M05-01 · GIBALJA VISUAL MANUAL

데이터를 표·관계·규칙으로 설계하기

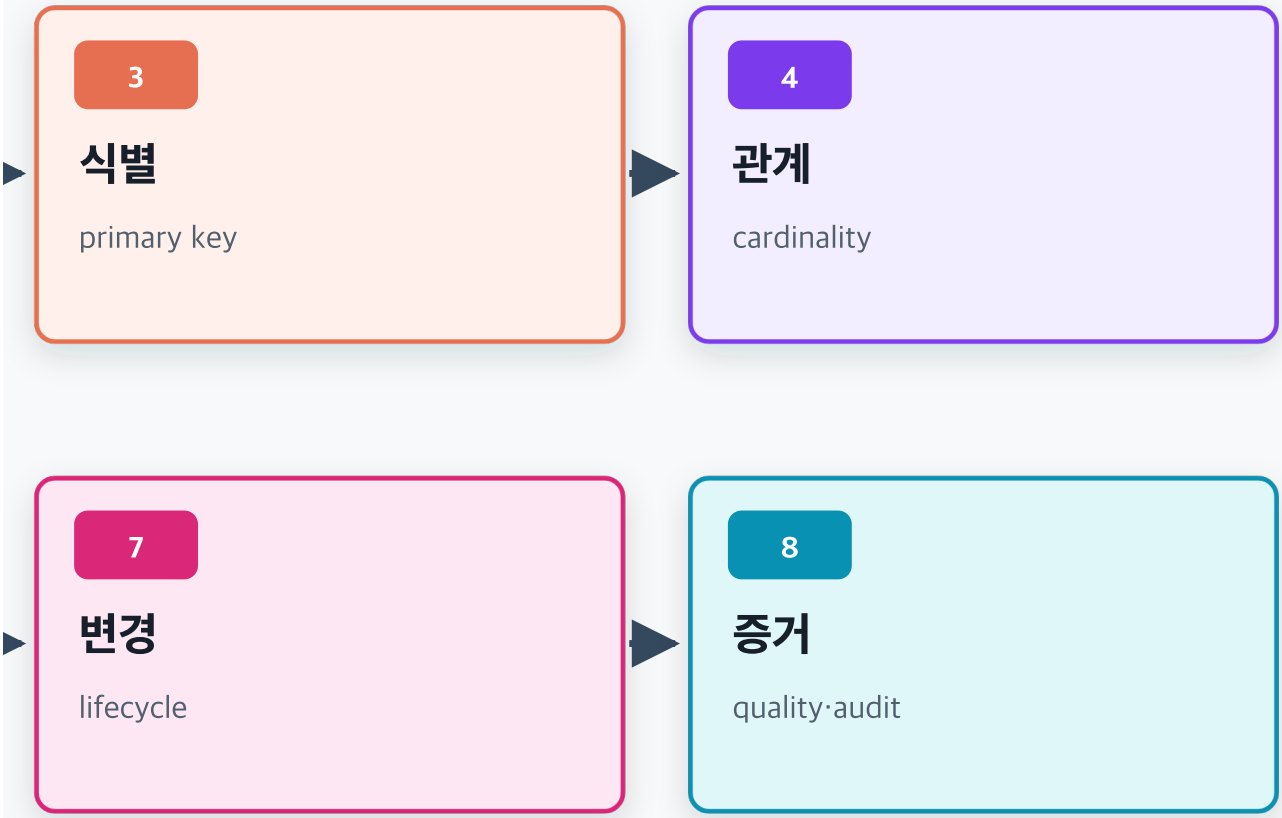
> **한 문장 목표:** 화면의 입력 칸을 그대로 열로 옮기지 않고, 어떤 업무 사실을 한 행으로 남길지, 그 행을 무엇으로 식별할지, 다른 사실과 몇 개씩 연결되는지, 어떤 값만 허용할지, 변경 뒤 무엇을 증거로 남길지를 하나의 데이터 계약으로 작성합니다.

난이도	개념	실습	셀프 테스트	최종 산출물
Level 1	60분	60분	15분	데이터 항목 정의서, 관계 지도, 제약·품질표, lifecycle 시나리오

"회원 화면에 이름·이메일·신청 강좌 세 칸이 있으니 회원 표에 `course_1` · `course_2` · `course_3`을 만들자"는 설계는 네 번째 강좌에서 무너집니다. 좋은 모델은 회원과 강좌를 따로 식별하고, "한 회원의 한 강좌 신청"을 별도 행으로 남깁니다. 화면은 바뀌어도 업무 사실과 규칙은 오래 살아야 합니다.

그림 1. 데이터 모델은 표 모양보다 여덟 가지 결정의 일관성입니다. 앞 결정이 뒤 제약과 품질 지표로 이어져야 합니다.


1
대상
entity·event
2
한 행
grain
5
값
type·unit
6
제약
integrity
좋은 표 = 한 행의 뜻 + 안정적인 ID + 명시



적 관계 + DB가 지키는 규칙 + 변경 증거

0. 이 PDF를 공부하는 방법

1회차 · 그림만 읽기 · 20분

각 그림의 제목과 검은 결론 띠만 읽습니다. 다음 여덟 문장을 소리 내어 말합니다.

```
화면 field와 database column은 같은 것이 아니다.
한 table의 한 행은 하나의 업무 사실을 뜻한다.
이름과 email은 표시값이지 언제나 row의 생명은 아니다.
관계는 선 하나가 아니라 양쪽의 0·1·N 계약이다.
NULL·빈 문자열·0·false는 서로 다른 사실이다.
UI 검증만으로 database 무결성을 지킬 수 없다.
현재 상태와 변경 사건과 감사 증거는 목적이 다르다.
품질은 느낌이 아니라 metric과 threshold로 운영한다.
```

2회차 · 사례 row를 손으로 쓰기 · 25분

회원 2명, 강좌 2개, 수강 신청 3개, 레슨 진도 4개를 종이에 적습니다. 각 표 위에 `한 행 = ...` 문장을 씁니다.

3회차 · 실습기에서 모델 깨뜨리기 · 50분

데이터 모델 계약 스튜디오를 열어 15개 시나리오를 실행합니다. 한 값만 바꾸고 어느 gate와 품질 지표가 달라졌는지 설명합니다.

4회차 · 내 기능을 데이터 계약으로 바꾸기 · 40분

데이터 모델 계약서를 채웁니다. 마지막에 셀프 테스트를 풀고 정답과 비교합니다.

1. 데이터 모델은 화면의 저장 위치가 아니라 업무 사실의 계약입니다

같은 업무가 모바일·웹·관리자 화면·API·batch에 여러 모습으로 나타날 수 있습니다. 화면마다 별도 저장 구조를 만들면 같은 회원·강좌·신청의 의미가 갈라집니다.

화면 문장	곧바로 만든 열	빠진 질문
신청 강좌를 세 개까지 고릅니다	`course_1` · `course_2` · `course_3`	네 번째 강좌는 어떻게 합니까

화면 문장	곧바로 만든 열	빠진 질문
태그를 쉼표로 입력합니다	`tags = "AI,기획"`	태그 존재·중복·이름 변경을 누가 지킵니까
이메일로 로그인합니다	`email PRIMARY KEY`	이메일이 바뀌거나 재사용되면 참조는 어떻게 됩니까
상태를 적습니다	`status TEXT`	허용 상태와 전이는 무엇입니까
금액을 입력합니다	`amount FLOAT`	통화·소수 자릿수·정확한 합계는 무엇입니까
삭제 버튼이 있습니다	`ON DELETE CASCADE`	주문·수강·감사 이력도 함께 지워야 합니까

같은 화면도 여러 사실을 보여 줍니다

```
수강 화면
├─ 회원이라는 대상
├─ 강좌라는 대상
├─ 회원이 강좌를 신청한 사건·관계
├─ 강좌 안의 레슨이라는 구성
└─ 회원별 레슨 진도라는 변화하는 상태
```

한 화면이 한 table이라는 법은 없습니다. 반대로 여러 화면이 같은 table의 서로 다른 view일 수도 있습니다.

30초 확인 1
회원 상세 화면에 회원·수강 신청·강좌·진도가 함께 보인다고 해서 한 table로 저장하면 어떤 grain이 섞입니까?

2. 여덟 gate로 빠진 결정을 찾습니다

그림 1의 여덟 gate는 국제 표준의 보편적 architecture가 아닙니다. 비전공 기획자가 데이터 설계 회의에서 빠뜨리기 쉬운 질문을 찾기 위한 **YEONCORE 학습용 검수 프레임**입니다.

gate	결정 질문	남길 증거
1 대상	사람·사물·사건·상태 중 무엇입니까	entity·event 후보 목록
2 한 행	한 row가 정확히 무엇을 뜻합니까	`한 행 = ...` 문장·사례 row
3 식별	시간이 지나도 같은 row를 무엇으로 가리킵니까	primary key·업무 UNIQUE

gate	결정 질문	남길 증거
4 관계	양쪽에서 최소·최대 몇 개입니까	cardinality·optionality·FK
5 값	허용 type·format·unit·NULL 의미는 무엇입니까	data dictionary
6 제약	잘못된 값을 어느 층에서 막습니까	NOT NULL·UNIQUE·CHECK·FK
7 변경	생성·상태 변경·삭제 때 무엇을 보존합니까	lifecycle·referential action
8 증거	품질과 변경을 어떻게 측정·추적합니까	metric·threshold·audit

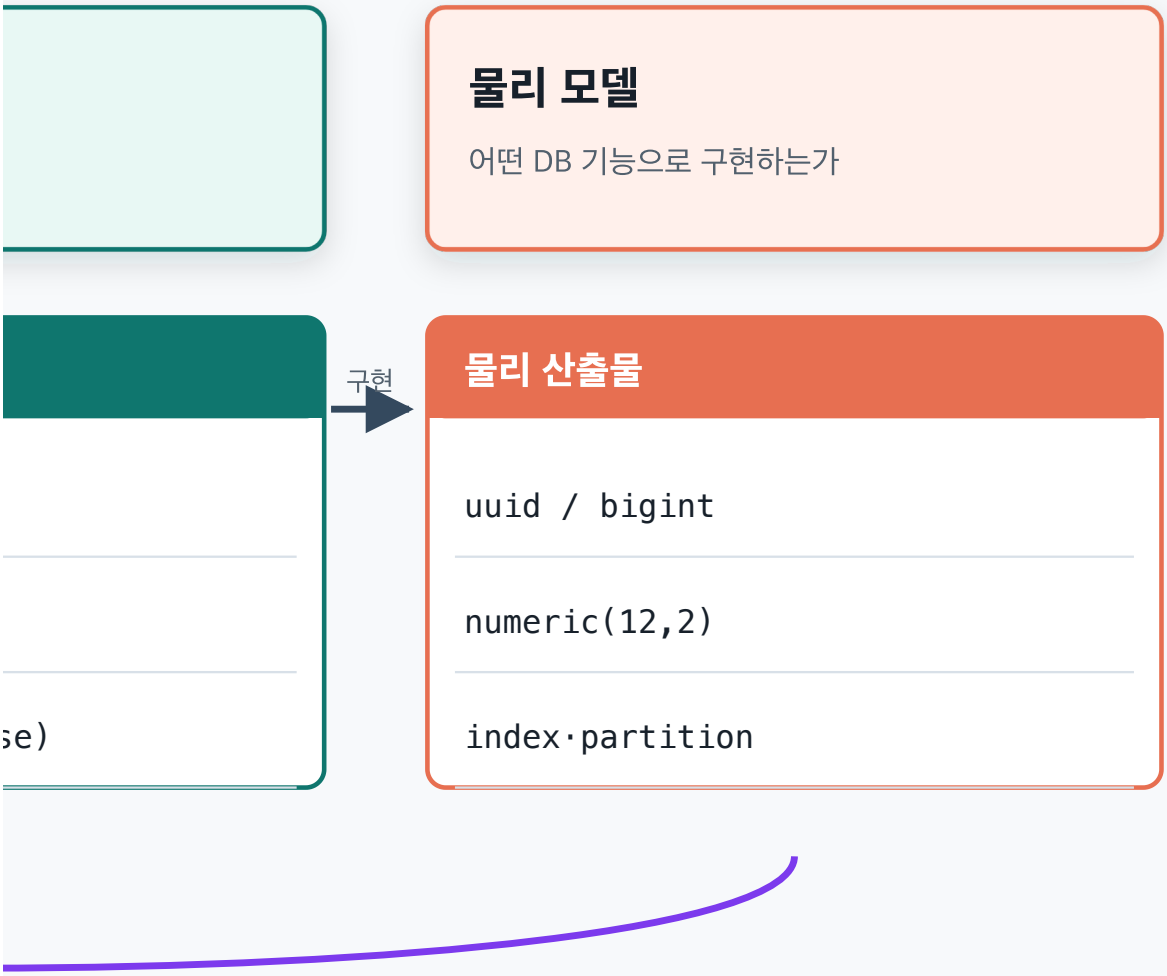
기획자의 한 문장 데이터 계약

```
[table]의 한 행은 [업무 사실]을 뜻하며 [primary key]로 식별한다.
[parent]와 [최소..최대] 관계이고 [foreign key / bridge]로 연결한다.
[attribute]는 [type·unit·NULL·domain]만 허용하며 [constraint]로 지킨다.
[생성·변경·삭제] 시 [event·audit·quality evidence]를 남긴다.
```

3. 개념·논리·물리 모델을 구분합니다

그림 2. 같은 업무를 세 해상도로 봅니다. 초급 기획자의 주력 산출물은 개념·논리 모델이며, 물리 구현은 개발자·DB 담당자와 함께 결정합니다.


개념 모델
무엇이 존재하는가
논리 모델
어떻게 식별·연결하는가
개념 산출물
회원
강좌
수강 신청
정교화
논리 산출물
member_id PK
course_id FK
UNIQUE(member, cours


물리 모델
어떤 DB 기능으로 구현하는가
구현
물리 산출물
uuid / bigint
numeric(12,2)
index·partition
se)

3.1. 개념 모델 · 무엇이 존재합니까

업무 사람이 쓰는 말로 대상과 관계를 합의합니다. 아직 `varchar(100)`이나 index를 정하지 않습니다.

3.2. 논리 모델 · 어떻게 식별하고 연결합니까

table·attribute·key·cardinality·constraint를 제품 중립적인 수준에서 정합니다.

논리 질문	예
한 행의 뜻	`enrollment` 한 행 = 한 회원의 한 강좌 신청
row 식별	`enrollment_id`
parent 연결	`member_id`, `course_id` foreign key
업무 중복	`UNIQUE(member_id, course_id)`
상태 허용값	`pending`, `active`, `completed`, `canceled`

3.3. 물리 모델 · 선택한 DB에서 어떻게 구현합니까

PostgreSQL의 구체 type, index, partition, generated column, constraint timing, migration 순서처럼 제품·버전·운영 조건이 들어갑니다.

Microsoft의 Entity Framework 개요도 domain model, relational logical model, physical model을 구분합니다. 이 세 단계는 일방통행이 아닙니다. 물리 성능·제품 제약이 논리 가정을 바꾸면 개념 모델까지 되돌아가 확인합니다.

이번 매뉴얼의 경계

여기서 다룸	다음 매뉴얼로 넘김
entity·attribute·relationship	SQL 조회·추가·수정·삭제 실습 M05-02
PK·FK·UNIQUE·CHECK 의미	query·join·transaction 조작 M05-02
삭제 관계의 업무 의미	보존·백업·복구·파기 정책 M05-03

