Ripasso Inter disciplinare INGLESE – INFORMATICA

V LICEO SCIENTIFICO

DATA MODELS E DATABASE

Data models

A **model** is a representation of real world entities and their relationships. Data models are conceptual tools for describing data in an organization. The descriptions include: data type, data constraints, data relationship and data semantics. There are many data models and they can be classified in three different groups, depending on different levels of generality: conceptual (or object level) models, logical (or record based) models, physical data models.

Conceptual (object level) **data models** represent data of an organisation by means of entities, attributes and relationship. Entities are concrete or abstract objects (examples of entities are: student, exam, course, car, ...); attributes are properties of an entity (attributes of a car entity might be: seats number, speed, engine power, ...). Relationships are associations between entities (example: the association between a course and the students attending it). E/R (Entity/Relationship) model is one of the most relevant techniques for database design methodology.

Logical (or record based) **data models** represent data as a collection of data set with a fixed format record. Each record of a data set consists of a fixed number of fields. The logical data model defines at very high level the physical implementation and it is non dependent from a specific DBMS adopted for the implementation. There are three main types of record based data models: **hierarchical** data model, **network** data model and **relational** data model. Most of the modern DBMS are based on the relational data model; in this model data are stored as tables and the relationships between tables are modelled with data values in the tables.

Physical data model describe data at lowest level and how to store the data in the computer memory. It deals with the real implementation of the record structure and index structure to access data.

Database

A database is any collection of related information about a subject, organised in a way that enables rapid access to selected data. Computer databases are organised as collections of data files, while a data file is a set of data records organised as a group of data fields. The smallest unit of information is the data field. A data field that uniquely identifies a specific record is called a **primary key.**
Usually, some level of quality in terms of accuracy, availability, usability is required which implies the use of a database management system (DBMS). Typically, a DBMS is a software system that meets a number of requirements.

Relational database

A relational database is based on the relational data model first proposed by E. F. Codd in 1970. According to the relational model, a database is a set of tables called relations. Each table is made up of named attributes or columns of data. A row of data in the table contains as many attributes as the number of columns in the table. In the terminology of the relational model a table has a degree and a cardinality. The degree of a

relation is the number of attributes it contains, the cardinality of a table is the number of rows it contains and a relational database is a collection of tables with different names.

GLOSSARY

Database
A collection of logically related data designed for the information needs of an organisation. In a database the data are stored together with their metadata (i.e. their description).

Database Management System (DBMS)
A set of application programs for defining and managing a database and its users.

Primary key
An attribute, or a set of attributes, in a table record that uniquely identifies one record in the table.

Query language
A language for querying, or retrieving data from, a database or other file system.

Relational data model
A data model in which data are stored as tables (or relations) and the relationships between tables are modelled with data values in the tables.

SQL
A language used to work with databases.

ACRONYMS

DBMS DataBase Management System

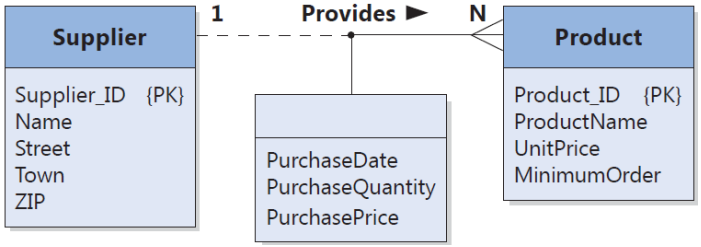
CONCEPTUAL E/R DATA MODEL

To develop a new database application it is necessary to gather all relevant information about the user's requirements. At the end of this stage of the analysis it is necessary to deliver the collected information in a document. The document's content must be unambiguous and clearly understood by two different groups of persons, with different technical background: database designers and end users of the application.
To prevent misunderstanding on the meaning of the data and their use by the organization, the document must be written in not technical terms but the ambiguity of the natural language has to be avoided.

The **Entity/Relationship** (**E/R**) model is one of the techniques that meets such conflicting requests. It is simple enough to be understood by non technical persons and it defines in detailed and unambiguous terms the data and their meaning in the analyzed organization.
E/R data model represents the analyzed reality first identifying the building blocks of the model called **entities.** An **entity** is relevant object in the analyzed reality which has a meaning in itself. In the model also

the **relationships** between entities are represented. The properties of entities and relationships are stored in data called **attributes**. E/R data model represents information in graphical form. It is generally accepted what the meaning of entity, relationship and attributes are, but there are different graphical notations to draw an E/R diagram. The notation chosen in this book is inspired by the Unified ModelingLanguage (UML).

