Atomik dan Kardinalitas
Dua Konsep Berbeda dalam Perancangan Basis Data Relasional

Muhammad Riduwan Khafidi
Program Studi Teknik Rekayasa Perangkat Lunak
Politeknik Negeri Batam
4342611034@students.polibatam.ac.id

November 5, 2026

Abstrak

Atomik dan kardinalitas adalah dua konsep yang sering tertukar dalam perancangan basis data relasional. Atomik berbicara tentang isi satu sel, sedangkan kardinalitas berbicara tentang aturan hubungan antar tabel. Makalah ini meluruskan kesalahpahaman yang lazim muncul: bahwa 1:1 berarti foreign key hanya ditulis satu kali, dan 1:N berarti foreign key boleh ditulis berkali-kali. Intuisi tersebut hampir benar, tetapi mekanisme penegaknya perlu diperjelas, yaitu constraint `UNIQUE` untuk 1:1 dan ketiadaan `UNIQUE` untuk 1:N. Makalah ini menyajikan sembilan visualisasi: perbandingan atomik dan non-atomik, peta posisi kedua konsep, intuisi kemunculan foreign key, anatomi tiga jenis kardinalitas, simulasi insert yang menunjukkan kapan constraint menolak, pohon keputusan pemilihan kardinalitas, dan timeline studi kasus delapan langkah. Melalui kerangka ini, pembaca memperoleh mental model yang dapat dipakai untuk menjawab soal ujian maupun merancang skema basis data yang benar.

1 Pendahuluan

1.1 Analogi Pembuka: Loker Kantor dan Aturan Tamu

Bayangkan sebuah kantor dengan dua hal yang perlu diatur. Pertama, loker karyawan. Setiap loker punya satu pintu, satu kunci, satu label nama. Seorang karyawan tidak boleh menaruh dua tas di satu loker, karena tas kedua akan tertumpuk dan sulit diambil. Aturan "satu loker satu barang" inilah yang disebut atomik.

Kedua, aturan tamu. Ada perusahaan yang menerapkan "satu direktur hanya boleh punya satu sekretaris pribadi". Ada perusahaan lain yang menerapkan "satu manajer boleh punya banyak staf, tetapi satu staf hanya dilayani satu manajer". Ada juga firma konsultan yang menerapkan "banyak konsultan boleh mengerjakan banyak proyek, dan banyak proyek boleh dikerjakan banyak konsultan". Aturan "siapa boleh berpasangan dengan siapa" inilah yang disebut kardinalitas.

Dua hal ini berbeda. Loker adalah soal isi satu sel. Kardinalitas adalah soal hubungan antar

tabel. Keduanya sering tertukar karena seorang pemula yang mencoba membuat relasi 1:N kadang menyelesaikannya dengan menaruh daftar nama di satu sel, yang justru melanggar atomik.

Tesis makalah ini: atomik adalah batasan pada level sel, sedangkan kardinalitas adalah batasan pada level relasi, dan keduanya ditegakkan oleh mekanisme yang berbeda di dalam perintah `CREATE TABLE`.

1.2 Mengapa Topik Ini Penting

Dalam ujian basis data, pertanyaan tentang kardinalitas hampir selalu muncul. Bentuknya bisa bermacam-macam: menentukan jenis relasi dari sebuah studi kasus, menggambar ERD, atau mengonversi ERD menjadi skema tabel. Kesalahan paling umum bukan pada menggambar, melainkan pada penempatan foreign key. Mahasiswa sering menaruh foreign key di sisi yang salah, atau lupa menambahkan `UNIQUE` ketika seharusnya 1:1, sehingga skema yang dihasilkan tidak sesuai dengan aturan bisnis yang diminta.

Selain itu, ada kesalahpahaman konseptual yang lebih dalam: mengira bahwa 1:1 berarti foreign key hanya ditulis sekali di dalam perintah `CREATE TABLE`, dan 1:N berarti boleh ditulis berkali-kali. Padahal dalam perintah `CREATE TABLE`, foreign key selalu ditulis sekali, terlepas dari kardinalitasnya. Yang berbeda adalah ada atau tidaknya constraint `UNIQUE` yang menemaninya.

1.3 Peta Pembahasan

Makalah ini dibagi menjadi delapan bagian utama. Bagian kedua membahas atomik dan mengapa ia adalah fondasi yang tidak bisa ditawar. Bagian ketiga membahas kardinalitas dan meluruskan tiga kesalahpahaman umum. Bagian keempat menyajikan anatomi tiga jenis kardinalitas dengan simulasi insert yang menunjukkan kapan constraint menolak data. Bagian kelima menyandingkan ketiganya dalam satu tabel perbandingan dan satu pohon keputusan. Bagian keenam membawa seluruh konsep ke studi kasus delapan langkah membangun skema dosen-mahasiswa. Bagian ketujuh dan kedelapan merangkum kesalahan umum dan lembar rujukan cepat. Bagian kesembilan menutup dengan tiga prinsip penutup.

