Conceptual Modelling - Part 2

IDATG2204 Data Modelling and Database Systems

Norwegian University of Science and Technology

Where are We Now?

- W02: Introduction, Relational Algebra
- W03: SQL
- W04: SQL, Conceptual Modelling
- W05: Conceptual Modelling
- W06: Normalisation
- W07: Logical Modelling, NOSQL
- W08: DB Application Development
- W09: DB Security, Project Kick-off
- W10-W14: Project Work with Peer Review
- W15: Indexing, query processing, concurrency
- W16: Recovery
- W17: More SQL and NOSQL
- W18: Review and Wrap-up

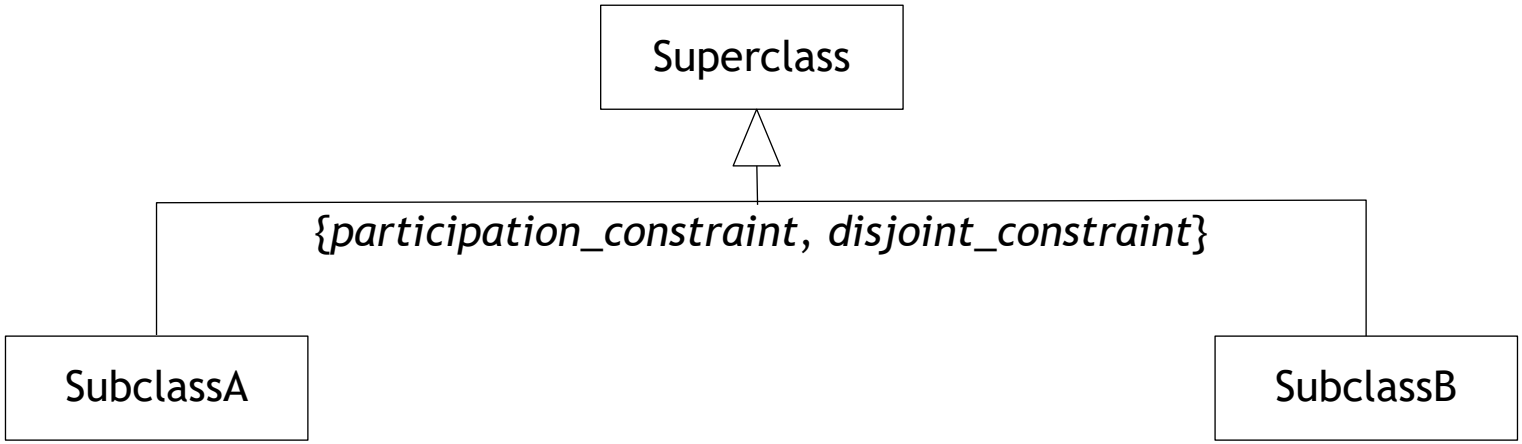
Outline

- Extended ER modelling:
  - Specialisation/generalisation
  - Aggregation
  - Composition
- ER model quality:
  - ER modelling traps
  - Validating ER models

Specialisation/Generalisation

- Superclass/subclass relationship:
  - A subclass entity **is a** superclass entity
- Attribute inheritance:
  - A subclass entity inherits **all** superclass entity attributes
- Design process:
  - Specialisation:
    - Top-down approach defining each subclass' distinguishing characteristic
  - Generalisation:
    - Bottom-up approach identifying similarities among subclasses

UML Notation



Participation Constraints

- Mandatory:
  - The superclass is an abstract class that cannot be instantiated
  - Example:
    - A student is either a bachelor student, a master student, or a PhD student
    - A university user is either a staff member or a student
- Optional:
  - The superclass may be instantiated
  - Example:
    - A student may be a "blåfadder", may be a"rødfadder", may be a "gulfadder“, or may be just a student
    - A member of staff may be a researcher, may be a lecturer, or may be just a staff member (e.g., an administrative person)

