Inheritance: Comparison

Inheritance Mapping Strategies

Inheritance Mapping Strategies

- **Single table**
- **Table per class**
- **Joined table**
- **Mapped superclass**

Inheritance Mapping Strategies

- **Single table**
- **Table per class**
- **Joined table**
- **Mapped superclass**

Which one???

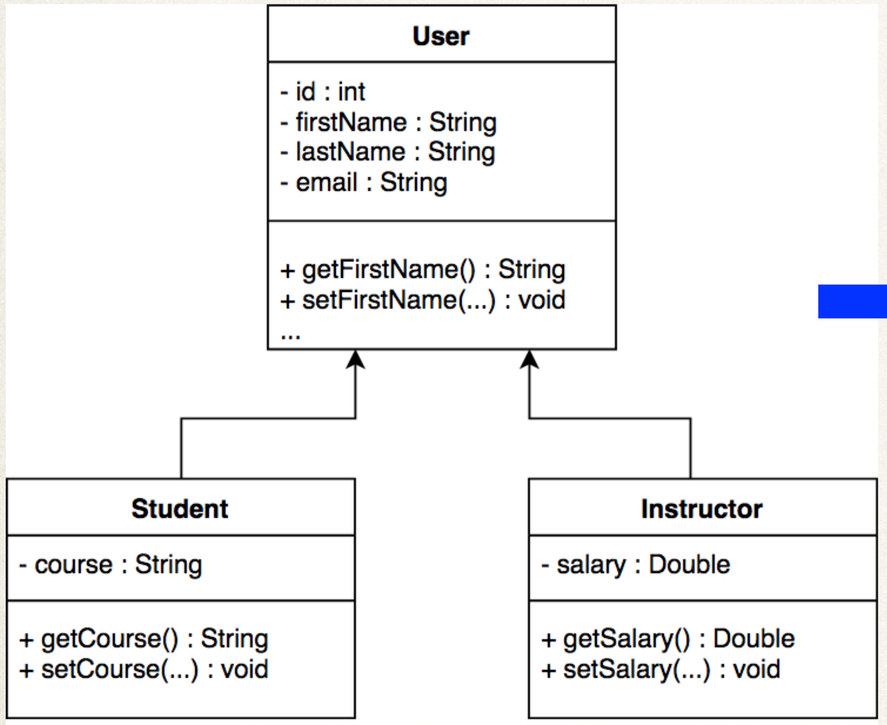
Inheritance Mapping Strategies

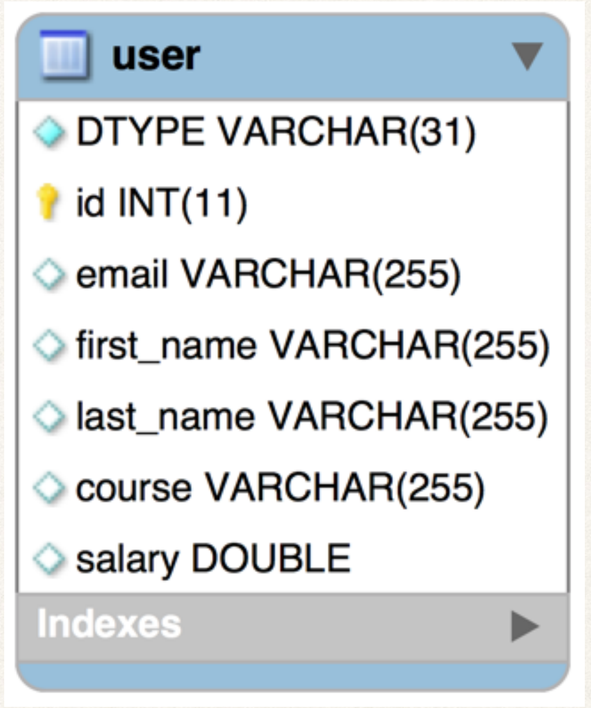
- **Single table**
- **Table per class**
- **Joined table**
- **Mapped superclass**

