QUANG HIEU VU, BENG CHIN OOI, DIMITRIS PAPADIAS,
ANTHONY K. H. TUNG

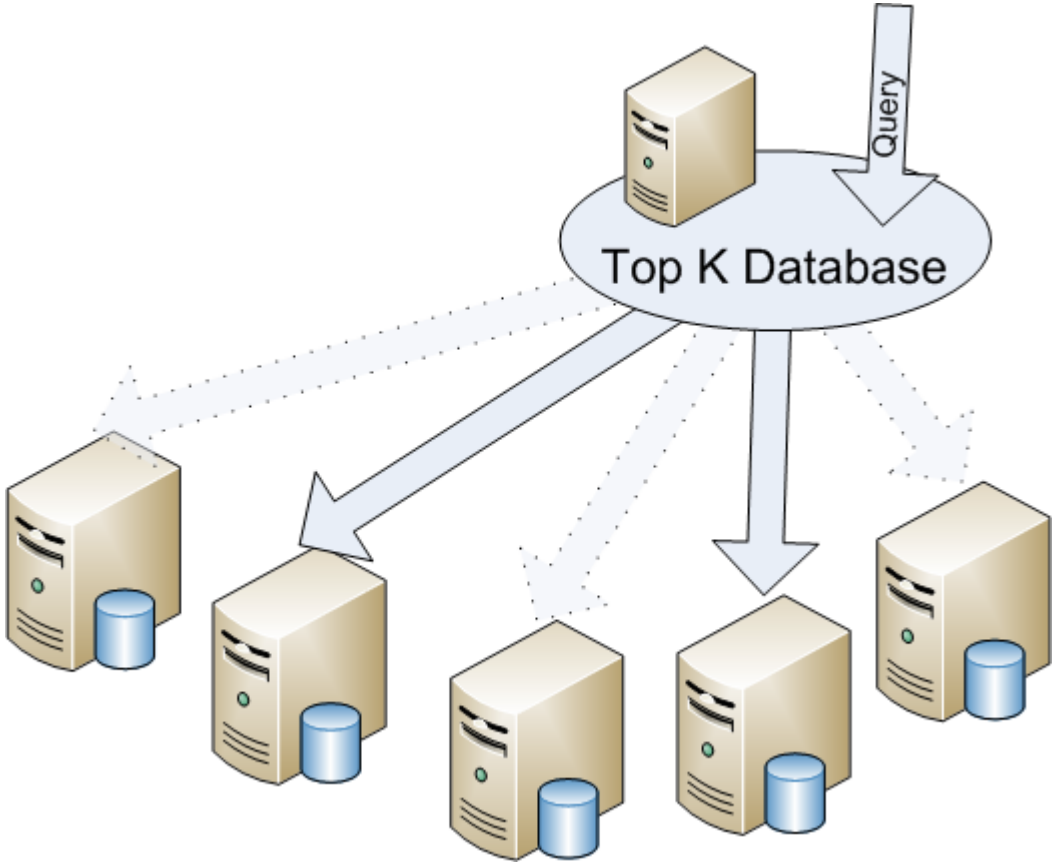
A GRAPH METHOD FOR KEYWORD-BASED SELECTION OF THE TOP-K DATABASES

Gruppo 9

Cacciari Alessandro, Franceschelli Alessio, Iuliani Matteo

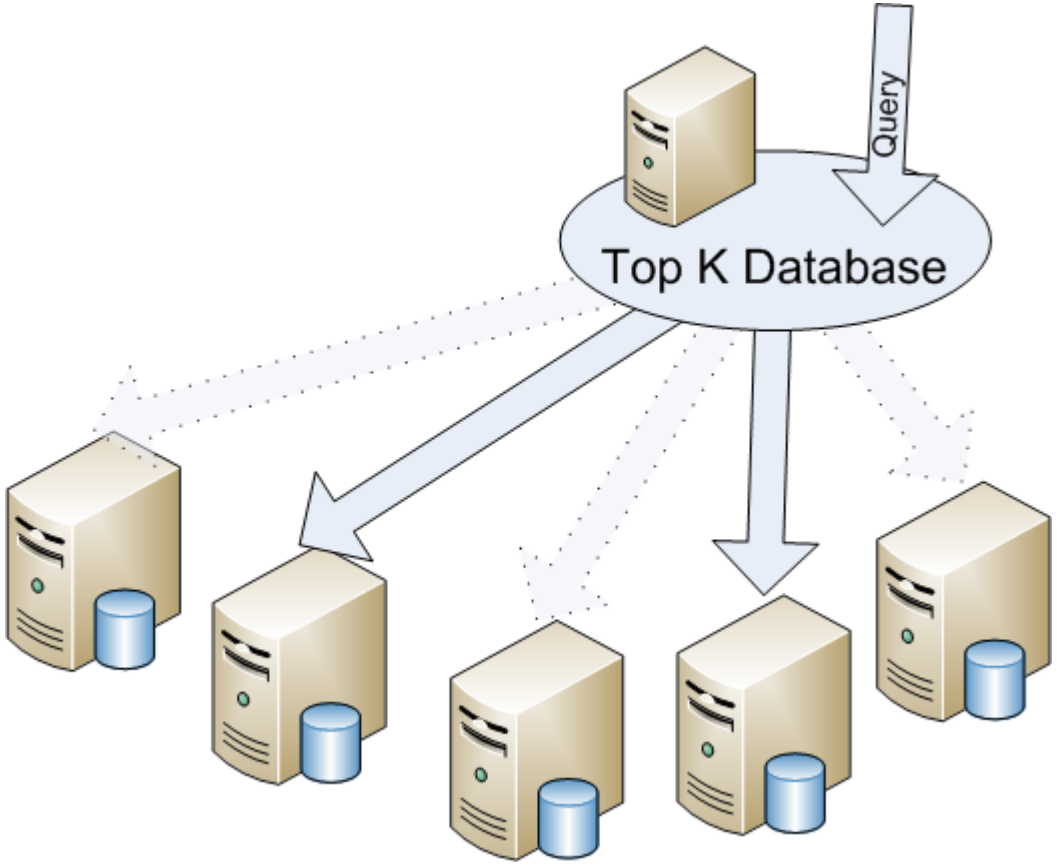
Scopo del lavoro

- Ottimizzare l'esecuzione di **Keyword Search (KS) query** su più database
  - **Selezionare i top-K database** su cui effettuare la **KS query** (**K** parametro di ingresso)
  - Data una **query *q***, il sistema dovrà eseguire *q* solo su un **sottoinsieme K dei database,** in modo da *minimizzare il costo totale* di esecuzione di *q*, senza pregiudicare la bontà del risultato