2 Atomik: Fondasi yang Sering Terlupakan

2.1 Definisi dan Mengapa Ia Hadir

Atomik berarti tidak dapat dibagi lagi. Dalam konteks basis data relasional, atomik berarti setiap sel dalam tabel hanya boleh berisi satu nilai dari tipe yang telah ditentukan. Bukan daftar, bukan objek, bukan array, bukan JSON bertumpuk. Satu sel berisi satu nama, atau satu angka, atau satu tanggal. Tidak lebih.

Aturan ini adalah syarat pertama dari Normal Form pertama (1NF). Tanpa atomik, seluruh operasi SQL menjadi tidak dapat diandalkan. Pencarian menjadi lambat karena harus memindai

isi sel, bukan mencocokkan nilai. Pengurutan menjadi tidak mungkin karena tidak ada cara membandingkan daftar dengan daftar secara konsisten. Agregasi menjadi tidak akurat karena fungsi seperti `COUNT` hanya melihat baris, bukan elemen di dalam sel.

2.2 Perbandingan Atomik dan Non-Atomik

Gambar 1 membandingkan dua tabel yang menyimpan data mahasiswa. Tabel kiri atomik, tabel kanan tidak.

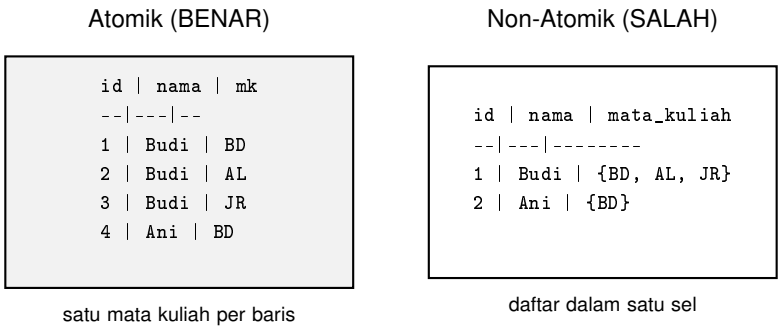


Figure 1. Perbandingan tabel atomik dan non-atomik untuk data mahasiswa.

Tabel kanan menyimpan daftar di dalam satu sel. Ini terlihat hemat karena hanya ada dua baris, tetapi menimbulkan tiga masalah. Pertama, pencarian "siapa saja yang mengambil Algoritma" harus membaca seluruh sel, bukan sekadar mencocokkan nilai. Kedua, tidak mungkin menambahkan indeks pada sel yang berisi daftar. Ketiga, tidak ada cara mudah untuk menghapus satu mata kuliah tanpa menulis ulang seluruh sel.

2.3 Mengapa Atomik Bukan Kardinalitas

Ini titik yang paling sering membingungkan. Atomik dan kardinalitas terlihat mirip karena keduanya membatasi apa yang boleh masuk ke tabel, tetapi keduanya berada di level yang berbeda.

Gambar 2 memetakan posisi keduanya dalam perancangan basis data.

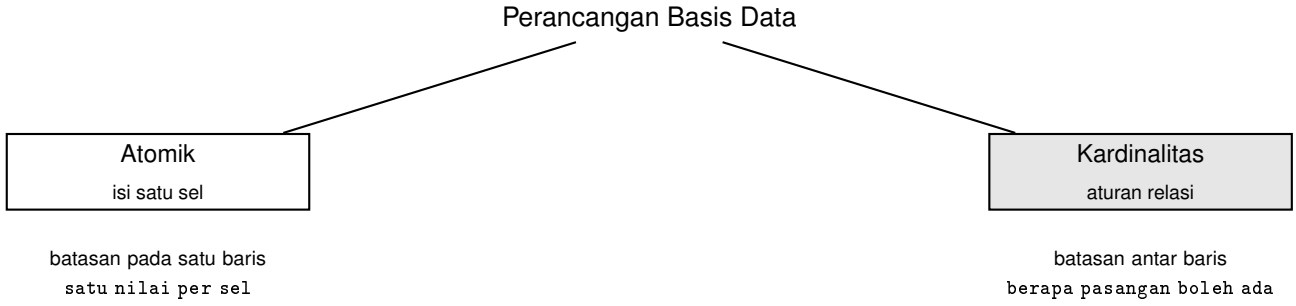


Figure 2. Posisi atomik dan kardinalitas dalam perancangan basis data.

