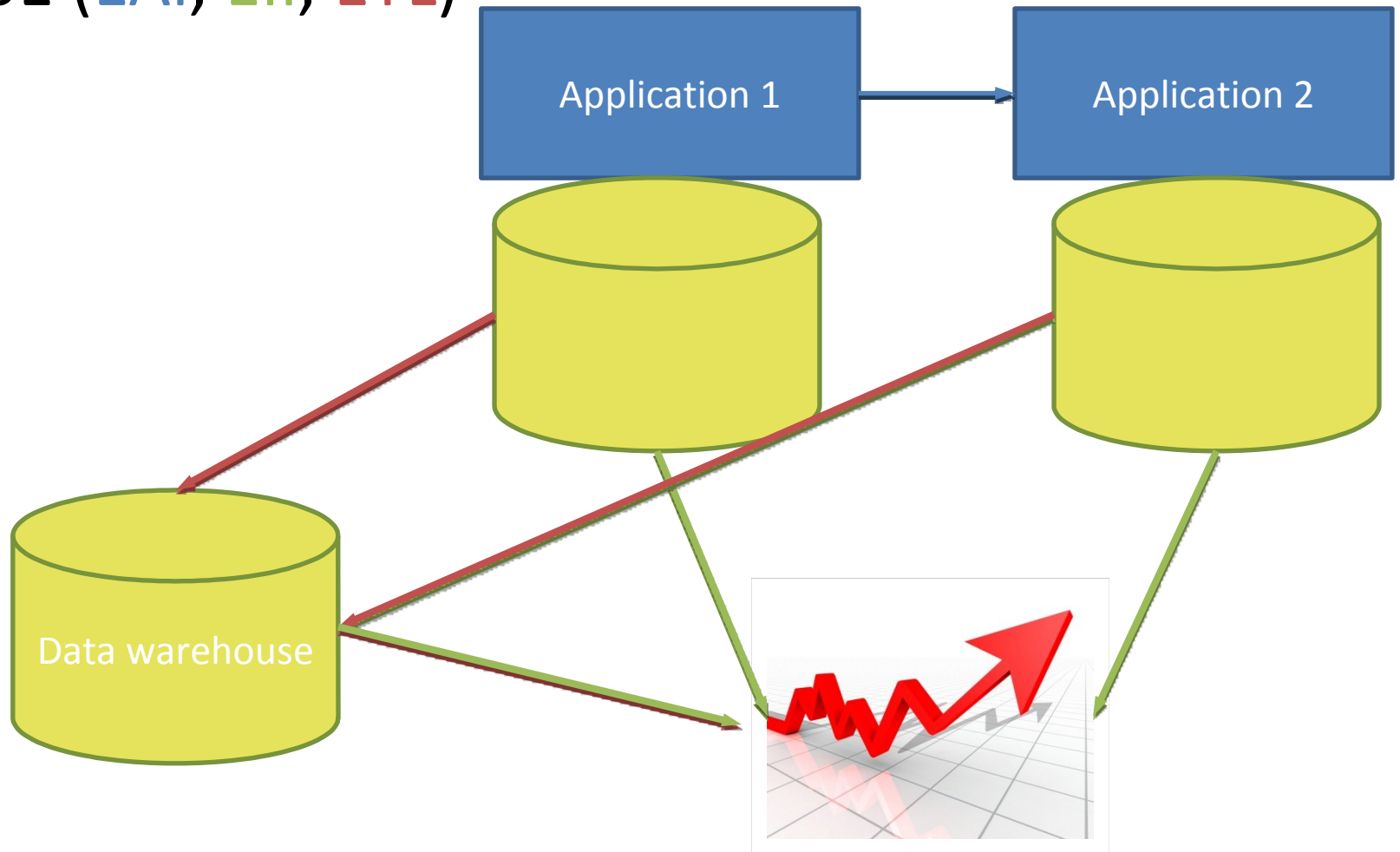
- EAI (Enterprise Application Integration)
  - Vise à faire communiquer des applications
    - Plus grande automatisation des processus
  - Messages courts, peu de données à la fois
  - Event-driven





# Concept associés

- 3E (EAI, EII, ETL)



# Plan

- Introduction
- **Extract, Transform, Load**
  - Définition
  - Historique
  - Complexité des solutions
  - Design
- Démonstration
- Conclusion