여기서 다룸	다음 매뉴얼로 넘김
품질 metric 설계	운영 자동화·대시보드 구현은 이후 과정

4. 업무 문장을 데이터 후보로 분해합니다

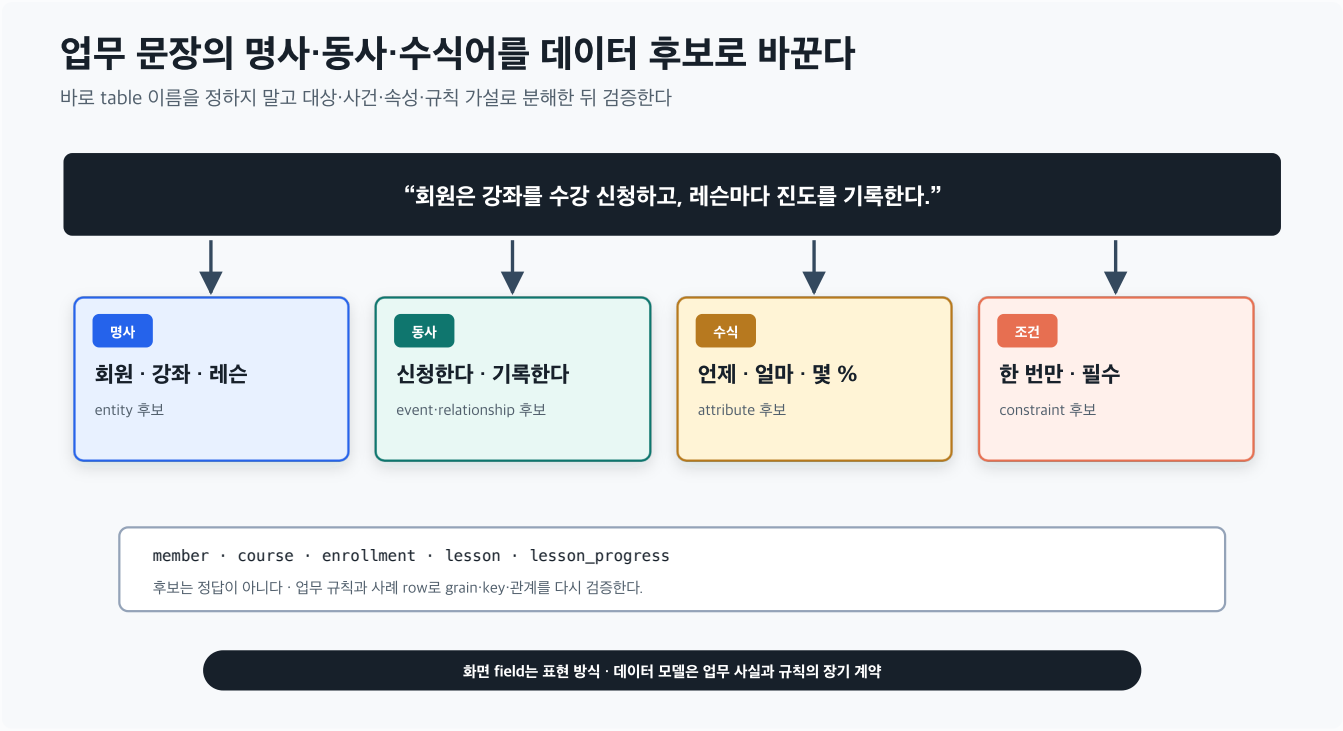
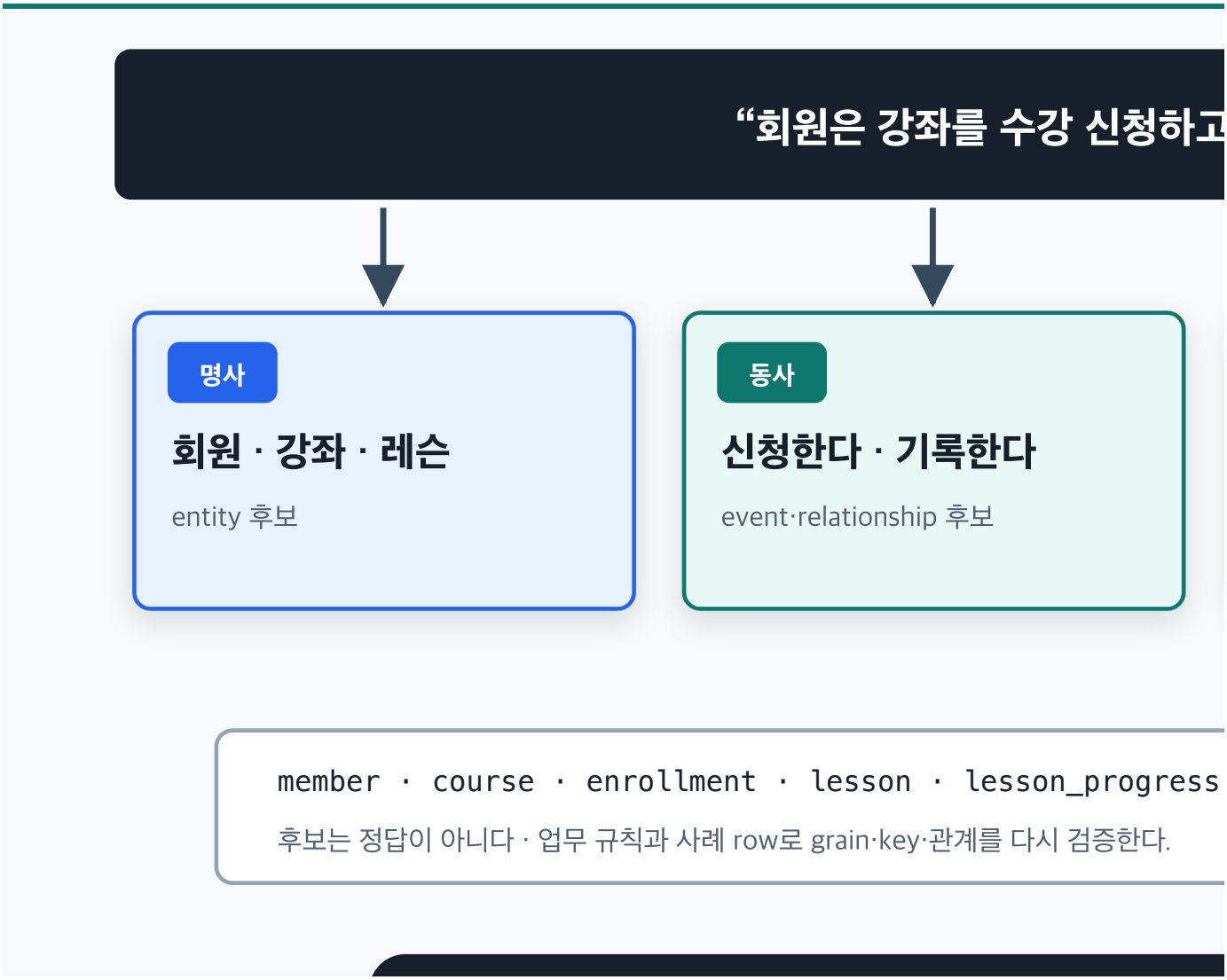


그림 3. 명사는 entity, 동사는 event·relationship, 수식어는 attribute, 조건은 constraint 후보가 됩니다. 후보는 사례로 검증하기 전까지 가설입니다.


“회원은 강좌를 수강 신청하고
명사
회원 · 강좌 · 레슨
entity 후보
동사
신청한다 · 기록한다
event·relationship 후보
member · course · enrollment · lesson · lesson_progress
후보는 정답이 아니다 · 업무 규칙과 사례 row로 grain·key·관계를 다시 검증한다.


, 레슨마다 진도를 기록한다.”
수식
언제 · 얼마 · 몇 %
attribute 후보
조건
한 번만 · 필수
constraint 후보

4.1. 문장에 색을 칠합니다

> **회원**은 **강좌**를 **수강 신청**하고, **레슨마다 진도**를 **기록**한다. 같은 회원은 같은 강좌에 **한 번만** 신청한다.

표현	후보	검증 질문
회원·강좌·레슨	entity	독립적으로 식별·변경·참조합니까
수강 신청	event·bridge entity	자체 상태·시점·금액이 있습니까
기록	event	여러 번 생깁니까, 현재값만 필요합니까
진도	attribute 또는 snapshot	어느 대상의 어느 시점 값입니까
레슨마다	relationship	한 강좌에 레슨이 몇 개입니까
한 번만	uniqueness constraint	범위가 전체입니까, 회원·강좌 조합입니까

4.2. 네 종류를 구분합니다

종류	판별 질문	수강 사례
entity	다른 사실이 이 대상을 참조합니까	member, course, lesson
value object·attribute	owner 밖에서 독립 생명이 필요합니까	title, email, amount
event·transaction	언제 일어났고 자체 상태·금액·이유가 있습니까	enrollment, payment, refund
snapshot·state	특정 시점의 현재값을 빠르게 봅니까	current progress, course status

4.3. 동사를 무시하면 관계 table을 놓칩니다

“회원과 강좌”만 보면 둘 사이 선 하나로 끝나기 쉽습니다. “신청한다”에는 신청일·가격·상태·취소 이유가 붙습니다. 그래서 `enrollment`는 단순 접착제가 아니라 업무 사건입니다.

4.4. 형용사를 독립 table로 만들지 않습니다

`active`, `premium`, `completed` 같은 표현은 곧바로 entity가 아닙니다. 안정적인 작은 허용 집합인지, 운영자가 관리하는 code인지, 시간에 따라 바뀌는 event인지부터 구분합니다.

30초 확인 2

“회원은 강좌를 찜한다”에서 `favorite`를 단순 boolean column으로 둘 때와 별도 row로 둘 때의 차이는 무엇입니까?

5. 한 행의 뜻인 grain을 먼저 고정합니다

그림 4. `enrollment`와 `lesson_progress`는 row가 생기는 이유가 다릅니다. 한 table에 섞으면 key·NULL·집계가 흔들립니다.

나쁜 표 · mixed_grain

member_id	course_id	lesson_id	progress
mem_81	crs_7	les_1	100
mem_81	crs_7	les_2	40
mem_81	crs_9	NULL	NULL

수강 신청과 레슨 진도가 한 표에 섞임

한 행을 “신청”이라고도 “진도”라고도 설명해야 한다.

enrollment
enrollment_id PK
member_id FK
course_id FK
status
enrolled_at

lesson_progress
progress_id PK
enrollment_id FK
lesson_id FK
percent
updated_at

표마다 하나의 grain

한 행 = 한 회원의 한 강좌 신청 / 한 레슨의 한 진도

5.1. 모든 table 위에 한 문장을 씁니다

```
member          한 행 = 한 회원
course          한 행 = 한 판매·학습 강좌
lesson          한 행 = 한 강좌 안의 한 학습 단위
enrollment      한 행 = 한 회원의 한 강좌 신청
lesson_progress 한 행 = 한 수강 신청의 한 레슨 현재 진도
```

5.2. grain이 섞였다는 신호

신호	숨은 문제
한 행을 “A 또는 B”라고 설명함	서로 다른 사건·entity 혼합
일부 row만 쓰는 column이 많음	여러 subtype·grain 혼합 가능성
같은 사실이 row마다 반복됨	owner가 다른 attribute
합계를 내면 join 전부터 중복됨	one 쪽 값이 many 쪽마다 복제됨
primary key 후보를 찾을 수 없음	한 행의 차이를 정의하지 못함

5.3. 사례 row 세 개로 검증합니다

enrollment_id	member_id	course_id	status	paid_amount
enr_81	mem_1	crs_7	active	42000.00
enr_82	mem_1	crs_9	pending	0.00
enr_83	mem_2	crs_7	completed	42000.00

다음 질문을 답합니다.

1. 두 row는 무엇이 달라서 별도 행입니까?
2. 어떤 사건에서 새 row가 생깁니까?
3. 어떤 변화는 새 row가 아니라 기존 row 변경입니까?
4. 취소 뒤 같은 강좌를 재신청하면 새 row입니까, 같은 row 재활성화입니까?

마지막 질문에 따라 `UNIQUE(member_id, course_id)`가 맞을 수도, `UNIQUE(member_id, course_id, enrollment_round)`가 맞을 수도 있습니다. constraint는 일반 상식이 아니라 **우리 업무의 시간 규칙**입니다.

5.4. 집계 grain도 말로 확인합니다

“회원별 완료 강좌 수”와 “강좌별 완료 회원 수”는 원본 grain은 같아도 집계 방향이 다릅니다. M05-02에서 SQL을 쓰기 전에 원본 한 행의 뜻을 먼저 고정해야 중복 합계를 피할 수 있습니다.

6. row의 생명을 식별자로 고정합니다

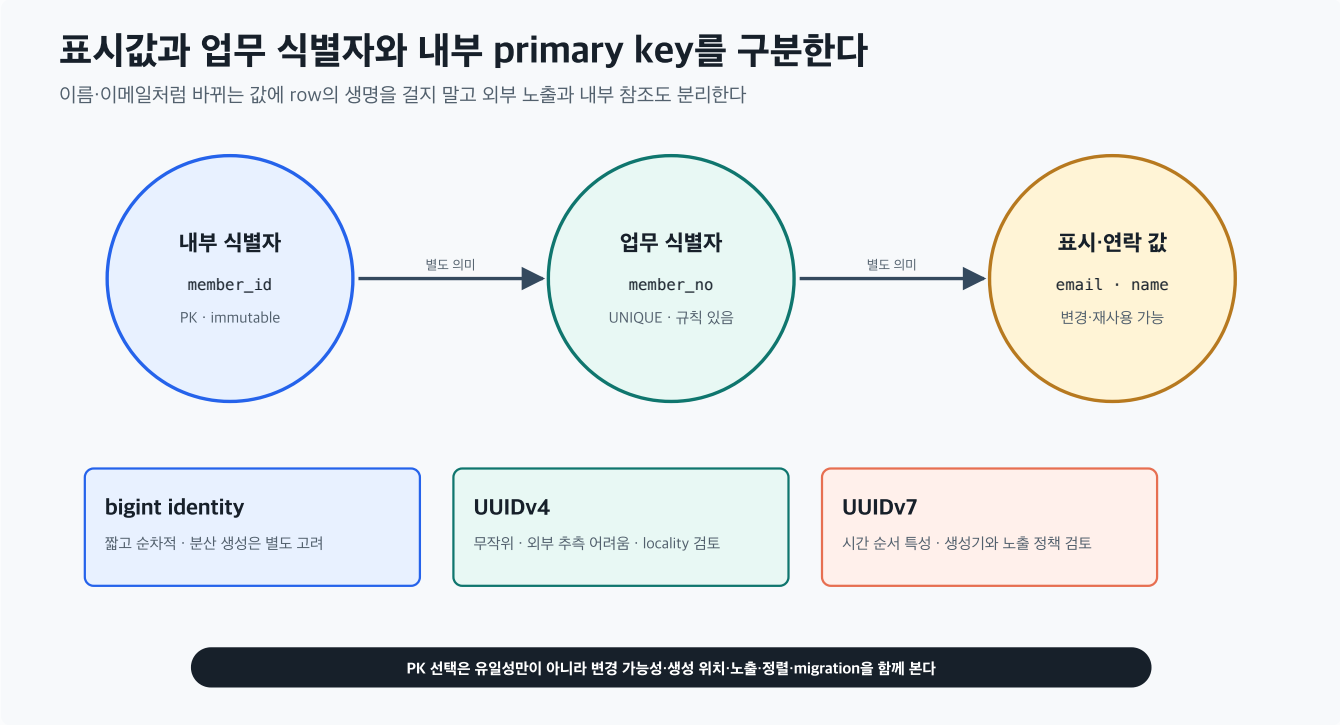
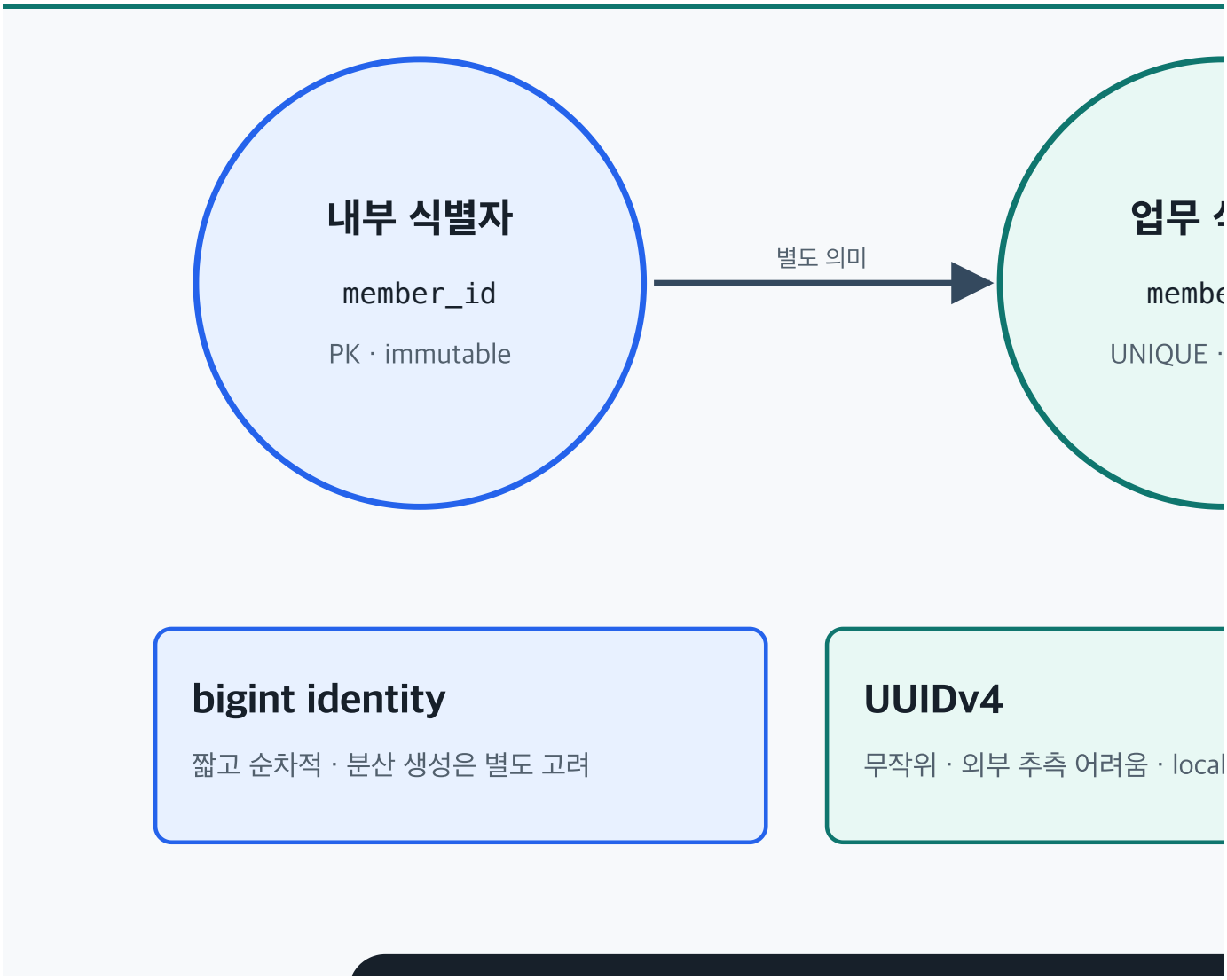
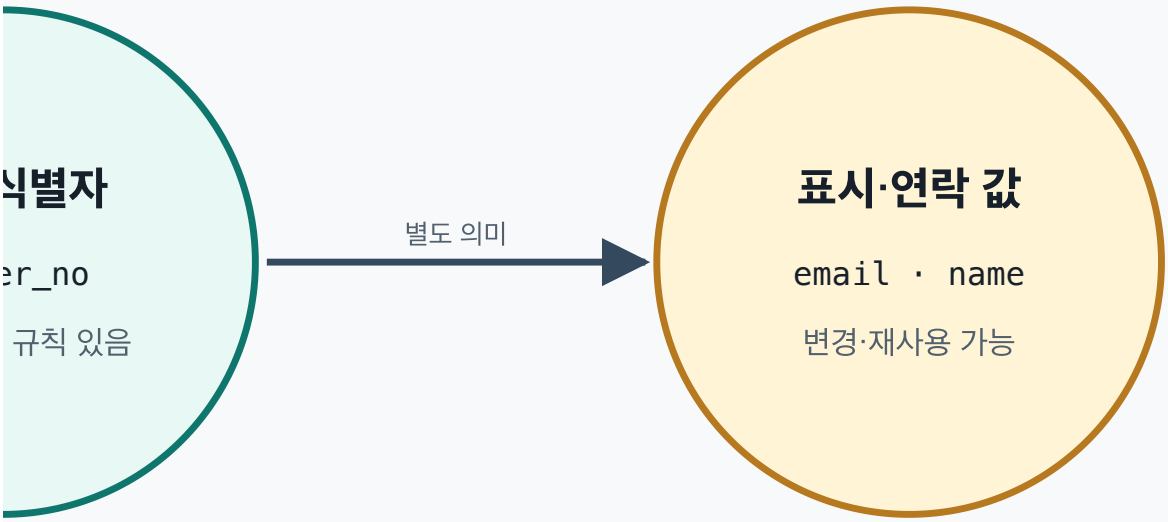


그림 5. 내부 PK, 업무상 유일한 번호, 사용자에게 보이는 값은 목적이 다릅니다. 하나의 값에 세 책임을 몰지 않습니다.


