

# Informatics 1: Data & Analysis

## Lecture 4: From ER Diagrams to Relational Models

Ian Stark

School of Informatics  
The University of Edinburgh

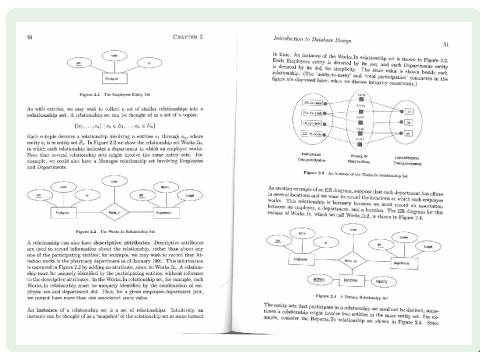
Friday 27 January 2016  
Semester 2 Week 2

# Homework from Tuesday

## Read This

Before the next lecture, on Friday, read the remaining sections, §§2.5 onwards, of Ramakrishnan and Gehrke, completing Chapter 2.

These consider trade-offs and choices in the design of Entity-Relationship models, as well as more on the wider context of modelling.



R. Ramakrishnan and J. Gehrke.  
*Database Management Systems*.  
McGraw-Hill, third edition, 2003.

## Data Representation

This first course section starts by presenting two common **data representation models**.

- The *entity-relationship (ER)* model
- The *relational* model

## Data Manipulation

This is followed by some methods for manipulating data in the relational model and using it to extract information.

- *Relational algebra*
- The *tuple-relational calculus*
- The query language *SQL*

# Homework

The tutorial worksheet was posted to the course website earlier this week. Download it, read the instructions, follow them.

The sheet has six questions where you progressively design and then draw an Entity-Relationship model for part of a database system.

There is flexibility in how you design the model, and there is more than one possible solution to the problem.

If you find parts difficult, or have questions about the exercise: ask other students; ask on [Piazza](#); or bring your problem to the tutorials.

Each tutorial worksheet also has several more example problems and notes on solving them.

Before the tutorial, work through the exercise sheet. Ask and answer questions on Piazza, talk to other students, write up your work.

Bring your solutions and your working. You will need to be able to show these to your tutor and exchange them with other students.

Come to tutorials properly prepared. Students who have not attempted the exercises will be sent away to spend the time working on them.

Data & Analysis tutorials and exercises are not given marks or grades, but their content is examinable and they are an important part of the course. If you do not do the exercises then you are unlikely to pass the exam.

Course participation requires attendance at tutorials: if you are ill or otherwise unable to attend one week then email your tutor, and if possible attend another tutorial group in the same week.

You can access many DICE services and Informatics resources remotely.

- Files over the web. <https://ifile.inf.ed.ac.uk>
- Command line. `ssh student.ssh.inf.ed.ac.uk` then `ssh student.login`  
(On Microsoft Windows, use PuTTY to reach `student.ssh.inf.ed.ac.uk`)
- Desktop. Graphical login with NX to [nx.inf.ed.ac.uk](http://nx.inf.ed.ac.uk)

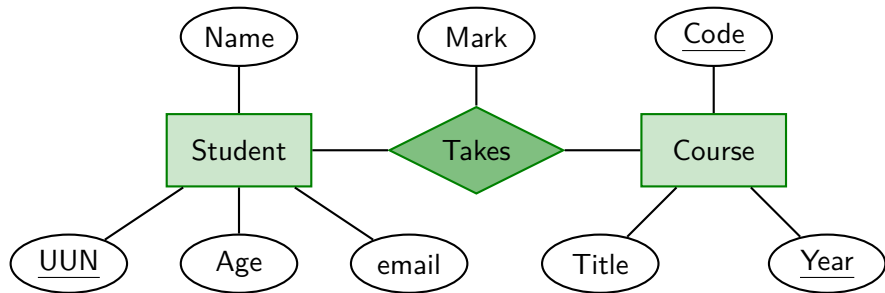
You can also use *Virtual DICE*, tunnel X Windows, access files over AFS, and connect by VPN to internal networks at Informatics and the University.

