5.3: Creating SQL Tables

IT2306 – Database Systems I

Level I - Semester 2

Detailed Syllabus

- 5.3.1 Defining tables:
  - CREATE TABLE, ALTER TABLE, DROP TABLE.
- 5.3.2 Specifying integrity constraints:
  - PRIMARY KEY, UNIQUE, NOT NULL, CHECK, Referential Integrity constraints (Cyclic, Self-referencing, Multiple path) FOREIGN KEY (CASCADE, RESTRICT, NULL, DEFAULT.)
- 5.3.3 Creating indexes:
  - CREATE INDEX, DROP INDEX.

Data Manipulation using SQL

- Define a relational database schema in SQL

- Early versions did not include this concept of a relational database schema, all tables were considered part of the same schema.
- This concept is used in SQL2 to group tables and other constructs that belong to the same database application.

SQL Schema

- A SQL schema is composed of
  - A **schema name**
  - **Authorization identifier** (To indicate user-account who own the schema)
  - **Descriptors for Schema elements**
    (tables, constraints, views, domains, etc…)

- A schema is created via the **CREATE SCHEMA** statement, which can include all the schema elements' definitions
- Alternatively, the schema can be assigned a name and authorization identifier, and the elements can be defined later
  - Ex: CREATE SCHEMA COMPANY AUTHORIZATION 'jsmith';

- CREATE TABLE COMPANY.EMPLOYEE
  - Will make the employee table part of the company schema.

SQL Catalog

- SQL Catalog is a **named collection of schemas**.
- Contains a special schema called **INFORMATION_SCHEMA**
  - provides information on all the schemas in the catalog and all the element descriptors in these schemas.
- Integrity constraints such as referential integrity can be defined between relations only if they exist in schemas within the same catalog.
- Schemas within the same catalog can also share certain elements, such as type and domain definitions.

Data Manipulation using SQL

- Explain basic elements in the structure of an SQL information schema

- SQL Basics
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
  - Data Control Language (DCL)

SQL Basics

Data Definition Language (DDL)

CREATE TABLE	Adds new table
DROP TABLE	Removes existing tables
ALTER TABLE	Modifies structure of tables
CREATE VIEW	Adds a new view
DROP VIEW	Removes a view
CREATE INDEX	Build an index for a column
DROP INDEX	Removes an index
CREATE SYNONYM	Defines an alias for a database object
DROP SYNONYM	Remove an alias
COMMENTS	Describes a table or column
LABEL	Defines a title for a table or column

DDL defines the database: Physical Design

SQL Basics

Data Manipulation Language(DML)

SELECT	Retrieves data
INSERT	Adds new rows of data
DELETE	Removes row of data
UPDATE	Modifies existing data

More Data Manipulation

DECLARE	Defines a cursor for query
EXPLAIN	Describes data access for a query
OPEN	Opens a cursor to retrieve query results
FETCH	Retrieves a row of query
CLOSE	Closes a cursor

DML load the database: Implementation

SQL Basics

Data Control Language(DCL)

GRANT	Gives user access privileges
REVOKE	Removes privileges
COMMIT	Ends current transaction
ROLLBACK	Aborts current transaction

DCL control the database: Maintenance

SQL Basics

ANSI/ISO SQL Keywords

- ANSI/ISO specifies standard SQL keywords which cannot be used to name databases objects like tables, columns and users

Note: Keywords used may varies with the different implementations of SQL

SQL Basics

Some example keywords:

ALL	COUNT	FOUND	MAX	PRIVILEGES
AND	CREATE	FROM	MIN	REFERENCE
AVG	DEFAULT	GO	NOT	ROLLBACK
BEGIN	DELETE	GRANT	NULL	SELECT
BETWEEN	DISTINCT	GROUP	NUMERIC	SET
BY	END	HAVING	OF	SQL
C	EXEC	IN	OR	TABLE

Data Manipulation using SQL

- Creating a Database:
  - CREATE DATABASE
    - Creating a database schema;
    - Database options: Connect, Disconnect, Select, Close, Create, Drop

SQL for Data Definition

SQL lets a user define the structure and organisation of the stored data and relationships among the stored data items.

