Pengerjaan Soal UTS PDL (IF4040)

Tata cara yang langsung dikerjakan pada soal, langkah demi langkah:
UTS sample dan kasus prediksi KursusKita

Dokumen ini pendamping *Analisis_Soal_UTS_PDL*, *UTS_PDL*, dan *UTS_Sample_PDL*.

Bedanya dengan Analisis_Soal_UTS_PDL. Di catatan analisis, tata cara dan jawaban ditulis terpisah. Di sini **tiap langkah tata cara langsung dikerjakan pada soal**, dan hasil sementaranya ditunjukkan. Jawaban akhir terbentuk sedikit demi sedikit, sehingga terlihat dari mana setiap tabel, dokumen, kolom, edge, atau elemen berasal.

Cara membaca tiap kotak langkah:

- **Aturan:** aturan dari materi yang dipakai di langkah itu.
- **Dikerjakan:** aturan diterapkan ke kasus.
- **Hasil sementara:** potongan jawaban setelah langkah itu.

Di akhir tiap soal ada kotak hijau **jawaban akhir**, yaitu yang ditulis di lembar ujian.

Konvensi skema relasional: atribut bergaris bawah adalah primary key; `FK a -> T` berarti kolom `a` merujuk tabel `T`.

Daftar Isi

1 Kartu Hafalan Langkah **3**

2 UTS Sample: Aplikasi Media Sosial **4**

2.1 Langkah Nol: B-T-T-A 4
2.2 Soal 1: Model Relasional 5
2.3 Soal 2a: Desain Dokumen 8
2.4 Soal 2b: Desain Column-Family (BigTable-like) 9
2.5 Soal 2c: Desain Graf 11
2.6 Soal 3: Use Case Key-Value 12
2.7 Soal 4a: XML (DTD) 13
2.8 Soal 4b: XQuery 14

3 Kasus Prediksi: KursusKita **14**

3.1 Langkah Nol 16
3.2 Soal 1: Model Relasional 16
3.3 Soal 2a: Dokumen 18
3.4 Soal 2b: Column-Family 20
3.5 Soal 2c: Graf 21
3.6 Soal 3: Key-Value 23

3.7 Soal 4: DTD, XQuery, XPath 23
3.8 Soal 5: Model Paling Cocok per Query 24

4 Ringkasan Inti **24**

1 Kartu Hafalan Langkah

Soal	Langkah (hafalkan urutannya)
Langkah nol (sekali, untuk semua soal)	**B-T-T-A**: **B**aca semua sub-soal → **T**andai konstruksi ER → **T**ulis ulang query (entity, atribut, filter, urutan) → **A**sumsi
Relasional	R1 strong entity → R2 atribut (komposit, multivalued, derived, opsional) → R3 weak entity → R4 relasi 1:N/1:1 → R5 relasi M:N dan rekursif → R6 ternary/aggregation → R7 ISA → R8 constraint → R9 optimasi
Dokumen	D1 collection dari query → D2 uji 4 faktor tiap relasi (embed/reference) → D3 snapshot atribut → D4 ISA dan rekursif → D5 tulis JSON + index → D6 cek tiap query
Column-family	C1 satu tabel per query → C2 row key dari kriteria cari → C3 CF = kolom yang dibaca bersama → C4 relasi jadi kolom dinamis → C5 tabel index/nilai turunan → C6 cek 7 guideline
Graf	G1 query → G2 vertex → G3 edge (label, arah, properti) → G4 query jadi jalur
Key-value	K1 cari akses kecil, sering, lewat satu key → K2 key → K3 value → K4 operasi + alasan
XML	X1 bentuk hasil query → X2 root → X3 atribut vs elemen → X4 kemunculan → X5 cek data untuk soal berikutnya → X6 tulis DTD/XSD
XQuery	F1 `for` (iterasi apa) → F2 `where` (filter) → F3 `order by` → F4 `return` (bentuk hasil)
Pilih model	P1 ciri tiap query → P2 cocokkan dengan model → P3 alasan

Empat faktor embed vs reference (dipakai di D2)

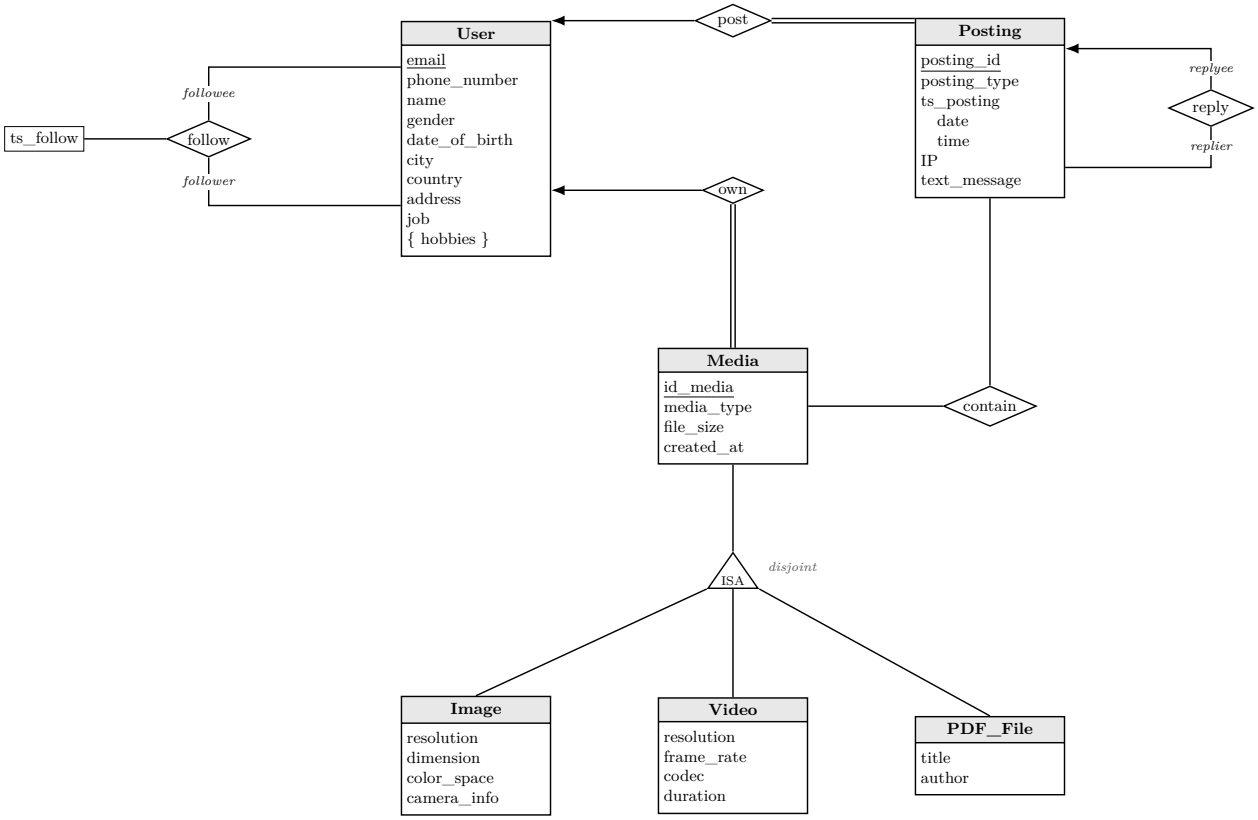
Faktor	Embed bila...	Reference bila...
Pola akses	Sering diakses bersama induk	Diakses mandiri
Ukuran, pertumbuhan	Kecil atau tetap	Besar atau terus bertambah
Frekuensi update	Perlu update atomik bersama induk	Sering diupdate sendiri
Kompleksitas	Hierarki sederhana	Kompleks atau berevolusi

Aturan cepat: 1:1 → embed; 1:N anak kecil → embed array; M:N atau anak besar → reference.

