Data Integrity

Agenda

- What is Data Integrity
- Constraints on Relation
- Referential Integrity

What is Data Integrity?

According to data management community and data Governance,

Integrity of data ensures an appropriate lineage of business entities in the data flow process.

Data Integrity

- Ensures smooth business flow of business.
- It is the key aspect of design of data flow process.
- It ensures the consistency of the data through entire project life cycle. It means the data should be relevant across the business.

What is Data Integrity?

In order to establish the data lineage, organization should

- Receive ***accurate*** data as if the records are not ignored without any flaws. Hence the loss of data costs to organization.

- Consistency of data flow across business processes.

  So, None of the entities(tables) are not struck in the middle of the business process due invalid or irrelevant data.

- Data generated should be reliable within business units. So that the data is consistent when one business unit shares the data with other business unit.

Loss of Data integrity results in

- Audit penalty by data governance.
- Costs associated to unused data maintenance in billions due to additional storage.
- Consuming of additional RAM memory by SQL programs running on the data.
- Purchase of additional disk space either in standalone or cloud environments.
- Analytics may show inappropriate results.

Loss of Data integrity results in

- Quality control is an overhead
- Inappropriate decision making when multiple strategic reports are generated without any quality of data

Data Integrity is classified as:

1. Row level integrity

2. Column level integrity

3. Referential integrity

Row level integrity

- Each row in the table is referenced by unique values, and avoids conflicts with other rows.
- Unique identifiers are normally known as primary keys in tables.

Example : CUSTOMER table uses single column ***Cust_Id*** as the unique identifier

```
Select * from CUSTOMER where Cust_Id=123001
```

customer_id	customer_name	Address	state_code	Telephone
123001	Oliver	225-5, Emeryville	CA	1897614500

Column level integrity

- Every attribute or Column of a table held a business meaning.
- It consists of metadata means an information of a column. The information of the column explains the meaning and purpose of the column defined in the table and how it is later referenced across the organization.
- In order to achieve this, the data that is stored in a column is the format and definition must be the same.

- That includes its ***data type, size, range*** **of possible values.**

  E.g: Cust_Id data-type is integer which accepts only integer values.

Column level integrity

SELECT * FROM COURSE.

course_id	course_name
10	Python
20	SQL
30	BigData

DESC COURSE

Field	Type	Null	Key
course_id	int	NO	PRI
course_name	varchar(20)	NO	

In this table,
Course_id column will not accept Alphabets because it is defined as an Integer.

insert into course values ('One', 'Python');

```
Error Code: 1366. Incorrect integer value: 'One' for column 'course_id' at row 1
```

Column level integrity - ensures to insert correct values according to its datatype. So the below statement will successfully insert the record.

insert into course values(10,'Python')

Referential Integrity

Referential Integrity -

Ensures consistency of records between two related tables.

It establishes relationships between two different tables using referencing columns.

It sets up a parent child relationship between two tables.

So the child table always references the parent table.

At the same time, the parent table prevents entering a row in child table.

Establishing a Parent - child relationship between tables is achieved by defining the tables using - FOREIGN KEY and PRIMARY KEY.

Defining Foreign Key to show referential Integrity

```
#Set up a Foreign key
create table student(
sid integer primary key,
sname varchar(20) NOT NULL,
course_id integer,
foreign key (course_id)
references course(course_id));
```

Field	Type	Null	Key
sid	int	NO	PRI
sname	varchar(20)	YES	
course_id	int	YES	MUL

COURSE_ID defined as MUL, means the field can accept duplicate values, but the values should already present in its parent Column in Course table

Here Foreign Key **will *not* allow** INSERT or UPDATE a record in the child table for which there is no existing record in parent table.

E.g:

```
insert into student values (2,'John',20);
```

Root cause and resolution

Error : Can’t ADD OR UPDATE a child row:

The COURSE_ID: 20 is not present in COURSE - parent table.

Hence, the STUDENT - child table cannot insert the record with COURSE_ID = 20 to generate a new invoice .