- Commands:
  - **CREATE**
  - **DROP**
  - **ALTER**

- SQL also allows for:
  - Definition of indexes to make table access faster
  - Control of physical storage by DBMS

Command: CREATE

Function: *Define new objects (database, schema, location, table, index and views*)

CREATE DATABASE	Initial allocation of storage space to contain database objects (not in SQL-92)
CREATE SCHEMA	define a portion of a database that a particular user owns
CREATE LOCATION	defines the location of the database (distributed systems)
CREATE TABLE	defines a new table
CREATE INDEX	defines an index that enables rapid access
CREATE VIEW	defines logical table from one or more tables or views.

Data Manipulation using SQL

- Defining tables and views:
  - CREATE TABLE,
  - ALTER TABLE,
  - DROP TABLE

Examples of ANSI/ISO SQL Data Types

Note: Data types may varies in different implementations of SQL

Data Type	Description
CHAR(*length*)	Fixed length character strings
CHARACTER	
INT	Integer numbers
INTEGER	
SMALLINT	Small integer numbers
NUMERIC(*precision,scale*)	Integer or Decimal numbers
NUMBER(*precision,scale*)	
DECIMAL(*precision, scale*)	
DEC(*precision,scale*)	
FLOAT(*precision*)	Floating points numbers
REAL	Low-precision floating point no.
DOUBLE PRECISION	High-precision floating point no.

Command: CREATE TABLE

Function: *Defines a new table and its columns*

CREATE TABLE *table-name*
(*column-definition*)
(*primary-key-definition*)
(*foreign-key-definition*)
(*uniqueness-constraint*)
(check-constraint)

column definition: *column-name* *data-type*
{NOT NULL} *{WITH DEFAULT}*

primary-key: *PRIMARY KEY (column-name)*

foreign-key: *FOREIGN KEY {rel-name} (column-name)*
REFERENCES table-name
{ON DELETE [RESTRICT, CASCADE, SET NULL]}

uniqueness: *UNIQUE (column-name)*

check: *CHECK (expression)*

Examples of ANSI/ISO SQL Data Types

- *Examples of Extended Data Types*
  - Variable-length character strings (VARCHAR)
  - Money Amount (MONEY / CURRENCY)
    - Dates and Times (DATE / TIMESTAMP)
    - Boolean Data (LOGICAL)
    - Long Text (LONG / TEXT)
    - Unstructured Data (RAW)
    - Asian Characters

Data type differences across SQL implementations is one barrier to portability

Data Manipulation using SQL

- Specifying integrity constraints:
  - PRIMARY KEY,
  - UNIQUE,
  - NOT NULL,
  - CHECK,
  - Referential Integrity constraints (Cyclic, Self referencing, Multiple path)
  - FOREIGN KEY (CASCADE, RESTRICT, NULIFIES),
  - DEFAULT.

SQL for Data Integrity

- **Value of Stored Data can be lost in many ways:**
  - Invalid data added to data base
  - Existing data modified to a incorrect value
  - Changes made lost due to system error or power failure
  - Changes partially applied

- **Types of integrity constraints:**
  - Required Data (NOT NULL)
  - Validity Checking (CHECK)
  - Entity Integrity (PRIMARY KEY & NOT NULL)
  - Referential Integrity (FOREIGN KEY)
  - Business Rules (ASSERTION, TRIGGER)
  - Consistency (CASCADE, RESTRICT, SET NULL)

SQL for Data Integrity

- NULL Values: What are Null values?
  - Null values provides a systematic way of handling missing or inapplicable data in SQL.
  - It is inevitable that in real-world, some data are missing, not yet known or do not apply.
  - Null value is not a real data value.

- Special Handling
  - Null values require special handling by SQL and the DBMS. Null values can be handled inconsistently by various SQL products
  - Example: How do we handle null values in summaries like SUM,AVERAGE, etc.?

