

HTE – Távközlési Klub

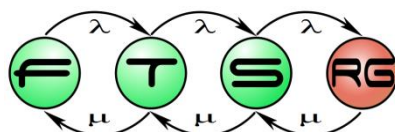
2019. február 28.

Gráflekérdezések leképzése relációs adatbázisokra

Elekes Márton, Antal János Benjamin

Konzulens: Szárnyas Gábor, Marton József (TMIT)

**Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport**



**Magyar Tudományos
Akadémia**

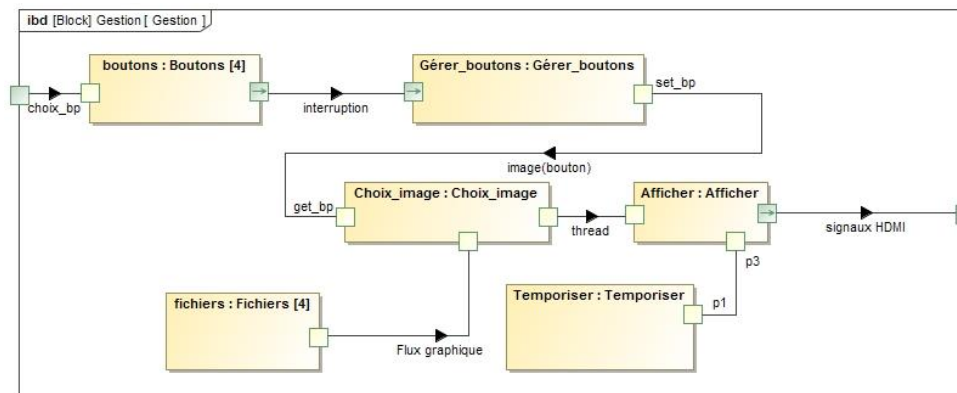
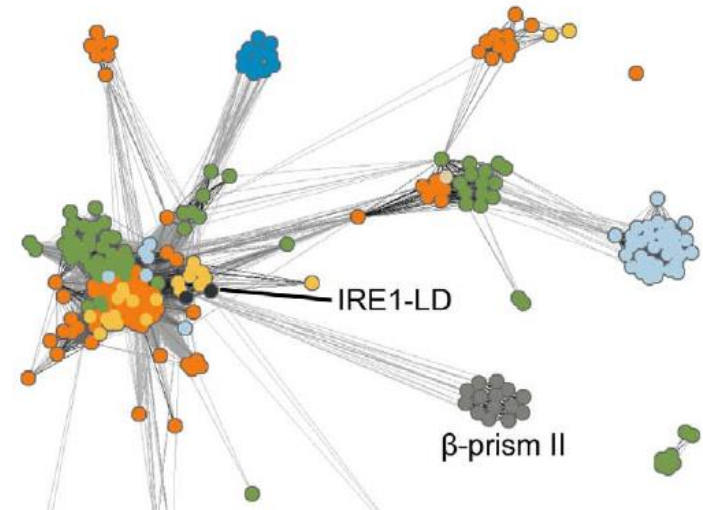


Az Emberi Erőforrások Minisztériuma ÚNKP-18-2 kódszámú Új Nemzeti Kiválóság Programjának támogatásával készült.

Az MTA-BME Lendület Kiberfizikai Rendszerek Kutatócsoport szakmai támogatásával készült.

Motiváció

- Gráfok: műszaki és tudományos problémákban gyakoriak
- Cél: feldolgozáshoz hatékony eszközkészlet



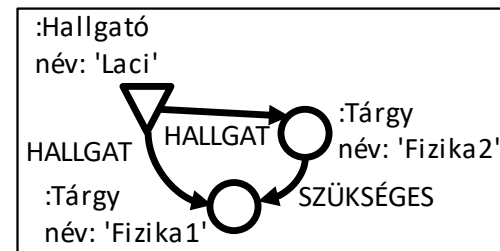
β-Propeller Blades as Ancestral Peptides in Protein Evolution - Scientific Figure on ResearchGate.

