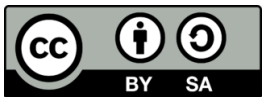
THE UNIVERSITY of EDINBURGH

Data Science in Medicine

Lecture 6: The Relational Model

Dr Areti Manataki

Usher Institute
The University of Edinburgh



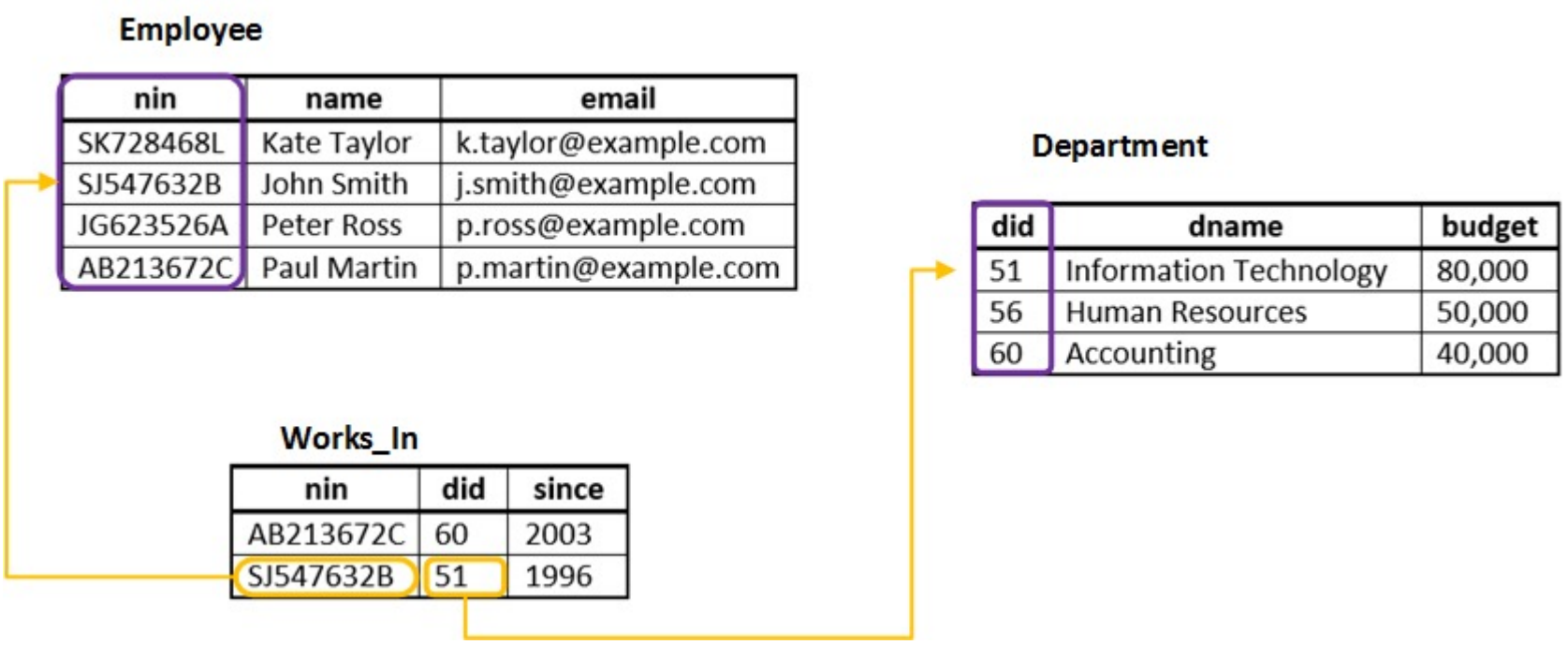
Usher Institute
Population Health Sciences & Informatics

Introduction to databases

- Database: a well-organised collection of data that are related in a meaningful way, describing a domain of interest
  - Examples: university, library, biomedical

Introduction to databases

- Relational databases: focus on *things* in the domain of interest and the *relations* between them