Tujuh guideline column-family (dipakai di C6)

(1) denormalize instead of join; (2) valueless columns; (3) nama kolom dan nilai sama-sama menyimpan data; (4) satu entity satu row; (5) hindari hotspotting; (6) simpan versi secukupnya; (7) hindari struktur kompleks di value.

2 UTS Sample: Aplikasi Media Sosial



Ringkasan soal

Info tambahan: address, hobi, job boleh kosong; atribut user lain wajib. `media_type`: image, video, pdf file, dll. (ada jenis lain). `posting_type`: normal atau reply.
Query sering: (1) atribut user email, name, phone_number, gender, date_of_birth + daftar follower (email, nama); (2) semua jenis media ditampilkan bersama; (3) posting tertentu seorang pengguna + semua media terkait (id_media, media_type, file_size, created_at) + semua posting_id reply-nya.
Soal: 1 relasional; 2a dokumen, 2b column-family, 2c graf; 3 use case key-value; 4a XML Schema atau DTD untuk hasil query 3; 4b XQuery posting milik `fazat@informatika.org`.

2.1 Langkah Nol: B-T-T-A

0.1 Baca semua sub-soal dulu

Aturan: soal saling menyambung; data yang dibutuhkan soal belakang harus sudah disiapkan di desain depan.

Dikerjakan: soal 4a meminta XML untuk hasil query 3, dan soal 4b memfilter posting berdasarkan **email** pengguna.

Hasil sementara:

Catatan untuk nanti: **XML di 4a harus memuat email penulis**, walaupun query 3 tidak menyebut email.

0.2 Tandai konstruksi ER

Aturan: cari panah (sisi "one"), garis ganda (total), ISA (disjoint/overlapping, total/partial), multivalued, komposit, derived, rekursif, weak, ternary, aggregation.

Hasil sementara:

Elemen	Konstruksi	Akibatnya nanti
User.{hobbies}	Multivalued, opsional	Tabel/array terpisah, boleh kosong
User.address, job	Opsional	NULL / tidak ada
Posting.ts_posting (date, time)	Komposit	Dua kolom/subelemen
post: panah ke User, Posting garis ganda	Many-to-one, Posting total	FK NOT NULL di Posting
own: panah ke User, Media garis ganda	Many-to-one, Media total	FK NOT NULL di Media
follow: tanpa panah, ts_follow, role	M:N rekursif beratribut	Tabel relasi, dua FK ke User
reply: panah ke replyee	Many-to-one rekursif	FK ke Posting sendiri
contain: tanpa panah	M:N Posting–Media	Tabel relasi
Media ISA, disjoint, garis tunggal	Disjoint, **partial** ("dll.")	Superclass tetap butuh tabel
posting_type: normal/reply	Domain terbatas	CHECK

0.3 Tulis ulang query: entity + atribut + filter + urutan

Hasil sementara:

- **Q1:** User **berdasarkan email** → {email, name, phone_number, gender, date_of_birth} + follower {email, name}.
- **Q2: semua** Media + atribut khusus tiap jenis (dibaca bersama).
- **Q3:** satu Posting **milik seorang User** → data posting + media {id_media, media_type, file_size, created_at} + daftar posting_id reply.

0.4 Tulis asumsi

Hasil sementara:

Email unik dan tidak berubah; atribut standar media tidak berubah setelah dibuat; nama user jarang berubah; satu reply menanggapi tepat satu posting.

2.2 Soal 1: Model Relasional

R1 Strong entity → satu tabel, PK = key entity

Dikerjakan: strong entity di diagram: User, Posting, Media (subclass ditunda ke R7).

Hasil sementara:

```
User(email, phone_number, name, gender, date_of_birth, city, country, address, job)
Posting(posting_id, posting_type, ts_posting, IP, text_message)
Media(id_media, media_type, file_size, created_at)
```

R2 Atribut: komposit, multivalued, derived, opsional

Aturan: komposit diratakan; multivalued jadi tabel sendiri (nilai relasional harus atomik); derived tidak disimpan; opsional boleh NULL.

Dikerjakan: ts_posting → `ts_date`, `ts_time`; {hobbies} → tabel `User_Hobby` dengan PK gabungan (user tanpa hobi cukup tidak punya baris); address, job → NULL; tidak ada atribut derived.

Hasil sementara:

```
User(email, phone_number, name, gender, date_of_birth, city, country,
     address NULL, job NULL)
User_Hobby(email, hobby)                    FK email -> User
Posting(posting_id, posting_type, ts_date, ts_time, IP, text_message)
```

R3 Weak entity → PK = PK identifying + discriminator

Dilewati: konstruksi ini tidak ada di diagram.

R4 Relasi 1:N → FK di sisi "many" (NOT NULL bila total)

Dikerjakan:

- **post**: panah ke User = tiap Posting paling banyak satu User; garis ganda = tepat satu → FK `email NOT NULL` di Posting.
- **own**: sama → FK `owner_email NOT NULL` di Media.
- **reply** (rekursif): panah ke replyee = satu reply menanggapi paling banyak satu posting → FK `replyee_id` ke Posting sendiri, NULL untuk posting normal.

Hasil sementara:

```
Posting(posting_id, posting_type, ts_date, ts_time, IP, text_message,
        email NOT NULL, replyee_id NULL)  FK email -> User, FK replyee_id -> Posting
Media(id_media, media_type, file_size, created_at, owner_email NOT NULL)
                                          FK owner_email -> User
```

R5 Relasi M:N (termasuk rekursif) → tabel relasi

Aturan: PK = gabungan PK kedua sisi; atribut relasi jadi kolom; pada rekursif, dua FK diberi nama sesuai role.

Dikerjakan: **follow** (M:N rekursif, atribut ts_follow) dan **contain** (M:N Posting–Media).

Hasil sementara:

```
Follow(followee_email, follower_email, ts_follow)
       FK followee_email -> User, FK follower_email -> User
Contain(posting_id, id_media)   FK posting_id -> Posting, FK id_media -> Media
```

R6 Ternary dan aggregation → tabel relasi

Dilewati: konstruksi ini tidak ada di diagram.

R7 ISA → superclass + tabel per subclass

Aturan: PK subclass = PK superclass, sekaligus FK.

Dikerjakan: ISA disjoint dan **partial** (media_type "dll."), sehingga tabel superclass Media **wajib** ada: media jenis lain tidak punya tabel subclass. `media_type` menjadi penanda subclass.