내부 식별자
member_id
PK · immutable
별도 의미
업무 식
membe
UNIQUE ·
bigint identity
짧고 순차적 · 분산 생성은 별도 고려
UUIDv4
무작위 · 외부 추측 어려움 · local



lity 검토

UUIDv7

시간 순서 특성 · 생성기와 노출 정책 검토

6.1. primary key의 역할

PostgreSQL 문서에서 primary key는 한 행을 고유하게 식별하는 column 또는 column 묶음이며 unique와 not null을 함께 요구합니다. 한 table에는 primary key를 하나만 선언할 수 있지만, unique constraint는 여러 개 둘 수 있습니다.

```
CREATE TABLE enrollment (
  enrollment_id bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  member_id bigint NOT NULL,
  course_id bigint NOT NULL,
  UNIQUE (member_id, course_id)
);
```

`enrollment_id`는 row identity이고, `(member_id, course_id)`는 업무 중복 규칙입니다. 둘은 같은 질문이 아닙니다.

6.2. 좋은 key 후보의 질문

질문	확인할 위험
정말 유일합니까	동명이인·중복 코드·범위별 재사용
시간이 지나도 변하지 않습니까	이메일·전화번호·사업자 정책 변경
생성 주체가 하나입니까	offline·분산 시스템·외부 import 충돌
외부에 노출해도 됩니까	순차 ID 추측·내부 규모 노출
merge·migration에서 유지됩니까	시스템 간 key 충돌·재매핑
사람이 입력합니까	오타·앞자리 0·대소문자·공백

6.3. 자연 key와 surrogate key

선택	장점	주의
자연·업무 key	의미가 있고 중복 규칙을 직접 표현	정책 변화·길이·복합 key·외부 재사용
surrogate key	짧고 immutable하게 설계 가능	업무 중복을 별도 UNIQUE로 막아야 함
복합 primary key	관계의 정체성을 직접 표현	child FK가 길어지고 시간 규칙 추가가 어려울 수 있음

정답은 하나가 아닙니다. 초급 과정에서는 **안정적인 내부 PK + 실제 업무 중복을 막는 UNIQUE**를 기본 검토안으로 사용하되, 기존 표준 번호와 통합 조건이 있으면 개발자·데이터 담당자와 함께 선택합니다.

6.4. identity column은 자동 생성이지 자동 유일성 보장이 아닙니다

PostgreSQL의 identity column은 내부 sequence로 값을 자동 생성할 수 있습니다. 그러나 identity column 자체가 자동으로 unique임을 뜻하지는 않습니다. key로 쓸 때는 primary key 또는 unique constraint를 함께 선언합니다.

6.5. UUID도 계약이 필요합니다

RFC 9562는 UUID를 128-bit 식별자로 정의하고 v4·v7 등 여러 layout과 생성 고려사항을 설명합니다. UUID를 쓴다는 말만으로 다음 결정이 끝나지 않습니다.

결정	질문
version	무작위 v4, 시간 순서 특성이 있는 v7 중 무엇입니까
생성 위치	client·API·DB 중 누가 만듭니까
노출	public resource ID로 그대로 씁니까
정렬	UUID 순서를 업무 발생 순서로 오해하지 않습니까
충돌·보안	안전한 생성기와 opaque 처리 정책을 확인했습니까

RFC 9562는 UUID가 본질적으로 접근 권한이나 비밀이 아님을 전제로 봐야 합니다. ID를 알았다는 이유로 resource 접근을 허용하지 않습니다.

30초 확인 3
surrogate PK를 추가했는데도 같은 회원의 같은 강좌 신청이 두 번 생길 수 있는 이유는 무엇입니까?

7. 관계는 양쪽에서 최소·최대 개수를 묻습니다

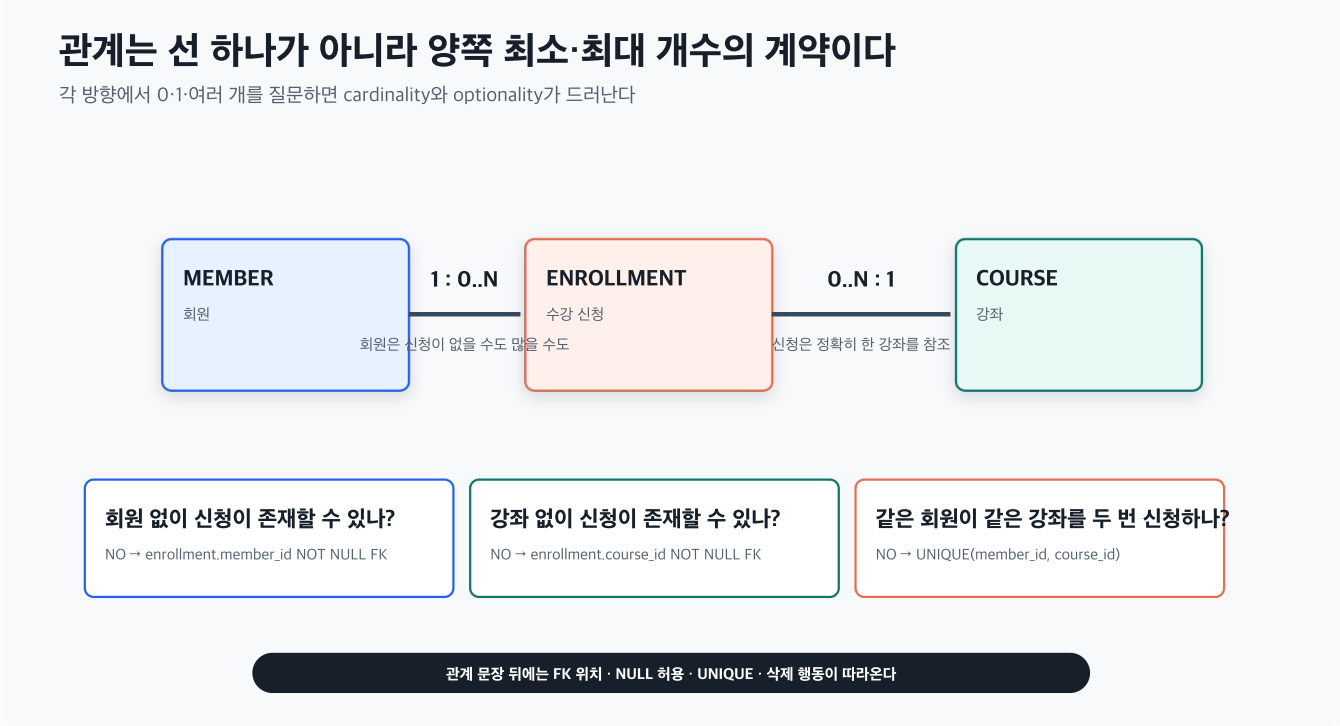
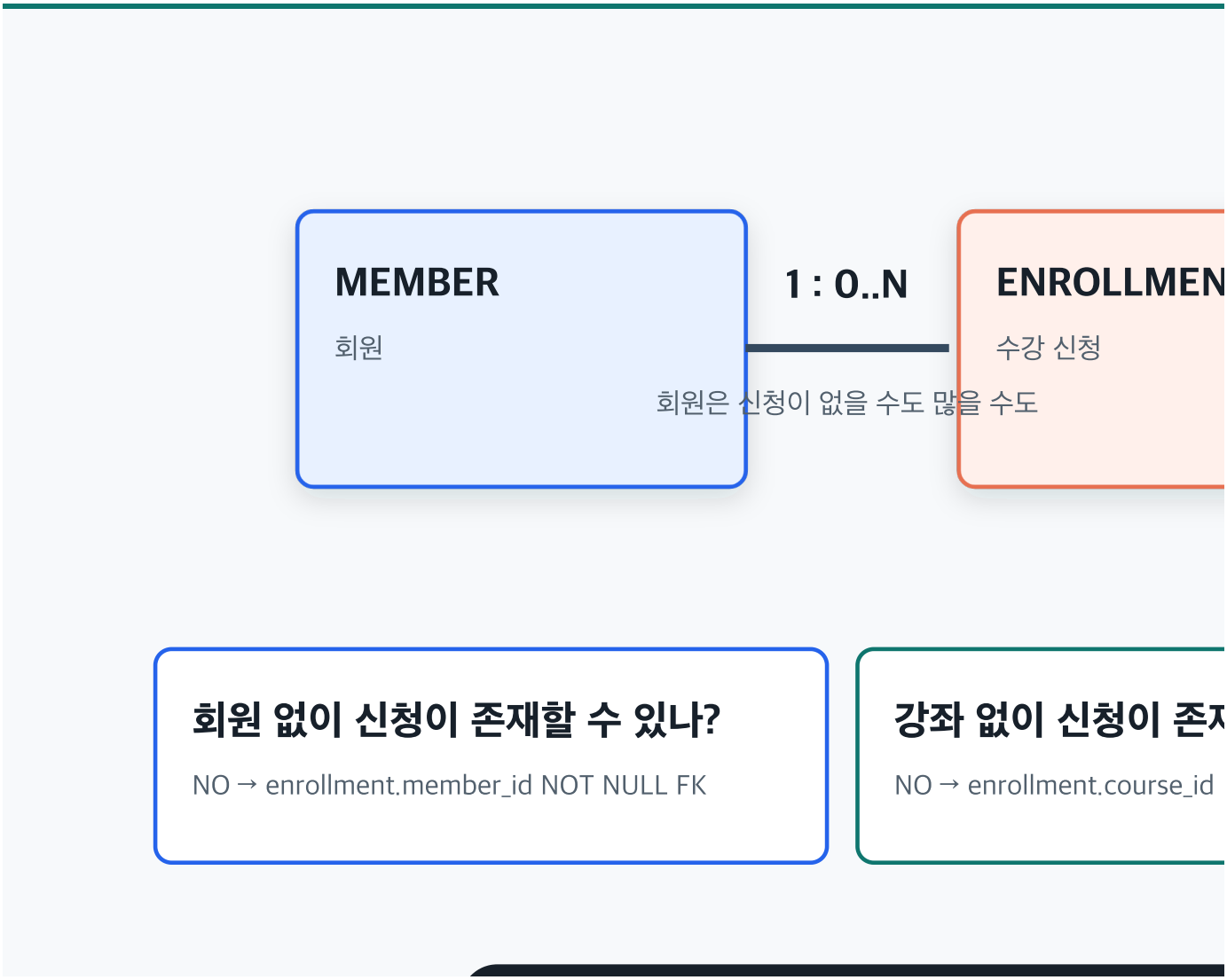
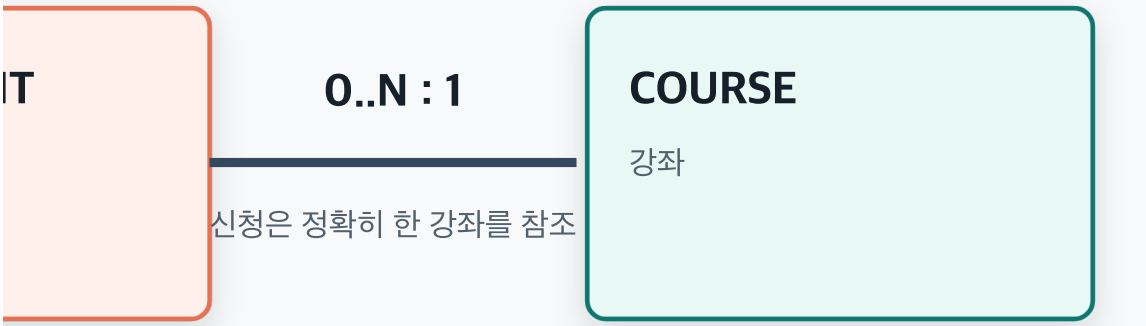


그림 6. 관계는 `member_id` 선 하나가 아닙니다. 각 방향의 0·1·N, NULL 허용, FK, UNIQUE, 삭제 행동이 한 묶음입니다.


MEMBER
회원
1 : 0..N
회원은 신청이 없을 수도 많을 수도
ENROLLMEN
수강 신청
회원 없이 신청이 존재할 수 있나?
NO → enrollment.member_id NOT NULL FK
강좌 없이 신청이 존
NO → enrollment.course_id



해할 수 있나?

NOT NULL FK

같은 회원이 같은 강좌를 두 번 신청하나?

NO → UNIQUE(member_id, course_id)

7.1. 두 방향으로 읽습니다

```
회원 한 명은 수강 신청이 0개 이상이다.
수강 신청 하나는 정확히 한 회원에 속한다.

강좌 하나는 수강 신청이 0개 이상이다.
수강 신청 하나는 정확히 한 강좌에 속한다.
```

7.2. cardinality와 optionality

표기	뜻	구현 단서
0..1	없어도 되고 최대 하나	nullable FK 또는 별도 1:1 table
1..1	반드시 정확히 하나	NOT NULL FK, 필요 시 UNIQUE
0..N	없어도 되고 여러 개	child table의 FK
1..N	적어도 하나 이상	FK만으로 “최소 한 개”까지는 자동 보장되지 않을 수 있음

Microsoft EF Core 관계 문서는 relational model에서 PK와 FK가 entity 사이의 관계를 표현하며, FK 값이 parent PK와 맞도록 constraint된다고 설명합니다.

7.3. one-to-one은 UNIQUE가 필요합니다

`member_profile.member_id`가 FK이기만 하면 한 회원에 profile 여러 개가 생길 수 있습니다. 정말 1:1이면 child FK에 `UNIQUE`를 더합니다.

```
CREATE TABLE member_profile (
  profile_id bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  member_id bigint NOT NULL UNIQUE REFERENCES member(member_id)
);
```

7.4. 관계 이름을 동사로 읽습니다

모호한 선	읽을 수 있는 관계
member — course	member **enrolls in** course
course — lesson	course **contains** lesson

모호한 선	읽을 수 있는 관계
enrollment — payment	enrollment **is paid by** payment
member — member	member **is managed by** member

self-referencing FK는 조직도·댓글·카테고리 tree처럼 같은 table 안의 parent를 표현할 수 있습니다. root의 `parent_id`가 NULL인지, cycle을 어떻게 막을지 별도 계약합니다.