<http://computing.help.inf.ed.ac.uk>

# The Story so Far

## Entity-Relationship diagrams

- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship



# The Story so Far

## Entity-Relationship diagrams

- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship
- Key constraints as arrows





# The Story so Far

## Entity-Relationship diagrams

- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship
- Key constraints as arrows; total participation as thick/double lines



# The Story so Far

## Entity-Relationship diagrams

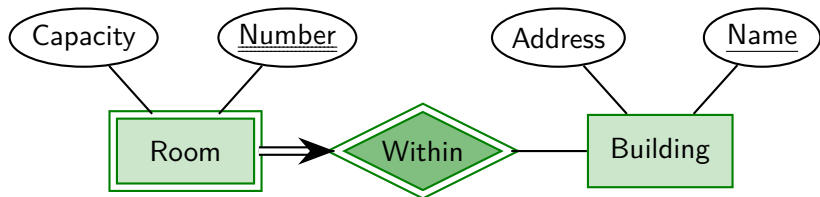
- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship
- Key constraints as arrows; total participation as thick/double lines



# The Story so Far

## Entity-Relationship diagrams

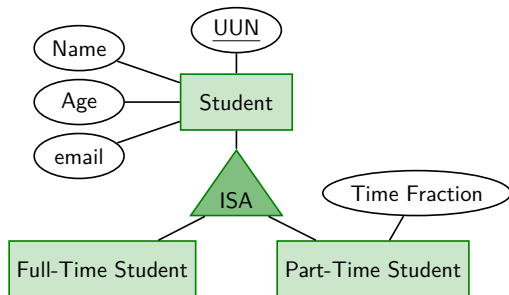
- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship
- Key constraints as arrows; total participation as thick/double lines
- Weak entities with their identifying relationship



# The Story so Far

## Entity-Relationship diagrams

- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship
- Key constraints as arrows; total participation as thick/double lines
- Weak entities with their identifying relationship; Entity hierarchies



# The Story so Far

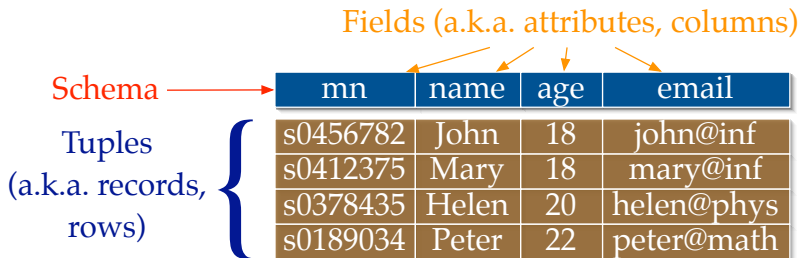
## Entity-Relationship diagrams

- Entities: Rectangles
- Relationships: Diamonds linked to the entities
- Attributes: Ovals linked to entity or relationship
- Key constraints as arrows; total participation as thick/double lines
- Weak entities with their identifying relationship; Entity hierarchies

## Relational models

- Relations: Tables matching schemas
- Schema: A set of field names and their domains
- Table: A set of tuples of values matching these fields
- Primary key: Chosen uniquely-valued field or set of fields
- Foreign key: Must be drawn from key values in another table

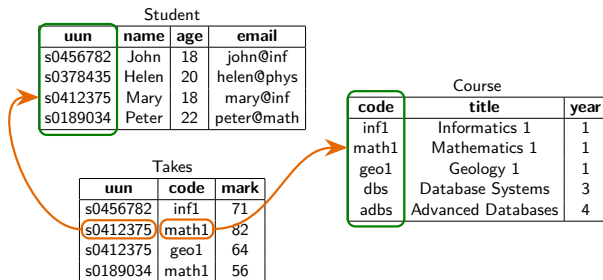
# The Story so Far



## Relational models

- Relations: Tables matching schemas
- Schema: A set of field names and their domains
- Table: A set of tuples of values matching these fields
- Primary key: Chosen uniquely-valued field or set of fields
- Foreign key: Must be drawn from key values in another table

# The Story so Far



## Relational models

- Relations: Tables matching schemas
- Schema: A set of field names and their domains
- Table: A set of tuples of values matching these fields
- Primary key: Chosen uniquely-valued field or set of fields
- Foreign key: Must be drawn from key values in another table

# The Story so Far

```
CREATE TABLE Takes (  
    uun    VARCHAR(8),  
    code   VARCHAR(20),  
    mark   INTEGER,  
    PRIMARY KEY (uun, code),  
    FOREIGN KEY (uun) REFERENCES Student,  
    FOREIGN KEY (code) REFERENCES Course )
```

## Relational models

- Relations: Tables matching schemas
- Schema: A set of field names and their domains
- Table: A set of tuples of values matching these fields
- Primary key: Chosen uniquely-valued field or set of fields
- Foreign key: Must be drawn from key values in another table



# Aim and Scale

The University of Edinburgh has 36 491 matriculated students studying 7 675 different courses across 20 different schools in 731 buildings.