Quick Recap

Single Table

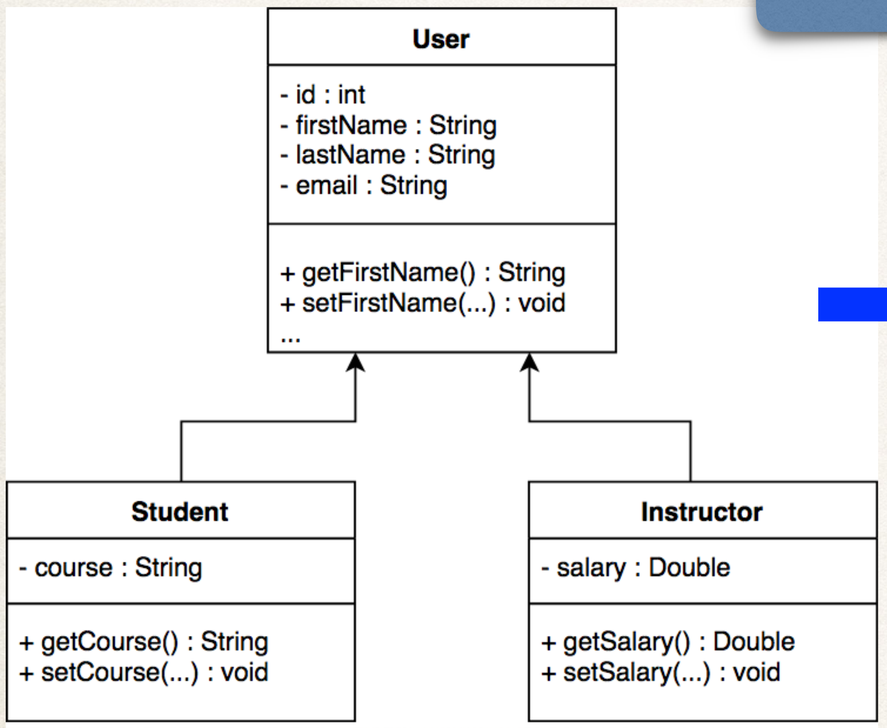
Single Table

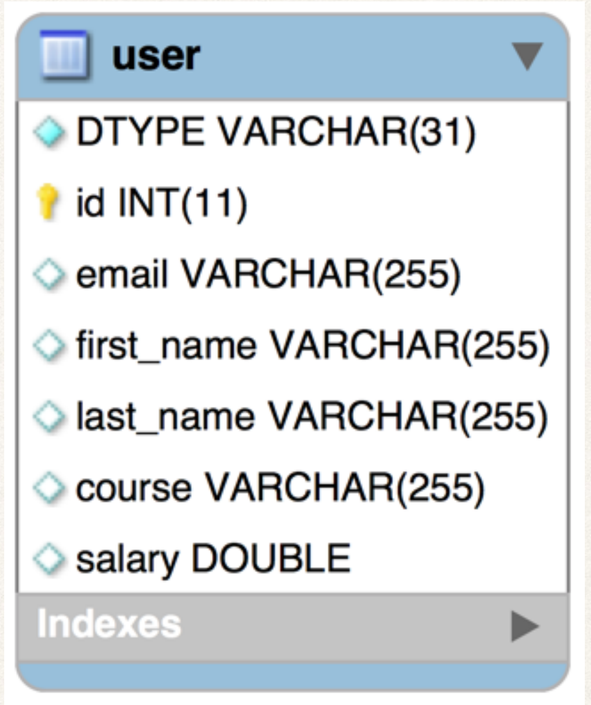




Single Table

Table has columns for all fields in inheritance tree





Single Table

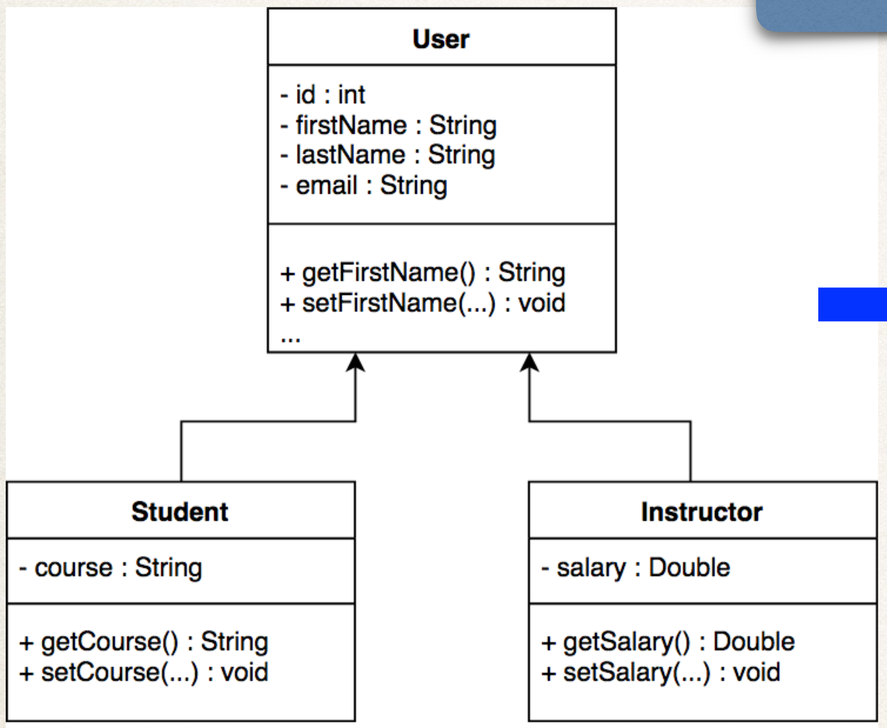
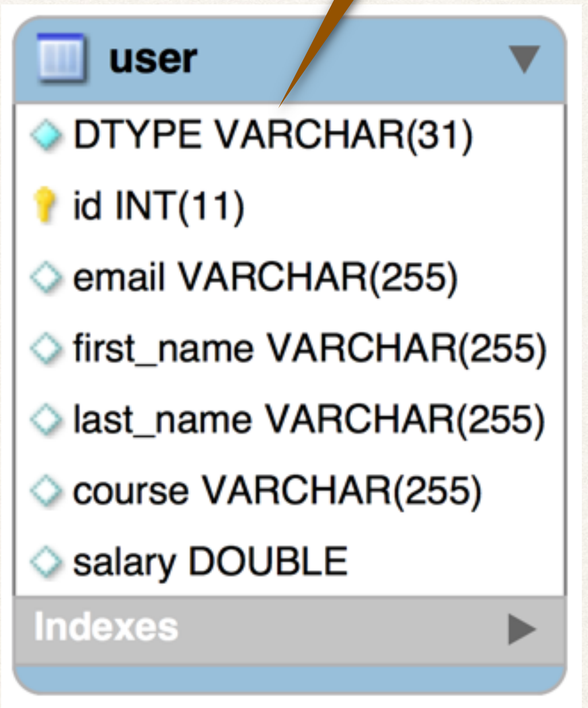
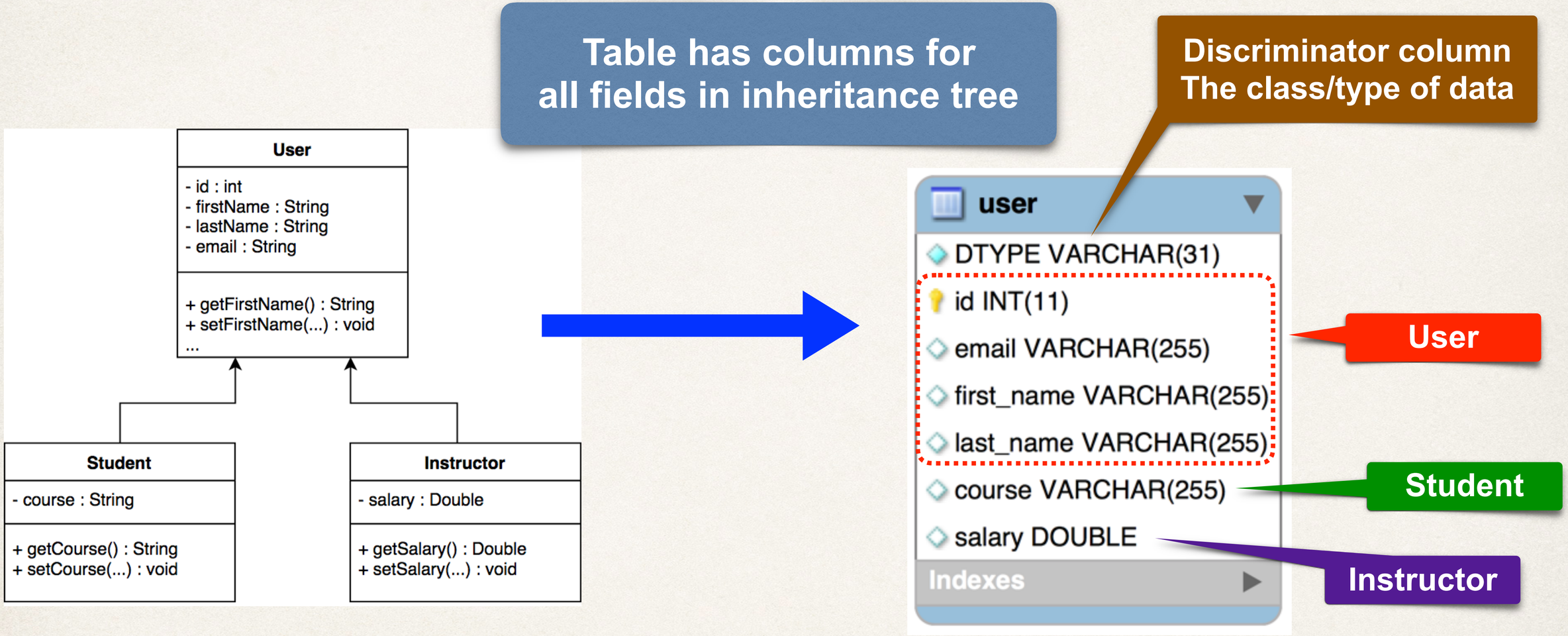


Table has columns for all fields in inheritance tree

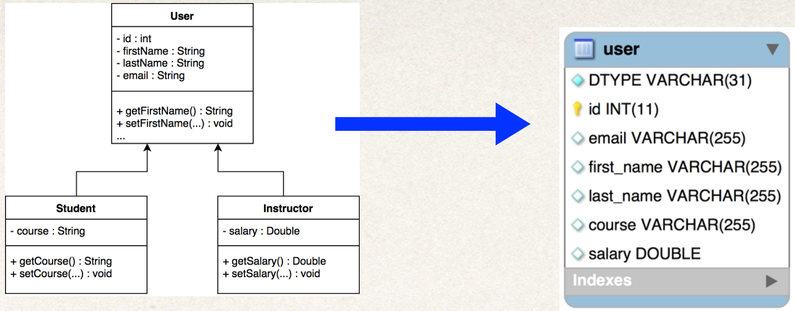
Discriminator column The class/type of data



Single Table

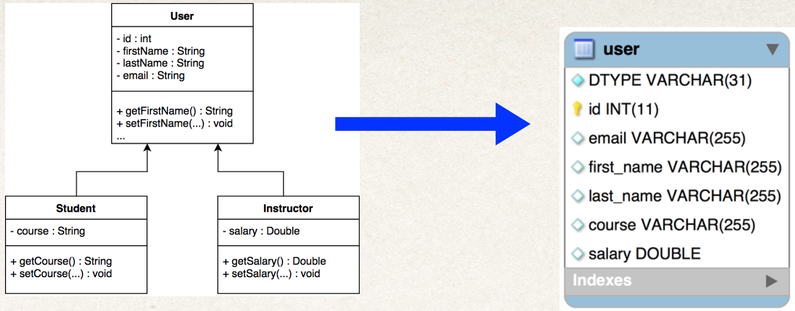


Single Table



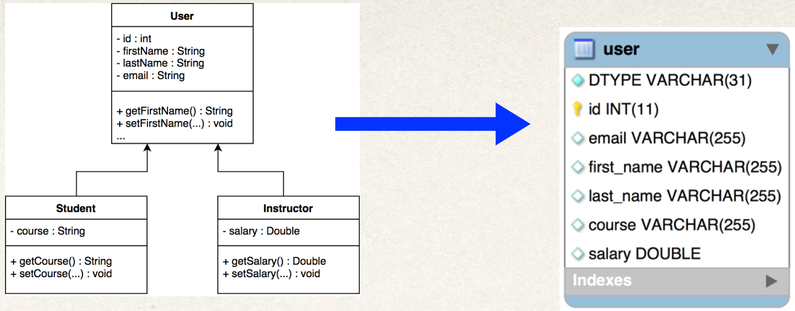
Single Table

- Advantages

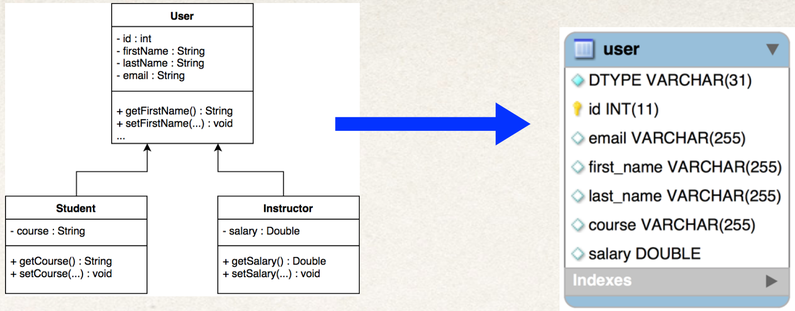


Single Table

- Advantages
  - Simple and straight-forward implementation

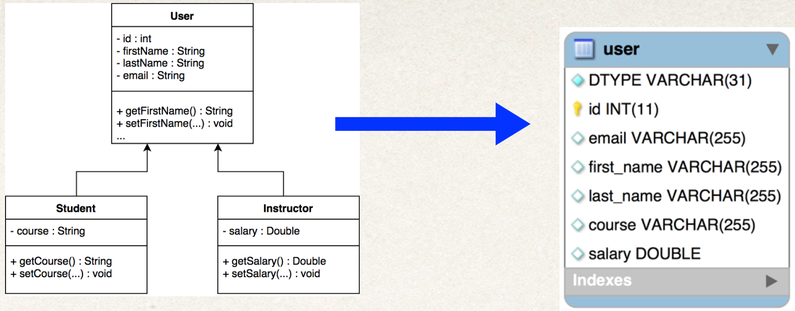


Single Table



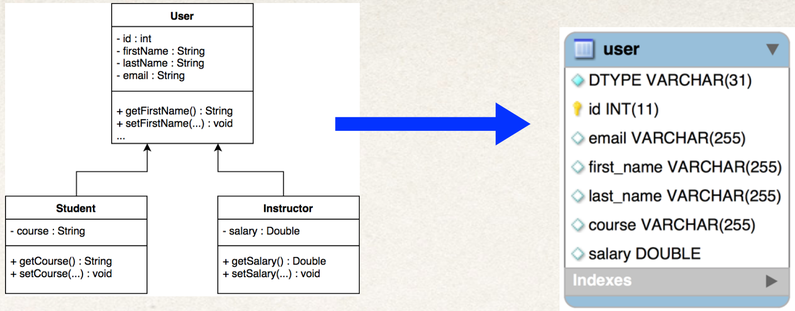
- Advantages
  - Simple and straight-forward implementation
  - Since data is in a single table, results in the best query performance