# Historique

- Génération 0
  - Depuis que les bases de données existent
  - ETL sur mesure
  - Continuent à exister, mais en déclin

# Historique

- Première génération d'outils ETL (15 ans)
  - Toolset pour générer du code source
  - Exécuté en batch, nécessité de gérer des scripts à la main.
  - Fichier intermédiaires pour chaque étape de la transformation
  - Parfois une interface graphique pour définir des processus simples
    - Sinon, écrire du code pour la transformation

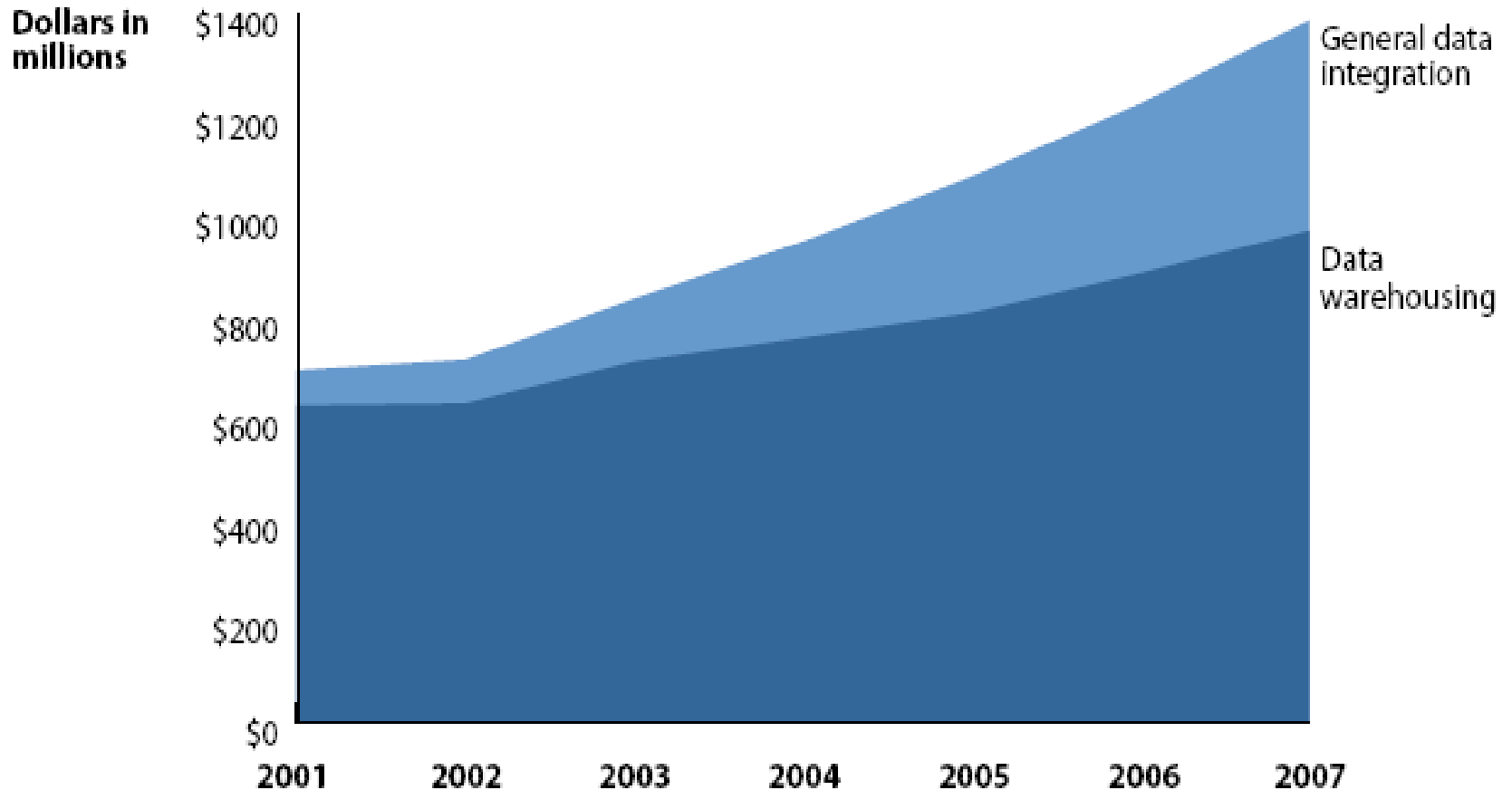
# Historique

- Seconde génération d'outils ETL (10 ans)
  - Toolset pour générer des applications
  - Traitement parallèle des données
  - Tout en mémoire, transformation en pipeline.
  - Nombreuses transformations supportées

