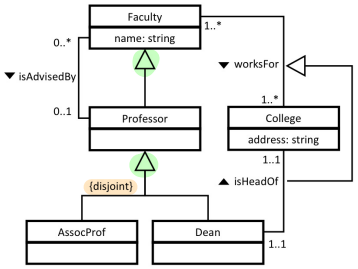
Example of DL-Lite$_A$ TBox

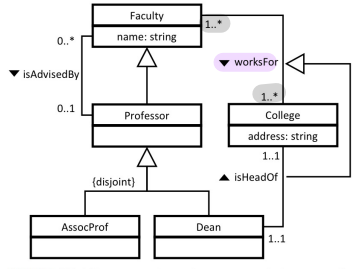


AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).

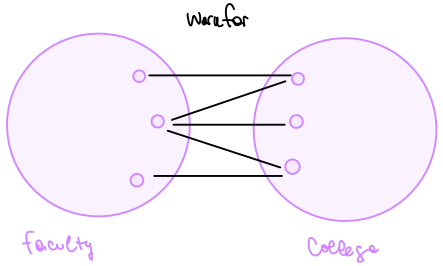
- **Generalizzazione/specializzazione: esiste una superclasse e le proprie sottoclassi (ereditare)**
- **nome associazione e direzione (come leggere) -> viene trattato come un ruolo**
- **nome dell'attributo e il suo tipo primitivo**
- **molteplicità: numero di oggetti che possono essere associati esattamente a un oggetto del lato opposto**

Example of DL-Lite$_A$ TBox



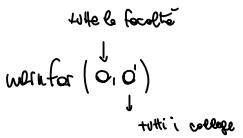
AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).

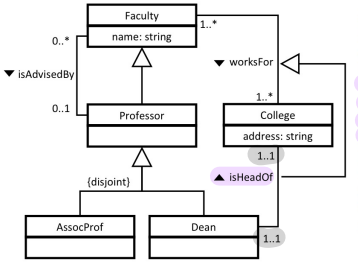


- **tutti gli elementi sono coinvolti del ruolo *workFor***
- ***utilizzo tutte le istanze del concetto faculty nel ruolo***

- **essendo degli insiemi anche se si ripetono le facoltà la tiro fuori solo una volta all'interno dell'associazione**
- **quindi sicuramente abbiamo una coincidenza a livello insiemistico**

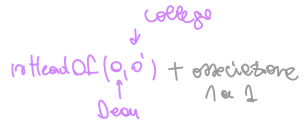


Example of DL-Lite$_A$ TBox



AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

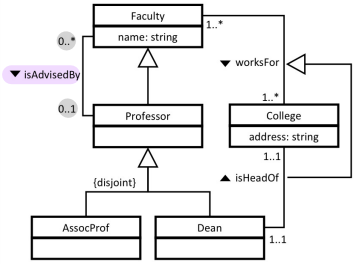
NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).



il ruolo ha una corrispondenza esatta con i concetti

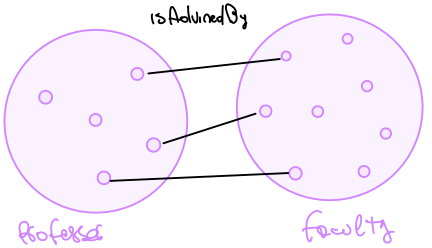
- **facciamo in questo modo perché tutte le tuple in *isAdvisedBy* utilizzando sicuramente un faculty e un professore**
- **l'insieme Faculty possiede tante istanza a cui non é associato alcun professore => tramite il ruolo**
- **lo stesso ragionamento vale per i professori**
- **sono elementi non coinvolti nel ruolo**

Example of DL-Lite$_A$ TBox



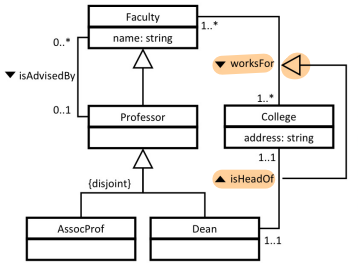
AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).



- **i due insiemi generati dall'interpretazione non cambiano: Professor e faculty**
- **quello che dipende dall'associazione é il ruolo**
- **parto sempre dal ruolo e lo definisco**

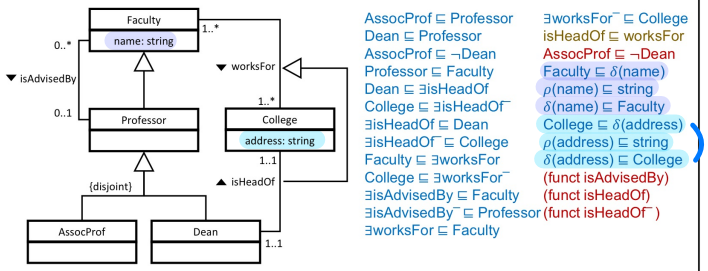
Example of DL-Lite$_A$ TBox



AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).

Example of DL-Lite$_A$ TBox



AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

l'attributo address viene collegato al college

il valore di un attributo è di tipo stringa

gli oggetti di college e attributo address coincidono

NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).

$$Som(T,D) = \{ B \mid q_S(D) \subseteq q_G(B) \text{ for each } \langle q_S, q_G \rangle \in M \}$$

$$Rem(T,D) = \{ B \mid q_S(D) \supseteq q_G(B) \ \forall \langle q_S, q_G \rangle \in M \}$$

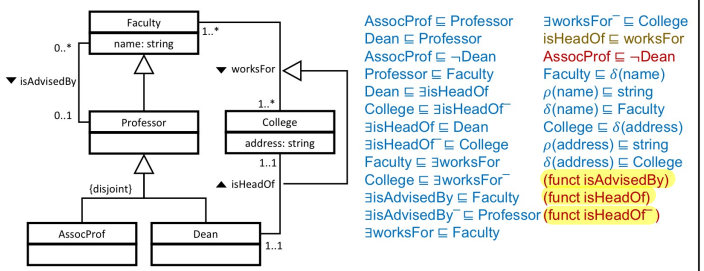
$$\equiv$$

GAV $\quad \langle q_S, g \rangle \quad \forall \bar{x}.\ q_S(\bar{x}) \rightarrow g(\bar{x})$

LAV $\quad \langle s, q_G \rangle \quad \forall \bar{y}.\ s(\bar{y}) \rightarrow q_G(\bar{y})$

$$M(D) = \{ g(\bar{c}) \mid \langle q_S, g \rangle \in M \text{ s.t. } \bar{D} \in q_S(D) \}$$

Example of DL-Lite$_A$ TBox



AssocProf ⊑ Professor	∃worksFor⁻ ⊑ College
Dean ⊑ Professor	isHeadOf ⊑ worksFor
AssocProf ⊑ ¬Dean	AssocProf ⊑ ¬Dean
Professor ⊑ Faculty	Faculty ⊑ δ(name)
Dean ⊑ ∃isHeadOf	ρ(name) ⊑ string
College ⊑ ∃isHeadOf⁻	δ(name) ⊑ Faculty
∃isHeadOf ⊑ Dean	College ⊑ δ(address)
∃isHeadOf⁻ ⊑ College	ρ(address) ⊑ string
Faculty ⊑ ∃worksFor	δ(address) ⊑ College
College ⊑ ∃worksFor⁻	(funct isAdvisedBy)
∃isAdvisedBy ⊑ Faculty	(funct isHeadOf)
∃isAdvisedBy⁻ ⊑ Professor	(funct isHeadOf⁻)
∃worksFor ⊑ Faculty	

NOTE: DL-Lite **cannot** capture **completeness** of a hierarchy. This would require disjunction (OR).

- asserzioni funct sono per ruoli e attributi

isHeadOf:
- Ogni Dean ha esattamente un college (perché é funzionale e obbligatorio)
- Ogni College ha esattamente un Dean (per l'inverso funzionale)

isadvisedBy:
- per ogni faculty c'é al più Professor
- per un professor possono esserci piu faculty => no funzionale

more expressive
1 a 1

ESE 1)

Exercise

Consider the following information integration specification J = ⟨G,M,S⟩, where:

- Global schema **G** = { **Author**(id, name), **Book**(code, title, authorName) }
- Source schema **S** = { **bestSeller**(bookCode, bookTitle, authorId), **award**(authorName, bookTilte, date) }
- Mapping **M** = { m_1: ∀x,y,z,k,w.**bestSeller**(x, y, z) ∧ **award**(k, y, w) → **Book**(x, y, k),
  m_2: ∀x,y,z,k,w.**bestSeller**(x, y, z) ∧ **award**(k, y, w) → **Author**(z, k) }

- Tell if M is a GAV, LAV, or GLAV mapping.
- Provide the formal definition of retrieved global database for a GAV information integration specification w.r.t. a database.
- Compute the retrieved global database for **J** w.r.t. **D**, where **D** is as follows:
  D = { **bestSeller**(1, "Less", 10), **bestSeller**(2, "Il cardellino", 20), **award**("Greer", "Less", 2018), **award**("Tartt", "Il cardellino", 2014), **award**("Desiati", "Spatriati", 2020) }.
- Given the following CQ over G: q = {() | ∃x,y,k,t.Author(x,y) ∧ Book(k,t,y) }, compute cert(q,J,D).
- Write at least two LAV mapping between S and G.

1) GAV cosa specifica GLAV

3) M(D) = { Book(1, "Less", "Greer"),
Book(2, "Il cardellino", "Tartt"),
Author(10, "Greer"),
Author(20, "Tartt") }

4) cert(q, J, D) = q(M(D)) = {()} // vero

5)

∀x,y,z. bestSeller(x,y,z) → ∃h. Book(x,y,h) ∧ Author(z,u)

∀x,y,z. award(x,y,z) → ∃h,w. Book(h,y,x) ∧ Author(w,x)

Exercise

Consider the following TBox T:

T = { book ⊑ product,
e-book ⊑ product,
~~product ⊑ book ⊔ e-book,~~
book ⊑ ∃printedBy,
∃printeBy⁻ ⊑ PrintCompany,
e-book ⊑ ¬book }

Logical programming rules (PIs)

product(z) ← book(z)
product(z) ← e-book(z)

printedBy(z, f(z)) ← book(z)
printed(Company(z) ← printedBy(f(z), z)

- Tell, motivating your answer, whether TBox **T** can be expressed in DL-LiteA. If it cannot, write a TBox **T'** containing only the axioms of **T** that are expressible in DL-LiteA.
- Given the ABox **A** = { product(*p1*), book(b1), e-book(eb1), printedBy(*b2, pc1*) }, tell whether the ontology ⟨**T**,**A**⟩ is satisfiable (consistent), and if so, show an *interpretation* that is a *model* for it.
- Given the following conjunctive query ***q***: *{ (x) | ∃y.product(x) ∧ printedBy(x,y) ∧ PrintCompany(y) }*, compute the perfect rewriting of ***q*** w.r.t. **T'**, where **T'** is the DL-LiteA TBox provided by you above.
- Given the ABox **A** above, compute the certain answers of the query ***q*** over the ontology ⟨**T**,**A**⟩.
- Add to the TBox **T** the DL-Lite assertions needed for formalizing the following statement "*every e-book is distributed by at least one e-book store*" (consider to have the role *distributedBy*).

1) In DL-Lite you can't specialize an atomic concept and also use conjuction for computational purposes.

T' = T / { product ⊑ book ⊔ e-book }

2) ∃ NI ∈ T, N: e-book ⊑ ¬book

q_N = {() | ∃x. e-book(x) ∧ book(x)}

$Eval_{CWA}(SQL(q_N), DB(A)) = \emptyset \Rightarrow$ O = ⟨T,A⟩ soddisfacibile

$I = (\Delta^I, \cdot^I)$ $\Delta^I = \{p_1, b_1, eb_1, b_2, pc_1\}$

$book^I = \{b_1\}$
$e\text{-}book^I = \{eb_1\}$
$product^I = \{p_1, b_1, eb_1\}$
$printedBy^I = \{(b_2, pc_1), (b_1, pc_1)\}$
$PrintCompany^I = \{pc_1\}$

I ⊨ O = ⟨T,A⟩ tutte le assertions sono soddisfatte

3) logical programming rules (PIs)

$T = \{$ book $\sqsubseteq$ product,
e-book $\sqsubseteq$ product,
~~product $\sqsubseteq$ book $\sqcup$ e-book~~,
book $\sqsubseteq \exists$printedBy,
$\exists$printeBy$^- \sqsubseteq$ PrintCompany,
~~e-book $\sqsubseteq \neg$book~~ $\}$

product(z) ← book(z)
product(z) ← e-book(z)
printedBy(z, f(z)) ← book(z)
printCompany(z) ← printedBy(z, z)

$r_{q,T}$ = perfectRef(q, Tp) = $\{(x) \mid \exists y.\ product(x) \wedge printedBy(x,y) \wedge printCompany(y)\} \cup$
$\{(x) \mid \exists y.\ book(x) \wedge printedBy(x,z) \wedge printCompany(z)\} \cup$
$\{(x) \mid \exists y.\ e\text{-}book(x) \wedge printedBy(x,z) \wedge printCompany(z)\} \cup$
$\{(x) \mid \exists y,u.\ product(x) \wedge printedBy(x,z) \wedge printedBy(u,y)\} \cup$
$\{(x) \mid \exists y,u.\ book(x) \wedge printedBy(x,z) \wedge printedBy(u,y)\} \cup$
$\{(x) \mid \exists y,u.\ e\text{-}book(x) \wedge printedBy(x,z) \wedge printedBy(u,y)\} \cup$
$\{(x) \mid \exists y.\ product(x) \wedge printedBy(x,z)\} \cup$
$\{(x) \mid \exists y.\ book(x) \wedge printedBy(x,z)\} \cup$
$\{(x) \mid \exists y.\ e\text{-}book(x) \wedge printedBy(x,z)\} \cup$
$\{(x) \mid \exists y.\ book(x)\}$

4) $cert(q_1, \langle T, A\rangle) = \underset{r_{q,T}}{Eval}(SQL(r_{q,T}), DB(A))$
$= \{(b1)\}$

5) e-book $\sqsubseteq \exists$distributed
$\exists$distributed $\sqsubseteq$ e-book
$\exists$distributed$^- \sqsubseteq$ e-store

Esame 2)

Exercise
Consider the following information integration specification **J** = ⟨G,M,S⟩, where:

- Global schema **G** = { **Researcher**(rid, name), **Paper**(pid, pTitle, authorName) }
- Source schema **S** = { **conferencePaper**(pid, pTitle, rid), **bestPaper**(authorName, pTilte, year) }
- Mapping **M** = { m_1: ∀x,y,z.**conferencePaper**(x, y, z) → ∃w.**Paper**(x, y, w) ∧ **Researcher**(z, w)
  m_2: ∀x,y,z.**bestPaper**(x, y, z) → ∃v.**Paper**(v, y, x) }
  - Tell, motivating your answer, if **M** is a GAV, LAV, or GLAV mapping.
  - Provide the formal definition of universal solution for an information integration specification w.r.t to a source database **D**.
  - Compute a global instance **K** that is an universal solution for **J** w.r.t. **D**, where **D** is as follows:
    D = { **conferencePaper**(1,"title1",10), **conferencePaper**(2,"title2",20),
    bestPaper("Bob","title1",2010),
    bestPaper("Sam", "title3", 2012), **bestPaper**("Bill", "title4", 2021) }.
  - Given the following CQ over **G**: **q** = {(y,z) | ∃x.**Paper**(x,y,z)}, compute, motivating your answer, cert(**q**,**J**,**D**).
  - Write at least two (possibly new) GAV mapping between **S** and **G**.