8. 다대다는 연결 table로 업무 사건을 드러냅니다

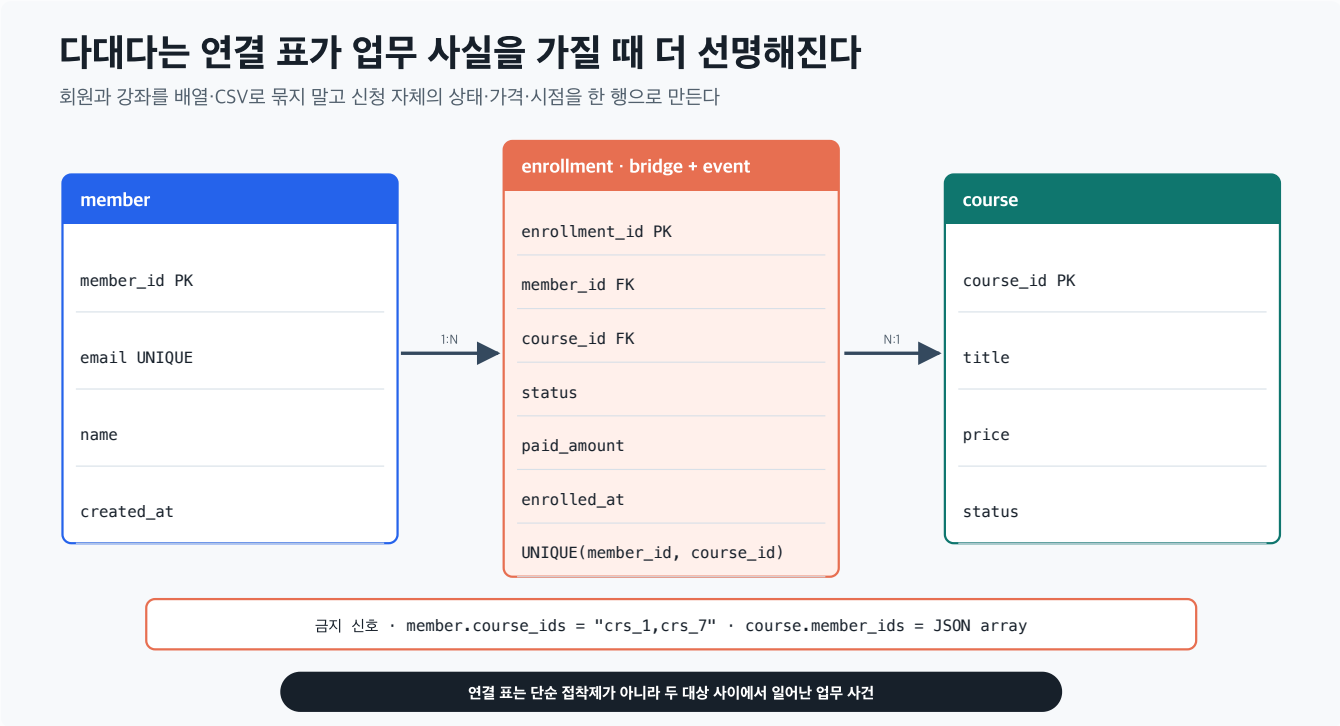
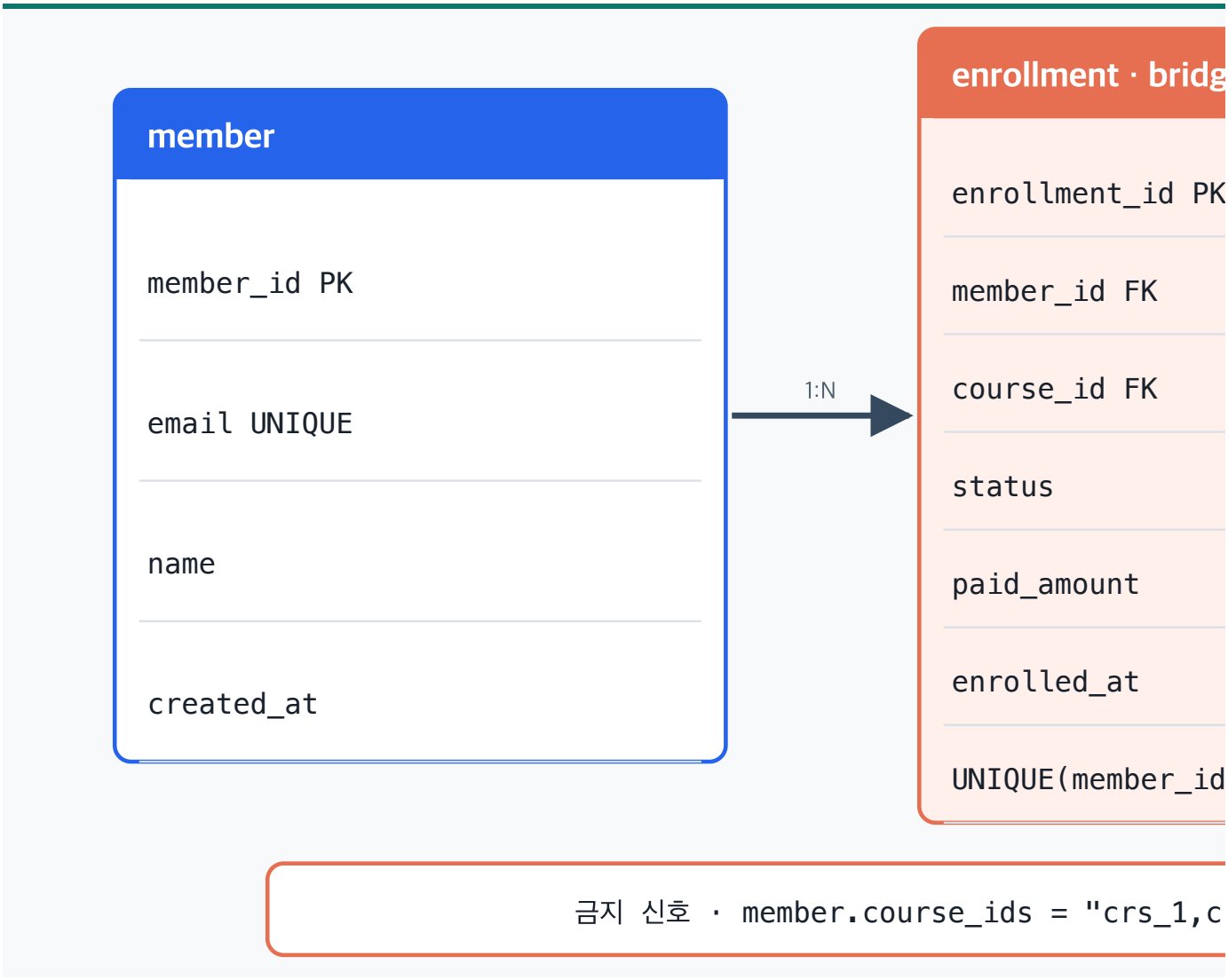
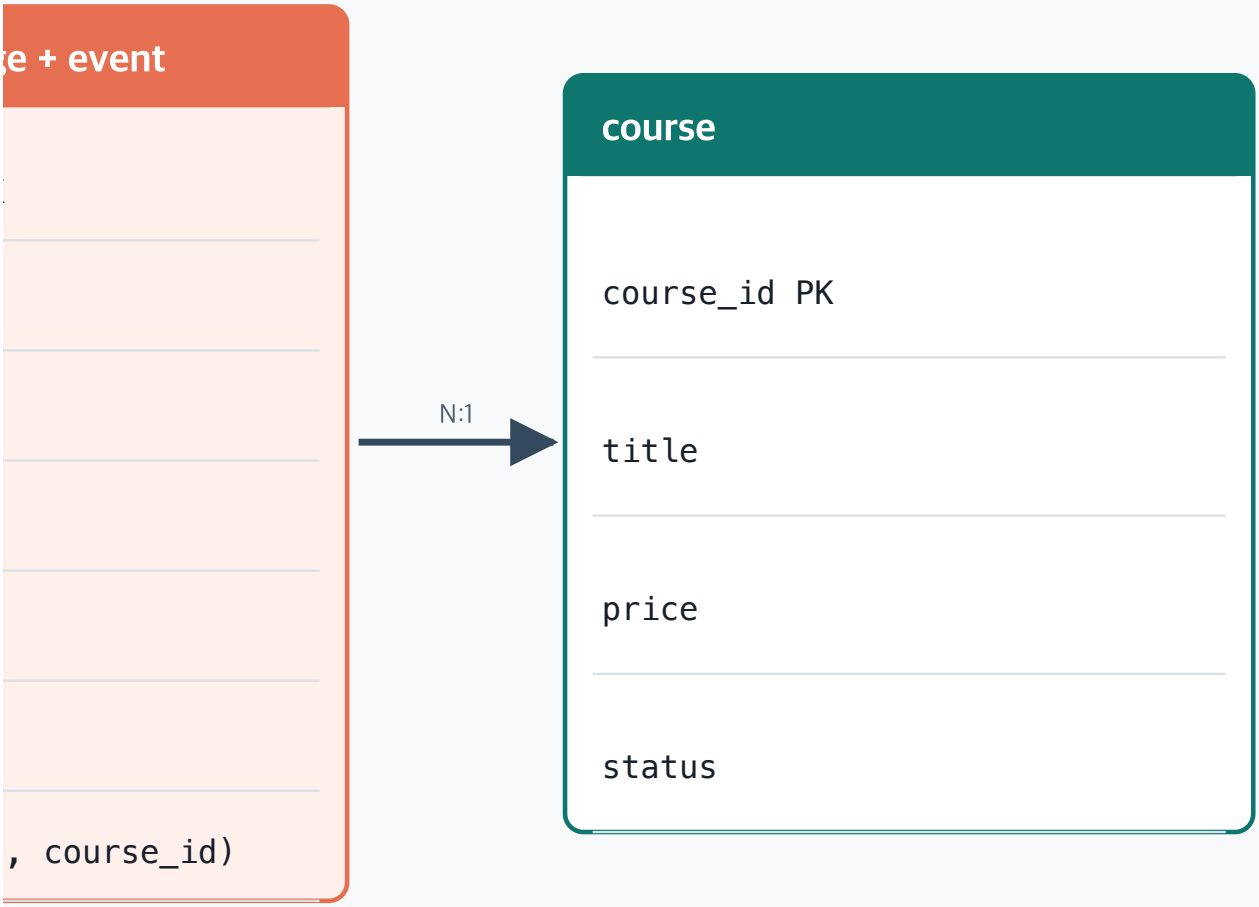


그림 7. 한 회원은 여러 강좌, 한 강좌는 여러 회원을 가집니다. `enrollment`가 두 일대다 관계와 신청 자체의 속성을 맡습니다.


member
member_id PK
email UNIQUE
name
created_at
1:N
enrollment · bridg
enrollment_id PK
member_id FK
course_id FK
status
paid_amount
enrolled_at
UNIQUE(member_id
금지 신호 · member.course_ids = "crs_1,c



rs_7" · course.member_ids = JSON array

8.1. bridge table의 최소 구조

```
CREATE TABLE enrollment (
  enrollment_id bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  member_id bigint NOT NULL REFERENCES member(member_id),
  course_id bigint NOT NULL REFERENCES course(course_id),
  status text NOT NULL,
  paid_amount numeric(12,2) NOT NULL,
  enrolled_at timestamp with time zone NOT NULL,
  UNIQUE (member_id, course_id)
);
```

PostgreSQL constraint 문서도 주문과 상품의 다대다 관계를 order_items 같은 연결 table과 두 FK로 표현하는 예를 제시합니다.

8.2. 연결 table에 붙는 업무 속성

잘못 둔 위치	올바른 질문	후보 위치
member에 `course_status`	어느 강좌에 대한 상태입니까	enrollment.status
course에 `paid_amount`	어느 회원이 실제 낸 금액입니까	enrollment.paid_amount
member에 `enrolled_at`	어느 신청의 시점입니까	enrollment.enrolled_at
course에 `completion_percent`	어느 수강자의 어느 진도입니까	lesson_progress

8.3. CSV·JSON·array가 언제나 나쁜 것은 아닙니다

배열과 JSON은 DB가 지원하는 정식 type일 수 있습니다. 문제는 기술 자체가 아니라 **독립적으로 식별·참조·검색·수정·제약해야 하는 관계를 한 cell에 숨기는 것**입니다.

별도 관계 table 쪽	한 row가 소유한 구조 값 쪽
각 항목에 FK가 필요	외부에서 개별 항목을 참조하지 않음
항목별 중복·삭제·권한 필요	owner와 항상 함께 생성·삭제
항목별 검색·집계가 핵심	payload 보존·표시가 주목적
항목이 자체 lifecycle을 가짐	구조 schema와 validation을 별도 관리

`course_ids = "1,2,3"`은 course 관계를 숨깁니다. 반면 외부 API 원문 payload를 감사 목적으로 JSON에 보관하는 것은 source와 schema version이 명확하다면 다른 선택입니다.

9. data dictionary는 이름보다 의미를 계약합니다

column 목록만 있으면 개발자·기획자·AI가 같은 단어를 다르게 해석합니다. 항목마다 다음을 채웁니다.

항목	질문	예
table·column	일관된 기술 이름입니까	`enrollment.paid_amount`
업무명	현업이 쓰는 말은 무엇입니까	실제 결제 금액
정의	포함·제외 범위는 무엇입니까	할인 후 승인된 강좌 대금
grain·owner	어느 row의 사실입니까	한 수강 신청
type	어떤 연산을 허용합니까	`numeric(12,2)`
format·unit	소수·통화·시간대는 무엇입니까	KRW, scale 2
required	모든 생성 경로에서 알 수 있습니까	NOT NULL
default	생략 때 누가 어떤 값을 넣습니까	없음
domain	허용값과 전이는 무엇입니까	0 이상
source	최초 source of truth는 어디입니까	payment confirmed event
sensitivity	개인정보·민감도는 무엇입니까	일반 거래정보
owner	의미와 품질 책임자는 누구입니까	결제 PO·data owner
example	정상·경계·오류 예시는 무엇입니까	0.00, 42000.00, -1 금지

이름은 문서 전체에서 같아야 합니다

```
화면       결제 금액
API        paidAmount
logical    paid_amount
event      paid_amount
metric     paid_amount_null_rate
```

표기법은 계층마다 달라도 **정의·단위·NULL·source**는 같아야 합니다. 이름이 비슷하다고 같은 데이터로 단정하지 않습니다.

파생값에는 계산식과 source를 씁니다

항목	source	계산식	갱신 시점
completion_percent	lesson_progress	완료 레슨 / 전체 레슨 × 100	진도 event 뒤
order_total	order_item	수량 × 확정 단가 합	item 변경 transaction
member_age	birth_date + 기준일	만 나이 규칙	조회 시 또는 기준일 batch

30초 확인 4

`amount` 하나만 정의하고 통화와 세금 포함 여부를 적지 않으면 어떤 서로 다른 숫자가 같은 column에 들어갈 수 있습니까?

10. NULL·빈 문자열·0·false의 의미를 분리합니다

그림 8. 값이 없거나 거짓인 이유는 서로 다릅니다. 한 column의 NULL이 무엇을 뜻하는지 생성 경로와 조회 규칙까지 씁니다.

NULL

아직 모름

unknown

""

알지만 빈 텍스트

empty

0

측정값이

zero

예 · completion_percent

`0` = 진도를 측정했고 시작하지 않음

`NULL` = 아직 진도 `row`가 없거나 측정 불가 · 어느 뜻인지 계약 필요

“모름”과 “해당 없음”이 업무에 중요하면 reason/status를 별도 column 또는 관계로 모델링

영

false

명시적 거짓

boolean

N/A

적용 대상 아님

separate reason

한다.

10.1. 다섯 상태를 구분합니다

값	가능한 뜻	예
NULL	모름·미수집·미정	아직 환불 결정 안 됨
빈 문자열 `""`	text 값은 있으나 길이 0	사용자가 설명을 비움
0	측정·계산 결과가 영	진도 0%, 금액 0원
false	명시적 거짓	마케팅 동의 안 함
not applicable	이 row에는 질문 자체가 적용 안 됨	무료 강좌의 결제 승인 ID

"모름", "미수집", "해당 없음"이 업무적으로 다르면 `status`나 `reason_code`를 별도 모델링합니다. 모든 차이를 NULL 하나로 압축하지 않습니다.

10.2. NOT NULL 결정 질문

```
1. row가 처음 생길 때 값을 항상 압니까?
2. import·관리자·batch·migration 경로도 압니까?
3. parent 없이 row가 존재할 수 있습니까?
4. 아직 모르는 상태를 별도 status로 표현했습니까?
5. 기존 데이터에 NULL이 있다면 어떻게 backfill합니까?
```

10.3. CHECK만으로 NULL을 막는다고 오해하지 않습니다

PostgreSQL에서 CHECK 식은 true 또는 NULL이면 통과할 수 있습니다. `CHECK (price > 0)`만으로 NULL을 막지 못하므로 값이 필수라면 `NOT NULL`을 별도 선언합니다.

```
paid_amount numeric(12,2) NOT NULL CHECK (paid_amount >= 0)
```

10.4. 0을 NULL 대신 억지로 넣지 않습니다

"아직 측정하지 않음"을 0으로 저장하면 실제 0과 구분할 수 없습니다. 반대로 모든 0을 NULL로 바꾸면 무료·진도 0·재고 0 같은 실제 사실을 잃습니다.

11. type·unit·시간을 업무 의미와 함께 고릅니다

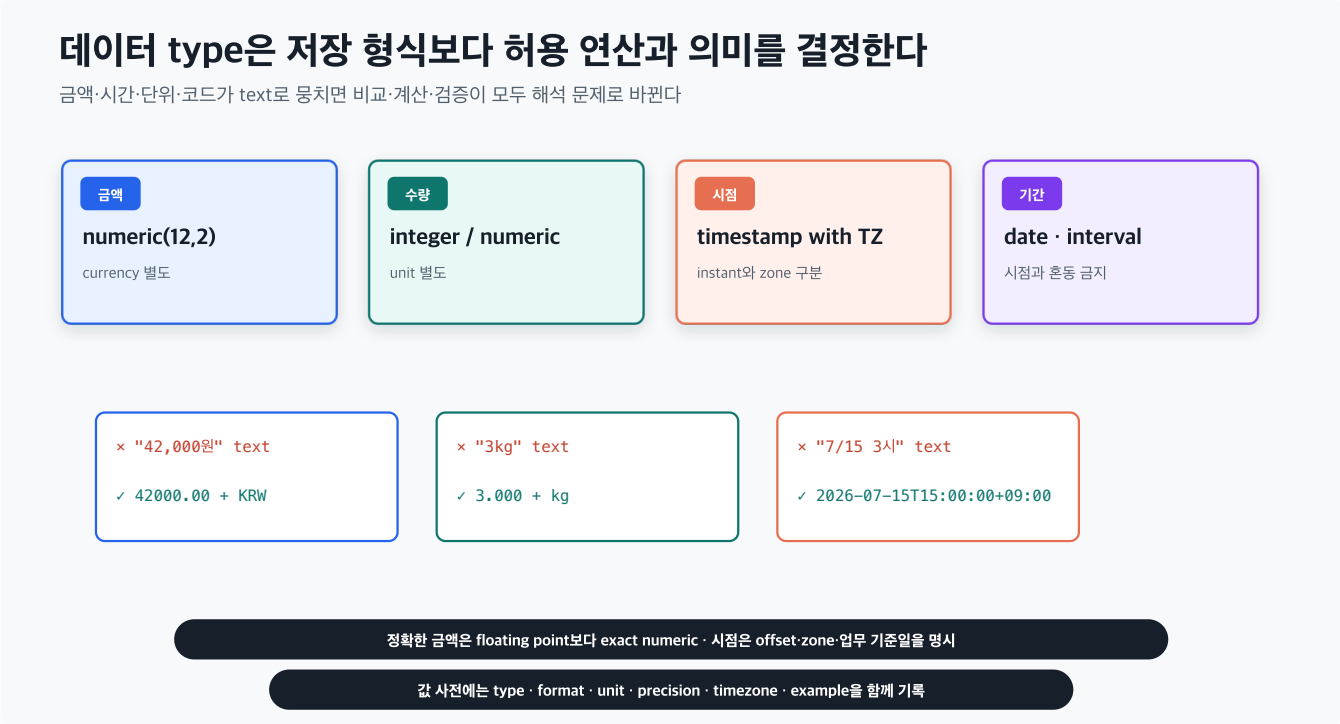


그림 9. type은 저장 모양이 아니라 허용 연산과 정확성을 정합니다. 숫자·시간·단위를 text 한 칸에 뭉치지 않습니다.