Referential Integrity

- Referential integrity constraints define the rules for associating rows with each other,
  - i.e. columns which reference columns in other tables:

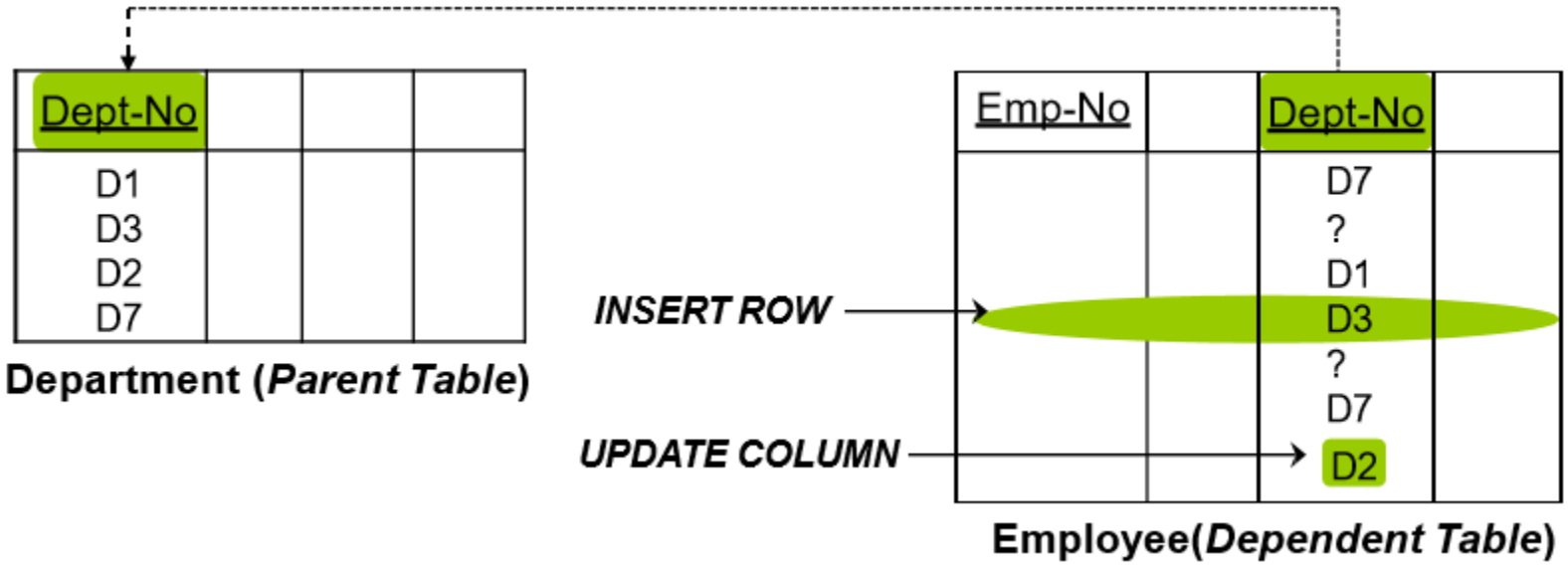
Every non-null value in a foreign key must have a corresponding value in the primary key which it references.