Single Table



- Advantages
  - Simple and straight-forward implementation
  - Since data is in a single table, results in the best query performance
- Disadvantages

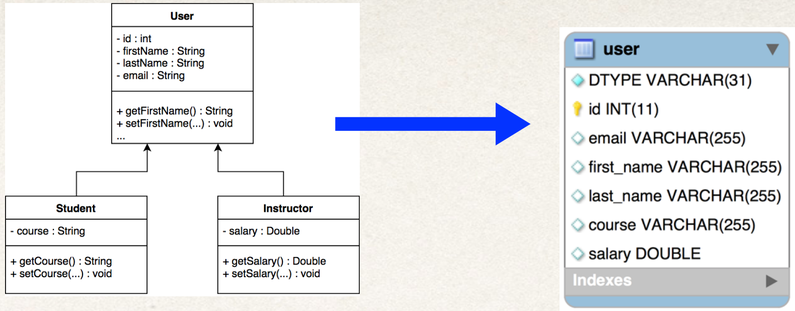
Single Table



- Advantages
  - Simple and straight-forward implementation
  - Since data is in a single table, results in the best query performance

- Disadvantages
  - Each row only uses a subset of fields and sets others to null

Single Table



- Advantages
  - Simple and straight-forward implementation
  - Since data is in a single table, results in the best query performance

- Disadvantages
  - Each row only uses a subset of fields and sets others to null
  - Since fields are nullable, this may present issues with data integrity

Table per Class

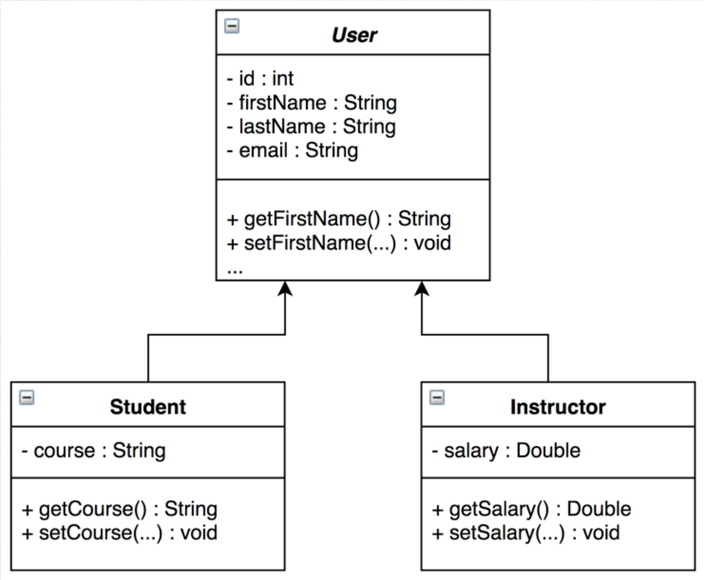


Table per Class

Table per class for the concrete classes

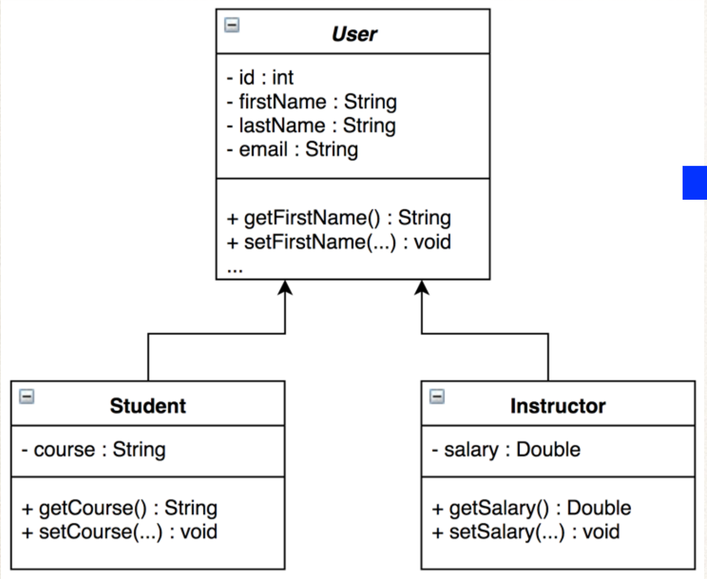


Table: Student

id	email	first_name	last_name	course
1	mary@luv2code.com	Mary	Public	Hibernate

User

Student

Table: Instructor

id	email	first_name	last_name	salary
2	john@luv2code.com	John	Doe	20000

User

Instructor

Table per Class

Table per class for the concrete classes

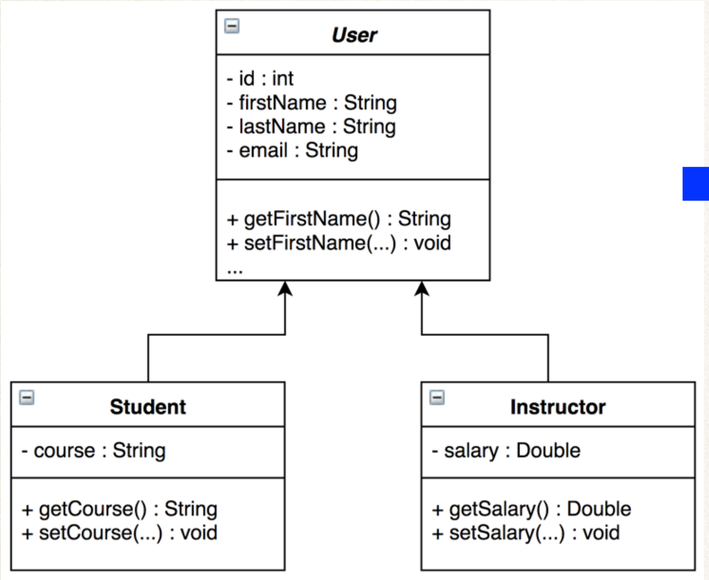


Table: Student

id	email	first_name	last_name	course
1	mary@luv2code.com	Mary	Public	Hibernate

User **Student**

Table: Instructor

id	email	first_name	last_name	salary
2	john@luv2code.com	John	Doe	20000

User **Instructor**

Note: Only the related fields
No extra columns

Table per Class

Table per class for the concrete classes

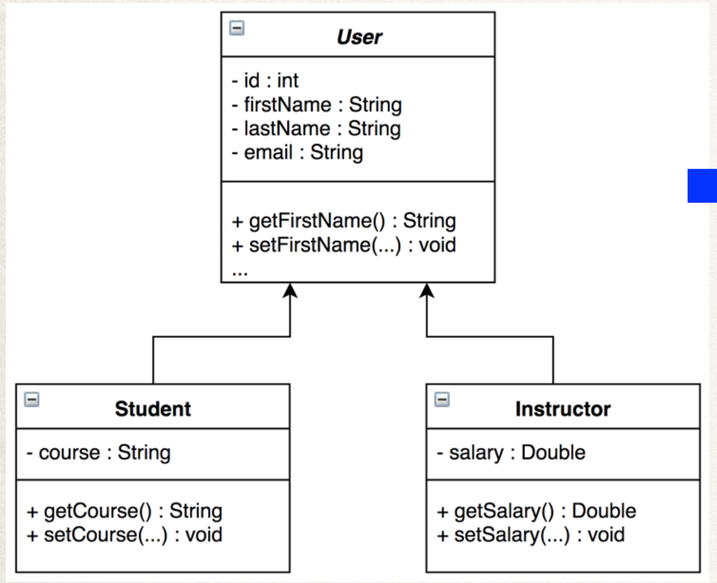


Table: Student

id	email	first_name	last_name	course
1	mary@luv2code.com	Mary	Public	Hibernate

User **Student**

Table: Instructor

id	email	first_name	last_name	salary
2	john@luv2code.com	John	Doe	20000

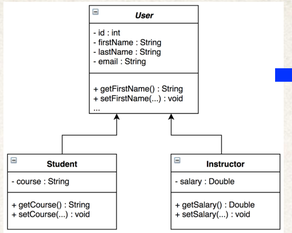
User **Instructor**

Note: Only the related fields
No extra columns

Required a "sequence table"