금액

numeric(12,2)

currency 별도

수량

integer / numeric

unit 별도

× "42,000원" text

✓ 42000.00 + KRW

× "3kg" text

✓ 3.000 + kg

정확한 금액은 floating point보다 exact num

시점

timestamp with TZ

instant와 zone 구분

기간

date · interval

시점과 혼동 금지

× "7/15 3시" text

✓ 2026-07-15T15:00:00+09:00

eric · 시점은 offset·zone·업무 기준일을 명시

11.1. 대표 type 선택표

업무 의미	후보 type	함께 정할 것
개수·순번	integer·bigint	음수 허용·최대 범위
정확한 금액·비율	numeric·decimal	precision·scale·currency
측정 근삿값	real·double	오차 허용·비교 방식
참·거짓	boolean	NULL의 제3상태 허용 여부
짧은 코드·이름	text·varchar	정규화·대소문자·길이
날짜	date	업무 기준 지역
절대 시점	timestamp with time zone	입력 offset·표시 zone
지역 벽시계 시간	timestamp without time zone	어떤 지역 규칙인지 별도 보존
기간	interval 또는 시작·종료	경계 포함·월 계산 규칙
외부 payload	JSON type	schema version·검색·보존 목적

11.2. 금액은 exact인지 먼저 묻습니다

PostgreSQL numeric type 문서는 `numeric`을 exact type으로, `real`·`double precision`을 inexact floating-point type으로 설명하며 금액처럼 정확성이 필요한 값에는 numeric을 권합니다.

```
좋음  paid_amount = 42000.00, currency = KRW
주의  paid_amount = 42000.0 float
나쁨  paid_amount = "42,000원"
```

통화별 소수 자리와 반올림 규칙은 별도 계약입니다. 무조건 scale 2가 세계 모든 통화에 맞는다는 뜻은 아닙니다.

11.3. 식별 번호는 숫자처럼 보여도 text일 수 있습니다

전화번호·우편번호·상품 코드·주민 식별 문자열은 덧셈 대상이 아닙니다. 앞자리 0, 하이픈, 국가 코드, check digit을 보존해야 하면 text와 format validation이 더 적합할 수 있습니다.

11.4. 시점·날짜·지역 시간을 구분합니다

PostgreSQL 문서는 timestamp with time zone 값을 내부적으로 UTC 기준으로 다루고 session timezone에 맞춰 표시한다고 설명합니다. 원래 사용자가 고른 시간대 이름까지 자동 보존되는 것은 아닙니다.

업무	저장 후보	이유
결제 승인 순간	absolute timestamp	세계 어디서나 같은 순간
서울 강의 7월 20일 14시	local date/time + zone ID	미래 DST·정책 변화와 일정 의미
생일	date	시각이 필요 없음
30분 제한	duration·interval	시점이 아니라 길이

RFC 3339는 인터넷 timestamp 표현 형식을 정의합니다. API와 event에서 `2026-07-15T15:00:00+09:00` 처럼 날짜·시간·offset을 명시하되, DB 내부 type과 업무상 zone 보존 여부를 따로 결정합니다.

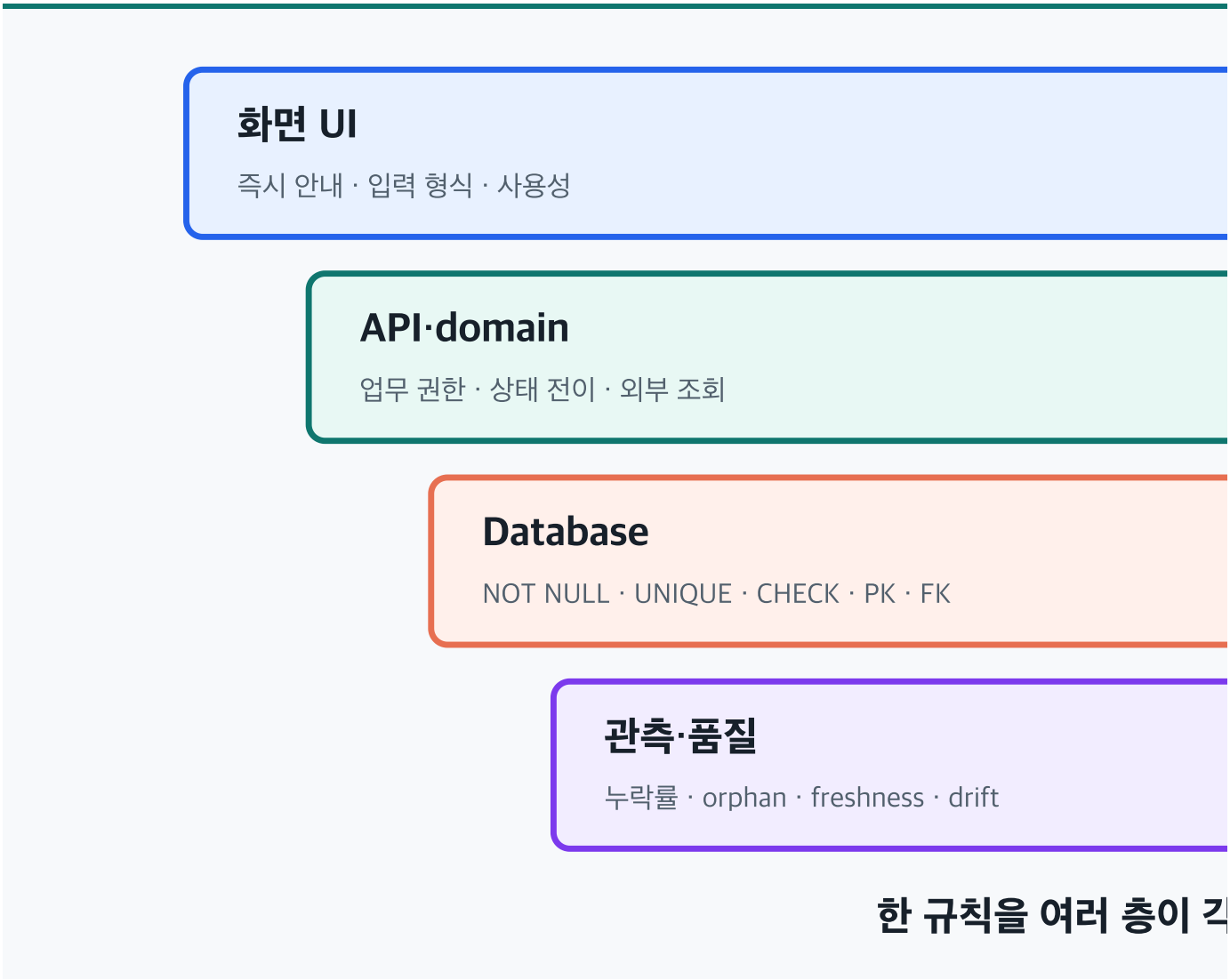
11.5. 허용 상태는 작은 집합인지 관리 대상인지 봅니다

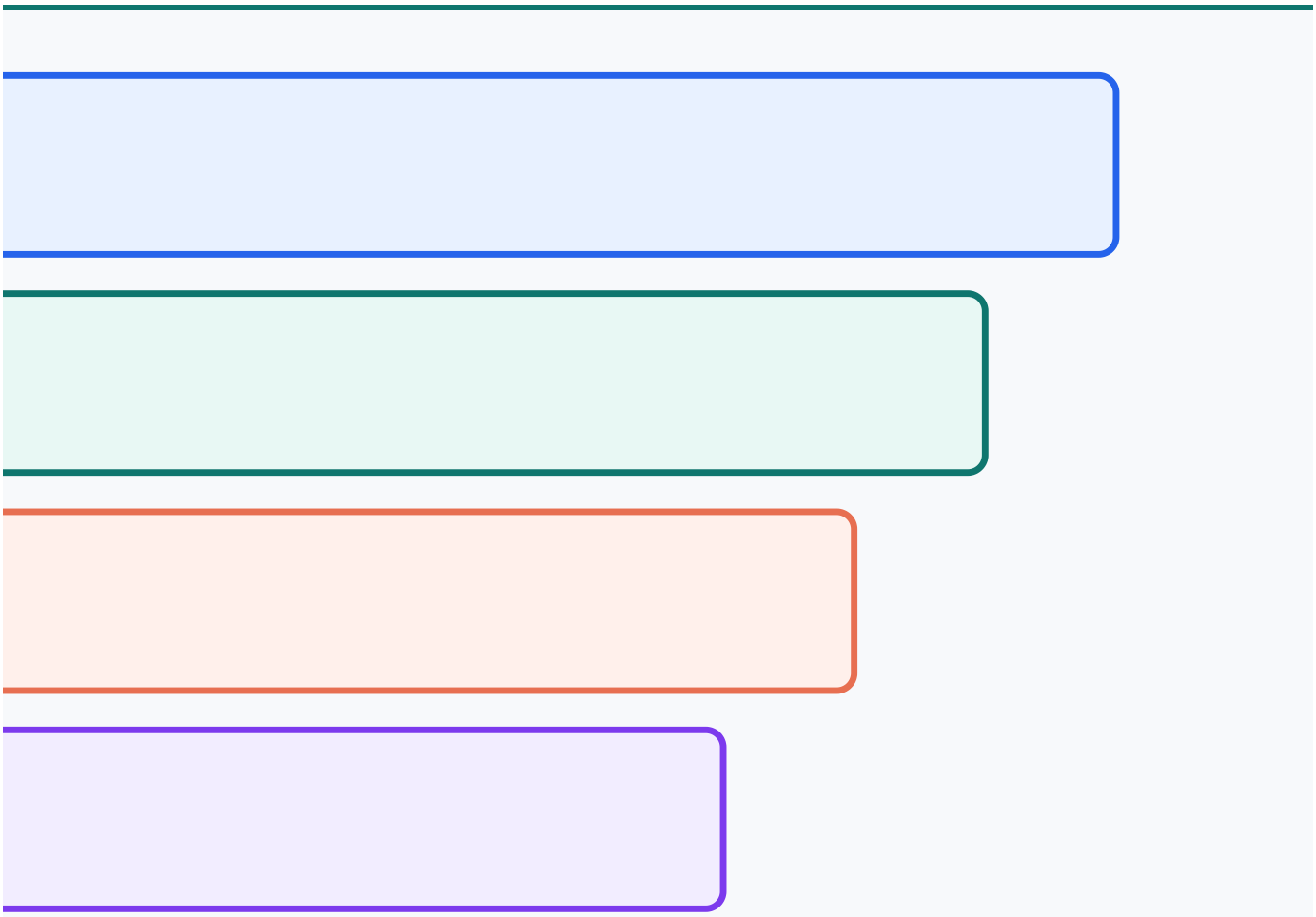
선택	적합한 경우	주의
CHECK	작고 안정적인 허용값	변경 migration 필요
enum type	DB 제품 안의 강한 type	값 제거·변경·이식성 검토
code table + FK	운영자가 설명·순서·활성 여부 관리	join과 lifecycle 필요
자유 text	정말 자유 서술	상태·분류에는 부적합

PostgreSQL enum 문서도 요일이나 상태처럼 정적인 값 집합을 예로 듭니다. 변경이 잦고 metadata가 붙는 상태라면 code table이 더 적합할 수 있습니다.

12. 제약은 잘못된 상태를 database가 거부하게 합니다

그림 10. 같은 규칙을 UI는 친절하게 안내하고, API는 업무 맥락을 확인하며, DB는 모든 쓰기 경로의 마지막 무결성을 지킵니다.


화면 UI
즉시 안내 · 입력 형식 · 사용성
API·domain
업무 권한 · 상태 전이 · 외부 조회
Database
NOT NULL · UNIQUE · CHECK · PK · FK
관측·품질
누락률 · orphan · freshness · drift
한 규칙을 여러 층이 ㄱ



자의 역할로 방어한다

12.1. 핵심 constraint 지도

constraint	지키는 질문	수강 사례
PRIMARY KEY	이 row는 누구입니까	enrollment_id
NOT NULL	row 생성 때 반드시 압니까	member_id, course_id
UNIQUE	같은 업무 값이 둘 생겨도 됩니까	member_id + course_id
FOREIGN KEY	parent가 실제 존재합니까	course_id → course
CHECK	이 row 안의 값·조합이 유효합니까	amount ≥ 0
DEFAULT	값 생략 때 새 row에 무엇을 넣습니까	status = pending
GENERATED	다른 column으로 항상 계산합니까	line_total = qty × unit_price

12.2. 화면 검증만으로 부족합니다

동시에 두 요청이 들어오거나 batch·migration·관리 도구가 직접 쓰면 화면의 중복 확인을 건너뜁니다. 업무 중복을 절대 허용하지 않는다면 DB UNIQUE가 경쟁 조건의 마지막 방어가 됩니다.

```
CONSTRAINT uq_enrollment_member_course UNIQUE (member_id, course_id)
```

12.3. database constraint만으로도 충분하지 않습니다

DB error를 그대로 사용자에게 보여 주면 어떤 값을 고쳐야 하는지 알기 어렵습니다. UI·API는 미리 설명하고, DB error code·constraint name을 안정적인 problem response로 변환합니다.

층	책임
UI	입력 직후 형식·필수·범위를 친절히 안내
API·domain	권한·상태 전이·외부 사실·교차 aggregate 검증
DB	row·relation의 불변 무결성 거부
관측	constraint error·누락률·orphan·drift 추세 감시

12.4. constraint 이름도 운영 증거입니다

```
CONSTRAINT ck_enrollment_paid_amount_nonnegative CHECK (paid_amount >= 0)
CONSTRAINT fk_enrollment_member FOREIGN KEY (member_id) REFERENCES member(member_id)
```

명시적 이름은 migration·오류 mapping·운영 로그에서 어떤 규칙이 깨졌는지 찾기 쉽게 합니다.

12.5. CHECK의 범위를 압니다

PostgreSQL은 다른 row나 다른 table을 참조하는 CHECK를 지속적인 무결성 보장으로 지원하지 않습니다. cross-row 규칙은 가능하면 UNIQUE·FK·EXCLUDE 같은 적합한 constraint를 쓰고, 그 밖의 aggregate·상태 규칙은 transaction·trigger·domain service·quality check 중 책임 위치를 명시합니다. DB 제품마다 기능은 다르므로 실제 구현 전 공식 문서를 확인합니다.

30초 확인 5
API에서 “이미 신청했는지” 조회한 뒤 INSERT하는 코드만으로 동시 중복 신청을 완전히 막기 어려운 이유는 무엇입니까?

13. referential action은 parent와 child의 수명 계약입니다

FK는 parent가 존재하는지만 정하지 않습니다. parent key를 수정·삭제할 때 child를 어떻게 할지도 정합니다.

action	parent 삭제 시	적합성 질문
RESTRICT	참조 중이면 거부	child가 있는 parent 삭제를 업무상 막아야 합니까
NO ACTION	constraint 확인 시점에 위반이면 거부	transaction 끝까지 재배치가 필요합니까
CASCADE	child도 함께 삭제	child가 parent의 순수 구성요소이고 독립 보존 가치가 없습니까
SET NULL	child FK를 NULL로 변경	parent 없는 child가 의미 있고 FK가 optional입니까
SET DEFAULT	기본 FK 값으로 변경	의미 있는 기본 parent가 정말 존재합니까

PostgreSQL constraint 문서는 `ON DELETE`와 `ON UPDATE` action을 설명합니다. 실제 `RESTRICT`와 `NO ACTION`의 검사 시점 등 세부 의미는 DB 제품과 constraint 설정을 확인합니다.

13.1. CASCADE를 편의 기능으로 고르지 않습니다

```
course 삭제
├─ lesson          강좌의 순수 구성 → cascade 후보
├─ enrollment      거래·수강 이력 → restrict 또는 별도 비활성화 검토
├─ payment         재무 증거 → 보존 정책과 법적 요구 검토
└─ audit event     조사 증거 → 일반적으로 parent 편의 삭제와 분리
```

parent와 함께 child가 사라져도 되는지는 기술 team이 아니라 업무 owner도 승인해야 합니다.