Available from: https://www.researchgate.net/figure/Cluster-map-of-b-propellers-and-their-potential-homologs-Dots-represent-proteins_fig9_258062452 [accessed 27 Feb, 2019]

Motiváció

■ Gráfadatbázisok

- Gráf alapú modellezés: intuitív
- Gráflekérdező nyelv: olvasható, tömör



■ Hagyományos relációs adatbázis

- Üzleti adatok tárolásának fő módja
- Hatékony, stabil



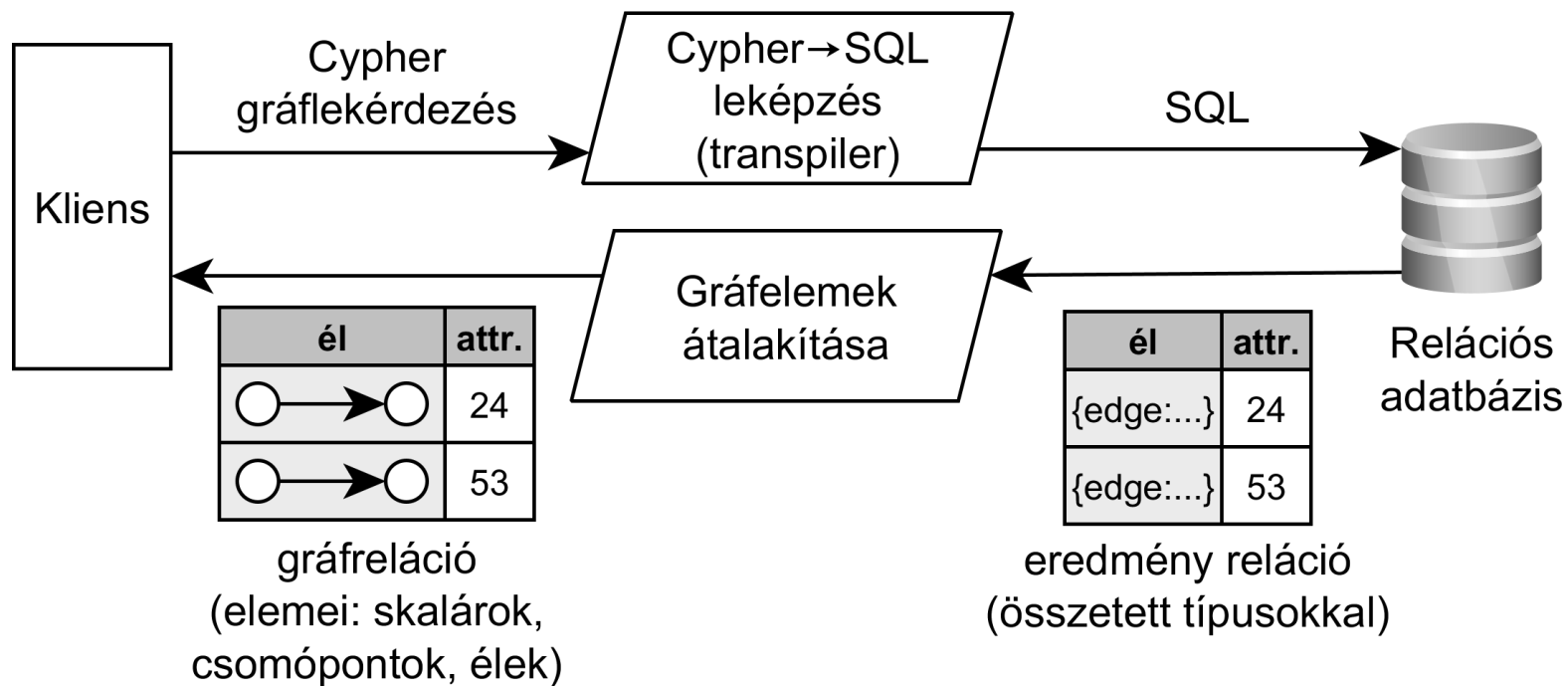
Relációk	
Oszlop1	Oszlop2
1	A
2	B

■ Nem létezik eszköz

- Relációs adatbázisban tárolt (legacy) adatok
- Gráfként való kezelésére
- Gráflekérdező nyelvek gazdag funkcionalitását támogatva