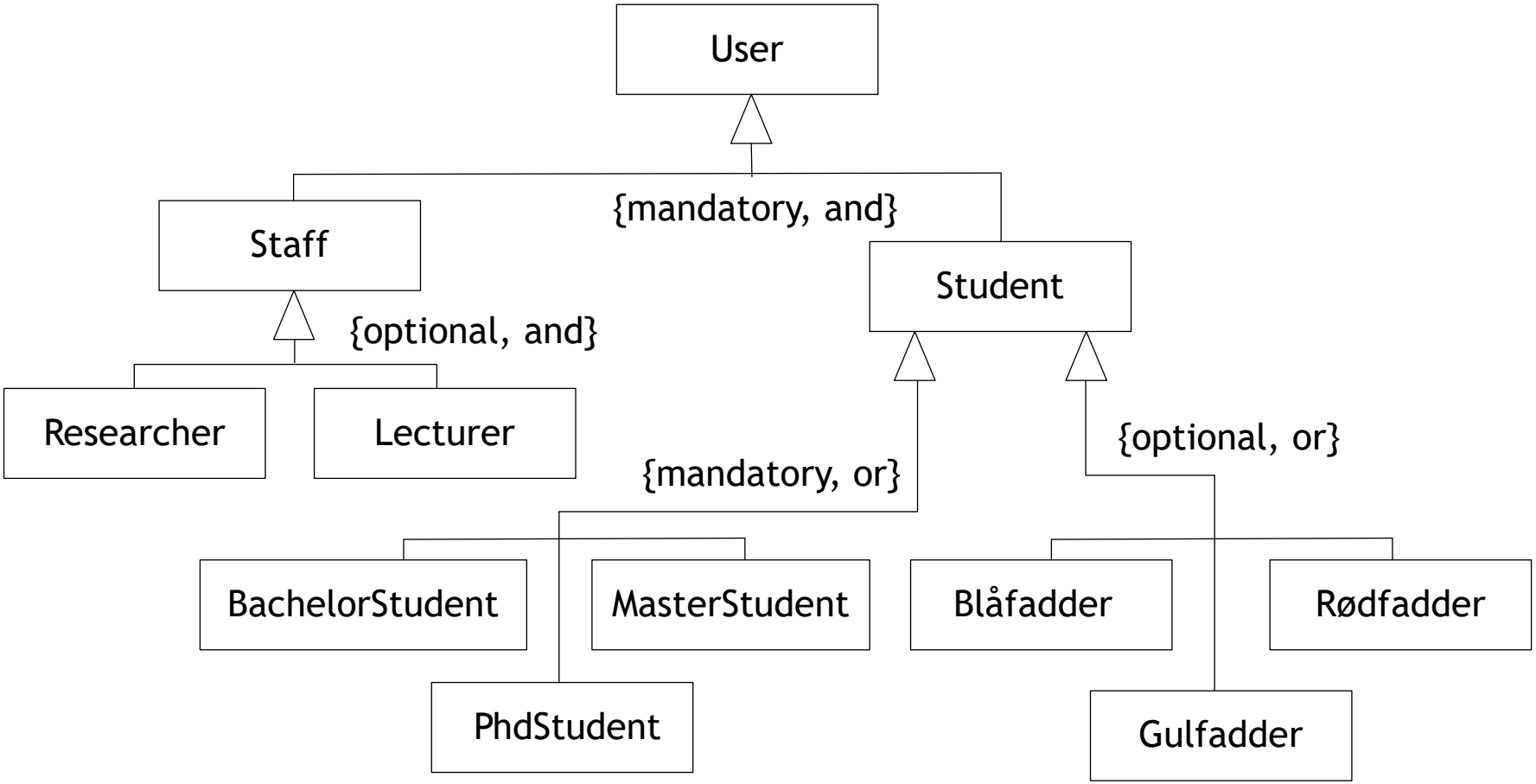
Disjoint Constraints

- Disjoint:
  - An entity occurrence may only be of one subclass type:
    - A student is **either** a bachelor student, a master student, or a PhD student
    - A student may be a "blåfadder", may be a"rødfadder", or may be a "gulfadder"
- Overlapping:
  - An entity occurrence may belong to more that one subclass at one point in time:
    - A university user is a staff member, a student, or both
    - A member of staff may be a researcher and/or may be a lecturer

Participation/Disjoint Combinations

- Mandatory, disjoint:
  - A student is a bachelor student, a master student, or a PhD student
- Mandatory, overlapping:
  - A university user is a staff member, a student, or both
- Optional, disjoint:
  - A student may be a "blåfadder", may be a"rødfadder", or may be a "gulfadder"
- Optional, overlapping:
  - A member of staff may be a researcher and may be a lecturer