13.2. “삭제”의 네 의미를 구분합니다

사용자 말	실제 후보
목록에서 숨김	status = inactive, view filter
판매 중지	course 판매 상태 변경
개인 탈퇴	식별정보 분리·가명화·파기 workflow
잘못 만든 임시 row 제거	physical delete 가능성

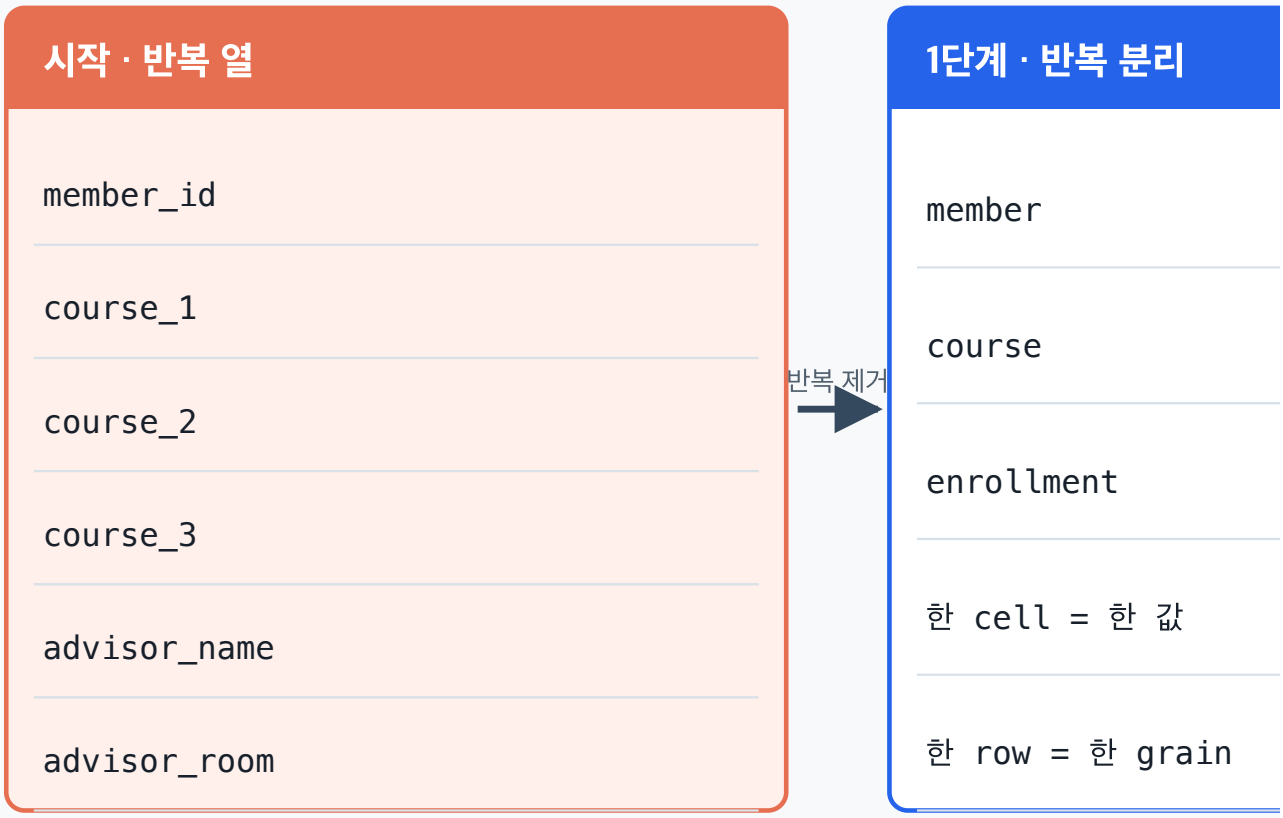
soft delete는 모든 삭제 문제의 해답이 아닙니다. `deleted_at`을 추가해도 unique·FK·검색·보존·파기·복구 규칙은 남습니다. 데이터 보존·백업·파기는 M05-03에서 별도로 설계합니다.

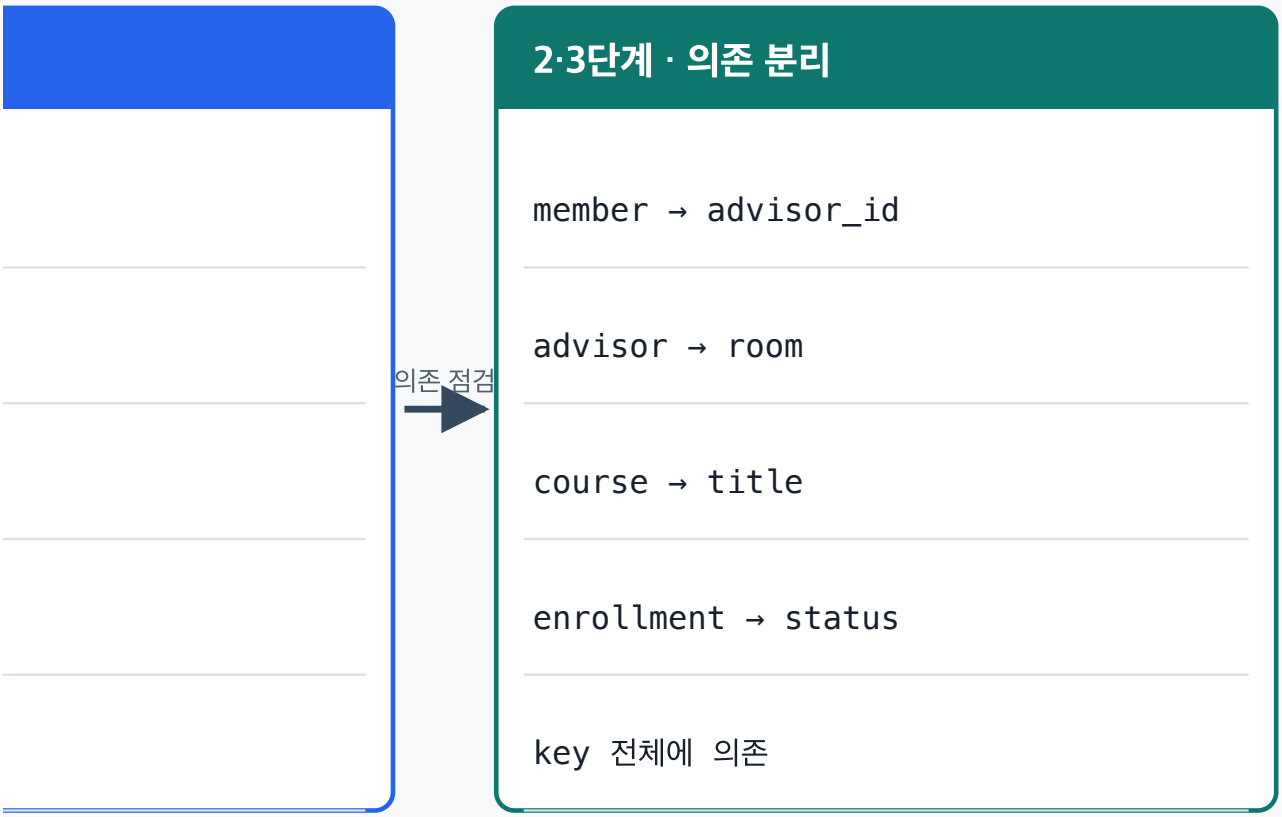
13.3. delete scenario를 문장으로 검수합니다

```
Given  완료된 enrollment와 payment가 있다.
When   운영자가 course 삭제를 요청한다.
Then   course 삭제는 거부되고 판매 상태만 retired가 된다.
And    기존 enrollment·payment·audit는 조회 가능하다.
And    learner 화면에서는 과거 학습 기록으로 표시된다.
```

14. 정규화는 변경 이상을 줄이는 질문입니다

그림 11. 정규화는 표 개수를 늘리는 의식이 아니라 한 사실이 여러 곳에서 서로 다르게 바뀌는 위험을 줄이는 과정입니다.


시작 · 반복 열
member_id
course_1
course_2
course_3
advisor_name
advisor_room
반복 제거
1단계 · 반복 분리
member
course
enrollment
한 cell = 한 값
한 row = 한 grain
추가 이상 · course_4마다 열 변경
중복 감소 · 관



계가 row가 됨

수정 이상 감소 · 책임 위치 명확

14.1. 세 가지 변경 이상

이상	수강 사례	결과
insert anomaly	회원이 없으면 강좌를 등록할 수 없음	독립 사실을 못 만듦
update anomaly	강좌명이 수강 row마다 반복됨	일부만 바뀌어 불일치
delete anomaly	마지막 수강 row 삭제 때 강좌 정보도 사라짐	다른 사실까지 손실

Microsoft의 database normalization 초급 문서는 중복과 inconsistent dependency를 줄이기 위해 table과 관계를 조직한다고 설명합니다.

14.2. 초급용 1·2·3단계 질문

아래는 formal 정의를 대신하는 완전한 이론이 아니라, 초보자가 반복과 의존을 찾기 위한 학습용 질문입니다.

단계	학습 질문	나쁜 예	개선
1NF 관점	반복 열·목록 cell이 있습니까	course_1·course_2	enrollment row로 분리
2NF 관점	복합 key의 일부에만 의존합니까	enrollment에 member_name	member로 이동
3NF 관점	non-key가 다른 non-key에 의존합니까	member에 advisor_room	advisor로 이동

14.3. 함수 종속을 쉬운 문장으로 읽습니다

```
member_id        → member_name, email
course_id        → course_title, list_price
enrollment_id    → member_id, course_id, status, paid_amount
(member, course) → 한 번만 신청한다는 업무 uniqueness
```

`course_title`은 `enrollment_id`를 통해 간접적으로 알 수 있어도 course가 owner입니다. enrollment에 복제하면 어느 값이 source of truth인지 계약해야 합니다.

14.4. 정규화가 무조건적인 끝은 아닙니다

읽기 성능·분석·검색·snapshot·외부 전송을 위해 일부 값을 복제할 수 있습니다. 다만 “편해서”가 아니라 측정된 필요와 동기화 계약이 있어야 합니다.

복제 결정에 필요한 것	예
canonical source	`course.title`
복제 목적	구매 당시 강좌명 snapshot
갱신 규칙	과거 주문에는 새 제목을 반영하지 않음
drift metric	source와 같아야 하는 복제값 불일치 0
rebuild 방법	event replay 또는 batch backfill

15. default·generated·derived 값을 구분합니다

15.1. default는 생략된 새 값에 적용됩니다

PostgreSQL 문서에서 default는 새 row를 만들 때 값을 지정하지 않은 column에 채워집니다. 기존 row를 바꾸거나 다른 column 변화에 따라 자동 재계산하는 규칙이 아닙니다.

```
status text NOT NULL DEFAULT 'pending'
```

default가 업무상 “항상 pending으로 시작”을 뜻하는지, 단지 client 편의인지 문서화합니다. 잘못된 default는 누락을 숨길 수 있습니다.

15.2. generated column은 현재 row에서 계산됩니다

PostgreSQL generated column 문서는 default와 달리 row가 바뀔 때 생성식에 따라 값이 갱신되고 사용자가 override할 수 없다고 설명합니다. 제품별 제약이 있으므로 공식 문서를 확인합니다.

```
line_total numeric(14,2)
  GENERATED ALWAYS AS (quantity * unit_price) STORED
```

15.3. 저장할지 계산할지 결정합니다

질문	계산 쪽	저장 쪽
값이 현재 source로 항상 재현됩니까	유리	snapshot 필요성 낮음

질문	계산 쪽	저장 쪽
계산 비용이 큽니까	불리	유리
과거 당시 값을 보존해야 합니까	불리	유리
source가 바뀌면 과거도 바뀌어야 합니까	유리	동기화 필요
여러 system이 같은 계산식을 공유합니까	중앙 계산 필요	event·version 필요

`order_total`을 client가 보내고 server가 그대로 믿으면 item과 불일치할 수 있습니다. canonical source와 계산 책임자를 정합니다.

15.4. 파생값 계약서

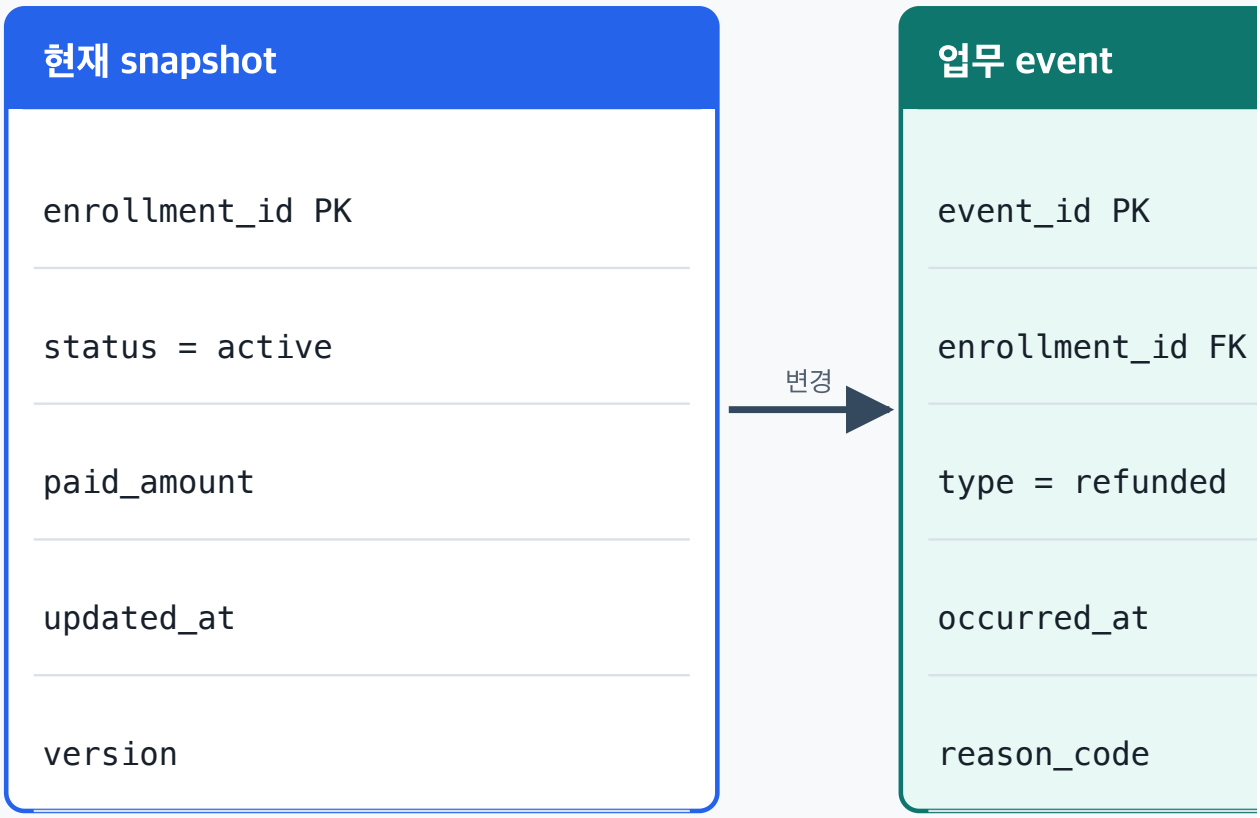
```
name        completion_percent
source      lesson_progress.status, course.required_lesson_count
formula     completed_required / required_total × 100
rounding    소수 첫째 자리 반올림
null        required_total=0이면 NULL + reason=no_required_lesson
refresh     progress event commit 후
owner       learning domain
drift SLO   canonical 재계산과 차이 0
```

30초 확인 6

`created_at DEFAULT now()`와 `age GENERATED ...`는 언제 계산되는지가 어떻게 다릅니까?

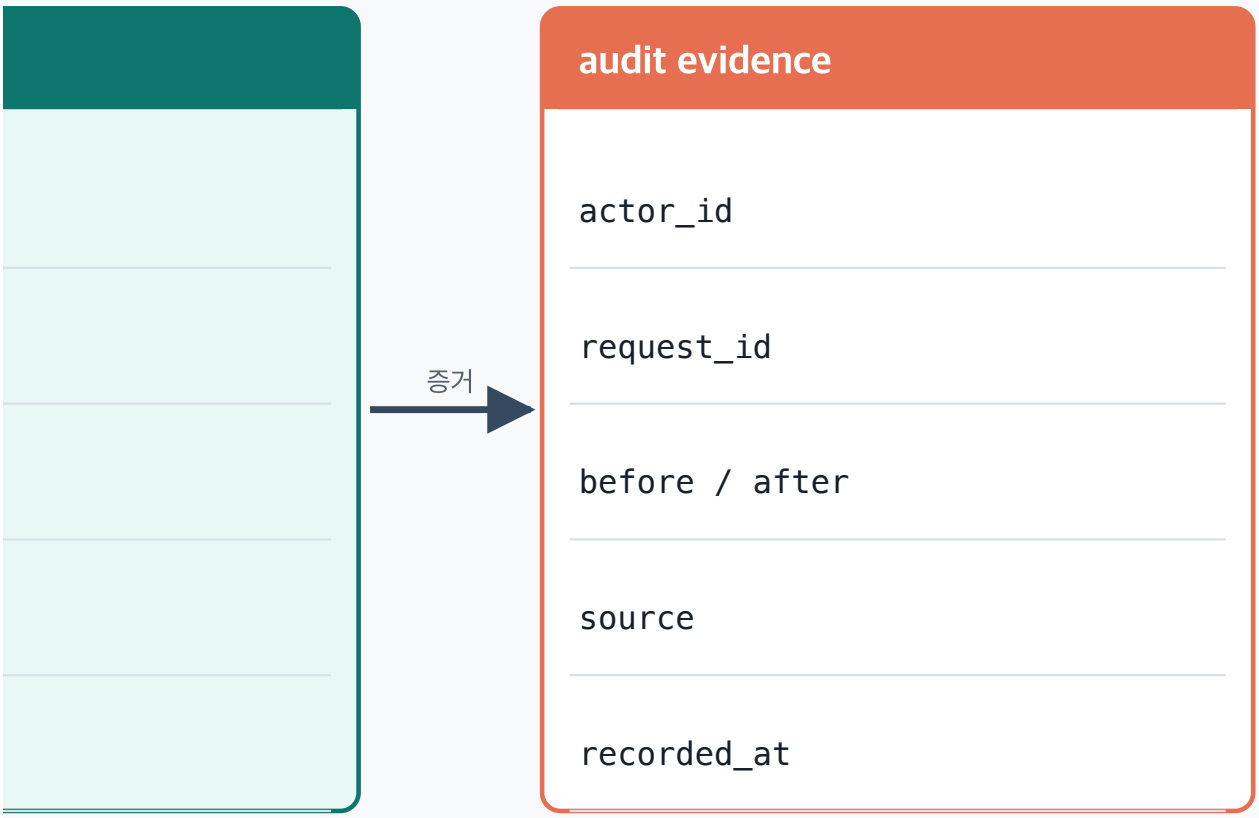
16. 현재 상태·업무 event·감사 증거를 분리합니다

그림 12. 현재 조회용 snapshot, 업무 의미를 가진 event, 책임 추적용 audit는 서로 보완하지만 같은 table·column이라고 단정할 수 없습니다.