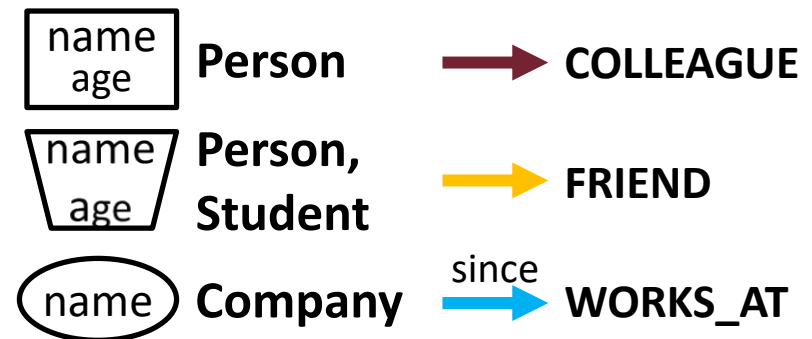
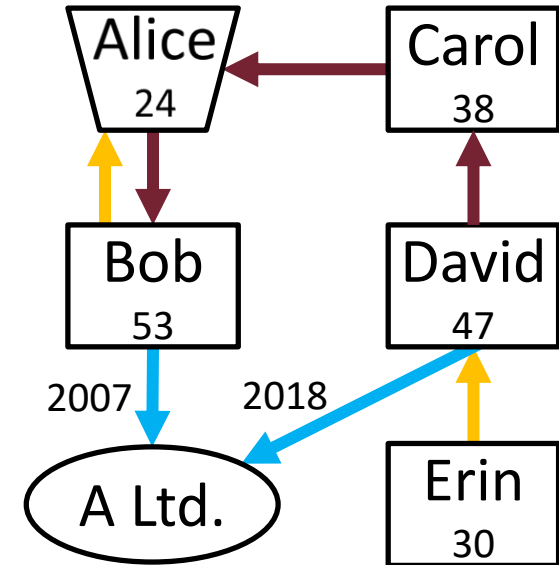
Architektúra

- Cypher gráflekérdezések leképzése SQL nyelvre
 1. Gráflekérdezések megfogalmazása
 2. Lekérdezés futtatása a meglévő relációs adatbázisban
 3. Eredmények megjelenítése gráfként



Tulajdonsággráf

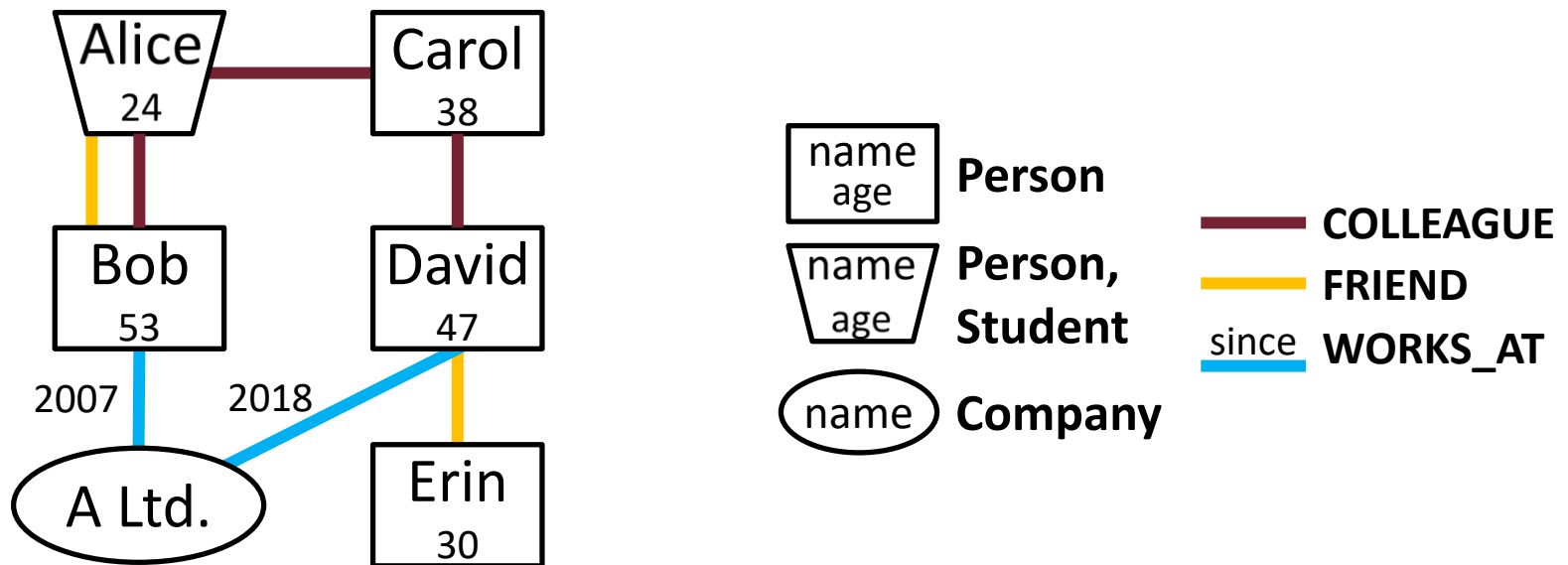
- Gráfadatbázisok adatmodellje
- Tankönyvi gráf: $G = (V, E)$
- Kiterjesztve:
 - Címkézett csúcsok
 - Típusos élek
 - Tulajdonságok
- Implicit séma
 - Adatok felépítése adja
 - Bármikor bővíthető



