

# **SILK, egy modellalapú információintegrációs eszközészlet**

BME

# Tartalomjegyzék

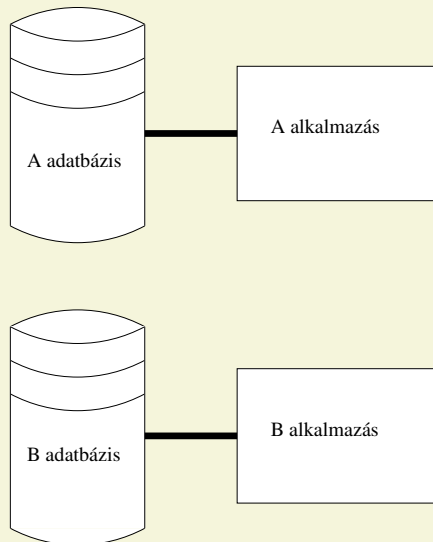
- Miről lesz szó?
- Integrációs probléma és integrációs rendszerek
- Az ontológiák szerepe
- A SILK rendszer környezete és architektúrája
- A modelltárház
- A modelltárház nyelve: SILan
- A SILK megvalósítása, komponensei
- A logikai alapú folyamatok
- A SILK továbbfejlesztésének irányai

## Miről lesz szó?

- A SILK projekt
  - \* SILK: System Integration using Logic and Knowledge (EU 5. keretprogram, IST alprogram)
  - \* 2000–2003
  - \* IQSOFT (magyar), EADS (francia), ICI (román), IDEC (görög)
  - \* A cél: heterogén információforrások integrációja
  - \* Az indíték: egy nagy, magyar távközlési cég magas szintű integrációs problémájának megoldása
- Modelleken és logikai következtetésen alapuló információintegrációs rendszer jellemzői
  - \* metaadat-kezelés, modellezés
  - \* logikai következtetés felhasználása
  - \* deklaratív információintegrálás, mediálás

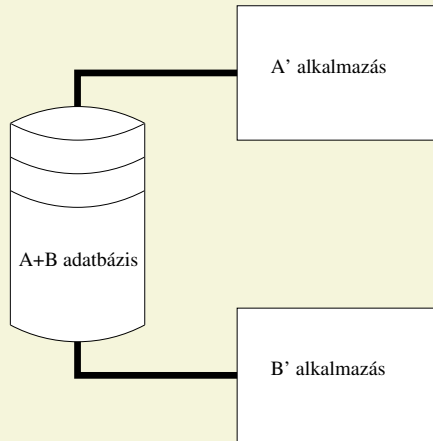
# Integrációs probléma

- Kialakulás
  - \* Adott egy nagy szervezet, független információforrásokkal
  - \* Az igények változnak → adatforrások születnek, változnak
  - \* Redundancia keletkezik
- Problémák
  - \* Melyik forráshoz forduljunk?
  - \* Konzisztencia?
  - \* Tudjuk együtt használni őket?
    - § technológiai különbségek
    - § szemantikai különbségek
- Az integrációs módszerek ezeket a problémákat igyekeznek megoldani