1) LAV così specchio di GLAV

3) $n = ch(M,D) = \{$ Paper(1, "title1", z_1), Researcher(10, z_1),
Paper(2, "title2", z_2), Researcher(20, z_2),
Paper(z_3, "title1", "Bob"),
Paper(z_4, "title3", "Sam"),
Paper(z_5, "title4", "Bill") $\}$

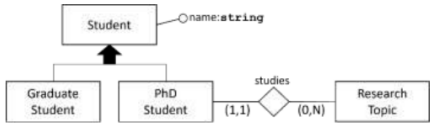
dove z_i sono labelled nulls

4) $cert(q,J,D) = q^{\downarrow}(ch(M,D)) = \{$ ("title1", "Bob"),
("title3", "Sam"),
("title4", Bill) $\}$

Restituire variabile nella tupla

5) $\forall x,y,z.\ conferenceP(x,y,u) \wedge bestP(t,y,w) \rightarrow Paper(x,y,z)$
$\exists u,w.$

Exercise
Consider the following Entity-Relationship (ER) diagram modelling a portion of the university domain:



- Provide an equivalent formalization with a TBox **T** in Description Logics of the represented domain.
- Tell, motivating your answer, whether the domain can be formalized through a TBox **T'** in DL-LiteA or not. If it cannot, express in DL-LiteA only the portion of the domain that can be captured.
- Given the ABox **A** = { PhdStudent(sam), Student(bill), studies(bill, OBDM) }, tell whether the ontology ⟨**T**,**A**⟩ is satisfiable (consistent), and if so, show an *interpretation* that is a *model* for it.
- Given the following conjunctive query ***q***: *{ (x) | ∃y.Student(x) ∧ studies(x,y) }*, compute the perfect rewriting of ***q*** w.r.t. **T'**, where **T'** is the DL-LiteA TBox provided by you above.
- Let **A** be the ABox above, compute the certain answers of the query ***q*** over the ontology ⟨**T'**,**A**⟩.

1) $T = \{$ gs ⊔ phds ⊑ Student
Student ⊑ gs ⊔ phds
phds ⊑ ∃studies
∃studies⁻ ⊑ phds
∃studies ⊑ research
∃(name) ⊑ Student
Student ⊑ δ(name)
ρ(name) ⊑ xsd:String
(funct studies)
(funct name) $\}$

2) DL-Lite non cattura le generalizzazioni constraints

$T' = T \setminus \{$ Student ⊑ gs ⊔ phds $\}$
$\cup \{$ gs ⊑ Student $\}$
$\cup \{$ phds ⊑ Student $\}$

aggiungere anche:
GS ⊑ ¬ phds

3) soddisfacibile:

$\exists f \in T,\ f_1 : (funct\ studies)$
$q_{f_1} = \{() \mid \exists o,o',o^2.\ studies(o,o') \wedge studies(o,o^2) \rightarrow o' \neq o^2\}$
$Eval_{cwA}(sql(q_{f_1}), DB(A)) = \emptyset$

$\exists f \in T,\ f_2 : (funct\ name) \rightarrow q_{f_2} = \{() \mid$
$q_{f_2} = \{() \mid \exists o,v',v^2.\ name(o,v') \wedge name(o,v^2) \rightarrow v' \neq v^2\}$
$Eval_{cwA}(sql(q_{f_2}), DB(A)) = \emptyset$

$\Rightarrow \exists I : I \models O = \langle T, A \rangle$

da aggiungere qua.

considering $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with $\Delta^{\mathcal{I}} = \{ sam, lice, obda \}$

$student^{\mathcal{I}} = \{ lice, sam \}$

$gs^{\mathcal{I}} = \{ \}$ $\Rightarrow \mathcal{I} \not\models \mathcal{O}$

$phdS^{\mathcal{I}} = \{ sam, lice \}$

$research^{\mathcal{I}} = \{ obds \}$

$name^{\mathcal{I}} = \{ (lice, 'lice'), (sam, 'sam') \}$

$studies^{\mathcal{I}} = \{ (lice, obds), (sam, obds) \}$

q) $\mathcal{D}$:

$student(x) \leftarrow gs(x)$

$student(x) \leftarrow phdS(x)$

$studies(x, f(x)) \leftarrow phdS(x)$

$phdS(x_1) \leftarrow studies(x_1, x_2)$

$research(x_2) \leftarrow studies(x_1, x_2)$

$PerfectRew(q, T_P) = r_{q,T} = \{ (x) \mid \exists y. \; student(x) \wedge studies(x, y) \} \cup$

$= \{ (x) \mid \exists y. \; gs(x) \wedge studies(x, y) \} \cup$

$= \{ (x) \mid \exists y. \; phdS(x) \wedge studies(x, y) \} \cup$

$= \{ (x) \mid \exists y_1, u. \; student(x) \wedge phdS(x) \} \cup$

$= \{ (x) \mid gs(x) \wedge phdS(x) \} \cup$

$= \{ (x) \mid phdS(x) \wedge phdS(x) \} = \{ (x) \mid phdS(x) \}$

5) $cert(q, \langle T, A \rangle) = Eval_{CWA}(SQL(r_{q,T}), DB(A)) = \{ (sam), (lice) \}$

Est 3

Exercise

Consider the following information integration specification **J** = ⟨G,M,S⟩, where:

- Global schema **G** = { **Patient**(pId, pName, pAge), **Hospitalization**(pId, doctor, date) }
- Source schema **S** = { **MediacalRecord**(pId, pName, hospDate, doctoId) }
- Mapping **M** = { m_1: ∀x,y,z,k.**MediacalRecord**(x, y, z, k) → ∃w.**Patient**(x, y, w),
  m_2: ∀x,y,z.**MediacalRecord**(x, y, z, k) → **Hospitalization**(x, k, z)}
  - Tell, motivating your answer, if **M** is a GAV, LAV, or GLAV mapping.
  - Describe the Naïve Chase Algorithm
  - Compute a **ch(M,D)**, where **M** is the above mapping and **D** is as follows: **D** = { **MediacalRecord**(10, "Rossi", 06/08/2010, 35), **MediacalRecord**(20, "Bianchi", 09/11/2012, 35), **MediacalRecord**(30, "Verdi", 15/01/2013, 45) }.
  - Given the following CQ over **G**: **q** = {(y,x,w) | ∃z,k. **Hospitalization**(x,y,z) ∧ **Hospitalization** (w,y,k)}, compute the perfect UCQ rewriting of **q** w.r.t. **J**.
  - Compute cert(**q**,**J**,**D**).

1) LAV special case of GLAV

3) $k = ch(M,D) = \{$ Patient(10, "Rossi", z_1), Patient(20, "Bianchi", z_2), Patient(30, "Verdi", z_3), Hosp.(10, 35, 06/08/2010), Hosp.(20, 35, 09/11/2012), Hosp.(30, 45, 15/01/2013) $\}$

z_i: Jersl labeled nulls

4) $Rew_{JCQ}^{(q,T)} = q_M = \{(y,x,w) \mid \exists r_1, r_2, z, k.\ MediaR(x, r_1, z, y) \wedge MediaR(w, r_2, k, y)\}$

5) $cert(q,J,D) = q^{\downarrow}(ch(M,D)) = q_M(D)$
$= \{(35,10,10), (35,20,20), (45,30,30), (35,10,20), (35,20,10)\}$

considero i tabelle di MediaR con le stesse tuple

Exercise

Consider the following domain description about electronic devices:

"We know that both phones and computers are devices, and that a device that is both a phone and a computer is a smartphone. As we also know, a smartphone is both a phone and a computer. Furthermore, we know that each phone has at least one number associated with it".

- Provide a Description Logic TBox **T** that formalizes all the aspects in the domain description.
- Tell, giving motivation for the answer, whether the domain can be formalized through a TBox **T'** in DL-LiteA or not. If it cannot, express in DL-LiteA only the portion of the domain that can be captured.
- Given the ABox **A** = { Phone(d1), Computer(d1) }, tell whether the ontology ⟨**T**,**A**⟩ is satisfiable (consistent), and if so, show an *interpretation* that is a *model* for it.
- Given the following conjunctive query ***q***: *{ (x) | ∃y.Device(x) ∧ hasNumber(x,y) }*, compute the perfect rewriting of ***q*** w.r.t. **T'**, where **T'** is the DL-LiteA TBox provided by you above.
- Let **A'** be the ABox { Smatphone(**sm**), Computer(**co**), hasNumber(**de**, **num**) }, compute the certain answers of the query ***q*** over the ontology ⟨**T'**,**A'**⟩.

1) $T = \{$ phone ⊔ pc ⊑ devices,
~~phone ⊓ pc ⊑ smarth,~~
smarth ⊑ phone ⊓ pc,
phone ⊑ ∃hasNumber,
∃hasNumber ⊑ phone,
∃hasNumber⁻ ⊑ Number,
phone ⊑ ¬pc,
(funct hasNumber) $\}$

2) No.
phone ⊑ devices
pc ⊑ devices
Smarth ⊑ phone
Smarth ⊑ pc

3)

Advanced Data Management & Laboratory

Date: 13/02/2024 — **9 CFU**

Exercise 1

Describe the procedure for answering queries over ontologies through query rewriting. Moreover, given a query language **Q** and an ontology language **L**, provide the formal definition of **Q**-rewritability.

Exercise 2

Consider the following information integration specification **J** = ⟨G,M,S⟩, where:

- Global schema **G** = { **Resources**(eId, surname), **Projects**(name, budget), **Costs**(project, resource, cost) }
- Source schema **S** = { **employees** (eId, surname, salary), **activities**(aId, name, budget), **worksIn**(eId,aId)}
- Mapping

  M = { m_1: ∀x,y,w ∃z,k,t. **employees**(x, y, z) ∧ **worksIn**(x, k) ∧ **activities**(k, w, t) → **Resources**(x, y, w),
  m_2: ∀x,z,w ∃y,t. **employees**(x, y, z) ∧ **worksIn**(x, k) ∧ **activities**(k, w, t) → Costs(w, x, z),
  m_3: ∀y,z ∃x. **activities**(x, y, z) → **Projects**(y, z) }

a) Tell whether **M** is a GAV, LAV, or GLAV mapping.
b) Give the formal definition of Retrieved Global Database and describe its properties with respect to sem(**J**,**D**), where **D** is a database for **S**.
c) Compute the perfect rewriting w.r.t. **J** of the query **q** = {(y,z,k) | ∃x.**Resources**(x, y, z) ∧ **Projects**(z, k)}
d) Compute cert(**q**,**J**,**D**), where **D** is as follows: **D** = {**employees**(1, "white", 30k), **employees**(2, "brown", 40k), **employees**(3, "smith", 50k), **activities**(p1, "prj1", 100k), **activities**(p2, "prj2", 200k), **activities**(p3, "prj3", 300k), **worksIn**(1,p1), **worksIn**(2,p2) } .
e) Write at least one (possibly new) LAV mapping between **S** and **G**.

Exercise 3

Consider the following DL-Lite TBox T:

T = { car ⊔ bike ⊑ vehicle,
~~vehicle ⊑ car ⊔ bike,~~
car ⊑ ∃ownerOf⁻,
∃ownerOf ⊑ person,
∃ownerOf⁻ ⊑ car,
vehicle ⊑ δ(numberOfWheels),
car ⊑ ¬ bike,
(funct ∃ownerOf⁻) }

a) Tell, motivating your answer, whether TBox **T** can be expressed in DL-LiteA. If it cannot, write a TBox **T'** containing only the axioms of **T** that are expressible in DL-LiteA.
b) Given the ABox **A** = { car(*c1*), vehicle(v), ownerOf(p, v2) }, tell whether the ontology ⟨**T**,**A**⟩ is satisfiable (consistent), and if so, show an *interpretation* that is a *model* for it.
c) Given the following conjunctive query ***q***: *{ (x) | ∃y.car(x) ∧ ownerOf(y,x) }*, apply the PerfectRef algorithm to the query ***q*** and TBox **T'**.
d) Compute the certain answers of the query ***q*** over the ontology ⟨**T'**, **A**⟩.
e) Add to the TBox **T** the DL-Lite assertions needed for formalizing the following statement "every bike has exactly a brand" (suppose to have the hasBrand attribute).

(separate sheet)

Exercise 4

Describe the role of l-diversity in data anonymization using k-anonymity.

Exercise 5

Classify the approaches for multidimensional indexing.

ESE 2)

1) GAV ≡ GLAV mapping because respects the following def.

3) $ren_L^{(q,J)} = Unq_{q,m} = \{(x,z,w) \mid \exists t,r,s,y.\ employees(x,y,t) \wedge worksIn(x,r) \wedge activities(r,t,s) \wedge activities(w,t,u)\}$

4) $ren_L^{(q,J)}(D) = Unq_{q,m}(D) = cert(q,J,D)$
$= \{("white", "prj1", 100k), ("brown", "prj2", 200k)\}$

5) $\forall x,y,z.\ employees(x,y,z) \longrightarrow \exists \alpha,\beta.\ Resources(x,y,\alpha) \wedge Costs(\alpha,\beta,z)$

ESE 3)

1) T cannot be expressed in DL-LiteA
- an atomic concept cannot be specialized
- disjunctions are not allowed → computational complexity

T = { car ⊑ vehicle,
line ⊑ vehicle,
car ⊑ ∃ownerOf,
∃ownerOf ⊑ person,
∃ownerOf⁻ ⊑ car,
vehicle ⊑ δ(numOfWheels),
car ⊑ ¬ line,
(funct ∃ownerOf) }

2)

Theorem: O satisfiability depends only on NI and functionality assertions.