Example Specialisation Hierarchy



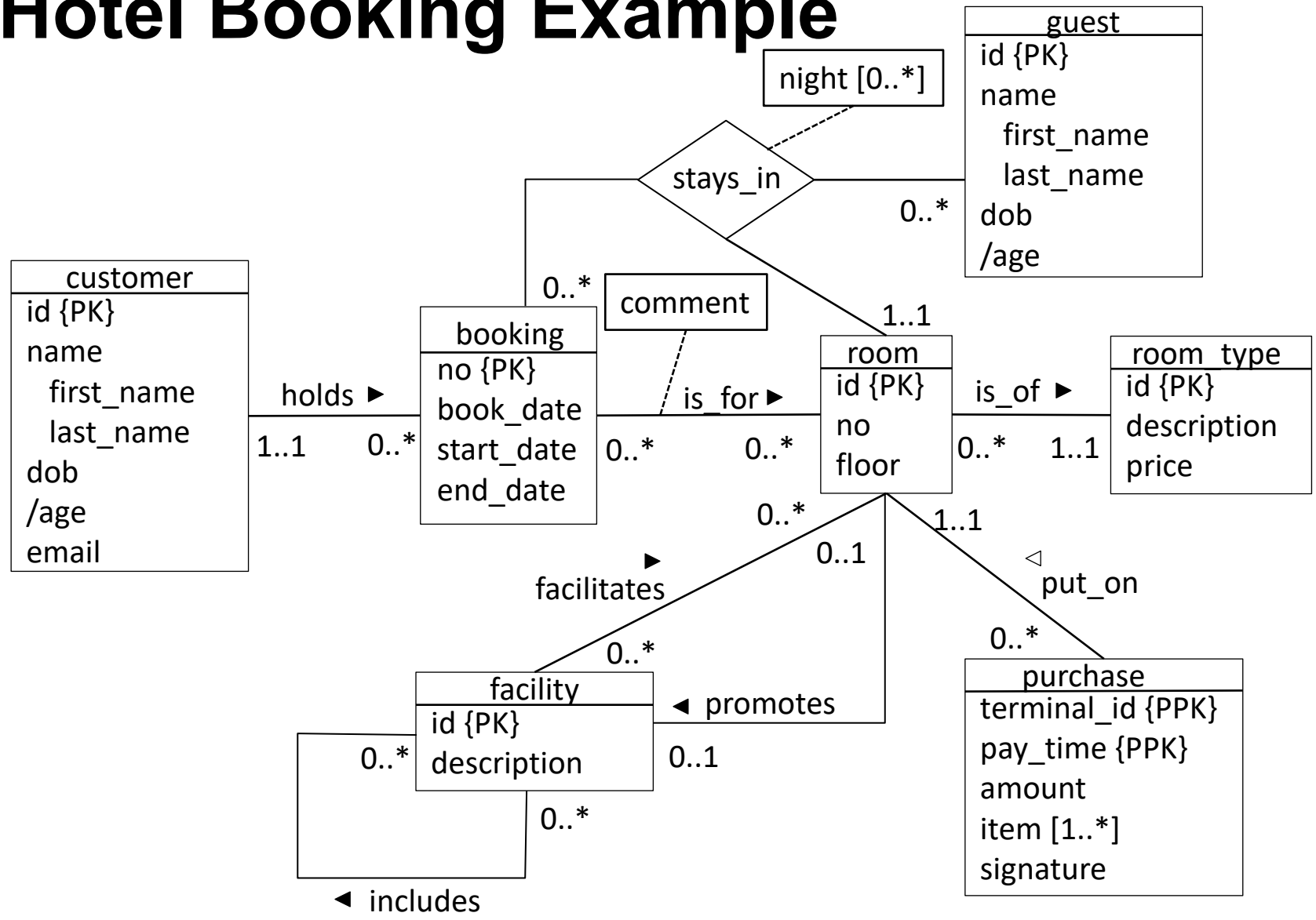
Subclasses – an Additional Note

- Subclasses are only needed when they are structurally different from their superclasses:
  - Having different set of attributes, and/or
  - Participating in different relationship types
    - Moderators may moderate blogs beside owning blogs
- Otherwise we use the entity type pattern:
  - Types of hotel rooms
  - Types of wines
  - …
- Or just an enumeration attribute domain:
  - Female/male
  - …

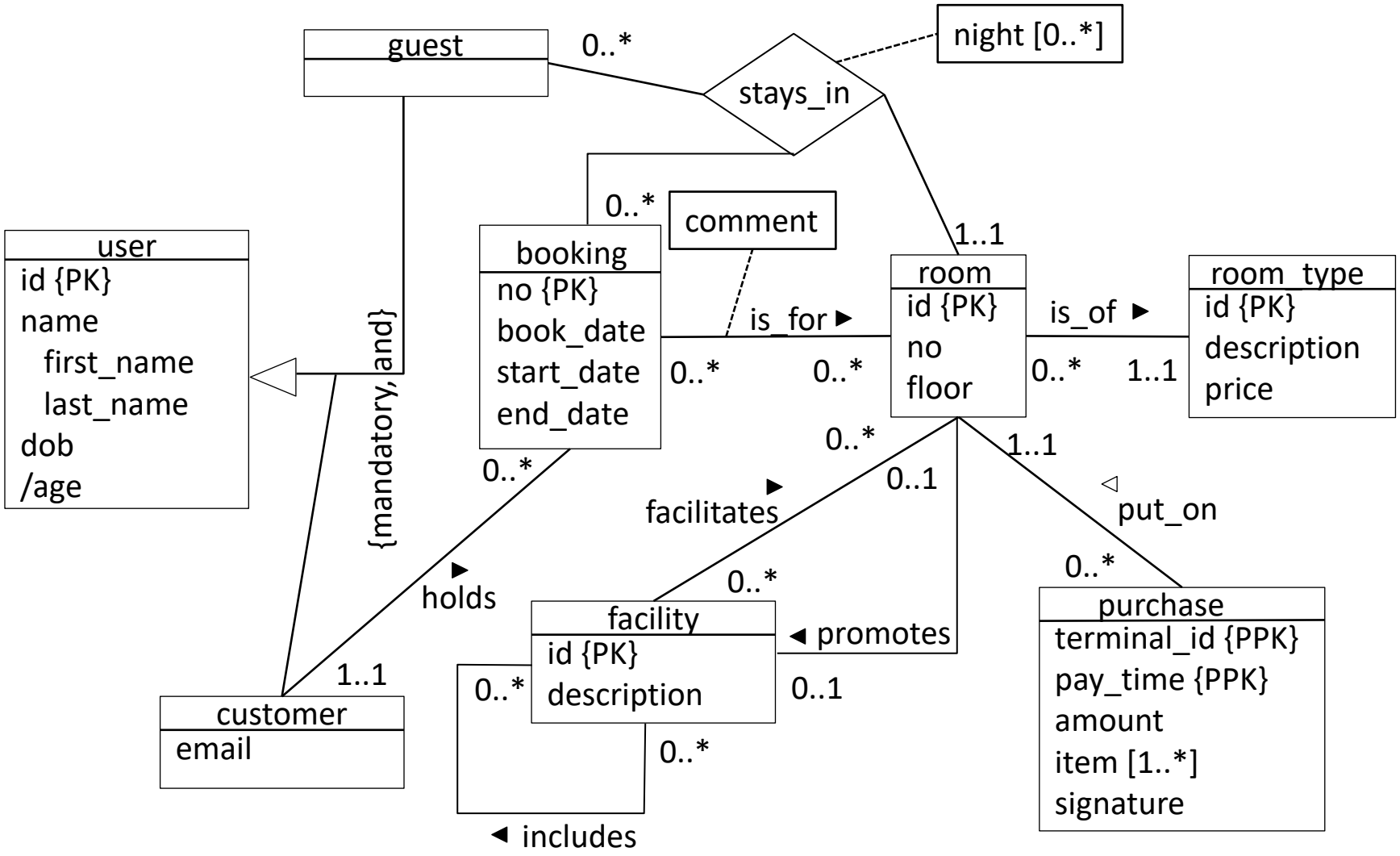
Example

- Could any of the entity types in the hotel booking case be joined into a generalised type?