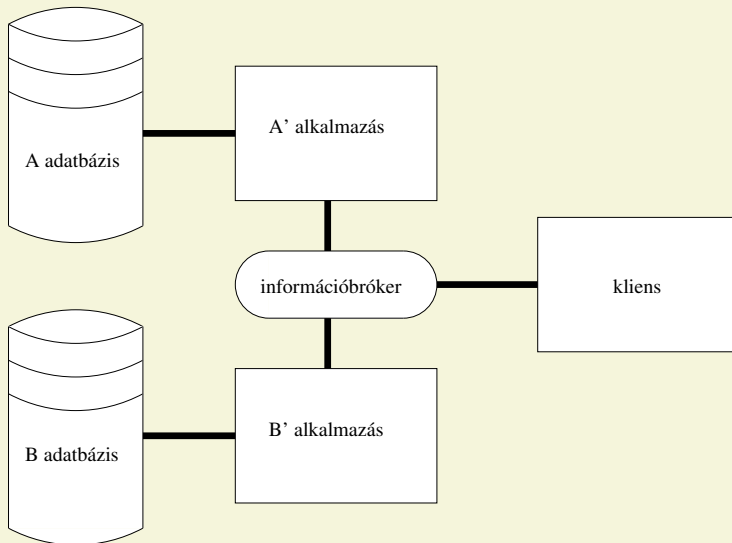


## Ad hoc integráció

- A természetes megoldás
- Összevonjuk az adatforrásokat (adatorientált integráció)
- Összevonjuk az alkalmazásokat (alkalmazásorientált integráció)
- Esetleg mindkettőt meg tesszük
- Jellemzők
  - + biztosítja a konzisztenciát
  - + a lehető leghatékonyabb működés
  - szoros csatolás
  - nem skálázható
  - nagyon költséges fejlesztés
  - \* *revolúciós* megoldás



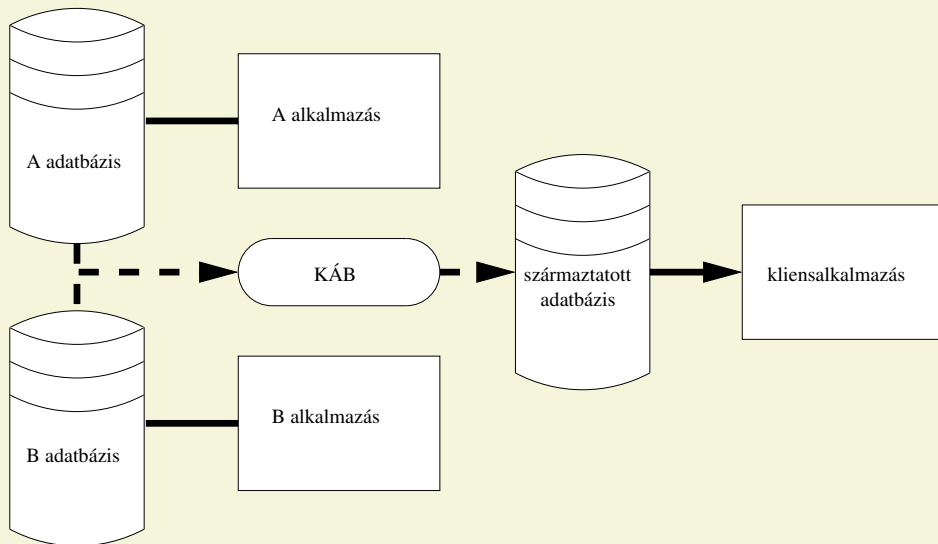
## Az alkalmazásintegráció architektúrája



## Alkalmazásintegráció (megvalósítás és jellemzők)

- Megvalósítás
  - \* Laza csatolás az *információbrókeren* keresztül
  - \* A részt vevő rendszerek üzeneteket küldenek a brókernek
  - \* A bróker átalakítja és továbbküldi az üzeneteket a megfelelő helyekre
  - \* A brókert programozni kell
  - \* Az integrált rendszerekhez csatolók kellenek
- Jellemzők
  - + biztosítja a konzisztenciát
  - + a brókerben koncentrálódik az integrációs logika
  - + *follyamatokat* kezel
  - azokat az új lekérdezések tudja megvalósítani, amelyeket beprogramozunk
  - nem ad átfogó képet az integrált rendszerekről

