1. **Explain how referential integrity has been enforced in this database and illustrate, using specific examples, how this ensures the integrity of the data.**

|  |  |
| --- | --- |
| **Table Name** | **Table Contents** |
| Child | The details of all registered children on the Playscheme, including those who have not signed up for an activity. |
| Carer | The details of all carers employed by the Playscheme, including those with no assigned children. |
| Activity | The details of all activities provided as part of the Playscheme such as activity name, day and fee. |
| Childactivity | The Playscheme register where each child is linked to the activity(ies) for which they have been registered. |

1. *Child* and *Carer*

There is a one-to-many relationship between the Carer and Child entities where one carer may be responsible for one or more children, while each child is the responsibility of only one carer. Each record on the one side of the relationship (‘parent’ table) may be related to one or more records on the many side of the relationship (‘child’ table), (Fehily 2008). To represent this, I have added a foreign key column, child\_carer, in the Child table. It stores carer\_id, the primary key of the Carer table, which uniquely identifies each record there.

The foreign key value must match one of the values in the primary key of the parent table (Fehily 2008) so I have set it to NO NULL and assigned datatype INT. Data will have to be imported first into the Carer table (‘one’ side of the relationship) and then into the Child table (‘many’ side of the relationship), (Tahaghoghi and Williams, p. 183). There will be no need to type in the carer\_id since the carer\_id column at each side of the relationship has an auto\_increment datatype.

1. *Child* and *Activity*

There is a many-to-many relationship between children and activities: a child may attend one or more activities, while an activity may be attended by one or more children. This type of relationship cannot be represented directly in a relational database because neither table is a ‘parent’ table (Tahaghoghi and Williams, p. 119). It can be broken down using a linking table*—*Childactivity, which has a one-to-many relationship with both the Child and Activity entities.

A combination of two foreign key fields, also known as a composite key, guarantees the uniqueness of each row in this table: child\_id references the child\_id column in the Child (‘parent’) table, while activity\_id references the activity\_id column in the Activity (‘parent’) table. The two foreign key columns mirror the datatype and properties of the primary keys in the Child and Activity entities: INT(11), AI.

1. **Describe the process of deleting a carer record from the database assuming there is a child or children associated with this carer (for example, Liane Cartman) with the current foreign key settings of ‘On Delete: NO ACTION’.**

‘On Delete: NO ACTION’ will prevent the deletion of a row in the Carer (‘parent’) table if it is referenced by a record in the Child (‘child’) table(MySQL 5.7 Reference Manual). An attempt to remove Liane Cartment from the Playscheme database would therefore result in an error, but associated data in the ‘many’ side of the relationship can be deleted. In order to generate a list of the children associated with Liane, I needed to confirm her carer\_id:

select carer\_id

from Carer

where carer\_fname = ‘Liane’ and carer\_sname = ‘Cartmen’

select \*

from Child

where child\_carer =

However, the records in the Child table also had a one-to-many relationship with some values in the Childactivity table. I had to delete those values first. I could then remove all children linked to Liane and delete her record from the Carer table.

select \*

from Chiladactivity

where child\_id =

1. **What would be the effect of changing the foreign key settings to ‘On Delete: CASCADE’? Explain the advantages and disadvantages of using this setting.**

‘On Delete: CASCADE’will remove all related records in a ‘child’ table if the ‘parent’ record is deleted. Consequently, all foreign key values in the ‘many’ side of a relationship will be updated if the related primary key value is changed (MySQL 5.7 Reference Manual).

On the one hand, using ‘On Delete: CASCADE’ in the example of Liane Cartman would reduce the number of SQL statements needed to perform the action. Since all foreign key values will be removed, the presence of orphaned rows is also highly unlikely. On the other hand, if the wrong record from the ‘parent’ table (Carer) is deleted, all rows in the ‘child’ table (Child) will be removed and it will be difficult to ascertain which values need to be entered again.

Overall, the decision to use ‘On Delete: CASCADE’depends on the type of the entities: if the foreign key values cannot exist without the primary key values, it will make sense to have this option (Fehily p. 358). Children’s details do not need to be removed if the company no longer employs Liane since a new member of staff will be hired to look after them.