현재 조회에 빠름

업무 의미와



순서를 남김

책임·추적·조사에 사용

16.1. `updated_at` 하나로 알 수 없는 것

```
누가 바꿨는가?
사용자 요청인가 batch인가?
무엇에서 무엇으로 바뀌었는가?
왜 바꿨는가?
어떤 request·job·event가 원인인가?
실패 뒤 보상으로 바꾼 것인가?
```

16.2. 세 저장 목적

목적	대표 질문	필드 예
current snapshot	지금 상태가 무엇입니까	status, version, updated_at
business event	어떤 업무 사건이 일어났습니까	type, occurred_at, reason_code
audit evidence	누가 어떤 경로로 무엇을 바꿨습니까	actor, request_id, before/after, source

모든 서비스가 event sourcing을 해야 한다는 뜻은 아닙니다. 복원·설명·규제·분쟁·운영 요구에 맞춰 필요한 수준을 선택합니다.

16.3. 상태 전이와 증거를 연결합니다

from	command	to	조건	event	audit
pending	activate	active	payment confirmed	enrollment_activated	actor·payment_id
active	complete	completed	required lessons done	enrollment_completed	progress summary
active	cancel	canceled	cancel policy allows	enrollment_canceled	reason·actor
completed	cancel	거부	terminal rule	command_rejected	policy·request_id

16.4. 동시 수정에는 version을 고려합니다

두 사용자가 같은 row를 읽고 각각 수정하면 나중 저장이 먼저 저장을 덮을 수 있습니다. `version` · `updated_at` 조건을 사용한 optimistic concurrency는 충돌을 발견하는 한 방법입니다. 정확한 구현은 M05-02의 transaction·update와 연결합니다.

16.5. 감사 log에는 민감정보를 무조건 복제하지 않습니다

before/after 전체 payload는 비밀번호·token·주민 식별정보·민감 메모를 과다 저장할 수 있습니다. 어떤 field를 mask·hash·제외할지, 누가 얼마 동안 볼지 M05-03과 G10 개인정보 과정에서 검토합니다.

17. schema 변경은 기존 데이터와 consumer를 함께 이동시킵니다

모델은 한 번 그리고 끝나는 그림이 아닙니다. 실제 row와 API·event·report가 존재한 뒤에는 column 하나도 호환성 작업입니다.

17.1. 안전한 변경의 학습 순서

```
1 새 구조를 추가한다.
2 새·옛 consumer가 함께 읽을 수 있게 한다.
3 기존 row를 backfill한다.
4 quality query로 누락·불일치를 검증한다.
5 새 쓰기 경로로 전환한다.
6 NOT NULL·FK·UNIQUE를 검증·강화한다.
7 관찰 기간 뒤 옛 구조를 제거한다.
```

17.2. 예 · `course_ids CSV`를 enrollment로 분리

단계	데이터 작업	검증
inventory	CSV 형식·중복·없는 course 조사	invalid token count
create	enrollment table·PK·FK 후보 생성	DDL review
backfill	token별 row 생성	source count와 target distinct count
dual read	새 관계 우선, 옛 값 fallback	result diff
dual write 또는 freeze	migration 창의 새 변경 처리	lag=0
constrain	orphan 정리 후 FK·UNIQUE	violation=0
retire	옛 column·code 제거	consumer inventory=0

17.3. constraint 추가 전 기존 row를 확인합니다

PostgreSQL ALTER TABLE 문서는 constraint를 추가하면 기존 데이터가 조건을 만족해야 함을 설명합니다. 제품에 따라 online validation·lock·deferrable 기능이 다르므로 데이터 양과 운영 시간에 맞춰 계획합니다.

17.4. rename은 문자열 변경 이상의 일입니다

API field, event schema, BI query, export, 문서, 권한 정책, metric, AI prompt가 옛 이름을 참조할 수 있습니다. consumer inventory와 deprecation 기간을 둡니다.

18. 데이터 품질은 dimension·metric·threshold로 운영합니다

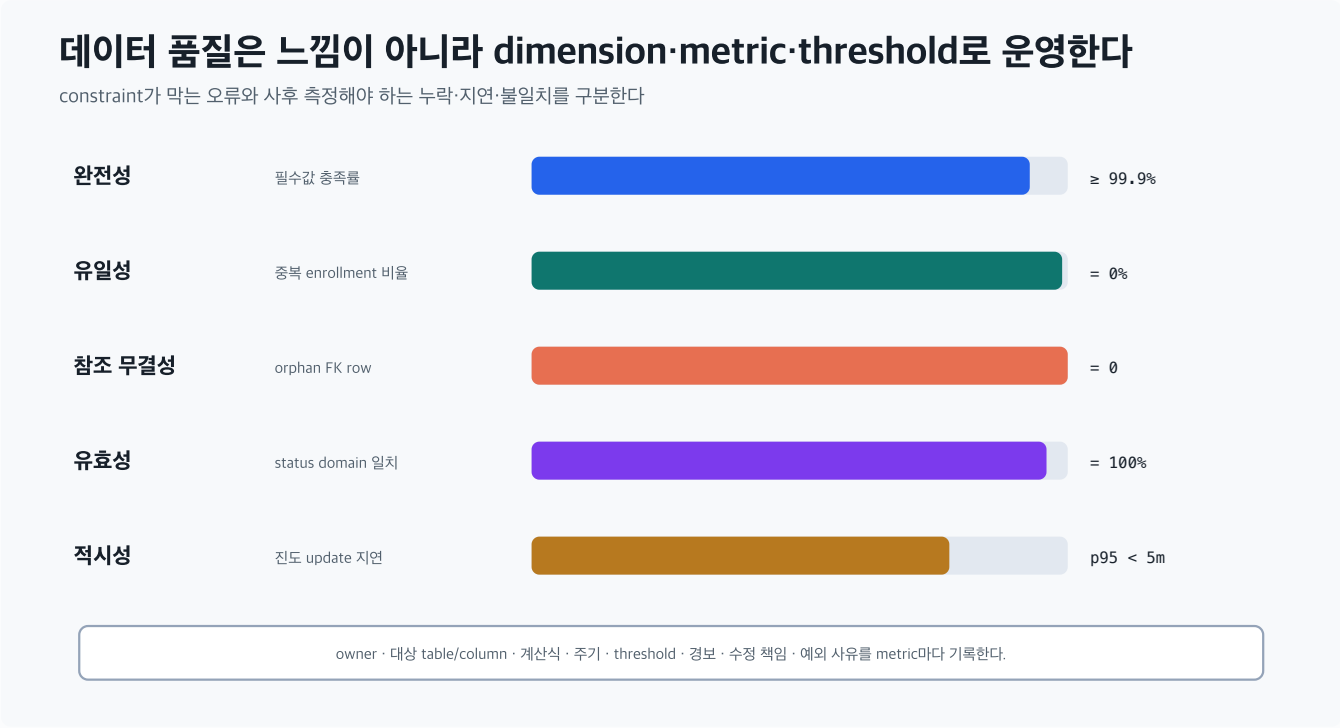
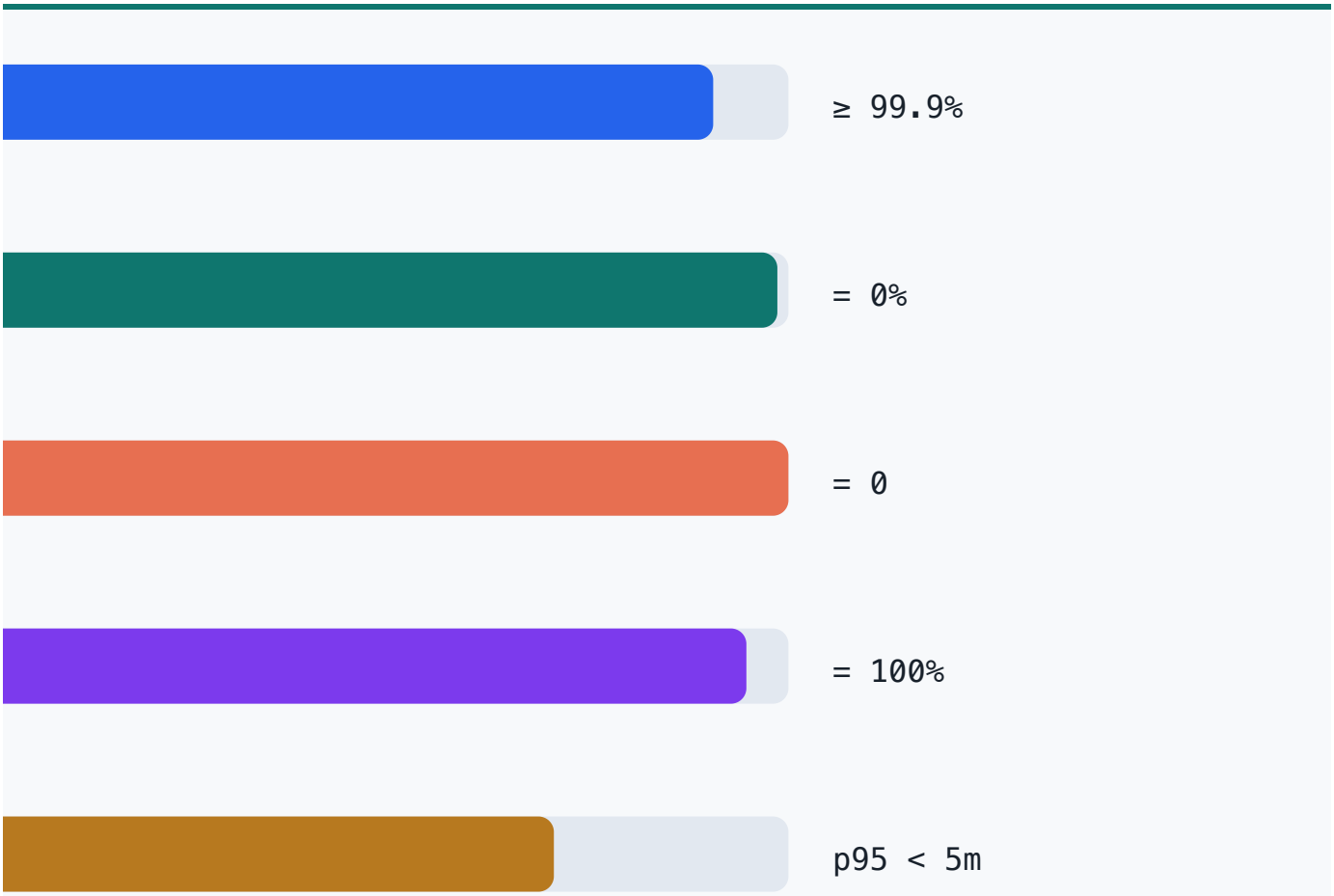


그림 13. “데이터가 깨끗하다”는 말 대신 무엇을 어떻게 재고 어느 값에서 누가 행동할지 씁니다.

완전성	필수값 충족률	
유일성	중복 enrollment 비율	
참조 무결성	orphan FK row	
유효성	status domain 일치	
적시성	진도 update 지연	

owner · 대상 table/column · 계산식 · 주기 · threshold



d · 경보 · 수정 책임 · 예외 사유를 metric마다 기록한다.

W3C Data Quality Vocabulary는 dataset 품질을 목적 적합성 관점에서 판단할 수 있도록 dimension·metric·measurement·policy를 표현하는 틀을 제시합니다. 이 문서는 2016년 Working Group Note이며 W3C Recommendation이라고 부르지 않습니다.

18.1. 초급 품질 지도

dimension	metric 예	계산	threshold	owner 행동
완전성	필수값 충족률	non-null / 대상 row	≥ 99.9%	누락 source 차단
유일성	중복 신청 수	중복 key group count	0	merge·UNIQUE 추가
참조 무결성	orphan row	parent 없는 FK row	0	backfill·FK 강화
유효성	허용 status 비율	domain 일치 / 전체	100%	invalid code 수정
일관성	source·복제값 차이	mismatched rows	0	resync·formula 점검
적시성	진도 갱신 지연	event→snapshot lag	p95 < 5분	worker·queue 조사

18.2. constraint와 metric은 역할이 다릅니다

constraint가 즉시 막기 좋은 것	metric으로 계속 볼 것
NULL 금지	source별 NULL 시도 추세
row 내부 범위	외부 사실과의 정확성
key 중복	dedupe 전 retry 폭주
FK orphan	event와 snapshot lag
허용 domain	운영상 오래 멈춘 상태

18.3. 품질 metric 계약

```
metric_id     dq_enrollment_duplicate
purpose       한 회원의 같은 강좌 중복 신청 감시
scope         status IN (pending, active, completed)
formula       GROUP BY member_id, course_id HAVING COUNT(*) > 1
schedule      10분마다
threshold     0
owner         enrollment domain owner
alert         1 이상이면 incident + 신규 쓰기 경로 확인
repair        canonical row 선택·관계 재연결·중복 row 폐기
evidence      run_id, query_version, count, sample_ids
```

18.4. 품질은 목적에 따라 달라집니다

모든 NULL이 나쁜 것이 아니고 모든 최신 데이터가 정확한 것도 아닙니다. 품질 dimension은 consumer의 사용 목적과 함께 정의합니다. 예를 들어 재무 정산은 정확성·완전성이, 실시간 추천은 적시성이 더 높은 우선순위일 수 있습니다.

30초 확인 7

FK constraint가 있는데도 “진도 데이터가 30분 늦게 반영됨” 문제를 막지 못하는 이유는 무엇입니까?

19. 실습 · 데이터 계약 8-gate를 판정합니다

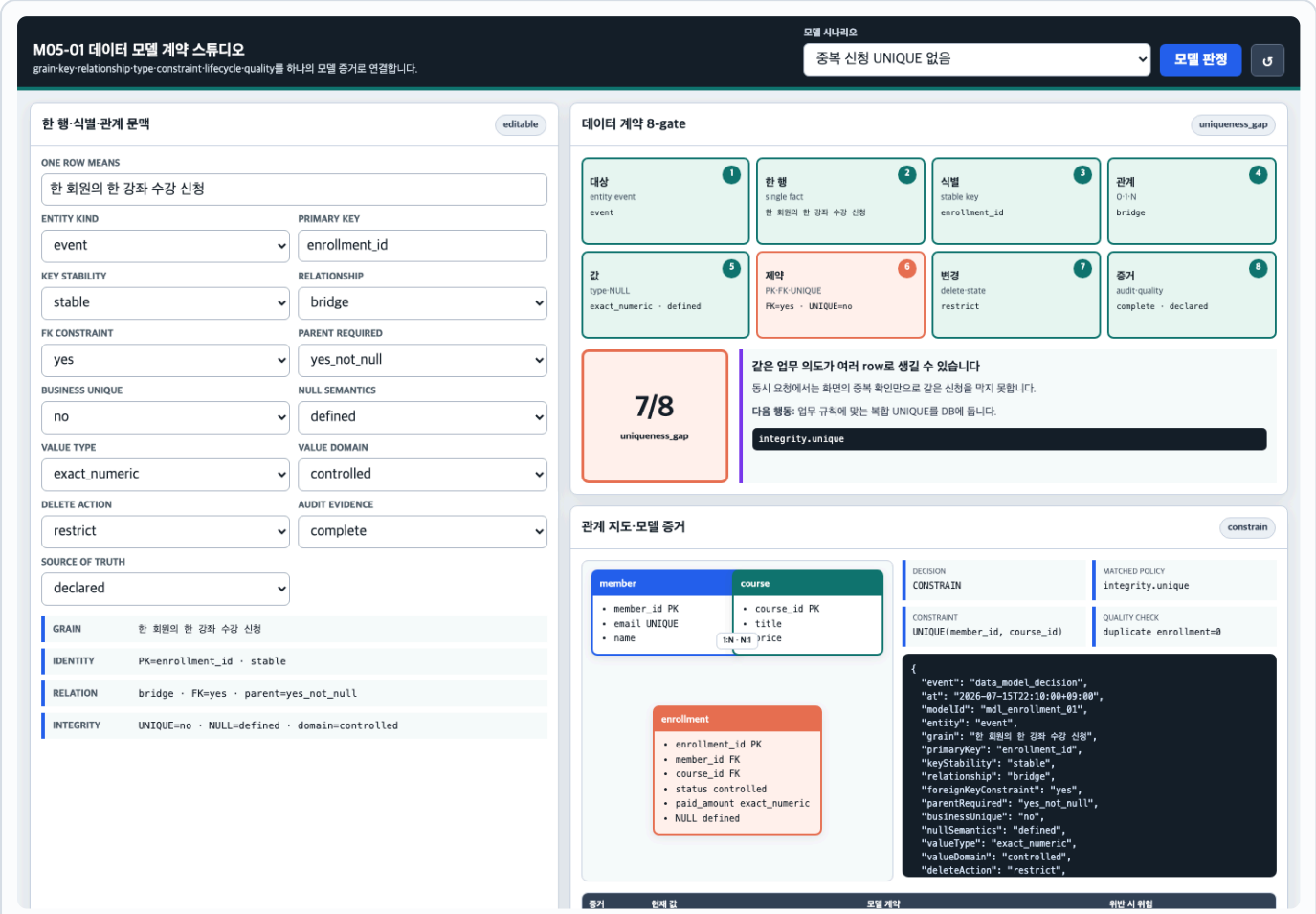


그림 14. `business UNIQUE=no` 한 값 때문에 제약 gate가 실패하고, 복합 UNIQUE와 중복 품질 지표가 다음 행동으로 연결됩니다.