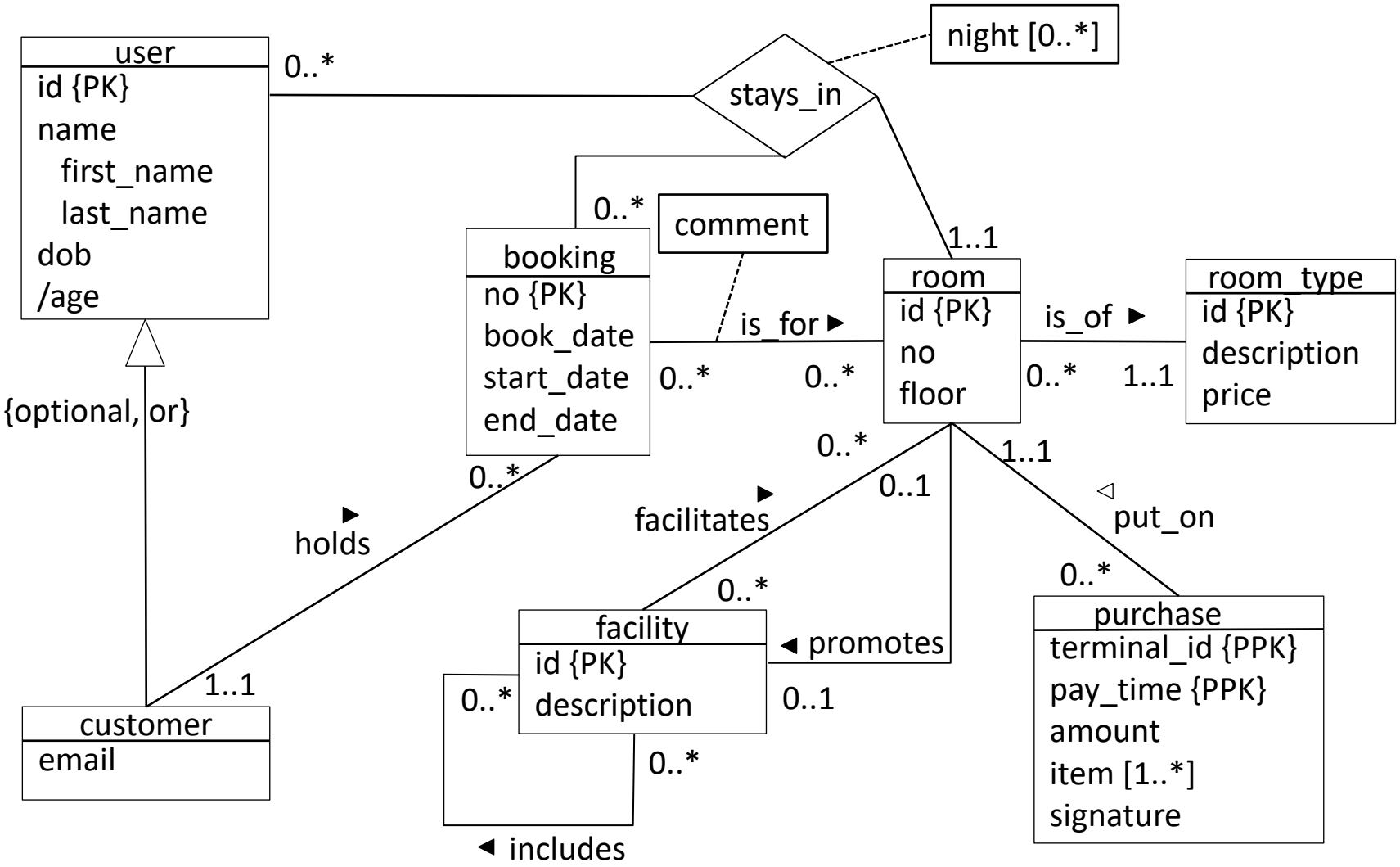
Hotel Booking Example



Hotel Booking Example (1)



Hotel Booking Example (2)



To think about

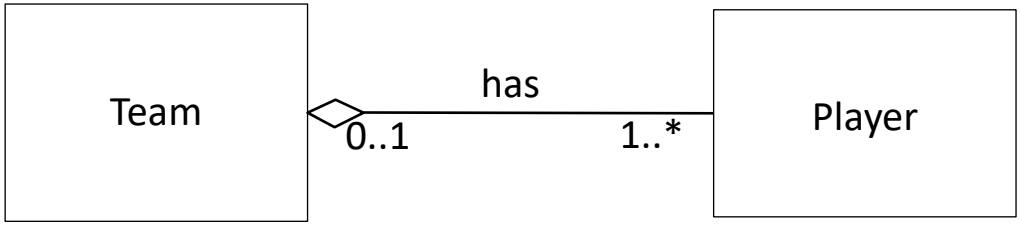
- Why having a `room_type` entity type and not having subclasses for the various types of rooms?
  - Hint: Are the various types of rooms structurally different?

Outline

- Extended ER modelling:
  - Specialisation/generalisation
  - Aggregation
  - Composition
- ER model quality:
  - ER modelling traps
  - Validating ER models

Aggregation

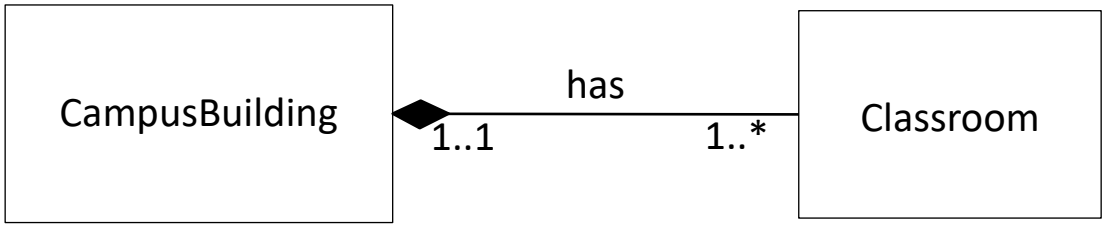
- Represents a `has`/`isPartOf` relationships between entity parts:
  - A team has players
  - A server is part of a server cluster
- UML:



- Could the "whole" exist without the "parts"?
- The "part" may be part of more than one "whole":
  - A researcher may be part of more than one research team