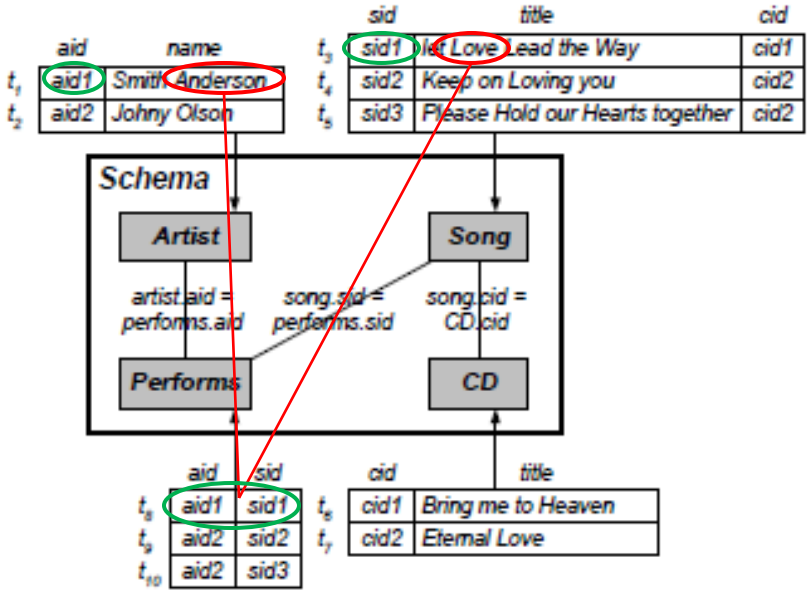


Scopo del lavoro

- Ottimizzare l'esecuzione di **Keyword Search (KS) query** su più database
  - L'accento è sulla **scelta dei migliori K database, per quella determinata query**
  - Evitando la ricerca su **database irrilevanti** ai fini del risultato (DB che non produrranno "buoni risultati")



Keyword search su database



$$q = \{\text{Anderson}, \text{love}\}$$

$$r = \{t_1 \rhd\lhd t_8 \rhd\lhd t_3\}$$

d=2

Risultato di una KS su database:

- insieme di **tuple**
- ordinate in base alla **distanza** (numero di join)

Stato dell'arte: summaries

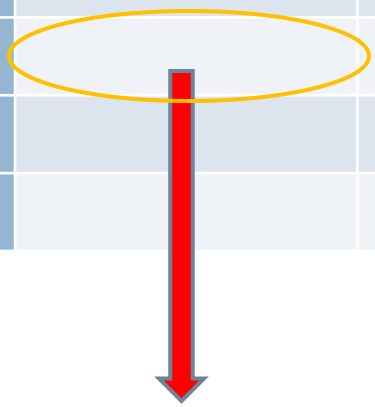
SUMMARY

Termine	Frequenza
t1	3
t2	2
t3	4
…	…

- Fase di **pre-processing** per scartare dall'insieme di ricerca i **database che non promettono buoni risultati**
  - **Summary**
    - Collezione dei termini presenti in un database con indicazione di **frequenza**
    - Manca l'informazione sulla **connessione** fra i termini
  - Per ovviare al problema:
    - **Keyword Relationship Matrix (KRM)**

Keyword Relationship Matrix (KRM)

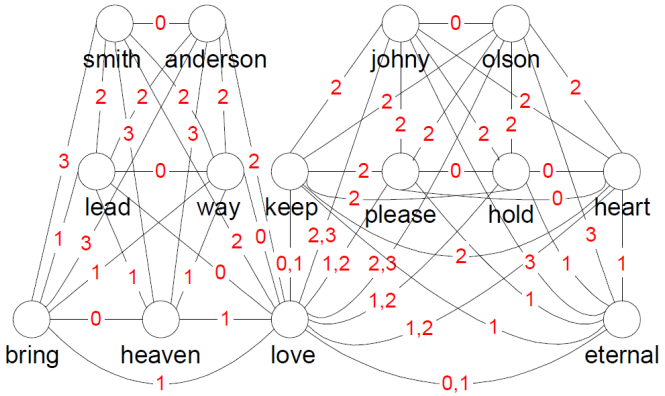
	Termine 1	Termine 2	Termine 3	Termine 4
Termine 1				
Termine 2				
Termine 3				
Termine 4				



Distanza	Frequenza
0	3
1	2
2	4
…	…

- Il metodo di selezione **M-KS** utilizza le informazioni descritte da KRM per stabilire quali sorgenti **escludere** dalla query
- ☹ **Si catturano solo le relazioni binarie**
  - **aumento falsi positivi** per query a più di due termini
- ☹ **Non è possibile gestire l'operatore OR**

Keyword Relationship Graph (KRG)



N.B. In questa figura e nelle successive i pesi di nodi e archi non si rappresentano per motivi di chiarezza. **I numeri sugli archi rappresentano le distanze** (vedi dopo)

- Tecnica di **summary** che riassume un database attraverso un **grafo di relazione fra le keyword**
- Cattura i termini e le loro relazioni tramite **nodi** e **archi** pesati
- ☺ **Minimizza le possibilità di incorrere in falsi positivi**
  - Impone condizioni più stringenti rispetto alle semplici relazioni binarie di KRM
- ☺ **Rispetta le semantiche AND e OR**
- Il metodo di selezione **G-KS utilizza le informazioni descritte da KRG** per stabilire **quali sorgenti escludere dalla query**