Entities are represented with rectangles labelled with the entity name, relationships are drawn as a line, connecting the associated entities, labelled with the relationship name. Between the entities *Car* and *Person* there are two relationships: *Has*, that represents the logical association "*a Person Has a Car*" and the inverse relationship *Owned*, representing the logical association: "*a Car Owned by a Person*". By convention, only one of the two relationships is represented in the E/R diagram showing, through the arrowhead, the direction in which it makes sense. The attributes of a relationship are represented in a similar way.



In the above example the *Product* entity is characterized by the attributes listed in the diagram. The marking {PK} alongside *Product_ID* means that it is the **primary key** of the entity. This means that one value of *Product_ID* uniquely identifies an instance of the entity *Product*.

The E/R diagram also gives information on the number of occurrences of an entity's instance which relates with the instance of the associated entity. In the above diagram appear the symbols 1 and N, written close to the *Supplier* and *Product* entities, to indicate that an instance of *Supplier* is associated with one or many instances of *Product* and that a *Product*'s instance is associated with a single instance of *Supplier*. It is said that the relationship between *Supplier* and *Product* is a **One-to-Many**, or **1 : N**, relationship. The crow's foot drawn on the line representing the relationship *Provides* is another way to graphically indicate that the *Product* entity plays the role *to-Many* in the *One-to-many* relationship.

We observe that the E/R diagram conveys important information about the business rules of the analyzed reality. The example allows to state that the company under consideration is provided by certain suppliers, each providing one or more products, but that a given product is purchased by a single supplier. Note that if, with different business rules, the same product was purchased from multiple vendors, you would have a different E/R diagram, with the symbol N close to either *Product* and *Supplier* entities. The relationship *Provides* would be called a **Many-to-Many** or **N : N** relationship, meaning that: one supplier may provide one ore more products and that one product may be purchased by one or more suppliers.

A relationship may have attributes such as in the case of *PurchaseDate* for the association *Provides*. We observe that *PurchaseDate* is a property of the association between a supplier and the product supplied under a specific provision. Similarly *PurchasePrice* and *PurchaseQuantity* are the properties of the association between the two entities considered.

GLOSSARY

Attribute
A property of an entity or a relationship.

Attribute domain
The set of allowable values for that attribute.

Conceptual model
A high level data model that represents a reality in terms of objects called entities and the relationship among them.

Degree of a relationship
The number of entities that participate in the relationship. A relationship of degree 2 is called a binary relationship.

E/R model
A conceptual data model which describes a reality in terms of a collection of objects called entities, relationships among these objects and attributes.

Entity
An entity, more precisely an entity type, groups the objects with the same properties of a reality. An entity has a meaning in itself in the modelled reality.

Relationship
A set of logical associations among entities.

Unified Modeling Language
A graphical language to show, define, specify and document the building of software systems.

ACRONYMS

1:1 One-to-One
1:N One-to-Many
N:N Many-to-Many
E/R Entity Relationship
PK Primary Key
UML Unified Modeling Language

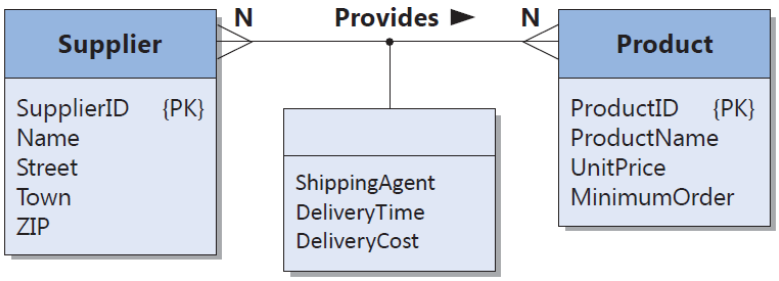
RELATIONAL DATABASE

1970, Edgar F. Codd, a mathematician working at the IBM San Jose Research Lab, published a paper titled: *"A Relational Model of Data for Large Shared Data Banks"* which showed that it was possible to access the information stored in a database without knowing how they were structured or how they were memorized in the database. At the heart of Codd's work there was a very simple but powerful idea: developing a system to request information to the computer while the computer itself had the task of finding the requested data in mass storage and return them. Another basic idea was that relationships between data items (for example: the relationship between a product and the supplier providing the product) should be based on the item's values, and not on separately specified links. In simple terms this approach provided a level of data independence that allowed users to access information without having to master details of the physical structure of a database. As a result the relational data model has become the most relevant and widespread record-based logical model.

