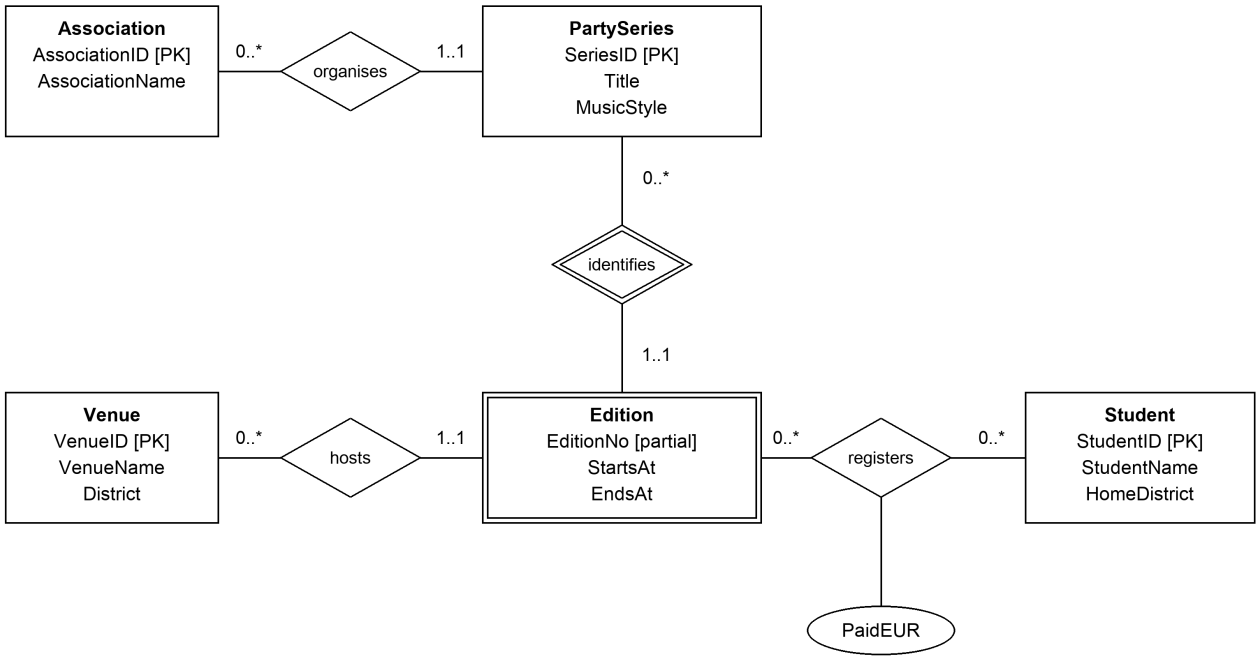
Exam 1 Solutions

Lyon Student Party Scene

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

Association: (AssociationID) → AssociationName

PartySeries: (SeriesID) → Title, MusicStyle, AssociationID

Venue: (VenueID) → VenueName, District

Edition: (SeriesID, EditionNo) → StartsAt, EndsAt, VenueID

Student: (StudentID) → StudentName, HomeDistrict

Registration: (SeriesID, EditionNo, StudentID) → PaidEUR

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 1 Lyon Student Party Scene

Exercise 1 Model answer

Strong entities: Association, PartySeries, Venue and Student, identified by their respective IDs. Weak entity: Edition, identified by (SeriesID, EditionNo); EditionNo is its discriminator. Attributes appear in the relational reduction below.

One Association has 0..* PartySeries records; each PartySeries has exactly one Association. One PartySeries has 0..* Edition records; each Edition has exactly one identifying PartySeries. One Venue has 0..* Edition records; each Edition has exactly one Venue. Each Student participates in 0..* Edition records, and each Edition has 0..* participating Student records. The many-to-many relationship stores PaidEUR.

Association(AssociationID, AssociationName)

PK: AssociationID

PartySeries(SeriesID, Title, MusicStyle, AssociationID)

PK: SeriesID; FK: AssociationID → Association.AssociationID

Venue(VenueID, VenueName, District)

PK: VenueID

Edition(SeriesID, EditionNo, StartsAt, EndsAt, VenueID)

PK: SeriesID, EditionNo; FK: SeriesID → PartySeries.SeriesID; VenueID → Venue.VenueID

Student(StudentID, StudentName, HomeDistrict)

PK: StudentID

Registration(SeriesID, EditionNo, StudentID, PaidEUR)

PK: SeriesID, EditionNo, StudentID; FK: (SeriesID, EditionNo) → Edition(SeriesID, EditionNo); StudentID → Student.StudentID

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 Registration by both SeriesID and EditionNo; join Student on StudentID. Return person ID/name and PaidEUR.

b. Join Edition to PartySeries on SeriesID and to Venue on VenueID; filter Venue.District = 7 and start/take-off ≥ 2027-01-01 and < 2027-02-01. Return Title, EditionNo and VenueName.

c. Start with all Student records and exclude every StudentID appearing in Registration. 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 1 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.