Composition

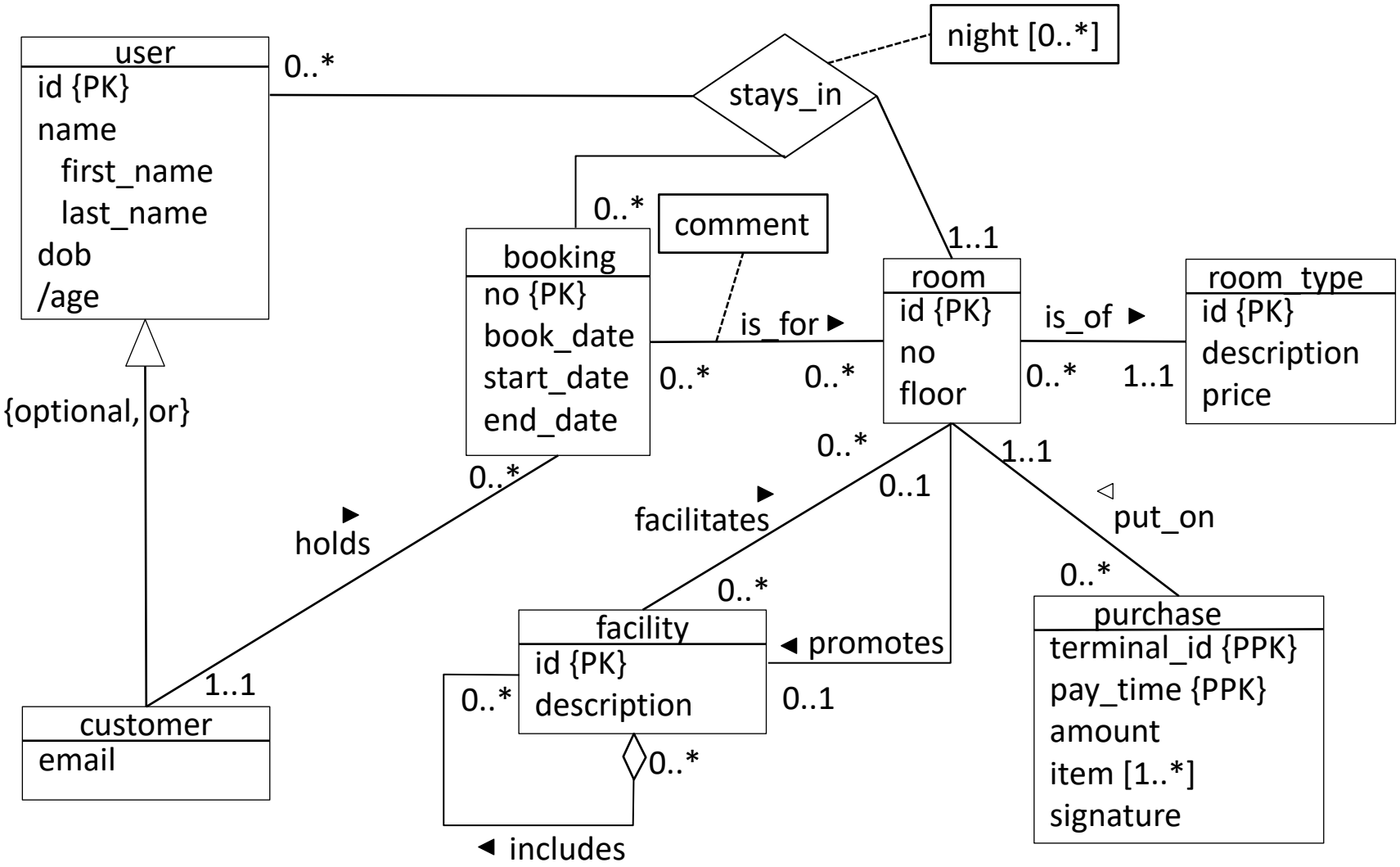
- A specific form of aggregation where there is a strong ownership and coincidental lifetime between the "whole" and the "part":
  - compositions are aggregations
  - strong (not shared) ownership of parts
  - coincident lifetimes of whole and parts
    - The "parts" cannot exist without the "whole"
- UML:



Example

- Could aggregation and/or composition be of use in the example model?

Hotel Booking Example



Aggregation/Composition or not?

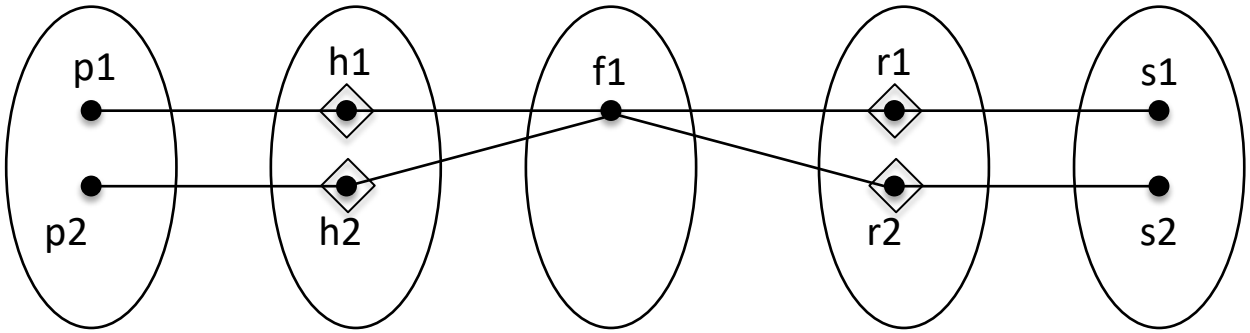
- Should only be used to emphasise special relationships (`has`, `isPartOf`, ...), which has implications for creation, update, and deletion of these closely related objects
  - Weak entity types or composition?
- We will revisit this when discussing how to map the ER model to a relational database model

Outline

- Extended ER modelling:
  - Specialisation/generalisation
  - Aggregation
  - Composition
- ER model quality:
  - ER modelling traps
  - Validating ER models