With a relational database we can ask, and answer, questions like:

- What are the names and email addresses of all students taking a first-year course taught by the School of Informatics?
- Which lectures have been scheduled for rooms which will be at greater than 90% of their seating capacity?

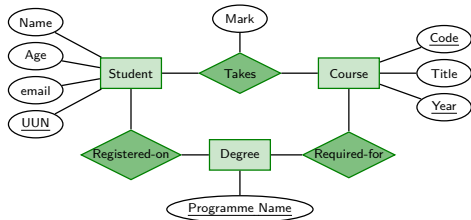
Many databases are much, much larger than this with much more complex queries. However, the aim is the same: how to go beyond answering a single query with a single program to a general system in which many queries can be expressed and answered efficiently.

# Translating ER Diagrams to Relational Models

An ER diagram captures a conceptual model of the data to be managed in a database: what there is and how it is connected.

We can use this as a basis for a relational schema, as a step towards implementation in a working RDBMS.

This translation will be approximate: some constraints expressed in an ER diagram might not naturally fit into relational schemas.



**CREATE TABLE** Student ( ... )

**CREATE TABLE** Takes ( ... )

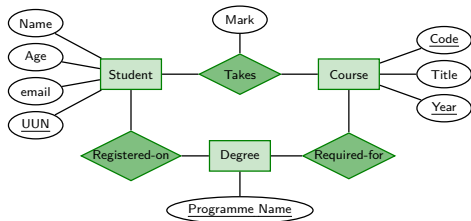
**CREATE TABLE** Course ( ... )

*etc.*

# Translating ER Diagrams to Relational Models

There may be more than one possible translation: different alternatives lead to different implementations. These may be efficiency trade-offs, for which we might go back to the requirements to assess their relative impact.

It is possible to make these translations complete (work for any diagram) and automatic (in a push-button tool); but here we shall just consider a few examples illustrating some of the main ideas.



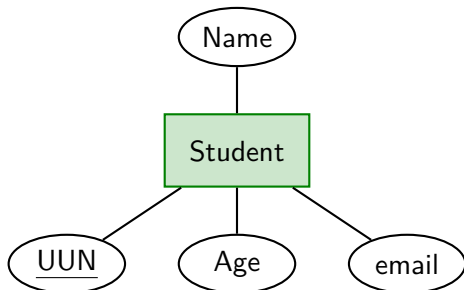
**CREATE TABLE** Student ( ... )

**CREATE TABLE** Takes ( ... )

**CREATE TABLE** Course ( ... )

*etc.*

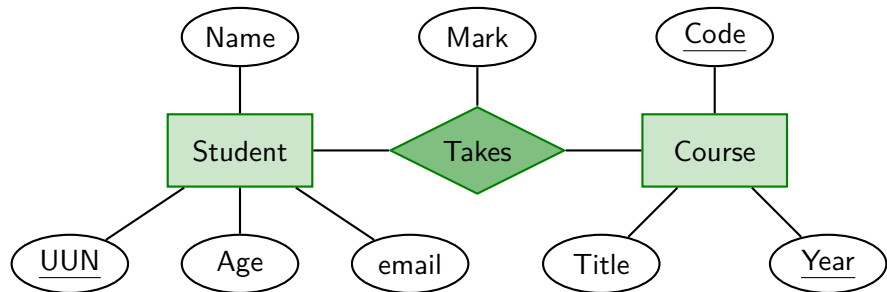
# Mapping an Entity



- Create a table for the entity.
- Make each attribute of the entity a field of the table, with an appropriate type.
- Declare the field or fields which make up the primary key