Hasil sementara:

```
Image(id_media, resolution, dimension, color_space, camera_info)   FK id_media -> Media
Video(id_media, resolution, frame_rate, codec, duration)           FK id_media -> Media
PDF_File(id_media, title, author)                                  FK id_media -> Media
```

R8 Constraint: NOT NULL, CHECK, UNIQUE

Hasil sementara:

- Atribut user selain address, job, hobi: NOT NULL.
- `CHECK (posting_type IN ('normal','reply'))`.
- Konsistensi: `posting_type = 'reply'` jika dan hanya jika `replyee_id` tidak NULL.

R9 Optimasi berdasarkan query (opsional, wajib dengan trade-off)

Aturan: hitung join tiap query; bila berat, boleh gabung tabel ISA, simpan atribut turunan, duplikasi atribut, atau vertical partitioning.

Hasil sementara:

Query	Join model dasar	Optimasi	Trade-off
Q1	User + Follow + User	`follower_name` di Follow	Nama berubah → update semua baris Follow
Q1	Baris User penuh	Vertical partitioning `User_Main` / `User_Detail`	Data lengkap butuh join 1:1
Q2	Media + 3 subclass	Gabung jadi satu tabel Media	Banyak NULL; jenis baru butuh ALTER TABLE
Q3	Posting + Contain + Media + Posting	Index `Posting(replyee_id)`, `Contain(posting_id)`	Ruang dan biaya tulis index

Jawaban akhir (yang ditulis di lembar ujian)

```
User(email, phone_number, name, gender, date_of_birth, city, country,
     address NULL, job NULL)
User_Hobby(email, hobby)                  FK email -> User
Follow(followee_email, follower_email, ts_follow)
       FK followee_email -> User, FK follower_email -> User
Posting(posting_id, posting_type, ts_date, ts_time, IP, text_message,
        email NOT NULL, replyee_id NULL)  FK email -> User, FK replyee_id -> Posting
Media(id_media, media_type, file_size, created_at, owner_email NOT NULL)
                                          FK owner_email -> User
Image(id_media, resolution, dimension, color_space, camera_info)   FK -> Media
Video(id_media, resolution, frame_rate, codec, duration)           FK -> Media
PDF_File(id_media, title, author)                                  FK -> Media
Contain(posting_id, id_media)   FK posting_id -> Posting, FK id_media -> Media
```

Ditambah constraint R8, reasoning satu kalimat per langkah R2–R7, dan tabel optimasi R9.

2.3 Soal 2a: Desain Dokumen

D1 Tentukan collection dari query

Aturan: entity yang diambil sebagai "satu kesatuan" oleh query menjadi collection.
Dikerjakan: Q1 mengambil user; Q2 mengambil media; Q3 mengambil posting.

Hasil sementara:

Tiga collection: `users`, `media`, `postings`.

D2 Uji tiap relasi dengan empat faktor → embed atau reference

Hasil sementara:

Relasi	Dibaca bersama?	Ukuran	Diakses mandiri?	Keputusan
User–hobi	Ya (profil)	Kecil	Tidak	**Embed** array `hobbies`
User–follower	Ya (Q1)	Asumsi wajar	Tidak	**Embed** array `followers`
User–Posting	Tidak	Terus bertambah	Ya (Q3)	**Reference** `author_email`
User–Media	Tidak	Banyak	Ya (Q2)	**Reference** `owner_email`
Posting–Media (M:N)	Ya (Q3)	Sedikit per posting	Ya (Q2)	**Reference + salinan** (D3)
Posting–reply	Ya, hanya ID	Sedang	Ya	`reply_to` + `replies`

D3 Reference + salinan sebagian atribut (snapshot)

Aturan: salin atribut yang dibutuhkan query dan jarang berubah, agar salinan tidak cepat basi.
Dikerjakan: Q3 butuh atribut standar media bersama posting; atribut itu tidak berubah setelah dibuat → disalin ke array `media` di posting.

D4 ISA dan rekursif

Aturan: subclass yang sering dibaca bersama → satu collection polymorphic (field tipe + field khusus).
Dikerjakan: Q2 membaca semua jenis media bersama → `media` polymorphic; jenis baru tidak butuh perubahan skema. Rekursif reply sudah ditangani di D2 (parent + children reference).

D5 Tulis contoh JSON dan index

```
// users
{ "_id": "fazat@informatika.org", "name": "Fazat", "phone_number": "0812...",
  "gender": "F", "date_of_birth": "1990-01-01", "city": "Bandung",
  "country": "Indonesia", "address": "...", "job": "Dosen",   // opsional
  "hobbies": ["membaca", "berenang"],                          // D2: embed
  "followers": [ { "email": "a@x.org", "name": "Andi",         // D2: embed
                   "ts_follow": "2026-09-01T10:00" } ] }

// media (D4: polymorphic)
{ "_id": "M1", "media_type": "image", "file_size": 2048,
  "created_at": "2026-09-02T08:00", "owner_email": "fazat@informatika.org",
  "resolution": "1920x1080", "dimension": "...", "color_space": "sRGB",
  "camera_info": "..." }
{ "_id": "M2", "media_type": "pdf file", "file_size": 512,
  "created_at": "2026-09-03T09:00", "owner_email": "fazat@informatika.org",
  "title": "Silabus", "author": "Fazat" }

// postings
{ "_id": "P10", "posting_type": "normal",
  "ts_posting": { "date": "2026-09-05", "time": "10:00" },
  "IP": "10.0.0.1", "text_message": "Halo",
  "author_email": "fazat@informatika.org",                     // D2: reference
  "media": [ { "id_media": "M1", "media_type": "image",        // D3: salinan
               "file_size": 2048, "created_at": "2026-09-02T08:00" } ],
  "reply_to": null, "replies": ["P11", "P12"] }
```

Index: `postings.author_email`, `postings.reply_to`, `media.media_type`.

D6 Cek tiap query: berapa kali baca?

Hasil sementara:

Q1 = satu dokumen `users`; Q2 = satu collection `media` tanpa join; Q3 = satu dokumen `postings`.

Jawaban akhir (yang ditulis di lembar ujian)

JSON di D5 + tabel keputusan D2 sebagai reasoning embed/reference. Tambahkan satu kalimat: redundansi (nama follower, atribut media) disengaja dan dipilih dari atribut yang jarang berubah; integritas referensi dijaga aplikasi karena schemaless.

2.4 Soal 2b: Desain Column-Family (BigTable-like)

C1 Satu tabel per kebutuhan baca utama (bukan per entity)

Hasil sementara:

Q1 → tabel **User**; Q2 → tabel **Media**; Q3 → tabel **Posting**.

C2 Row key dari kriteria pencarian; cek hotspotting

Aturan: row tersimpan terurut menurut row key; hindari nilai sekuensial murni di awal row key (hash atau prefix acak).