# Historique

- Aujourd'hui – Troisième génération
  - Architecture ETL modulable, « fully integrated »
  - Architecture distribuée
  - Large gamme de choix
    - Interface user-friendly
    - Nombreuses transformations prédéfinies
    - Nombreuses source (et destinations) pour les données
  - ETL “real-time”
    - Utilisation de “micro-batch” vers des “real-time partitions” de data marts.
  - ELT
    - Le moteur de base de données devient, au travers de SQL l'outil de transformation.

# Marché que représente ETL



Source: Forrester, 2005

# Plan

- Introduction
- **Extract, Transform, Load**
  - Définition
  - Historique
  - **Complexité des solutions**
  - Design
- Démonstration
- Conclusion



# Complexité : intégrité

- Des données elle-mêmes

Conserver la cohérence des données si un processus est interrompu : **recoverability**.

- De l'ensemble du processus

Le processus doit pouvoir être relancé en obtenant le même résultat : **rerunnability**.

# Complexité : intégrité

- Procédures pour assurer l'intégrité
  - Data profiling : bien connaître le type de données à traiter afin de prévoir tous les cas possibles.
  - Monitoring : surveiller les données sources  
Récouter des méta-données à chaque étape, détection précise d'erreurs et possibilité de rollback.
  - Points de contrôles : Snapshot à certains moments clés du processus

# Complexité : performances

- Les quantités de données sont énormes
- L'utilisation d'ETL croît continuellement

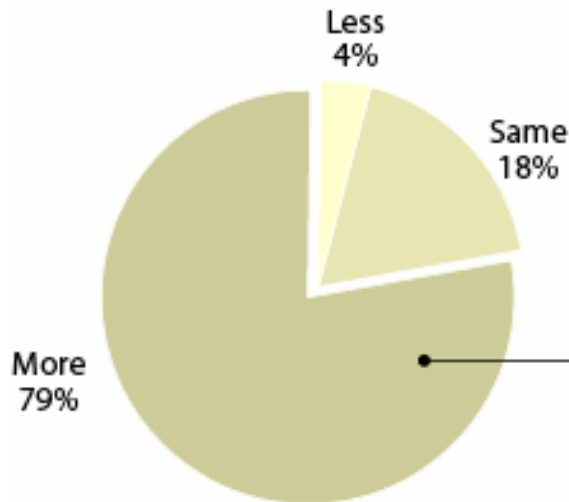
<b>Daily volume</b>	<b>Moderate ETL scalability</b>	<b>High ETL scalability</b>	<b>Extreme ETL scalability</b>
Rows or records	250 million	500 million	1 billion
Transaction	5 million	12 million	25 million
Storage	10GB	100GB	500GB
Files	100	500	1,000
<b>Daily complexity</b>			
ETL jobs	100	500	1,000
Source databases	25	50	100
Target databases	25	50	100
Dimensions per fact	6	12	24

Source: Forrester, 2005

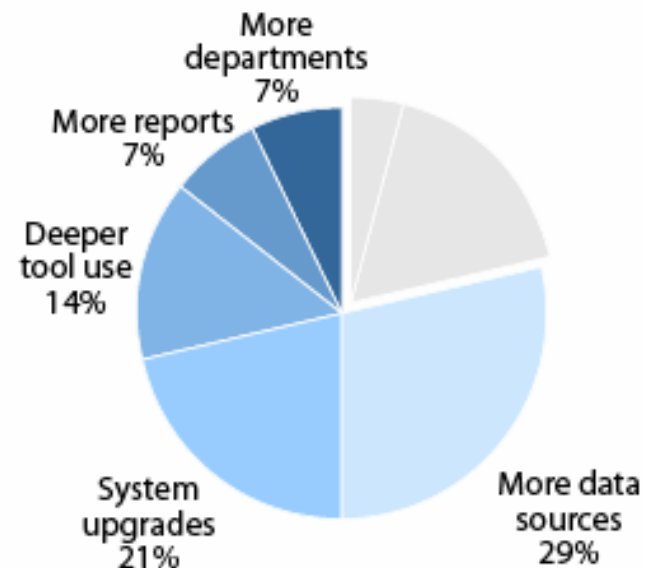
# Complexité : performances

- Les quantités de données sont énormes
- L'utilisation d'ETL croît continuellement