$\nexists N \in \mathcal{T},\ N: q_N = \{() \mid \exists x.\ car(x) \wedge line(x)\}$

$ans_{CWA}(SQL(q_N), DB(A)) = \emptyset \quad \langle \mathcal{T}, A \rangle \not\models q_N$

$\nexists F \in \mathcal{T},\ f: q_f = \{() \mid \exists o', o'', o.\ ownerOf(o', o) \wedge ownerOf(o'', o) \wedge o' \neq o''\}$

$ans_{CWA}(q_f, DB(A)) = \emptyset \quad A \not\models q_f$

⇒ Don't exist objects violating the ontology assertions ⇒ $\exists \mathcal{I} \models \mathcal{O} - \langle \mathcal{T}, A \rangle$

$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}) \quad \Delta^{\mathcal{I}} = \{c_1, v_1, p_1, v_2\}$

$vehicle^{\mathcal{I}} = \{c_1, v_1, v_2\}$
$car^{\mathcal{I}} = \{c_1, v_2\}$
$line^{\mathcal{I}} = \{\}$
$ownerof^{\mathcal{I}} = \{(p, v_2), (p, c_1)\}$
$numOfWheels^{\mathcal{I}} = \{(c_1, \text{"4"}), (v_1, \text{"3"}), (v_2, \text{"3"})\}$
$person^{\mathcal{I}} = \{p\}$

3) $\mathcal{T} = \{ car \sqsubseteq vehicle,$
$line \sqsubseteq vehicle,$
$car \sqsubseteq \exists ownerOf,$
$\exists ownerOf \sqsubseteq person,$
$\exists ownerOf^- \sqsubseteq car,$
~~$vehicle \sqsubseteq \delta(numOfWheels),$~~
~~$car \sqsubseteq \neg line,$~~
~~$(funct\ ownerOf^-)\}$

$vehicle(z) \leftarrow car(z)$
$vehicle(z) \leftarrow line(z)$
$ownerof(f(z), z) \leftarrow car(z)$
$person(z_1) \leftarrow ownerof(z_2, z_1)$
$car(z_1) \leftarrow ownerof(z_2, z_1)$

$perfectRef(q, \mathcal{T}) = r_{q,\mathcal{T}} = \{(x) \mid \exists y.\ car(x) \wedge ownerOf(y, x)\} \cup$
$= \{(x) \mid \exists y, u.\ ownerOf(u, x) \wedge ownerOf(y, x)\} \cup$
$= \{(x) \mid car(x) \wedge car(x)\} \cup$
$= \{(x) \mid \exists y.\ ownerOf(y, x)\} \cup$
$= \{(x) \mid car(x)\}$

4) $cert(q, \langle \mathcal{T}, A \rangle) = ans_{CWA}(SQL(r_{q,\mathcal{T}}), DB(A))$
$= \{(c_1), (v_2)\}$

5) $line \sqsubseteq \exists hasBrand$
$\exists hasBrand \sqsubseteq line$?
$(funct\ hasBrand)$

In realtà non esiste un concetto nullo del brand

Esercizio

Exercise
Consider the following information integration specification J = ⟨G,M,S⟩, where:

- Global schema **G** = { **Author**(id, name), **Book**(code, title, authorName) }
- Source schema **S** = { **bestSeller**(bookCode, bookTitle, authorId), **award**(authorName, bookTilte, date) }
- Mapping **M** = { m_1: ∀x,y,z,k,w.**bestSeller**(x, y, z) ∧ **award**(k, y, w) → **Book**(x, y, k),
  m_2: ∀x,y,z,k,w.**bestSeller**(x, y, z) ∧ **award**(k, y, w) → **Author**(z, k) }

- Tell if M is a GAV, LAV, or GLAV mapping.
- Provide the formal definition of retrieved global database for a GAV information integration specification w.r.t. a database.
- Compute the retrieved global database for **J** w.r.t. **D**, where **D** is as follows:
  D = { **bestSeller**(1, "Less", 10), **bestSeller**(2, "Il cardellino", 20), **award**("Greer", "Less", 2018), **award**("Tartt", "Il cardellino", 2014), **award**("Desiati", "Spatriati", 2020) }.
- Given the following CQ over G: q = {() | ∃x,y,k,t.Author(x,y) ∧ Book(k,t,y) }, compute cert(q,J,D).
- Write at least two LAV mapping between S and G.

Risoluzione

- GAV and a special case of GLAV
- Compute the retrieved global database for J w.r.t. $\mathbf{D}$, where $\mathbf{D}$ is as follows:
  - Prendo tutte le mapping asssertions m_1, m_2 e le calcolo sull'istanza $\mathbf{D}$
  - Il risultato verrá aggiunto come fatto a $\mathbf{M(D)}$
  - Fisso un fatto in $\mathbf{D}$ e lo faccio scorrere su tutti gli altri fino a quando non viene rispettato il mapping su S e creo il nuovo fatto in G. Un volta terminato passo al prossimo fatto successivo in $\mathbf{D}$ (devo scorrere tutto fra tutti).

$$M(D) = \{Book(1, "Less", "Greer"), \quad Book(2, "Il\ cardellino", "Tartt")$$
$$Author(10, "Greer"), \quad Author(20, "Tartt")\}$$

- Compute the $\mathbf{cert}(q, J, \mathbf{D})$:

$$\mathbf{cert}(q, J, \mathbf{D}) = \{\bar{c} \mid \bar{c} \in q(\mathbf{B})\ \forall B \in sem(J, \mathbf{D})\}$$

$$\mathbf{cert}(q, J, \mathbf{D}) = \{()\}$$

- Write at least two LAV mapping between S and G:

$$m_1 : \forall x, y, z.\ \text{bestSeller}(x, y, z) \rightarrow \exists k.\ \text{Book}(x, y, k) \wedge \text{Author}(z, k)$$

$$m_2 : \forall x, y, z.\ \text{award}(x, y, z) \rightarrow \exists k.\ \exists w.\ \text{Book}(k, y, x) \wedge \text{Author}(w, x)$$

Exercise
Consider the following DL-Lite TBox T:
T = { doctor ⊑ person,
doctor ⊑ ∃worksFor,
∃worksFor⁻ ⊑ hospital,
medStudent ⊑ doctor,
medStudent ⊑ ¬∃worksFor }

- Tell whether the TBox T is satisfiable, and if so, show a model for T.
- Given the ABox A = {medStudent(bob)}, tell whether the ontology ⟨T,A⟩ is satisfiable (consistent), and if so, show an interpretation that is a model for it.
- Given the following conjunctive query ***q***: *{ (x) | ∃y.person(x) ∧ worksFor(x,y) ∧ hospital(y) }*, compute the perfect rewriting of ***q*** w.r.t. T.
- Given the ABox A' = { doctor(sam), person(bill), worksFor(bill, H3) }, compute the certain answers of the query ***q*** over the ontology ⟨T,A'⟩.
- Add to the TBox T the DL-Lite assertions needed for formalizing the following statement "every doctor has a specialization".

Risoluzione

- Determine whether the **TBox** $\mathcal{T}$ is satisfiable:

  $\mathcal{T}$ is satisfiable, if it admits at least one model such that $\mathbf{I} \models \mathcal{T}$.

$$\mathbf{I} = (\Delta^I, \cdot^I) \quad \Delta^I - \{Alice, H1\}$$

$person^I = \{Alice\}$	
$doctor^I = \{Alice\}$	1 okay
$worksFor^I = \{(Alice, H1)\}$	2 okay
$hospital^I = \{H1\}$	3 okay
$medStudent^I = \varnothing$	4, 5 okay (in modo banale)

1)	$doctor \sqsubseteq person$	$doctor^I \subseteq person^I$
2)	$doctor \sqsubseteq \exists worksFor$	$doctor^I \subseteq \{o \mid \exists o'.\ (o, o') \in worksFor^I\}$
3)	$\exists worksFor^- \sqsubseteq hospital$	$\{o' \mid \exists o.\ (o, o') \in worksFor^I\} \subseteq hospital^I$
4)	$medStudent \sqsubseteq doctor$	$medStudent^I \subseteq doctor^I$
5)	$medStudent \sqsubseteq \neg\exists worksFor$	$medStudent^I \cap \{o \mid \exists o'.\ (o, o') \in worksFor^I\} = \varnothing$

  $\mathbf{I}$ is a model for $\mathcal{T}$ so $\mathcal{T}$ is satisfable.

- **Is the ontology satisfiable?**

  We are given the ABox:

$$\mathcal{A} = \{medStudent(bob)\} \Rightarrow bob^I \in medStudent^I$$

  Since $bob^I \in medStudent^I$, by (4) we also have $bob^I \in doctor^I$. Then (2) implies $bob^I \in \exists worksFor$. This contradicts axiom (5), which requires that $bob^I \notin \exists worksFor$.

Conclusion:

$$\langle \mathcal{T}, \mathcal{A} \rangle \text{ is } \textbf{unsatisfiable} \quad \text{(i.e., it admits no model).}$$

There is a contradiction between axioms (2), (4), and (5) when applied to the individual `bob`.

- Compute the perfect rewriting of the query **q:**

$doctor \sqsubseteq person$	$person(z) \leftarrow doctor(z)$
$doctor \sqsubseteq \exists worksFor$	$worksFor(z, f(z)) \leftarrow doctor(z)$
$\exists worksFor^{-} \sqsubseteq hospital$	$hospital(z) \leftarrow worksFor(f(z), z)$
$medStudent \sqsubseteq doctor$	$doctor(z) \leftarrow medStudent(z)$

We apply the perfect rewriting algorithm:

$$\begin{aligned}\textbf{PerfctRef(q, T)} = r_{q,T} = \{(x) \mid \exists y.\, person(x) \wedge worksFor(x,y) \wedge hospital(y)\} \\ \vee \{(x) \mid \exists y, k.\, person(x) \wedge worksFor(x,y) \wedge worksFor(k,y)\} \\ \vee \{(x) \mid \exists y.\, person(x) \wedge worksFor(x,y)\} \\ \vee \{(x) \mid \exists y.\, doctor(x) \wedge worksFor(x,y)\} \\ \vee \{(x) \mid \ doctor(x) \wedge doctor(x)\} = \{(x) \mid \ doctor(x)\}\end{aligned}$$

NB: per applicare la riduzione, il join viene fatto su una var esistenziale e ne introduco un'altra con l'unificazione.

- Since the q_N is not applicable, we can say that the ontology is satisfable and the result is the following:

$$\textbf{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \textbf{Eval}_{CWA}(r_{q,T}, \textbf{DB}(\mathcal{A})) = \{(Sam), (bill)\}$$

- Add to the **TBox:**

every doctor has a specialization

$doctor \sqsubseteq \delta(specilization)$	ogni dottore é un dottore specializzato
$\rho(specilization) \sqsubseteq xsd : string$	la specializzazione é espressa in stringa
$\delta(specilization) \sqsubseteq doctor$	i specializzati sono dottori

8.1 Perplessitá

Conjunctive Mapping – Examples (1)

- Conjunctive **LAV** mapping

  $\forall x, y\,.\,Tab1(x,y) \rightarrow \exists z.\, Person(x) \wedge Parent(y) \wedge hasFather(x,z)$

- Conjunctive **GAV** mapping

  $\forall x\,.\,\forall y.\,\exists z.\, Tab1(x,y) \wedge Tab2(y,z) \rightarrow hasFather(x,y)$

- Conjunctive **GLAV** mapping

  $\forall x\,.\,\forall y.\,\exists z.\, Tab1(x,y) \wedge Tab2(y,z) \rightarrow \exists v.\, hasFather(x,y) \wedge hasFather(y,v)$

Checking violations of negative inclusions

For each NI N in $\mathcal{T}$ we compute a boolean CQ q_N according to the following rules:

N		q_N
$A_1 \sqsubseteq \neg A_2$	$\Rightarrow$	$\{(\ ) \mid \exists x.\, A_1(x) \wedge A_2(x)\}$
$\exists P \sqsubseteq \neg A$ or $A \sqsubseteq \neg \exists P$	$\Rightarrow$	$\{(\ ) \mid \exists x, y.\, P(x,y) \wedge A(x)\}$
$\exists P^{-} \sqsubseteq \neg A$ or $A \sqsubseteq \neg \exists P^{-}$	$\Rightarrow$	$\{(\ ) \mid \exists x, y.\, P(x,y) \wedge A(y)\}$
$\exists P_1 \sqsubseteq \neg \exists P_2$	$\Rightarrow$	$\{(\ ) \mid \exists x, y, z.\, P_1(x,y) \wedge P_2(x,z)\}$
$\exists P_1 \sqsubseteq \neg \exists P^{-}_2$	$\Rightarrow$	$\{(\ ) \mid \exists x, y, z.\, P_1(x,y) \wedge P_2(z,x)\}$
$\exists P^{-}_1 \sqsubseteq \neg \exists P_2$	$\Rightarrow$	$\{(\ ) \mid \exists x, y, z.\, P_1(x,y) \wedge P_2(y,z)\}$
$\exists P^{-}_1 \sqsubseteq \neg \exists P^{-}_2$	$\Rightarrow$	$\{(\ ) \mid \exists x, y, z.\, P_1(x,y) \wedge P_2(z,y)\}$
$P_1 \sqsubseteq \neg P_2$ *or* $P^{-}_1 \sqsubseteq \neg P^{-}_2$	$\Rightarrow$	$\{(\ ) \mid \exists x, y.\, P_1(x,y) \wedge P_2(x,y)\}$
$P^{-}_1 \sqsubseteq \neg P_2$ *or* $P_1 \sqsubseteq \neg P^{-}_2$	$\Rightarrow$	$\{(\ ) \mid \exists x, y.\, P_1(x,y) \wedge P_2(y,x)\}$

