Chapter 7

SQL: Data Definition

Pearson Education © 2009

Main SQL DDL Directives

- Create and alter tables, including specification of fields and their data types
- Specify integrity constraints on data
- Define and maintain views

ISO SQL Data Types

Table 6.1 ISO SQL data types.

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

Boolean

- Normal true/false values
- There is a 3rd truth value called UNKNOWN which is represented as NULL

Character Data

- Examples
  - branchNo CHAR(4): a fixed width string 4 characters long
  - address VARCHAR(30): a variable length string of up to 30 characters
- If a string is fixed length, then a short string will be padded to the right with blankspaces
- CHAR and VARCHAR can store up to 255 character strings
- Use TEXT to store longer length strings (up to 65,535 characters)
- See mysql documentation for representing even longer strings

Numeric Data

- **Precise real numbers: NUMERIC or DECIMAL (DEC)- they are identical**
  - **Example: salary DECIMAL(7,2): stored as a string but can be manipulated like a number and aggregation functions like avg and sum work with it**
    - Precision: The first number represents the total number of digits
    - Scale: The second number represents the total number of decimal digits
    - salary can store any number up to 99,999.99
- **Integers: INTEGER (INT) or SMALLINT: Use SMALLINT to conserve space when your field values are small**

Numeric Data (cont.)

- **Approximate numeric data: Like C/C++, real numbers in these formats can only be approximated**
  - **Different types**
    - FLOAT(precision, decimaldigits): a small decimal number
      - precision controls the total number of digits
      - decimaldigits controls number of digits to right of decimal point
    - DOUBLE(precision, decimaldigits): a large decimal number

Dates

- **Date: allows you to store a date**
  - **Three types of dates**
    - DATE: Stores calendar dates using year, month, and day
      - format YYYY-MM-DD
    - TIME [timePrecision]: Stores time as hours, minutes, and seconds
      - timePrecision specifies number of decimal digits for the seconds field
      - format: HH:MM:SS
    - TIMESTAMP[timePrecision]: Stores date and times
      - format: YYYY-MM-DD HH:MM:SS
  - **Handy date functions in mysql**
    - NOW(): Returns the current date and time
    - CURDATE(): Returns the current date
    - CURTIME(): Returns the current time
    - DATE(): Extracts the date part of a date or date/time expression
    - DATE_ADD(): Adds a specified time interval to a date
    - DATE_SUB(): Subtracts a specified time interval from a date
    - DATEDIFF(): Returns the number of days between two dates

Intervals

- **Interval: allows you to specify a time interval, either as year-month intervals or day-time intervals (day-time allows days, hours, minutes, and seconds)**
  - **Examples**
    - INTERVAL YEAR(2) TO MONTH represents an interval of time from 0 years 0 months to 99 years 11 months
    - INTERVAL HOUR(2) TO SECOND(4) represents an interval between 0 hours 0 minutes 0 seconds to 99 hours 59 minutes 59.9999 seconds
    - CURRENT_DATE BETWEEN dateFROM AND DATE_SUB(dateTo, INTERVAL 1 DAY)
  - **If you want to further restrict the interval (e.g., to 5 years rather than 99 years) than you must use the CHECK command presented in a later slide**

Integrity Constraints

- Consider five types of integrity constraints:
  - required data: Whether a data field must contain a value
  - domain constraints: A set of legal values for a field
  - entity integrity: Each primary key of a table must contain a unique, non-null value
  - referential integrity: Foreign keys must refer to a valid, existing row in the parent relation
  - general constraints: General, organization specific constraints, such as requiring that no staff member handle 100 properties
- These constraints are defined in the CREATE TABLE and ALTER TABLE commands

Integrity Constraints

<u>Required Data:</u> **SQL's "NOT NULL" command requires that a field have a defined value**

```
position  VARCHAR(10)   NOT NULL
```