## Az adatintegráció architektúrája

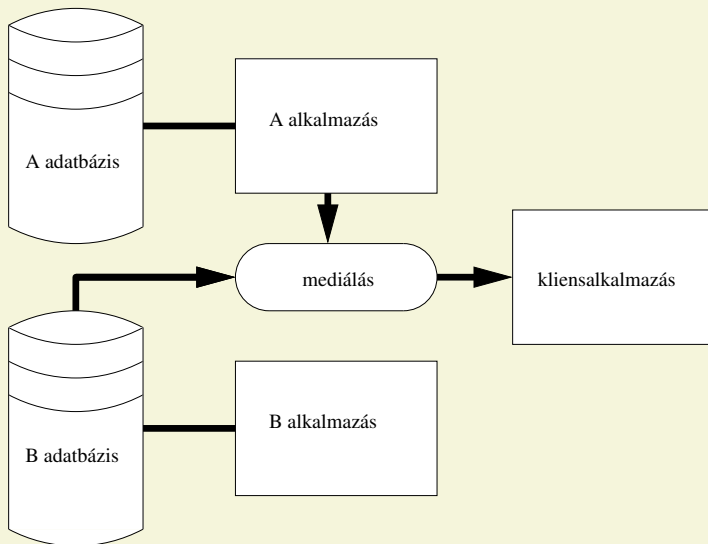




## Adatintegráció (megvalósítás és jellemzők)

- Megvalósítás
  - \* Laza, off-line csatolás a *KÁB* segítségével
  - \* A származtatott adatbázis időnként frissül az integrált adatokkal
  - \* Az adatoknak több példányát tároljuk → adattörténet is rendelkezésre áll
  - \* Az új kliens lekérdezheti a teljes egyesített adatbázist
- Jellemzők
  - nem biztosítja a konzisztenciát, csak egy konzisztens képet (adattisztítás)
  - + a *KÁB* komponensben koncentrálódik az integrációs logika
  - + adattörténetet kezel → adatbányászat lehetséges
  - az adatok ritkán frissek
  - hatalmas adatmennyiség → nagy erőforrásigény (tár, processzor, hálózat)
  - igazából csak adatbázisokat lehet így integrálni

## Az információintegráció architektúrája



## Információintegráció (megvalósítás és jellemzők)

- Megvalósítás
  - \* Laza csatolás a *mediálást* végző komponens segítségével
  - \* Az adatok helyett csak a metaadatokat gyűjtjük össze
  - \* Az adatokat on-line módon állítjuk elő
  - \* Az integrált rendszerekhez csatolók kellenek
- Jellemzők
  - nem biztosítja a konzisztenciát, csak egy konzisztens képet (adattisztítás)
  - + a mediálást végző komponensben koncentrálódik az integrációs logika
  - + az adatok mindig frissek
  - az adatokat esetleg bonyolult úton kell előállítani → lassabb lekérdezések
  - + csak a szükséges adatmennyiség → kis erőforrásigény (tár, processzor, hálózat)
  - + bármilyen adatforrást (pl. webszolgáltatást is) integrálhat

## Az evolúciós integrációs módszerek összehasonlítása

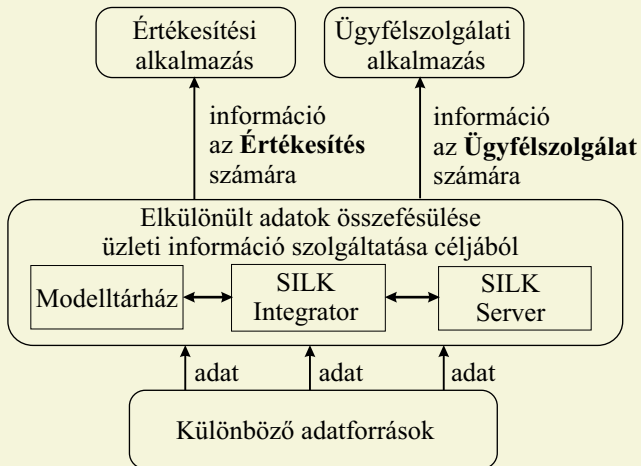
Típus	Hozadék	Tárgy	Fő komponens	Felhasználó
<i>alkalmazásintegráció</i>	szinkronizáció	folyamatok	információbróker	szervezet
<i>adatintegráció</i>	trendanalízis	adattörténet	adattárház	döntéshozók
<i>információintegráció</i>	termelékenység	élő adatok	mediátor	végfelhasználók

- Az egyes módszerek más feladatokra koncentrálnak
- Egyik sem jobb a másiknál
- Egyidejűleg többet is érdemes lehet alkalmazni

## Az ontológiák/metaadatok szerepe

- Rengeteg adat áll a rendelkezésünkre
  - \* Nem az a kérdés, van-e adat, hanem hogy megtaláljuk-e
  - \* Az adatokat rendszerezni kell, hogy ne vesszünk el bennük
  - \* A metaadatok (modellek) egyre fontosabbá/értékesebbé válnak
- A változással a metaadatok szintjén könnyebb lépést tartani
  - \* A modellek kisebbek
  - \* A modellek alapján automatikusan kell feldolgozni az adatokat
  - \* Az alkalmazásfejlesztésben is érvényes ez az elv (ld. MDA)
- Elosztott rendszereket nem igazán lehet metaadatok nélkül üzemeltetni
  - \* Grid
  - \* P2P rendszerek
  - \* Webszolgáltatások
  - \* A világháló
- Az információintegráció ezekre a tulajdonságokra épít