Table per Class

- Advantages



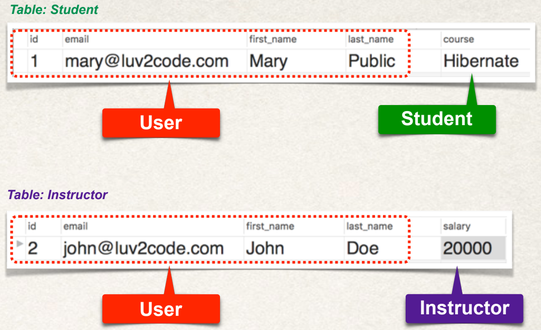
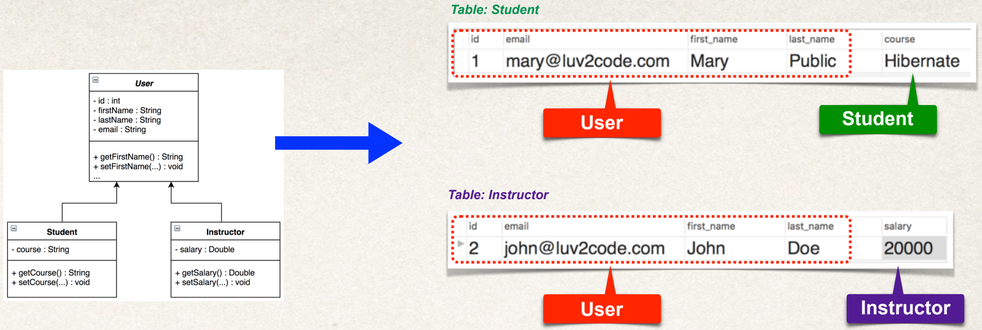
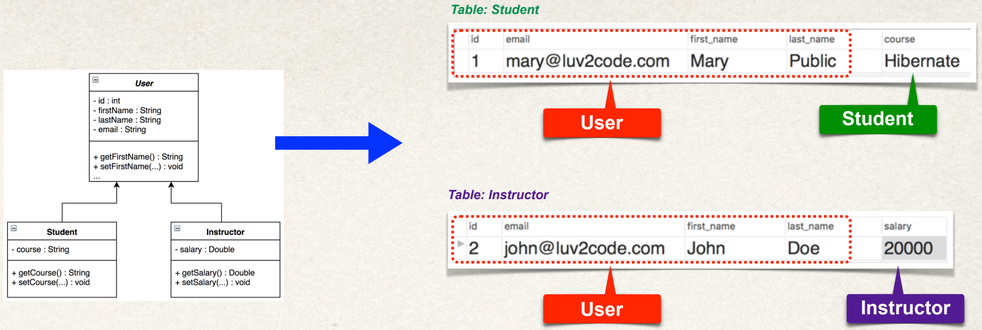


Table per Class



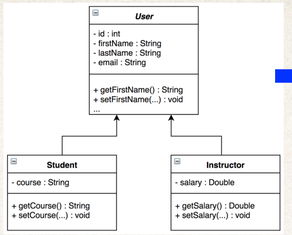
- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete sub-classes perform generally well

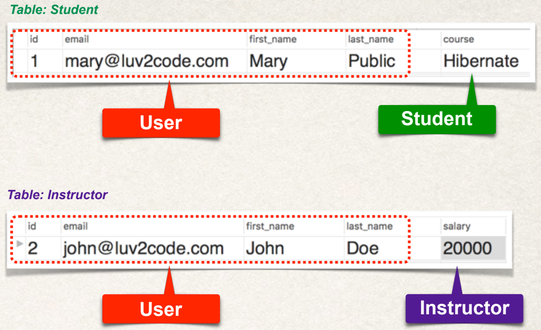
Table per Class



- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete sub-classes perform generally well
- Disadvantages

Table per Class

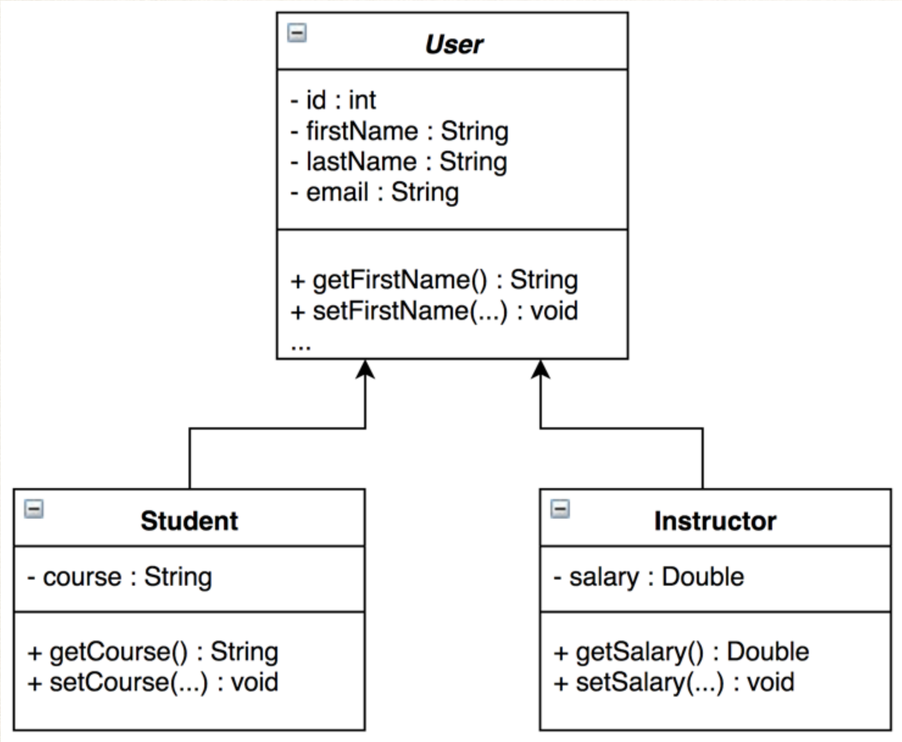




- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete sub-classes perform generally well

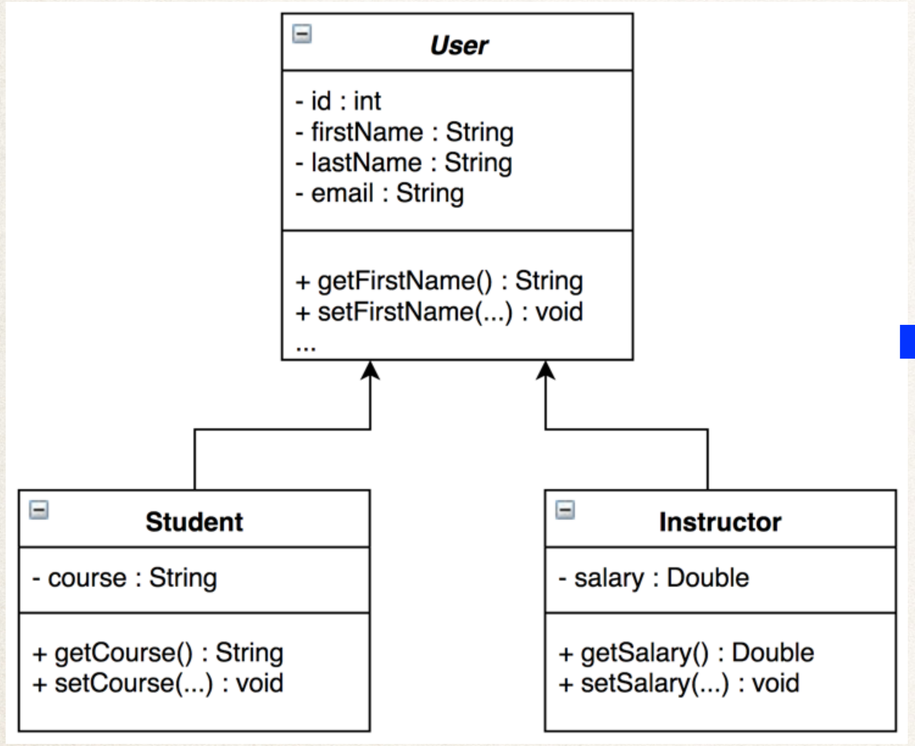
- Disadvantages
  - Slow performance for queries on superclasses (results in many joins)
  - Query performance slows down for very deep inheritance trees
  - ID generation with sequence tables in a high-volume multi-threaded environment is slow