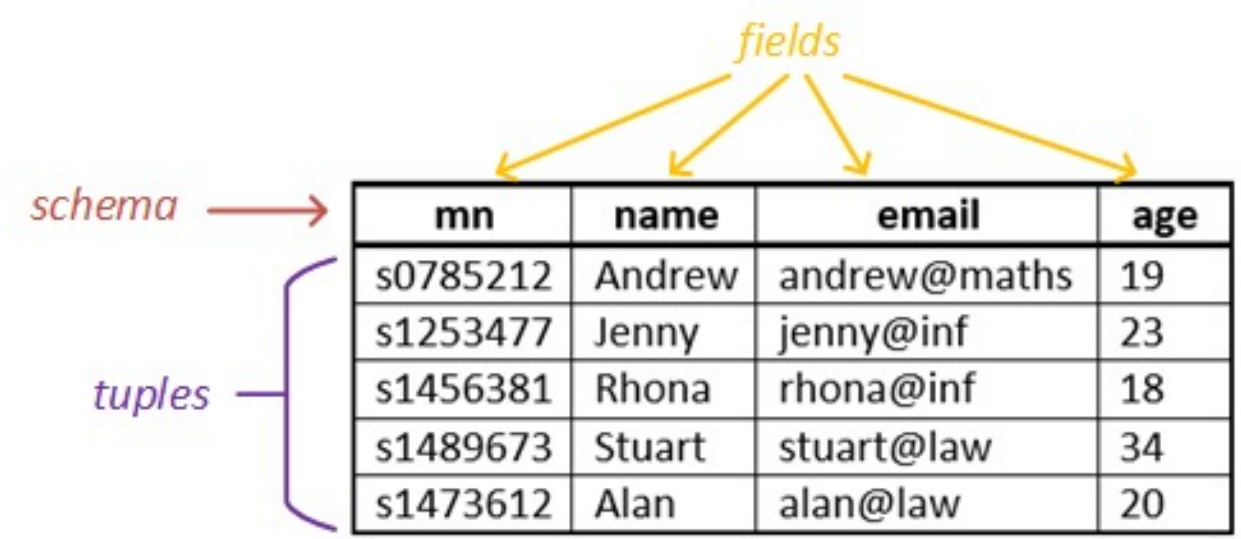
Introduction to databases

- Database management system: software that helps maintain and utilise large collections of data in an efficient manner
  - Examples: Ms Access, Ms SQL Server, Oracle DBMS, MySQL and many more
- Advantages include:
  - Data independence
  - Efficient data access
  - Data integrity
  - Concurrent access and recovery from crashes

Introduction to the relational model

- Relational database: a collection of relations
- A relation consists of:
  - Relation schema: describes the format of a table, consisting of:
    - Relation's name
    - Name of each field (column)
    - Domain of each field
  - Relation instance: the content of a table
    - A set of tuples (records)

Introduction to the relational model



mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@inf	18
s1489673	Stuart	stuart@law	34
s1473612	Alan	alan@law	20

- Arity (also called degree): number of fields
- Cardinality: number of rows

Introduction to the relational model

- Database: a set of tables with rows and columns, and links between them

Student

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@inf	18
s1489673	Stuart	stuart@law	34
s1473612	Alan	alan@law	20

Course

cid	title	credits
dbs	Database Systems	20
inf1	Informatics 1	10
sls	Scottish Legal System	10
lalg	Linear Algebra	10

Takes

mn	cid
s0785212	lalg
s1253477	dbs
s1253477	inf1
s1489673	sls

Creating relations with DDL

- Structured Query Language (SQL): standard language for creating, manipulating and querying data in relational database management systems
- Data Definition Language (DDL): subset of SQL that allows us to create, delete and modify tables
- `CREATE TABLE` declaration: define a new relation (called ‘table’ in SQL) with a particular schema

Creating relations with DDL

```
CREATE TABLE Student (
     mn CHAR(8),
     name CHAR(20),
     email CHAR(25),
     age INTEGER )
```

Student

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@inf	18
s1489673	Stuart	stuart@law	34
s1473612	Alan	alan@law	20

- Domain constraint: for each row in this table, the values in each column must be drawn from the appropriate domain

Creating relations with DDL

- General form:

```
CREATE TABLE table-name (
    ⟨attribute declarations⟩
    ⟨integrity constraints⟩
)
```