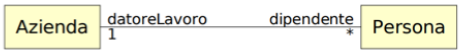
Checking violations of functionality assertions

For each functionality assertions F in $\mathcal{T}$ we compute a Boolean FOL query q_F according to the following rules:

functionality assertion		q_F
(*funct P*)	$\Rightarrow$	$\{(\ ) \mid \exists x, y, z.\, P_1(x,y) \wedge P_2(x,z) \wedge y \neq z\}$
(*funct P*$^{-}$)	$\Rightarrow$	$\{(\ ) \mid \exists x, y, z.\, P_1(y,x) \wedge P_2(z,x) \wedge y \neq z\}$

Le associazioni

- Molteplicità:



Un'Azienda impiega molte Persone
Una Persona lavora per un'unica Azienda

Esempio di vincolo di cardinalità

- Ad ogni impiegato sono assegnati da 1 a 5 incarichi
- Ogni incarico è assegnato ad al più 50 impiegati

Notes about the skolem terms:

- Every **PI** is a logical programming rule **(only concepts and roles inclusions)**.
- Skolem terms are used when the TBox contains an existential quantification on the right-hand side of a concept inclusion
- Skolem term that stands in for the anonymous individua
- Skolem terms are only used to satisfy existential quantifiers over roles, not concepts directly.
- You apply Skolem terms in rules where a concept implies the existence of a role

$$Professor \sqsubseteq \exists teaches$$

$$teaches(z, f(z)) \leftarrow Professor(z)$$

- You do not apply Skolem terms when reasoning about a concept inclusion like

$$\exists teaches^{-} \sqsubseteq Course$$

$$Course(z_1) \leftarrow teaches(z_2, z_1)$$

When can't you apply rules with Skolem terms? Even if the rule uses a Skolem term, you can't always apply it during rewriting. You **can't** apply a rule when:

- The Skolem term can't be unified with a costant
- The skolem term can't be unified with a distiguished variable (output variable)
- The skolem term can't be unified with a join variable

- The disjuction in not allowed in D-Lite in fact can be reformulated, but the conjuction is permitted since is in $\mathcal{AL}$

$$B \sqsubseteq C_1 \sqcap C_2 \qquad C_1 \sqcup C_2 \sqsubseteq B$$
$$B \sqsubseteq C_1 \qquad C_1 \sqsubseteq B$$
$$B \sqsubseteq C_2 \qquad C_2 \sqsubseteq B$$

  To preserve DL-Lite's good computational properties, the disjunction in the original TBox must be removed or reformulated using only allowed constructs and also.

- Captures all the basic constructs of UML Class Diagrams and of the ER Model except covering **constraints in generalizations** $\Rightarrow$ atomic concepts can't be specialized

---

Generalizzazioni/specializzazioni

- si riempie il corpo della freccia freccia se è totale (altrimenti parziale)
- si riempie la punta della freccia nel caso sia esclusiva (non esclusiva)
- **totale:** unione degli insiemi delle istanze delle entità figlie dà l'insieme delle istanze dell'entità genitore
- **esclusiva:** l'intersezione degli insiemi delle istanze delle entità figlie dà l'insieme vuoto

---

- La maximally-sound rewriting deve avere al piú n distinti atomi
- **NB:** la maximally-sound CQ-rewriting della query q_G deve contenere $\overline{x}$ come risposta della query.

The Perfect Rewriting Algorithm

Input: An I.I.S. **J** = ⟨**G**, **M**, **S**⟩ where M is a Conjunctive LAV mapping, and a CQ $q_g = \{\bar{x} \mid \varphi(\bar{x})\}$ where φ has **n** conjuncts.
Output: A UCQ Q over **S**

$Q \leftarrow \emptyset$
For each CQ $q' = \{\bar{x} \mid \psi(\bar{x})\}$ **over S with at most n conjuncts**
  If q' **is a sound rewriting of** q_g**, i.e., if** $exp^{(q',M)} \sqsubseteq q_g$
    $Q \leftarrow Q \cup \{q'\}$
  End If;
End For;
return Q**;**

9 Domande Savo

- **Let $\langle J, D\rangle$ be information integration system, where $J = \langle G, M, S\rangle$. Provide the formal definition of when a global database B satisfies M w.r.t. D in case of sound, complete, and exact mappings semantics.**

  Given an $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ and $\mathbf{D}$ is a source database for S:

  - **Sound-mapping semantics:** the sources contain *partial* but *correct* information regarding the global schema:

    $$\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid q_S(\mathbf{D}) \subseteq q_G(\mathbf{B}) \text{ for each } \langle q_S, q_G\rangle \in M\}.$$

  - **Complete-mapping semantics:** the sources contain *complete* but possibly *incorrect* information:

    $$\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid q_S(\mathbf{D}) \supseteq q_G(\mathbf{B}) \text{ for each } \langle q_S, q_G\rangle \in M\}.$$

  - **Exact-mapping semantics:** the sources contain *exact* information regarding the global schema:

    $$\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid q_S(\mathbf{D}) \equiv q_G(\mathbf{B}) \text{ for each } \langle q_S, q_G\rangle \in M\}.$$

  Extact and complete mapping are the most reasonable seman- tics for IIS. However, complete and exact mappings can exhibit very counterintuitive behaviour and generate inconsistent systems (with no semantics).

- **Tell if M is a GAV, LAV, or GLAV mapping.**

  This is a **GAV** mapping **M** which means is composed by a finite set of **GAV** mapping assertions of this form:

  $$\forall m = \langle q_s, g\rangle \in \mathbf{M}$$

  where q_s is a FOL query over the source schema S and $g \in G$ is a single atom of the global schema. The logical formalization of GAV mapping assertions are the following:

  $$\forall \overline{x}.\, \varphi_S(\overline{x}) \rightarrow g(\overline{x})$$

  We can also consider this mapping, in a more genric way as **GLAV** mapping **M**, since **GAV** $\subseteq$ **GLAV**.

- **Provide the formal definition of retrieved global database for a GAV information integration specification w.r.t. a database.**

  Consider an information integration system $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ is a **GAV** information integration specification, and $\mathbf{D}$ is a source database for S. We call the **retrieved global database** for J with respect to $\mathbf{D}$, denoted by $\mathbf{M(D)}$, the global database obtained by applying the queries in all mapping assertions in **M** and transferring to the elements of G the corresponding tuples retrieved from $\mathbf{D}$. Formally:

  $$\mathbf{M(D)} = \{g(\bar{c}) \mid \langle q_S, g\rangle \in \mathbf{M} \text{ and } \bar{c} \in q_S(\mathbf{D})\}.$$

  Note that, since mappings are of type **GAV**, the tuples to be transferred to the global schema G are always definite, i.e., they do not contain existentially quantified elements. Also it is easy to prove the following properties:

  - **Property 1:** $\mathbf{M(D)} \in \mathbf{sem}(J, \mathbf{D})$.
  - **Property 2:** $\mathbf{M(D)} \subseteq \mathbf{B}$ for each $\mathbf{B} \in \mathbf{sem}(J, \mathbf{D})$.

  Thus the retrieved global database is the smallest global database in $\mathbf{sem}(J, D)$.

- **Provide the formal definition of universal solution for an information integration specification w.r.t to a source database D.**

  Consider an information integration system $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ is a **(G)LAV** information integration specification and $\mathbf{D}$ is a source database. A global instance $\mathbf{K}$ over G is like a global database over G except that it may contain ***labeled nulls***.

  Definition: a global instance $\mathbf{K}$ is a **solution** for J w.r.t. $\mathbf{D}$ if $q_S(\mathbf{D}) \subseteq q_G(\mathbf{K})$ for each mapping assertion $m = \langle \boldsymbol{q_S}, \boldsymbol{q_G}\rangle$ in M, i.e., $(\mathbf{K}, \mathbf{D}) \models M$.

  Definition: a global instance $\mathbf{K}$ is a **universal solution** for J w.r.t. $\mathbf{D}$ if $\mathbf{K}$ is a **solution** for J w.r.t. $\mathbf{D}$ such that, for each $\mathbf{B} \in \text{sem}(J, \mathbf{D})$, there exists a homomorphism from $\mathbf{K}$ to $\mathbf{B}$.

- **Describe the Naïve Chase Algorithm.**

  Definition: a mapping assertion $\langle \boldsymbol{q_S}, \boldsymbol{q_G}\rangle \in M$ is **triggerable** over $\mathbf{D}$ for a tuple of constants $\overline{a}$ (called the trigger) if $\overline{a} \in q_S(\mathbf{D})$.

  The definition of Universal Solutions **does not guarantee existence**. Indeed, the definition is not constructive and does not provide a procedure to compute it. **Universal solutions may not be unique.** For this reasons we will use the **chase algorithms** to ompute universal solutions. In the literature, there are different versions of this algorithm; we will see the simplest which uses **existential rules**. The algorithm is based on the following intuitions:

  - we need to materialize all triggerable mappings
  - we can use variables (labeled nulls) to deal with existentially quantified attributes
  - we have to materialize in the most general way

  The output of the algorithm is denoted with **ch(M,D)** and according to the **C theorem** is also a universal solution of J w.r.t. $\mathbf{D}$.

- **Let $J = \langle G, M, S\rangle$ be an information integration specification and let L be a class of queries. Provide the formal definition of maximally-sound L-rewriting of a query q w.r.t. J.**

  Definition: a **sound L-rewriting** of q w.r.t. M is a query $\boldsymbol{q'}$ in **L** such that, for every database $\mathbf{D}$ for S, the following condition holds:

  $$\text{if } \overline{a} \in q'(\mathbf{D}) \text{ then } \overline{a} \in \mathit{cert}(q, J, \mathbf{D})$$

  Definition: a **sound L-rewriting** q' of q w.r.t. J is a **maximally-sound L-rewriting** of q w.r.t. J if for every database $\mathbf{D}$ for **S** there exists no **sound L-rewriting** q'' of q w.r.t. J such that:

  $$q'(\mathbf{D}) \subseteq q''(\mathbf{D})$$

- Provide the definition of the query containment problem. Given the queries:

$$q = \{(y) \mid P(x, y) \wedge P(y, \text{`c'})\}$$

$$q' = \{(w) \mid P(w, k)\}$$

Tell, motivating your answer, if there is a **containment relationship** between them.

Definition: let q and q' be two queries of the same arity over a schema **S**. Query containment is the problem of checking whether $q(\mathbf{D}) \subseteq q'(\mathbf{D})$, for every database **D** for **S**. We write $\boldsymbol{q \sqsubseteq q'}$ to denote that q is contained in q'.

Theorem: let q and q' be two **CQs** of the same arity over **S**. We have that $q \sqsubseteq q'$ if and only if there exists a homomorphism h from q' to q such that $h(\overline{x_1}) = \overline{x_2}$.

We can tell that: $q \sqsubseteq q'$ becuase it exists an homomorphism from q' to q such that:

$$h(w) = y \qquad h(k) =' c'$$

- **Let Q be a query language and L be an ontology language. Provide the definition of Q-rewritability in L.**

Query answering can be thought as done in two phases:

- **Perfect rewriting:** produce from **q** and the **TBox** $\mathcal{T}$ a new query $r_{q,T}$)called the perfect rewriting of **q** w.r.t. $\mathcal{T}$).
- **Query evaluation:** evaluate $r_{q,T}$ over the **ABox** $\mathcal{A}$ seen as a complete database (without considering the **TBox** $\mathcal{T}$). The result of such evaluation is **cert(q,**$\langle \mathcal{T}, \mathcal{A} \rangle$

Definition [$\mathcal{Q}$**-rewritability**]: query answering is $\mathcal{Q}$-rewritable if for every $\mathcal{T}$ in $\mathcal{L}$ and query q, the perfect rewriting $r_{q,\mathcal{T}}$ of q w.r.t. $\mathcal{T}$ can be expressed in $\mathcal{Q}$.

10 Domande Paraboschi

- **Describe the advantages of the multidimensional model.**
- **Describe the principles used for the application of k-anonymity over large data collections.**
- **Describe the main ideas of the Mondrian algorithm and the requirements guaranteed on the produced output**
- **Describe the role of ℓ-diversity in data anonymization using k-anonimity.**
- **Describe the difference between k-anonimity and ℓ-diversity**
- **Describe the use of Randomized response to support differential privacy**
- **Classify the approaches for multidimensional indexing**
- Multidimensional index per poche dimensioni
- **Describe the main motivations behind the development of the key-value model.**
- **Describe the advantages of the map-reduce paradigm.**
- **Talk about the CAP theorem**

10.1 Describe the advantages of the multidimensional model

- **Conceptual expressiveness:** The multidimensional model provides a natural and semantically rich framework for organizing data in decision support systems. It is based on the separation between facts, representing quantitative measures of interest, and dimensions, which define the contextual axes along which the facts are analyzed. This separation facilitates the interpretation and formulation of analytical queries by end users, aligning with the cognitive process involved in multidimensional reasoning.
- **Support for OLAP operations:** The model is designed to natively support Online Analytical Processing (OLAP), enabling a wide range of operations that are essential for interactive data exploration. These operations include:
  - *Slice* — restricts the data to a single value of one dimension. projection
  - *Dice* — selects a subcube by filtering on multiple dimensions. selection
  - *Roll-up* — aggregates data by ascending a hierarchy in a dimension. aggregate
  - *Drill-down* — refines the analysis by descending a hierarchy to more detailed levels.
  - *Pivot* — reorients the dimensional view to examine data under different perspectives. change dimensional orientation

  These operations are expressive, efficient, and essential for multidimensional data analysis in business intelligence contexts.
- **Efficient query performance:** By leveraging precomputed aggregates and materialized views, often organized as multidimensional cubes, the model supports high-performance analytical queries even over large volumes of data. Indexing structures and storage techniques (e.g., MOLAP, ROLAP, HOLAP) further optimize the execution of aggregation queries.
- **Schema modularity and maintainability:** The strict distinction between facts and dimensions enhances the modularity of the data schema. This allows for more straightforward maintenance and evolution of the data warehouse, including the addition of new measures or dimensions without impacting existing queries.
- **Facilitation of user interaction:** The multidimensional model is well aligned with the structure of visual analytics tools and dashboards. This compatibility enables intuitive navigation and manipulation of data by users who may not possess technical expertise in query languages or relational modeling.
- **~~Temporal and hierarchical analysis~~:** The model naturally accommodates temporal dimensions and hierarchical relationships within dimensions, which are crucial for trend analysis, time-series comparisons, and aggregated reporting at multiple levels of granularity.
- **Integration with data warehousing:** The model fits seamlessly within the architecture of data warehouses, where data is often preprocessed, cleaned, and transformed into a format optimized for multidimensional access. It supports star and snowflake schemas, which are standard modeling techniques in the warehousing domain.

