Australian institute

of

music

# Part-1: Entity-relationship data model

ER model is a graphical representation of entities and their relationship to each other, typically used in computing in regard to the organization of data within database. The meaning is described in terms of a conceptual or ER schema. ER schemas are comparable class diagrams in UML



List of entities

Entity: An object in the world that can be distinguished from other objects

Entity set: A set of similar entities

Here the following entities are:

* staff
* staff\_course
* course
* instrument
* student
* student\_course
* invoice

List of attributes

The basic unit of information about any entity occurrence. Here describe the list of attributes i.e;

* staff: staff\_id, staff\_name, staff\_family\_name, staff\_dob, staff\_phone, staff\_add, staff\_degree
* staff\_course: staff\_id, course\_id
* course: course\_id, course\_name, cost, time
* instrument: instrument\_id, instrument\_name, course\_id, stock\_qty, manufature\_name
* student: s\_id, s\_name, s\_add, s\_dob, s\_phone, s\_family\_name, s\_email
* student\_course: s\_id, course\_id, datetime, course\_duration, staff\_id, payment, date\_paid
* invoice: invoice\_no, s\_id, amount, date

Business Rules:

Business rules are the constraints that you wish to impose protect the database from becoming incomplete, inaccurate, or inconsistent. Although you may not be able to implement some business rules within the DBMS. It is concerned only with high-level design that is, specifying what business rules are required irrespective of how this might be achieved. Having identified the business rules, you will have a logical data model that is a complete and accurate representation of the organization to be supported by the database.

We consider the following types of business rules i.e, required data, column domain constraints, entity integrity, multiplicity, referential integrity, other business rules.

Here the business rule defined by the database i.e

1. The minimum course time to teach is 30 minutes.
2. The Maximum Course time to teach is 1 hr.
3. Invoice will generate only after course is finished for the student.
4. Invoice will be generated only for previous month completed course.

Assumptions:

1. Student(s) cannot enrol is same course again if the last payment is due.

# Part-II Relational database implementation

**Normalization**: Normalization is a formal process for design which attributes should be grouped together in a relation. Normalization can be a conventional method regarding design which attributes needs to be grouped together in a relation. Just before proceeding with all the physical design we'd like a strategy to confirm the particular logical design to this point. Normalization is a tool to be able to confirm & enhance the logical design. It requires to fulfil a few constrain.

There are several normal forms defined,

First Normal Form (1NF)

Second Normal Form (2NF)

Third Normal Form (3NF)

Boyce codd normal form (BCNF)

Fourth Normal Form (4NF)

Fifth Normal Form (5NF)

First Normal Form:

 First Normal Form eliminates repeating groups by putting each into a separate table and connecting them with a one-to-many relationship.

Two rules follow this definition:

* Each table has a primary Key made of one or several fields and uniquely identifying each record
* Each field is atomic, it does not contain more than one value.

Second Normal Form:

Second Normal Form eliminates functional dependencies on a partial key by putting the fields in a separate table from those that are dependent on the whole key.

Third Normal Form:

Third Normal Form eliminates functional dependencies on non-key fields by putting them in a separate table. At this stage, all non-key fields are dependent on the key, the whole key and nothing but the key.

**Here the table are fully normalised i.e,**

1. staff (staff\_id, staff\_name, staff\_family\_name, staff\_dob, staff\_phone, staff\_add, staff\_degree)

It’s Primary Key: staff\_id

1. staff\_course (staff\_id, course\_id)

Foreign Key (staff\_id) References staff(staff\_id), Foreign Key (course\_id) References course(course\_id)

1. course(course\_id, course\_name, cost, time)

It’s Primary Key: course\_id

1. Instrument (instrument\_id, instrument\_name, course\_id, stock\_qty, manufature\_name It’s Primary Key: instrument\_id)

Foreign Key (course\_id) References course(course\_id)

1. Student ( s\_id, s\_name, s\_add, s\_dob, s\_phone, s\_family\_name, s\_email)

It’s Primary Key: s\_id

1. student\_course: s\_id, course\_id, datetime, course\_duration, staff\_id, payment, date\_paid

Foreign Key (course\_id) References course(course\_id), Foreign Key (s\_id) References student(s\_id)

1. invoice (invoice\_no, s\_id, amount, date)

It’s Primary Key: invoice\_no, Foreign Key (s\_id) References student(s\_id)

Functional Dependencies

A functional dependency (FD) is a constraint between two sets of attributes in a relation from a database.