Hasil sementara:

Tabel	Dicari lewat	Row key	Hotspot?
User	email	`email`	Tidak, nilai tersebar
Media	id_media	`id_media`	Hash/prefix bila sekuensial
Posting	posting_id	`posting_id`	Hash/prefix bila sekuensial

C3 Column family = kolom yang dibaca bersama

Dikerjakan: Q1 membaca phone_number, name, gender, date_of_birth → CF `profile`. City, country, address, job tidak dibaca Q1 → CF `detail` (tidak ikut terbaca).

C4 Relasi menjadi kolom dinamis (denormalize instead of join)

Aturan: nama kolom = ID entity lain (valueless), atau nama kolom = ID dan nilai = data.

Dikerjakan:

- follower (M:N): CF `follower`, kolom `<email_follower>` = `<nama_follower>`.
- hobi (multivalued): CF `hobby`, kolom `<nama_hobi>` kosong (valueless).
- media di posting: CF `media`, kolom `<id_media>:type`, `<id_media>:size`, `<id_media>:created`.
- reply: CF `reply`, kolom `<posting_id_reply>` kosong.
- ISA: CF `ext` di Media berisi kolom berbeda per jenis (sparse).

C5 Tabel index buatan atau nilai turunan

Dikerjakan: "posting milik seorang user" dicari lewat email, bukan posting_id → tabel index `Posting_by_User` (seperti `Cust_by_Prod` di materi).

C6 Cek tujuh guideline

Hasil sementara:

Guideline	Diterapkan di
1 Denormalize instead of join	Follower dan media standar disalin
2 Valueless columns	hobby, reply, Posting_by_User
3 Nama kolom dan nilai menyimpan data	follower: email = nama
4 Satu entity satu row	User dan Posting masing-masing satu row (tulis atomik)
5 Hindari hotspotting	Row key di-hash bila sekuensial
6 Versi secukupnya	Versi default kecil
7 Hindari struktur kompleks di value	Atribut media jadi kolom terpisah

Jawaban akhir (yang ditulis di lembar ujian)

```
Tabel User                          (Q1)
  Row key     : email
  CF profile  : phone_number, name, gender, date_of_birth
  CF detail   : city, country, address, job
```

```
  CF hobby    : <nama_hobi> = ""                      (valueless)
  CF follower : <email_follower> = <nama_follower>
  CF follow_ts: <email_follower> = <ts_follow>
Tabel Media                         (Q2)
  Row key     : id_media
  CF basic    : media_type, file_size, created_at, owner_email
  CF ext      : image -> resolution, dimension, color_space, camera_info
                video -> resolution, frame_rate, codec, duration
                pdf   -> title, author                (kolom dinamis, sparse)
Tabel Posting                       (Q3)
  Row key     : posting_id   (hash/prefix bila sekuensial)
  CF post     : posting_type, ts_date, ts_time, IP, text_message, author_email,
    reply_to
  CF media    : <id_media>:type, <id_media>:size, <id_media>:created
  CF reply    : <posting_id_reply> = ""
Tabel Posting_by_User               (index buatan)
  Row key     : email
  CF posting  : <posting_id> = ""
```

Ditambah tabel C2 dan C6 sebagai reasoning.

2.5 Soal 2c: Desain Graf

G1 Identifikasi query

Hasil sementara:

Q1 follower seorang user; Q2 semua media; Q3 posting + media + reply.

G2 Entity → vertex (label + properti)

Hasil sementara:

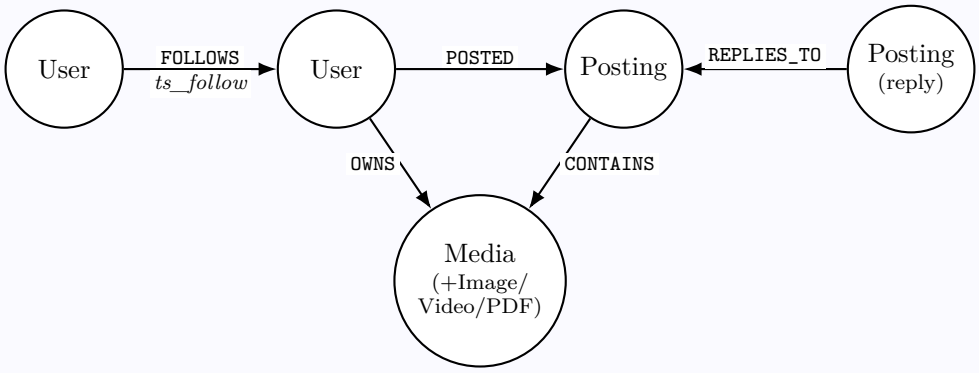
Label	Properti
User	email (ID), name, phone_number, gender, date_of_birth, city, country, address?, job?, hobbies? (list)
Posting	posting_id (ID), posting_type, ts_date, ts_time, IP, text_message
Media (+ label Image / Video / PDF_File)	id_media (ID), media_type, file_size, created_at, + properti khusus jenis

G3 Relasi → edge: label, arah, properti

Aturan: edge berarah untuk relasi tidak simetris; atribut relasi menjadi properti edge.

Hasil sementara:

Edge	Dari → Ke	Properti	Asal ER
FOLLOWS	User (follower) → User (followee)	ts_follow	follow (tidak simetris)
POSTED	User → Posting		post
OWNS	User → Media		own
CONTAINS	Posting → Media		contain
REPLIES_TO	Posting (replier) → Posting (replyee)		reply



G4 Query → jalur di graf

Hasil sementara:

- **Q1:** vertex User (email) → vertex di ujung edge FOLLOWS yang **masuk** = follower.
- **Q2:** semua vertex berlabel Media.
- **Q3:** User –POSTED→ Posting; Posting –CONTAINS→ Media; Posting yang punya edge REPLIES_TO masuk ke posting itu (ambil posting_id).

Jawaban akhir (yang ditulis di lembar ujian)

Tabel vertex G2, tabel edge G3, gambar, dan jalur G4. Reasoning: semua relasi menjadi edge sehingga query mengikuti edge tanpa join; ISA ditangani label tambahan; hobi cukup properti list karena tidak ada query yang menelusurinya.

2.6 Soal 3: Use Case Key-Value

K1 Cari akses yang kecil, sering, dan selalu lewat satu key

Dikerjakan: Q1 sangat sering dan selalu dicari lewat email → **cache kartu profil**.

K2 Rancang key `<entity>:<id>:<atribut>`

Hasil sementara:

`user:fazat@informatika.org:profileCard`

K3 Rancang value: gabungkan atribut yang selalu dibaca bersama

Hasil sementara:

`{name, phone_number, gender, date_of_birth, followers: [{email, name}]}`

K4 Operasi dan alasan

Hasil sementara:

`get(key)` saat profil dibuka; bila kosong, ambil dari basis data utama lalu `set(key, value)` dengan **TTL** 10 menit; hapus key saat profil atau follower berubah. Alasan: read-heavy, point lookup lewat satu key, satu akses mendapat semua atribut Q1.

2.7 Soal 4a: XML (DTD)

X1 Bentuk hasil query 3 sebagai pohon

Hasil sementara:

```
posting
 |-- data posting: posting_type, ts_posting(date, time), IP, text_message
 |-- media*   : id_media, media_type, file_size, created_at
 `-- reply*   : posting_id
```

X2 Root tunggal

Hasil sementara:

`postings` berisi `posting*`.