Atomik adalah batasan pada level sel: satu sel satu nilai. Kardinalitas adalah batasan pada

level relasi: berapa banyak baris dari tabel A boleh berpasangan dengan baris dari tabel B. Keduanya bisa dilanggar secara terpisah. Sebuah tabel bisa atomik tetapi kardinalitasnya salah, atau sebaliknya.

Namun ada satu titik temu. Ketika seseorang mencoba membuat relasi 1:N tetapi tidak tahu caranya, ia sering menyelesaikannya dengan menaruh daftar nama mahasiswa di satu sel tabel dosen. Solusi ini melanggar atomik. Jadi pelanggaran atomik sering kali adalah gejala dari kesalahpahaman kardinalitas. Cara yang benar untuk membuat 1:N adalah dengan foreign key, bukan dengan daftar.

3 Kardinalitas: Apa yang Sebenarnya Dibatasi

3.1 Definisi Kardinalitas

Kardinalitas adalah aturan yang menentukan berapa banyak baris dari satu tabel boleh berpasangan dengan baris dari tabel lain. Ada tiga jenis yang perlu dikuasai, yaitu 1:1, 1:N, dan N:M. Dalam makalah ini, istilah "foreign key" dan "primary key" dipakai di dalam prosa, sedangkan singkatan "FK" dan "PK" dipakai di dalam tabel dan diagram agar ringkas.

Yang dibatasi oleh kardinalitas bukan primary key, karena primary key selalu unik di dalam tabelnya sendiri. Yang dibatasi adalah pasangan relasinya. Contohnya, dalam relasi dosen-mahasiswa, yang menjadi pertanyaan kardinalitas adalah: bolehkah satu dosen berpasangan dengan banyak mahasiswa, dan bolehkah satu mahasiswa berpasangan dengan banyak dosen.

3.2 Meluruskan Tiga Miskonsepsi

Miskonsepsi pertama, mengira 1:1 berarti foreign key hanya ditulis sekali di dalam perintah `CREATE TABLE`. Ini keliru. Dalam perintah `CREATE TABLE`, foreign key selalu ditulis sekali, terlepas dari kardinalitasnya. Yang membuat 1:1 menjadi 1:1 adalah tambahan constraint `UNIQUE` pada foreign key tersebut.

Miskonsepsi kedua, mengira 1:N berarti foreign key boleh ditulis berkali-kali di dalam perintah `CREATE TABLE`. Ini juga keliru. Yang membuat 1:N menjadi 1:N adalah tidak adanya constraint `UNIQUE`, bukan jumlah penulisan foreign key dalam perintah.

Miskonsepsi ketiga, mengira penempatan foreign key di 1:1 bebas di sisi mana saja tanpa syarat. Ini setengah benar. Foreign key memang boleh ditaruh di salah satu sisi, tetapi harus ditambah `UNIQUE` agar relasinya benar-benar 1:1. Tanpa `UNIQUE`, relasinya akan menjadi 1:N.

3.3 Intuisi yang Benar

Di balik tiga miskonsepsi tersebut, ada satu intuisi yang sebenarnya sudah benar: dalam 1:1, nilai foreign key hanya boleh muncul paling banyak satu kali di kolomnya; dalam 1:N, nilai foreign key boleh muncul berkali-kali. Gambar 3 memvisualkan intuisi ini.

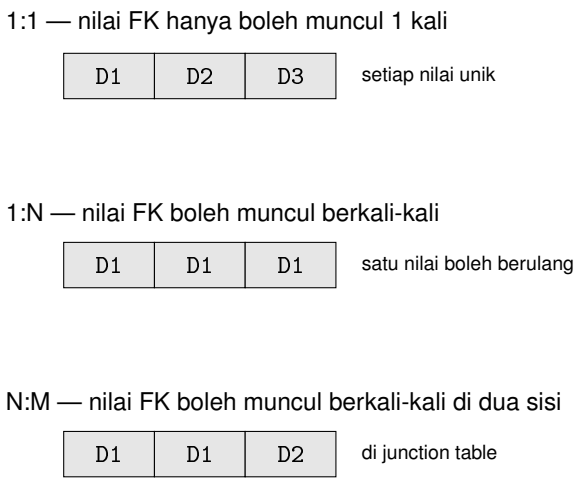


Figure 3. Intuisi yang benar tentang kemunculan nilai foreign key.

Yang perlu ditambahkan pada intuisi ini adalah bagaimana aturan tersebut ditegakkan. Untuk 1:1, penegaknya adalah `UNIQUE`. Untuk 1:N, penegaknya adalah ketiadaan `UNIQUE`. Untuk N:M, penegaknya adalah primary key komposit di junction table.