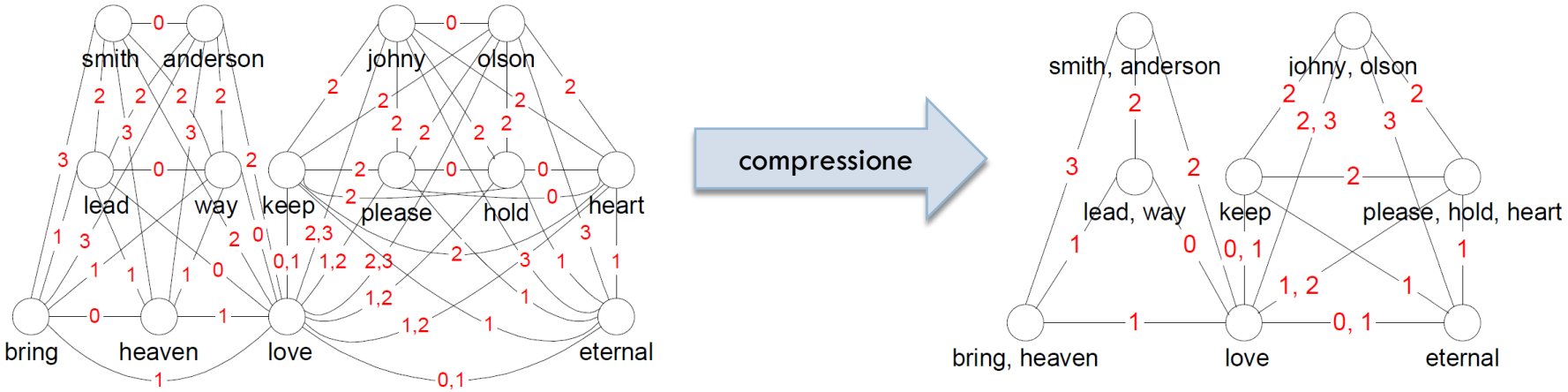
KRG: Nodi

- Un **nodo** corrisponde a un **termine** presente nel database
- Ha un **peso** *w* che riflette l'importanza relativa del termine rispetto agli altri termini del database
  - la **frequenza** di un termine in una tupla è il numero relativo delle sue occorrenze rispetto al numero totale di termini nella tupla
  - La **frequenza inversa** di un termine riflette l'importanza del termine rispetto all'intero insieme delle tuple
  - Il **peso di un termine** in una tupla è il prodotto della frequenza del termine nella tupla e la frequenza inversa
  - Il **peso di un nodo** è la media dei pesi tra tutte le tuple che contengono il termine relativo al nodo

KRG: Archi

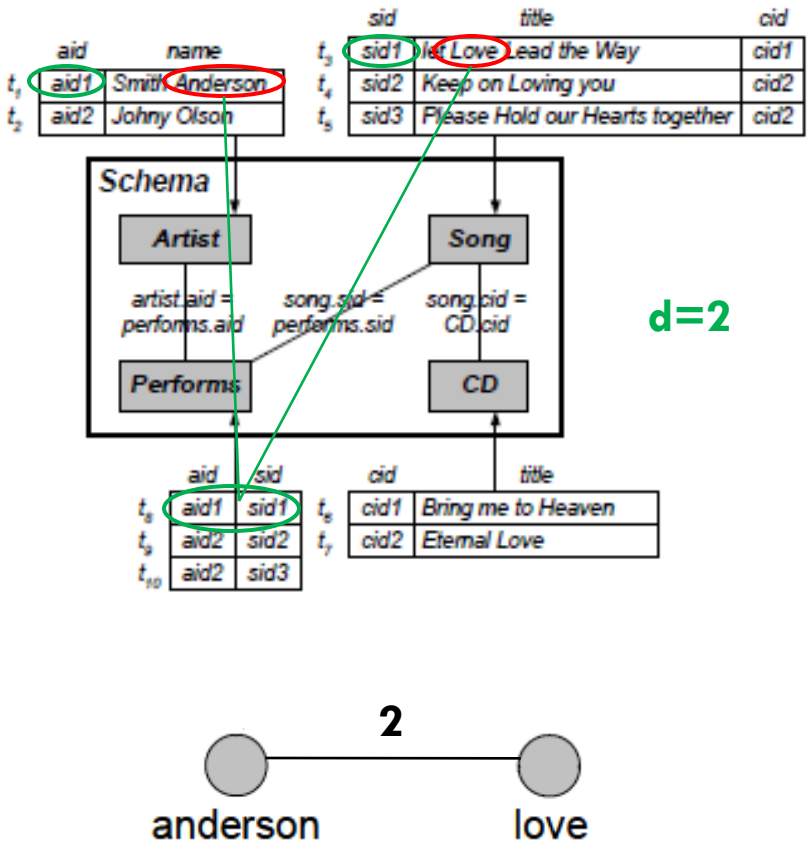
- Gli **archi** collegano due **nodi** rappresentanti termini appartenenti a tuple che possono essere collegate fra loro da una **sequenza di join**
  - Su ogni arco è indicata la **distanza *d***
    - numero di join necessari per collegare i termini
  - Due nodi potrebbero essere collegati con **cammini multipli** di differenti distanze
    - devono essere indicate tutte le diverse distanze
  - **A ciascuna distanza viene assegnato un peso** che misura l'importanza della connessione, analogamente ai nodi
    - Intuitivamente:
      - per ogni **coppia di tuple** ad una data distanza d viene associato un **peso**
      - il **peso di una connessione di distanza d** è la **media dei pesi di tutte le coppie di tuple del database che sono connesse a distanza d e che includono i due termini**

KRG: Compressione



- Sperimentalmente si vede che **<u>oltre la metà dei termini appare una volta sola nel DB</u>**
- Se questi termini appaiono nella **stessa tupla** avranno:
  - Stesso **peso, connessioni** agli altri nodi, **peso delle connessioni**
- Si raggruppano in un unico nodo i termini che **appaiono una sola volta nel DB** e che **appartenengono alla stessa tupla**
  - → **<u>Compound node</u>**: il peso del nodo e degli archi coinvolti è calcolato a partire da uno qualsiasi dei termini inclusi

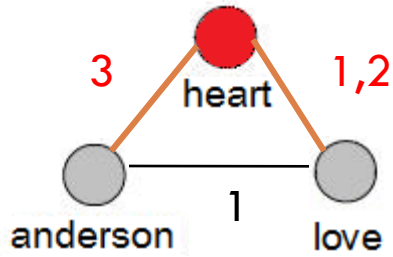
KRG: Costruzione



1. Estrazione dei **termini** dalle tuple per creare i **nodi**
   - Genera già i compound node ove necessario
   - Assegna i **pesi** ai nodi
2. Costruzione delle **relazioni** fra le tuple (➔d>0)
   - Due tuple sono connesse a **distanza d**, se esiste una relazione di join in d passi
3. Creazione degli **archi**
   - Si basa sulle relazioni individuate al passo precedente
   - Due nodi corrispondenti a termini presenti nella stessa tupla sono a distanza 0