X3 Atribut vs elemen

Aturan: identifier → atribut; isi → subelemen.

Hasil sementara:

posting_id dan id_media menjadi atribut; sisanya subelemen.

X4 Kemunculan

Hasil sementara:

media dan reply boleh 0..n → `*`; data posting wajib → tanpa tanda.

X5 Cek data untuk soal berikutnya

Dikerjakan: soal 4b memfilter email (langkah 0.1) → tambahkan atribut `author_email`.

X6 Tulis DTD

```
<!ELEMENT postings (posting*)>
<!ELEMENT posting (posting_type, ts_posting, IP, text_message, media*, reply*)>
<!ATTLIST posting
    posting_id   ID     #REQUIRED
    author_email CDATA  #REQUIRED>
<!ELEMENT posting_type (#PCDATA)>
<!ELEMENT ts_posting (date, time)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT IP (#PCDATA)>
<!ELEMENT text_message (#PCDATA)>
<!ELEMENT media (media_type, file_size, created_at)>
<!ATTLIST media id_media CDATA #REQUIRED>
<!ELEMENT media_type (#PCDATA)>
<!ELEMENT file_size (#PCDATA)>
<!ELEMENT created_at (#PCDATA)>
<!ELEMENT reply EMPTY>
<!ATTLIST reply posting_id CDATA #REQUIRED>
```

Nilai bertipe `ID` harus diawali huruf (`P10`). `reply` memakai CDATA, bukan IDREF, karena posting reply-nya tidak ikut di dokumen (IDREF wajib menunjuk ID yang ada). Versi XML Schema ada di *Analisis_Soal_UTS_PDL* bagian 4.6.

2.8 Soal 4b: XQuery

F1–F4 Susun FLWOR dari pertanyaan

Hasil sementara:

Klausa	Pertanyaan	Jawaban
F1 `for`	Iterasi apa?	Setiap posting: `$p in doc("postings.xml")/postings/posting`
F2 `where`	Filter apa?	`$p/@author_email = "fazat@informatika.org"`
F3 `order by`	Urutan?	Tidak diminta; boleh terbaru dulu
F4 `return`	Bentuk hasil?	Seluruh posting: `$p`

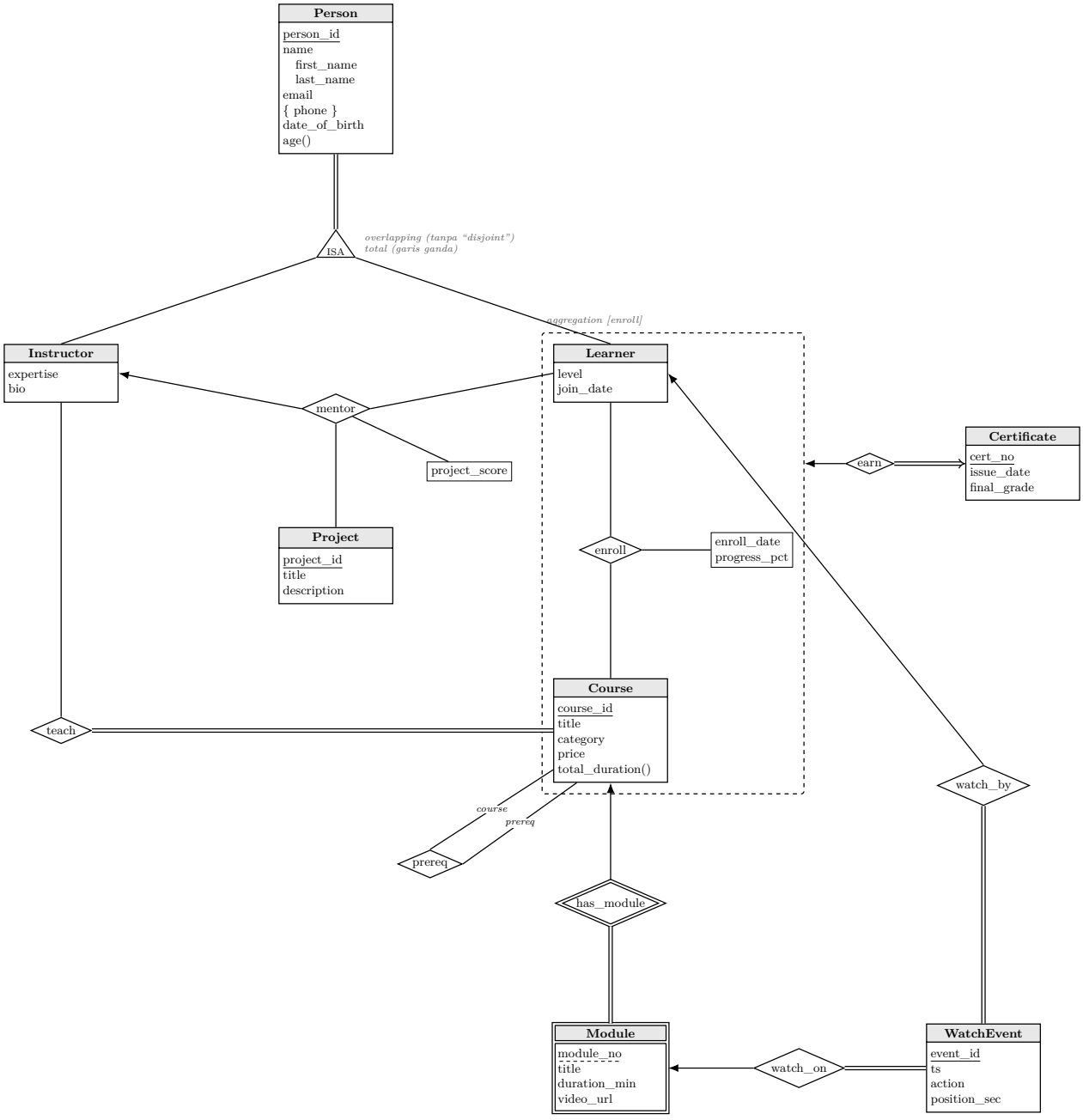
Jawaban akhir (yang ditulis di lembar ujian)

```
for $p in doc("postings.xml")/postings/posting
where $p/@author_email = "fazat@informatika.org"
order by $p/ts_posting/date descending, $p/ts_posting/time descending
return $p
```

Padanan XPath: `/postings/posting[@author_email = "fazat@informatika.org"]`.

3 Kasus Prediksi: KursusKita

Kasus ini memuat konstruksi yang **tidak ada** di sample: weak entity, ISA overlapping dan total, derived, ternary, aggregation, dan 1:1. Soal lengkapnya ada di *Analisis_Soal_UTS_PDL* bagian 5.1.