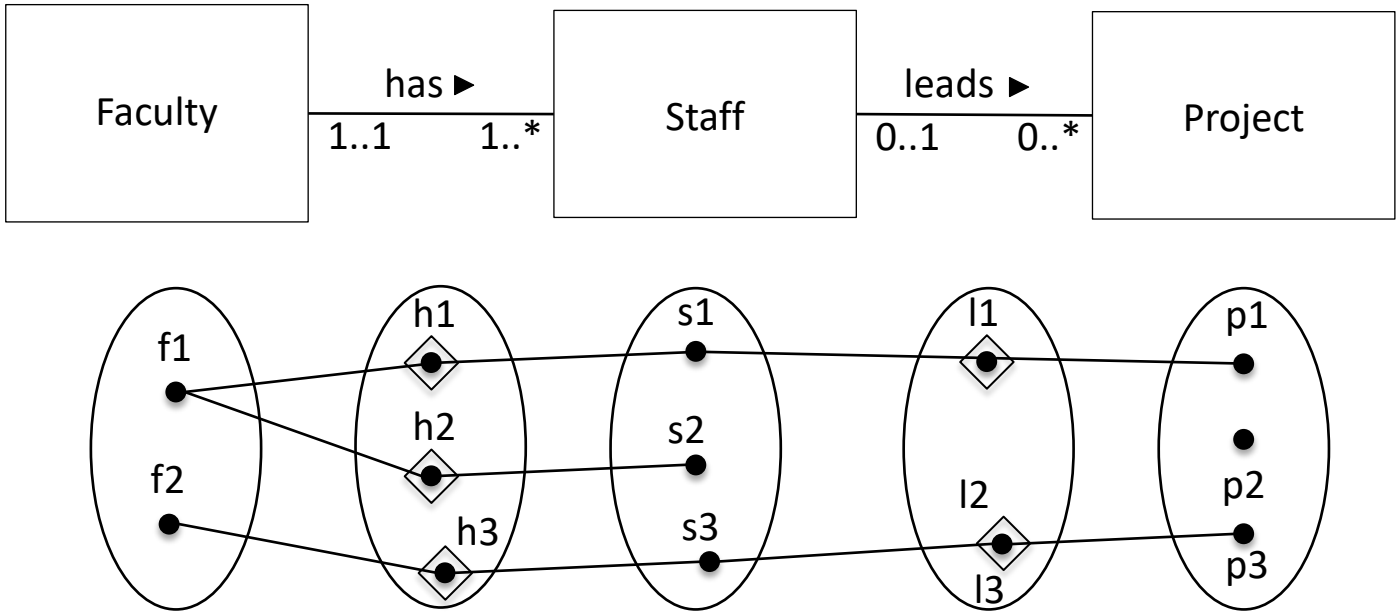
Fan Traps

- Where a model represents a relationship between entity types, but the pathway between certain entity occurrences are ambiguous
  - Example: Some staff members work on some projects, but who works on which project?



Chasm Traps

- Where a model suggests the existence of a relationship between types, but the pathway does not exist between certain occurrences
  - Example: Projects may only be lead by a member the faculty, but some projects may not have a project manager:



Outline

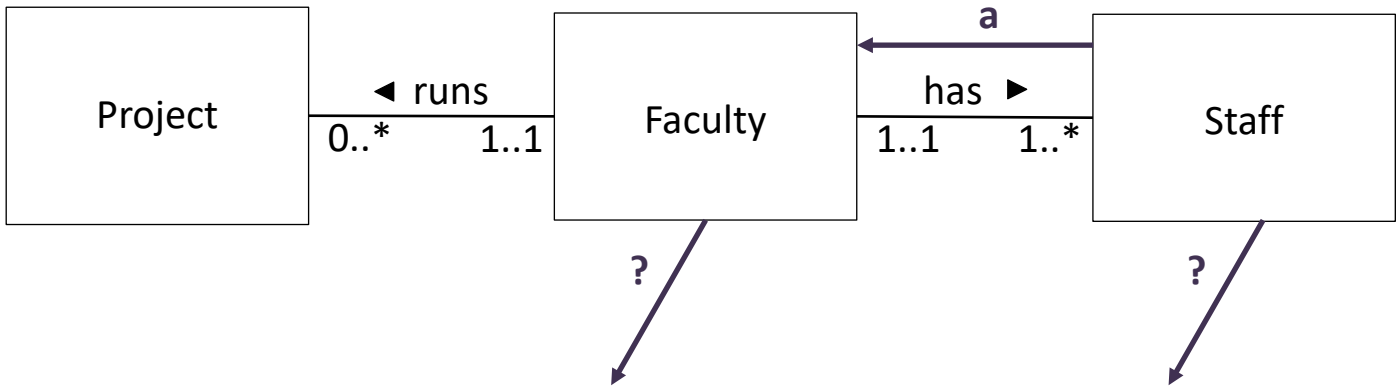
- Extended ER modelling:
  - Specialisation/generalisation
  - Aggregation
  - Composition
- ER model quality:
  - ER modelling traps
  - Validating ER models

Checking, Validating, and Reviewing the Model

- Re-examine one-to-one relationships types:
  - Could two entities related by a one-to-one possibly represent the same entity?
- Remove redundant relationship types:
  - Are there more than one path between entities?
  - Can the same information be retrieved along several paths?
- Consider time dimension:
  - Are alternative paths always "aligned"?
    - Example: In a family situation, is fatherOf the same as marriedTo motherOf?