## A SILK rendszer környezete és architektúrája



## SILK információintegráció

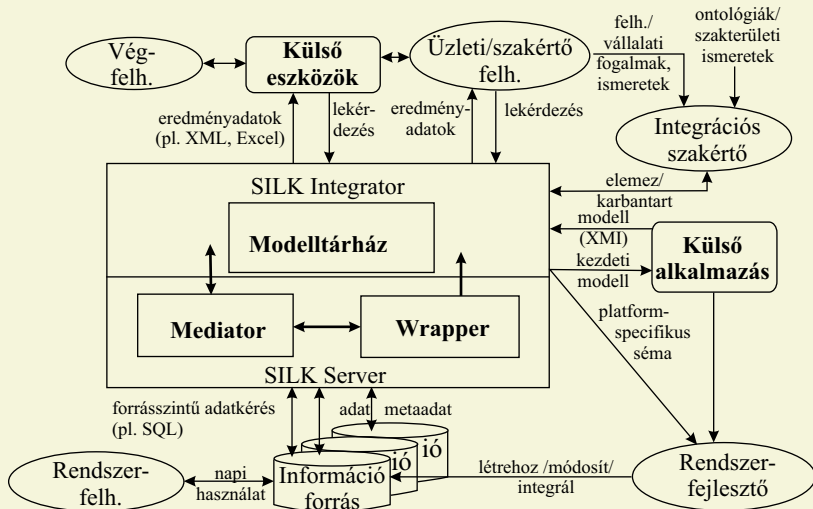
- A három réteg alulról felfele:
  - \* különböző forrásokból származó adatok
  - \* SILK: a mediálást végző komponens
  - \* információ különböző fogyasztók számára
- A fő SILK komponensek:

**SILK Server** a „futási idejű” komponens, a lekérdezéseket válaszolja meg

**Modelltárház** az integrációhoz szükséges modelleket tárolja

**SILK Integrator** a modelltárházat tartja karban

## A SILK alkalmazási környezete





## A SILK felhasználási módjai

- Modellezés
  - \* meglévő modellek betöltése
    - § külső alkalmazásokból
    - § magukból az adatforrásokból
  - \* új modellek kialakítása
  - \* modellek összekapcsolása
- Lekérdezés
  - \* külső eszközökkel vagy közvetlenül
  - \* a modelltárházbeli modelleken
  - \* egyszerre több adatforrást felhasználva
- Integrációs visszacsatolás
  - \* a kialakított új modellek és kapcsolatok alapján
  - \* javaslat egy integrált rendszer kialakítására
  - \* pl. ad hoc integrációhoz vagy adattárház kialakításához