19.1. 실습 목표

다음 네 결과를 말과 문서로 설명합니다.

1. 왜 이 table의 grain이 하나인지
2. PK와 업무 UNIQUE가 왜 별도인지
3. 관계·NULL·type·삭제 action이 어떤 constraint로 이어지는지
4. constraint 뒤에도 어떤 품질 metric과 audit가 필요한지

19.2. 실행 순서

1. 데이터 모델 계약 스튜디오를 엽니다.
2. 일관된 수강 모델을 판정해 8/8 기준선을 봅니다.

3. 14개 오류 시나리오를 하나씩 판정합니다.
4. 같은 시나리오에서 문제 field 하나만 고쳐 다시 판정합니다.
5. matched policy와 constraint·quality check를 실습 기록에 옮깁니다.
6. 내 서비스 table 하나를 같은 8-gate로 검수합니다.

19.3. 반드시 비교할 다섯 쌍

A	B	관찰할 변화
stable PK	mutable email PK	identity policy
FK constraint yes	no	orphan risk
business UNIQUE yes	no	concurrent duplicate
exact numeric	floating point	money precision
restrict	unreviewed cascade	history lifetime

19.4. 실습 기록

scenario	깨진 gate	outcome	matched policy	수정할 계약	재판정

실습기는 실제 DB engine이나 migration tool이 아니라 학습용 논리 모델 판정기입니다. 실제 schema는 제품·version·volume·transaction·security·retention 조건을 추가 검토합니다.

20. 기발자의 데이터 모델 의사결정표

20.1. 회의에서 반드시 묻는 질문

주제	질문	나쁜 답	제출 증거
grain	한 행이 무엇입니까	화면 한 줄입니다	`한 행 =` 문장·row 예

주제	질문	나쁜 답	제출 증거
identity	무엇이 row의 생명입니까	이름이면 되죠	PK·변경 가능성 표
uniqueness	어떤 조합이 중복이면 안 됩니까	API에서 확인합니다	UNIQUE scope·시간 규칙
relation	양쪽 최소·최대는 몇 개입니까	선으로 연결합니다	cardinality·FK·NULL
value	type·unit·NULL은 무엇입니까	text로 받고 나중에	data dictionary
constraint	어디서 반드시 막습니까	화면에서 막습니다	UI·API·DB 책임표
deletion	parent가 사라지면 child는요	cascade 하면 됩니다	lifecycle scenario
evidence	변경·품질을 어떻게 압니까	updated_at 있습니다	audit·metric·threshold

20.2. AI가 만든 schema를 검수하는 prompt

```
아래 업무 규칙과 schema를 검토하라.

1. 각 table마다 "한 행 =" 문장을 작성하라.
2. entity·event·snapshot이 섞인 table을 찾아 근거를 제시하라.
3. PK, 업무 UNIQUE, FK, cardinality, optionality를 표로 분리하라.
4. NULL·빈 문자열·0·false·not applicable의 의미 충돌을 찾아라.
5. 금액·시간·단위·상태 type의 정확성과 domain을 검토하라.
6. 반복 열·CSV 관계·부분·간접 의존으로 생길 변경 이상을 찾아라.
7. parent별 delete/update action과 이력 손실 위험을 작성하라.
8. UI·API·DB·quality layer별 constraint 책임을 제안하라.
9. migration·backfill·rollback·consumer 호환성 위험을 작성하라.
10. 가정과 확인 질문을 사실과 분리하라.

근거 없는 table·column을 새로 발명하지 말고,
제안마다 정상 row·경계 row·위반 row 예를 하나씩 포함하라.
```

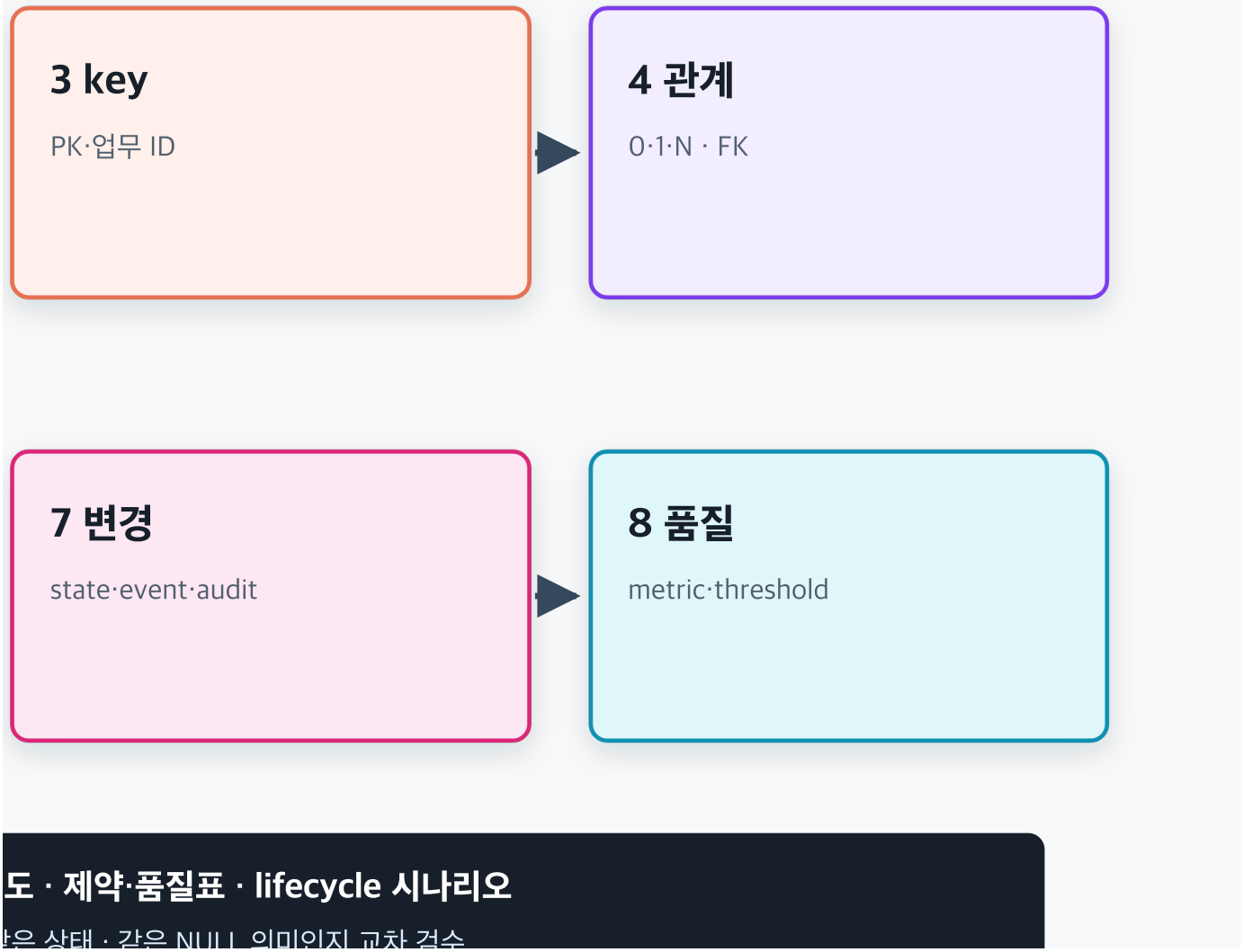
20.3. AI 제안을 그대로 적용하지 않습니다

AI는 업무 시간 규칙, 기존 데이터의 예외, 법적 보존, DB 제품별 제약, 실제 volume을 모를 수 있습니다. `왜 이 grain인가`, `어떤 사례가 constraint를 깨는가`, `기존 row를 어떻게 옮기는가`를 사람이 확인합니다.

21. 한 장 요약

그림 15. 문장→grain→key→관계→값→제약→변경→품질 순서로 읽으면 표 모양보다 업무 계약을 먼저 볼 수 있습니다.


1 문장
명사·동사·조건
2 grain
한 행 = 한 사실
5 값
type·unit·NULL
6 제약
NOT NULL·UNIQUE
제출물 · 데이터 항목 정의서 · 관계 지
같은 용어 · 같은 ID · 같은 cardinality · 같



도 · 제약·품질표 · lifecycle 시나리오

같은 상태 · 같은 NULL 의미인지 교차 검수

기억할 여덟 문장

1 명사·동사·조건은 후보이지 곧바로 table·column이 아니다.
2 모든 table에 “한 행 = 한 업무 사실”을 쓴다.
3 PK와 사용자 표시값과 업무 UNIQUE를 구분한다.
4 관계는 양쪽 최소·최대와 FK·NULL·삭제 action까지 쓴다.
5 NULL·0·false·해당 없음의 의미를 섞지 않는다.
6 무결성은 UI·API·DB·관측이 각 역할로 함께 지킨다.
7 현재 상태·업무 event·감사 증거는 목적을 분리한다.
8 품질은 dimension·metric·threshold·owner·repair로 운영한다.

최종 산출물 네 개의 교차 검수

산출물	핵심	다른 문서와 맞출 것
데이터 항목 정의서	의미·type·unit·NULL·source	API·화면·event 명칭
관계 지도	grain·PK·FK·cardinality	constraint·delete action
제약·품질표	막을 오류·측정 metric	실제 업무 규칙·SLO
lifecycle 시나리오	생성·변경·삭제·audit	상태 전이·보존 경계

22. 셀프 테스트

문제 1

회원 화면의 `course_1`, `course_2`, `course_3` 열을 그대로 저장하는 설계의 가장 큰 구조적 문제를 한 문장으로 설명하십시오.

문제 2

`enrollment` table의 grain 문장을 쓰고, `lesson_progress`와 같은 table에 두면 안 되는 이유를 설명하십시오.

문제 3

surrogate primary key가 있는데도 `UNIQUE(member_id, course_id)`가 필요할 수 있는 이유는 무엇입니까?

문제 4

회원 한 명이 profile을 최대 하나만 가질 때 FK 외에 어떤 constraint가 필요합니까?

문제 5

NULL, 빈 문자열, 0, false 중 “아직 측정하지 않은 진도”와 “측정했으며 0%”를 각각 무엇으로 표현할지 계약하십시오.

문제 6

정확한 결제 금액에 floating-point type이 위험한 이유와 대안을 쓰십시오.

문제 7

UI에서 중복 신청을 확인해도 DB UNIQUE가 필요한 이유를 동시 요청 관점에서 설명하십시오.

문제 8

course 삭제 때 enrollment에 무검토 CASCADE를 적용하면 어떤 업무 증거가 사라질 수 있습니까?

문제 9

반복 열 제거, 복합 key 전체 의존, non-key 간접 의존 제거를 초급 정규화 질문으로 각각 설명하십시오.

문제 10

default column과 generated column의 계산 시점 차이를 설명하십시오.

문제 11

`updated_at`만으로는 알 수 없는 감사 정보 네 가지를 쓰십시오.

문제 12

`duplicate enrollment = 0` 품질 metric에 필요한 scope·formula·주기·owner·repair를 간단히 설계하십시오.

답안 메모

문제	내 답의 핵심어	확신 1~5	다시 볼 그림·절
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

23. 셀프 테스트 정답과 해설

정답 1

관계의 최대 개수를 schema 열 수에 고정해 네 번째 강좌부터 구조 변경이 필요하고, 각 강좌 존재·중복을 FK·UNIQUE로 지키기 어렵습니다. enrollment row로 반복을 분리합니다.

정답 2

`enrollment 한 행 = 한 회원의 한 강좌 신청`입니다. lesson_progress는 한 신청의 한 레슨 진도라는 다른 생성 사건·key·갱신 빈도를 가지므로 같은 table에 두면 mixed grain과 다수 NULL·중복이 생깁니다.

정답 3

surrogate PK는 각 물리 row만 고유하게 만들 뿐 같은 회원·강좌 의도의 두 row를 금지하지 않습니다. 업무가 한 번만 신청을 허용한다면 해당 조합에 별도 UNIQUE가 필요합니다.

정답 4

`member_profile.member_id`에 FK와 함께 UNIQUE를 둡니다. 관계가 필수라면 NOT NULL도 검토합니다.

정답 5

아직 측정하지 않은 진도는 NULL 또는 별도 `measurement_status=not_measured`, 측정 결과 0%는 숫자 0으로 구분합니다. NULL이 “row 없음”인지 “측정 불가”인지도 계약합니다.

정답 6

일부 십진 소수는 이진 floating point에 정확히 저장되지 않아 합계·비교·반올림 오차가 납니다. 통화·scale을 명시한 exact numeric/decimal 또는 최소 화폐 단위 integer를 검토합니다.

정답 7

두 요청이 거의 동시에 “중복 없음”을 읽은 뒤 모두 INSERT할 수 있습니다. UI·API는 안내를 맡고 DB UNIQUE가 경쟁 조건에서 마지막 원자적 거부를 맡습니다.

정답 8

수강 상태·구매 당시 가격·진도 연결·결제 참조·취소 이유·감사 추적이 함께 사라질 수 있습니다. parent와 child의 업무 수명·보존 요구를 먼저 검토합니다.

정답 9

반복 열·목록 cell을 별도 row로 바꾸고, 복합 key table의 non-key 값은 key 전체에 의존하게 하며, non-key 값이 다른 non-key 값에 의존하면 실제 owner entity로 옮깁니다. 이는 초급 학습 질문이며 formal 정의 전체는 아닙니다.

정답 10

default는 INSERT 때 값을 생략한 새 row에 한 번 적용됩니다. generated column은 source column이 바뀔 때 생성식으로 다시 계산되며 사용자가 임의 override하지 못합니다. 세부 기능은 DB 제품별로 확인합니다.

정답 11

actor, 변경 전후 값, 이유, source·channel, request/job/event ID, 승인자 가운데 네 가지 이상을 들 수 있습니다. 필요한 증거와 민감정보 제외 정책을 함께 정합니다.

정답 12

예: active 계열 enrollment를 scope로 하고 `(member_id, course_id)` 중복 group 수를 10분마다 계산해 threshold 0으로 둡니다. enrollment owner가 경보를 받고 canonical row 선택·child 재연결·중복 폐기·쓰기 경로 수정을 수행합니다.

24. 최종 제출 체크리스트와 공식 근거

24.1. 제출 전 체크리스트

확인	질문	완료
범위	이번 기능의 entity·event·snapshot 후보를 구분했는가	☐
grain	모든 table에 `한 행 =` 문장이 있는가	☐
사례	정상·경계·위반 row를 적어 보았는가	☐
key	stable PK와 업무 UNIQUE를 구분했는가	☐
관계	양쪽 0·1·N, FK 위치, NULL을 정했는가	☐
다대다	bridge entity와 관계 속성을 확인했는가	☐
항목	의미·type·format·unit·NULL·source·owner가 있는가	☐
제약	UI·API·DB·관측 책임을 나눴는가	☐

확인	질문	완료
삭제	관계별 update/delete action을 scenario로 검수했는가	☐
정규화	반복·부분·간접 의존과 source of truth를 확인했는가	☐
파생	default·generated·snapshot 계산 책임이 분명한가	☐
증거	state·event·audit의 필요 수준을 정했는가	☐
변경	migration·backfill·호환성·rollback 계획이 있는가	☐
품질	metric·threshold·owner·repair가 있는가	☐
교차 검수	화면·API·event·DB 문서가 같은 의미를 말하는가	☐

24.2. 공식·권위 자료

- PostgreSQL 18 · Data Definition
- PostgreSQL 18 · Constraints
- PostgreSQL 18 · Identity Columns
- PostgreSQL 18 · Generated Columns
- PostgreSQL 18 · Modifying Tables
- PostgreSQL 18 · Numeric Types
- PostgreSQL 18 · Date/Time Types
- PostgreSQL 18 · Enumerated Types
- PostgreSQL 18 · Domain Types
- RFC 9562 · Universally Unique IDentifiers
- RFC 3339 · Date and Time on the Internet
- Microsoft Learn · Introduction to relationships in EF Core
- Microsoft Learn · Entity Framework Overview
- Microsoft Learn · Database normalization description
- W3C · Data Quality Vocabulary

24.3. 정확성 메모

- 여덟 gate는 YEONCORE 학습 프레임이며 특정 표준이 강제하는 architecture가 아닙니다.
- 본문 SQL은 관계와 constraint를 읽기 위한 PostgreSQL 18 학습 예이며 migration·index·transaction·보안·성능 설계를 완성한 production DDL이 아닙니다.

- `identity`는 자동 값 생성을 돕지만 그 자체가 모든 업무 중복을 막지 않습니다.
- PK는 row identity이고 업무 UNIQUE는 별도 규칙일 수 있습니다.
- CHECK는 DB 제품별 기능 범위가 다르며 PostgreSQL에서는 cross-row·cross-table 지속 보장 용도로 쓰지 않습니다.
- UUID는 identifier이지 인증·인가·비밀이 아닙니다.
- 정규화 1·2·3단계 설명은 초급 검수 질문이며 formal relational theory 전체를 대신하지 않습니다.
- JSON·array는 항상 잘못이 아니며 독립 관계와 owner-owned 구조 값의 차이를 검토합니다.
- W3C Data Quality Vocabulary는 2016년 Working Group Note이며 Recommendation이라고 표시하지 않습니다.
- 보존·backup·복구·파기와 개인정보 lifecycle의 상세 설계는 M05-03과 G10 과정에서 이어집니다.

24.4. 완료 기준

다음 문장을 사례 row와 constraint로 증명할 수 있으면 완료입니다.

> “이 table의 한 행은 무엇이고 어떤 key로 식별되며 누구와 몇 개씩 연결되는지, 어떤 값만 허용되고 parent가 바뀌거나 사라질 때 무엇이 남는지, 중복·누락·orphan·지연을 어떤 metric으로 발견하는지 설명할 수 있다.”