1.9 - What is the difference between controlled and uncontrolled redundancy?

**Answer:**

Redundancy is when the same fact is stored multiple times in several places in a database. For example, in Figure 1.5(a) the fact that the name of the student with StudentNumber=8 is Brown is stored multiple times. Redundancy is controlled when the DBMS ensures that multiple copies of the same data are consistent; for example, if a new record with StudentNumber=8 is stored in the database of Figure 1.5(a), the DBMS will ensure that StudentName=Smith in that record. If the DBMS has no control over this, we have uncontrolled redundancy.

1.12 – Cite some examples of integrity constraints that you think can apply to the database shown in Figure 1.2.

**Answer:**

We give a few constraints expressed in English. Following each constraint, we give its type in the relational database terminology that will be covered in Chapter 6, for reference purposes.

(a) The StudentNumber should be unique for each STUDENT record (key constraint).

(b) The CourseNumber should be unique for each COURSE record (key constraint).

(c) A value of CourseNumber in a SECTION record must also exist in some COURSE record (referential integrity constraint).

(d) A value of StudentNumber in a GRADE_REPORT record must also exist in some STUDENT record (referential integrity constraint).

(e) The value of Grade in a GRADE_REPORT record must be one of the values in the set \{A, B, C, D, F, I, U, S\} (domain constraint).

(f) Every record in COURSE must have a value for CourseNumber (entity integrity constraint).

(g) A STUDENT record cannot have a value of Class=2 (sophomore) unless the student has completed a number of sections whose total course CreditHours is greater than 24 credits (general semantic integrity constraint).
2.14 - if you were designing a Web-based system to make airline reservations and to sell airline tickets, which DBMS Architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?

**Answer:**

2.5.4 Three-Tier Client/Server Architecture for Web Application is the best choice. The Client consists of Web User Interface. The Web Server contains the application logic which includes all the rules and regulations related to the reservation process and the issue of tickets; the Database Server contains the DBMS.

2.5.1 Centralized DBMS Architecture would not work since the user interface and database server are on different machines for a web-based system.

2.5.2 Basic Client/Server Architecture and 2.5.3 Two-Tier Client/Server Architecture would work if the Business Logic can reside on server other than the DBMS Server. In general, if the business logic was on the DBMS Server, it will put an excessive burden on the server. If the business logic were to reside on the web client, it will burden the communication network as well a possibly thin client.

2.15 - Consider Figure 2.1. In addition to constraints relating the values of columns in one table to columns in another table, there are also constraints that impose restrictions on values in a column or a combination of columns within a table. One such constraint forces that a column or a group of columns must be unique across all rows in the table. For example, in the STUDENT table, the StudentNumber column must be unique (to prevent two different students from having the same StudentNumber). Identify the column or the group of columns in the other tables that must be unique across all rows in the table?

**Answer:**

<table>
<thead>
<tr>
<th>Table</th>
<th>Column(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE</td>
<td>CourseNumber</td>
</tr>
<tr>
<td></td>
<td>Since this contains the combination of the department and the number that must be unique within the department. Note we will overlook the fact this does not accommodate a department from offering several “Special Topics” course with the same CourseNumber but different titles. We could make this a combination of CourseNumber and CourseName, but this is more susceptible to someone mistyping while entering data.</td>
</tr>
<tr>
<td>PREREQUISITE</td>
<td>The combination of CourseNumber and PrerequisiteNumber</td>
</tr>
<tr>
<td>SECTION</td>
<td>SectionIdentifier</td>
</tr>
<tr>
<td></td>
<td>We assume that no two sections can have the same SectionIdentifier. If we were to consider that SectionIdentifier is unique only within a given course offered in a given term (such as section 2 of CS101) then the answer changes to the combination of SectionIdentifier, CourseNumber, Semester, and Year.</td>
</tr>
<tr>
<td>GRADE_REPORT</td>
<td>StudentNumber and SectionIdentifier</td>
</tr>
<tr>
<td></td>
<td>As per assumption stated in SECTION, the SectionIdentifier will be...</td>
</tr>
</tbody>
</table>
different if a student takes the same course or a different course in another term.