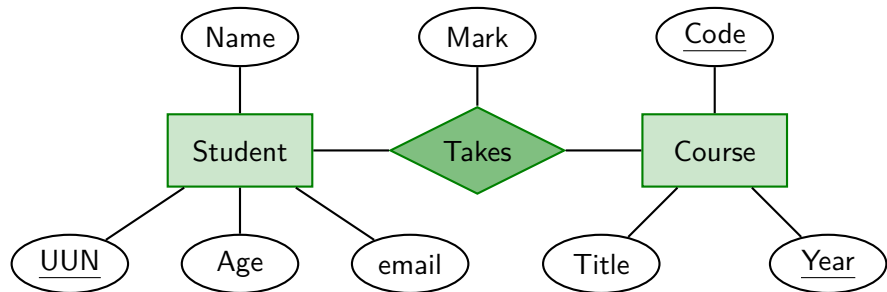
```
CREATE TABLE Student (  
    uun VARCHAR(8),  
    name VARCHAR(20),  
    age INTEGER,  
    email VARCHAR(25),  
    PRIMARY KEY (uun) )
```

# Mapping a Relationship



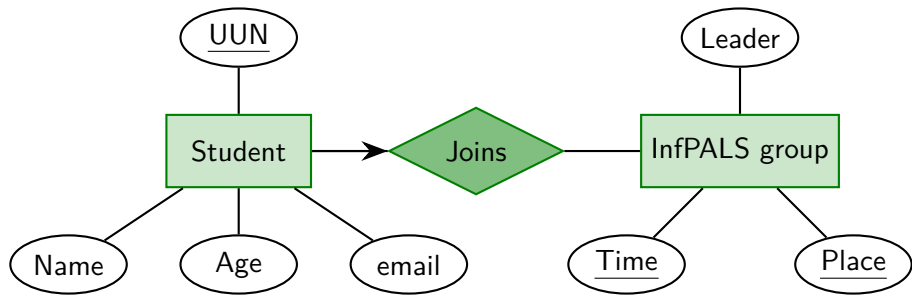
- Create tables for each entity, as before.
- Create a table for the relationship.
- Add all key attributes of all participating entities as fields.
- Add fields for each attribute of the relationship.
- Declare primary key using all key attributes from participating entities.
- Declare foreign key constraints for all these entity attributes.

# Mapping a Relationship



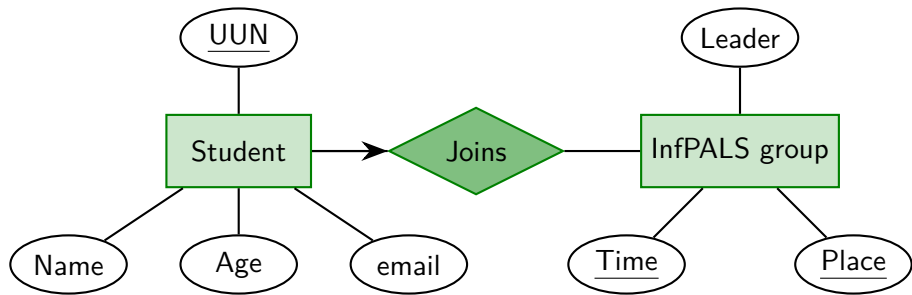
```
CREATE TABLE Takes (  
  uun VARCHAR(8), year INTEGER,  
  code VARCHAR(20), mark INTEGER,  
  PRIMARY KEY (uun, code, year),  
  FOREIGN KEY (uun) REFERENCES Student,  
  FOREIGN KEY (code,year) REFERENCES Course )
```

## Mapping Key Constraints, Method 1



- Create tables for each entity, as before.
- Create a table for the relationship.
- Add all key attributes of all participating entities as fields.
- Add fields for each attribute of the relationship.
- Declare primary key using only key attributes of the source entity.
- Declare foreign key constraints on key attributes of all entities.

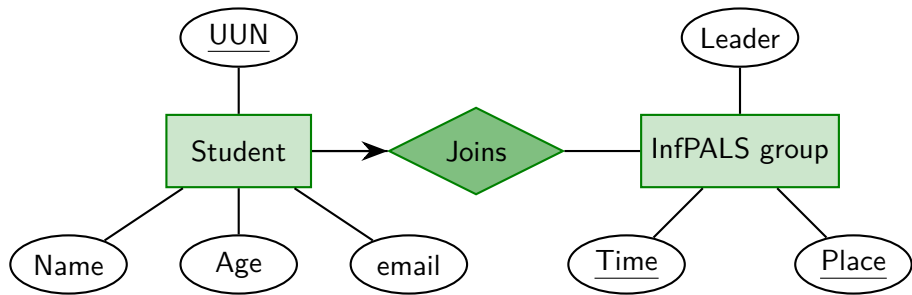
# Mapping Key Constraints, Method 1



```
CREATE TABLE Joins (  
  uun  VARCHAR(8),  
  time VARCHAR(20),  
  place VARCHAR(32),  
  PRIMARY KEY (uun),  
  FOREIGN KEY (uun) REFERENCES Student,  
  FOREIGN KEY (time,place) REFERENCES InfPALSGroup )
```