Gráflekérdezések

- Cypher nyelvű gráflekérdezés
 - Gráfmintával leírható, gráfbejárást végez
 - Alice kollégái és az ő kollégáik:

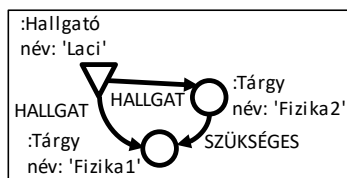
```
MATCH (p1:Person {name: 'Alice'})-[c:COLLEAGUE*1..2]-(p2:Person)
RETURN p2.name
```



Adatbázisok közötti megfeleltetés

1) Adatsémák közti megfeleltetés

- Meglévő relációs séma esetében (meglévő adatbázis)



Relációk	
Oszlop1	Oszlop2
1	A
2	B

2) Lekérdezések leképzése (transpilation)

- Cypher nyelvi elemek leképzése
- Szükséges adatforrások keresése 1) alapján

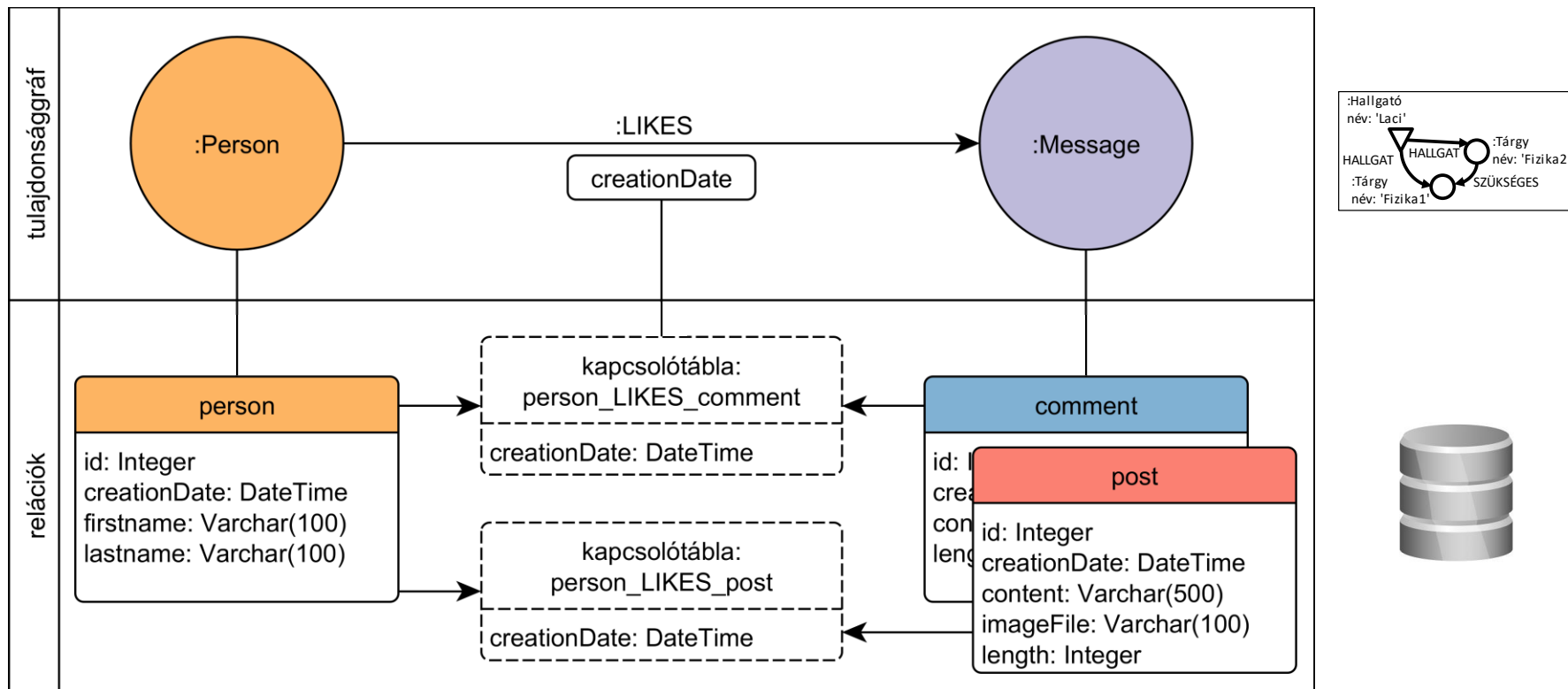
```
MATCH (p1:Person ...)  
  -[c:COLL]-(p2:Person)  
RETURN p2.name
```



```
SELECT p2.name FROM person p1  
  JOIN colleague c ON p1.id = c.p1  
  JOIN person p2 ON c.p2 = p2.id  
...  
UNION ALL ...
```