Referential Integrity

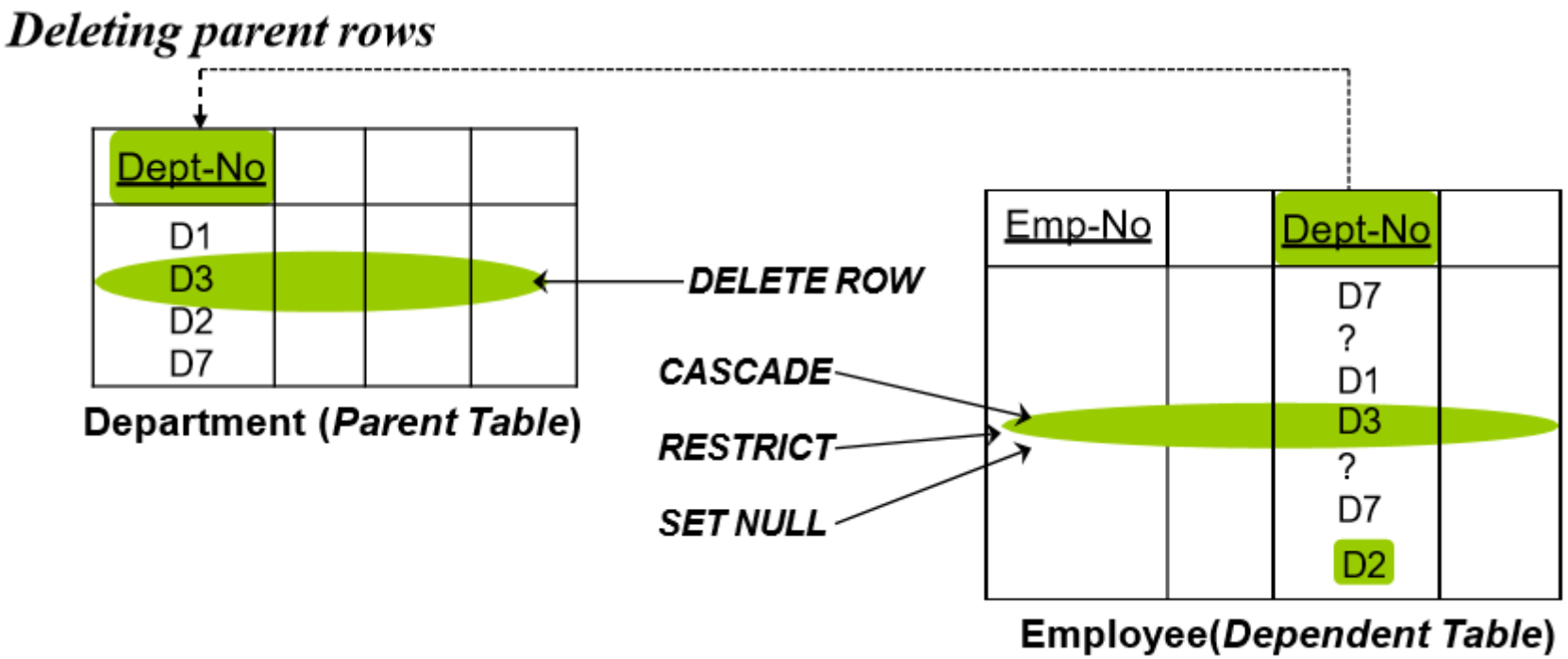
A row can be inserted or a column updated in the dependent table only if

(1) there is a corresponding primary key value in the parent table, or

(2) the foreign key value is set null.



Referential Integrity



Referential Integrity

Database designers must explicitly declare the effect if a delete from the parent table on the dependent table:

CASCADE deletes associated dependent rows

RESTRICT will not allow delete from the parent table if there are associated dependent rows.

SET NULL sets the value of associated dependent columns to null values.

SET DEFAULT

Referential Integrity

- Employee(Emp_No, NID, Address, Salary, Gender, DOB, First_Name, Mid_Initials, Last_Name, *Dept_No*, *Supervisor*)

```
CREATE TABLE Employee
  (Emp_No      CHAR(5)        NOT NULL,
   NID         CHAR(10),
   Address     VARCHAR(50),
   Salary      DEC(7,2)       CHECK (Salary >= 0),
   Gender      CHAR(1)        CHECK (Gender IN ('M', 'F')),
   DOB         DATE,
```