Fix the problem:

Insert a record first in COURSE table to ensure the machine is ready, then you can take a lease .

```
Step1:
INSERT INTO COURSE Values (20,’SQL');
Step2:
INSERT INTO STUDENT VALUES
 (2,’John’,20) ;
```

Records

COURSE

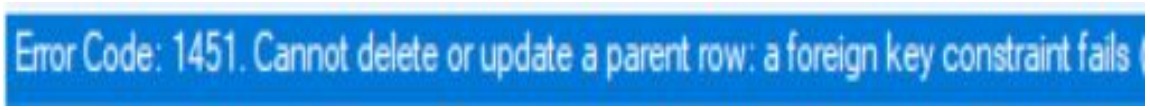
course_id	course_name
10	Python
20	SQL
30	BigData

STUDENT

sid	sname	course_id
1	Ram	10
2	John	20
3	Steve	10

Now let's delete an existing record from COURSE table and see if its corresponding matching records on child table is automatically deleted or not

```
DELETE from COURSE
where course_ID = 20;
```




```
Similarly , You cannot delete a row from Parent table
because child row is existing in a table.
```

Constraints on Single Relation

Constraints on Single Relation

- There are rules that limit what type of data can be put into a column/table using SQL constraints.

- A set of constraints ensures that the data in the column or table is accurate and reliable.

Constraints can be divided into the following two types,

1. Column level constraints: Limits only column level data values
2. Table level constraints: Limits complete table dataset

Constraints on Single Relation

- Following are the most used constraints on Single relationship that are applied to Columns in a single table
- NOT NULL, DEFAULT

Other constraints like below will be discussed later.

- UNIQUE
- PRIMARY KEY
- CHECK

NOT NULL:

- Using the NOT NULL constraint, you prevent a column from having a NULL value.

- After applying NOT NULL constraint to column, you cannot pass a null value into that column

Field	Type	Null	Key
course_id	int	NO	PRI
course_name	varchar(20)	NO	

Course_name is NULL
It means Course_name value must not be blank.

NOT NULL:

```
insert into course values ( 50, NULL);
```

Column Check constraints

CHECK Constraint

- CHECK constraint conditionally validates the column values before insertion of the record, and prevents from any unwanted entry of record.

```
alter table employees add check ( salary>2001);
```

- The table ensures that salary amount is not less than 2001 for any new entry.

Try:

```
insert into employees values(108,'Ashly','Mishra', 'AMISHRA',
'773.453.1489', '1999-12-19','MK_REP', 1300,NULL, 66,19);
```

Error Code: 3819. Check constraint 'employees_chk_1' is violated.

Uniqueness Constraints

Uniqueness Constraints

- UNIQUE constraint ensures a column will only have unique values

  Eg:

```
create table city(city_id char(3) primary key,
city_name varchar(30) UNIQUE);
```

Field	Type	Null	Key
city_id	char(3)	NO	PRI
city_name	varchar(30)	YES	UNI

UNIQUE constraint added to city_name will prevent adding duplicate values .

Uniqueness Constraints

Select * from city

city_id	city_name
BLR	Bangalore
CHN	Chennai

Try :

```
insert into city values('BAN', 'Bangalore');
```

Error Code: 1062. Duplicate entry 'Bangalore' for key 'city.city_name'

PRIMARY KEY:

- Primary key constraint is defined on a column and it uniquely identifies each record in a table.
- A Primary Key by default creates an Index on a column. An index key is used for search records based on conditions.