P = ρP(Party), S = ρS(Student), R = ρR(Registration)

J = S ⋈[S.StudentID = R.StudentID] R ⋈[R.PartyID = P.PartyID] P

Query 1

Give all party titles.

π[Title](Party)

```
SELECT DISTINCT Title
FROM Party;
```

Query 2

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

π[StudentID, StudentName](σ[HomeDistrict = 7](Student))

```
SELECT DISTINCT StudentID, StudentName
FROM Student
WHERE HomeDistrict = 7;
```

Query 3

Give the IDs and names of students registered for at least one party in district 7 during January 2027.

π[S.StudentID, S.StudentName](σ[P.District = 7 ∧
P.EventDate ≥ '2027-01-01' ∧ P.EventDate < '2027-02-01'](J))

```
SELECT DISTINCT S.StudentID, S.StudentName
FROM Student S
JOIN Registration R ON S.StudentID = R.StudentID
JOIN Party P ON R.PartyID = P.PartyID
WHERE P.District = 7
AND P.EventDate >= '2027-01-01' AND P.EventDate < '2027-02-01';
```

Exam 1 Query solutions 2

P = ρP(Party), S = ρS(Student), R = ρR(Registration)

J = S ⋈[S.StudentID = R.StudentID] R ⋈[R.PartyID = P.PartyID] P

Query 4

Give the IDs and titles of parties with at least one registered student whose home district equals the party district.

π[P.PartyID, P.Title](σ[S.HomeDistrict = P.District](J))

```
SELECT DISTINCT P.PartyID, P.Title
FROM Student S
JOIN Registration R ON S.StudentID = R.StudentID
JOIN Party P ON R.PartyID = P.PartyID
WHERE S.HomeDistrict = P.District;
```

Query 5

Give the IDs and names of students with no registration for any party during January 2027. Include students who have never registered.

π[S.StudentID, S.StudentName](S) −
π[S.StudentID, S.StudentName](σ[P.EventDate ≥ '2027-01-01' ∧ P.EventDate < '2027-02-01'](J))

```
SELECT DISTINCT S.StudentID, S.StudentName
FROM Student S
WHERE NOT EXISTS (
SELECT 1 FROM Registration R
JOIN Party P ON R.PartyID = P.PartyID
WHERE R.StudentID = S.StudentID
AND P.EventDate >= '2027-01-01' AND P.EventDate < '2027-02-01'
);
```

Checks on the illustrative instance

Q1: ('Campus Beats',); ('Late January',); ('River Night',); ('Winter Groove',)

Q2: (10, 'Amira'); (40, 'Noah')

Q3: (10, 'Amira'); (20, 'Hugo')

Q4: (1, 'Campus Beats'); (2, 'River Night'); (4, 'Late January')

Q5: (30, 'Lina'); (40, 'Noah')

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 1 Theory solutions

Open question 1

X → Y means that, in every legal instance, any two tuples agreeing on X must also agree on Y. Here, all registration rows with the same StudentID must have the same StudentName.

A student registered for several parties has their name repeated in several rows. Changing it in only one row leaves inconsistent names for the same student. Marking: general dependency definition 0.5; interpretation of StudentID → StudentName 0.5; repeated information 0.5; incomplete update and resulting inconsistency 0.5.

Open question 2

(a) Atomicity: both changes take effect or neither does. The failed transaction must undo the inserted registration, leaving no partial effect. (b) Durability: once the transaction commits, both changes must survive the subsequent crash, using recovery if necessary.

Marking: identifying atomicity 0.5; rollback of the partial transaction 0.5; identifying durability 0.5; persistence of both committed changes 0.5. No particular recovery algorithm is required.

Multiple-choice answers

Q3: B. {Email}

Email alone identifies a student and is minimal. The sets containing StudentID and another attribute are superkeys but not minimal; Name is not guaranteed unique.

Q4: C. Adding a BirthDate attribute

Adding an attribute changes the relation's structure. Insertion, update and deletion change its instance while leaving the schema unchanged.

Q5: A. 2

Relational projection eliminates duplicates. The result contains the two distinct values Lyon and Paris.