# Database Management Systems

Module 14: Intermediate SQL/3

## Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

*ppd@cse.iitkgp.ac.in*

# Module Recap

- SQL expressions for Join and Views

# Module Objectives

- To understand Transactions
- To learn SQL expressions for Integrity Constraints
- To understand more Data Types in SQL
- To understand Authorization in SQL

- Transactions
- Integrity Constraints
- SQL Data Types and Schemas
- Authorization

# Transactions

- Unit of work
- Atomic transaction
  - either fully executed or rolled back as if it never occurred
- Isolation from concurrent transactions
- Transactions begin implicitly
  - Ended by **commit work** or **rollback work**
- But default on most databases: each SQL statement commits automatically
  - Can turn off auto commit for a session (for example, using API)
  - In SQL:1999, can use: **begin atomic … end**
    - ▷ Not supported on most databases

# Integrity Constraints

# 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
  - A checking account must have a balance greater than Rs. 10,000.00
  - A salary of a bank employee must be at least Rs. 250.00 an hour
  - A customer must have a (non-null) phone number

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

- **not null**
    - Declare *name* and *budget* to be **not null**
        *name* **varchar**(20) **not null**
        *budget* **numeric**(12,2) **not null**
- **unique** $(A_1, A_2, \ldots, A_m)$
    - The unique specification states that the attributes $A_1, A_2, \ldots, A_m$ form a candidate key
    - Candidate keys are permitted to be null (in contrast to primary keys).

