Week 5: Integrity Rules and Constraints

- Constraints are a very important feature in a relational model. In fact, the relational model supports the well-defined theory of constraints on attributes or tables.
- Constraints are useful because they allow a designer to specify the semantics of data in the database.
- *Constraints* are the rules that force DBMSs to check that data satisfies the semantics.

Domain Integrity

- Domain restricts the values of attributes in the relation and is a constraint of the relational model.
- However, there are real-world semantics for data that cannot be specified if used only with domain constraints. We need more specific ways to state what data values are or are not allowed and which format is suitable for an attribute. For example, the Employee ID (EID) must be unique or the employee Birthdate is in the range [Jan 1, 1950, Jan 1, 2001]. Such information is provided in logical statements called *integrity constraints*.
- There are several kinds of integrity constraints.

Entity integrity

- To ensure *entity integrity*, it is required that every table have a primary key.
- Neither the PK nor any part of it can contain null values. This is because null values for the primary key mean we cannot identify some rows.
- For example, in the EMPLOYEE table, Phone cannot be a primary key since some people may not have a telephone.

Referential integrity

- *Referential integrity* requires that a foreign key must have a matching primary key or it must be null.
- This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables. It means the reference from a row in one table to another table must be valid.
- Examples of referential integrity constraint in the Customer/Order database of the Company:

    Customer(**CustID**, CustName)

    Order(**OrderID**, CustID, OrderDate)

- To ensure that there are no orphan records, we need to enforce referential integrity. An *orphan record* is one whose foreign key FK value is not found in the corresponding entity – the entity where the PK is located. A typical join is between a PK and FK.
- The referential integrity constraint states that the customer ID (CustID) in the Order table must match a valid CustID in the Customer table. Most relational databases have declarative referential integrity. In other words, when the tables are created the referential integrity constraints are set up.

- Here is another example from a Course/Class database:

  Course(**CourseCode**, DeptCode, Description)

  Class(**CourseCode, Section**, ClassTime)

  - The referential integrity constraint states that CourseCode in the Class table must match a valid CourseCode in the Course table. In this situation, it's not enough that the CourseCode and Section in the Class table make up the PK, we must also enforce referential integrity.
  - When setting up referential integrity it is important that the PK and FK have the same data types and come from the same domain, otherwise the relational database management system (RDBMS) will not allow the join. RDBMS is a popular database system that is based on the relational model introduced by E. F. Codd of IBM's San Jose Research Laboratory. Relational database systems are easier to use and understand than other database systems.

Referential integrity in Microsoft Access

In Microsoft (MS) Access, referential integrity is set up by joining the PK in the Customer table to the CustID in the Order table.
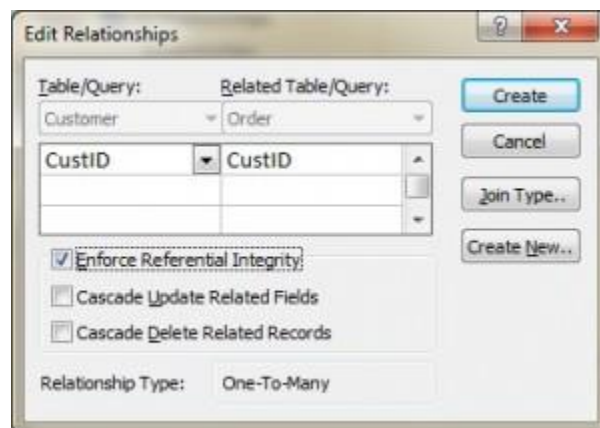
Referential access in MS Access, by A. Watt.

Referential integrity using Transact-SQL (MS SQL Server)

When using Transact-SQL, the referential integrity is set when creating the Order table with the FK. Listed below are the statements showing the FK in the Order table referencing the PK in the Customer table.

CREATE TABLE Customer
(CustID INTEGER PRIMARY KEY,
CustName CHAR(35))
CREATE TABLE Orders
(OrderID INTEGER PRIMARY KEY,

CustID INTEGER REFERENCES Customer(CustID),
OrderDate DATETIME)

Foreign key rules

- Additional foreign key rules may be added when setting referential integrity, such as what to do with the child rows (in the Orders table) when the record with the PK, part of the parent (Customer), is deleted or changed (updated).
- For example, the Edit Relationships window in MS Access shows two additional options for FK rules: **Cascade Update** and **Cascade Delete**. If these are not selected, the system will prevent the deletion or update of PK values in the parent table (Customer table) if a child record exists. The child record is any record with a matching PK.
- In some databases, an additional option exists when selecting the Delete option called Set to Null. In this is chosen, the PK row is deleted, but the FK in the child table is set to NULL. Though this creates an orphan row, it is acceptable.

Enterprise Constraints

- **Enterprise constraints** also referred to as **semantic constraints** are additional rules specified by users or database administrators and can be based on multiple tables.
- Here are some examples.
    - A class can have a maximum of 30 students.
    - A teacher can teach a maximum of four classes per semester.
    - An employee cannot take part in more than five projects.
    - The salary of an employee cannot exceed the salary of the employee's manager.