Ringkasan soal

Entity: Person (person_id, name(first_name, last_name), email, {phone}, date_of_birth, age()); Instructor ISA (expertise, bio); Learner ISA (level, join_date); Course (course_id, title, category, price, total_duration()); **Module** weak dari Course (module_no, title, duration_min, video_url); WatchEvent (event_id, ts, action, position_sec); Project (project_id, title, description); Certificate (cert_no, issue_date, final_grade).
Relasi: ISA overlapping + total; has_module (identifying); teach (M:N, Course total); enroll (M:N, enroll_date, progress_pct); prereq (M:N rekursif); watch_by, watch_on (many-to-one, WatchEvent total); **mentor** ternary Instructor–Learner–Project, panah ke Instructor, atribut project_score; **earn** aggregation [enroll]–Certificate, 1:1, Certificate total.
Info: phone dan bio boleh kosong; level ∈ {beginner, intermediate, advanced}; seseorang bisa instructor sekaligus learner; WatchEvent jutaan per hari, hanya ditambah.
Query: Q1 halaman course (course + module terurut + nama instructor + judul prasyarat); Q2 dashboard learner (course + progress + posisi terakhir menonton); Q3a riwayat menonton learner dalam rentang waktu, Q3b jumlah play per module per hari; Q4a prasyarat langsung dan tidak

langsung, Q4b rekomendasi course; Q5 sertifikat ke mitra dalam XML.
Soal: 1 relasional; 2a/b/c dokumen, column-family, graf; 3 key-value; 4a DTD, 4b XQuery sertifikat `ana@kursuskita.id` terbaru, 4c XPath cert_no dengan grade "A"; 5 model paling cocok per query.

3.1 Langkah Nol

0.1 Baca semua sub-soal

Hasil sementara:

XML (soal 4) butuh nama dan **email** learner, judul course, dan judul module; soal 4b mengurutkan **issue_date**; soal 4c memfilter **final_grade**.

0.2 Tandai konstruksi

Hasil sementara:

Elemen	Konstruksi	Akibatnya
Person.name	Komposit	first_name, last_name
Person.{phone}	Multivalued, opsional	Tabel/array
age(), total_duration()	Derived	Tidak disimpan di relasional
ISA	Overlapping, total	Tabel per subclass; kolom "type" tunggal tidak cukup
Module	Weak entity	PK (course_id, module_no)
watch_by, watch_on	Many-to-one total	Dua FK NOT NULL
teach, enroll, prereq	M:N	Tabel relasi
mentor	Ternary, panah ke Instructor	PK (learner_id, project_id)
earn	Aggregation 1:1, Certificate total	FK ke Enroll di Certificate + UNIQUE

0.3 dan 0.4 Ciri query dan asumsi

Hasil sementara:

Q3 bervolume besar, append-only, dicari per rentang waktu; Q4 berupa penelusuran berantai. Asumsi: email unik; nama dan judul jarang berubah; event tidak pernah diubah.

3.2 Soal 1: Model Relasional

R1 Strong entity

Hasil sementara:

```
Person(person_id, name, email, date_of_birth)
Course(course_id, title, category, price)
WatchEvent(event_id, ts, action, position_sec)
Project(project_id, title, description)
Certificate(cert_no, issue_date, final_grade)
```

R2 Atribut

Dikerjakan: name → first_name, last_name; {phone} → tabel `Person_Phone`; **age() dan total_duration() derived → tidak disimpan** (dihitung lewat view: age dari date_of_birth, total_duration = SUM duration_min); bio opsional → NULL.

Hasil sementara:

```
Person(person_id, first_name, last_name, email UNIQUE NOT NULL, date_of_birth)
Person_Phone(person_id, phone)     FK person_id -> Person
```

R3 Weak entity → PK = PK identifying + discriminator

Dikerjakan: Module bergantung pada Course lewat has_module (double diamond).

Hasil sementara:

```
Module(course_id, module_no, title, duration_min, video_url)
       FK course_id -> Course
```

R4 Relasi 1:N

Dikerjakan: watch_by dan watch_on many-to-one, WatchEvent total → FK NOT NULL di WatchEvent. FK ke Module **gabungan**, karena PK Module gabungan.

Hasil sementara:

```
WatchEvent(event_id, learner_id NOT NULL, course_id NOT NULL,
           module_no NOT NULL, ts, action, position_sec)
           FK learner_id -> Learner, FK (course_id, module_no) -> Module
```

R5 M:N dan rekursif

Hasil sementara:

```
Teach(instructor_id, course_id)           FK -> Instructor, Course
Enroll(learner_id, course_id, enroll_date, progress_pct)
                                                FK -> Learner, Course
Prereq(course_id, prereq_id)              FK keduanya -> Course
```

Course total di teach tidak bisa dijamin FK → trigger/aplikasi.

R6 Ternary dan aggregation

Dikerjakan:

- **mentor** (panah ke Instructor): tiap pasangan (learner, project) paling banyak satu instructor → PK = (learner_id, project_id); instructor_id kolom biasa NOT NULL.
- **earn** (aggregation 1:1, Certificate total): aturan 1:1 → FK di sisi total (Certificate) agar tidak ada NULL; UNIQUE agar satu enrollment paling banyak satu sertifikat.

Hasil sementara:

```
Mentor(learner_id, project_id, instructor_id NOT NULL, project_score)
       FK learner_id -> Learner, FK project_id -> Project,
       FK instructor_id -> Instructor
Certificate(cert_no, issue_date, final_grade, learner_id NOT NULL,
            course_id NOT NULL)
       FK (learner_id, course_id) -> Enroll, UNIQUE (learner_id, course_id)
```

R7 ISA overlapping dan total

Dikerjakan: superclass + tabel per subclass. Overlapping: orang yang instructor sekaligus learner punya baris di kedua tabel (satu kolom "type" tidak bisa menyatakan "keduanya"). Total tidak bisa ditegakkan FK → trigger/aplikasi.

Hasil sementara:

```
Instructor(person_id, expertise, bio NULL)   FK person_id -> Person
Learner(person_id, level, join_date)         FK person_id -> Person
```

R8 Constraint

Hasil sementara:

`CHECK (level IN ('beginner','intermediate','advanced'))`; atribut wajib NOT NULL.

R9 Optimasi

Hasil sementara:

Query	Masalah	Optimasi	Trade-off
Q2	Mencari event terakhir di antara jutaan WatchEvent	`last_module_no`, `last_position_sec` di Enroll	Redundansi; Enroll harus diperbarui
Q3a	Scan WatchEvent	Index `WatchEvent(learner_id, ts)`	Biaya tulis index
Q3b	Agregasi per hari	Materialized view, refresh berkala	Data sedikit tertinggal

3.3 Soal 2a: Dokumen

D1 Collection dari query

Hasil sementara:

Q1 → `courses`; Q2 → `enrollments`; Q3 → `watch_events`; data orang → `persons`; ternary → `mentorships`.