4 Anatomi Tiga Jenis Kardinalitas

4.1 Relasi 1:1: Foreign Key dengan UNIQUE

Relasi 1:1 berarti satu baris di tabel A berpasangan dengan paling banyak satu baris di tabel B, dan sebaliknya. Contoh paling sederhana adalah relasi mahasiswa dengan kartu tanda mahasiswa (KTM). Satu mahasiswa punya satu KTM, satu KTM hanya untuk satu mahasiswa.

Implementasinya memakai foreign key di salah satu tabel, ditambah constraint `UNIQUE`.

```
CREATE TABLE mahasiswa (
    id_mahasiswa INT PRIMARY KEY,
    nama VARCHAR(100) NOT NULL,
    id_ktm INT UNIQUE,
    FOREIGN KEY (id_ktm) REFERENCES ktm(id_ktm)
);
```

```
Query OK, 0 rows affected
```

Kata kunci `UNIQUE` pada kolom `id_ktm` inilah yang membuat relasinya menjadi 1:1. Tanpa `UNIQUE`, kolom tersebut bisa berisi nilai yang sama berkali-kali, dan relasinya menjadi 1:N. Gambar 4 menunjukkan anatomi lengkapnya.

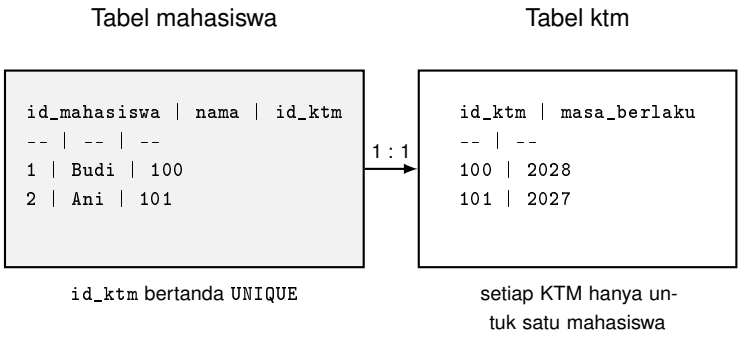


Figure 4. Anatomi relasi 1:1 dengan foreign key ber-UNIQUE.

4.2 Relasi 1:N: Foreign Key Tanpa UNIQUE

Relasi 1:N berarti satu baris di tabel A boleh berpasangan dengan banyak baris di tabel B, tetapi satu baris di tabel B hanya berpasangan dengan satu baris di tabel A. Contohnya adalah relasi dosen dengan mahasiswa dalam konteks bimbingan akademik. Satu dosen membimbing banyak mahasiswa, tetapi satu mahasiswa hanya dibimbing satu dosen.

Implementasinya memakai foreign key di sisi N, yaitu di tabel mahasiswa, tanpa `UNIQUE`.

```
CREATE TABLE mahasiswa (
    id_mahasiswa INT PRIMARY KEY,
    nama VARCHAR(100) NOT NULL,
    id_dosen INT,
    FOREIGN KEY (id_dosen) REFERENCES dosen(id_dosen)
);
```

```
Query OK, 0 rows affected
```

Perhatikan bahwa tidak ada `UNIQUE` pada kolom `id_dosen`. Inilah yang memungkinkan satu dosen muncul berkali-kali di kolom tersebut, sesuai dengan relasi 1:N. Gambar 5 menunjukkan anatomi lengkapnya.

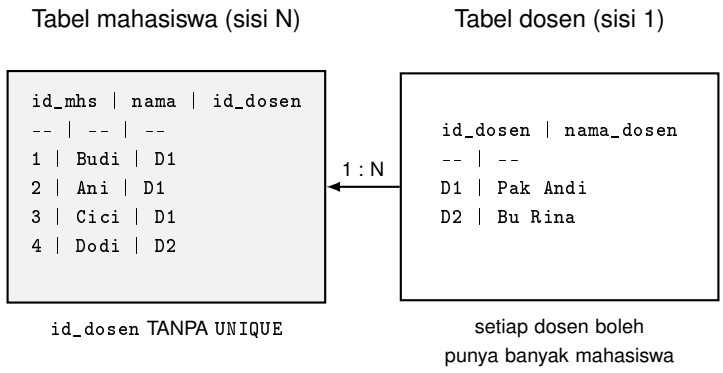


Figure 5. Anatomi relasi 1:N dengan foreign key di sisi N tanpa UNIQUE.

Nilai D1 muncul tiga kali di kolom `id_dosen`, dan itu sah. Jika constraint `UNIQUE` ditambahkan, insert kedua akan ditolak, dan relasinya berubah menjadi 1:1.