<u>Domain Constraints</u>

(a)<u>CHECK:</u> CHECK clause allows you to make an assertion about the types of values that may appear in a column

- **(a) general form: CHECK(SEARCH CONDITION)**
- **(b) Example:**

```
semester    CHAR    NOT NULL
    CHECK (semester IN ('Fa', 'Sp', 'Su'))
```

Domain Constraints

(b) CREATE DOMAIN: allows you to define a restricted domain of values

```
CREATE DOMAIN DomainName [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)]
```

For example:

```
CREATE DOMAIN SemesterType AS CHAR(2)
      CHECK (VALUE IN (‘Fa’, ‘Sp’, ‘Su’));
semester    SemesterType   NOT NULL
```

Domain Constraints

- *searchCondition* **can involve a table lookup:**

```
CREATE DOMAIN BranchNo AS CHAR(4)
CHECK (VALUE IN (SELECT branchNo
                 FROM Branch));
```

Entity Integrity

- Primary key of a table must contain a unique, non-null value for each row.
- AUTO_INCREMENT can be used to automatically assign unique integers to a field

```
StudentId int NOT NULL AUTO_INCREMENT
```

- PRIMARY KEY declares primary keys

```
PRIMARY KEY(staffNo)
PRIMARY KEY(clientNo, propertyNo)
```

- Can only have one PRIMARY KEY clause per table. Can still ensure uniqueness for alternate keys using UNIQUE:

```
UNIQUE(telNo)
```

- Any field declared UNIQUE must also be declared to be NOT NULL

Referential Integrity

- A foreign key is a column or set of columns that links each row in child table to a row of parent table containing a matching primary key.
- Referential integrity means that, if a foreign key contains a value, that value must refer to an existing row in the parent table.
- ISO standard supports definition of foreign keys with FOREIGN KEY clause in CREATE and ALTER TABLE:

```
FOREIGN KEY(branchNo) REFERENCES Branch(branchNo)
```

- The attributes referenced in the “parent” relation must be declared as a primary key or as UNIQUE

Referential Integrity

- **Any INSERT/UPDATE attempting to create a foreign key value in the child table without matching primary key value in the parent is rejected.**
- **Action taken attempting to update/delete a primary key value in the parent table with matching rows in child is dependent on referential action specified using ON UPDATE and ON DELETE subclauses:**
  1. **CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.**
  2. **SET NULL: Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are allowed to be NULL.**
  3. **SET DEFAULT: Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.**
  4. **NO ACTION: Reject delete from parent. Default.**

Referential Integrity

Examples:

```
FOREIGN KEY (staffNo) REFERENCES staff (staffNo)
      ON DELETE SET NULL
FOREIGN KEY (ownerNo) REFERENCES privateOwner (ownerNo)
      ON UPDATE CASCADE
```

General Constraints

- **Use CHECK command outside a field definition**
  - **Example**

```
CONSTRAINT StaffNotHandlingTooMuch
CHECK (NOT EXISTS   (SELECT staffNo
                     FROM PropertyForRent
                     GROUP BY staffNo
                     HAVING COUNT(*) > 100))
```

The CONSTRAINT keyword names the constraint so that it can be dropped by an ALTER TABLE statement

Limitations on Integrity Checking in MYSQL

- **CREATE DOMAIN is not supported in MYSQL**
- **CHECK is partially enforced by MYSQL**
  - Does not support non-deterministic functions, including aggregate functions like min, max, avg
  - Does not support environmental variables such as CURRENT_DATE
  - Does not support sub-queries
- **The ENUM type is enforced**

  Example: semester ENUM('Fa', 'Sp', 'Su'));
- **Foreign key constraints are enforced in MYSQL**

Data Definition

- **SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed.**
- **Main SQL DDL statements that we will consider are:**