In this chapter we have stated that the shift from the E/R data model of a reality to the corresponding relational model can be summarized in the following rules:

1) Create a relation (table) for every entity in E/R model. The relation has the same attributes of the entity; the attribute, or the set of attributes, chosen as the entity-key becomes the primary key of the table.
2) Representation of relationship between entities:

- In case of a 1:1 relationship both entities are combined in one table and one of the primary keys of the entities is chosen to be the primary key of the new table. There are exceptions to this rule: for instance, in case of optional participation on one side of the 1:1 relationship, it is more convenient to consider the 1:1 relationship if it were a 1:N relationship; the entity with the optional participation is considered the one that plays the 1 role in the 1:N relationship.
- In case of a 1:N relationship, a copy of the primary-key of the entity which plays the 1 role is posted, as a foreign key, in the table representing the entity which plays the "to many" role. In the case where a 1:N relationship has attributes, they also are added to the attributes of the table representing the "to many" entity.
- The N:N relationship is represented with a new table including a copy of the primary key of the entities that participate in the relationship together with the relationship attributes. For example the following E/R diagram:



is represented with the following tables, where the primary key attributes are underlined and the foreign key attributes are italicized.
Suppliers (SupplierID, Name, Street, Town, ZIP)
Products (ProductID, ProductName, UnitPrice, MinimumOrder)
Providing (*SupplierID*, *ProductID*, ShippingAgent, DeliveryTime, DeliveryCost)
The *SupplierID* and *ProductID* attributes, acting as foreign keys in the table *Providing*, take part in the composite primary key of the table. To provide uniqueness of the primary key it is sometimes (but not in this example) necessary to extend the set of attributes acting as primary key with one or more attributes of the relationship.

GLOSSARY

Candidate key
An attribute, or a minimal set of attributes, which uniquely identifies a row in a table.

Cardinality
The cardinality of a relation (table) is the number of rows it contains.

Foreign key
An attribute or a set of attributes of a relation that uniquely identifies a row in another relation.

Primary key
The candidate key chosen to uniquely identify a row in a table.

SQL LANGUAGE

SELECT is the SQL statement to query data in a relational database and it has the following general syntax:
SELECT Expression1, Expression2, … (Which columns or expressions are included in the output)
FROM Table1, Table2, … (The tables used in the query)
WHERE Condition (Condition to be satisfied by the rows to be considered)
GROUP BY Column, … (To form groups of rows with the same value of the column)
ORDER BY Column, … (How to sort the output list)

Selection example: list all the attributes of the books published by Atlas Books publisher. The selection condition must be written in the *Where* clause. The resulting query is:
SELECT * (* is a shorter way to list all columns)
FROM Books
WHERE Publisher = 'Atlas Books';

GLOSSARY

Alter
The *Alter* command changes one or more of the properties of a table, i.e. the name of the table or any table field, to add or delete a column in a table or to change a column definition.

Create
The *Create* statements are used to create a database (*Create Database* statement) and all the database objects: *Create Table* statement, *Create Index* statement, *Create View* statement.

Delete statement
The *Delete* command removes one or more rows from a table.

Drop
The *Drop* statement destroys the objects created with a corresponding *Create* command.

From
The *From* clause is used for specifying the table or tables used in the query.

Group By
The *Group By* clause considers the rows selected by a *Where* clause splitting them in homogeneous groups characterized by the same value of the column (or columns) that are specified after the words *Group By*.

Insert
The *Insert* command adds a new line to an existing table.

Order By
Specifies the order in which the rows of the query output are displayed.

Select clause
Defines the desired output, expressed as a list of columns or expressions obtained by combining columns extracted from the database tables.

Update
Modifies data in one or more rows of a table.

Where
The *Where* clause places conditions that must be satisfied by the rows of the table (or tables) specified in the *From* clause, to be included in the query