4.3 Relasi N:M: Tabel Penghubung

Relasi N:M berarti banyak baris di tabel A boleh berpasangan dengan banyak baris di tabel B, dan sebaliknya. Contohnya adalah relasi mahasiswa dengan mata kuliah. Satu mahasiswa mengambil banyak mata kuliah, satu mata kuliah diambil banyak mahasiswa.

Relasi ini tidak bisa diimplementasikan dengan foreign key di salah satu tabel, karena foreign key hanya bisa menampung satu nilai per baris. Solusinya adalah junction table, yaitu tabel baru yang berisi dua foreign key dan primary key komposit.

```
CREATE TABLE krs (
    id_mahasiswa INT,
    id_mk INT,
    semester INT,
    PRIMARY KEY (id_mahasiswa, id_mk),
    FOREIGN KEY (id_mahasiswa)
        REFERENCES mahasiswa(id_mahasiswa),
    FOREIGN KEY (id_mk)
        REFERENCES mata_kuliah(id_mk)
);
```

```
Query OK, 0 rows affected
```

Primary key komposit `(id_mahasiswa, id_mk)` mencegah pasangan yang sama muncul dua kali. Gambar 6 menunjukkan tiga tabel yang terhubung.

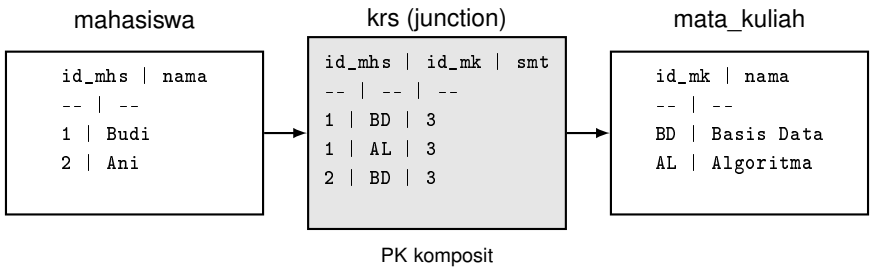


Figure 6. Relasi N:M melalui junction table dengan PK komposit.

4.4 Simulasi Insert: Kapan Constraint Menolak

Untuk melihat langsung bagaimana constraint bekerja, mari simulasikan insert ke dua skenario. Gambar 7 menampilkan hasilnya.

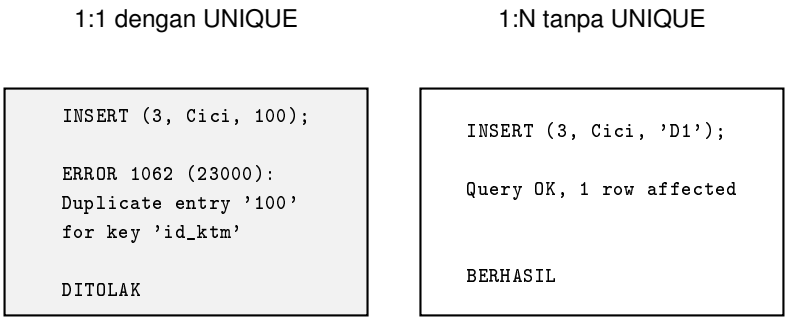


Figure 7. Simulasi insert: UNIQUE menolak duplikasi, tanpa UNIQUE mengizinkan.

Perhatikan dua hal. Pertama, insert ke skema 1:1 dengan nilai `id_ktm` yang sudah ada akan ditolak dengan pesan `Duplicate entry`. Kedua, insert ke skema 1:N dengan nilai `id_dosen` yang sudah ada akan berhasil. Inilah bukti bahwa `UNIQUE` adalah penegak kardinalitas 1:1.

5 Perbandingan dan Pohon Keputusan

Tabel 1 menyandingkan tiga jenis kardinalitas pada delapan aspek. Setelah tabel, Gambar 8 merangkum cara memilih kardinalitas dari aturan bisnis.

Table 1. Perbandingan tiga jenis kardinalitas.

Aspek	1:1	1:N	N:M
Definisi	satu berpasangan satu	satu berpasangan banyak	banyak berpasangan banyak
Contoh	mahasiswa dan KTM	dosen dan mahasiswa	mahasiswa dan mata kuliah
Penempatan FK	salah satu sisi	sisi N	junction table
Constraint kunci	`UNIQUE`	tanpa `UNIQUE`	PK komposit
Kemunculan FK	paling banyak sekali	berkali-kali	berkali-kali di dua sisi
Tabel tambahan	tidak perlu	tidak perlu	wajib junction
Kesalahan lazim	lupa `UNIQUE`	FK di sisi salah	tidak buat junction
Insert kedua FK sama	ditolak	diterima	lihat PK komposit

Dua paragraf penutup dari perbandingan ini. Pertama, perbedaan 1:1 dan 1:N terletak pada satu kata kunci saja, yaitu `UNIQUE`. Kedua, perbedaan 1:N dan N:M terletak pada keberadaan tabel ketiga. Keduanya adalah perbedaan struktural yang bisa langsung dibaca dari perintah `CREATE TABLE`, bukan dari jumlah penulisan dalam perintah.