CREATE/ALTER TABLE	**DROP TABLE**
CREATE VIEW	**DROP VIEW**

- **Many DBMSs also provide:**

CREATE INDEX	**DROP INDEX**

CREATE TABLE-Informal Definition

```
CREATE TABLE TableName
    column declarations
    PRIMARY KEY (listOfColumns)
    UNIQUE (listOfColumns1), … (listOfColumnsn)
    FOREIGN KEY declarations
    general constraints
```

CREATE TABLE

```
CREATE TABLE TableName
  {(colName dataType [NOT NULL] [UNIQUE]
        [DEFAULT defaultOption]
        [CHECK searchCondition] [,...]}
   [PRIMARY KEY (listOfColumns),]
   {[UNIQUE (listOfColumns),] [...,]}
   {[FOREIGN KEY (listOfFKColumns)
        REFERENCES ParentTableName [(listOfCKColumns)],
              [ON UPDATE referentialAction]
              [ON DELETE referentialAction ]] [,...]}
  {[CHECK (searchCondition)] [,...] })
```

CREATE TABLE

- **Creates a table with one or more columns of the specified *dataType*.**
- **With NOT NULL, system rejects any attempt to insert a null in the column.**
- **Can specify a DEFAULT value for the column.**
- **Primary keys should always be specified as NOT NULL.**
- **FOREIGN KEY (FK) clause specifies**
  - **the foreign key**
  - **the parent relation**
  - **actions to perform when the associated primary key is updated/deleted from the parent relation.**

Example: Create PropertyForRent Table

PropertyForRent(<u>propertyNo</u>, street, city, postcode, propertyType, rooms, rent, ownerNo, staffNo, branchNo)

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT ownerNo FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
  CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN PNumber AS VARCHAR(5);
CREATE DOMAIN PRooms AS SMALLINT;
  CHECK(VALUE BETWEEN 1 AND 15);
…
```

Example (cont): Create PropertyForRent Table

```
CREATE TABLE PropertyForRent (
    propertyNo PNumber          NOT NULL,
    ....
    rooms      PRooms           NOT NULL  DEFAULT 4,
    rent       PRent            NOT NULL  DEFAULT 600,
    ownerNo    OwnerNumber      NOT NULL,
    staffNo    StaffNumber
               Constraint StaffNotHandlingTooMuch ....
    branchNo   BranchNumber     NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff (staffNo)
     ON DELETE SET 'S001'
     ON UPDATE CASCADE ....);
```

ALTER TABLE

- Add a new column to a table.
- Drop a column from a table.
- Add a new table constraint.
- Drop a table constraint.
- Set a default for a column.
- Drop a default for a column.

Example: ALTER TABLE

Change Staff table by removing default of 'Assistant' for position column and setting default for sex column to female ('F').

```
ALTER TABLE Staff
        ALTER position DROP DEFAULT;
ALTER TABLE Staff
        ALTER branchNo SET DEFAULT 'B003';
```

Example: ALTER TABLE

Remove constraint from PropertyForRent that staff are not allowed to handle more than 100 properties at a time. Add new column to Client table.

```
ALTER TABLE PropertyForRent
    DROP CONSTRAINT StaffNotHandlingTooMuch;
ALTER TABLE Client
    ADD prefNoRooms PRooms;
```

DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE]

e.g. DROP TABLE PropertyForRent;

- **Removes named table and all rows within it.**
- **With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.**
- **With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).**

Views

- **Definition: Virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.**
- **Example: A view that shows how many properties are managed by each staff member:**

branchNo	staffNo	count
B003	SG14	7
B003	SG37	23
B005	SL41	10

Views

- **Contents of a view are defined as a query on one or more base relations.**
- **Alternative system approaches to supporting views**
  - **view resolution: any operations on the view are automatically translated into operations on the relations from which it is derived.**
  - **view materialization: the view is stored as a temporary table, which is maintained as the underlying base tables are updated.**

Horizontal View

- A horizontal view restricts a user's access to selected rows of one or more tables
- Example: Create view so that manager at branch B003 can only see details for staff who work in his or her office.