**"Looking ahead to 2005, will you use your ETL tool more, less, or the same?"**



**"In what areas will you use your ETL tool more?"**



# Complexité : performances

- Éliminer les goulets d'étranglement
  - Manque de mémoire vive
  - Opérations inefficaces dans la DB
  - Trop d'opérations d'entrée/sortie
  - Reconstruction d'agrégats inefficace
  - Écritures inutiles suivies de lecture
  - Filtrage trop tardif des données

# Complexité : performances

- Solutions
  - Minimiser le nombre de transactions avec la DB
  - Partitionner les données : découpage temporel, spatial (offre la scalability)
  - Partitionner des tâches : Chaque tâche doit être effectué à l'endroit le plus optimal dans la chaîne de traitement.

# Complexité : performances

- Monitoring du workflow :
  - Complémentaire aux mesures déjà évoquées
  - Mesure des performance de la plateforme (CPU, RAM, etc) aux différentes étapes du processus
  - Permet d'identifier les goulets d'étranglement lorsqu'ils apparaissent.
- Contraintes de performance et d'intégrité demandent une connaissance détaillée de l'état du système à chaque instant.

# Plan

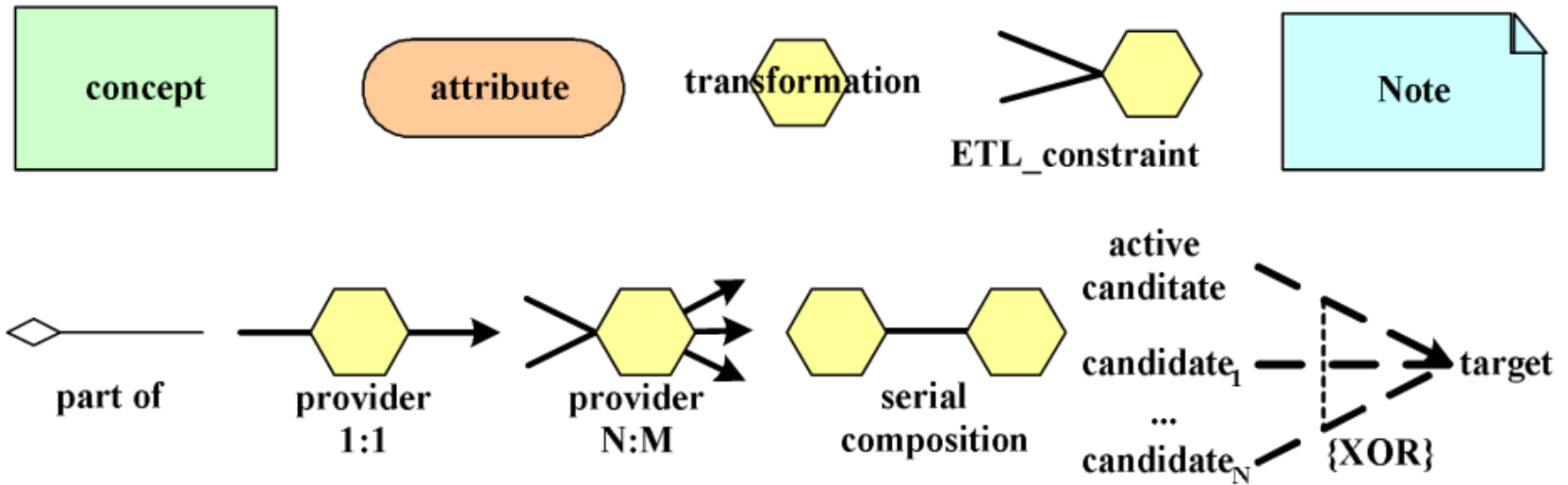
- Introduction
- **Extract, Transform, Load**
  - Définition
  - Historique
  - Complexité des solutions
  - **Design**
- Démonstration
- Conclusion