10.2 Describe the principles used for the application of k-anonymity over large data collections

- **Privacy-preserving motivation:** The application of k-anonymity in large-scale datasets arises from the need to mitigate re-identification risks in the release or analysis of microdata. When datasets contain quasi-identifiers—attributes that, while not directly identifying, can be linked to external information—they pose a significant threat to individual privacy. k-anonymity seeks to prevent such threats by guaranteeing that each record is indistinguishable from at least $k-1$ other records with respect to the quasi-identifiers.
  - data are de-identified (identifier is removed)
  - combining QI = identifier
  - equivalence class: group of records in a database that share the same values for the QI attributes
- **Definition and theoretical basis:** A dataset satisfies k-anonymity if every equivalence class formed by identical values of quasi-identifiers contains at least k records. This ensures that an adversary cannot uniquely associate a record with an individual with a confidence greater than $1/k$, assuming no access to sensitive attributes.
- **Techniques for achieving k-anonymity:** The enforcement of k-anonymity involves modifying quasi-identifying attributes to form sufficiently large equivalence classes. The two principal techniques are:
  - **Generalization** increases data indistinguishability by replacing specific attribute values with more general categories, thereby coarsening the resolution of the data space. can be applied at different levels of granularity (cell, row, column)

– **Suppression** removes values or entire records that cannot be adequately generalized without violating utility constraints or resulting in excessive information loss.

These transformations are applied selectively to minimize utility degradation while satisfying the anonymity condition.

- **Domain generalization hierarchies:** To guide the generalization process, domain generalization hierarchies (DGHs) are constructed for each quasi-identifier. A DGH defines permissible abstraction levels, allowing the algorithm to select the minimal generalization level that satisfies the k-anonymity constraint. These hierarchies are typically domain-specific and must be designed in accordance with semantic coherence and analytical needs.

- **Partitioning-based algorithms:** Efficient implementation of k-anonymity over large datasets relies on scalable partitioning strategies. Algorithms such as Mondrian perform multidimensional recursive partitioning, where the dataset is iteratively split along chosen attributes until all resulting partitions satisfy the minimum cardinality requirement. Each partition is then generalized independently to ensure consistency and privacy.

- **Scalability and performance concerns:** Applying k-anonymity to large data collections requires computationally efficient methods due to the combinatorial nature of the generalization space. Heuristics, indexing structures, and distributed processing are often employed to ensure scalability. Nevertheless, the trade-off between anonymity guarantees and data utility remains a critical design consideration.

- **Limitations and context of applicability:** While k-anonymity provides a clear and interpretable privacy guarantee, it is limited in scope. It does not address attribute disclosure or homogeneity attacks, where sensitive values within an equivalence class are uniform or nearly so. Consequently, k-anonymity is often used in combination with more robust models such as ℓ-diversity or t-closeness to achieve stronger privacy protection.
  - grants identity disclosure/data privacy
  - but no attribute disclosure: homogeneity and background knowledge attacks

10.3 Describe the main ideas of the Mondrian algorithm and the requirements guaranteed on the produced output

- **General overview and purpose:** The Mondrian algorithm is a multidimensional, recursive partitioning method designed to enforce k-anonymity on tabular datasets. It generalizes quasi-identifying attributes such that each record becomes indistinguishable from at least $k-1$ others within a partition. The key objective is to preserve data utility while satisfying privacy constraints, using a heuristic approach that balances granularity and anonymity.

how it works:

- Each attribute in QI represents a dimension
- Each tuple in PT represents a point in the space defined by QI
- Tuples with the same QI value are represented by giving a multiplicity value to points
- The multi-dimensional space is partitioned by splitting dimensions such that each area contains at least k occurrences of point values
- All the points in a region are generalized to a unique value
- The corresponding tuples are substituted by the computed generalization

- **Handling multidimensional generalization:** Unlike global recoding approaches that generalize values across the entire dataset, Mondrian applies local generalization within each partition. Once the recursion halts for a given partition, all records in that group are generalized uniformly according to the attribute ranges within the partition. This leads to more adaptive and information-preserving generalizations, particularly effective in high-dimensional spaces.

- **Output guarantees:** The algorithm ensures that the resulting dataset satisfies the k-anonymity property by construction. Each equivalence class, defined by the generalized values of quasi-identifiers, contains at least k records. The quasi-identifiers in each class are made identical through generalization, thus achieving indistinguishability among the records in the class.

- **Efficiency and scalability:** Mondrian is designed to be computationally efficient and scalable to large datasets. Its recursive structure and dimension-driven splitting strategy make it suitable for real-world scenarios where data volume and dimensionality pose significant challenges to privacy-preserving transformations.

- **Heuristic nature and information loss:** Although Mondrian does not guarantee minimal information loss, it offers a favorable trade-off by producing high-utility anonymized datasets. The heuristic dimension selection and median split criteria lead to balanced partitions that often yield better data quality than global approaches. Nevertheless, the algorithm is not optimal and its performance may vary depending on data distribution and parameter settings.

10.4 Describe the role of ℓ-diversity in data anonymization using k-anonymity

- identity disclosure: l'attaccante riesce a identificare l'identità della persona
- attribute disclosure: attaccante non sa chi la persona ma scopre un attributo sensibile

- **Motivation for extended protection:** While k-anonymity ensures that each individual is indistinguishable from at least $k-1$ others based on quasi-identifiers, it does not inherently protect against attribute disclosure. Specifically, if all records in a k-anonymous equivalence class share the same sensitive attribute value, an adversary can infer that value with certainty, defeating the purpose of anonymization. ℓ-diversity was introduced to address this fundamental limitation.

applied to a generalized table T altrimenti posso avere blocchi l-diverse

- **Definition and conceptual foundation:** The ℓ-diversity principle strengthens k-anonymity by requiring that each equivalence class contains at least ℓ "well-represented" distinct values for the sensitive attribute. This condition increases the uncertainty of an adversary attempting to infer sensitive information, even after identifying the equivalence class to which a target individual belongs.

- **Variants of ℓ-diversity:** To account for different data distributions and adversarial models, several formulations of ℓ-diversity have been proposed:
  - **Distinct ℓ-diversity** requires that each equivalence class contains at least ℓ distinct sensitive values.
  - per ogni classe di equivalenza la distribuzione dei valori deve essere superiore a una soglia log(l)
  - **Entropy ℓ-diversity** imposes a minimum entropy threshold, ensuring that the distribution of sensitive values in each class is sufficiently diverse. $\exp(\epsilon)$
  - **Recursive (c, ℓ)-diversity** restricts the dominance of the most frequent values by bounding the cumulative frequency of the top values relative to the others.

These variants provide increasing levels of protection in contexts with skewed or semantically clustered sensitive attributes.

- **Integration with k-anonymity:** ℓ-diversity operates as an additional constraint layered on top of k-anonymity. After a dataset is partitioned into k-anonymous equivalence classes, each class is examined for compliance with the ℓ-diversity criterion. If a class fails, further generalization or suppression is applied until the required diversity level is achieved.

- **Privacy guarantees and adversarial resistance:** By requiring multiple distinct values for the sensitive attribute within each group, ℓ-diversity limits the adversary's confidence in correctly guessing the sensitive value of a given individual. This provides resistance to homogeneity attacks and improves robustness against background knowledge.

- **Limitations and further extensions:** Despite its strengths, ℓ-diversity is not without limitations. In datasets where the sensitive attribute has an inherently skewed distribution, satisfying ℓ-diversity may lead to excessive generalization or even infeasibility. Furthermore, it does not account for the semantic similarity between distinct sensitive values. These limitations have motivated the development of more refined models such as t-closeness, which considers the distance between the distribution of sensitive values in each class and their distribution in the overall dataset.

10.5 Describe the difference between k-anonymity and ℓ-diversity

- **Objective and threat model:** k-anonymity and ℓ-diversity address distinct aspects of privacy in microdata publishing. k-anonymity focuses on preventing identity disclosure by ensuring that each individual is indistinguishable from at least $k-1$ others based on quasi-identifiers. ℓ-diversity extends this protection to mitigate attribute disclosure, ensuring that sensitive attributes remain uncertain even after successful record linkage.

- **Privacy definition:** k-anonymity requires that the projection of the dataset on quasi-identifiers yields equivalence classes of size at least k. ℓ-diversity requires, in addition, that each such class contains at least ℓ well-represented sensitive values, thus increasing the diversity of information an attacker could observe.

- **Vulnerability coverage:** k-anonymity is vulnerable to homogeneity and background knowledge attacks. These occur when all records in an equivalence class share the same sensitive value or when external knowledge can help narrow down the true value. ℓ-diversity addresses these attacks by ensuring variation in the sensitive attributes within each group.

- **Implementation strategy:** Both models typically rely on generalization and suppression to achieve the required anonymity levels. However, ℓ-diversity places stricter requirements on equivalence classes, often necessitating more aggressive generalization or restructuring of the data, particularly in skewed distributions.

- **Data utility trade-off:** k-anonymity generally preserves more data utility because it only enforces indistinguishability on quasi-identifiers. ℓ-diversity introduces additional constraints that may require greater distortion of the data, potentially leading to increased information loss in exchange for stronger privacy guarantees.

- **Formal hierarchy:** ℓ-diversity is a strictly stronger privacy notion than k-anonymity. Every dataset that satisfies ℓ-diversity also satisfies k-anonymity (with $k \geq \ell$), but the converse does not hold. ℓ-diversity is therefore considered a refinement that addresses limitations inherent in k-anonymity.

- **Limitations and extensions:** While ℓ-diversity improves on k-anonymity, it still suffers from limitations such as vulnerability to semantic similarity and difficulty handling skewed sensitive value distributions. These challenges have motivated the development of more nuanced models like t-closeness, which measures the statistical distance between sensitive value distributions.

10.6 Describe the use of Randomized Response to support Differential Privacy

- **introduction to randomized response:** randomized response is a privacy-preserving technique originally proposed by Warner (1965) to allow individuals to respond truthfully to sensitive binary questions while retaining plausible deniability. it introduces intentional random noise into individual responses to obscure the true values, thus protecting respondents' privacy.

- **mechanism definition:** given a population of n individuals, each with a binary value $x_i \in \{0,1\}$ representing their truthful answer to a sensitive question, the randomized response mechanism produces an output y_i for each individual, according to the following probabilistic rule:

  - Il meccanismo RR viene applicato su ogni singola risposta che può variare tra x e negx che sarebbero i miei due neighbors
  - ogni volta che applico il meccanismo RR su ogni singolo valore xi, sto implicitamente confrontando due possibili input vicini e le distribuzioni delle uscite sono tra loro simili

$$y_i = \begin{cases} x_i & \text{with probability } \frac{e^{\varepsilon}}{1+e^{\varepsilon}} \\ \bar{x}_i & \text{with probability } \frac{1}{1+e^{\varepsilon}} \end{cases}$$

  - rumore che introduco
  - piu epsilon é grande meno rumore introduco perche la probabilità che yi=xi é altissima

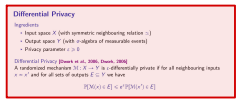
  where $\varepsilon \geq 0$ is the privacy parameter and $\bar{x}_i = 1 - x_i$ is the negated value of the true response. the mechanism is denoted by $\mathrm{RR}_{\varepsilon}(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$. the noise resposes

- **differential privacy guarantee:** this mechanism satisfies ε-differential privacy. the key property is that the probability distribution over outputs does not significantly change when a single input x_i is altered. in particular, for neighboring inputs $x \simeq x'$ and for all sets E of possible outputs,

Intuitions behind the definition:
- The neighbouring relation $\simeq$ captures *what* is protected
- The probability bounds capture *how much* protection we get

$$\Pr[M(x) \in E] \leq e^{\varepsilon} \Pr[M(x') \in E]$$

output output



  which ensures that the presence or absence of an individual's data has a limited impact on the final output, providing strong privacy guarantees.

- **plausible deniability:** since the reported response y_i may differ from the true value x_i, and this discrepancy is governed by a known probability distribution, it is impossible for an observer to determine with certainty whether a given answer reflects the respondent's actual data. this provides each individual with plausible deniability.

- **utility and unbiased estimation:** despite the introduced noise, statistical utility can be preserved. by averaging the observed responses and applying a correction

factor, an unbiased estimator for the true mean $\frac{1}{n}\sum_{i=1}^{n} x_i$ can be obtained. the expected error in estimating the mean using the randomized responses $\tilde{y}_i$ satisfies, with high probability:

- la formula mostra che possiamo stimare la media delle risposte vere con un errore controllato anche se ogni individuo ha aggiunto rumore alla propria risposta per proteggere la privacy
- l'errore diminuisce all'aumentare del numero di partecipanti aumenta se voglio più privacy cioé epsilon piccolo

$$\left|\frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{n}\sum_{i=1}^{n} \tilde{y}_i\right| \leq \mathcal{O}\left(\frac{1}{\varepsilon\sqrt{n}}\right)$$

I can apply a correction term to the observed values because I know their distribution

indicating that the accuracy improves with larger datasets and stronger privacy parameters (larger ε). ↓ε ↑ Privacy

- ~~**role in local differential privacy:**~~ randomized response is a canonical example of a *local differential privacy* (LDP) mechanism, where each individual perturbs their data before sharing it with the curator. this eliminates the need for a trusted central party and supports large-scale data collection with formal privacy guarantees.