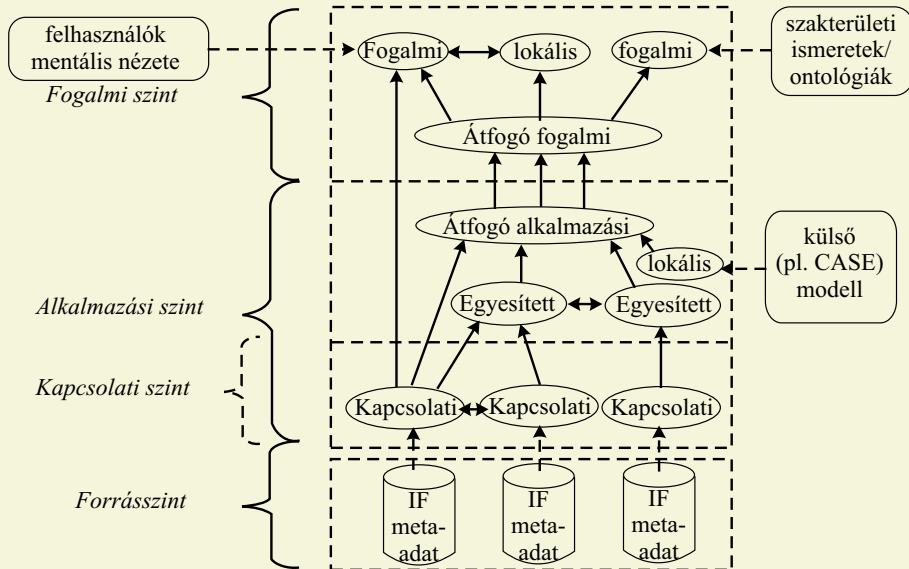
## A SILK fő komponenseinek feladatai

- Wrapper
  - \* elrejt a technológiai sokféleséget
  - \* adatot szolgáltat
  - \* metaadatot szolgáltat
- Mediátor
  - \* magasszintű, felhasználóspecifikus kérdéseket
  - \* visszavezet az egyes adatforrásokra vonatkozó részkérdésekre, és
  - \* a részeredményeket összeállítja egy értelmes válasszá
- Integrátor
  - \* támogatja a modellek kialakítását és kezelését
  - \* tartja a kapcsolatot a felhasználókkal és a külső alkalmazásokkal
  - \* tartja a kapcsolatot a Wrapper és a Mediator komponenssel

## A modelltárház tartalma

- objektumorientált modellek
  - \* UML osztálydiagramok: strukturális jellemzők, alapvető kapcsolatok leírása
    - § attribútumok
    - § öröklődés
    - § ...
  - \* (LL modellek: magasszintű fogalmak és szerepek leírása)
- OCL korlátok az egyéb jellemzők számára
  - \* invariánsok
- leképezések (kapcsolatok a modellek között)
  - \* megfeleltetések (pl. két elem ugyanazt modellezi)
  - \* absztrakciók (pl. az egyik elemből származtatható egy másik)

## A modelltárház felépítése



## A modellek osztályozása

- Absztrakciós szint szerint:

**alkalmazási szint** megvalósított vagy megvalósítható rendszerek modelljei (UML)

**fogalmi szint** felhasználói nézetek (UML vagy LL)

- A lefedett terület nagysága szerint:

**lokális modell** a teljes rendelkezésre álló információnak egy kis része

**kapcsolati modell** a lokális modell speciális esete

\* egy konkrét adatforrás modellje

\* importált modell + a kapcsolódáshoz szükséges információk

**egyesített modell** több rendszert átfogó (egyesítő) modell

## Az adatforrások modellezése

- Minden forrást az UML osztálydiagramok nyelvén kell modellezni
- Relációs adatbázisok
  - \* adatbázis → modell
  - \* tábla → osztály
  - \* oszlop → attribútum
- XML-fájlok (DTD alapján)
  - \* fájl → modell
  - \* elem → osztály
  - \* attribútum → attribútum
  - \* gyerekviszony → kompozíciós asszociáció

## SILan, objektumorientált elemek

```
model Gyár {
  class Termék {
    attribute String gyártási_szám;
    primary key gyártási_szám;
  };

  class Autó: Termék {
    attribute String gyártó;
    attribute Integer ár;
    constraint self.kerék.méret>2;
  };

  class Kerék: Termék {
    attribute Integer méret;
  };

  association 'Autó-Kerék' {
    connection Autó composite;
    connection Kerék;
  };
};
```

**osztály** egy halmaz; a halmaz elemei az osztály *példányai*

**asszociáció** egy halmaz;  $n$  végződésű asszociáció  $\rightarrow n$ -esek halmaza; a halmaz elemei az asszociáció *kapcsolatai*

**attribútum** egy függvény; osztály  $\rightarrow$  attribútum-típus

**művelet** egy függvény; osztály és paramétertípusok keresztszorzata  $\rightarrow$  a művelet típusa

**öröklődés** a speciális osztály minden eleme az általános osztálynak is eleme

**kompozíció** bináris asszociációban az adott asszociáció vég a másik végződés egy példányához tartozhat

**kulcs** az adott attribútumhalmaz azonosít egy példányt

**invariáns** az adott kifejezés az osztály (asszociáció) minden példányára (kapcsolatára) teljesül

## Az invariánsok értelmezése

Az Autó osztály `constraint self.kerék.méret > 2` invariánsa:

1. `self`: az adott osztály (asszociáció) egy tetszőleges példánya (kapcsolata),
2.  *navigációs lépés* az 'Autó-Kerék' asszociáció mentén az autó egy tetszőleges kerekéhez,
3.  *navigációs lépés* a `kerék méret` attribútumán keresztül,
4. azaz minden autó minden kerekének mérete nagyobb, mint kettő.