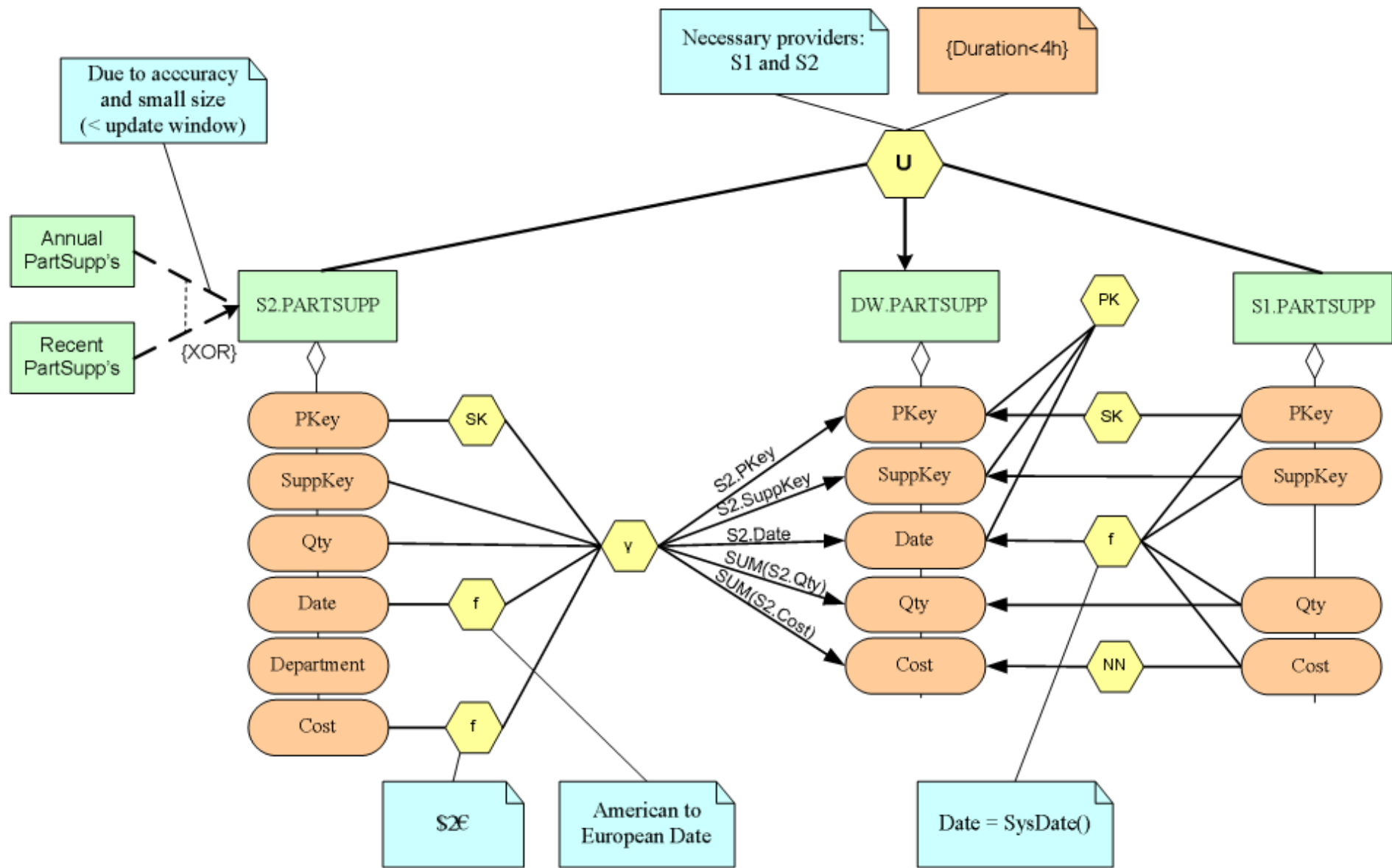
# Formalismes pour la définition de processus ETL

- Pas de standard en vigueur pour la modélisation de processus ETL
- La littérature scientifique propose différents modèles
  - Modèles conceptuels, logiques, etc.
  - Sur les modèles formels, des transformations permettent de réduire la complexité
- Les outils fournissent leur propre formalisation