- Integrity constraints:
  - Key constraints
  - Foreign key constraints

Primary key constraints in DDL

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

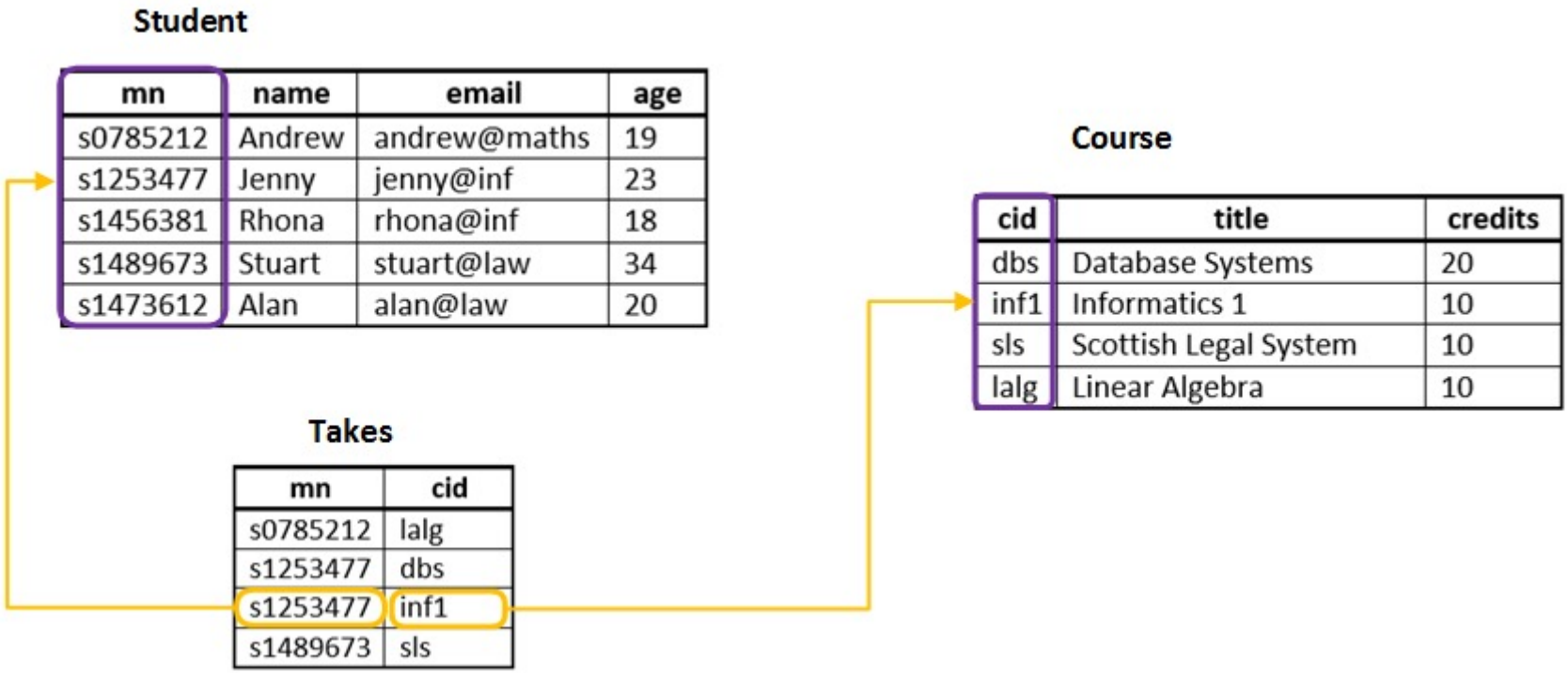
Student

mn	name	email	age
s0785212	Andrew	andrew@maths	19
s1253477	Jenny	jenny@inf	23
s1456381	Rhona	rhona@inf	18
s1489673	Stuart	stuart@law	34
s1473612	Alan	alan@law	20

- The primary key *uniquely identifies* a tuple.
- No two rows in the Student table have the same mn.

Foreign key constraints in DDL

- Foreign key constraints specify *links* between tables.



Foreign key constraints in DDL

- Foreign key constraints specify *links* between tables.

```
CREATE TABLE Takes (
    mn CHAR(8),
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student,
    FOREIGN KEY (cid) REFERENCES Course
)
```

  - This specifies that mn will always take a value that appears in the Student table.

Foreign key constraints in DDL

- Foreign key constraints specify *links* between tables.

```
CREATE TABLE Takes (
    mn CHAR(8),
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student,
    FOREIGN KEY (cid) REFERENCES Course
)
```

  - This specifies that cid will always take a value that appears in the Course table.

Foreign key constraints in DDL

```
CREATE TABLE Takes (
    mn CHAR(8),
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student,
    FOREIGN KEY (cid) REFERENCES Course
 )
```

- The foreign key in a referencing relation must match the primary key of the reference relation.
- Same number of columns and same domains.

Foreign key constraints in DDL

- What should happen if a row in Student is deleted?
  - 1st approach: All rows in Takes that refer to it should be deleted.

```
CREATE TABLE Takes (
    mn CHAR(8),
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student
                 ON DELETE CASCADE,
    FOREIGN KEY (cid) REFERENCES Course
 )
```

Foreign key constraints in DDL

- What should happen if a row in Student is deleted?
  - 2nd approach: For each row in Takes that refers to it, set its mn value to some pre-specified default value.

```
CREATE TABLE Takes (
    mn CHAR(8) DEFAULT ‘s0000000’,
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student
                 ON DELETE SET DEFAULT,
    FOREIGN KEY (cid) REFERENCES Course
 )
```

Foreign key constraints in DDL

- What should happen if a row in Student is deleted?
  - 3rd approach: For each row in Takes that refers to it, set its mn value to null.

```
CREATE TABLE Takes (
    mn CHAR(8),
    cid CHAR(20),
    mark INTEGER,
    PRIMARY KEY (mn, cid),
    FOREIGN KEY (mn) REFERENCES Student
                  ON DELETE SET NULL,
    FOREIGN KEY (cid) REFERENCES Course
 )
```

Conclusions

- The fundamental idea of the relational model is a relation.
- DDL allows us to specify:
  - the schema of our relations
  - primary and foreign keys