KRG: Aggiornamento

- **Aggiornamento** → **cancellazione** seguita da **inserimento**
- **Inserimento** di una tupla
  - Nuovi termini (simile alla costruzione):
    - Creazione nuovi nodi e nuove relazioni
  - Termini preesistenti
    - Aggiornamento pesi dei nodi
    - Creazione nuove relazioni e aggiornamento pesi relazioni esistenti



- **Cancellazione** di una tupla
  - Aggiornamento pesi dei nodi di termini contenuti nella tupla eliminata (cancellazione del nodo se il termine era unico nel DB)
  - Aggiornamento pesi degli archi e eventuali cancellazioni

- Il KRG di solito viene aggiornato dopo un certo numero di modifiche per ridurre l'**overhead di revisione** del grafo

Elaborazione query (1)

- Affinché un DB contenga **almeno un risultato** di una Keyword Search:
  1. **Ogni keyword** deve essere presente in **almeno una tupla** del DB
  2. **Ogni coppia di keyword** deve essere in **relazione di join** (diretta o indiretta)
  3. **Deve esistere almeno una sequenza di join che mette in relazione tutte le tuple contenenti le keyword**
     - Deve esistere t1 >< t2 >< ….>< tn dove t1, t2, … tn rappresentano tuple contenenti le keyword

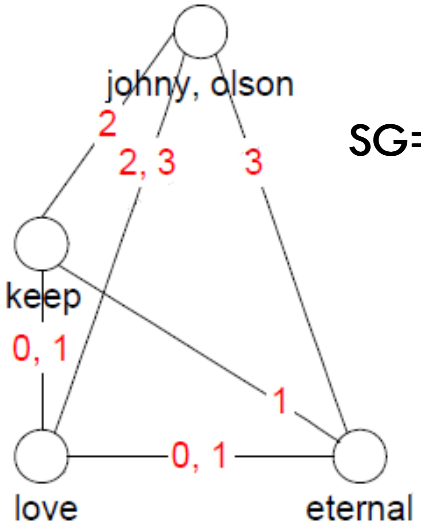
- **Di conseguenza:**
  1. Deve esistere un **sottografo SG** di KRG **contenente tutte le parole chiave** della query
  2. **SG** deve essere **completo**

Elaborazione query (2)

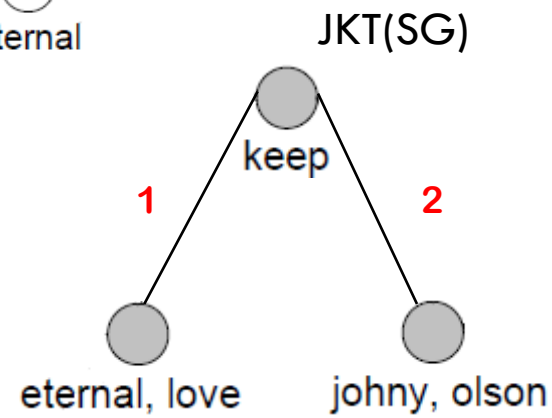
3. **Deve esistere almeno un albero (chiamato Join Keyword Tree JKT) di SG** tale che:
   1. Ogni **nodo di SG** deve essere mappato in almeno un **vertice di JKT**
   2. Ogni **arco di JKT** ha un'**unica distanza**
   3. Se due nodi di SG sono mappati in un **unico vertice di JKT**, quei due termini devono essere presenti nella **stessa tupla del DB** (distanza 0)
   4. Se **due vertici di JKT** sono **direttamente connessi** a **distanza d**, allora tutte le coppie dei corrispondenti **nodi di SG** devono essere collegate a **distanza d** in SG
   5. Se **due vertici x,y di JKT non** sono **direttamente connessi**, allora ogni tutte le coppie dei corrispondenti nodi di SG devono essere collegata a **distanza uguale al percorso che connette x e y**

- Il **sottografo** SG che rispetta le **3 proprietà** prende il nome di **Candidate Graph (CG)** in quanto indica che il DB è un possibile candidato a contenere un risultato
- **Esistenza di CG è condizione necessaria ma non sufficiente all'esistenza di un risultato** nel DB per la query data

Costruzione di JKT a partire da SG



$SG=\{n_{jonny,olson}, n_{keep}, n_{eternal}, n_{love}\}$



- **Si sceglie a caso un nodo di SG come radice dell'albero**
- **Si prende ciascun altro nodo e lo si inserisce nell'albero**
  - **Insieme** ad un nodo già presente, se esiste a **distanza 0 nel SG**
  - Come **figlio a distanza *d*** se in relazione di join in d passi (quindi a **distanza *d* in SG**)
    - Due termini appartenenti a nodi a distanza ***d*** in JKT devono corrispondere a nodi a distanza ***d*** in SG

$n_{jonny,olson}$ e $n_{eternal}$ hanno distanza 3 in SG → **OK!!!**

Non sufficienza del JKT

Artist

	aid	name
t_1	aid1	Smith **Anderson**
t_2	aid2	Johny Olson
t_{11}	aid3	Paulo **Anderson**

Song

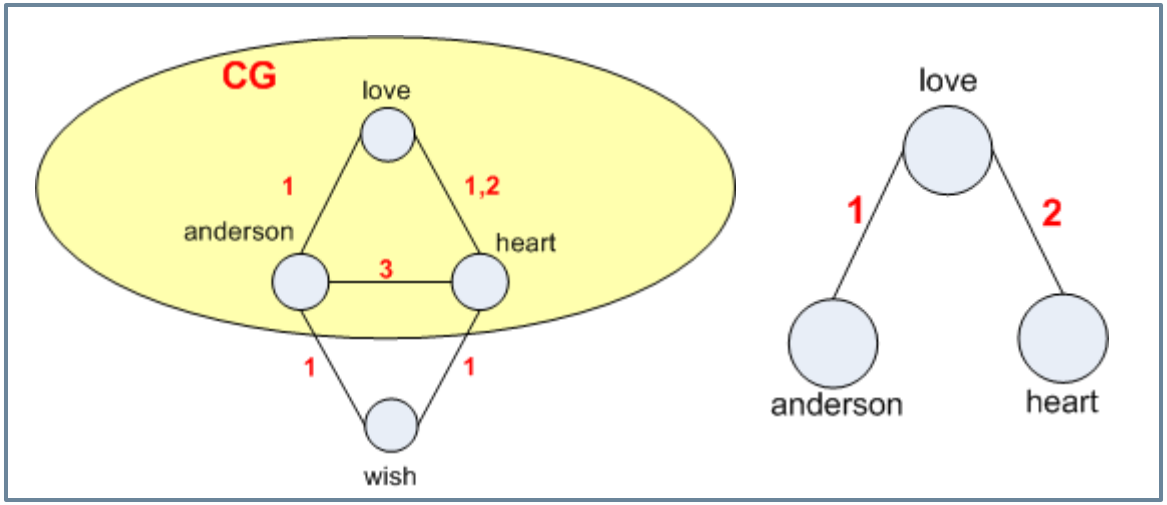
	sid	title	cid
t_3	sid1	let **Love** Lead the Way	cid1
t_4	sid2	Keep on Loving you	cid2
t_5	sid3	Please Hold our **Hearts** together	cid2
t_{12}	sid4	a Wish	cid3