Gambar 8 menyusun ketiga temuan tersebut menjadi satu pohon keputusan yang dapat dipakai saat membaca soal ujian.

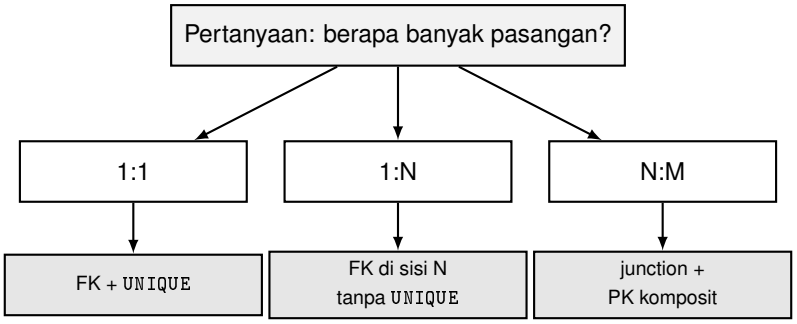


Figure 8. Pohon keputusan memilih kardinalitas dan penegaknya.

6 Studi Kasus: Membangun Skema dari Nol

Bagian ini mengikuti delapan langkah membangun skema dosen-mahasiswa dari nol, dengan setiap keputusan dijelaskan. Kebutuhan bisnisnya: sebuah kampus membutuhkan skema untuk mencatat dosen, mahasiswa, dan bimbingan akademik. Satu dosen membimbing banyak mahasiswa, satu mahasiswa hanya dibimbing satu dosen. Di akhir, aturan berubah untuk menguji dua skenario lain: 1:1 dan N:M.

6.1 Langkah 1: Buat Tabel Dosen

```
CREATE TABLE dosen (
    id_dosen INT PRIMARY KEY,
    nama_dosen VARCHAR(100) NOT NULL
);
```

```
Query OK, 0 rows affected
```

6.2 Langkah 2: Buat Tabel Mahasiswa dengan Foreign Key di Sisi N

```
CREATE TABLE mahasiswa (
    id_mahasiswa INT PRIMARY KEY,
    nama VARCHAR(100) NOT NULL,
    id_dosen INT,
    FOREIGN KEY (id_dosen)
        REFERENCES dosen(id_dosen)
);
```

```
Query OK, 0 rows affected
```

Foreign key ditaruh di tabel `mahasiswa` karena itu adalah sisi N. Tidak ada `UNIQUE` karena

relasinya 1:N.

6.3 Langkah 3: Insert Data Uji

```
INSERT INTO dosen VALUES (1, 'Pak Andi');
INSERT INTO dosen VALUES (2, 'Bu Rina');
INSERT INTO mahasiswa VALUES (1, 'Budi', 1);
INSERT INTO mahasiswa VALUES (2, 'Ani',  1);
INSERT INTO mahasiswa VALUES (3, 'Cici', 1);
```

6.4 Langkah 4: Verifikasi bahwa Relasinya Benar-Benar 1:N

```
SELECT * FROM mahasiswa WHERE id_dosen = 1;
```

```
+--------------+------+----------+
| id_mahasiswa | nama | id_dosen |
+--------------+------+----------+
|            1 | Budi |        1 |
|            2 | Ani  |        1 |
|            3 | Cici |        1 |
+--------------+------+----------+
3 rows in set
```

Tiga baris dengan `id_dosen` sama membuktikan bahwa relasinya 1:N. Verifikasi ini penting karena kardinalitas bukan sekadar niat di kertas, tetapi harus tampak di data.

6.5 Langkah 5: Naikkan ke 1:1 dengan UNIQUE

Jika aturan kampus berubah menjadi "satu dosen hanya membimbing satu mahasiswa", tambahkan `UNIQUE`.

```
ALTER TABLE mahasiswa
    ADD UNIQUE (id_dosen);
```

Perintah ini akan gagal jika sudah ada lebih dari satu baris dengan `id_dosen` yang sama. Pada data di atas, ada tiga baris dengan `id_dosen` = 1, sehingga perintah ini akan ditolak.

```
ERROR 1062 (23000): Duplicate entry '1' for key 'id_dosen'
```

Ini membuktikan bahwa `UNIQUE` adalah penegak kardinalitas 1:1. Tanpa pembersihan data terlebih dahulu, constraint tidak akan bisa ditambahkan.