Business Rules

- *Business rules* are obtained from users when gathering requirements.
- The requirements-gathering process is very important, and its results should be verified by the user before the database design is built.
- If the business rules are incorrect, the design will be incorrect, and ultimately the application built will not function as expected by the users.
- Some examples of business rules are:
    - A teacher can teach many students.
    - A class can have a maximum of 35 students.
    - A course can be taught many times, but by only one instructor.
    - Not all teachers teach classes.

Cardinality and connectivity

- Business rules are used to determine cardinality and connectivity.
- *Cardinality* describes the relationship between two data tables by expressing the minimum and maximum number of entity occurrences associated with one occurrence of a related entity.
- In the Figure below, you can see that cardinality is represented by the innermost markings on the relationship symbol. In this figure, the cardinality is 0 (zero) on the right and 1 (one) on the left.
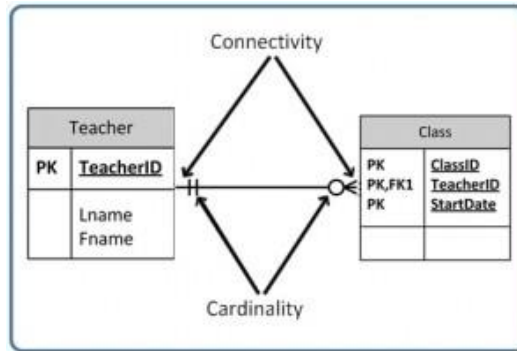
Position of connectivity and cardinality on a relationship symbol

- The outermost symbol of the relationship symbol, on the other hand, represents the connectivity between the two tables.
- *Connectivity* is the relationship between two tables, e.g., one to one or one to many. The only time it is zero is when the FK can be null.
- When it comes to participation, there are three options to the relationship between these entities: either 0 (zero), 1 (one) or many. In the Figure above, for example, the connectivity is 1 (one) on the outer, left-hand side of this line and many on the outer, right-hand side.

the symbol that represents a one to many relationship

- In the Figure below, both inner (representing cardinality) and outer (representing connectivity) markers are shown. The left side of this symbol is read as minimum 1 and maximum 1. On the right side, it is read as: minimum 1 and maximum many.

Relationship Types

- The line that connects two tables, in an ERD, indicates the *relationship type* between the tables: either identifying or non-identifying.
    - o An *identifying relationship* will have a solid line (where the PK contains the FK).
    - o A *non-identifying relationship* is indicated by a broken line and does not contain the FK in the PK
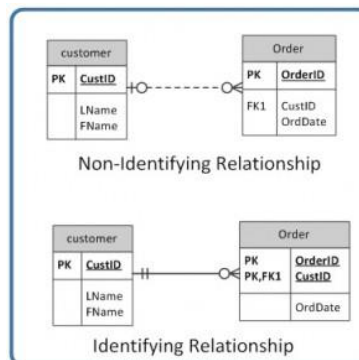

Identifying and non-identifying relationship

- In an *optional relationship*, the FK can be null or the parent table does not need to have a corresponding child table occurrence.
- The symbol, shown in the Figure below, illustrates one type with a zero and three prongs (indicating many) which is interpreted as zero OR many.

- For example, if you look at the Order table on the right-hand side in the Figure below, you'll notice that a customer doesn't need to place an order to be a customer. In other words, the **many side** is optional.
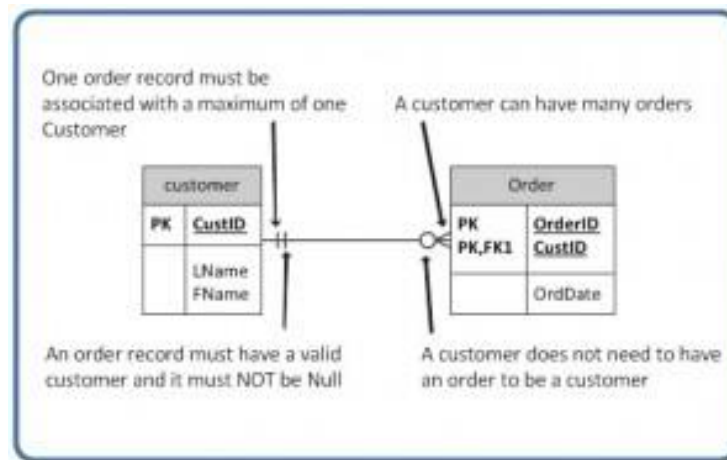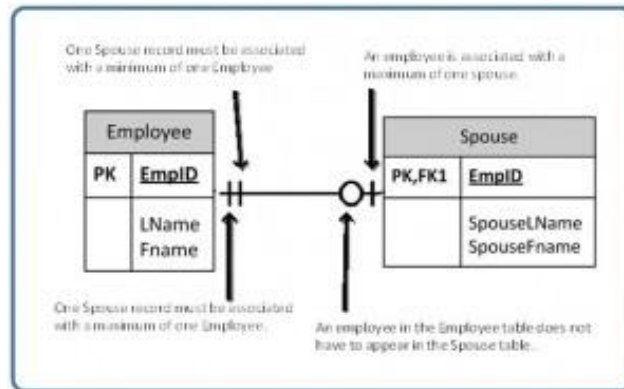

