Introduction to Database Technology

FMA

Alessandro Metelli

Student ID: ametel01

Tutor: Jerry Smallwood

Contents

1: Information Retrieval

1. Listing of children and their associated activities.

Query:

```
SELECT
   CONCAT(child_fname, ' ', child_sname) AS 'Child Name',
   activity_name,
   activity_day
FROM Child C
JOIN Childactivity CA ON C.child_id = CA.child_id
JOIN Activity A ON A.activity_id = CA.activity_id
ORDER BY child_sname, child_fname
```

View Query:

```
CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `ametel01`@`%`
   SQL SECURITY DEFINER
VIEW `Childlist` AS
   SELECT
      CONCAT(`C`.`child_fname`,
            ' ',
            `C`.`child_sname`) AS `Child Name`,
      `A`.`activity_name` AS `activity_name`,
      `A`.`activity_day` AS `activity_day`
   FROM
      ((`Child` `C`
      JOIN `Childactivity` `CA` ON ((`C`.`child_id` = `CA`.`child_id`)))
      JOIN `Activity` `A` ON ((`A`.`activity_id` = `CA`.`activity_id`)))
   ORDER BY `C`.`child_sname` , `C`.`child_fname`
```

Note: This query looks badly formatted here, but once pasted in workbench they will look as I intended to. This is due to tab spacing I believe.

Testing:

I counted the number each activity is appears in the output and cross checked with the output from query N.4 on the column ‘Number of Children’ for each activity.

2. Count the number of activities for each child.

Query:

```
SELECT
   CONCAT(child_fname, ' ',child_sname) AS 'Child Name',
   COUNT(activity_id) AS 'N.Of Activities'
FROM Child C
LEFT JOIN Childactivity CA ON C.child_id = CA.child_id
GROUP BY C.child_id
ORDER BY child_sname, child_fname
```

CSV output:

Child Name	N.Of Activities
Kyle Broflovski	1
Eric Cartman	1
Miracle Cure	1
Ben Flowerpot	1
Bill Flowerpot	1
Robin Hood	2
Jack Horner	1
Marian Maid	1
Stan Marsh	1
Kenny McCormick	1
Beatrice Menace	1
Dennis Menace	2
Alessandro Metelli	0
Minnie Minx	3
Angelica Pickles	1
Jemima Puddleduck	2
Florence Roundabout	2
Bart Simpson	2
Lisa Simpson	1

Testing:

I ran a quick query (SELECT COUNT(child_id) FROM Child) which gave me 19 as result and counted the number of rows in the csv file.

3. A child and activity list for a specified carer.

Query:

```
SET @carerfname = 'Fred';
SET @carersname = 'Drake';
SET @carerid =
(SELECT
   carer_id
   FROM Carer
   WHERE carer_fname = @carerfname
   AND carer_sname = @carersname);

SELECT
   CONCAT(child_fname, ' ', child_sname) AS 'Child Name',
   activity_name,
   activity_day
FROM Child C
JOIN Childactivity CA ON C.child_id = CA.child_id
JOIN Activity A ON A.activity_id = CA.activity_id
WHERE child_carer = @carerid
ORDER BY child_sname, child_fname
```

Testing:

I cross checked the query output, looking at how many children each carer_id was assigned to in the Child table, and running the query with several different carer names.

4. Total amount earned by each activity.

Query:

```
SELECT
   activity_name,
   COUNT(CA.activity_id) AS 'N.Of Children',
   CONCAT('£', activity_fee * COUNT(CA.activity_id)) as 'Total Earned'
FROM Activity A
JOIN Childactivity CA ON A.activity_id = CA.activity_id
GROUP BY CA.activity_id
```

Testing:

I counted the number of times each activity_id appear in Childactivity table and multiply by the activity_fee in Activity table.

2: Referential Integrity

A)

Childactivity is a link table, this means that it links two or more other tables (Child table and Activity table in this case). Both columns in the table are primary and foreign keys and they have to be related to an existing primary key value in the Child and Activity tables. If we try to insert a record that doesn't have a primary key associated we will encounter this error: "ERROR 1452: 1452: Cannot add or update a child row: a foreign key constraint fails". This is due to referential integrity: "**Referential integrity** is a property of data stating that all its references are valid. In the context of relational databases, it requires that if a value of one attribute (column) of a relation (table) references a value of another attribute (either in the same or a different relation), then the referenced value must exist".[i1]

A record can always be deleted, we are just deleting an activity that a child does in a certain day, deletion does not conflict with any referential integrity in this case because Childactivity is a "child" table (it contains foreign keys) and Activity and Child tables are "parent tables" and no other tables have foreign keys associated with Childactivity primary keys.

B)

The Carer table is linked to the Child table by a one to many relationship. Each carer can be associated to one or more child but not the opposite. In this case the "parent" table is Carer and the "child" table is Child. For the principles of referential integrity we cannot delete a record that contains a primary key associated to a foreign key of another table, this will break the relationship.

[1] https://en.wikipedia.org/wiki/Referential_integrity

A carer record can always be inserted as far as we respect the constrains on not null columns. This is the opposite situation of question A.

C)

The “On Delete” option in the “foreign keys” tab let us change the behaviour of the row with a foreign key associated with the parent row which has been deleted. The ‘NO ACTION’ option blocks the deletion of the parent row in case a child row is associated. The ‘SET TO NULL’ option instead set all the child rows associated with the deleted parent row to ‘NULL’.