6.6 Langkah 6: Turunkan Kembali ke 1:N

Andaikan data uji sudah dibersihkan sehingga hanya ada satu baris per dosen. Constraint `UNIQUE` bisa dihapus untuk kembali ke 1:N.

```
ALTER TABLE mahasiswa
    DROP INDEX id_dosen;
```

Setelah perintah ini, nilai `id_dosen` yang sama boleh muncul berkali-kali lagi. Relasi kembali menjadi 1:N. Perhatikan bahwa index yang dipakai `UNIQUE` memang berdiri sendiri di atas foreign key, sehingga menghapusnya mengembalikan perilaku default.

6.7 Langkah 7: Naikkan ke N:M dengan Junction Table

Jika aturan berubah menjadi "banyak dosen membimbing banyak mahasiswa", tidak ada cara untuk menyelesaikannya dengan satu foreign key. Perlu junction table.

```
CREATE TABLE bimbingan (
    id_dosen INT,
    id_mahasiswa INT,
    PRIMARY KEY (id_dosen, id_mahasiswa),
    FOREIGN KEY (id_dosen)
        REFERENCES dosen(id_dosen),
    FOREIGN KEY (id_mahasiswa)
        REFERENCES mahasiswa(id_mahasiswa)
);
```

```
Query OK, 0 rows affected
```

Primary key komposit `(id_dosen, id_mahasiswa)` mencegah pasangan yang sama muncul dua kali.

6.8 Langkah 8: Verifikasi Junction dan Primary Key Komposit

```
INSERT INTO bimbingan VALUES (1, 1);
INSERT INTO bimbingan VALUES (1, 2);
INSERT INTO bimbingan VALUES (2, 1);
SELECT * FROM bimbingan;
```

```
+----------+--------------+
| id_dosen | id_mahasiswa |
+----------+--------------+
|        1 |            1 |
```

```
|         1 |             2 |
|         2 |             1 |
+-----------+---------------+
3 rows in set
```

Dosen 1 membimbing mahasiswa 1 dan 2, dan mahasiswa 1 dibimbing dosen 1 dan 2. Ini adalah relasi N:M yang benar. Coba insert baris keempat dengan pasangan yang sama, misalnya `(1, 1)`, dan MariaDB akan menolak dengan pesan `Duplicate entry` karena melanggar primary key komposit.

6.9 Timeline Studi Kasus

Gambar 9 merangkum delapan langkah dalam satu garis waktu.

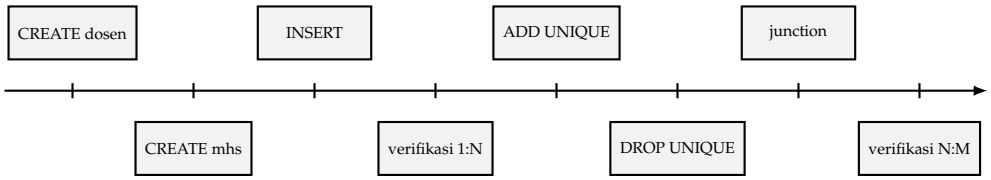


Figure 9. Timeline delapan langkah studi kasus.

7 Kesalahan Umum dan Cara Menghindarinya

Tabel 2 merangkum sepuluh kesalahan yang paling sering muncul, dampaknya, dan perbaikannya.

Table 2. Kesalahan umum pada kardinalitas dan atomik.

Kesalahan	Akibat	Perbaikan
daftar nama di satu sel	melanggar atomik	pecah jadi baris, pakai FK
foreign key di sisi salah pada 1:N	relasi jadi 1:1 atau rusak	taruh di sisi N
lupa `UNIQUE` pada 1:1	relasi jadi 1:N	tambah `UNIQUE` pada FK
tambah `UNIQUE` pada 1:N	relasi jadi 1:1	hapus `UNIQUE`
N:M tanpa junction	tidak bisa diimplementasikan	buat junction table
junction tanpa PK komposit	duplikat pasangan	tambah `PRIMARY KEY (a,b)`
`id_dosen` sebagai PK di mahasiswa	satu mahasiswa hanya satu, PK salah	gunakan PK sendiri
`id_mk` disimpan sebagai teks	melanggar atomik	pakai junction table
mengira `UNIQUE` sama dengan `PRIMARY KEY`	`UNIQUE` boleh NULL	PK tidak boleh NULL
`ON DELETE` tanpa aksi	FK orphan	pakai `ON DELETE CASCADE` atau `RESTRICT`

Dua kesalahan yang paling sering muncul saat ujian adalah lupa `UNIQUE` pada 1:1 dan menaruh foreign key di sisi yang salah pada 1:N. Keduanya bisa dideteksi dengan satu pertanyaan sederhana: setelah insert data uji, apakah nilai foreign key boleh berulang? Jika boleh, maka 1:N. Jika tidak, maka 1:1 dengan `UNIQUE`.