Érdekességek:

- az első  *navigációs lépésben* a `kerék` egy implicit szerepnév
- a  *navigáció* UML szemantikája kicsit más



## Megfeleltetés

```
model Peugeot {
  class Jármű {
    attribute String sorszám;
    attribute Integer ár;
    attribute String típus;
    primary key sorszám;
  };
};
model Suzuki {
  class Autó {
    attribute String azonosító;
    attribute Integer ár;
    primary key azonosító;
  };
};
correspondence(j: Peugeot::Jármű,
               a: Suzuki::Autó) {
  constraint
    j.sorszám=a.azonosító
    and j.ár*1000 = a.ár;
};
```

- megfelel egymásnak
  - \* Jármű és Autó
  - \* sorszám és azonosító
  - \* a két ár
- az árak akkor összehasonlíthatók, ha az egyiket szorozzuk ezerrel
- a „korlátnak” nincs logikai jelentése
- Jármű és Autó nem ugyanazokat a példányokat tartalmazzák

## Megfeleltetés logikai jelentéssel

```
model Értékesítés {
  class Áru {
    attribute String termékkód;
    attribute Integer ár;
    attribute String raktári_szám;
    primary key termékkód;
  };
};

correspondence(j: Peugeot::Jármű,
               a: Értékesítés::Áru) {
  constraint
    j.sorszám = a.termékkód
  implies j.ár*1000 = a.ár;
};
```

- megfelel egymásnak
  - \* Jármű és Áru
  - \* sorszám és termékkód
  - \* a két ár
- a legkülső `implies` miatt logikai jelentést rendelünk hozzá
- logikai jelentés: ha a sorszám megegyezik a termékkóddal, akkor a jármű árának ezerszerese az áru ára
- Jármű és Áru ugyanazokat a példányokat tartalmazzák csak más kódolással

## Absztrakció

- Életet (információt) lehel a magasabb szintű modellekbe
- Pontosabban: adatokat rendel a magasabb szintű modellelemekhez az alacsonyabb szintű modellek adatai alapján
- Két feladata van
  1. szűrés: mely adatokból kell származtatni
  2. átalakítás: az adatokat a magasabb szinten elvárt formára kell hozni
- Nem öröklődés jellegű dolog
  - \* nem igaz, hogy az egyik példányai egyben a másik példányai is lennének
  - \*  $m$  adatból  $n$  adatot származtathatunk a segítségével
  - \* a származtatott példányok vagy kapcsolatok struktúrája és tartalma is más lehet, mint amiből származtattunk

## Absztrakciós példa

```
abstraction (s: Peugeot::Jármű -> c: Gyár::Autó) {  
  constraint s.típus = "Autó"  
    implies s.sorszám = c.gyártási_szám  
      and "Peugeot" = c.gyártó  
      and s.ár*1000 = c.ár;  
};
```

- s-ből (*supplier*), egy Peugeot modellbeli Jármű példányból
- c-t (*client*), egy Gyár modellbeli Autó példányt származtatunk
- szűrés:
  - \* csak olyan s-ekből származtatunk, amelyek típusa "Autó"
- átalakítás:
  - \* a gyártási szám megegyezik a sorszámmal
  - \* a Peugeot modellben csak Peugeot gyártmányok vannak
  - \* az árat ugyanazért szorozzuk, amiért a megfeleltetésben