# Conceptual Model

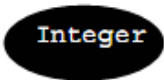






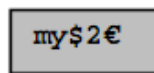
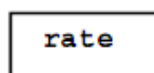

A Methodology for the Conceptual Modeling of ETL Processes, Alkis Simitsis, Panos Vassiliadis, 2004


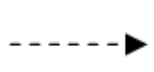




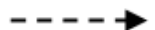
# Graphical Notation for the Architecture Graph

n-ti uB

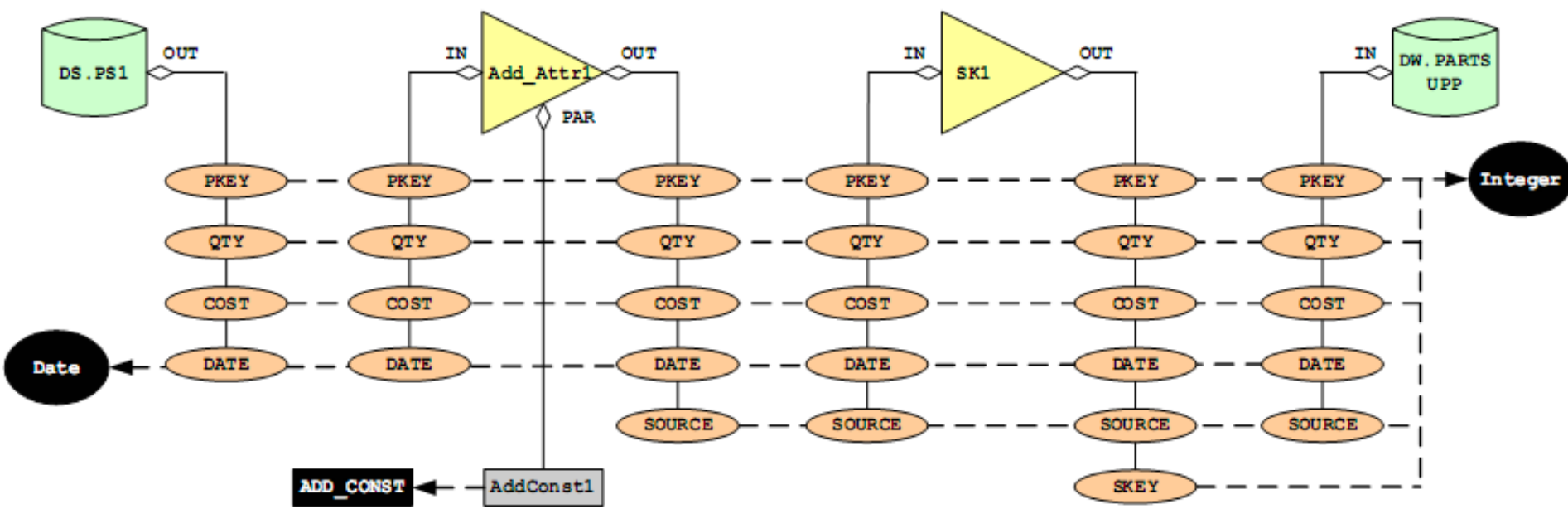
<b>Data Types</b>	Black ellipsis	
<b>Function Types</b>	Black squares	
<b>Constants</b>	Black cycles	
<b>Attributes</b>	Hollow ellipsoid nodes	