Performs

	aid	sid
t_8	aid1	sid1
t_9	aid2	sid2
t_{10}	aid2	sid3
t_{14}	aid3	sid4

CD

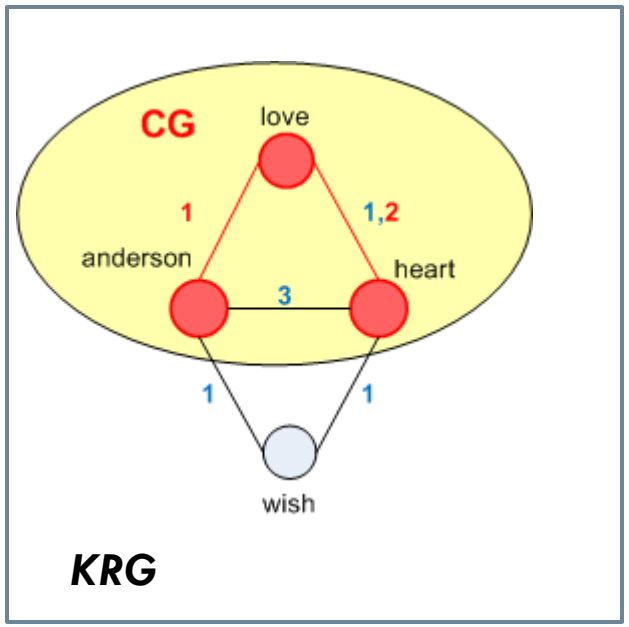
	cid	title
t_6	cid1	Bring me to Heaven
t_8	cid2	Eternal **Love**
t_{13}	cid3	something in my **Heart**



- ***q={anderson, love, heart}***
- **anderson** e **heart** sono connessi a **distanza 3**
- ma il grafo non ci dice se lo sono attraverso **wish** (che non fa parte della query) oppure attraverso **love**
- Caso in cui **esiste il JKT, ma la query non ha risultato**
- KRG non ha la nozione di ***percorso*** tra due nodi
- La probabilità che si verifichi tale situazione è molto bassa, soprattutto se le keyword sono molte

Ricerca dei Candidate Graph

q={anderson,love,heart}



- Data una query q, G-KS per <u>ogni DB</u>:
  1. Verifica l'**esistenza del sottografo SG** di KRG contenente tutte le keyword di q
  2. Verifica se **SG** è **completo**
  3. Determina se **esiste JKT(SG)**
     - Prova a costruirlo per determinare se è possibile
- Se un passo fallisce, l'algoritmo termina:
  - **il DB SICURAMENTE non conterrà un risultato!!** ☹
- Successo: **<u>SG è un Candidate Graph</u>**
  - **Il DB corrispondente potrebbe contenere un risultato (con buone probabilità)** 😐

Selezione TOP-K database (AND)

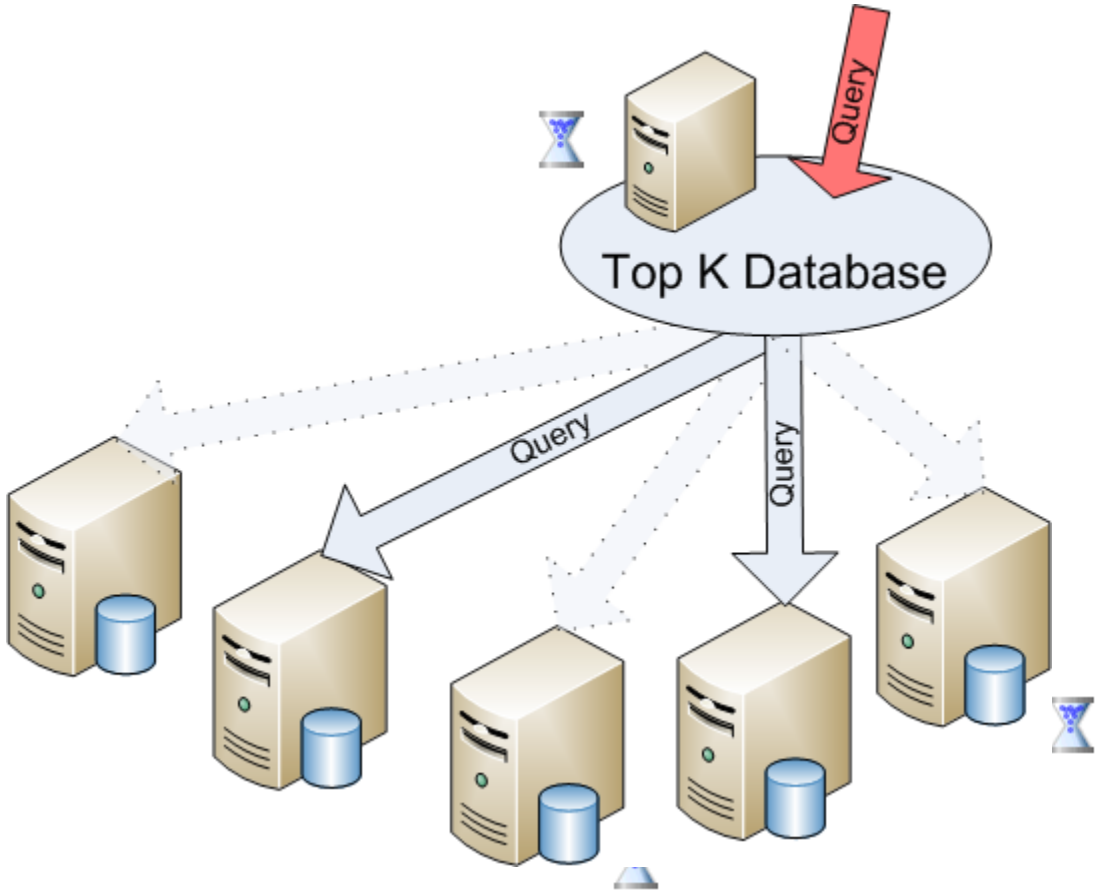
- Si attribuisce uno **score** a tutti i DB per cui esiste un CG per quella **query**
  - Lo score indica la **somiglianza** tra un DB e la query