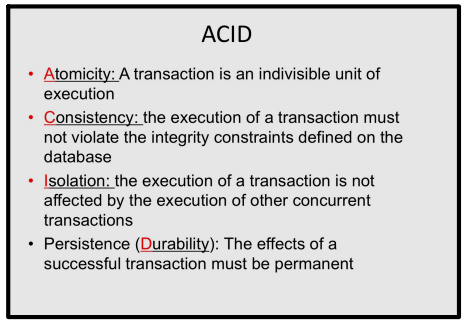
10.7 Classify the approaches for multidimensional indexing

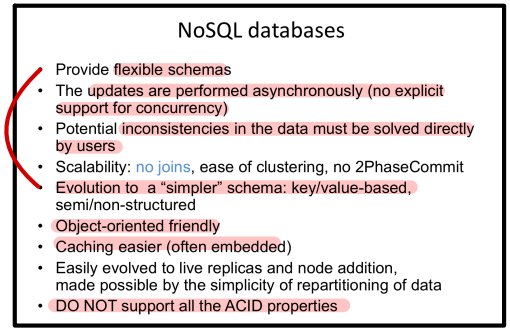
Multidimensional indexing techniques are essential for supporting efficient content-based retrieval in large data collections, especially when data are represented in terms of feature vectors or spatial descriptors. These techniques can be classified according to the dimensionality of the data they are designed to manage, the data model (vector-based vs. metric-based), and the strategy used for space partitioning.

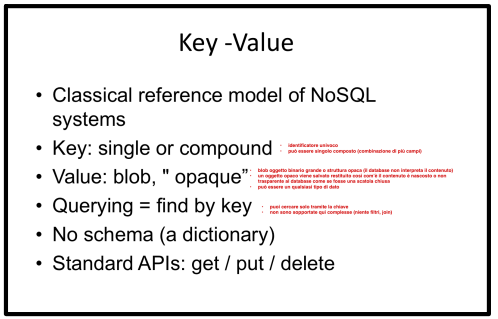
- **Low-dimensional indexing:** These approaches are typically employed for spatial data applications such as Geographic Information Systems (GIS) or Computer-Aided Design (CAD), where the data have two to four dimensions. The most common indexing methods include:
  - geometric entities
  - normalized
  - *k-d Trees:* Binary trees that alternate the splitting dimension at each level, effective for range queries.
  - ~~*Point*~~ k-d *Quadtrees:* Space is recursively partitioned into quadrants; suitable for dynamic insertion of spatial points.
  - Ogni ripartizione divide lo spazio lungo tutte le dimensioni contemporaneamente, creando più sotto regioni o quadranti
  - *R Trees and variants:* Designed for managing geometric objects (e.g., polygons, rectangles), using Minimum Bounding **Regions** (MBRs) to organize data in a balanced tree. Variants include:
    - MBR and MBS refer to the smallest rectangle or circle respectively that encloses the n-dimensional entity
    - partitioning criterias
    - ∗ *R* Trees:* Incorporate forced reinsertion to reduce overlap.
    - ∗ *R+ Trees:* Avoid overlap by allowing objects to appear in multiple leaves.
- **High-dimensional indexing:** Used in multimedia applications where feature vectors can have 64 to 500 dimensions. Traditional spatial indexes tend to degrade in performance due to the "curse of dimensionality". Key structures include:
  - performance degrade as number of dimensionality increases
  - *SS Trees:* Use Minimum Bounding Spheres (MBS) to form isotropic clusters with fewer dimensions per entry.
  - *SR Trees:* Combine bounding spheres and rectangles to balance space efficiency and overlap minimization.
- **Very high-dimensional or metric-based indexing:** These structures rely on metric properties (e.g., triangle inequality) rather than explicit coordinate-based partitioning. They are suitable for domains where distance functions are expensive or non-vectorial:
  - *M Trees:* Balanced metric trees that cluster data using routing objects and radii, supporting pruning via metric constraints.
  - *Distance-based indexes:* These include general-purpose metric trees that organize data solely based on inter-object distances.
- **Comparison with unidimensional indexing:** While structures like B+ Trees can simulate multidimensional indexing by concatenating multiple features into a composite key, this results in a linear ordering that fails to preserve spatial proximity. In contrast, true multidimensional indexes explicitly exploit geometric or metric nearness.

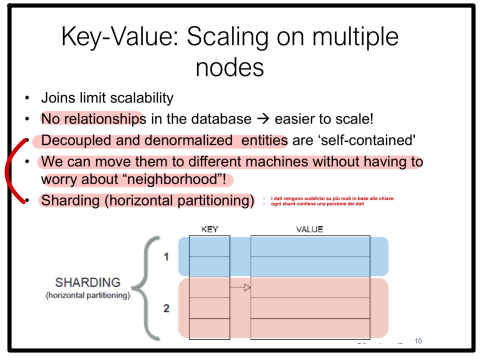
10.8 Describe the main motivations behind the development of the key-value model

The development of the key-value model within the NoSQL paradigm is primarily motivated by the necessity to handle the increasing complexity and scale of modern data management, particularly in the context of Big Data applications. Classical relational database management systems (RDBMS) offer strong transactional guarantees (ACID properties), rigid schemas, and support for complex queries and relationships. However, these features often impose significant limitations in terms of scalability, flexibility, and performance when applied to large, dynamic, and heterogeneous datasets.

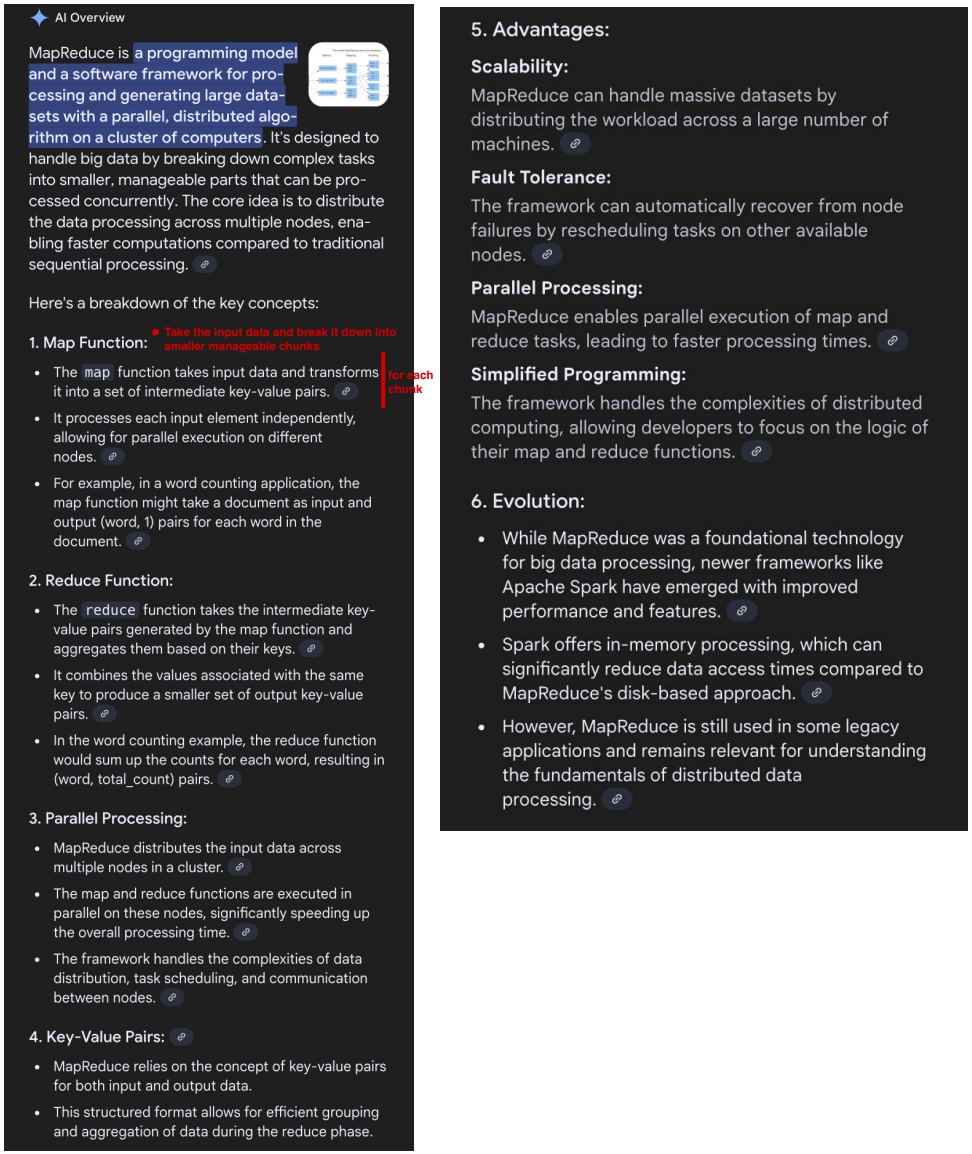








10.9 Describe the advantages of the map-reduce paradigm



AI Overview

MapReduce is a programming model and a software framework for processing and generating large datasets with a parallel, distributed algorithm on a cluster of computers. It's designed to handle big data by breaking down complex tasks into smaller, manageable parts that can be processed concurrently. The core idea is to distribute the data processing across multiple nodes, enabling faster computations compared to traditional sequential processing.

Here's a breakdown of the key concepts:

1. Map Function: Take the input data and break it down into smaller manageable chunks

- The `map` function takes input data and transforms it into a set of intermediate key-value pairs. for each chunk
- It processes each input element independently, allowing for parallel execution on different nodes.
- For example, in a word counting application, the map function might take a document as input and output (word, 1) pairs for each word in the document.

2. Reduce Function:

- The `reduce` function takes the intermediate key-value pairs generated by the map function and aggregates them based on their keys.
- It combines the values associated with the same key to produce a smaller set of output key-value pairs.
- In the word counting example, the reduce function would sum up the counts for each word, resulting in (word, total_count) pairs.

3. Parallel Processing:

- MapReduce distributes the input data across multiple nodes in a cluster.
- The map and reduce functions are executed in parallel on these nodes, significantly speeding up the overall processing time.
- The framework handles the complexities of data distribution, task scheduling, and communication between nodes.

4. Key-Value Pairs:

- MapReduce relies on the concept of key-value pairs for both input and output data.
- This structured format allows for efficient grouping and aggregation of data during the reduce phase.

5. Advantages:

Scalability:
MapReduce can handle massive datasets by distributing the workload across a large number of machines.

Fault Tolerance:
The framework can automatically recover from node failures by rescheduling tasks on other available nodes.

Parallel Processing:
MapReduce enables parallel execution of map and reduce tasks, leading to faster processing times.

Simplified Programming:
The framework handles the complexities of distributed computing, allowing developers to focus on the logic of their map and reduce functions.

6. Evolution:

- While MapReduce was a foundational technology for big data processing, newer frameworks like Apache Spark have emerged with improved performance and features.
- Spark offers in-memory processing, which can significantly reduce data access times compared to MapReduce's disk-based approach.
- However, MapReduce is still used in some legacy applications and remains relevant for understanding the fundamentals of distributed data processing.

- **Support for iterative enhancements:** Systems like Apache Spark extend the map-reduce paradigm by enabling in-memory computation and support for advanced features such as directed acyclic graphs (DAGs), thereby improving efficiency in iterative workloads.

In summary, the map-reduce paradigm offers a robust, scalable, and flexible framework for processing massive datasets, making it a cornerstone in the design of modern data-intensive applications in both industry and research.

10.10 Describe the CAP theorem

The CAP theorem, also known as Brewer's theorem, formalizes the trade-offs inherent in the design of distributed data management systems. It states that in a distributed system, it is impossible to simultaneously guarantee all three of the following properties:

- **Consistency (C):** All nodes in the system see the same data at the same time. This means that after an update operation completes, all clients accessing the system receive the updated value, regardless of the node they contact.
- **Availability (A):** Every request receives a (non-error) response, even if some nodes in the system are unreachable. The system continues to operate and serve client requests under normal and degraded conditions.
- **Partition Tolerance (P):** The system continues to function even when there are arbitrary delays or failures in communication between nodes. This includes tolerance to network partitions that may divide the system into disjoint subsets unable to communicate.

According to the CAP theorem, a distributed system can guarantee at most two of these three properties at any given time. This leads to three archetypal system designs:

- **CP (Consistency + Partition Tolerance):** These systems maintain consistency and can tolerate network partitions but may sacrifice availability during a partition (e.g., Google BigTable, HBase).
- **AP (Availability + Partition Tolerance):** These systems remain available and partition-tolerant but may serve stale or inconsistent data (e.g., Amazon DynamoDB, Cassandra).
- **CA (Consistency + Availability):** These systems provide both consistency and availability but cannot function correctly if a network partition occurs. Such configurations are feasible only in non-partitioned or single-node environments.

It is important to note that the CAP theorem applies under the assumption of a network partition. In the absence of partitions, it is possible for a system to exhibit all three properties. Moreover, modern systems often provide tunable consistency and allow developers to choose trade-offs at the granularity of operations, data items, or users.

In conclusion, the CAP theorem highlights the fundamental limitations of distributed systems and guides the design of NoSQL databases and cloud storage infrastructures by encouraging explicit trade-offs among consistency, availability, and partition tolerance.

11 Theory notes

- $\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid \mathbf{B} \text{ is a global database such that } (\mathbf{B}, \mathbf{D}) \models M\}$
- $\text{cert}(q, J, \mathbf{D}) = \{(c_1, \ldots, c_n) \mid (c_1, \ldots, c_n) \in q(\mathbf{B}) \text{ for each } \mathbf{B} \in \text{sem}(J, \mathbf{D})\}$
- 1) **Theorem:** let $J = \langle G, M, S\rangle$ be an information integration specification, $\mathbf{D}$ be a source database for S, and q be a FOL query over G. Computing $\text{cert}(q, J, D)$ is **undecidable**.
- **Definition:** a mapping assertion m is of the form: $m = \langle q_S, q_G\rangle$ where:
  - $q_S = \{\overline{x} \mid \varphi_S(\overline{x})\}$ is a FOL query over S.
  - $q_G = \{\overline{x} \mid \varphi_G(\overline{x})\}$ is a FOL query over G.

