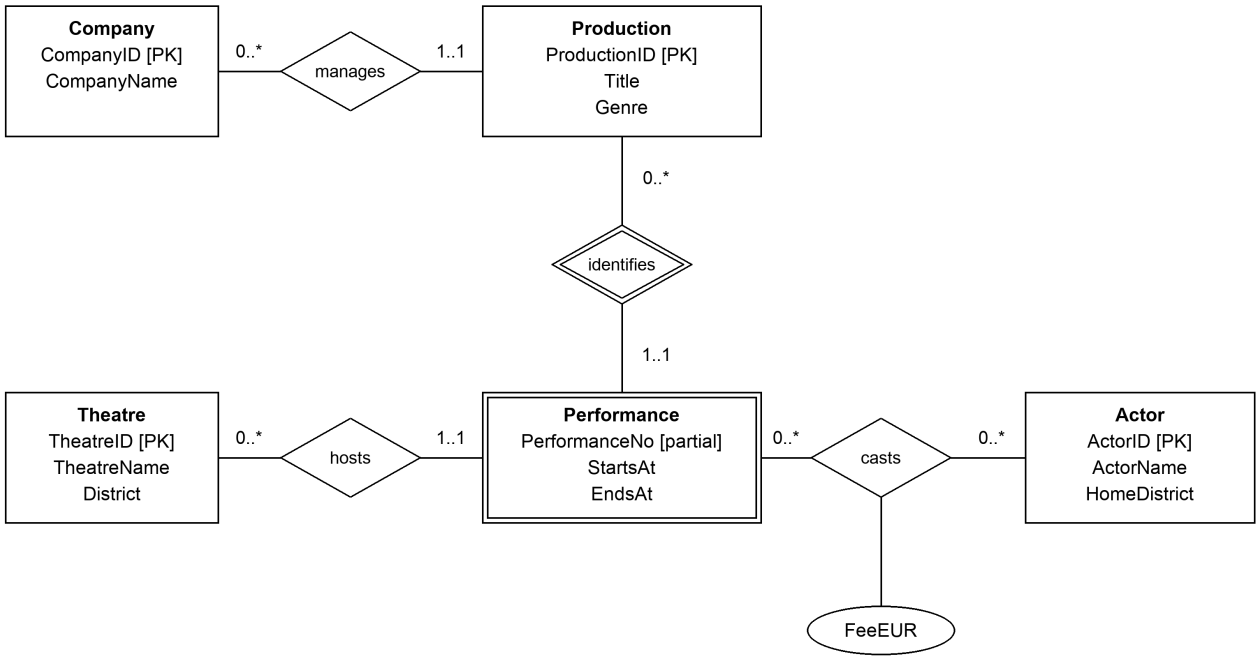
Exam 2 Solutions

Lyon Theatre Shows and Actors

Model answers for all three exercises: ER design, relational reduction, validation requests, normal forms, relational algebra, SQL and theory. Equivalent correct answers are acceptable.

Exercise 1 ER diagram



PK denotes a strong-entity identifier. The double rectangle denotes a weak entity, [partial] its discriminator, and the double diamond its identifying relationship. Each min/max label gives the number of relationship instances allowed for one instance of the adjacent entity. A minimum of 1 denotes total participation.

Functional dependencies

Company: (CompanyID) → CompanyName

Production: (ProductionID) → Title, Genre, CompanyID

Theatre: (TheatreID) → TheatreName, District

Performance: (ProductionID, PerformanceNo) → StartsAt, EndsAt, TheatreID

Actor: (ActorID) → ActorName, HomeDistrict

Casting: (ProductionID, PerformanceNo, ActorID) → FeeEUR

These are a basis for the stated functional dependencies. In every relation the determinant is a candidate key, so all six relations are in BCNF. The next page gives the reduction and full key constraints.

Exam 2 Lyon Theatre Shows and Actors

Exercise 1 Model answer

Strong entities: Company, Production, Theatre and Actor, identified by their respective IDs. Weak entity: Performance, identified by (ProductionID, PerformanceNo); PerformanceNo is its discriminator. Attributes appear in the relational reduction below.

One Company has 0..* Production records; each Production has exactly one Company. One Production has 0..* Performance records; each Performance has exactly one identifying Production. One Theatre has 0..* Performance records; each Performance has exactly one Theatre. Each Actor participates in 0..* Performance records, and each Performance has 0..* participating Actor records. The many-to-many relationship stores FeeEUR.

Company(CompanyID, CompanyName)

PK: CompanyID

Production(ProductionID, Title, Genre, CompanyID)

PK: ProductionID; FK: CompanyID → Company.CompanyID

Theatre(TheatreID, TheatreName, District)

PK: TheatreID

Performance(ProductionID, PerformanceNo, StartsAt, EndsAt, TheatreID)

PK: ProductionID, PerformanceNo; FK: ProductionID → Production.ProductionID; TheatreID → Theatre.TheatreID

Actor(ActorID, ActorName, HomeDistrict)

PK: ActorID

Casting(ProductionID, PerformanceNo, ActorID, FeeEUR)

PK: ProductionID, PerformanceNo, ActorID; FK: (ProductionID, PerformanceNo) → Performance(ProductionID, PerformanceNo); ActorID → Actor.ActorID

All columns are atomic and non-null. Require start < end (TakeoffAt < ArrivalAt for the airport) and monetary amount ≥ 0. The association primary key prevents duplicate participation. The composite foreign key must reference one complete occurrence, not independently chosen parent and occurrence-number values.

