University of Glasgow

Dip / MSc Information Technology

Information Systems and Databases

Tutorial Week 5 Databases and ER Diagrams

Richard Cooper

October 30th 2008

Databases

1. *What is meant by the logical data model of a database system?*

The structure to which all kinds of data must conform to be stored in the database – e.g. tables.

2. *Describe the main facilities that a DBMS will provide to an application.*

Describe efficient access, integrity checking, concurrency, security, reliability and recovery, distribution, etc.

3. *What are the features of a relational database?*

The database consists of a set of tables, each of which has named and typed columns (also called fields or attributes) and rows (records or tuples) identified only by their content. Rows and columns are theoretically unordered, although order must be imposed for storage and presentation to the user.

4. *How is information related in a relational database?*

All data is stored in tables. Tables have named and typed columns (columns are called fields and types are called domains in databases) and rows (called records or tuples) which are only identified by their content and not by any external mechanism.. Each cell has only one atomic value – usually a small string or a number, although a field can hold dates which are distinguished from other numbers. No two records can have exactly the same set of values to ensure that they can all be distinguished from each other. The mechanism actually used for distinguishing them is to identify one (or more) columns to act as the primary key.

5. *What is the role of a primary key and a foreign key? Why must there always be a primary key?*

The primary key identifies records in a table. There must be one as no two records have exactly the same data. So you could identify a record by listing all the values in it, but it is much better to have one or two fields which you can use. They must be unique and non-null (i.e. they must have a value) for all records.

A foreign key is a field in a table which holds values from another (target) field in the database and is used to relate two tables. The values in the target field must be unique for all records. The values in the foreign key field must exist in the target field. The domains of the two fields must be the same. Usually the target field is a primary key. Sometimes (e.g. if the target is a primary key spanning more than one field), the foreign key is also more than one field and then the equality must over all the fields simultaneously and not one field at a time.

6. *The Entity Relationship diagram on the other side of the sheet shows a simplified schema for an airline booking system. Extract from the diagram, the requirements and constraints that resulted in the schema. Try to be as precise as possible in your specification.*

The Specification will look something like:

We wish to store booking information regarding a set of flights. The information to be stored includes:

There is a set of scheduled (possibly multi-leg) flights, identified by a flight number, including the airlines, which days of the week it runs (e.g. "MWF" or "135"), which fares are available and the sequence of legs that make up the flight. The set of fares for each flight includes a code, the amount it costs and possible restrictions. The leg of a flight must contain a number which identifies the leg within the flight (1st leg, 2nd leg, etc.), a scheduled departure and arrival point and scheduled departure and arrival times. Thus Fares, Flight, Flight Leg, Departure Airport and Arrival Airport correspond to a timetable holding details about regular advertised flights.

There are also a set of actual flight legs which are available for booking. These are instances of the advertised legs for particular dates and these are the units of booking. For each of these, we record the number of seats left, which aeroplane has been assigned to this particular journey and the reservations for the set of seats. Each reservation links a seat number with a customer, for each of whom we store their name and

phone number. Each leg instance may also have recorded where and when it actually starts from and ends at (Arrives and Departs).

We also store a set of airports with their code, name, city and state.

and a set of aeroplanes, with an identifier and number of seats and the type of the aeroplane. For each aeroplane type we store a type name, the maximum number of seats it can hold and the manufacturing company. We also record which types of aeroplane can land at each airport.

Some general points:

The weak entity types require the primary key of a related entity for their identification. This in turn may be a weak entity type. For instance, *Seat* depends on *Leg Instance* which in turn depends on *Flight* and this finally depends on *Flight*. Therefore the primary key for *Seat* is made up of the partial keys of *Seat*, *Leg Instance* and *Flight Leg* and the key of *Flight*. Such chains of weak entity types must always end with a strong entity type.

If your ER diagram contains a relationship in which one entity type is an instance of another, then the former (e.g. *Leg Instance*) may be a weak entity depending on the latter (e.g. *Flight Leg*).

The use of partial participation usually means that the relationship is one which is optional for the entities of that type. However, sometimes this would be nonsense and it means instead, either that historical data might be left in the database (airports that no longer operate) or that we do not have the information to store (perhaps we don't know which airports and a particular type of plane can land at) or that the application will infer some general data if there is an absence (maybe if an airport does not take part in the Can Land relationship, it can land all types of plane). The absence of the participation of Leg Instance in Arrives and Departs, for instance, might mean that these are only recorded for those flights which deviate significantly from the schedule.

There are many things we might like to say about the database for which the ER diagram is insufficient, for instance:

We cannot say that the number of seats in a particular plane is not more than the maximum number of seats for that type.

We cannot say that the departure and arrival airports are different.

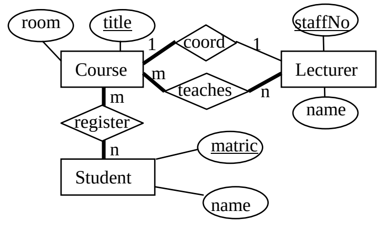
We cannot say that some types of fare are not available for flights involving particular airports.

etc.

Finally, it should be clear that ER diagram is intended to describe a database rather than the real world. You should always be guided by what use you intend to make of the data. In particular, adding constraints such as total participation, single valued attributes or having a cardinality of one means that the database is restricted in some way – non-nulls for total participation and nowhere to put a second piece of data in the other two cases.

7. Given the following description build a set of tables to hold the data. You will need to choose appropriate attributes for each of the main kinds of data.

A department runs a set of courses, each given by one or more lecturers, but having only one co-ordinator. Each course is held in several rooms – we only want to store the room number and nothing else about rooms. We also want to store which students register for which courses.



Course(<u>title</u> varchar2(20), room varchar2(10), coordinator number(6))

Lecturer(staffNo number(6), name varchar2(30))

Student(matric number(7), name varchar2(30))

Teaches(course varchar2(20), teacher number(6))

Registers(course varchar2(20), student number(7))