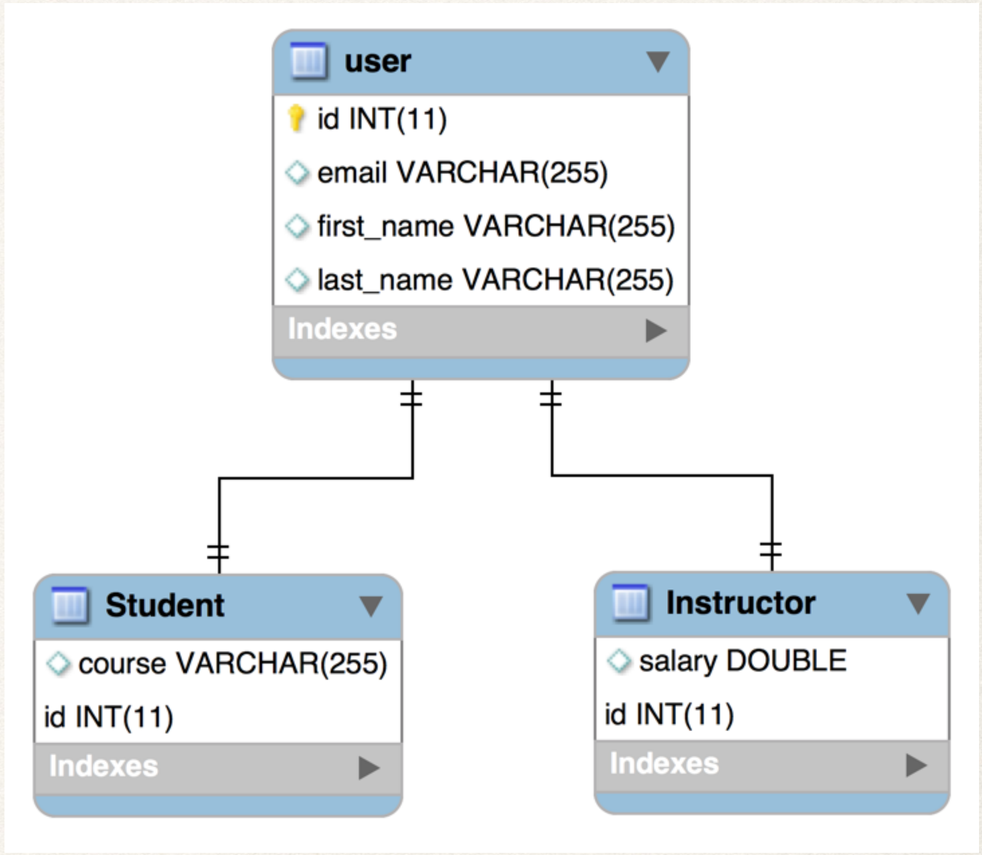
Joined Tables



Joined Tables

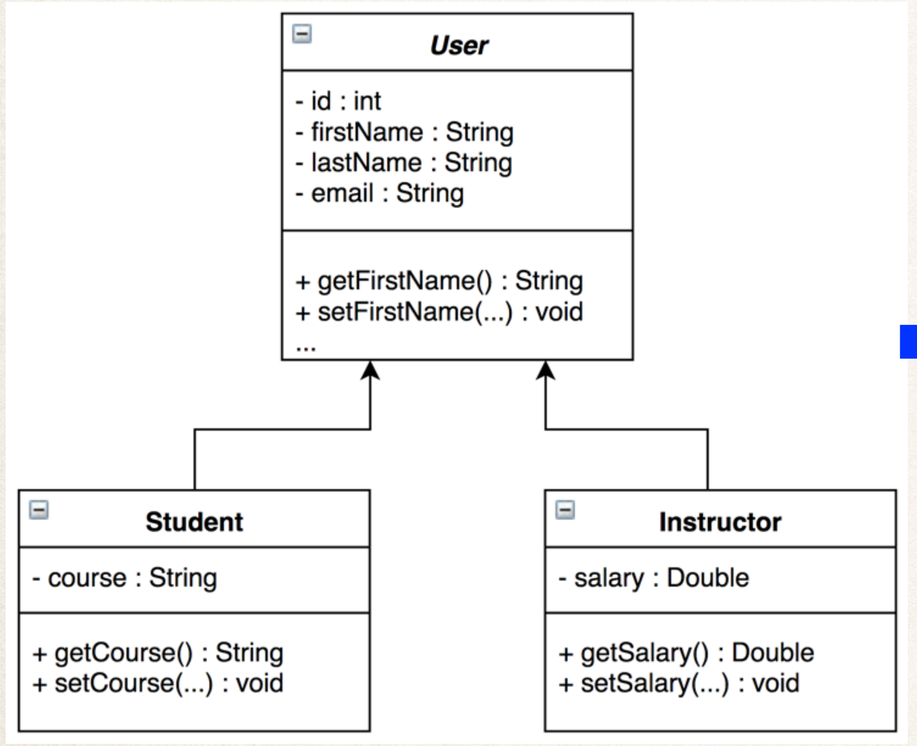
Inheritance is modeled by joining on primary and foreign keys

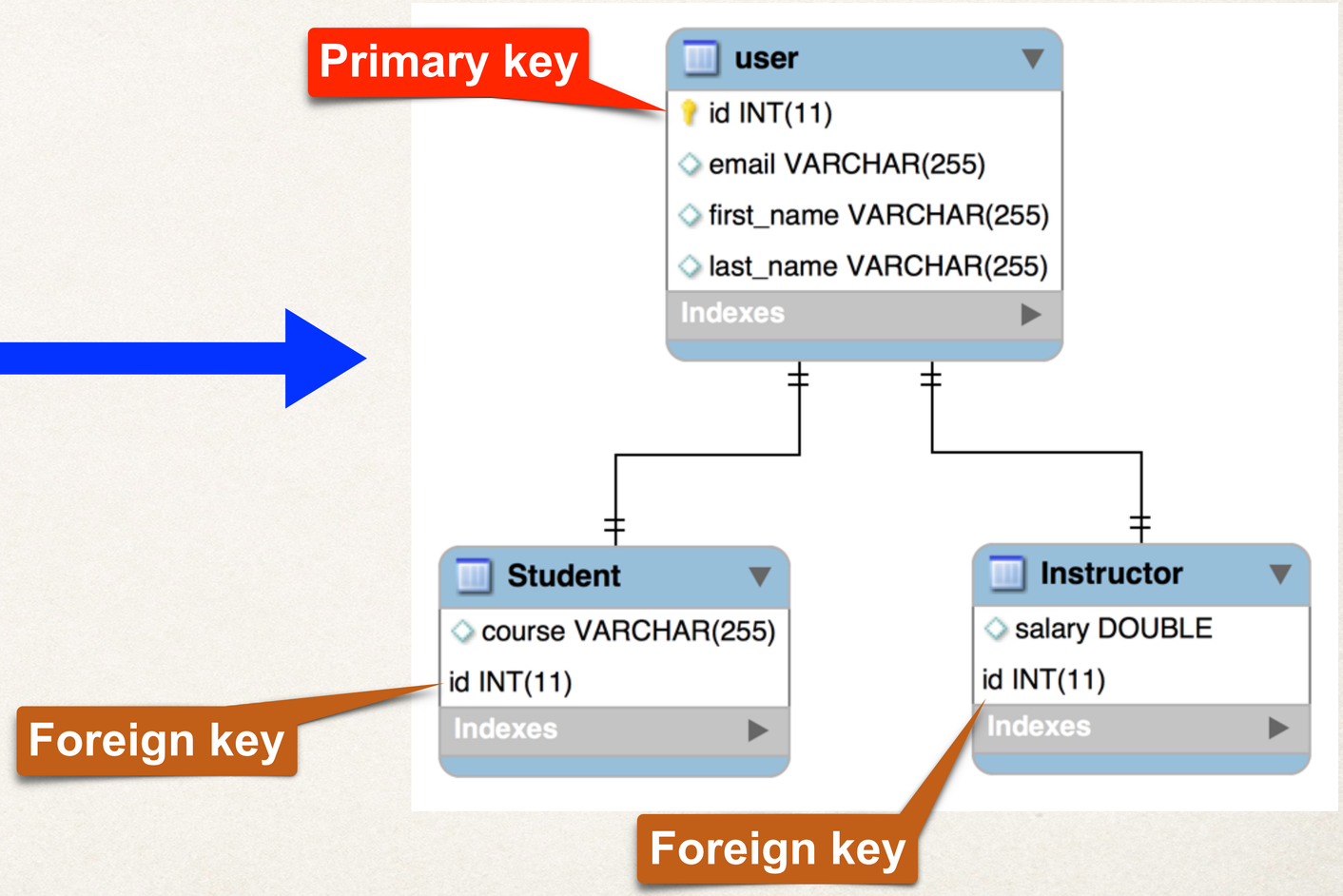




Joined Tables

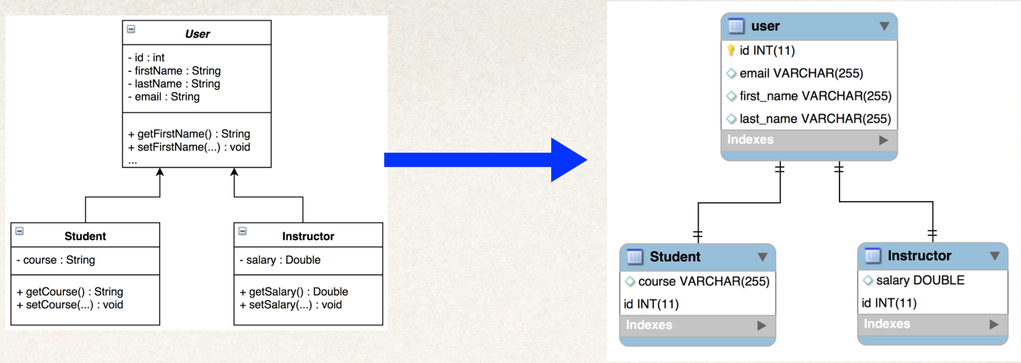
Inheritance is modeled by joining on primary and foreign keys