5.16 - Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT (SSN, Name, Major, Bdate)
COURSE (Course#, Quarter, Grade)
ENROLL (SSN, Course#, Quarter, Grade)
BOOK_ADOPTION (Course#, Quarter, Book_ISBN)
TEXT (Book_ISBN, Book_Title, Publisher, Author)

Specify the foreign keys for this schema, stating any assumptions you make.

Answer:

The schema of this question has the following four foreign keys:
a. the attribute SSN of relation ENROLL that references relation STUDENT,
b. the attribute Course# in relation ENROLL that references relation COURSE,
c. the attribute Course# in relation BOOK_ADOPTION that references relation COURSE, and
d. the attribute Book_ISBN of relation BOOK_ADOPTION that references relation TEXT.

5.18 - Database design often involves decisions about the storage of attributes. For example a Social Security Number can be stored as a one attribute or split into three attributes (one for each of the three hyphen-delimited groups of numbers in a Social Security Number—XXX-XX-XXXX). However, Social Security Number is usually stored in one attribute. The decision is usually based on how the database will be used. This exercise asks you to think about specific situations where dividing the SSN is useful.

Answer:

a. We need the area code (also know as city code in some countries) and perhaps the country code (for dialing international phone numbers).
b. I would recommend storing the numbers in a separate attribute as they have their own independent existence. For example, if an area code region were split into two regions, it would change the area code associated with certain numbers, and having area code in a separate attribute will make it is easier to update the area code attribute by itself.
c. I would recommend splitting first name, middle name, and last name into different attributes as it is likely that the names may be sorted and/or retrieved by the last name, etc.
d. In general, if the each attribute has an independent logical existence based on the application, it would make sense to store it in a separate column otherwise there is no clear advantage. For example, SSN need not be split into its component unless we are using the subsequences to make deductions about validity, geography, etc. In the two cases above, it made logical and business sense to split the attributes.

5.20 - Recent changes in privacy laws have disallowed organizations from using SSN to identify individuals unless certain restrictions are satisfied. As a result, most US universities cannot use SSNs as primary keys (except for financial data). In practice, StudentID, a unique ID, a unique
identifier, assigned to every student, is likely to be used as the primary key rather than SSN since StudentID is usable across all aspects of the system.

a. Some database designers are reluctant to use generated keys (also known as surrogate keys) for primary keys (such as StudentID) because they are artificial. Can you propose any natural choices of keys that can be used to store the student record in a UNIVERSITY database?

b. Suppose that you were able to guarantee uniqueness of a natural key that included last name. Are you guaranteed that the last name will not change during the lifetime of the database? If the last name can change, what solutions can you propose for creating a primary key that still includes last name but remains unique?

c. What are the advantages and disadvantages of using generated (surrogate) keys?

**Answer:**

a. By keeping the name attributes separated, we allow the possibility of looking these pieces of their name. In a practical use, it is not likely that the user will know the correct primary key for a given student and so we must consider how a user will locate the correct row without this information. If we were to collapse the name into a single attribute, then we have complicated any sort of “lookup by name” query; such a query would then require partial string matching and any results might not disambiguate between FirstName and LastName. Therefore, a practical system should allow name searches by FirstName and LastName; we must leave MiddleInitial separated still to avoid ambiguities from combining these pieces together.

b. A single attribute Phone# would no longer suffice if a student were able to have multiple phone numbers. We could possibly have multiple rows for a single student to allow this to happen, but then we have violated key principles of database design (e.g., having redundant data). A better solution would be to include the additional attributes HomePhone, CellPhone, and OfficePhone and allow the possibility of these attributes to have no value. Again, this is not most desirable because most students will not have all three of these attributes, and we will have many valueless key/attribute pairs. An excellent solution would be add an additional relation Phone# (SSN, Type, Number) while removing PhoneNumber from the Student relationship. This new relationship would allow the one-to-many relationship from students to phone numbers without creating redundant data or wasting space on sparse, valueless attributes.