Example usage of a zero to many optional relationship symbol

- The relationship symbol in the above Figure can also be read as follows:
  - o Left side: The order entity must contain a minimum of one related entity in the Customer table and a maximum of one related entity.
  - o Right side: A customer can place a minimum of zero orders or a maximum of many orders.
- The Figure below shows another type of optional relationship symbol with a zero and one, meaning zero OR one. The **one side** is optional.
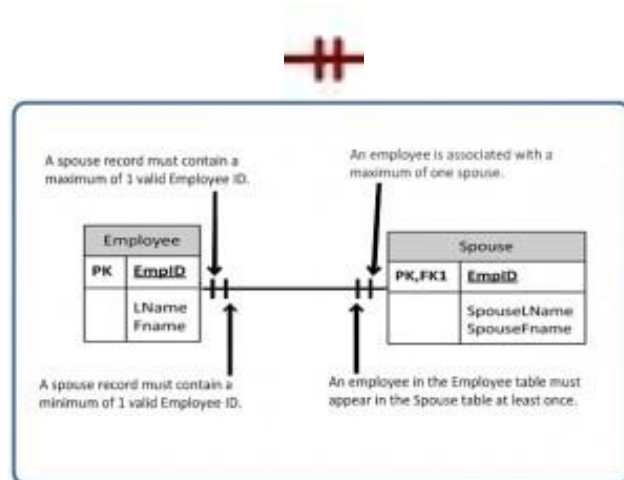

- The Figure below gives an example of how a zero to one symbol might be used.

Example usage of a zero to one optional relationship symbol
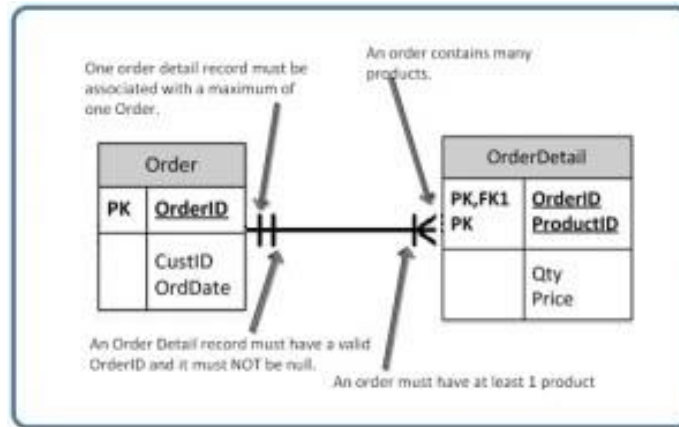
Mandatory relationships

- In a *mandatory relationship*, one entity occurrence requires a corresponding entity occurrence. The symbol for this relationship shows *one and only one* as shown in Figure below. The one side is mandatory.

Example of a one and only one mandatory relationship symbol

- The Figure below illustrates what a one to many relationship symbol looks like where the **many side** is mandatory.

Example of a one to many mandatory relationship symbol

- So far we have seen that the innermost side of a relationship symbol (on the left-side of the symbol) can have a 0 (zero) cardinality and a connectivity of many (shown on the right-side of the symbol), or one (not shown).

- However, it cannot have a connectivity of 0 (zero), as displayed in Figure below. The connectivity can only be 1.
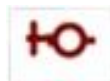
- The connectivity symbols show maximums. So if you think about it logically, if the connectivity symbol on the left side shows 0 (zero), then there would be no connection between the tables.
- The way to read a relationship symbol, such as the one in the Figure is as follows.
  - The CustID in the Order table must also be found in the Customer table a minimum of 0 and a maximum of 1 times.
    - The 0 means that the CustID in the Order table may be null.
    - The left-most 1 (right before the 0 representing connectivity) says that if there is a CustID in the Order table, it can only be in the Customer table once.
    - When you see the 0 symbol for cardinality, you can assume two things:
      1. the FK in the Order table allows nulls, and
      2. the FK is not part of the PK since PKs must not contain null values.
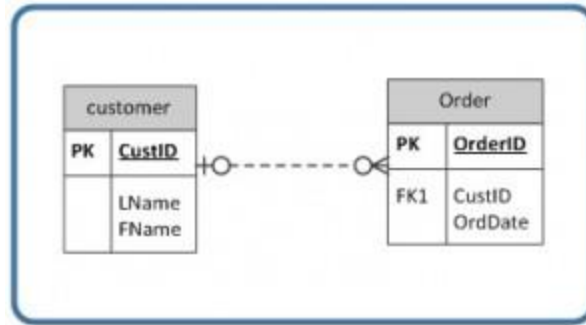
The relationship between a Customer table and an Order table