Referential Integrity

- Employee(Emp_No, NID, Address, Salary, Gender, DOB, First_Name, Mid_Initials, Last_Name, *Dept_No*, *Supervisor*)

```
First_Name   CHAR(10),
Mid_Initials   CHAR(10),
Last_Name   CHAR(15)      NOT NULL,
Dept_No      CHAR(2)      NOT NULL,
Supervisor   CHAR(5),
PRIMARY KEY (Emp_No),
FOREIGN KEY (Dept_No) REFERENCES Department,
FOREIGN KEY (Supervisor) REFERENCES Employee);
```

Referential Integrity

Dependent(Emp_No, Depd_Name,Gender, DOB, Relation)

```
CREATE TABLE Dependent
    (Emp_No         CHAR(4)      NOT NULL,
     Depd_Name      CHAR(15)     NOT NULL,
     Gender         CHAR(1)      CHECK (Gender IN ('M', 'F')),
     DOB            DATE         NOT NULL,
     Relation       VARCHAR(15),
    PRIMARY KEY (Emp_No, Depd_Name),
    FOREIGN KEY (Emp_No) REFERENCES Employee
                                 ON DELETE CASCADE);
```

```
CONSTRAINT Dependent_PK
        PRIMARY KEY (Emp_No, Depd_Name)
CONSTRAINT Dependent_FK FOREIGN KEY (Emp_No)
        REFERENCES Employee (Emp_No)
```

Referential Integrity

Single-field constraint:

```
CONSTRAINT name
        {PRIMARY KEY | UNIQUE | NOT NULL |
  REFERENCES foreign-table [(foreignfield1, foreignfield2)]}
```

Multiple-field constraint:

```
CONSTRAINT name
        {PRIMARY KEY (primary1[, primary2 [, ...]]) |
        UNIQUE (unique1[, unique2 [, ...]]) |
        NOT NULL (notnull1[, notnull2 [, ...]]) |
        FOREIGN KEY (ref1[, ref2 [, ...]]) REFERENCES
          foreign-table [(foreignfield1 [, foreignfield2 [, ...]])]}
```

Data Manipulation using SQL

- Creating indexes:
  - CREATE INDEX,
  - DROP INDEX

Command: CREATE

Create Index Command

CREATE [UNIQUE] INDEX *index-name*
ON *table-name* (field [ASC|DESC][, field [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL |
IGNORE NULL }]

Example

- CREATE UNIQUE INDEX Dept_Name_IDX ON Department (Dept_Name)
- CREATE INDEX Name_IDX ON Employee (Last_Name)
- CREATE INDEX Emp_Name_IDX ON Employee (First_Name, Mid_Initials, Last_Name)

Command: DROP

Function

- *Remove (erase) an existing object that is no longer needed*

DROP Command

DROP [*table-name* | *index-name* | *view-name*]

Example

- DROP TABLE Dependent
- DROP INDEX Name_IDX
- DROP VIEW Emp_VIEW

Note: *Most RDBMSs will ensure that users dropping different kinds of objects must possess the authority (privileges)*

Command: ALTER

Function

– *Change the definition of an existing table.*

ALTER TABLE *table-name* { *option(s)* }

{ADD *column-name data-type* {NOT NULL} {WITH DEFAULT},

|DELETE *column-name* [, ...]

|RENAME *old-column-name new-column-name* [, ...]

|MODIFY *column-name column-type*

|UNIQUE KEY *key-name (column-list)*

- |PRIMARY KEY *key-name (column-list)*
- |FOREIGN KEY [*constraint-name*] *(column-list)* REFERENCES *table-name*
- [ON DELETE {RESTRICT | CASCADE | SET NULL}]
- |DROP PRIMARY KEY
- |DROP FOREIGN KEY *constraint-name*]
- |DROP KEY *key-name*]
- |DROP CHECK }

;

Command: ALTER

Example

- *Adding a Column*

```
ALTER TABLE Project
        ADD Proj_Manager CHAR(5)
```

- *Changing Primary or Foreign Key*

```
ALTER TABLE Department
        DROP PRIMARY KEY
        PRIMARY KEY (Dept_Name)
        FOREIGN KEY (Manager) REFERENCES Employee
```