D2 Uji empat faktor

Hasil sementara:

Relasi	Keputusan	Faktor penentu
ISA overlapping	Satu `persons` + sub-dokumen `instructor` dan/atau `learner` + array `roles`	1:1 ke tiap peran → embed
Module (weak)	**Embed** array di `courses`	Bergantung pada course, kecil, dibaca bersama (Q1)
teach, prereq	Reference + salinan nama/judul (D3)	M:N; Q1 butuh nama/judul yang jarang berubah
enroll	Collection sendiri `enrollments`	Diakses per learner, progress sering berubah, terus bertambah
earn (1:1)	**Embed** `certificate` di `enrollments`	1:1 → embed
WatchEvent	Collection sendiri, **satu dokumen per learner per hari**	Volume sangat besar (pola truk di materi)
mentor (ternary)	Collection `mentorships`, tiga referensi	Tidak bisa di-embed ke satu sisi tanpa duplikasi

D3 dan D4 Snapshot, ISA, rekursif

Hasil sementara:

Nama instructor dan judul prasyarat disalin ke `courses`; judul course disalin ke `enrollments`. ISA sudah di D2. Prereq rekursif → array `prerequisites` berisi {course_id, title}.

D5 JSON dan index

```
// persons
{ "_id": "P1", "name": {"first_name": "Ana", "last_name": "Putri"},
  "email": "ana@kursuskita.id", "phones": ["0812..."], "date_of_birth": "2003-05-01",
  "roles": ["learner", "instructor"],
  "learner": {"level": "intermediate", "join_date": "2025-01-10"},
  "instructor": {"expertise": "SQL", "bio": "..."} }
// courses
{ "_id": "C10", "title": "Basis Data Lanjut", "category": "Data", "price": 150000,
  "modules": [ {"module_no": 1, "title": "ER", "duration_min": 30, "video_url": "..."},
               {"module_no": 2, "title": "NoSQL", "duration_min": 45, "video_url":
    "..."} ],
  "instructors": [ {"person_id": "P7", "name": "Budi Santoso"} ],
  "prerequisites": [ {"course_id": "C3", "title": "Basis Data"} ] }
// enrollments
{ "_id": "E55", "learner_id": "P1",
  "course": {"course_id": "C10", "title": "Basis Data Lanjut"},
  "enroll_date": "2026-08-01", "progress_pct": 40,
  "last_watch": {"module_no": 2, "position_sec": 615, "ts": "2026-10-12T20:15"},
  "certificate": {"cert_no": "K001", "issue_date": "2026-10-01", "final_grade": "A"} }
// watch_events (satu dokumen per learner per hari)
{ "_id": "P1#2026-10-12", "learner_id": "P1", "date": "2026-10-12",
  "events": [ {"ts": "20:10:00", "course_id": "C10", "module_no": 2,
               "action": "play", "position_sec": 0} ] }
// mentorships
{ "_id": "MT1", "learner_id": "P1", "project": {"project_id": "PR4", "title":
    "Capstone"},
  "instructor": {"person_id": "P7", "name": "Budi Santoso"}, "project_score": 88 }
```

Index: `enrollments.learner_id`, `watch_events.learner_id + date`, `persons.email`.

D6 Cek query

Hasil sementara:

Q1 = satu dokumen course; Q2 = dokumen enrollments milik learner (posisi terakhir sudah ada); Q3a = dokumen per hari dalam rentang. Q3b dan Q4 kurang cocok di dokumen (agregasi besar, penelusuran rekursif) → dijawab di soal 5.

3.4 Soal 2b: Column-Family

C1 dan C2 Tabel per query dan row key

Hasil sementara:

Tabel	Dicari lewat	Row key	Alasan
Course (Q1)	course_id	`course_id`	Point lookup
Learner_Course (Q2)	learner_id	`learner_id`	Semua course seorang learner dalam satu row
WatchEvent (Q3a)	learner + rentang waktu	`learner_id#ts#event_id`	Row terurut → **range scan**; prefix learner_id menyebarkan tulisan (anti-hotspot); event_id mencegah dua event di detik yang sama saling menimpa
PlayCount (Q3b)	module + hari	`course_id#module_no`	Satu row per module, kolom per hari

C3 dan C4 Column family dan kolom dinamis

Hasil sementara:

Course: CF `info`, `module` (`m001_title`, `m001_dur`; nomor diberi nol di depan agar terurut), `instructor` (`<person_id> = <nama>`), `prereq` (`<course_id> = <judul>`). Learner_Course: CF `enroll`, `resume`. WatchEvent: CF `ev`. PlayCount: CF `day`.

C5 Nilai turunan

Hasil sementara:

total_duration disimpan di CF `info` (dihitung saat module berubah); jumlah play per hari disimpan di PlayCount dan di-increment setiap event play.

C6 Cek

Hasil sementara:

WatchEvent write-intensive (alasan utama memilih column-family); Q1 dan Q2 masing-masing satu row; Q3a satu range scan, mis. 1–13 Oktober = scan dari `P1#20261001` sampai **sebelum** `P1#20261014`.

Jawaban akhir (yang ditulis di lembar ujian)

```
Tabel Course              Row key: course_id
  CF info       : title, category, price, total_duration
  CF module     : m001_title, m001_dur, m002_title, m002_dur, ...
  CF instructor : <person_id> = <nama>
  CF prereq     : <course_id_prasyarat> = <judul>
Tabel Learner_Course      Row key: learner_id
  CF enroll     : <course_id>:title, <course_id>:progress, <course_id>:enroll_date
  CF resume     : <course_id>:module, <course_id>:pos
Tabel WatchEvent          Row key: learner_id#ts#event_id
  CF ev         : course_id, module_no, action, position_sec
Tabel PlayCount           Row key: course_id#module_no
  CF day        : <yyyymmdd> = <jumlah play>
```

3.5 Soal 2c: Graf

G1 dan G2 Query dan vertex

Hasil sementara:

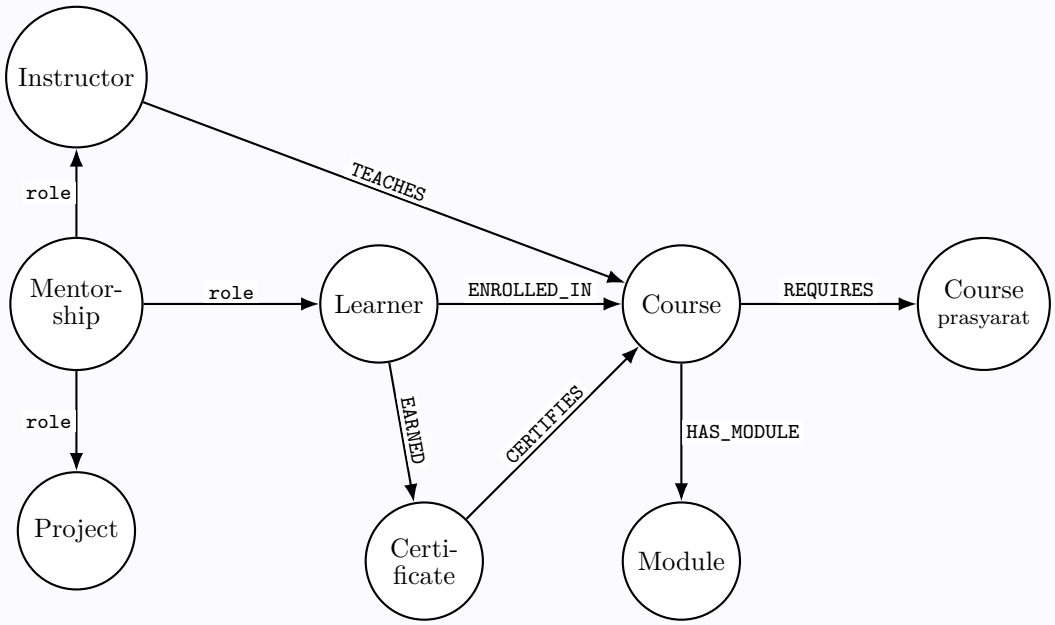
Query yang cocok: Q4a (prasyarat berantai) dan Q4b (rekomendasi). Vertex: Person (label `Instructor` dan/atau `Learner`; overlapping = **dua label** di vertex yang sama), Course, Module, Project, Certificate, **Mentorship**.