Validating Model against User Transactions

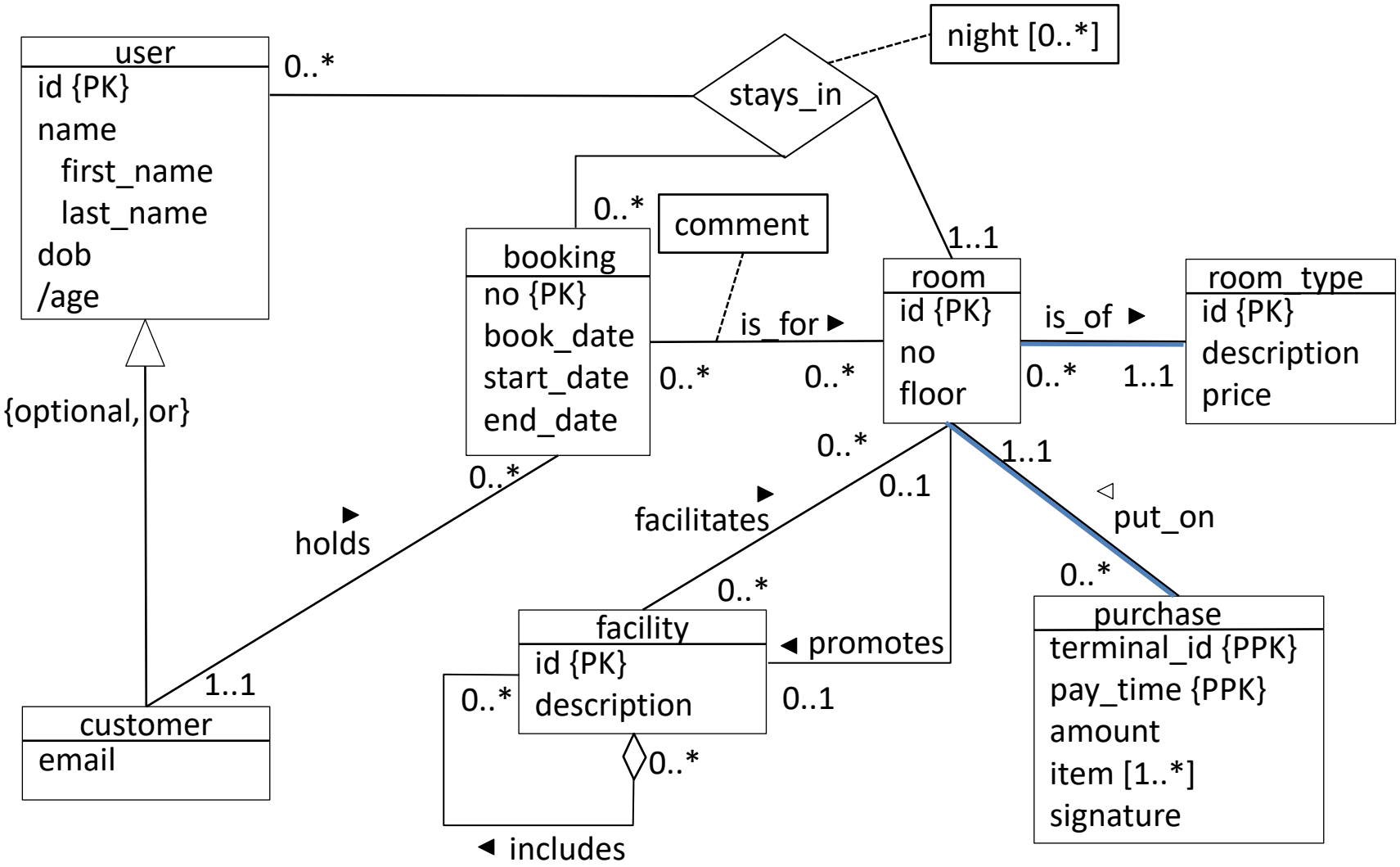
- To ensure that the model support the required transactions
- Simulating the operations manually:
  - Using pathways:
    - What projects are staff member x working on?



Validating the Example Model

- Sample queries:
  - What's the average purchase per room per day and room type for 2020?
  - Who were the guests stayed in room #331 on the nights of 21 - 23 December 2020 of a booking made by customer NN?
  - How many customers have booked rooms for the month of December where at least one of the rooms had a private sauna?