```
CREATE VIEW Manager3Staff
        AS   SELECT *
             FROM Staff
             WHERE branchNo = 'B003';
```

Table 6.3 Data for view Manager3Staff.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

Vertical View

- A **vertical view** restricts a user's access to selected columns of one or more tables
- Example: Create view of staff details at branch B003 excluding salaries.

```
CREATE VIEW Staff3
AS  SELECT staffNo, fName, lName, position, sex FROM Staff
    WHERE branchNo = 'B003';
```

Table 6.4 Data for view Staff3.

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

Pearson Education © 2009

Views that Use Aggregate Operations and Draw Data From Multiple Relations

Create a view of staff who manage properties for rent, including branch number they work at, staff number, and number of properties they manage.

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
    AS SELECT s.branchNo, s.staffNo, COUNT(*)
    FROM Staff s, PropertyForRent p
    WHERE s.staffNo = p.staffNo
    GROUP BY s.branchNo, s.staffNo;
```

Table 6.5 Data for view StaffPropCnt.

branchNo	staffNo	cnt
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

SQL – View Creation Syntax

CREATE VIEW ViewName [(newColumnName [,...])]
AS subselect [WITH CHECK OPTION]

- **Can assign a name to each column in view.**
- **If list of column names is specified, it must have same number of items as number of columns produced by *subselect*.**
- **If omitted, each column takes name of corresponding column in *subselect*.**

SQL - CREATE VIEW

- **List must be specified if there is any ambiguity in a column name (e.g., an aggregated value produced by the SELECT does not have a name, so it must be given one)**
- **The *subselect* is known as the <u>defining query</u>.**
- **WITH CHECK OPTION prevents a row from being inserted into the view if it violates the WHERE condition of the defining query**

SQL Restrictions on Queries on a View

If a column in a view is based on an aggregate function:

- **Column may appear only in SELECT and ORDER BY clauses of queries that access view.**
- **Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.**

For example, the following queries would fail:

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

```
SELECT *
      FROM StaffPropCnt
      WHERE cnt > 2;
```

View Updatability

- View updatability and restrictions on it were discussed in The Relational Model lecture

Advantages/Disadvantages of Views

- Advantages and disadvantages of views were discussed in The Relational Model lecture

View Materialization

- **View resolution mechanism may be slow, particularly if view is accessed frequently.**
- **View materialization stores view as temporary table when view is first queried.**
- **Thereafter, queries based on materialized view can be faster than recomputing view each time.**

View Maintenance

- **Difficulty is maintaining the currency of the view while base tables(s) are being updated.**
- **<u>View maintenance</u> aims to apply only those changes necessary to keep view current.**
- **Consider following view:**

```
CREATE VIEW StaffPropRent(staffNo)
AS SELECT DISTINCT staffNo
    FROM PropertyForRent
    WHERE branchNo = 'B003' AND
        rent > 400;
```

Table 6.8 Data for view StaffPropRent.

staffNo
SG37
SG14

View Materialization

- If insert row into PropertyForRent with rent $\leq$400 then view would be unchanged.
- If insert row for property PG24 at branch B003 with staffNo = SG19 and rent = 550, then row would appear in materialized view.
- If insert row for property PG54 at branch B003 with staffNo = SG37 and rent = 450, then no new row would need to be added to materialized view.
- If delete property PG24, row should be deleted from materialized view.
- If delete property PG54, then row for PG37 should not be deleted (because of existing property PG21).

Things We Won't Discuss

- **Transactions: Sometimes you want to group a set of queries, especially insert/update/delete queries, so that they either all take effect, or none of them take effect.**
  - **A transaction allows you to group a set of queries based on COMMIT and ROLLBACK.**
  - **Example: I might want my create table queries to be one transaction, and each set of table insertions to be transactions**
- **Access Privileges: SQL allows the DBA to grant select/insert/update/delete privileges to users on a per relation basis**