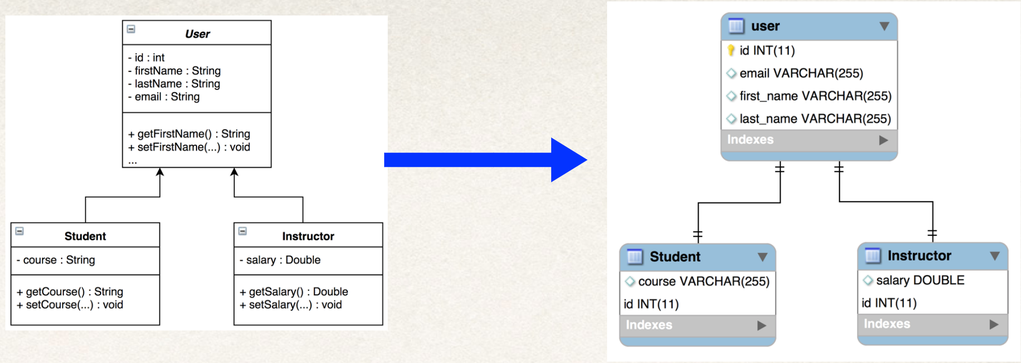
Joined Tables

- Advantages

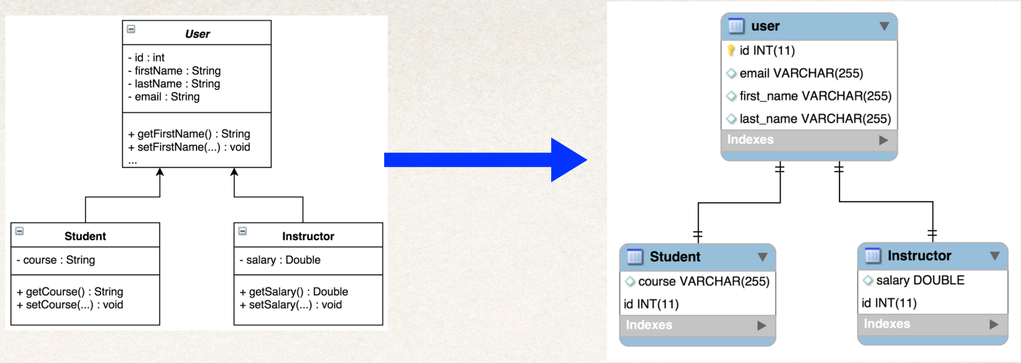


Joined Tables

- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields

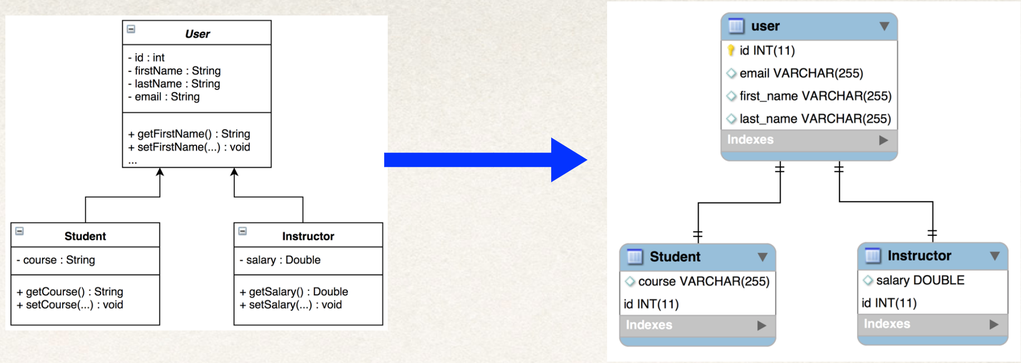


Joined Tables



- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields
- Disadvantages

Joined Tables

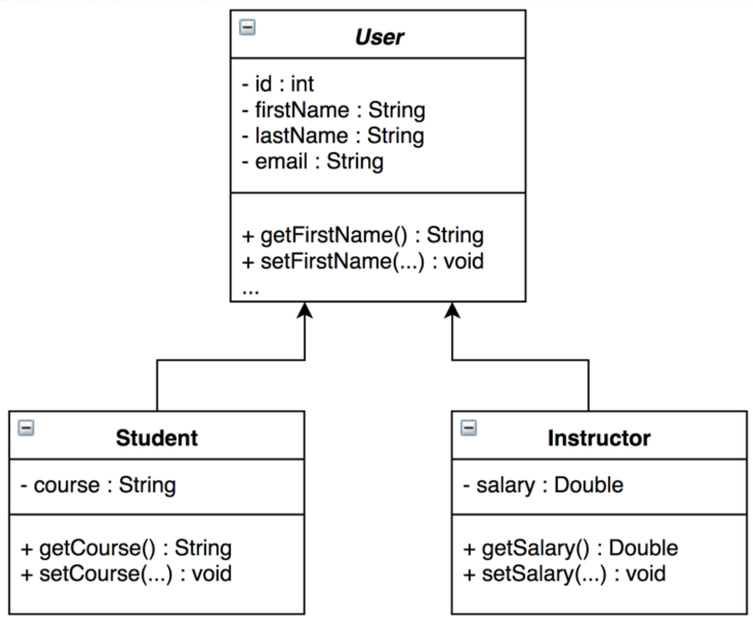


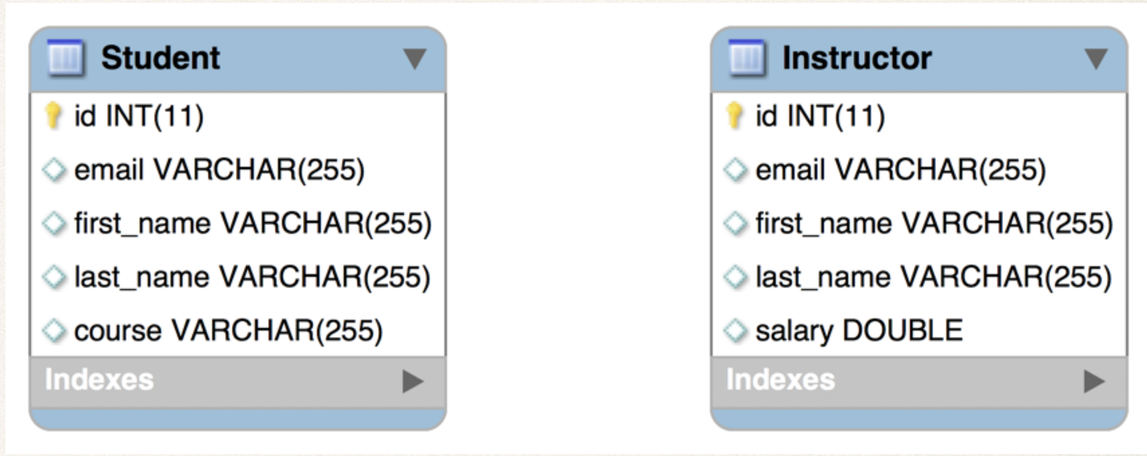
- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields

- Disadvantages
  - Slow performance for queries on subclasses (results in many joins)
  - Query performance slows down for very deep inheritance trees