$$Score(DB,q) = \sum_{\{k_i,k_j\}\subseteq q, i\neq j} Score(k_i,k_j)$$

$$Score(k_i,k_j) = w_i \cdot w_j \cdot \sum_d w_{ij}(d)$$

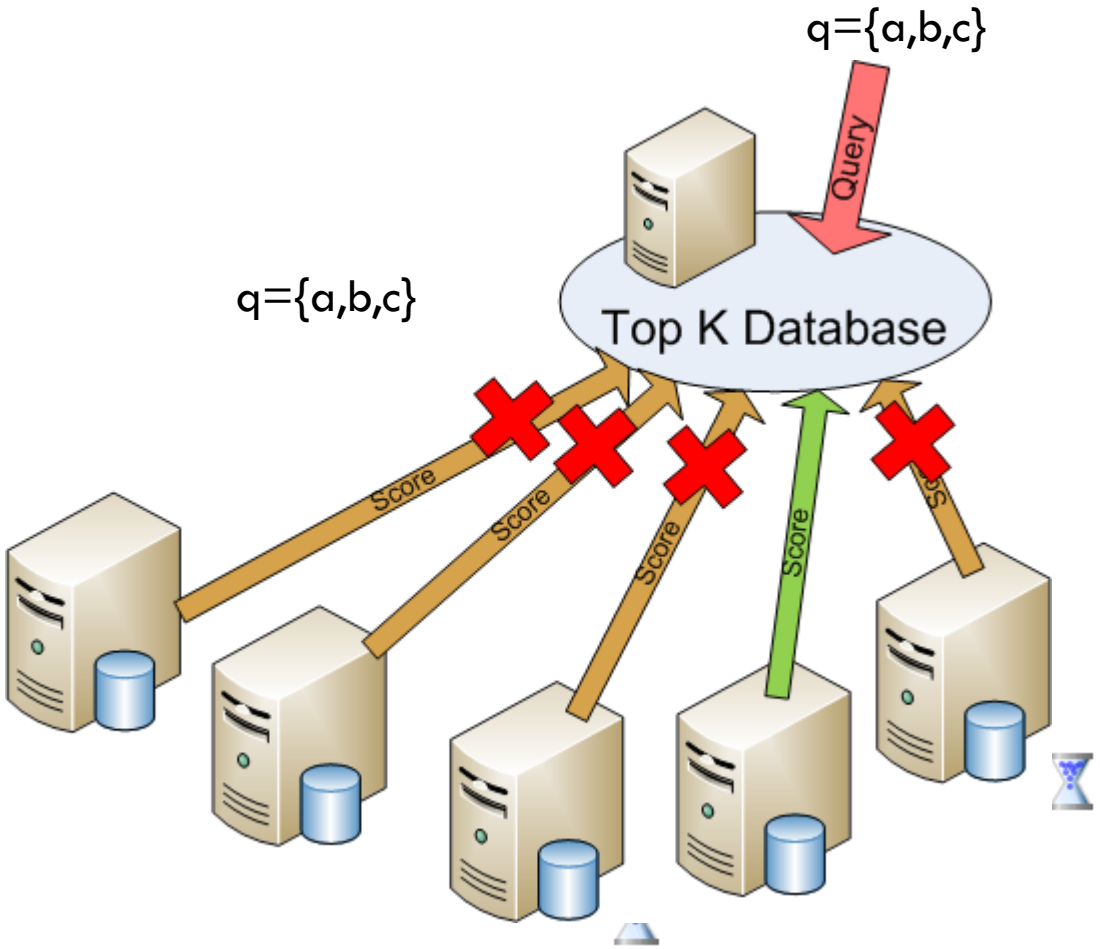
  - $\mathbf{k_i}$ keyword di q
  - $\mathbf{W_i}$ peso del nodo $\mathbf{n_i}$ rappresentante la keyword $\mathbf{k_i}$
  - $\mathbf{W_{ij}(d)}$ peso dell'arco che connette $\mathbf{n_i}$ e $\mathbf{n_j}$ a distanza **d**