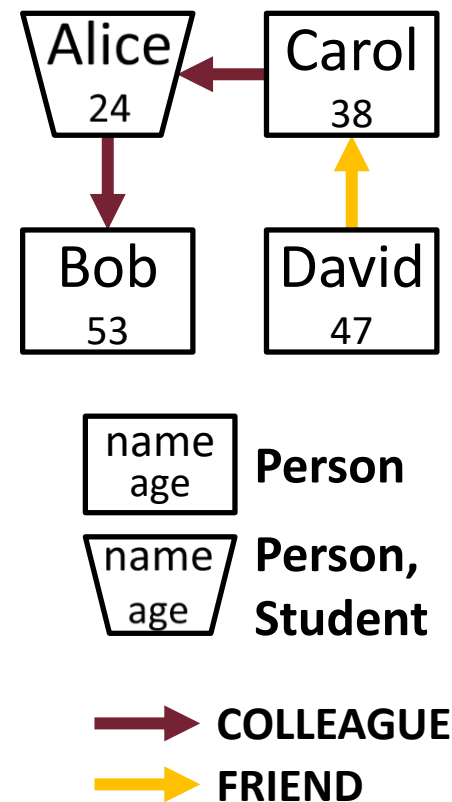
Adatsémák közti megfeleltetés

- Meglévő relációs séma és gráfséma megfeleltetése
 - gTop sémaleíró nyelv



Gráfminták leképzésének kihívásai

- Egyszerű gráfminták megfelelői több SQL allekérdezésből állhatnak
 - Irányított élek irányítatlan lekérdezése:
 - Élek felsorolása mindkét irányban (2×)
 - Változó hosszú útvonalak ($1 \dots U$, $*$):
 - Utak felsorolása minden hosszra ($U \times$)
 - WITH RECURSIVE használata
 - Több lehetséges csúcs-/éltípus:
 - Minden forrástábla felsorolása ($N \times$)
- $2 \times U \times N$ db. SQL lekérdezés uniója



```
MATCH (p1:Person ...) - [:COLLEAGUE | :FRIEND*1..2] - (p2:Person)
```

Komplex gráfmenták leképzése

```
MATCH (:Person {id:$personId})-[:KNOWS]-
(friend:Person)<-[:HAS_CREATOR]-(message:Message)
WHERE message.creationDate <= $maxDate
RETURN
    friend.id AS personId,
    friend.firstName AS personFirstName,
    friend.lastName AS personLastName,
    message.id AS postOrCommentId,
```