Mapped Superclass

Mapped Superclass

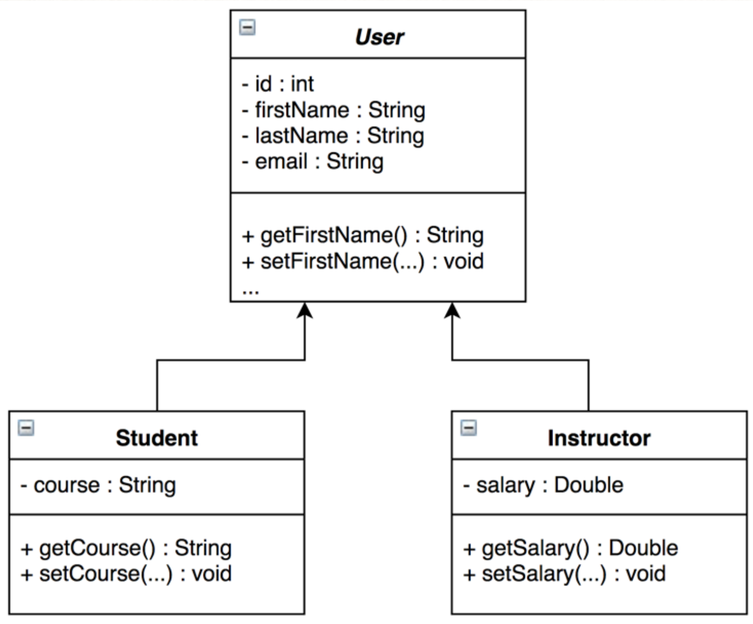


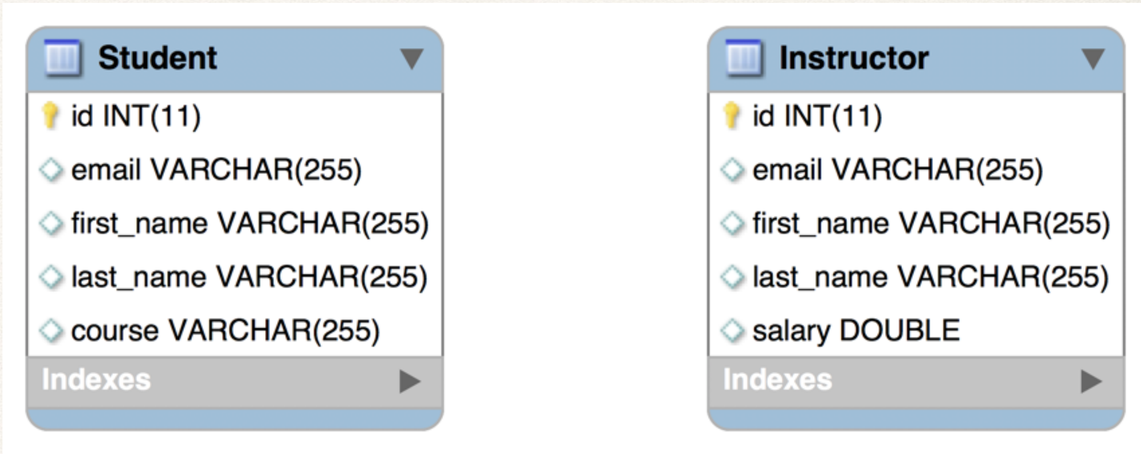


Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields

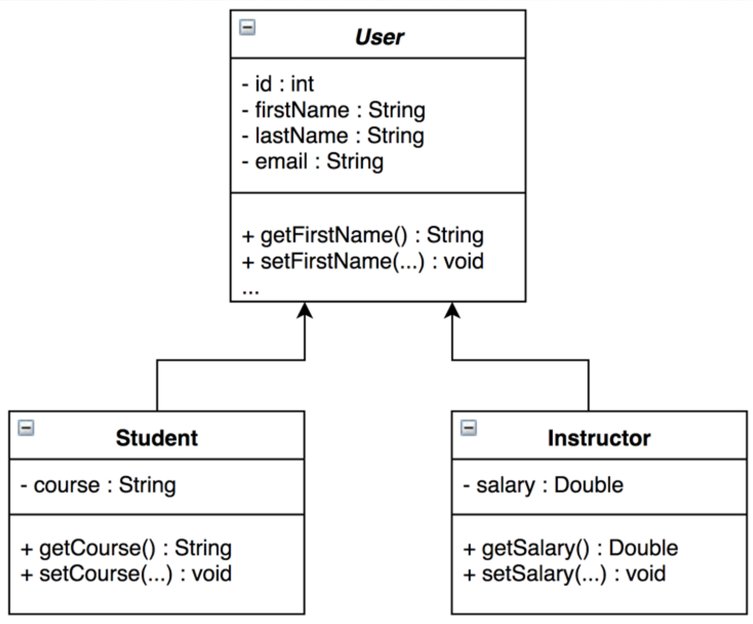


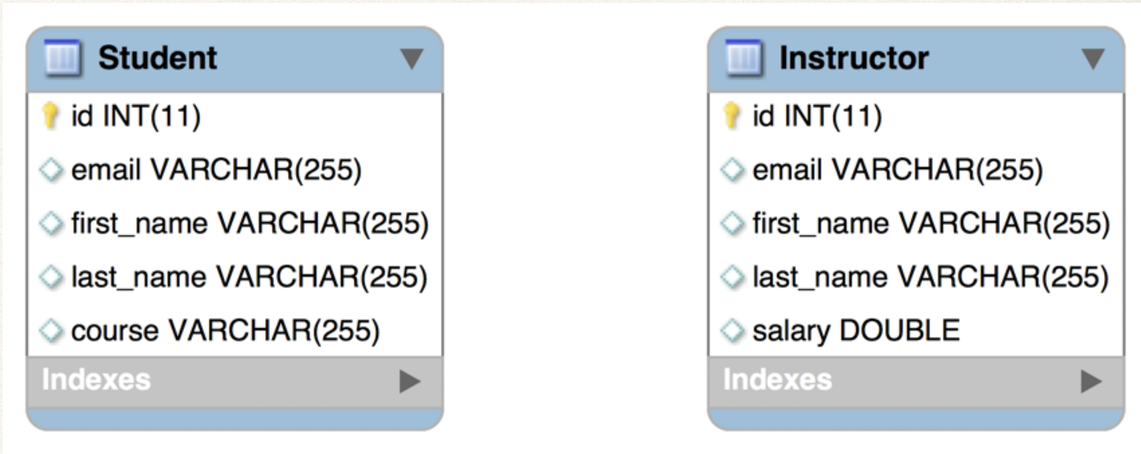


Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields



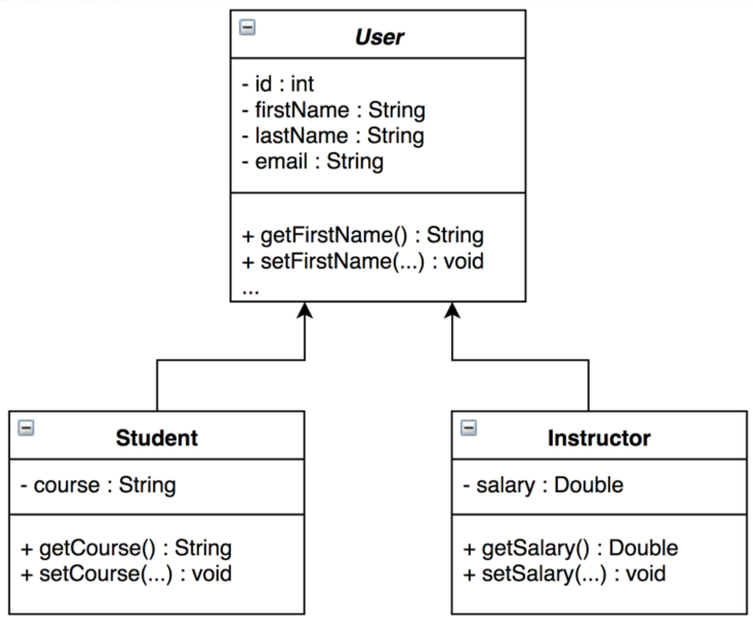


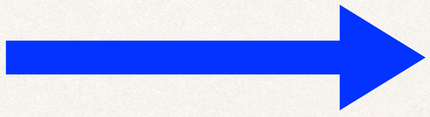
Only subclasses have tables in the database

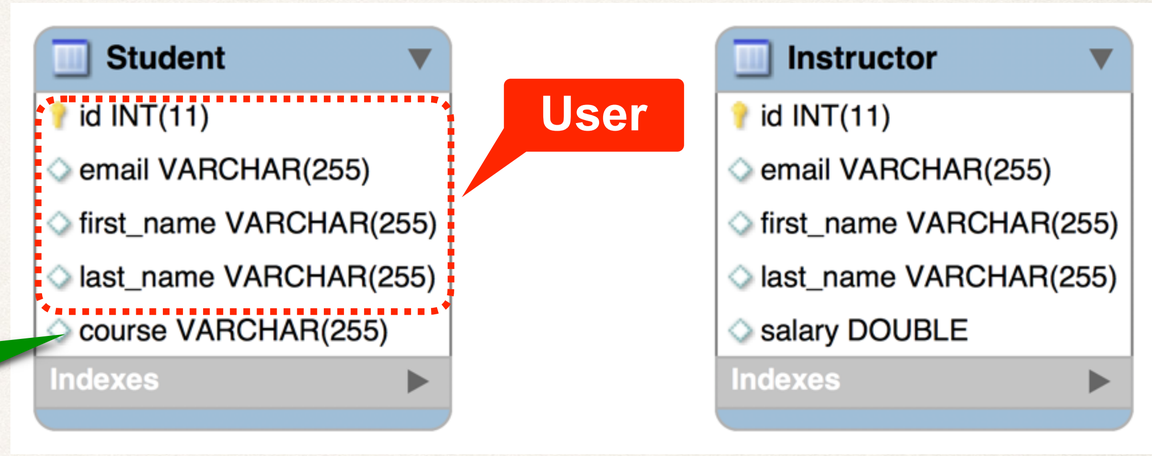
Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields





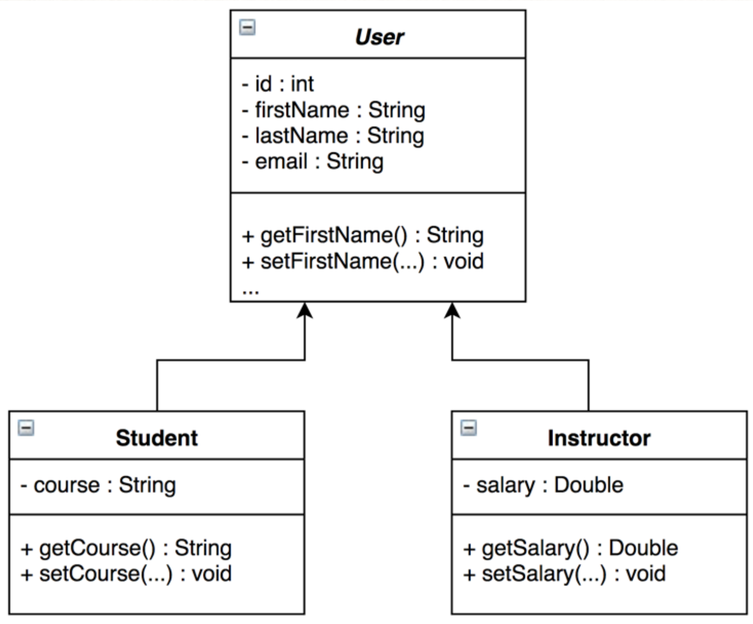


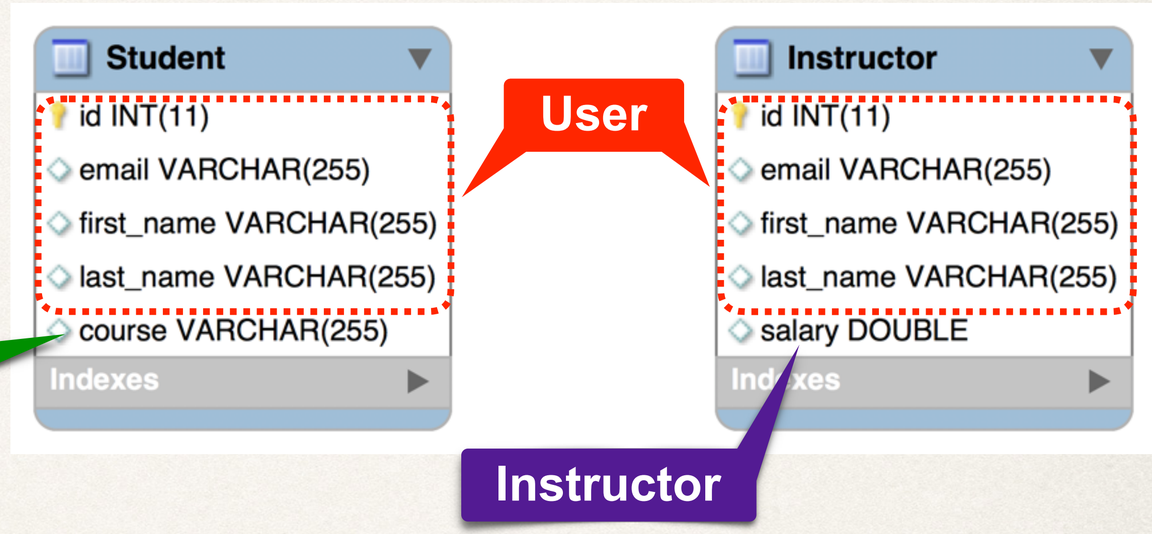
Only subclasses have tables in the database

Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields





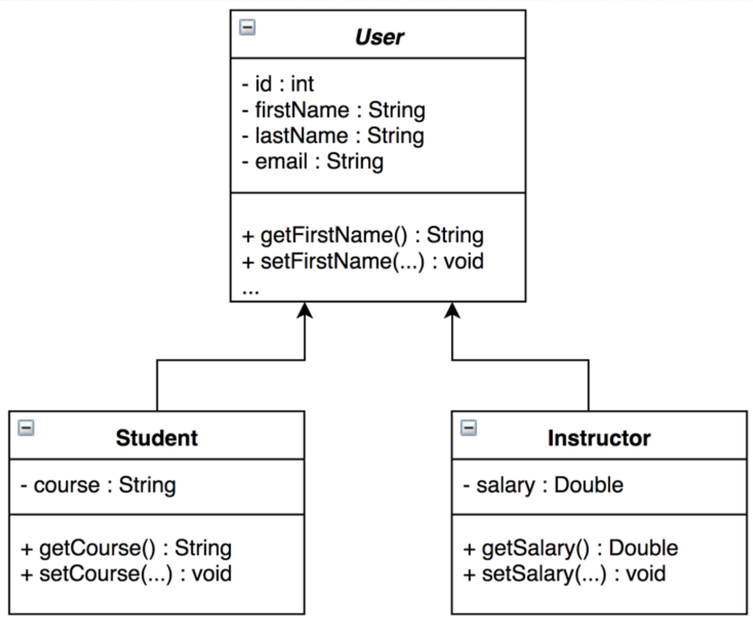
Only subclasses have tables in the database

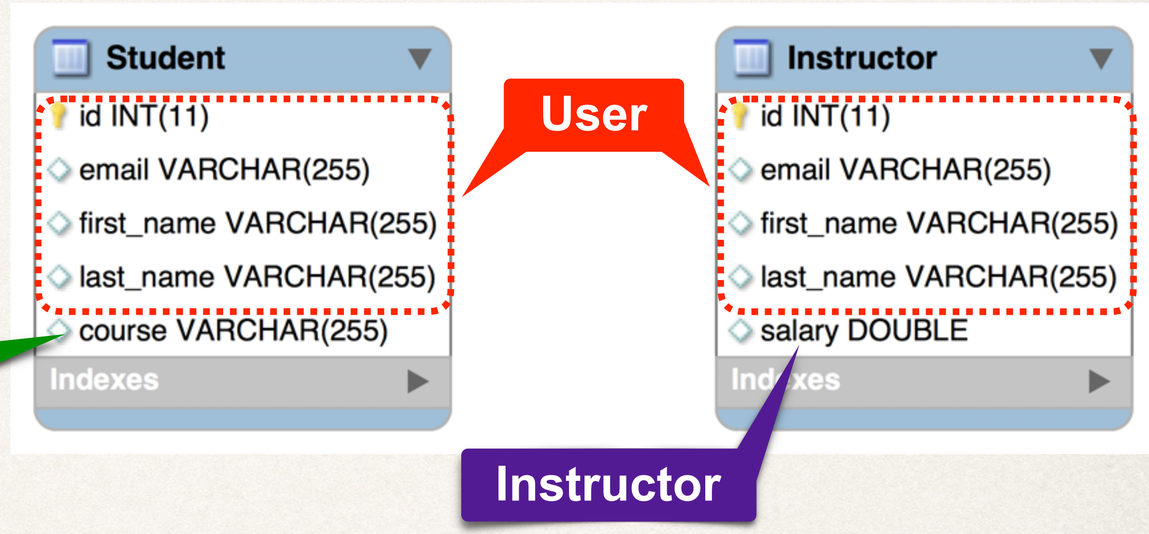
Mapped Superclass

Note: No table for superclass

Superclass defines common fields

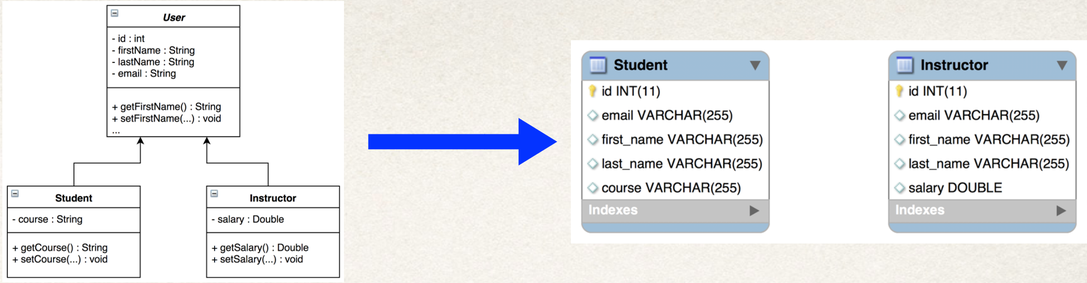
Subclass table contains superclass fields and subclass fields





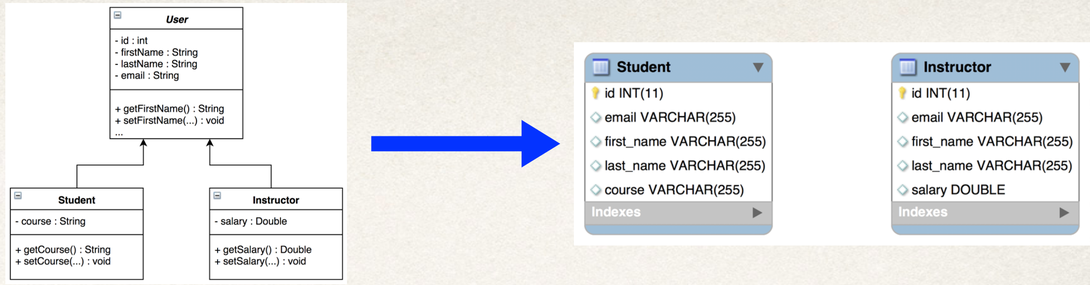
Only subclasses have tables in the database

Mapped Superclass



- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

Mapped Superclass



- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

- Disadvantages
  - For polymorphic queries, need to manually code HQL joins
  - For example, to query for all **`User`**s, need to join across **`Student`**, **`Instructor`** etc …
    - Requires manual HQL coding
    - May result in slower performance (results in many joins)