Project details
Attendance

COL 362 & COL 632

Constraints and Views, Indexes, ACL in SQL

31 Jan 2023

Constraints for DB Consistency

- Security Contraints
  - Access control mechanisms
  - Restricted views
  - Prevent unauthorized access to relations and attributes
  - *Eg. A student may not be allowed to see records of another student,*
- 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
  - *A current account must have a monthly balance greater than Rs. 5,000*
  - *Minimum salary of an employee with graduate degree in Delhi should be Rs. 21,000*
  - *Every faculty member must have a (non-null) phone number*

Constraints on Single Relation

- **not null**
- **primary key**
- **unique**
- **check** (P), where P is a predicate

Specified as part of the relation declaration

Constraints on Single Relation

- **not null**
  - Declare name and budget to be not null (as part of the relation defn.)

```
name varchar(20) not null
budget numeric(12,2) not null
```

- **primary key**
- **unique**
- **check** (P), where P is a predicate

Constraints on Single Relation

- **not null**
- **primary key**
  - **Non-null-able candidate key**
- **unique**
- **check** (P), where P is a predicate

Constraints on Single Relation

- **not null**
- **primary key**
- **Unique**
  unique (A1, A2, …, Am)
  - The unique specification states that the attributes A1, A2, …, Am form **a candidate key**
  - Candidate keys are permitted to be null (in contrast to primary keys).
- **check** (P), where P is a predicate

Constraints on Single Relation

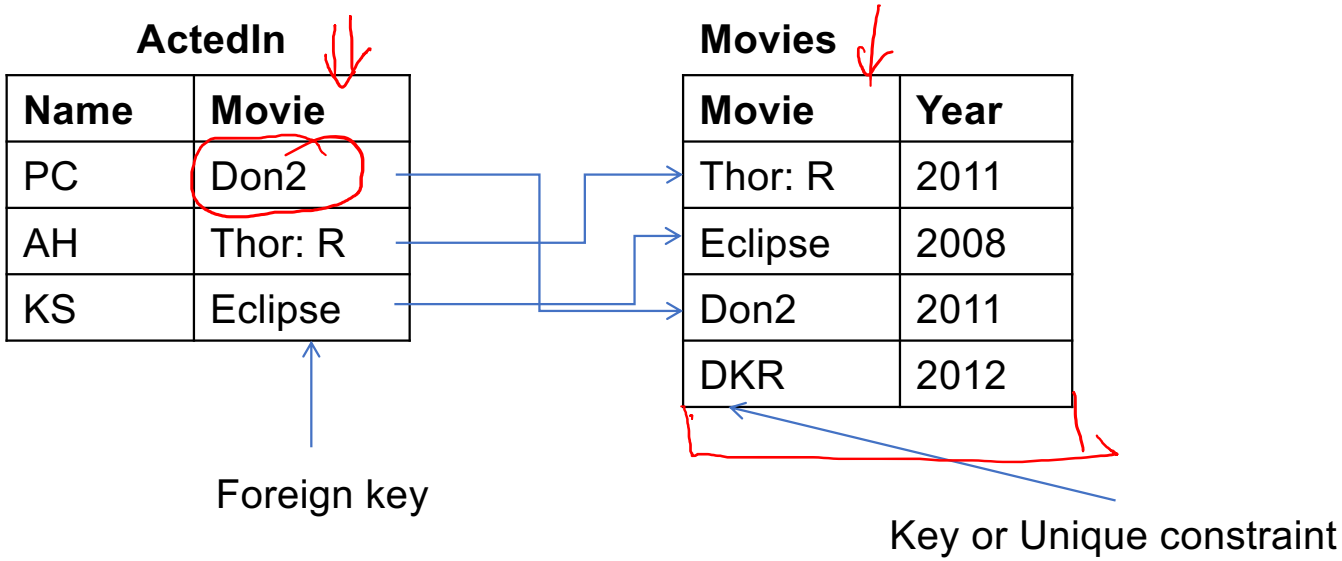
- **check** (P), where P is a predicate

The **check** (P) clause specifies a predicate P that must be satisfied by every tuple in a relation.

Example: ensure that semester is one of fall, winter, spring or summer

```
create table section
     (course_id varchar (8),
      sec_id varchar (8),
      semester varchar (6),
      year numeric (4,0),
      building varchar (15),
      room_number varchar (7),
      time_slot_id varchar (4),
      primary key (course_id, sec_id, semester, year),
      check (semester in ('Fall', 'Winter', 'Spring', 'Summer')))
```

Referential Integrity (1/2)




```
CREATE TABLE ActedIn (
     Name varchar(30), Movie varchar(30),
     CONSTRAINT actedIn_fkey
         FOREIGN KEY (Movie) REFERENCES Movies (Movie));
```

Referential Integrity (2/2)

- Reject, Cascade, Set-null

ActedIn

Name	Movie
PC	Don2
AH	Thor: R
KS	Eclipse
CB	DKR

Movies

Movie	Year
Thor: R	2011
Eclipse	2008
Don2	2011
DKR	2012

```
CREATE TABLE ActedIn (
     Name varchar(30), Movie varchar(30),
     CONSTRAINT actedIn_fkey
          FOREIGN KEY (Movie) REFERENCES Movie (movieid)
          ON DELETE CASCADE)
```

Reject modifications which violate constraints

"Transfer" modifications

Set attribute(s) to null if needed

insert into actedIn: ('CB', 'DKR')
insert into actedIn: ('AH', 'Thor: Ragnarok')

delete from actedIn: ('KS', 'Eclipse')
delete from Movie: ('DKR', 2012)
delete from Movie: ('Eclipse', 2008)

update Movie: ('DKR', 2012) to ('DK', 2011)

“Cyclic” Constraints (1/3)