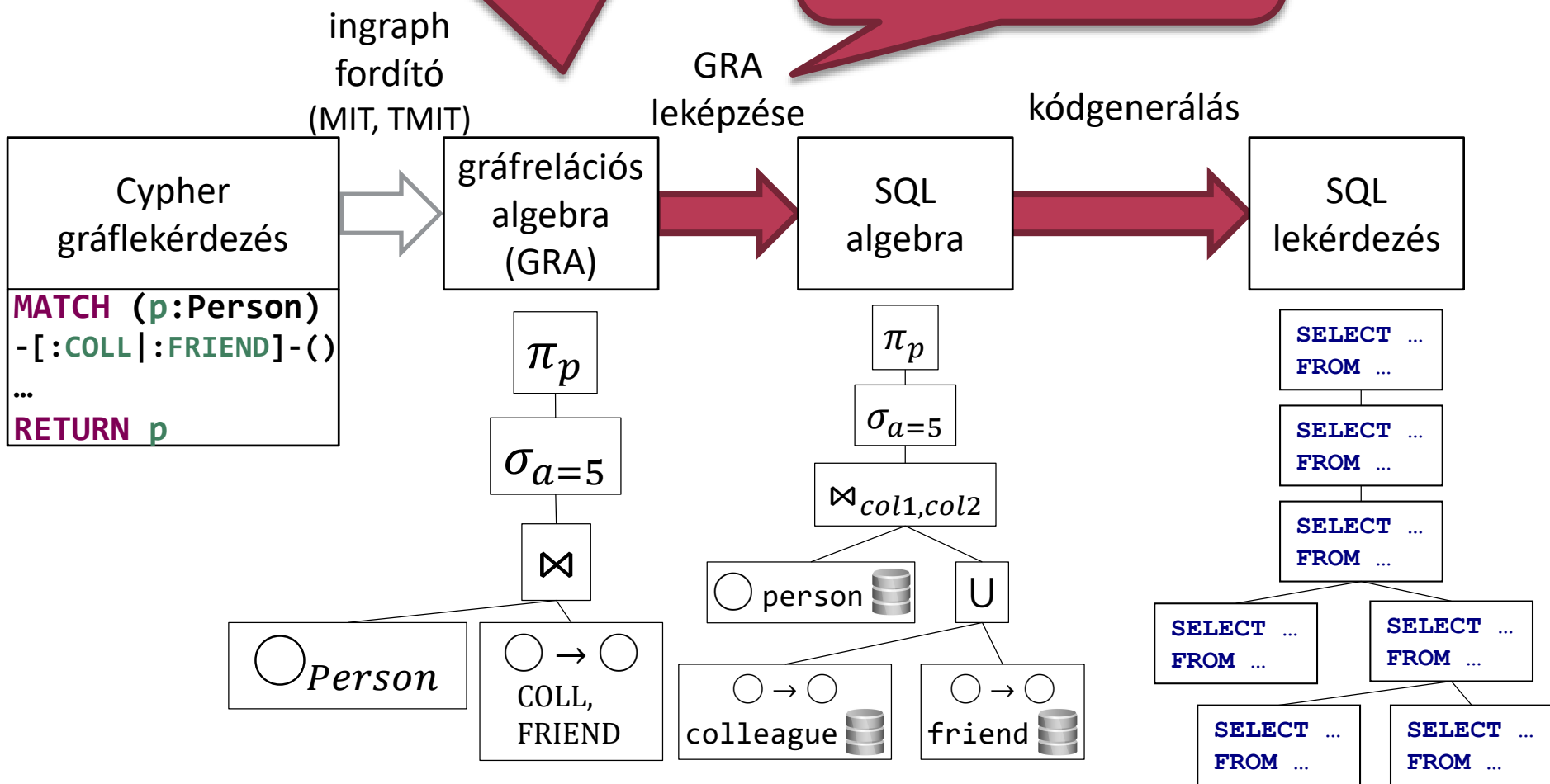
```
CASE exists(message.content)
    WHEN true THEN message.content
    ELSE message.imageFile
END AS postOrCommentContent,
message.creationDate AS postOrCommentCreationDate
ORDER BY postOrCommentCreationDate DESC,
toInteger(postOrCommentId) ASC
LIMIT 20
```

```
WITH
q0 AS
  (
    -- GetVerticesWithGTop
    SELECT
      ROW(0, p_personid)::vertex_type AS "_e186#0",
      "p_personid" AS "_e186.id#0"
    FROM person),
q1 AS
  (
    -- Selection
    SELECT * FROM q0 AS subquery
    WHERE ("_e186.id#0" = :personId)),
q2 AS
  (
    -- GetEdgesWithGTop
    SELECT ROW(0, edgeTable."k_personid")::vertex_type AS "_e186#0", ROW(0, edgeTable."k_personid",
edgeTable."k_person2id")::edge_type AS "_e187#0", ROW(0, edgeTable."k_person2id")::vertex_type AS "friend#2",
toTable."p_personid" AS "friend.id#2", toTable."p_firstname" AS "friend.firstName#1", toTable."p_lastname" AS
"friend.lastName#2"
    FROM "knows" edgeTable
    JOIN "person" toTable ON (edgeTable."k_person2id" = toTable."p_personid")),
q3 AS
  (
    -- GetEdgesWithGTop
    SELECT ROW(0, edgeTable."k_personid")::vertex_type AS "friend#2", ROW(0, edgeTable."k_personid",
edgeTable."k_person2id")::edge_type AS "_e187#0", ROW(0, edgeTable."k_person2id")::vertex_type AS "_e186#0",
fromTable."p_personid" AS "friend.id#2", fromTable."p_firstname" AS "friend.firstName#1",
fromTable."p_lastname" AS "friend.lastName#2"
    FROM "knows" edgeTable
    JOIN "person" fromTable ON (fromTable."p_personid" = edgeTable."k_personid")),
q4 AS
  (
    -- UnionAll
    SELECT "_e186#0", "_e187#0", "friend#2", "friend.id#2", "friend.firstName#1", "friend.lastName#2" FROM q2
    UNION ALL
    SELECT "_e186#0", "_e187#0", "friend#2", "friend.id#2", "friend.firstName#1", "friend.lastName#2" FROM q3),
q5 AS
  (
    -- EquiJoinLike
    SELECT left_query."_e186#0", left_query."_e186.id#0", right_query."friend#2", right_query."friend.id#2",
right_query."_e187#0", right_query."friend.lastName#2", right_query."friend.firstName#1" FROM
    q1 AS left_query
    INNER JOIN
    q4 AS right_query
    ON left_query."_e186#0" = right_query."_e186#0"),
q6 AS
  (
    -- GetEdgesWithGTop
    SELECT ROW(6, fromTable."m_messageid")::vertex_type AS "message#17", ROW(8, fromTable."m_messageid",
fromTable."m_creatorid")::edge_type AS "_e188#0", ROW(0, fromTable."m_creatorid")::vertex_type AS "friend#2",
fromTable."m_messageid" AS "message.id#2", fromTable."m_content" AS "message.content#2",
fromTable."m_creationdate" AS "message.creationDate#13"
    FROM "message" fromTable