In our case question A would not have been affected since in case of deletion we wouldn’t drop any parent rows. Question B instead would be affected in case we’d decide to delete a row from Activity or Child tables all the rows in Childactivity table associated with those children or activities would be set to null.

D)

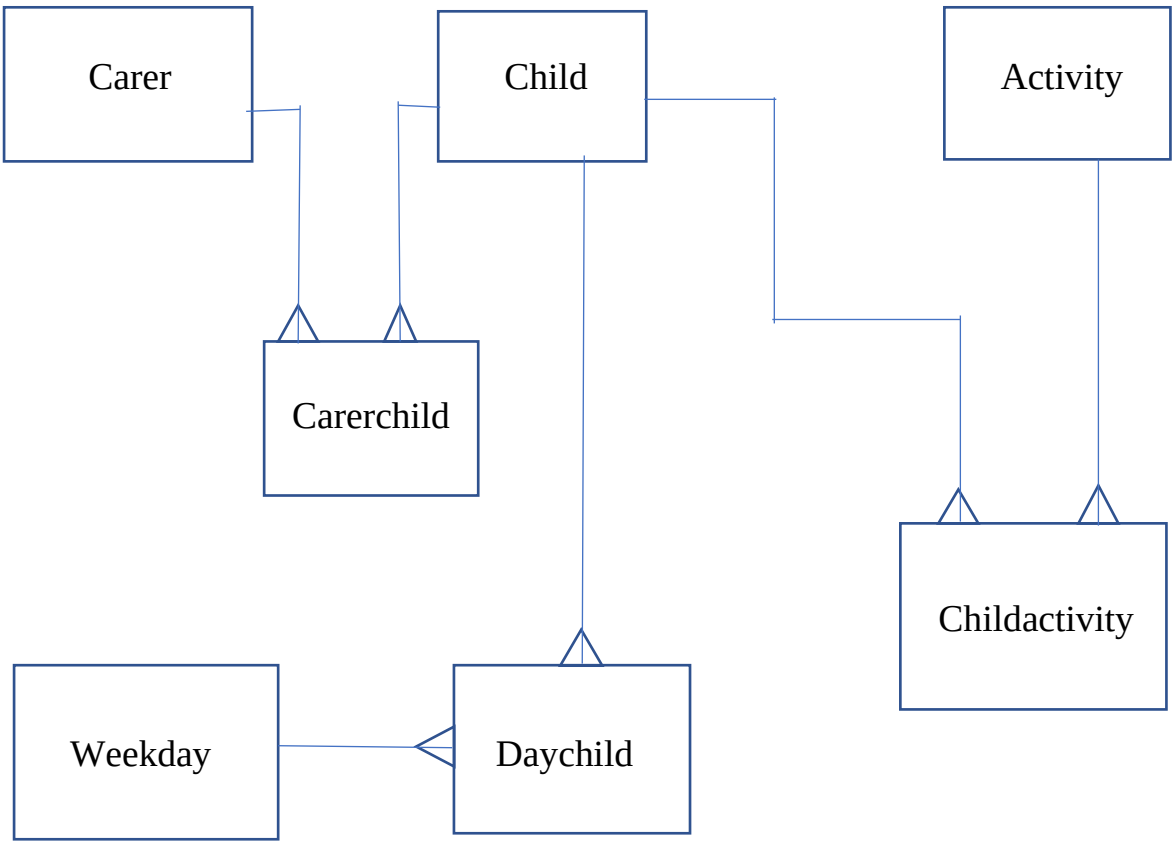
There are 3 foreign keys in the database: 2 in the Childactivity table and 1 in the Child table. The foreign key in the Child table is associated with “carer_id” primary key in the Carer table. Since no child can be registered without an associated carer the best setting to use here is “NO ACTION”. This is because if we would try to delete a carer’s entry in the Carer table we would receive an error message, and this a kind of restriction is something that we want in this case because we don’t want any child without an associated carer.

With the Childactivity foreign keys instead we want something different; setting “On Delete” to “SET NULL” will enable us to delete any entry from the Activity table or Child table without breaking any referential integrity. In case a child will be removed from the database the row in Childactivity will just be set to NULL value without effecting other entries in the table. Same thing in case an activity will be removed from Activity table, all the entries in Childactivity associated to that activity will be set to NULL. This is a quite useful feature

since it allows us to just remove 1 entry in the activity table without manually remove every entry in the Childactivity associated with that activity.

3: Design an extension to the database

To implement the extensions I will create 3 new tables: one table to store days ID and days names, and 2 linking tables. Since the relation between Child and Carer tables is now many-to many the column child_carer from Child table has been deleted and the link between the two table has been provided by a new Carerchild table. The second link table will relate Child and Weekday tables.



New table to store the week days:

Table: Weekday	Column1	Column2
Column	**Datatype**	**Attributes**
day_id	INT	PK, NN, AI, FK to Daychild.day_id
day_name	VARCHAT(3)	NN

New Child table columns (deleted child_carer):

Table: Child	Column1	Column2
Column	**Datatype**	**Attributes**
child_id	INT	PK, NN, AI
child_fname	VARCHAT(20)	NN
child_sname	VARCHAT(20)	NN
child_gender	ENUM('M', 'F')	NN
child_dob	DATE	NN

New link table to relate Child to Weekday:

Table: Daychild	Column1	Column2
Column	**Datatype**	**Attributes**
day_id	INT	PK, NN
child_id	INT	PK, NN

New link table to relate Carer and Child:

Table: Carerchild	Column1	Column2
Column	**Datatype**	**Attributes**
day_id	INT	PK, NN
child_id	INT	PK, NN