 FD: X → Y is called **trivial** if Y is a subset of X.

**zip 🡪 city, state**

1. Multi-valued Dependencies

Multivalued dependency *is a full constraint* between two sets of attributes in a relation.

x🡪🡪 y

s\_name 🡪 s\_phone

s\_name 🡪 s\_id

1. Candidate Keys

A candidate key is an attributes or set of attributes that uniquely identifies individual occurrences or an entity type. It is always NOT NULL and unique, which means that the values in theses column(s) must never change.

staffl: {staff\_id, staff\_phone}

1. 4NF Violators

add 🡪 city, state

1. Decomposition

**1st Decomposition**

**Decomposition on** x🡪🡪y or s\_name🡪 s\_phone, s\_name🡪 s\_id

Then r(x) or student(s\_name, s\_phone, s\_id)

**New Candidate Keys**
student: { s\_id, s\_name, s\_phone, s\_email}

**Remaining 4NF Violators**
s\_add 🡪 city, state

**2nd Decomposition**

**Decomposition on** x🡪🡪y or s\_phone🡪 s\_name, s\_phone 🡪 s\_email
Then r(x) or student(s\_name, s\_phone, s\_email)

**New Candidate Keys**student: { s\_id, s\_phone, s\_email}

**Remaining 4NF Violators**
add 🡪 city, state

**Final Decomposition**

student: { s\_id, s\_phone, s\_email}

Tables with sample data:

Relational database store in tables. Defined by a collection of columns and contain zero or more rows. Tables typically represent a type of object or entity. Here describe the table entities and attributes with its primary and foreign key with data.

**Primary Key:**

* Primary key cannot accept null values
* By default, primary key is clustered index and data in the database table is physically organized in the sequence of clustered index.
* We can have only one primary key in a table
* Primary key can be made foreign key into another table.

**Foreign Key:**

* Foreign key can accept multiple null values.
* More than one foreign key in a table.
* Foreign key is a field in the table that is primary key in another table

**1) staff: staff\_id, staff\_name, staff\_family\_name, staff\_dob, staff\_phone, staff\_add, staff\_degree**

Primary Key: staff\_id

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| staff\_id | staff\_name | staff\_family\_name | staff\_dob | staff\_phone | staff\_add | staff\_degree |
| S01 | PETER | NATALIA | 27-02-1984 | 176253762 | Queen Victoria Market MELBOURNE | DEPLOMA IN MSIC |
| S02 | DELTA | MICHAEL | 04-05-1981 | 231872771 | The Crown, MELBOURNE | DEPLOMA |
| S03 | JIMMY | MISSY | 02-11-1980 | 134322121 | Chinatown, MELBOURNE | DEGREE |
| S04 | NICK | KATE | 09-04-1976 | 128565899 | Eureka Skydeck MELBOURNE | DEPLOMA |

**2) staff\_course: staff\_id, course\_id**

Foreign Key (staff\_id) References staff(staff\_id), Foreign Key (course\_id) References course(course\_id)

|  |  |
| --- | --- |
| staff\_id | course\_id |
| S01 | C01 |
| S01 | C04 |
| S02 | C01 |
| S02 | C03 |
| S03 | C04 |
| S04 | C02 |

**3) course: course\_id, course\_name, cost, time**

It’s Primary Key: course\_id

|  |  |  |  |
| --- | --- | --- | --- |
| course\_id | course\_name | cost | time |
| C01 | Singing | 100 | 60 |
| C02 | Piano | 75 | 60 |
| C03 | Vialon | 120 | 60 |
| C04 | Guitar | 50 | 60 |

**4) instrument: instrument\_id, instrument\_name, course\_id, stock\_qty, manufature\_name** It’s Primary Key: instrument\_id

Foreign Key (course\_id) References course(course\_id)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| instrument\_id | instrument\_name | course\_id | stock\_qty | manufature\_name |
| I01 | PIANO | C02 | 3 | YAMAHA |
| I02 | VIOLIN | C03 | 4 | ROCKY |
| I03 | Guitar | C04 | 5 | TAYLOR |
| I04 | AMPLIFIER KIOSK | C04 | 2 | STAR |