G3 Edge

Hasil sementara:

Edge	Dari → Ke	Properti	Catatan
TEACHES	Instructor → Course		
ENROLLED_IN	Learner → Course	enroll_date, progress_pct	Atribut relasi
REQUIRES	Course → Course		Rekursif, berarah
HAS_MODULE	Course → Module		Weak entity
EARNED	Learner → Certificate		
CERTIFIES	Certificate → Course		
IN_MENTORSHIP (×3)	Mentorship → Instructor / Learner / Project	role	**Ternary → vertex perantara**

Aggregation earn: edge tidak bisa terhubung ke edge; aturan penuhnya menjadikan enrollment vertex. Di sini disederhanakan (Certificate dihubungkan ke Learner dan Course) karena earn 1:1, dengan catatan aplikasi memastikan enrollment-nya ada. WatchEvent **tidak** dimasukkan: jutaan data dan tidak ditelusuri sebagai relasi.



G4 Query → jalur

Hasil sementara:

Q4a: jalur REQUIRES dengan panjang berapa pun (waspadai cycle):

```
MATCH (c:Course {course_id: $id})-[:REQUIRES*1..]->(p:Course) RETURN DISTINCT p.title
```

Q4b: aku –ENROLLED_IN→ course ←ENROLLED_IN– learner lain –ENROLLED_IN→ course rekomendasi (yang belum kuambil):

```
MATCH (me:Learner {person_id: $id})-[:ENROLLED_IN]->(c)<-[:ENROLLED_IN]-(o)
      -[:ENROLLED_IN]->(rec)
WHERE NOT (me)-[:ENROLLED_IN]->(rec)
RETURN rec.title, count) AS skor ORDER BY skor DESC
```

Sintaks Cypher dan path `*1..` adalah tambahan di luar slide; yang wajib adalah jalurnya.

3.6 Soal 3: Key-Value

K1–K4 Posisi "lanjutkan menonton"

Hasil sementara:

- **K1:** posisi menonton ditulis setiap beberapa detik dan dibaca lewat (learner, course) → kecil, sangat sering, lewat satu key.
- **K2:** `resume:<learner_id>:<course_id>`.
- **K3:** `{module_no, position_sec, updated_at}`.
- **K4:** `set` berkala saat video diputar, `get` saat membuka course atau dashboard. Di relasional ini berarti update baris yang sangat sering.

3.7 Soal 4: DTD, XQuery, XPath

X1–X5 Bentuk hasil dan keputusan

Hasil sementara:

```
certificate (cert_no)
 |-- learner (first_name, last_name, email)
 |-- course  (title, module* (title))
 |-- issue_date
 `-- final_grade
```

- **X2:** root `certificates` berisi `certificate*`.
- **X3:** cert_no, learner_id, course_id, module_no → atribut; sisanya subelemen.
- **X4:** `module*` (Module total terhadap has_module, bukan Course; boleh `module+` dengan asumsi setiap course punya module).
- **X5:** email learner ada (4b), issue_date ada (diurutkan 4b), final_grade ada (difilter 4c).

X6 DTD

```
<!ELEMENT certificates (certificate*)>
<!ELEMENT certificate (learner, course, issue_date, final_grade)>
<!ATTLIST certificate cert_no ID #REQUIRED>
<!ELEMENT learner (first_name, last_name, email)>
<!ATTLIST learner learner_id CDATA #REQUIRED>
<!ELEMENT course (title, module*)>
<!ATTLIST course course_id CDATA #REQUIRED>
<!ELEMENT module (title)>
<!ATTLIST module module_no CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT issue_date (#PCDATA)>
<!ELEMENT final_grade (#PCDATA)>
```

DTD tanpa tipe → tanggal ditulis `YYYY-MM-DD` agar urutan teks = urutan tanggal; `cert_no` bertipe ID harus diawali huruf (`K001`).

F1–F4 XQuery (4b)

Dikerjakan: F1 iterasi certificate → F2 filter email learner → F3 urut issue_date menurun → F4 bentuk hasil.

```
for $c in doc("certificates.xml")/certificates/certificate
where $c/learner/email = "ana@kursuskita.id"
order by $c/issue_date descending
return <sertifikat no="{ $c/@cert_no }">
         { $c/course/title, $c/issue_date, $c/final_grade }
       </sertifikat>
```

XPath (4c)

Dikerjakan: jalur ke certificate → predikat grade → ambil atribut.

```
/certificates/certificate[final_grade = "A"]/@cert_no
```

3.8 Soal 5: Model Paling Cocok per Query

P1 ciri → P2 model → P3 alasan

Hasil sementara:

Query	P1 ciri	P2 model	P3 alasan
Q1 halaman course	Satu kesatuan dibaca utuh	**Dokumen**	Module di-embed, satu baca
Q2 dashboard + posisi	Baca/tulis kecil sangat sering lewat key; daftar per learner	**Key-value + dokumen**	Point lookup; agregat per learner
Q3 event menonton	Volume raksasa, append-only, rentang waktu	**Column-family**	Tulis tinggi, range scan menurut row key
Q4 prasyarat, rekomendasi	Relasi berantai	**Graf**	Mengikuti edge tanpa join rekursif
Q5 kirim ke mitra	Pertukaran data lintas sistem	**XML**	Skema bisa divalidasi mitra
Pembayaran	Transaksi	**Relasional**	ACID, integritas kuat

Satu aplikasi boleh memakai beberapa model sekaligus.

4 Ringkasan Inti

Bawa ke ruang ujian (di kepala)

- **Selalu mulai dengan B-T-T-A:** baca semua sub-soal, tandai konstruksi ER, tulis ulang query, tulis asumsi.
- **Relasional R1–R9:** kerjakan dari strong entity sampai ISA, lalu constraint dan optimasi. Langkah yang konstruksinya tidak ada ditulis "dilewati".
- **Dokumen D1–D6:** collection dari query, uji 4 faktor tiap relasi, snapshot atribut yang jarang berubah, cek tiap query = 1 baca.
- **Column-family C1–C6:** tabel per query, row key dari kriteria cari (`id#waktu` untuk rentang, anti-hotspot), CF = dibaca bersama, relasi jadi kolom dinamis, cek 7 guideline.

- **Graf G1–G4:** vertex, edge berarah berproperti, ternary/aggregation jadi vertex perantara, query = jalur.
- **Key-value K1–K4, XML X1–X6, XQuery F1–F4, pilih model P1–P3.**