- **I Top-K DB sono i k DB con lo score più alto**



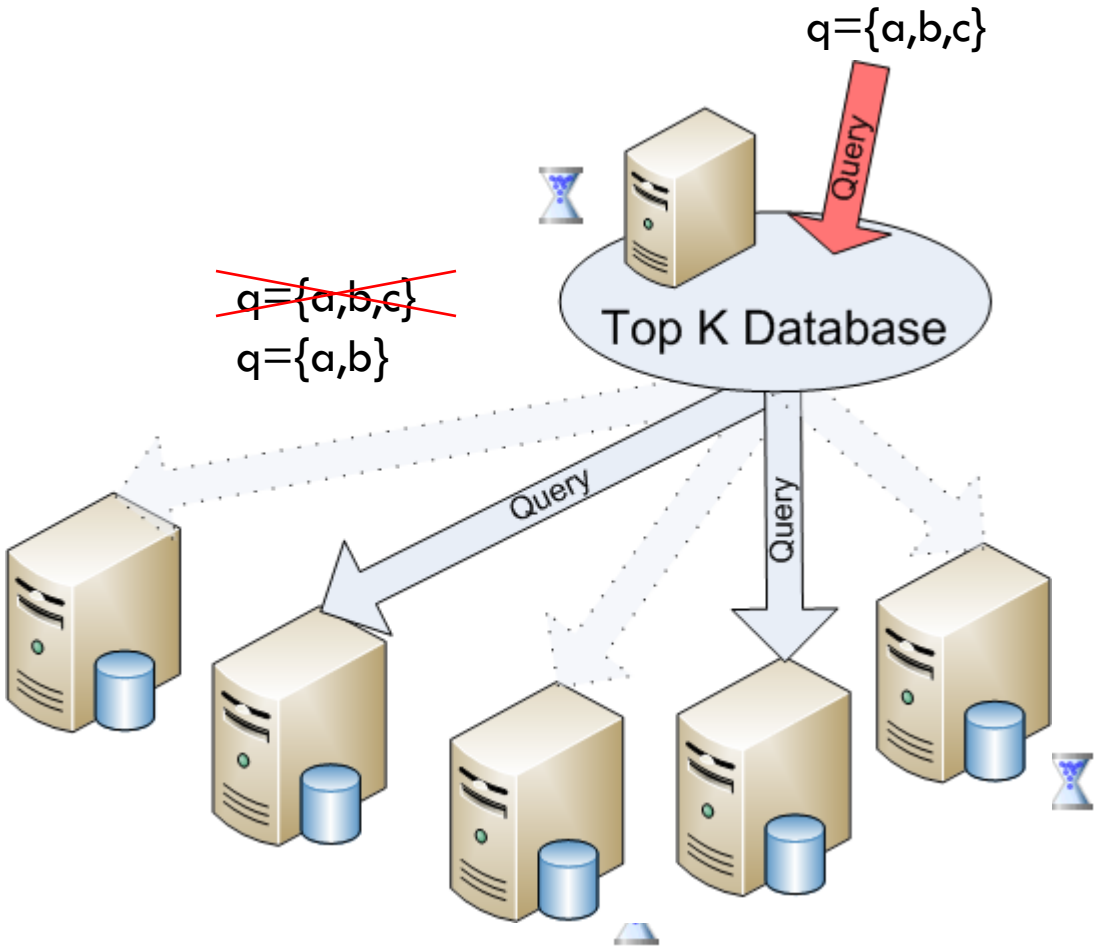
Selezione TOP-K database (OR)

- **k** = numero di keyword della query
- Si assegna uno **score** a tutti i DB per cui esiste un CG per quella query
- Per tutti i DB ancora **privi di score** per la query:
  - Si ripete la query considerando **k-1** keyword
  - In caso positivo si calcola lo **score** corrispondente
  - Si **decrementa di nuovo k**, e così via, finchè non si raggiunge una certa soglia
- **I top-K DB si calcolano sempre in base allo score**



Selezione TOP-K database (OR)

- **k** = numero di keyword della query
- Si assegna uno **score** a tutti i DB per cui esiste un CG per quella query
- Per tutti i DB ancora **privi di score** per la query:
  - Si ripete la query considerando **k-1** keyword
  - In caso positivo si calcola lo **score** corrispondente
  - Si **decrementa di nuovo k**, e così via, finchè non si raggiunge una certa soglia
- **I top-K DB si calcolano sempre in base allo score**



G-KS vs M-KS

- **Popular Terms Problem**
  - M-KS conta le **semplici occorrenze** di coppie di termini in relazione tra di loro
    - Considerando tutti i termini ugualmente importanti, per una data query un DB otterrà un punteggio elevato in presenza di popular terms
  - G-KS risolve tale problema utilizzando la **frequenza inversa** nel calcolo dei pesi di nodi e archi
- **Compressione**
  - M-KS **rappresenta esplicitamente le relazioni** tra ogni coppia di termini
  - G-KS mediante **compressione** riduce il numero di entry, ottenendo migliori prestazioni in spazio e tempo.
- **Falsi positivi**
  - G-KS, andando oltre le **relazioni binarie**, incorre in un **numero minore di falsi positivi** per query a più di 2 keyword

G-KS vs M-KS: valutazione (1)

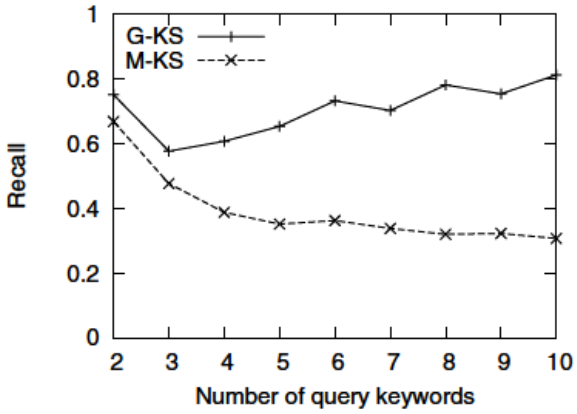
Pre-processing cost:

	Time (Hours)	Space (MB)
M-KS	7.26	438.12
G-KS	4.97	362.59
Improvement	31.54%	17.24%

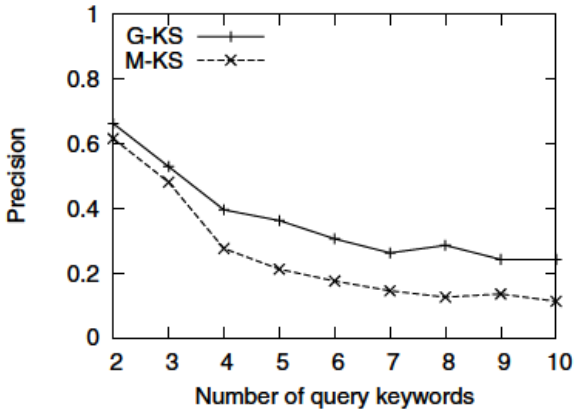
Client (selezione TOP-K): Intel Pentium 4 2.4 GHz
Server (gestione summary):Intel Xeon MP CPU 3.00 GHz.

