W23-CSC 430/530 (Jan 23th, 2023)
Lecture 13 - Violation of Integrity Constraints

§ Operations that violate the Integrity of a DB:

a) Deletion: the deletion operation can violate the referential Integrity when

Rel1 Rel2

p.k ⟶ f.k.

(X) ⟶ orphan

Soln: a) Reject the operation
b) Cascade to foreignkey(fk) & delete the f.k.

b) Insertion/Update: these operations can violate

(*) the referential Integrity constraint
i.e. if f.k is inserted into a table, it should be a p.k. in another table.

soh: a) Reject the operation.

(*) It can also violate the domain constraint

Soh: a) Reject the operation (or)
b) Update the domain of an attribute
ALTER TABLE <NAME> MODIFY <ATTRIBUTE> NEW-DOMAIN;

(*) It can also violate the Entity integrity
i.e. pk. cannot be null.

Soh: a) Reject the operation

(*) Key constraint i.e. p.k. is not unique

5 Integrity Checks:

(*) Operations of a) Insertions; b) Deletions; (c) Updates require constant checks for consistency.

(*) Its important during Recovery from failure & concurrent operations.

Therefore a DBMS should support tools / formal mechanisms so that checks can be made effectively.

- → correct detection
- → quick (efficient)

a) formal mechanism: The E-R model introduces and ensures integrity constraints.

* KEY declarations: (primary key, foreign key)
* Cardinality constraints: (1:1, 1:N, N:M)
* Types of entities: (Strong Vs Weak)

b) Relational Modeling: here the DBMS introduces the role of predicate rules (rules in SQL) for the DB created.

* predicate rules must always hold.
* predicate rules are invoked when a DB is archived. (periodically)

- However these predicate rules and their enforcement are time consuming.
  - especially while updating, inserting, or deleting data.

Objective therefore is that we want to conduct more number of checks easily and leave the more complex checks for the predicate rules by the system.

- Useful schemes for specifying & checking integrity constraints on a relational model
  - Domain constraint
  - Referential integrity constraint
  - Functional Dependencies.

Eg: BOOK (ACC_NO, YR-PUB, TITLE)
USER (CARD_NO, B-NAME, B-ADDR)
SUPPLIER (S_NAME, S:ADDR)
BORROW (ACC_NO, CARD_NO, DOI)
SUPPLY (ACC_NO, S_NAME, PRICE, DOS)

a) Domain Constraints:
- Definition of domains of each attribute
- DOI DATETYPE dd-MM-YYYY
  DOI ≥ 01/01/2019 → check for constraint
- SALARY (Name, DEPT, GROSS_SAL, DEDUCTIONS, NET_SAL)
  Here GROSS_SAL = DEDUCTIONS + NET_SAL must always hold.

Referential Integrity constraints:

case 1:

BOOK (ACC_NO, YR-PUB, TITLE)

BORROW (ACC_NO, CARD-NO, DOI)

There is a tuple with ACC_NO = 57326 in Borrow but no tuple with ACC_NO = 57326 in Book.

* this situation is INCONSISTENT

case 2: There is a ~~tuple~~ tuple with ACC_NO = 57326 in Book, but no tuple with ACC_NO = 57326 in Borrow — CONSISTENT

Illy with: SUP & SUPPLIER; BORROW & USER(card #)
(S_NAME)

Formal definition: Referential Integrity or subset Dependency

Let $r_1(R_1)$ & $r_2(R_2)$ be relations

suppose $\alpha \subseteq R_2$ is a foreign key referencing the primary key K_1 (the primary key of r_1)

IF: $\Pi_\alpha (r_2) \subseteq \Pi_{K_1} (r_1)$; then Referential Integrity HOLDS.

how this is maintained in SQL:

```
CREATE TABLE BOOK (
        ACC_NO    CHAR(10)    NOT NULL
        YR_PUB    DATE
        TITLE     CHAR(10)    NOT NULL.
        PRIMARY KEY (ACC_NO))
```

```
CREATE TABLE   BORROW
( ACC_NO      CHAR(10)    NOT NULL,
  CARD_NO     CHAR(5)     NOT NULL,
  DOI                     DATE,
  PRIMARY KEY (ACC_NO, CARD_NO)
  FOREIGN KEY (ACC_NO) REFRENCES BOOK(ACC_NO)
  FOREIGN KEY (CARD_NO) REFRENCES USER(CARD_NO) );
```

Database Modification:

CONSTRAINT: $\Pi_{\alpha}(r_2) \subseteq \Pi_{K}(r_1)$

* INSERTION: if t_2 is inserted in r_2
  ensure that $t_2[\alpha] \in \Pi_K(r_1)$

* DELETION: if t_1 is deleted for r_1,
  we must ensure that
  $\sigma_{\alpha = t_1[K]}(r_2)$ is empty.

* Updation: (i) if t_2 is updated in r_2 and update modifies values for foreign key α then check is similar to insert

(ii) if t_1 is update in r_1 and update modifies values of primary key K, then check is similar to deletion

SQL: CASCADE ON: (DELETE & UPDATE)

```
CREATE TABLE BORROW ( ACC_NO   CHAR(10) PRIMARY KEY,
                      CARD_NO  CHAR(5)  PRIMARY KEY,
                      DOI      DATE,
                      FOREIGN KEY (ACC_NO) REFRENCES BOOK(ACC_NO)
ON UPDATE CASCADE ON DELETE CASCADE,
                      FOREIGN KEY (CARD_NO) REFRENCES USER(CARD_NO)
ON UPDATE CASCADE ON DELETE CASCADE );
```

ALTER TABLE BOOKS

ADD CONSTRAINTS STORE-NAME

FOREIGN KEY (STORE_ID) REFERENCES STORE (STORE_ID)

ON DELETE CASCADE

ON UPDATE CASCADE

ALTER TABLE BOOKS

ADD CONSTRAINTS BOOK_GENRE_ID

FOREIGN KEY (BOOK-GENRE_ID) REFERENCES GENRE(BOOK GENRE-ID)

ON DELETE SET NULL
ON UPDATE CASCADE;

ALTER TABLE BOOKS

DROP FOREIGN KEY <.ATTRIBUTE.>

ALTER TABLE BOOK

ADD CONSTRAINTS BOOK_GENRE-ID

FOREIGN KEY (BOOK-GENRE-ID) REFERENCES GENRE(BOOK-GENRE-ID)

ON DELETE RESTRICT

Why ON DELETE NO ACTION.