8 Lembar Rujukan Cepat

Tabel 3 berisi dua puluh operasi yang paling sering dipakai saat merancang atau memeriksa skema. Tabel ini ditujukan sebagai lembar contekan untuk ujian dan kerja.

Table 3. Rujukan cepat kardinalitas dan atomik.

Kebutuhan	Solusi
satu sel satu nilai	atomik, tidak ada daftar
relasi 1:1	FK di salah satu sisi + `UNIQUE`
relasi 1:N	FK di sisi N, tanpa `UNIQUE`
relasi N:M	junction table + PK komposit
relasi 1:1 dengan FK sebagai PK	PK/FK combo, otomatis `UNIQUE`
relasi opsional 1:1	FK `NULL` + `UNIQUE`
relasi wajib 1:1	FK `NOT NULL` + `UNIQUE`
relasi wajib 1:N	FK `NOT NULL`
relasi opsional 1:N	FK `NULL`
multi-nilai per baris	pecah ke tabel terpisah
aturan bisnis “hanya satu”	`UNIQUE`
aturan bisnis “boleh banyak”	tanpa `UNIQUE`
verifikasi 1:N	cek apakah FK berulang di data uji
verifikasi 1:1	coba insert kedua dengan FK sama
hapus anak saat induk dihapus	`ON DELETE CASCADE`
tolak hapus induk jika ada anak	`ON DELETE RESTRICT`
nilai FK jadi NULL saat induk dihapus	`ON DELETE SET NULL`
cek constraint yang ada	`SHOW CREATE TABLE`
cek index pada FK	`SHOW INDEX FROM tabel`
lihat foreign key	`SELECT * FROM information_schema.KEY_COLUMN_USAGE`

9 Kesimpulan

Atomik dan kardinalitas adalah dua konsep yang berbeda. Atomik berbicara tentang isi satu sel, sedangkan kardinalitas berbicara tentang aturan hubungan antar tabel. Pelanggaran atomik sering kali adalah gejala dari kesalahpahaman kardinalitas, seperti ketika seseorang mencoba membuat relasi 1:N dengan menaruh daftar nama di satu sel. Intuisi bahwa 1:1 berarti nilai foreign key hanya boleh muncul sekali, dan 1:N berarti boleh muncul berkali-kali, sebenarnya sudah benar. Yang perlu ditambahkan adalah mekanisme penegaknya: `UNIQUE` untuk 1:1, ketiadaan `UNIQUE` untuk 1:N, dan primary key komposit untuk N:M.

Tiga prinsip penutup:

1. Atomik dulu, baru kardinalitas. Pastikan setiap sel berisi satu nilai sebelum memikirkan relasi antar tabel. Daftar di dalam sel selalu salah, bukan hanya kurang rapi.

2. `UNIQUE` adalah pembeda 1:1 dan 1:N. Tanpa `UNIQUE`, foreign key apa pun akan berperilaku 1:N. Menambahkan `UNIQUE` adalah satu-satunya cara membuatnya menjadi 1:1.

3. N:M butuh tabel ketiga. Tidak ada jalan pintas. Junction table dengan primary key komposit adalah satu-satunya solusi yang benar, dan solusi ini sekaligus mencegah

duplikat pasangan tanpa perlu trigger tambahan.

Dengan kerangka mental ini, pertanyaan ujian berbentuk “tentukan kardinalitas” atau “konversi ERD menjadi tabel” dapat dijawab dengan alasan yang bisa dipertanggungjawabkan, bukan dengan hafalan.

Referensi

References

[1] Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), 377–387.

[2] Date, C. J. (2003). *An Introduction to Database Systems* (8th ed.). Addison-Wesley.

[3] Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson.

[4] Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.

[5] MariaDB Foundation. (2026). *SQL Statements — CREATE TABLE, ALTER TABLE, DROP TABLE*. MariaDB Knowledge Base. `https://mariadb.com/kb/en/sql-statements/`

[6] Oracle Corporation. (2026). *MySQL 8.0 Reference Manual — Data Definition Statements*. `https://dev.mysql.com/doc/refman/8.0/en/sql-data-definition-statements.html`

[7] The PostgreSQL Global Development Group. (2026). *SQL Commands — CREATE TABLE, ALTER TABLE*. `https://www.postgresql.org/docs/current/sql-commands.html`

[8] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.