81 database di 5000 record in 4-5 tabelle
Teconologia: MySQL 4.1.7

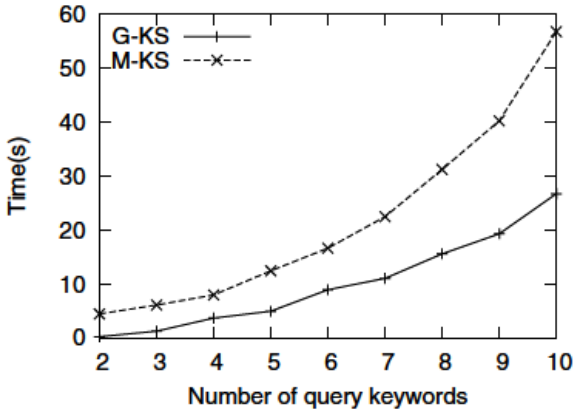
G-KS vs M-KS: valutazione (2)



Recall



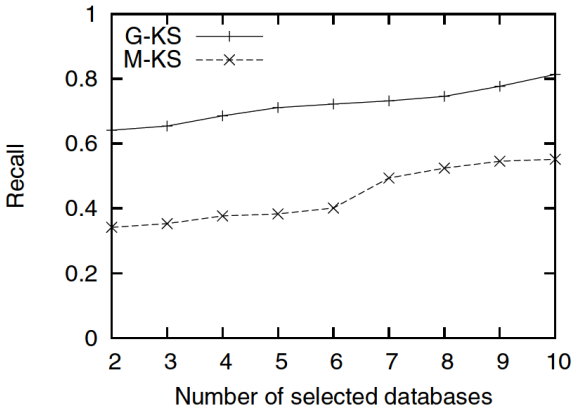
Precision



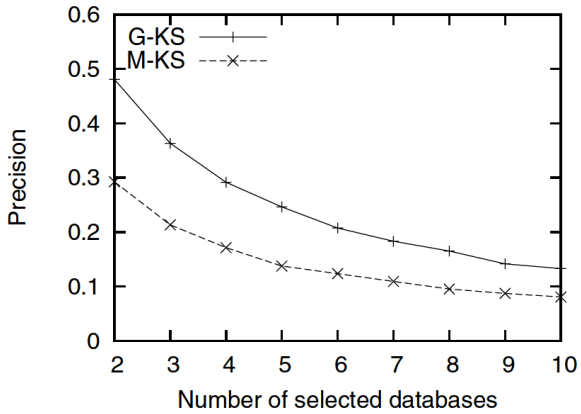
Query Response Time

- All'aumentare del **numero delle keyword,** il **recall** di M-KS crolla a causa del limite dato dall'utilizzo delle **relazioni binarie**

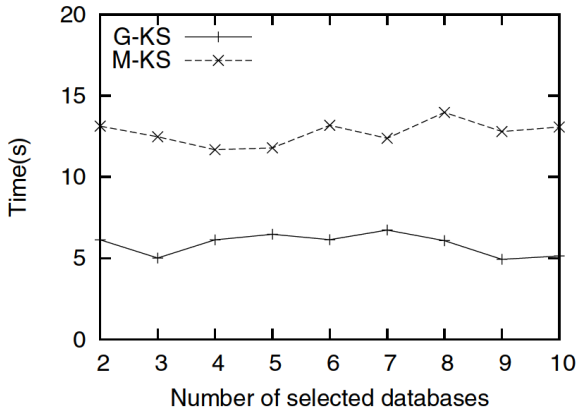
G-KS vs M-KS: valutazione (3)



Recall



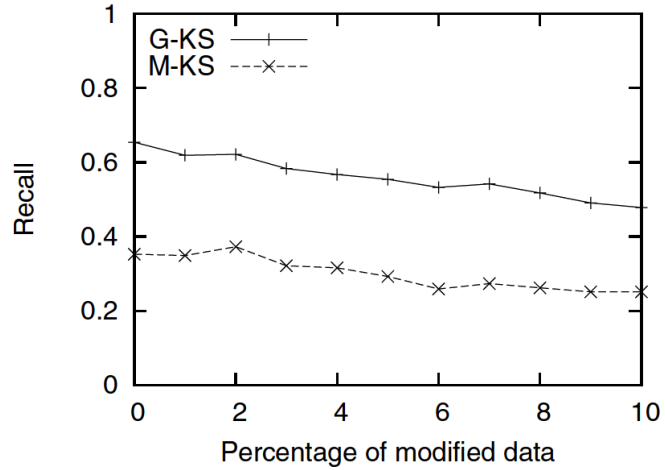
Precision

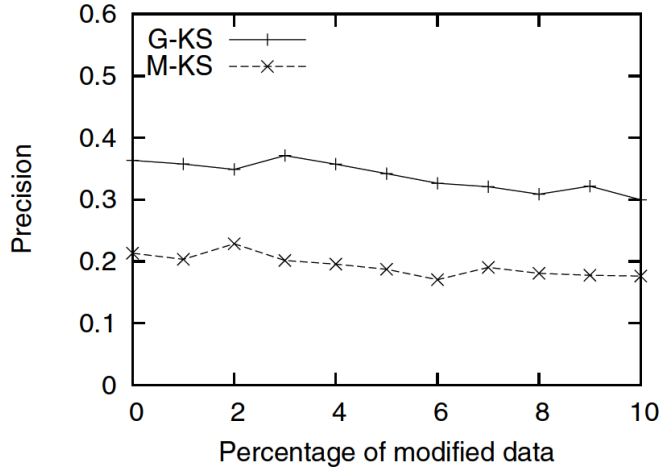


Query response time

- **Al crescere di K** (numero di DB in output), **il tempo di esecuzione dell'algoritmo è pressappoco costante** poiché il tempo di calcolo dei TOP-K DB è trascurabile rispetto al tempo di estrazione dei summaries

G-KS vs M-KS: valutazione (4)





- Le **performance** di **G-KS** quando il **10% delle modifiche del DB non sono state ancora riportate sul KRG** sono comunque **migliori** delle performance di **M-KS** anche se questo **aggiorna costantemente la sua KRM**

Conclusioni

- **G-KS**:
  - Metodo per la **selezione dei TOP-K database più promettenti** per l'esecuzione di una KS query
  - Rappresentazione delle relazioni mediante **grafi**
  - Calcolo di **pesi** di archi e nodi basato sui concetti di frequenza e frequenza inversa
- Confronto con **M-KS**:
  - Migliori risultati in termini di:
    - **efficacia** (**precision** e **recall**)
    - **efficienza** (in **spazio** e **tempo**)

Gruppo 9


धन्यवाद
Hindi
多謝
Traditional Chinese
ขอบคุณ
Thai
Спасибо
Russian
Gracias
Spanish
شكراً
Arabic
Thank You
Obrigado
Brazilian Portuguese
Grazie
Italian
Danke
German
Merci
French
多谢
Simplified Chinese
Multumesc
Romanian
감사합니다
Korean
ありがとうございました
Japanese