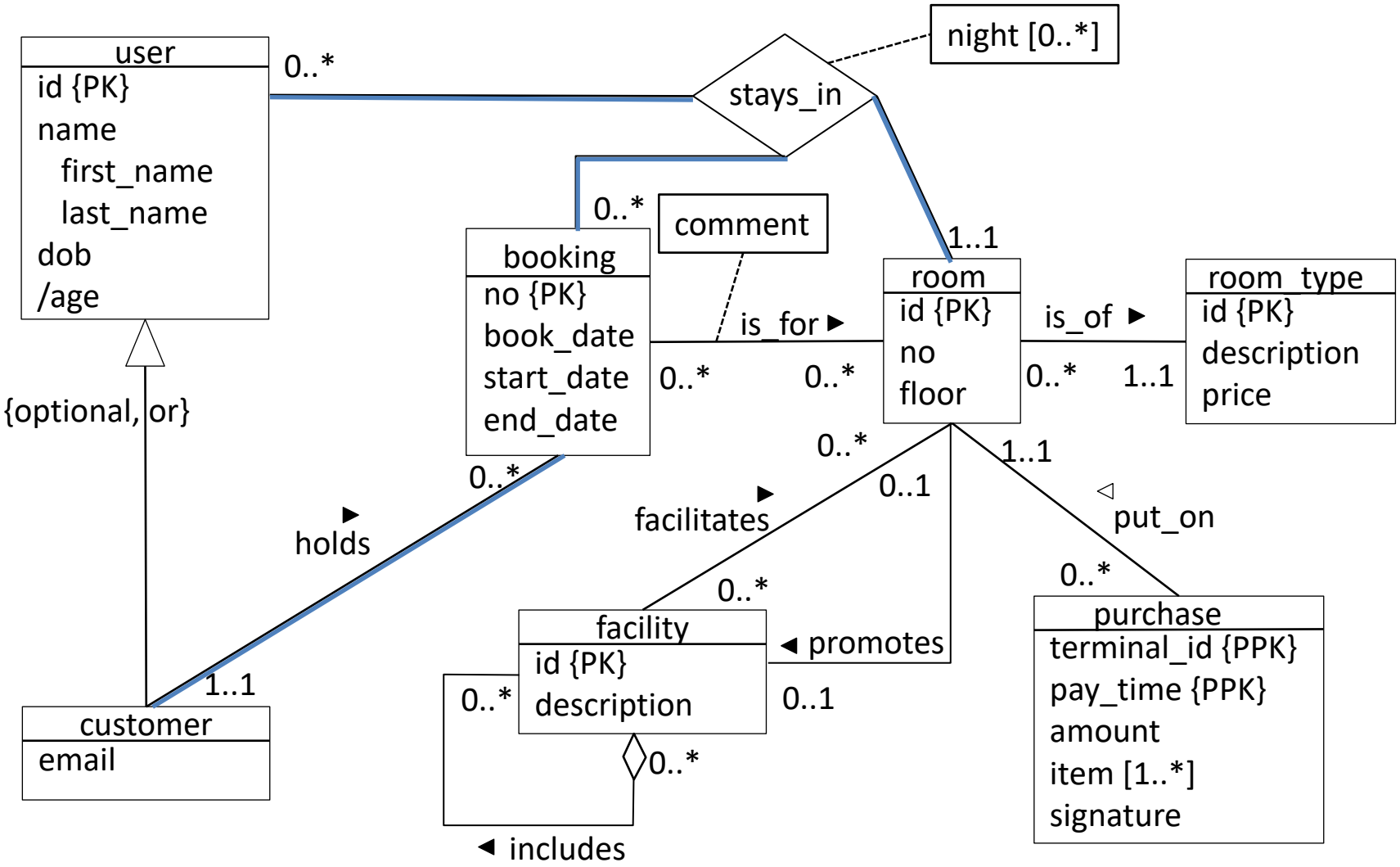
Hotel Booking Example



Validating the Example Model

- Sample queries:
  - What's the average purchase per room per day and room type for 2020?
  - Who were the guests stayed in room #331 on the nights of 21 - 23 December 2020 of a booking made by customer NN?
  - How many customers have booked rooms for the month of December where at least one of the rooms had a private sauna?

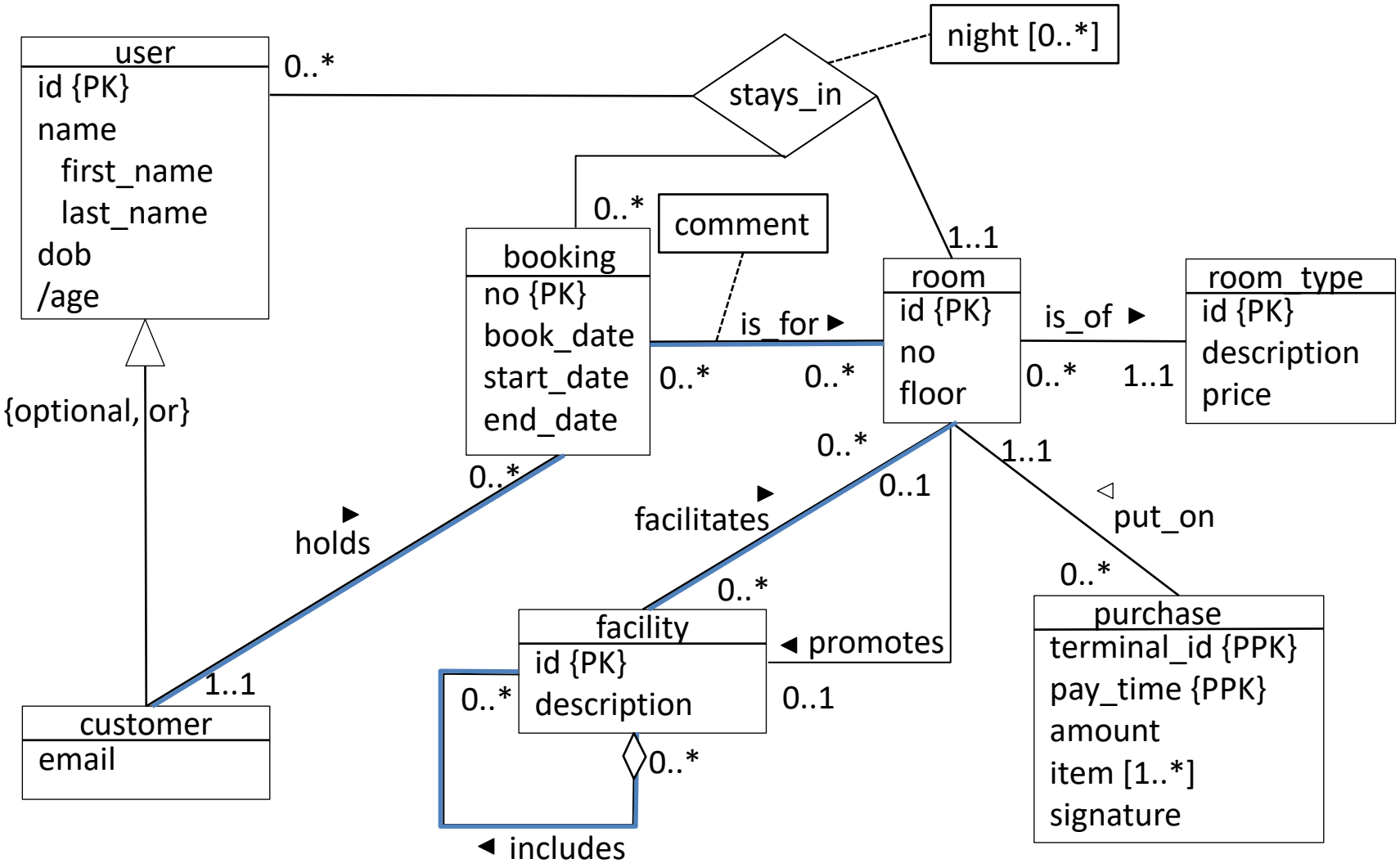
Hotel Booking Example



Validating the Example Model

- Sample queries:
  - What's the average purchase per room per day and room type for 2020?
  - Who were the guests stayed in room #331 on the nights of 21 - 23 December 2020 of a booking made by customer NN?
  - How many customers have booked rooms for the month of December where at least one of the rooms had a private sauna?

Hotel Booking Example



Reviewing Model with User

- The conceptual data model consisting of
  - ER diagram
  - the supporting documentation
- is to be "signed off" by customer
- May result in a number of revisions:
  - Assign date and revision numbers
  - Use a revision control system, such as Git