# The check clause

Module 14

Partha Pratim
Das

Objectives &
Outline

Transactions

Integrity
Constraints

Referential Integrity

SQL Data Types
and Schemas

Built-in Types

Index

UDT

Domains

Large Object

Authorization

Privileges

Revocation

Roles

Module Summary

- **check**(P), where P is a predicate
- 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'))
);
```

- Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation
- Example: If "Biology" is a department name appearing in one of the tuples in the instructor relation, then there exists a tuple in the *department* relation for "Biology"
- Let A be a set of attributes. Let R and S be two relations that contain attributes A and where A is the primary key of S. A is said to be a **foreign key** of R if for any values of A appearing in R these values also appear in S

- With cascading, you can define the actions that the Database Engine takes when a user tries to delete or update a key to which existing foreign keys point

- **create table** *course* (
  - *course_id* **char**(5) **primary key**,
  - *title* **varchar**(20),
  - *dept_name* **varchar**(20) **references** *department*
  )

- **create table** *course* (
  - . . .
  - *dept_name* **varchar**(20),
  - **foreign key** (*dept_name*) **references** *department*
    - **on delete cascade**
    - **on update cascade**,
  - . . .
  )

- Alternative actions to cascade: **no action, set null, set default**

- **create table** *person* (
    *ID* **char**(10),
    *name* **char**(40),
    *mother* **char**(10),
    *father* **char**(10),
    **primary key** *ID*,
    **foreign key** *father* **references** *person*,
    **foreign key** *mother* **references** *person*)

- How to insert a tuple without causing constraint violation?
    - Insert father and mother of a person before inserting person
    - OR, Set father and mother to null initially, update after inserting all persons (not possible if father and mother attributes declared to be **not null**)
    - OR Defer constraint checking (will discuss later)

# SQL Data Types and Schemas

# Built-in Data Types in SQL

Module 14

Partha Pratim Das

Objectives & Outline

Transactions

Integrity Constraints
Referential Integrity

SQL Data Types and Schemas
Built-in Types
Index
UDT
Domains
Large Object

Authorization
Privileges
Revocation
Roles

Module Summary

- **date**: Dates, containing a (4 digit) year, month and date
  - Example: **date** '2005-7-27'
- **time**: Time of day, in hours, minutes and seconds.
  - Example: **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: date plus time of day
  - Example: **timestamp** '2005-7-27 09:00:30.75'
- **interval**: period of time
  - Example: **interval** '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values

IIT Madras
BSc Degree

Index Creation

Module 14

Partha Pratim
Das

Objectives &
Outline

Transactions

Integrity
Constraints

Referential Integrity

SQL Data Types
and Schemas

Built-in Types

Index

UDT

Domains

Large Object

Authorization

Privileges

Revocation

Roles

Module Summary

- **create table** *student*
        (*ID* **varchar**(5),
        *name* **varchar**(20) **not null**,
        *dept_name* **varchar**(20),
        *tot_cred* **numeric** (3,0) **default** 0,
        **primary key** (*ID*))
- **create index** *studentID_index* **on** *student(ID)*
- Indices are data structures used to speed up access to records with specified values for index attributes
        **select \***
        **from** *student*
        **where** *ID* = '12345'
  ○ Can be executed by using the index to find the required record, without looking at all records of student
  ○ *More on indices in Chapter 9*

Module 14

Partha Pratim
Das

Objectives &
Outline

Transactions

Integrity
Constraints

Referential Integrity

SQL Data Types
and Schemas

Built-in Types

Index

UDT

Domains

Large Object

Authorization

Privileges

Revocation

Roles

Module Summary

# User-Defined Types

- **create type** construct in SQL creates user-defined type (alias, like typedef in C)
  - **create type** *Dollars* **as numeric** (12,2) **final**
  - **create table** *department* (
    - *dept_name* **varchar** (20),
    - *building* **varchar** (15),
    - *budget Dollars*);

# Domains

- **create domain** construct in SQL-92 creates user-defined domain types
        **create domain** *person_name* **char**(20) **not null**
- Types and domains are similar
- Domains can have constraints, such as **not null**, specified on them
        **create domain** *degree_level* **varchar**(10)
        **constraint** *degree_level_test*
        **check** (**value in** ('Bachelors', 'Masters', 'Doctorate'));

# Large-Object Types

- Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:
  - **blob**: binary large object – object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)
  - **clob**: character large object – object is a large collection of character data
  - When a query returns a large object, a pointer is returned rather than the large object itself

# Authorization

- Forms of authorization on parts of the database:
  - **Read** - allows reading, but not modification of data
  - **Insert** - allows insertion of new data, but not modification of existing data
  - **Update** - allows modification, but not deletion of data
  - **Delete** - allows deletion of data
- Forms of authorization to modify the database schema
  - **Index** - allows creation and deletion of indices
  - **Resources** - allows creation of new relations
  - **Alteration** - allows addition or deletion of attributes in a relation
  - **Drop** - allows deletion of relations

- The **grant** statement is used to confer authorization
    **grant** <privilege list>
    **on** <relation name or view name> **to** <user list>
- <user list> is:
  - a user-id
  - **public**, which allows all valid users the privilege granted
  - A role (more on this later)
- Granting a privilege on a view does not imply granting any privileges on the underlying relations
- The grantor of the privilege must already hold the privilege on the specified item (or be the database administrator)

# Privileges in SQL

- **select**: allows read access to relation, or the ability to query using the view
  - Example: grant users $U_1$, $U_2$, and $U_3$ **select** authorization on the *instructor* relation:
        **grant select on** *instructor* **to** $U_1$, $U_2$, $U_3$
- **insert**: the ability to insert tuples
- **update**: the ability to update using the SQL update statement
- **delete**: the ability to delete tuples.
- **all privileges**: used as a short form for all the allowable privileges

- The **revoke** statement is used to revoke authorization
   **revoke** <privilege list>
   **on** <relation name or view name> **from** <user list>
- Example:
   **revoke select on** *branch* **from** $U_1, U_2, U_3$
- <privilege-list> may be **all** to revoke all privileges the revokee may hold
- If <revokee-list> includes **public**, all users lose the privilege except those granted it explicitly
- If the same privilege was granted twice to the same user by different grantees, the user may retain the privilege after the revocation
- All privileges that depend on the privilege being revoked are also revoked

- **create role** *instructor*;
  - **grant** *instructor* **to** Amit;
- Privileges can be granted to roles:
  - **grant select on** takes **to** *instructor*;
- Roles can be granted to users, as well as to other roles
  - **create role** *teaching_assistant*
  - **grant** *teaching_assistant* **to** *instructor*;
  - ○ *Instructor* inherits all privileges of *teaching_assistant*
- Chain of roles
  - ○ **create role** dean;
  - ○ **grant** *instructor* **to** dean;
  - ○ **grant** *dean* **to** Satoshi;

IIT Madras
BSc Degree

Authorization on Views

Module 14

Partha Pratim
Das

Objectives &
Outline

Transactions

Integrity
Constraints

Referential Integrity

SQL Data Types
and Schemas

Built-in Types

Index

UDT

Domains

Large Object

Authorization

Privileges

Revocation

Roles

Module Summary

- **create view** *geo_instructor* **as**
  (**select \***
  **from** *instructor*
  **where** *dept_name* = 'Geology');
  **grant select on** *geo_instructor* **to** *geo_staff*
- Suppose that a *geo_staff* member issues
  **select \***
  **from** *geo_instructor*;
- What if
  - *geo_staff* does not have permissions on *instructor*?
  - creator of view did not have some permissions on *instructor*?

- **references** privilege to create foreign key
  **grant reference** (*dept_name*) **on** *department* **to** Mariano;
  - why is this required?
- Transfer of privileges
  - **grant select on** *department* **to** Amit **with grant option**;
  - **revoke select on** *department* **from** Amit, Satoshi **cascade**;
  - **revoke select on** *department* **from** Amit, Satoshi **restrict**;

- Introduced transactions
- Learnt SQL expressions for integrity constraints
- Familiarized with more data types in SQL
- Discussed authorization in SQL

**Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.
Edited and new slides are marked with "PPD".**