Views

View provides mechanism to hide certain data from the view of certain users.

Definition: A relation which is not conceptual model but it is made visible to a user as a virtual relation is called view.

- view causes the saving of creating a new relation by evaluting query expressions.

Example:
```
CREATE VIEW Faculty AS
    SELECT ID, name, dept-name
    FROM Instructor;
```

- A view relation is said to be recursive if it depend on itself.

```
Create VIEW phy-fall-17 AS
SELECT c.courseid, s.secid, building
FROM course, section
WHERE c.courseid = s.courseid
   AND c.dept-name = 'Physics'
   AND s.semester = 'Fall'
   AND s.year= '2017';
```

```
CREATE VIEW
   phy-fall-17_watson
AS (SELECT c-id, roomno
   FROM phy-fall-17
   WHERE building=
       'Watson');
```

Materialized view : A view which is physical copy is created when the view was defined. If relations in query are updated, The materialized view becomes out of date. So, we need to maintain the view by updating view according to query.

* Update করতে হলে Real Table এ কোন value না থাকলেও Null হিসেবে insert করতে হবে নাহলে view update হবে না, row insert হবে না Real table এ.

* Cross Join থাকলে Update issues হবে

* WHERE এর condition satisfy না করলে issues হবে

So, SQL এ view update হয় simple view তে

(I) From এ একটা relation থাকে

(II) select এ aggregation, distinct থাকবে না

(III) SELECT এ যদি কোন attribute না থাকে সেটা null set করবো,

(IV) Group by, Having থাকবে না

Transaction:

It ~~a~~ is a 'unit' of work (atomic transaction).

It is a sequence of query and/or update statement.

It must end with following statements:-

(I) Commit work (II) Rollback work

(I) Commit work: The updates performed by transactio become permanent in the database.

(II) Roll back work: All updates performed by SQL statement in the transaction are undone.

- Atomic transaction is either fully executed or rolled back as it never occurred.

Integrity Constraints

Integrity constraints guard against accidental damage to the database by ensuring that authorized changes to the database do not result in a loss of data consistency.

Constraints: (I) NOT NULL (III) UNIQUE

(II) Primary key (IV) CHECK (P)

(must be satisfied by every tuple in a relation)

Referential Integrity: It ensures that a value appears in one relation also appear for a certain set of attributes in another relation.

FOREIGN KEY dept_name REFERENCES dept;

Cascading actions in Referencial Integrity

- ON DELETE CASCADE
- ON UPDATE CASCADE

Instead cascade we can use set null, set default

Integrity constraint violation During Transactions.

```
CREATE TABLE person (
    |  |  |  |
    |  |  |  |
FOREIGN KEY father references person,
   "    "   mother    "         "    )
```

Solution: Insert father, mother in person, than insert person. OR, set father, mother NULL initially

ASSERTION

a peridate expressing a condition that we wish the database to alwaye satisfy. It is used to prevent invalid data entry.

```
CREAT ASSERTION <name> CHECK (<predi>)
```

BUILT IN Types in SQL

(I) date
(II) time
(III) timestamp (date, time)
(IV) Interval (subtracting date/time/timestamp)

Large object types

blob: binary large object
clob: character large object

$\Big\}$ photos, videos, CAD files

User defined type

Create type Dollars AS NUMERIC (12, 2) FINAL;

DOMAINS

CREATE DOMAIN person-name CHAR(20) NOT NULL;

INDEX

CREATE INDEX <name> ON <relation-name>(attribute);
CREATE INDEX studentID-Index ON Student(ID).

Authorization

Read, Insert, Update, Delete

Forms of authorization: Index, Resources, Alteration, Drop

grant select on department to Amit;

$$\begin{tikzcd}
\text{select} \arrow[d] & & \text{Amit} \arrow[d] \\
\text{privilige} \arrow[d] \arrow[dl] \arrow[dr] & & \text{user} \\
\text{Insert} & \text{Delete} & \text{Update}
\end{tikzcd}$$

Revoke select on student from U_1, U_2, U_3;

Roles

CREATE ROLE Instructor;
GRANT Instructor TO Amit;
GRANT SELECT ON takes TO Instructor;

GRANT select ON department TO Amit WITH GRANT OPTION;
REVOKE " " " FROM " RESTRICT;
" " " " " Amit, Satoshi CASCADE;