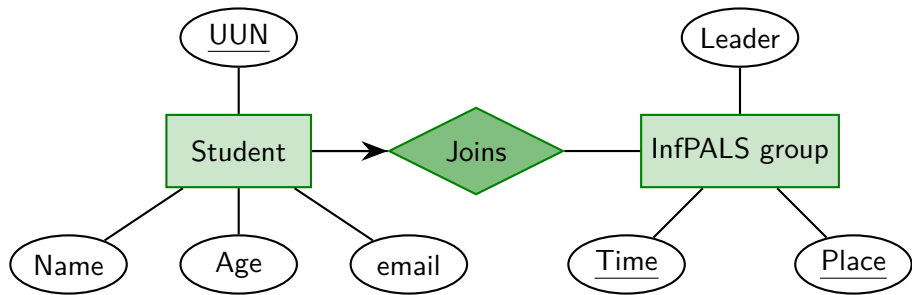
## Mapping Key Constraints, Method 2



In fact, because the **Joins** relationship is many-to-one, we don't need a whole table for the relation itself: information about InfPALS group membership can be added into the Student table.

- Create a table for the source and target entities as usual.
- Add every key attribute of the target as a field in the source table.
- Declare these fields as foreign keys.

## Mapping Key Constraints, Method 2



```
CREATE TABLE Student (  
  uun  VARCHAR(8),  age  INTEGER,  
  name VARCHAR(20), email VARCHAR(25),  
  time VARCHAR(10), place VARCHAR(32),  
  PRIMARY KEY (uun),  
  FOREIGN KEY (time,place) REFERENCES InfPALSGroup )
```

# Null Values

A field in SQL can have the special value **NULL**.

**NULL** can mean many things: that a field is unknown, or missing, or unavailable; or that this field may not apply in certain situations.

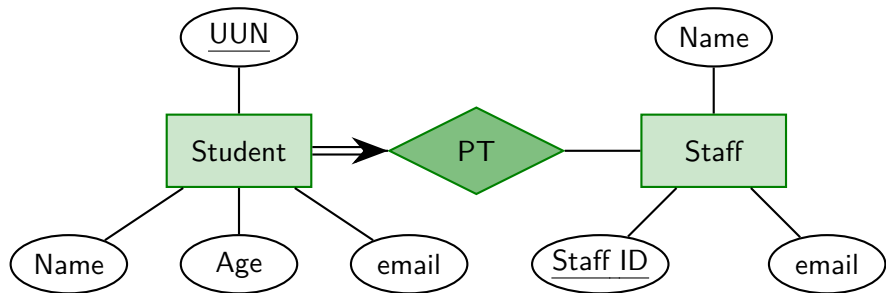
Some RDBMS forbid **NULL** from appearing in any field declared as a primary key.

Some of these may still allow **NULL** to appear in foreign key fields.

A schema can state that certain fields may not contain **NULL** using the **NOT NULL** declaration.

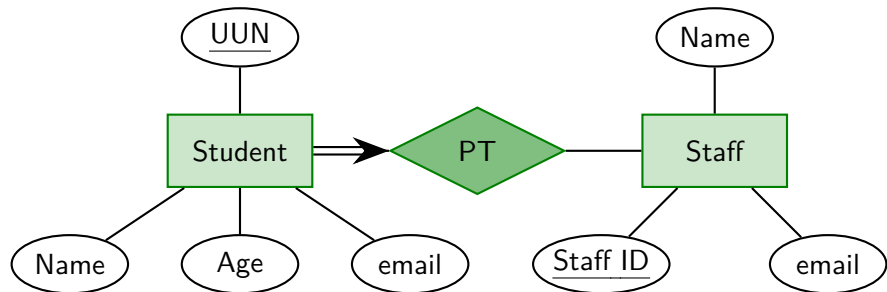
Forbidding **NULL** is in some cases a way to enforce total participation.