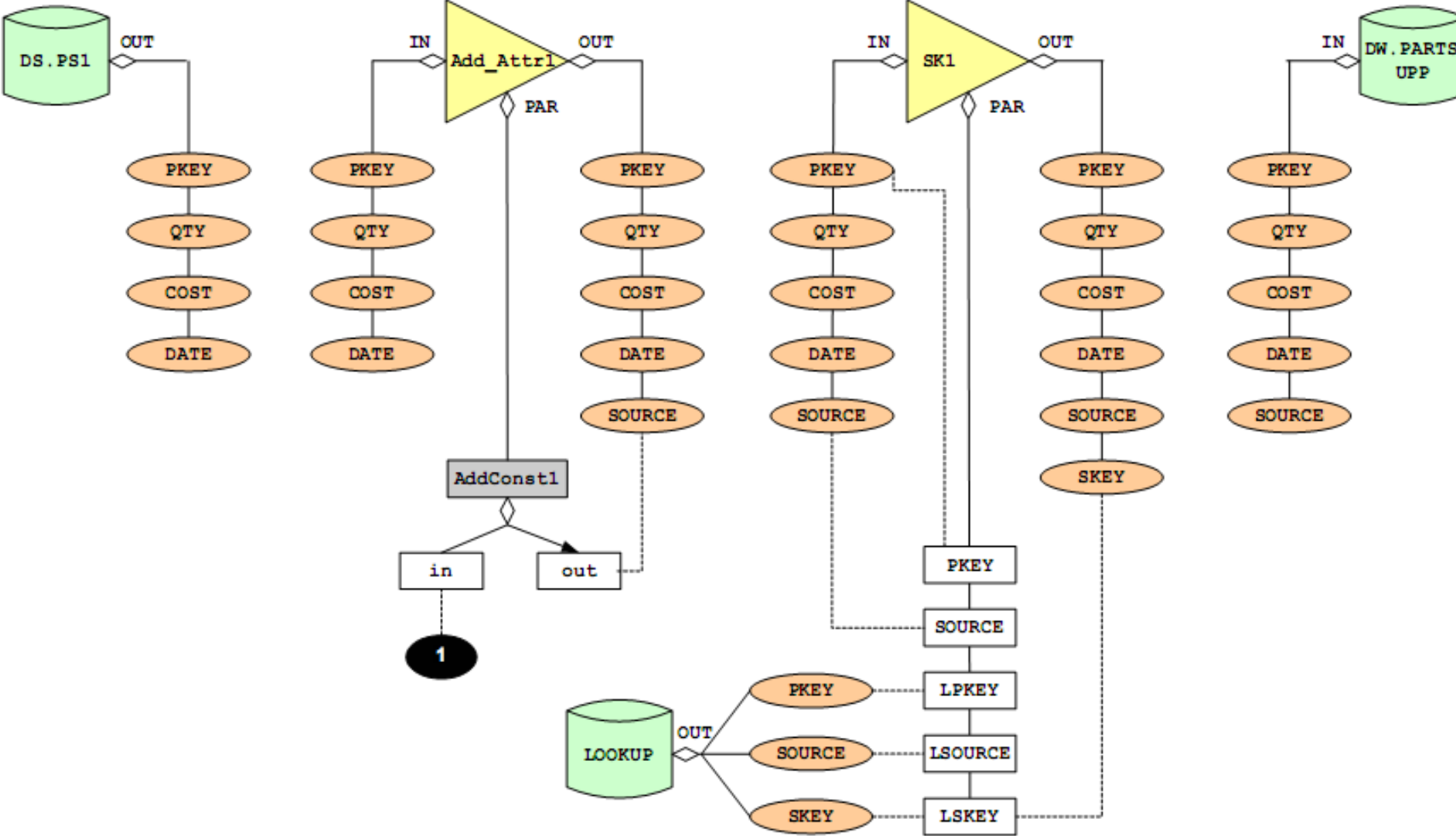
<b>RecordSets</b>	Cylinders	
<b>Functions</b>	Gray squares	
<b>Parameters</b>	White squares	
<b>Activities</b>	Triangles	

<b>Part-Of Relationships</b>	Simple edges annotated with diamond*	
<b>Instance-Of Relationships</b>	Dotted arrows (from instance towards the type)	
<b>Regulator Relationships</b>	Dotted edges	

<b>Provider Relationships</b>	Bold solid arrows (from provider to consumer)	
<b>Derived Provider Relationships</b>	Bold dotted arrows (from provider to consumer)	

\* We annotate the part-of relationship among a function and its return type with a directed edge, to distinguish it from the rest of the parameters.





# Démonstration

- Talend Open Studio
- Outil d'ETL basé sur Eclipse RCP
- Design des processus au moyen de composants graphiques
- Possibilité de compléter “à la main” avec du code
- Le code peut être généré en Java ou en Perl
- Souple : convient aussi bien pour rapidement convertir des données que pour des gros projets d'ETL
- Notions intéressantes : Metadata, Contexts

# Conclusion

- ETL est historiquement bien implanté
  - Nombreuses solutions
  - Mais pas de standard pour le design
- Outils visant à être de plus en plus riches
  - Analyser la qualité du processus
  - Scalability
- Convergence EII, EAI, ETL