*Van valakinek ötlete több szállítót alkalmazó  
absztrakcióra?*

*És több klienst használóra?*

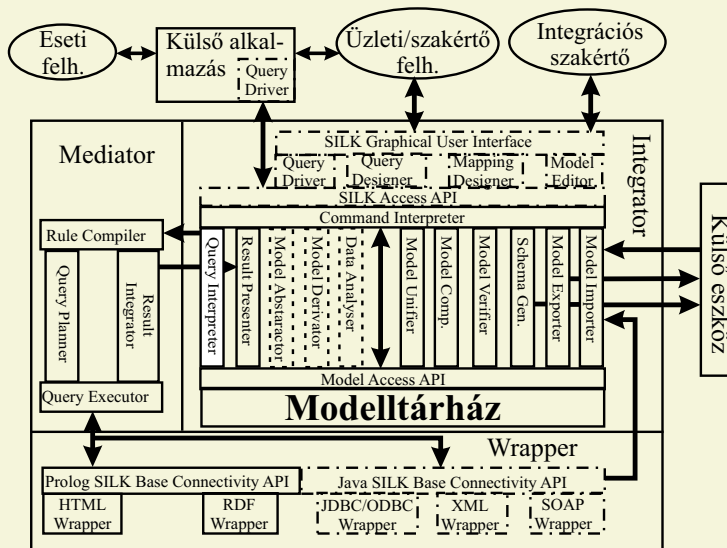
## Lekérdezés

- lekérdezés tetszőleges szinten megfogalmazható
- a magasabb szintűek érdekesebbek, mert
  - \* több adatforrást fedhetnek le
  - \* elvégzik a szükséges konverziókat

```
query OlcsóNagyKerekűAutó
select k.gyártási_szám, k.autó.gyártási_szám
from k: Gyár::Kerék
where k.méret > 4 and k.autó.ár < 5000;
```

- olcsó autókat keresünk, nagy keréssel
- from k: Gyár::Kerék: k a Gyár modell Kerék osztályának példánya
- where k.méret > 4 and k.autó.ár < 5000: a már említett navigációkat használjuk, a kerék legyen nagy, az autója legyen olcsó
- select k.gyártási\_szám, k.autó.gyártási\_szám: a gyártási számokra vagyunk kíváncsiak

# A SILK részletes architektúrája



**Jelmagyarázat:** [Prolog komponens] [Java komponens] [Tervezett Prolog komponens]



## A SILK által támogatott folyamatok

- A modelltárház felépítése
  - \* Létező modellek átvétele információforrásokból és modellezési (CASE) eszközökből
  - \* Modellek *összehasonlítása és összekapcsolása*
  - \* Modellek *konzisztenciaellenőrzése*
  - \* Modellek *egyesítése*
- Integrált rendszer használata és építése
  - \* *Összetett lekérdezések futtatása*
  - \* Egységes, integrált rendszer fejlesztésének támogatása

## Modellek összehasonlítása és összekapcsolása

- Bemenete két modellelemhalmaz (pl. a 23. és a 25. oldal modelljeiből)
- Gráfok hasonlóságát vizsgálja
- Megkeresi az egymáshoz leginkább hasonlókat és összekapcsolja őket
- Általános esetben  $m$  elemet  $n$  elemmel köt össze
- Az összekapcsolt elemekhez leképezést (megfeleltetést vagy absztrakciót) generál

```
abstraction (s: Peugeot::Jármű -> c: Gyár::Autó) {  
    constraint s.sorszám = c.gyártási_szám and s.ár = c.ár;  
};
```

- Pusztán a modellek alapján nem jöhet rá, hogy az árak között konverzióra van szükség
- Nem tudja származtatni a gyártó attribútumot
- Nem tudja, hogy a típus attribútum szerint szűrni kell
- A kézzel javított változat a 28. oldalon