    WHERE (fromTable."m_c_replyof" IS NULL)),
q7 AS
  (
    -- GetEdgesWithGTop
    SELECT ROW(6, fromTable."m_messageid")::vertex_type AS "message#17", ROW(8, fromTable."m_messageid",
fromTable."m_creatorid")::edge_type AS "_e188#0", ROW(0, fromTable."m_creatorid")::vertex_type AS "friend#2",
fromTable."m_messageid" AS "message.id#2", fromTable."m_content" AS "message.content#2",
fromTable."m_creationdate" AS "message.creationDate#13"
    FROM "message" fromTable
    WHERE (fromTable."m_c_replyof" IS NOT NULL)),
q8 AS
  (
    -- UnionAll
    SELECT "message#17", "_e188#0", "friend#2", "message.id#2", "message.content#2", "message.imageFile#0",
"message.creationDate#13" FROM q6
    UNION ALL
    SELECT "message#17", "_e188#0", "friend#2", "message.id#2", "message.content#2", NULL AS "message.imageFile#0",
"message.creationDate#13" FROM q7),
q9 AS
  (
    -- EquiJoinLike
    SELECT left_query."_e186#0", left_query."_e186.id#0", left_query."_e187#0", left_query."friend#2",
left_query."friend.id#2", left_query."friend.firstName#1", left_query."friend.lastName#2",
right_query."message#17", right_query."message.id#2", right_query."message.imageFile#0", right_query."_e188#0",
right_query."message.creationDate#13", right_query."message.content#2" FROM
    q5 AS left_query
    INNER JOIN
    q8 AS right_query
    ON left_query."friend#2" = right_query."friend#2"),
q10 AS
  (
    -- AllDifferent
    SELECT * FROM q9 AS subquery
    WHERE is_unique(ARRAY[::edge_type[] || "_e188#0" || "_e187#0"])),
q11 AS
  (
    -- Selection
    SELECT * FROM q10 AS subquery
    WHERE ("message.creationDate#13" <= :maxDate)),
q12 AS
  (
    -- Projection
    SELECT "friend.id#2" AS "personId#0", "friend.firstName#1" AS "personFirstName#0", "friend.lastName#2" AS
"personLastName#0", "message.id#2" AS "postOrCommentId#0", CASE WHEN ("message.content#2" IS NOT NULL = true) THEN
"message.content#2"
    ELSE "message.imageFile#0"
    END AS "postOrCommentContent#0", "message.creationDate#13" AS "postOrCommentCreationDate#0"
    FROM q11 AS subquery),
q13 AS
  (
    -- SortAndTop
    SELECT * FROM q12 AS subquery
    ORDER BY "postOrCommentCreationDate#0" DESC NULLS LAST, ("postOrCommentId#0")::BIGINT ASC NULLS FIRST
    LIMIT 20)
SELECT "personId#0" AS "personId", "personFirstName#0" AS "personFirstName", "personLastName#0" AS
"personLastName", "postOrCommentId#0" AS "postOrCommentId", "postOrCommentContent#0" AS "postOrCommentContent",
"postOrCommentCreationDate#0" AS "postOrCommentCreationDate"
    FROM q13 AS subquery
```