Actors

Name	Age	Addr	Famous_ Movie
Priyanka Chopra	36	Mumbai	Don-II
Anthony Hopkins	81	LA	Thor: R
Bill Nighy	69	LA	Valkyrie
Abhishek Bachchan	42	Mumbai	Raavan

Movies

Name	Year	Title
Priyanka Chopra	2011	Don-II
Anthony Hopkins	2011	Thor: R
Bill Nighy	2009	Valkyrie
Abhishek Bachchan	2010	Raavan
Anthony Hopkins	2003	TLS

insert into Actors: (‘Kristen Stewart’, 23, ‘LA’, ‘Breaking Dawn’);
insert into Movies: (‘Kristen Stewart’, 2011, ‘Breaking Dawn’);

“Cyclic” Constraints (2/3)

- Notion of a **Transaction**
  - An atomic unit of execution
    The state of the database is *consistent* before and after a successful completion of a transaction
  - Currently, the two inserts together form a single transaction

insert into Actors: (‘Kristen Stewart’, 23, ‘LA’, ‘Breaking Dawn’);
insert into Movies: (‘Kristen Stewart’, 2011, ‘Breaking Dawn’);

Not check
Not check
check here

- **Defer constraint checking until after transaction**

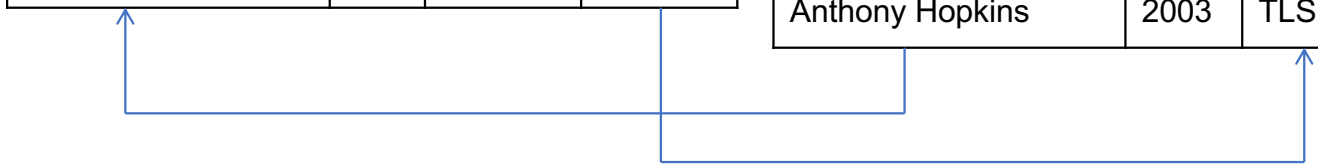
“Cyclic” Constraints (3/3)

Actors

Name	Age	Addr	Famous_Movie
Priyanka Chopra	36	Mumbai	Don-II
Anthony Hopkins	81	LA	Thor: R
Bill Nighy	69	LA	Valkyrie
Abhishek Bachchan	42	Mumbai	Raavan

Movies

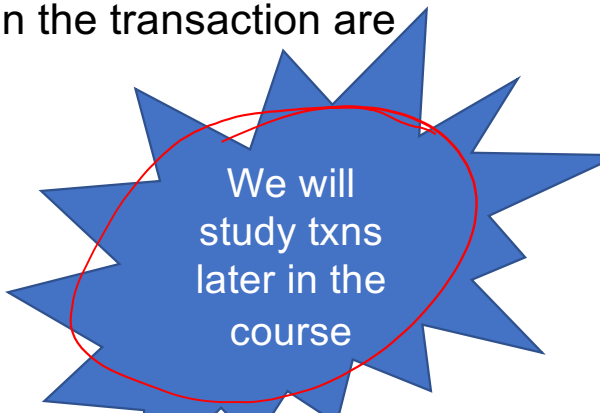
Name	Year	Title
Priyanka Chopra	2011	Don-II
Anthony Hopkins	2011	Thor: R
Bill Nighy	2009	Valkyrie
Abhishek Bachchan	2010	Raavan
Anthony Hopkins	2003	TLS



```
CREATE TABLE Actors (
    Name varchar(30), Age int, Addr varchar (30), Famous_Movie varchar(30),
    CONSTRAINT actors_fkey
        FOREIGN KEY (Famous_Movie) REFERENCES Movies (Title)
        DEFERRABLE INITIALLY DEFERRED)
```

Transactions

- A **transaction** consists of a sequence of query and/or update statements and is a “unit” of work
- The SQL standard specifies that a transaction begins implicitly when an SQL statement is executed.
- The transaction must end with one of the following statements:
  - **Commit work**. The updates performed by the transaction become permanent in the database.
  - **Rollback work**. All the updates performed by the SQL statements in the transaction are undone.
- Atomic transaction
  - either fully executed or rolled back as if it never occurred
- Isolation from concurrent transactions



Few other important features in SQL

- Indexes
- Views
- Authorization / ACL

Few other important features in SQL

- Indexes
- Views
- Authorization / ACL

- Data structure to access specific tuples “fast”
- Very important for query processing and query optimization
- Useful when no. of results *very small* compared to the total no. of tuples

```
SELECT * FROM Students
WHERE Name = ‘Rana Prathap’;
```

Likely to be just one result ☺

```
CREATE INDEX NameIndex
ON Actor (Name)
```

Few other important features in SQL

- Indexes
- Views
- Authorization / ACL

- Views are tables created from existing tables
  - Materialized (physically exist)
    - `CREATE MATERIALIZED VIEW`
  - Virtual (don't physically exist)
    - `CREATE VIEW`
- Offer a "simplified" view of the data
  - **Secure** data from non-authorized users

```
CREATE VIEW Teachers4Students AS
SELECT F_NAME, L_NAME, AGE, OFFICE
FROM Teachers WHERE DEPT = 'CSE';
--- we have not retrieved their salary
```

- Materialized views used in speeding up query processing

- How to update views when base tables are updated?
- How to propagate changes when views are updated?

Few other important features in SQL

- Indexes
- Views
- Authorization / ACL

- Nearly all database systems provide authorization and access control mechanisms for the databases
- Users: individual ”login”s to the database system
- Roles: functional roles defined by the DBA and users are assigned to these roles

Other topics that may be useful for your projects

- JDBC/ODBC
- Triggers
- SQL functions and procedures
- Order-by, stop-after, windowing and rollups