  These mappings have a natural **logical formulation**:

$$\rho : \forall \overline{\boldsymbol{x}}.\, \varphi_S(\overline{\boldsymbol{x}}) \rightarrow \varphi_G(\overline{\boldsymbol{x}})$$

- Given an $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ and $\mathbf{D}$ is a source database for S:
  - **Sound-mapping semantics:** the sources contain *partial* but *correct* information regarding the global schema:

$$\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid q_S(\mathbf{D}) \subseteq q_G(\mathbf{B}) \text{ for each } \langle q_S, q_G\rangle \in M\}.$$

  - **Complete-mapping semantics:** the sources contain *complete* but possibly *incorrect* information:

$$\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid q_S(\mathbf{D}) \supseteq q_G(\mathbf{B}) \text{ for each } \langle q_S, q_G\rangle \in M\}.$$

  - **Exact-mapping semantics:** the sources contain *exact* information regarding the global schema:

$$\text{sem}(J, \mathbf{D}) = \{\mathbf{B} \mid q_S(\mathbf{D}) \equiv q_G(\mathbf{B}) \text{ for each } \langle q_S, q_G\rangle \in M\}.$$

- Inside sound conjunctive mappings, we distinguish three important and widely used classes:

Global-As-View (GAV)	$\forall \overline{\boldsymbol{x}}.\, \varphi_S(\overline{\boldsymbol{x}}) \rightarrow g(\overline{\boldsymbol{x}})$.	$\exists$ a sx
Global-As-View (LAV)	$\forall \overline{\boldsymbol{x}}.s(\overline{\boldsymbol{x}}) \rightarrow \varphi_G(\overline{\boldsymbol{x}})$.	$\exists$ a dx
Global-As-View (GLAV)	$\forall \overline{\boldsymbol{x}}.\, \varphi_S(\overline{\boldsymbol{x}}) \rightarrow \varphi_G(\overline{\boldsymbol{x}})$.	$\exists$a sx e a dx

  in the $\forall$ goes the variables defined on the both parth of the mapping.

- **Definition (complete database):** the answers of evaluating the query q over a **database instance D** is the set $q(\mathbf{D})$, defined as:

$$q = \{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

$$q(\mathbf{D}) = \{\langle c_1, \ldots, c_n\rangle \in \mathbf{D}^n \mid \mathbf{D}, \langle c_1, \ldots, c_n\rangle \models \varphi(x_1, \ldots, x_n)\}$$

- **Definition (missing values):** consider a first-order logic (FOL) query

$$q = \{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

  A tuple $\langle c_1, \ldots, c_n\rangle \in \mathbf{dom}^n$ is a **certain answer** for q over a database $\mathbf{D}$ if:

$$\text{cert}(q, \mathbf{D}) = \{\langle c_1, \ldots, c_n\rangle \mid \langle c_1, \ldots, c_n\rangle \in q(v(\mathbf{D})) \text{ for every } \textbf{valuation } v\}.$$

In questo caso utilizzo $\exists$ per le variabili non incluse nella risposta della query, ossia le bounded variables. Infatti se voglio rispodere con tutte le var avró:

$$q = \{(x_1, \ldots, x_n) \mid \varphi(x_1, \ldots, x_n)\}$$

$$q_S = \{(n) \mid \exists d, r.\, Customers(n, d, r)\}$$
$$q_G = \{(n) \mid \exists i,\ Person(i, n)\}$$

Nel caso della formalizzazione logica utilizzo $\forall$ per indicare quelle variabili che compaoni in entrambe le parti della mappatura:

$$m_1 = \langle q_S, q_G\rangle$$
$$\rho : \forall \overline{\boldsymbol{x}}.\, \varphi_S(\overline{\boldsymbol{x}}) \rightarrow \varphi_G(\overline{\boldsymbol{x}})$$
$$m_1 : \quad \forall n.\, \exists d.\, \exists r.\, Customers(n, d, r) \rightarrow \exists i\, Person(i, n)$$

11.1 Materialization in GAV

- We want to produce a database $\mathbf{M(D)}$ to represent the result of the integration.
- We want to use the materialization $\mathbf{M(D)}$ to compute certain answers to queries.
- $\mathbf{M(D)}$ is computed by evaluating the source-side queries in each GAV mapping over the source database $\mathbf{D}$, and inserting the results into the global schema $\mathbf{G}$.
- We call ***retrieved global database*** for J w.r.t. $\mathbf{D}$, denoted by $\mathbf{M(D)}$:

$$\mathbf{M(D)} = \{g(\bar{c}) \mid \langle q_S, g\rangle \in M \text{ and } \bar{c} \in q_S(\mathbf{D})\}$$

- Since GAV mappings have no existential variables on the right-hand side, all tuples in $\mathbf{M(D)}$ are definite.
- **Properties:**
  - **(P1):** $\mathbf{M(D)} \in \mathbf{sem}(J, \mathbf{D})$
  - **(P2):** $\mathbf{M(D)} \subseteq \mathbf{B}$, for every $\mathbf{B} \in \mathbf{sem}(J, \mathbf{D})$

  Hence, $\mathbf{M(D)}$ is the *smallest* global database in $\mathbf{sem}(J,\mathbf{D})$.
- **Theorem:** consider an information integration system $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ is a **CQ-GAV** information integration specification, and $\mathbf{D}$ is a source database. For any **Conjunctive Query (CQ)** q over G, we have:

$$\mathbf{cert}(q, J, \mathbf{D}) = q(\mathbf{M(D)}).$$

- **Theorem:** consider an information integration system $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ is a **FOL-GAV** information integration specification, and $\mathbf{D}$ is a source database. For any **Conjunctive Query (CQ)** q over G, we have:

$$\mathbf{cert}(q, J, \mathbf{D}) = q(\mathbf{M(D)}).$$

11.2 Virtualization in GAV

- Rewrite a user query over the global schema **G** into a query over the source schema **S** and compute the certain answers by evaluating the rewritten query directly over the source database.
- **Virtualization in GAV is very simple:** we replace atoms in the user query with their definition in the mapping → ***unfolding***.
- Assume a class of queries **L** (a language of queries), e.g., **CQs**, **UCQs**, **FOL**, etc.
- **Definition:** given a query q over G, a **perfect L-rewriting** of q with respect to J is a query $\mathbf{rew}_{\mathbf{L}}^{(q,J)}$ in the language **L** such that the following condition holds for every database **D** for S:

$$\bar{a} \in \mathbf{cert}(q, J, \mathbf{D}) \text{ if and only if } \bar{a} \in \mathbf{rew}_{\mathbf{L}}^{(q,J)}(\mathbf{D}).$$

- The **unfolding** of a query q over G with respect to M (denoted by $\mathbf{unf}_{q,M}$) is the query over S obtained from q by substituting every atom $R(\bar{z})$ in q with the corresponding definition $\psi_S(\bar{z})$, such that $\langle \psi_S, R \rangle \in M$.
- The unfolding of a **UCQ** is obtained by unfolding each disjunct individually and then combining the results with a union.
- **Theorem:** given an information integration specification $J = \langle G, M, S \rangle$, where M is a **CQ-GAV** mapping, and given a **UCQ** q over G, we have that $\mathbf{unf}_{q,M}$ is a **perfect UCQ-rewriting** of q with respect to J.
- **Corollary:** for any **UCQ** q over G, and source database **D** for S, we have:

$$\mathbf{cert}(q, J, \mathbf{D}) = \mathbf{unf}_{q,M}(\mathbf{D}).$$

- **Theorem:** given an information integration specification $J = \langle G, M, S \rangle$, where M is a **FOL-GAV** mapping, and given a **UCQ** q over G, we have that $\mathbf{unf}_{q,M}$ is a **perfect FOL-rewriting** of q with respect to J.
- **Observe:** in this case, $\mathbf{unf}_{q,M}$ is a **FOL** query instead of a **UCQ**.
- **Theorem: UCQ-QA-IIS(CQ-GAV)** is polynomial in ***data complexity***, actually **LogSpace**. This result holds also in the case of **UCQ-QA-IIS(FOL-GAV)** which is polynomial (in **LogSpace**) in ***data complexity***.

11.3 Query Answering in (G)LAV Materialization

- **Thm: UCQ-QA(CQ-GLAV)** is decidable: via materialization and virtualization.
- As for **GAV**, we want to produce a database **M(D)** to represent the result of the integration, i.e., a global instance representing all the global databases in $\text{sem}(J, \mathbf{D})$.
- With (G)LAV we cannot simply materialize views from the sources:
  - There is no direct correspondence between definitions and global predicates.
  - Missing information (existential quantification) in the global schema definition
- Intuitively, to materialize GLAV mappings we need **null** values
- A global instance **K** over G is similar to a global database over G, but it may contain ***labeled nulls***, (placeholders for unknown constants). We denote the set of constants in **K** as **Const(K)**, and the set of labeled nulls in **K** as **Nulls(K)**. For convenience, we use the following notation for the domain of **K**:

$$\mathbf{dom(K)} = \mathbf{Const(K)} \cup \mathbf{Nulls(K)}$$

- **Definition:** let **A** and **B** be two database instances over the same schema. A ***homomorphism*** from **A** to **B** is a function h from **dom(A)** to **dom(B)** (where **dom(A)** and **dom(B)** can both contain constants, variables/labeled nulls) satisfying the following properties:

$$h : \mathbf{dom(A)} \to \mathbf{dom(B)}$$

1. For every constant $c \in \mathbf{Const(A)}$, $h(c) = c$. $\Rightarrow \mathbf{const(A)} \subseteq \mathbf{dom(B)}$
2. For every relational atom $R(\bar{a}) \in \mathbf{A}$, we have $R(h(\bar{a})) \in \mathbf{B}$. $\Rightarrow h(\mathbf{A}) \subseteq \mathbf{B}$

$$h(\mathbf{A}) = \{R(h(\bar{a})) \mid R(\bar{a}) \in \mathbf{A}\}.$$

Note: For any $x \in \mathbf{Nulls(A)}$, the homomorphism h may map x to either a constant or a labeled null in **dom(B)**; that is, $h(x) \in \mathbf{dom(B)}$, with no requirement that $h(x) = x$.

- **Thm:** the composition $h' \circ h$ is itself a homomorphism h'' from **A** to **C**.
- **Definition:** a global instance **K** is a **solution** for J w.r.t. **D** if $q_S(\mathbf{D}) \subseteq q_G(\mathbf{K})$ for each mapping assertion $m = \langle \boldsymbol{q_S}, \boldsymbol{q_G} \rangle$ in M, i.e., $(\mathbf{K}, \mathbf{D}) \models M$.
- **Definition:** a global instance **K** is a **universal solution** for J w.r.t. **D** if **K** is a **solution** for J w.r.t. **D** such that, for each $\mathbf{B} \in \text{sem}(J, \mathbf{D})$, there exists a homomorphism from **K** to **B**.
- **Theorem U:** let **K** be a universal solution for J w.r.t. **D**. For any **CQ** q, we have

$$\text{cert}(q, J, \mathbf{D}) = q^{\perp}(\mathbf{K}).$$

- The definition of Universal Solutions **does not guarantee existence**. Indeed, the definition is not constructive and does not provide a procedure to compute it. **Universal solutions may not be unique.**

- For GAV mappings, the retrieved global database is a universal solution.
- The **chase** is a family of **algorithms** that can compute universal solutions. In the literature, there are different versions of this algorithm; we will see the simplest.
- The chase can be used to perform reasoning tasks over **existential rules**.
- **Definition:** an **existential rule** is a first-order logic (FOL) formula of the following form, where φ and ψ are conjunctions of atoms:

$$\forall \overline{x}.\, \forall \overline{y}.\, \varphi(\overline{x}, \overline{y}) \rightarrow \exists \overline{z}.\, \psi(\overline{x}, \overline{z}).$$

- **Definition:** a mapping assertion $\langle \boldsymbol{q_S}, \boldsymbol{q_G} \rangle \in M$ is **triggerable** over $\mathbf{D}$ for a tuple of constants $\overline{a}$ (called the trigger) if $\overline{a} \in q_S(\mathbf{D})$.

Example: $\boldsymbol{m_1}$ is triggerable for $\langle a, a \rangle$ but not for $\langle b, c \rangle$

- The chase is based on a set of **intuitions**
  - we need to materialize all triggerable mappings → as facts into $\mathbf{K}$
  - we can use variables (labeled nulls) to feel up the existentially quantified attributes
  - we have to materialize in the most general way

The Naïve Chase Algorithm

Input: ⟨J, D⟩, where J = ⟨G, M, S⟩ is a GLAV information integration specification and D is a source database
Output: A global instance K over G (i.e., an incomplete database over G)

```
K ← ∅
For each m : ∀x̄.φ(x̄) → ∃ȳ.ψ(x̄,ȳ) occurring in M do:
    For each c̄ ∈ q_S(D), where q_S = {x̄ | φ(x̄)} do:
        For any y_i ∈ ȳ pick a fresh labeled null n_i ∈ n̄;
        Add to K all the atoms occurring in ψ(c̄, n̄);
    End For;
End For;
return K;
```

We denote by **ch(M,D)** the output K of the algorithm

- **Theorem C:** given $\langle J, \mathbf{D} \rangle$, where $J = \langle G, M, S \rangle$ is a **GLAV** information integration specification and $\mathbf{D}$ is a source database, we have that $\mathrm{ch}(M, \mathbf{D})$ is a universal solution of J w.r.t. $\mathbf{D}$.
- **Theorem:** Given $\langle J, \mathbf{D} \rangle$, where $J = \langle G, M, S \rangle$ is a **GLAV** information integration specification and $\mathbf{D}$ is a source database, and a **CQ** q, we have that

$$\mathrm{cert}(q, J, \mathbf{D}) = q^{\downarrow}(\mathrm{ch}(M, \mathbf{D})).$$

- **Thm: UCQ-QA(CQ-GLAV)** is decidable (polynomial in data complexity).
- **Proof:** for a **GLAV** mapping M and database $\mathbf{D}$, we can compute a universal solution $\mathbf{K}$ of $\mathbf{D}$ w.r.t. M via the naïve chase (in polynomial time w.r.t. the size of $\mathbf{D}$) and use the naïve evaluation to answer the query q over $\mathbf{K}$ (in polynomial time w.r.t. the size of $\mathbf{D}$).