# Mapping Key and Participation Constraints



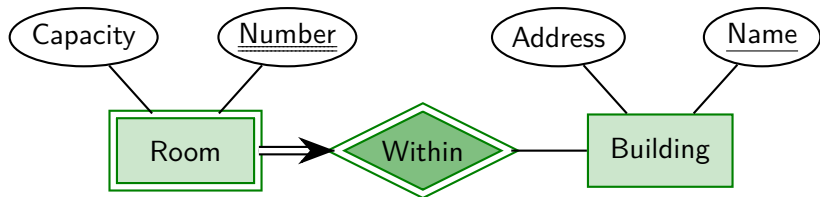
- Create a table for the source and target entities as usual.
- Add every key attribute of the target as a field in the source table.
- Declare these fields as **NOT NULL**
- Declare these fields as foreign keys.

# Mapping Key and Participation Constraints



```
CREATE TABLE Student (  
  uun  VARCHAR(8),  age  INTEGER,  
  name VARCHAR(20), email VARCHAR(25),  
  pt   VARCHAR(8) NOT NULL,  
  PRIMARY KEY (uun),  
  FOREIGN KEY (pt) REFERENCES Staff(staff_id) )
```

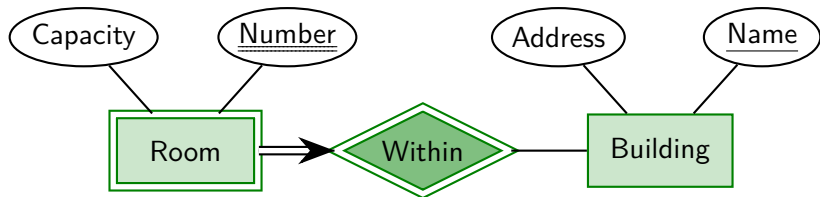
# Mapping Weak Entities and Identifying Relationships



A weak entity always has a participation and key constraint with its identifying relationship, which makes things similar to the previous case.

- Create a table for the weak entity.
- Make each attribute of the weak entity a field of its table.
- Add fields for the key attributes of the identifying owner.
- Declare a composite key using key attributes from both entities.
- Declare a foreign key constraint on the identifying owner fields.
- Instruct the system to automatically delete tuples in the table when their identifying owners are deleted.

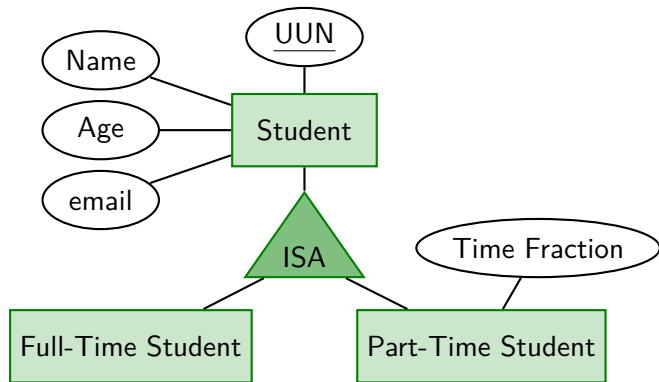
# Mapping Weak Entities and Identifying Relationships



```
CREATE TABLE Room (  
    number      VARCHAR(8),  
    capacity    INTEGER,  
    building_name VARCHAR(20),  
    PRIMARY KEY (number, building_name),  
    FOREIGN KEY (building_name)  
        REFERENCES Building(name)  
        ON DELETE CASCADE )
```

(Don't use **ON DELETE SET NULL** here)

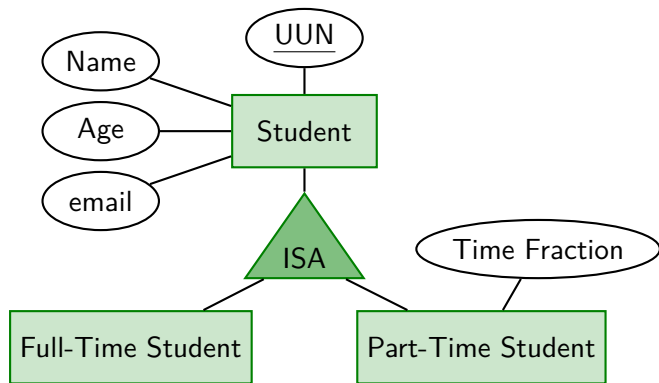
# Mapping Entity Hierarchies and Inheritance



- Declare a table for the superclass entity with all attributes.
- For each subclass entity declare another table using the primary key of the superclass and any extra attributes of the subclass.
- Declare the primary key from the superclass as the primary key of the subclass, with a foreign key constraint.