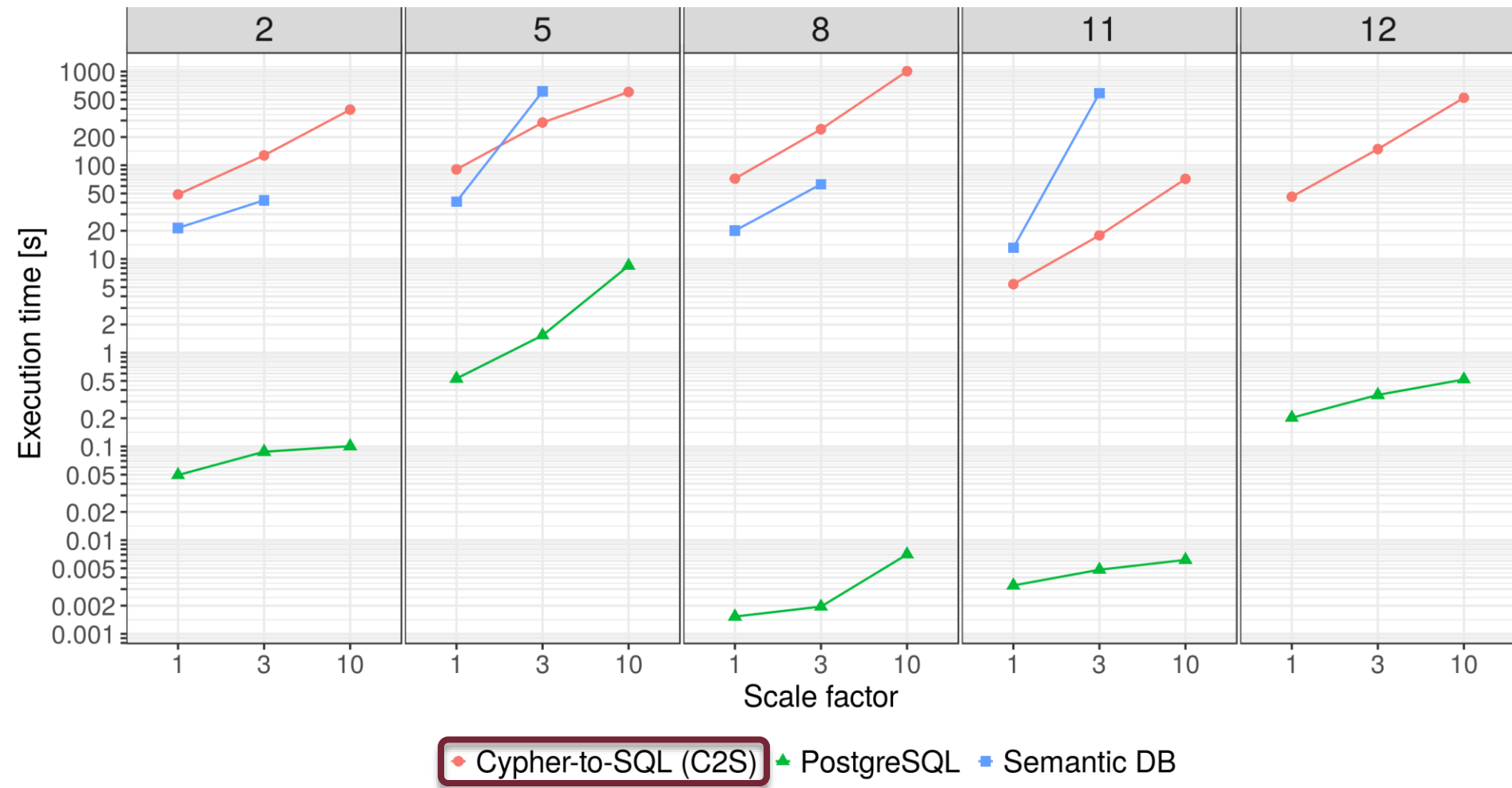
Cypher leképzése SQL-re

Gráfspecifikus operátorokkal
kiterjesztett relációs algebra

Metaadatok kigyűjtése,
komplex műveletek
szétbontása

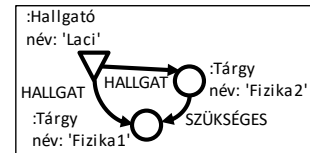


Teljesítménymérési eredmények



Összefoglalás

- Teljesítmény összemérhető egy kereskedelmi szemantikus adatbázis-kezelővel
- További optimalizációk várhatóak
- A Cypher-to-SQL implementáció
 - megoldja a Cypher lekérdezések leképezését az említett kihívásokkal együtt,
 - így egyszerre használható
 - egy intuitív gráflekérdezőnyelv és
 - a hatékony relációs adatbázis-kezelő



Relációk	
Oszlop1	Oszlop2
1	A
2	B



Az Emberi Erőforrások Minisztériuma ÚNKP-18-2 kódszámú Új Nemzeti Kiválóság Programjának támogatásával készült.



Magyar Tudományos Akadémia

Az MTA-BME Lendület Kiberfizikai Rendszerek Kutatócsoport szakmai támogatásával készült.