**5) student: s\_id, s\_name, s\_add, s\_dob, s\_phone, s\_family\_name, s\_email**

It’s Primary Key: s\_id

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s\_id | s\_name | s\_add | s\_dob | s\_phone | s\_family\_name | s\_email |
| ST0001 | MICKY | MELBOURNE | 08-02-1997 | 298435667 | MICHAEL | MICKY@xyz.com |
| ST0002 | ANNA | STREET NO - 75 MELBOURNE | 30-03-2001 | 273645789 | PETER | ANNA@xyz.com |
| ST0003 | LISA | STREET NO - 105 MELBOURNE | 15-12-2000 | 243654167 | KATE | LISA@xyz.com |
| ST0004 | KAITE | STREET NO - 11MELBOURNE | 22-06-2000 | 277677899 | NICK | KAITE@xyz.com |

**6) student\_course: s\_id, course\_id, datetime, course\_duration, staff\_id, payment, date\_paid**

Foreign Key (course\_id) References course(course\_id), Foreign Key (s\_id) References student(s\_id)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s\_id | course\_id | datetime | course\_duration | staff\_id | payment | date\_paid |
| ST0002 | C01 | 28-07-2018 | 60 | S01 | Y | 11-09-2018 |
| ST0001 | C01 | 25-07-2018 | 60 | S02 | Y | 02-09-2018 |
| ST0001 | C04 | 08-08-2018 | 30 | S03 | Y | 10-09-2018 |
| ST0003 | C02 | 18-09-2018 | 60 | S04 | N | 10-09-2018 |

**7) invoice: invoice\_no, s\_id, amount, date**

 It’s Primary Key: invoice\_no

Foreign Key (s\_id) References student(s\_id)

|  |  |  |  |
| --- | --- | --- | --- |
| invoice\_no | s\_id | amount | Date |
| IN0001 | ST0001 | 150 | 31-08-2018 |
| IN0001 | ST0002 | 100 | 31-08-2018 |
| IN0001 | ST0001 | 120 | 30-08-2018 |
| IN0001 | ST0002 | 110 | 30-08-2018 |

1. Relationships
* They represent logical links between two or more entities.
* There are three types of relationship i.e.

One to One Relationship,

One to many Relationship

Many to Many Relationship

One to one relationship

* One to one relationship set between course entity and instrument entity sets.

One to many relationship

* One to many relationship set between course staff and staff\_course entity sets.
* One to many relationship set between course and staff\_course entity sets.
* One to many relationship set between student and student\_course entity sets.
* One to many relationship set between student and invoice entity sets.
* One to many relationship set between course and student\_course entity sets.

Many to many relationship

* Many to many relationship set between staff\_course and student\_course entity sets.



1. Queries
* List details of students who are under 18 years of age.

|  |
| --- |
| SELECT student.s\_dob as Date\_Of\_Birth, Year(Now())-Year([s\_dob]) AS AgeFROM studentWHERE Year(Now())-Year([s\_dob])<=18; |

* Show details of lessons coming up in the next 7 days (if any).

|  |
| --- |
| SELECT a.course\_id as Course\_ID, b.course\_name as Course\_Name, b.timeFROM student\_course AS a, course AS bWHERE DateDiff('d',now(),a.datetime)<8 And b.course\_id=a.course\_id; |

* Show a list of students and the total number of lessons they have had. Show the student ID, family name, given name, and the number of lessons.

|  |
| --- |
| SELECT a.s\_id as Student\_ID, b.s\_name as Name, b.s\_add as Addess, b.s\_dob as Date\_Of\_BirthFROM student\_course AS a, student AS bWHERE a.course\_id =(select course\_id from course where course\_name='PIANO') and a.s\_id=b.s\_id; |

* Show details of any student who has had piano lesson.

|  |
| --- |
| SELECT a.s\_id, a.s\_name, a.s\_family\_name (select count(\*) from student\_course b where b.s\_id =a.s\_id)FROM student AS a; |

1. Report.

1)



2)



3)