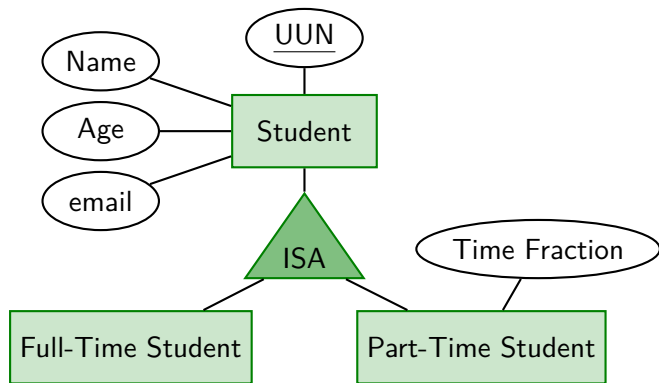


# Mapping Entity Hierarchies and Inheritance



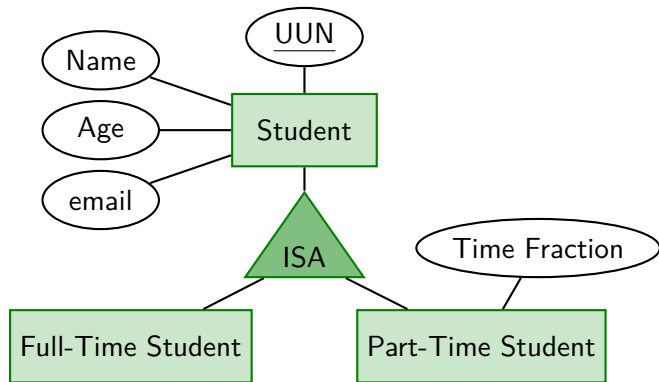
```
CREATE TABLE Student (  
  uun  VARCHAR(8),  age  INTEGER,  
  name VARCHAR(20), email VARCHAR(25),  
  PRIMARY KEY (uun) )
```

# Mapping Entity Hierarchies and Inheritance







```
CREATE TABLE FullTimeStudent (  
    uun VARCHAR(8),  
    PRIMARY KEY (uun),  
    FOREIGN KEY (uun) REFERENCES Student )
```

# Mapping Entity Hierarchies and Inheritance



```
CREATE TABLE PartTimeStudent (  
    uun VARCHAR(8), fraction FLOAT,  
    PRIMARY KEY (uun),  
    FOREIGN KEY (uun) REFERENCES Student )
```

## Database Textbooks (Page 1 of 2)

-  R. Ramakrishnan and J. Gehrke. 3 copies in HUB  
*Database Management Systems.* 6 other copies  
McGraw-Hill, third edition, 2003.
-  H. Garcia-Molina, J. Ullman, and J. Widom. 1 copy in HUB  
*Database Systems: The Complete Book.*  
Pearson, second edition, 2008.
-  J. Ullman and J. Widom. 1 copy in HUB  
*A First Course in Database Systems.* 1 other copy  
Prentice Hall, third edition, 2009.
-  M. Kifer, A. Bernstein, and P. M. Lewis. 1 copy in HUB  
*Database Systems: An Application-Oriented Approach, Introductory Version.*  
Pearson, second edition, 2005.

## Database Textbooks (Page 2 of 2)



A. Silberschatz, H. Korth, and S. Sudarshan.

1 copy in HUB

*Database System Concepts.*

McGraw-Hill, sixth edition, 2010.



T. Connolly and C. Begg.

1 copy

*Database System: A Practical Approach to Design, Implementation and Management.*

Addison-Wesley, fourth edition, 2005.



R. Elmasri and S. Navathe.

3 copies

*Fundamentals of Database Systems.*

Addison-Wesley, third edition, 2000.

# Summary

## Conceptual Model

ER-diagrams: entities with attributes and the relationships between them.

## Logical Model

Relational models: tables with fields that can refer to other tables.

## Translation

We can use an ER-diagram to guide the design of an appropriate relational model. This may require choosing between alternatives and making compromises: possibly involving simplicity of the design or its efficient implementation.