11.4 Query Answering in (G)LAV Virtualization

- We want to produce a query over the sources that captures the results of a user query over the global schema.
- We cannot use simple unfolding techniques with **LAV** mappings (as for **GAV**), since **LAV** does not define the global schema atoms directly → $\exists$ quantifier on the right
- **Definition:** given a query q over G, a **perfect L-rewriting** of q w.r.t. J is a query $\boldsymbol{rew_L^{(q,J)}}$ in the language $\mathbf{L}$ such that the following condition holds for every database $\mathbf{D}$ for $\mathbf{S}$:

$$\overline{a} \in cert(\boldsymbol{q}, \boldsymbol{J}, \boldsymbol{D}) \iff \overline{a} \in \boldsymbol{rew_L^{(q,J)}}(\boldsymbol{D})$$

- Fix **UCQ** as global query language ($\boldsymbol{L_G}$) and source query language ($\boldsymbol{L_S}$).
- **Definition:** we can show that **perfect UCQ-rewriting *always exists*** and is computable, and we can provide and algorithm for its computation. The algorithm is based on the following ***intuitions***:
  - **Sound rewritings:** rewritings that capture only portion of the certain answers
  - **Maximally-sound rewritings:** rewritings that capture as much as possible in a certain class
  - **Query containment:** checking whether the answers for a query are always contained in the answers for another.
- **Definition:** a **sound L-rewriting** of q w.r.t. M is a query $\boldsymbol{q'}$ in $\mathbf{L}$ such that, for every database $\mathbf{D}$ for S, the following condition holds:

$$\overline{a} \in q'(\mathbf{D}) \Rightarrow \overline{a} \in cert(q, J, \mathbf{D})$$

  Sound rewritings may not always produce the same results as the original query, they ensure that they **never return incorrect results**, instead **it is guaranteed to be a valid result** for the original query but not the **completeness oif the answer**.
- **Observation:** the unsatisfiable query $\{(\,) \mid \bot\}$, (denoted by $\bot$, representing an empty set) is always a ***sound rewriting***, because it doesn't return any incorrect results, there are simply no results at all.
- **Definition:** a **sound L-rewriting** q' of q w.r.t. J is a **maximally-sound L-rewriting** of q w.r.t. J if for every database $\mathbf{D}$ for $\mathbf{S}$ there exists no **sound L-rewriting** q'' of q w.r.t. J such that:

$$q'(\mathbf{D}) \subseteq q''(\mathbf{D}) \equiv q' \sqsubseteq q''$$

  maximally-sound rewritings are one of the **best possible approximations** of certain answers that one can obtained in a language. Of course, **they may not be unique**.
- **Definition:** let $\boldsymbol{q_m}$ be the **UCQ** defined as the union of all the **maximally-sound CQ rewritings** of $\boldsymbol{q}$ w.r.t. J.

- **Thm:** q_m is a **perfect UCQ-rewriting** of q w.r.t. J.
- **Observation:** in order to build a perfect rewriting we can
  - Compute all maximally sound rewritings of the given query
  - Output the union of all these rewritings

  This is still not an algorithm:
  - How can we generate the maximally sound rewritings?
  - How many of them are there?

  To answer these questions we must first introduce the following notions: **Query containment** and **Query Expansion**.
- **Definition:** the **expansion** $exp^{(q_S,M)}$ of q_S w.r.t. a mapping M is the query over G obtained by replacing each atom $R(\overline{z})$ in q_S with the corresponding view q_R over G, where $\langle R, q_R\rangle \in M$.
- **Observe:** $exp^{(q_S,M)}$ is a **CQ**.
- Intuitively, we use M as a reverse mapping from the source schema to the global schema and unfold q_S over M in a **GAV** fashion.
- **Definition:** let q and q' be two queries of the same arity over a schema **S**. **Query containment** is the problem of checking whether $q(\mathbf{D}) \subseteq q'(\mathbf{D})$, for every database **D** for **S**. We write $q \sqsubseteq q'$ to denote that q is contained in q'.
- **Theorem:** let q and q' be two **CQs** of the same arity over **S**. We have that $q \sqsubseteq q'$ if and only if there exists a homomorphism h from q' to q such that $h(\overline{x_1}) = \overline{x_2}$.
- **Theorem (check sound rewritings):** q_S is a sound rewriting of q_G if and only if $exp^{(q_S,M)} \sqsubseteq q_G$, i.e., $exp^{(q_S,M)}(D) \subseteq q_G(D)$, for every database **D** for G.
- **Theorem:** let $q_G = \{\overline{x} \mid \varphi(\overline{x})\}$ be a **CQ** over G such that φ consists of n distinct atoms. If a **CQ** $q_S = \{\overline{x} \mid \psi(\overline{x})\}$ is a **maximally-sound CQ-rewriting** of q_G w.r.t. J, then ψ is logically equivalent to an existential conjunction of at most n distinct atoms.

The Perfect Rewriting Algorithm

```
Input: An I.I.S. J = ⟨G, M, S⟩ where M is a Conjunctive LAV mapping, and a
       CQ q_g = {x̄ | φ(x̄)} where φ has n conjuncts.
Output: A UCQ Q over S

Q ← ∅
For each CQ q' = {x̄ | ψ(x̄)} over S with at most n conjuncts
    If q' is a sound rewriting of q_g, i.e., if exp^(q',M) ⊑ q_g
        Q ← Q ∪ {q'}
    End If;
End For;
return Q;
```

- **Theorem:** let $J = \langle \mathrm{G}, \mathrm{M}, \mathbf{S}\rangle$ be a **LAV** information integration specification and q_G be a **CQ** over G. For input J and q_G, the **perfect rewriting algorithm** computes a perfect **UCQ-rewriting** of q_G w.r.t. J.
- **Proof:** the algorithm computes the union of all the **maximally-sound CQ-rewritings** of the input query.
- **Theorem: UCQ-QA(CQ-GLAV)** is decidable, **LOGSPACE** in data complexity.
- **Proof:** For a **GLAV** mapping **M**, a query q over G, and database **D** for S:
  - We can compute the perfect **UCQ-rewriting** q' of q w.r.t. J (independently from the size of **D**)
  - Then we can evaluate q' over **D** in polynomial time w.r.t. the size of **D** (actually in **LOGSPACE**)

Materializzazione:

- Calcolare la materializzazione: ossia per ogni mappatura $\langle q_S, q_G\rangle$ importare tutte le tuple $q_S(D)$ all'interno della materialized seguengo la definizione della mappatura (inserisco il fatto q_G), ottenendo $\mathbf{M(D)}$ il retrived global db, oppure **K** global istance

$$\langle q_S, q_G\rangle$$
$$\forall \overline{x}.\, \varphi_S(\overline{x}) \rightarrow g(\overline{x})$$
$$\overline{c} \in q_S(D) : \forall \overline{x}.\, \varphi_S(\overline{x})$$
$$\mathbf{K} = \mathbf{M(D)} = \{g(\overline{c})\}$$

- Per generare **K** generico applicare il chase naive algorithm dove $\mathbf{K} = ch(M, D)$
- Calcolare la certain:

$$cert(q, J\mathbf{D}) = q((\mathbf{M(D)}) = q^{\perp}(\mathbf{ch(M,D)})$$

Vitualizzazione:

- Riscrivo le query q su G in query su S impiegando le mapping assertion in **M**
- In **GAV** basta effettuare $unf_{q,M}$ perché vie una definizione diretta senza **nulls**
- In **(G)LAV** devo computare la q_m impiegando il **the perfect rewriting algorithm**

$$cert(q, J, \mathbf{D}) = \mathbf{unf}_{q,\mathbf{M}}(\mathbf{D}) = q_m(D)$$

11.5 Ontology perfect rewriting

Algorithm *PerfectRef*(Q; T_P)
Input: union of conjunctive queries Q, set of DL-Lite$_A$ PIs T_P
Output: union of conjunctive queries *PR*
PR := Q;
repeat
  PR' := *PR*;
  for each $q \in PR'$ **do**
    for each atom $g \in q$ **do**
      for each PI $I \in T_P$ **do**
        if I is applicable to g **then** *PR* := *PR* ∪ { *ApplyPI*(q, g, I) };
    for each pair of atoms $g_1, g_2 \in q$ **do**
      if g_1 and g_2 unify then *PR* := *PR* ∪ { *Reduce*(q, g_1, g_2) };
until *PR'* = *PR*;
return *PR*;

Observations: 1) **Termination** follows from having only finitely many different rewritings.
2) **NIs** or **functionalities** do not play **any role** in the rewriting of the query.

12 Observations

- The architecture of an **IIS** can be defined using **FOL**
- Consider to have instance **D** of a schema S over a domain **dom** we can see **D** as a FOL interpretation.
- For each quantifier variable I have to specify it such as: $\exists x.\, \exists y.$ or $\forall x.\, \forall y.$
- **cert(q,J,D)** is, the intersection of the answers for q over all the global database in $sem(J, D)$.
- Querying $\langle J, D\rangle$ simply means posing queries over the global schema G of the information integration specification J.
- Complete and exact mappings can exhibit very counterintuitive behavior: may give rise to **inconsistent** systems (i.e., systems with no semantics $sem(J, \mathbf{D}) = \varnothing$).
- 1) The theorem above tells us that we cannot expect to answer queries over and Information integration system with arbitrarily expressive mappings. We need to impose further restrictions on the languages involved ⇒ The restriction that we will adopt is to assume mappings that only use **conjunctive queries (CQ)**.
- GAV mappings provide direct information on what data populates the relations of the global schema.
- LAV mappings do not provide direct information on what data populates the global schema. Single elements of the global schema may not be explicitly defined
- We can think of **GLAV mappings** as the ***composition*** of a **LAV** and a **GAV** mapping. Indeed, each **GLAV mapping** $\langle \boldsymbol{q_S}, \boldsymbol{q_G}\rangle$ can be rewritten as:

$$\langle \boldsymbol{q_S}, \boldsymbol{p}\rangle \cup \langle \boldsymbol{p}, \boldsymbol{q_G}\rangle$$

- From an algorithmic perspective, we can consider LAV and GLAV the same.
- The answer of a query over a database instance **D** is defined as $q(\mathbf{D})$
- we use **null** to represent incomplete information. The common assumption is that **NULL** represents an unknown value. SQL uses this notion of NULL. SQL engines evaluate conditions using three truth values. Tuples evaluating to true are then returned.
- We call a database instance **complete** if it does not contain nulls.
- A more reasoned approach to incomplete information in relational databases is the use of multiple null values. We often refer to this model of nulls as **Marked Nulls**.
- To capture the notion of incomplete information, we use the notion of **valuation**.
- G is an empty theory over the alphabet $\mathbf{A}_G$, meaning G is a database schema without integrity constraints.
- Assume an (incomplete) database **D** and a query $\boldsymbol{q}$. We denote by $q(\mathbf{D})$ the evaluation of q over **D**.
- Both a *database instance* and a *query* are defined as sets of *atoms*, also called facts in the case of databases.

A is an istance of a database schema S

$$A = \{R(a, b), S(t, v)\}$$

13 Theorems

- **Theorem:** let $J = \langle G, M, S\rangle$ be an information integration specification, **D** be a source database for S, and q be a **FOL** query over G. Computing $\mathbf{cert}(q, J, \mathbf{D})$ is **undecidable**. ⇒ We need restrictions for the mapping.
- **Theorem:** let q be a **FOL** query. Checking whether $\bar{a} \in \mathbf{cert}(q, \mathbf{D})$ can be done in **EXPTIME** in combined complexity and is **NP-Complete** in data complexity.
- **Theorem:** if q is a **Union of Conjunctive Queries (UCQ)**, then:

$$\mathbf{cert}(q, \mathbf{D}) = q^{\perp}(\mathbf{D}).$$

  This theorem states that the set of **certain answers** for a UCQ q over **D** is equivalent to the result of the naïve evaluation of q over **D**.
- **Theorem:** let q be a **Union of Conjunctive Queries (UCQ)**. Checking whether $\bar{a} \in \mathbf{cert}(q, \mathbf{D})$ is **NP-Complete** in combined complexity and is in **LOGSPACE** in data complexity.
- **Corollary:** query answering for **Union of Conjunctive Queries (UCQs)** is in **PTIME** in data complexity.
- **Definition: QA-IIS** is the following decision problem:
  - **Input:** an I.I.S. $J = \langle G, M, S\rangle$, a first-order logic (FOL) query $q(\bar{x})$, a source database **D**, and a tuple of constants $\bar{c}$.
  - **Question:** is $\bar{c} \in \mathbf{cert}(q, J, \mathbf{D})$?
  - **Theorem:** the problem **QA-IIS** is undecidable.
- **Definition (restrictions applied): UCQ-QA-IIS($\mathcal{L}$)**, where $\mathcal{L}$ is a class of mappings, is the following problem:
  - **Input:** an I.I.S. $J = \langle G, M, S\rangle$, where G and S are empty theories, M is a mapping in $\mathcal{L}$, a source database **D**, a **UCQ** q, and a tuple of constants $\bar{c}$.
  - **Question:** Is $\bar{c} \in \mathbf{cert}(q, J, \mathbf{D})$?
  - **Reply:** we will see, **UCQ-QA-IIS($\mathcal{L}$)** is decidable in several interesting cases: **CQ-GAV**, **CQ-LAV** and **CQ-GLAV** mappings.
- **Theorem:** consider an information integration system $\langle J, \mathbf{D}\rangle$, where $J = \langle G, M, S\rangle$ is a **CQ-GAV** information integration specification, and **D** is a source database. For any **Conjunctive Query (CQ)** q over G, we have:

$$\mathbf{cert}(q, J, \mathbf{D}) = q(\mathbf{M(D)}).$$

- **Theorem:** consider an information integration system $\langle J, \mathbf{D} \rangle$, where $J = \langle G, M, S \rangle$ is a **FOL-GAV** information integration specification, and $\mathbf{D}$ is a source database. For any **Conjunctive Query (CQ)** q over G, we have:

$$\mathbf{cert}(q, J, \mathbf{D}) = q(\mathbf{M(D)}).$$

- **Theorem:** given an information integration specification $J = \langle G, M, S \rangle$, where M is a **CQ-GAV** mapping, and given a **UCQ** q over G, we have that $\mathbf{unf}_{q,M}$ is a **perfect UCQ-rewriting** of q with respect to J.

- **Corollary:** for any **UCQ** q over G, and source database $\mathbf{D}$ for S, we have:

$$\mathbf{cert}(q, J, \mathbf{D}) = \mathbf{unf}_{q,M}(\mathbf{D}).$$

- **Theorem:** given an information integration specification $J = \langle G, M, S \rangle$, where M is a **FOL-GAV** mapping, and given a **UCQ** q over G, we have that $\mathbf{unf}_{q,M}$ is a perfect **FOL**-rewriting of q with respect to J.

- **Theorem: UCQ-QA-IIS(CQ-GAV)** is polynomial in ***data complexity***, actually **LogSpace**. This result holds also in the case of **UCQ-QA-IIS(FOL-GAV)** which is polynomial (in **LogSpace**) in ***data complexity***.