**Question A.**

APPLICATION (ApplicationId, MediaName)

STUDENTS (CWID, FName, LName, Email, Phone, Year)

USES (UsesID, *CWID, ApplicationId*, Date, Time, Activity)

STUDENTS\_APPLICATION (*CWID, ApplicationId,* TimeSpent)

All entities are normalized.

**APPLICATION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Key | Required | Default Value | Remarks |
| ApplicationId | Integer | Primary Key | Yes | DBMS supplied | Surrogate Key: Initial value = 1. Increment = 1 |
| MediaName | VarChar(100) | No | Yes | None |  |

**STUDENTS\_APPLICATION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column Name | Data Type | Key | Required | Default value | Remarks |
| CWID | Integer | Composite, Foreign | Yes | None | Ref: STUDENTS |
| ApplicationId | Integer | Composite, Foreign | Yes | None | Ref: APPLICATION |
| TimeSpent | Integer | None | Yes | 0 | Measured in minutes |

**Question B.**

The above tables document the column name, data types, null status’, default values, remarks, and the keys of all tables. Such descriptions help us organize and keep track of the relevant information needed when creating the tables in SQL.

**Question C.**

The importance of referential integrity constraint enforcement in database design is to prevent errors from being introduced into the database. It prevents data from being entered into a row of a child table for which you don’t have any corresponding row in the parent table.

|  |  |  |
| --- | --- | --- |
| Relationship | Referential Integrity Constraint | Cascading Behavior |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parent | Child |  | On Update | On Delete |
| APPLICATION | STUDENTS\_APPLICATION | ApplicationID in STUDENTS\_APPLICATION must exist in ApplicationID in APPLICATION | No | No |
| STUDENTS | STUDENTS\_APPLICATIONS | CWID in STUDENTS\_APPLICATION must exist in CWID in STUDENTS | No | No |
| APPLICATION | USES | ApplicationID in USES must exist in ApplicationID in APPLICATION | No | No |
| STUDENTS | USES | CWID in USES must exist in CWID in STUDENTS | No | No |

Justifications

* **APPLICATION and STUDENT\_APPLICATION**

If an application is discontinued and deleted from APPLICATION, the database users would still prefer to preserve data on how much time was spent by students on that application. Therefore, the delete cascading behavior is set to a no.

Because ApplicationId is a surrogate key, no cascading update behavior is necessary.

* **STUDENTS and STUDENTS\_APPLICATION**

As CWID in STUDENTS is an unchanging value, so these relationships do not need cascading updates.

Regarding cascading deletions, rows in the STUDENTS\_APPLICATION child table require a STUDENTS parent. We decided that students are never deleted from the database once the database has a student record to accumulate data on social media usage. Therefore, this relationship has no cascading deletion.

* **APPLICATION and USES**

On delete cascading behavior is set to “No” because we assume that the users prefer to preserve data about “Activity” of a deleted social media application.

Because ApplicationId is a surrogate key, no cascading update behavior is necessary.

* **STUDENTS and USES**

Since CWID in STUDENTS is an unchanging value, these relationships do not need cascading updates.

On delete, rows in the USES child table require a STUDENTS parent. We decided to keep the records in the USES table as students are never removed after being recorded into the database. Therefore, the relationship has no cascading deletion.

**Question E.**

The major difference between a data model and database design during the transformation phase is that we convert a general data model into a database to be implemented in a DBMS. More specifically, we use the data model to define the tables, keys, attributes and normalize the tables to ensure that the database will not have any modification problems.