Validation requests

a. Filter Casting by both ProductionID and PerformanceNo; join Actor on ActorID. Return person ID/name and FeeEUR.

b. Join Performance to Production on ProductionID and to Theatre on TheatreID; filter Theatre.District = 7 and start/take-off ≥ 2027-01-01 and < 2027-02-01. Return Title, PerformanceNo and TheatreName.

c. Start with all Actor records and exclude every ActorID appearing in Casting. No relationship to a particular occurrence is required. All three requests are supported.

Keys and normal forms

For each listed relation, the displayed primary key is its only candidate key under the stated rules and determines every other attribute in that relation. These six key-to-non-key dependencies form a sufficient basis for the dependencies in the reference design; no further independent non-trivial dependencies are stated. Thus every relation is in BCNF, and therefore also in 3NF, 2NF and 1NF.

Exam 2 Query solutions 1

Notation: π = projection; σ = selection; ⋈ = theta join; ρ = rename; − = set difference; ∧ = AND. Brackets specify attributes or predicates. Read join chains left to right.

S = ρS(Show), A = ρA(Actor), C = ρC(Casting)

J = A ⋈[A.ActorID = C.ActorID] C ⋈[C.ShowID = S.ShowID] S

Query 1

Give all show titles.

π[Title](Show)

```
SELECT DISTINCT Title
FROM Show;
```

Query 2

Give the IDs and names of actors whose home district is 7.

π[ActorID, ActorName](σ[HomeDistrict = 7](Actor))

```
SELECT DISTINCT ActorID, ActorName
FROM Actor
WHERE HomeDistrict = 7;
```

Query 3

Give the IDs and names of actors cast in at least one show in district 7 during January 2027.

π[A.ActorID, A.ActorName](σ[S.District = 7 ∧
S.ShowDate ≥ '2027-01-01' ∧ S.ShowDate < '2027-02-01'](J))

```
SELECT DISTINCT A.ActorID, A.ActorName
FROM Actor A
JOIN Casting C ON A.ActorID = C.ActorID
JOIN Show S ON C.ShowID = S.ShowID
WHERE S.District = 7
AND S.ShowDate >= '2027-01-01' AND S.ShowDate < '2027-02-01';
```

Exam 2 Query solutions 2

S = ρS(Show), A = ρA(Actor), C = ρC(Casting)

J = A ⋈[A.ActorID = C.ActorID] C ⋈[C.ShowID = S.ShowID] S

Query 4

Give the IDs and titles of shows with at least one cast actor whose home district equals the show district.

π[S.ShowID, S.Title](σ[A.HomeDistrict = S.District](J))

```
SELECT DISTINCT S.ShowID, S.Title
FROM Actor A
JOIN Casting C ON A.ActorID = C.ActorID
JOIN Show S ON C.ShowID = S.ShowID
WHERE A.HomeDistrict = S.District;
```

Query 5

Give the IDs and names of actors with no casting in any show during January 2027. Include actors who have never been cast.

π[A.ActorID, A.ActorName](A) −
π[A.ActorID, A.ActorName](σ[S.ShowDate ≥ '2027-01-01' ∧ S.ShowDate < '2027-02-01'](J))

```
SELECT DISTINCT A.ActorID, A.ActorName
FROM Actor A
WHERE NOT EXISTS (
SELECT 1 FROM Casting C
JOIN Show S ON C.ShowID = S.ShowID
WHERE C.ActorID = A.ActorID
AND S.ShowDate >= '2027-01-01' AND S.ShowDate < '2027-02-01'
);
```

Checks on the illustrative instance

Q1: ('Last Curtain',); ('Night Voices',); ('The Bridge',); ('Winter Stage',)

Q2: (10, 'Camille'); (40, 'Sacha')

Q3: (10, 'Camille'); (20, 'Jules')

Q4: (1, 'Night Voices'); (2, 'The Bridge'); (4, 'Last Curtain')

Q5: (30, 'Leila'); (40, 'Sacha')

Q5 includes both the person associated only with a February event and the person with no associations at all. An inner join followed only by a non-January filter would fail this requirement.

Exam 2 Theory solutions

Open question 1

Union-compatible relations have the same number of attributes and compatible corresponding domains (with names aligned or renamed where required). The union contains actor IDs in either relation, without duplicates; the difference contains cast actor IDs absent from AvailableActor.

CastActor has one attribute and Actor has two, so they are not union-compatible. Projecting Actor onto ActorID would make the proposed union valid. Marking: compatibility conditions 0.5; union interpretation 0.5; difference interpretation 0.5; arity mismatch 0.5.

Open question 2

A transaction is a sequence of database operations treated as one unit of work. COMMIT successfully ends the transaction and makes its changes permanent. ROLLBACK cancels the transaction and undoes its uncommitted changes.

The manager should use ROLLBACK to undo the incorrect fee update before starting a new transaction with the correct change. Marking: transaction as a unit of work 0.5; COMMIT explanation 0.5; ROLLBACK explanation 0.5; choosing ROLLBACK with a correct reason 0.5.

Multiple-choice answers

Q3: D. For every non-trivial functional dependency X → Y, X is a superkey.

BCNF requires every determinant of a non-trivial functional dependency to be a superkey. Atomic values alone concern 1NF; absence of partial dependencies alone concerns 2NF.

Q4: A. HAVING

HAVING applies a condition to groups or aggregates. WHERE filters input rows, DISTINCT removes duplicates, and ORDER BY sorts results.

Q5: C. Isolation

Isolation prevents transactions from observing another transaction's uncommitted changes at the isolation level assumed in the lecture. The read may use a fee that is later rolled back.