PRIMARY KEY:

Select * from COURSE

course_id	course_name
10	Python
20	SQL
30	BigData

Desc COURSE

Field	Type	Null	Key
course_id	int	NO	PRI
course_name	varchar(20)	NO	

Try :

insert into course values (30, 'Hive');

Error Code: 1062. Duplicate entry '30' for key 'course.PRIMARY'

Referential Integrity Problems

Referential Integrity Problems

- Enforcing referential integrity will cause limitations to its referenced tables on DML operations such as Insert, Delete and Update
- Referential integrity constraints decreases the performance of DML operations as the SQL engine should refers to both the table which is being updated and as well as referenced tables

Referential Integrity Problems

- Referential integrities are overhead when exporting the tables from one database to other database
- Problems like foreign key should be disabled and enabled after exporting
- re-enabling foreign keys lost data integrity as it will not scan appropriately

Delete and Update Rules

Delete and Update Rules

- Recapping :-

- Foreign key constraints always references a primary or unique key

- A foreign key value when assigned with NULL doesn’t references any primary key value in another table

- But if foreign key has any valid value, then it must have an associated value in primary key

- During any DML operation on Parent tables , there are different rules that effect on associated values in child tables.

Rules on Updating Foreign Key

- When updating foreign key column with a value that is not present in primary key column, it violates the referential integrity

Desc COURSE

Field	Type	Null	Key
course_id	int	NO	PRI
course_name	varchar(20)	NO	

Desc STUDENT

Field	Type	Null	Key
sid	int	NO	PRI
sname	varchar(20)	YES	
course_id	int	YES	MUL

Foreign Key (MUL) in STUDENT table is referring Primary Key in COURSE .

Rules on Updating Foreign Key

- *Primary Key table:*

```
SELECT * FROM COURSE WHERE COURSE_ID = 10
```

course_id	course_name
10	Python

- *Foreign key table*

```
SELECT * FROM STUDENT WHERE COURSE_ID = 10
```

sid	sname	course_id
1	Ram	10
3	Steve	10

Now, try updating the Foreign Key column value

Trying to update Foreign key

Try : ***update existing*** course **_ID =** 100 **in COURSE table.**

Update course set course_id = 100 where course_id = 10;

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`hr_emp`.`student`, CONSTRAINT `

Rules on deleting Primary Key records

- Delete of a record in a parent table that is referenced by child's foreign key columns
- However, delete of a record in child table doesn't effect in parent table
- Trying to delete parent table record that is being referenced by foreign key

Try:

```
delete from course where course_id = 20;
```

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`hr_emp`.`student`, CONSTRAINT `

Cascaded, Deletes and Updates

Cascading:

- To overwrite the DML rules on referential integrity fields, CASCADE is issued on the table having foreign key column
- So that changes to parent table records are automatically reflected in child table

Cascading can be done for two types of DML statements

- UPDATE CASCADE
- DELETE CASCADE

ON UPDATE CASCADE

Before CASCADE

```
Update course set course_id = 100 where course_id = 10;
```

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`hr_emp`.`student`, CONSTRAINT `

```
create table student
(sid integer primary key ,
sname varchar(20),
course_id integer,
foreign key (course_id) references course(course_id)
on delete cascade
on update cascade);
```

ON UPDATE CASCADE

Before Update

```
SELECT * FROM STUDENT WHERE COURSE_ID = 10
```

sid	sname	course_id
1	Ram	10
3	Steve	10

```
/* Update */
   Update course set course_id = 100 where course_id = 10;
```

sid	sname	course_id
1	Ram	100
3	Steve	100

ON DELETE CASCADE

Before CASCADE

```
delete from course where course_id = 20;
```

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`hr_emp`.`student`, CONSTRAINT `

ON UPDATE CASCADE

```
SELECT * FROM COURSE WHERE course_id = 20;
```

course_id	course_name
20	SQL

```
SELECT * FROM STUDENT WHERE course_id = 20;
```

sid	sname	course_id
2	John	20
4	Rishab	20

```
/* Delete */
delete from course where course_id = 20;

SELECT count(*) FROM student  WHERE course_id = 20;
```

count(*)
0

Summary

This deck explains the data integrity and how it works, its constraints on relations queries and the referential integrity. Here we generally, explained that how data integrity maintains the accuracy and consistency amongst the table in SQL.

Each condition is explained using a query and its output.

Thank You