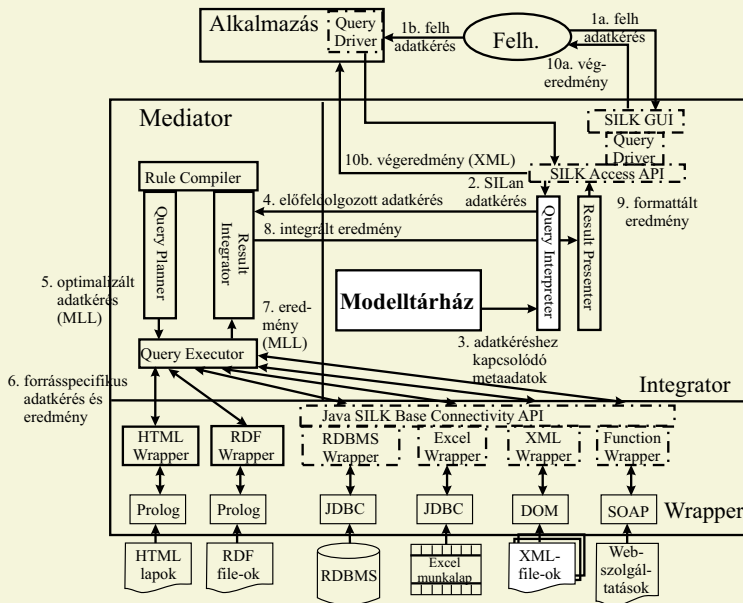
## Modellek ellenőrzése

- Konzisztenciaellenőrzés kizárólag a metainformációk alapján
- Elsősorban modellek összekapcsolása után fontos
- Pl. osztályinvariáns ellenőrzése
- $osztály(x_1, \dots, x_n) \rightarrow inv(x_1, \dots, x_n)$
- Legyen  $Peugeot :: \text{Jármű}$  (értelmetlen) invariánsa  $self.ár < 5$  and  $self.ár > 30$
- 'Jármű' ( $GySz, Típus, Ár$ )  $\rightarrow Ár < 5 \wedge Ár > 30$
- Ha a jobb oldal hamis, akkor az osztály üres
- CLP következtetőket használunk a jobb oldal vizsgálatára
- A CLP(R) következtető a fenti példán egymagában kimutatja, hogy  $Peugeot :: \text{Jármű}$  inkonzisztens
- Általános esetben többféle következtető együttműködésére lehet szükség

## Modellek egyesítése

- Megfeleltetések alapján
  1. egyesített modell és
  2. absztrakciók generálása
- Üzem módok
  - \* unióegyesítés: az összes asszociáció és osztály az összes attribútummal bekerül
  - \* metszetegyesítés: csak a megfeleltetett asszociációk, osztályok, és attribútumok kerülnek be
- Az absztrakciók automatikus generálása nehéz
  - \* lehetetlen, ha az attribútumok között elég „furcsa” reláció van (ez a gyakorlatban nem fordul elő)
  - \* működik, ha a megfeleltetett attribútumok függvényei között egyenlőség van

# Adatelérés összetett modellek alapján



## Az MLL

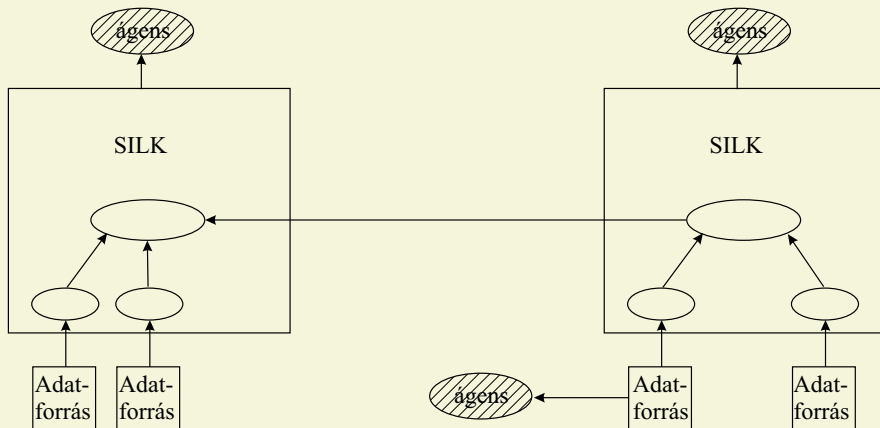
- Egységes logikai formalizmus a modelltárház leírására
- $\forall \bar{X}.(p_0(\bar{X}_0) \wedge \dots \rightarrow \exists \bar{Z}.q_0(\bar{Y}_0) \wedge \dots)$ , ahol
- $\bar{X} = \bigcup_i \bar{X}_i$ ,  $\bar{Y} = \bigcup_i \bar{Y}_i$  és  $\bar{Z} = \bar{Y} \setminus \bar{X}$
- A kvantálás implicit, azaz nem írjuk ki
- A self.kerék.méret  $> 2$  invariáns (23. oldal)
- MLL alakja:

'Autó' (GySz, Gy, Á), 'Autó-Kerék' ('Autó' (GySz, Gy, Á),  
'Kerék' (KGySz, Méret))  
----> Méret  $> 2$

- A Peugeot :: Jármű-ből Gyár :: Autó absztrakciójának (28. oldal)
- MLL alakja:

'Jármű' (GySz, 'Autó', Ár) ---->  
'Autó' (GySz, 'Peugeot', Autó\_ár), Autó\_ár = 1000\*Ár

## A SILK továbbfejlesztése: SILK-rendszerek együttműködése



## A SILK továbbfejlesztése: közvetítés ágensek között

