SQL Fundamentals

Chapter 3

FIGURE 3.1.1: Command procedures have to be realistic.

SQL

- ***SQL*** (Structured Query Language): A language that is used in relational databases to build and query tables.
- Earlier versions named Structured English Query Language (SEQUEL)

Syntax Conventions

- Keyword identifies the operation
- Statement ended with an optional semi-colon
- Statements divided into clauses
  - Most clauses are optional
  - Clauses usually must be in a specific sequence
- White space separates elements (keywords, operators, constants, etc.)

Constants in SQL

- Numeric constants are simply numbers; may include:
  - A decimal point
  - A sign
  - E (scientific notation)
- Character constants are enclosed in single quotes

Object Names

- SQL Standard specifies up to 128 character names
  - Many SQL implementations have lower length limits
- Names should not be case-sensitive per the standard
  - However, there are exceptions
- Names are composed of letters, numeric digits, and the underscore (spaces within names are not supported)
  - SQL Standard allows for quoted identifiers (using double-quotes to enclose the entire name)
    - Supports names with keywords, spaces, special characters
    - Quoted identifiers must match exactly, including case

SQL Statement Categories

- Data Query Language (DQL)
  - Statements that query data but change nothing
  - SELECT statement
- Data Manipulation Language (DML)
  - Statements that modify data values but not data structures (tables, views, etc.)
  - INSERT, UPDATE, DELETE statements

SQL Statement Categories

- Data Definition Language (DDL)
  - Statements that add/change/delete database objects, but not the data within them (the *containers* but not the *contents*)
  - CREATE, ALTER, DROP statements
- Data Control Language (DCL)
  - Statements that manage privileges users have regarding database objects
  - GRANT, REVOKE statements

Data Types

- SQL Standard specifies data types, but vendor extensions are common
- Data Type Classes:
  - String: for character strings and bit strings
  - Numeric: for numbers
  - Datetime: for dates, times, timestamps
  - Boolean: logical true/false (sparsely supported)

String Data Types

- CHARACTER (CHAR)
- CHARACTER VARYING (VARCHAR)
- CHARACTER LARGE OBJECT (CLOB)
- XML
- NATIONAL CHARACTER (NCHAR)
- NATIONAL CHARACTER VARYING (NVARCHAR)
- BINARY
- BINARY VARYING
- BINARY LARGE OBJECT (BLOB)

Numeric Data Types

- NUMERIC
- DECIMAL
- INTEGER
- SMALLINT
- BIGINT
- FLOAT
- REAL
- DOUBLE PRECISION

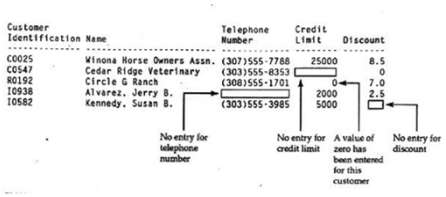
DateTime Data Types

- DATE
  - Some implementations store time down to seconds in DATE data types
- TIME
- TIME WITH TIME ZONE
- TIMESTAMP
  - Use DATETIME in SQL Server – it uses TIMESTAMP for an entirely different purpose
- TIMESTAMP WITH TIME ZONE

NULL Values

- NULL represents an unknown value (instead of using false data to represent "unknown")
- NULL is not the same as a blank or a zero or an empty string
- NULLs are not equal to anything else, including other NULLs
- Test for NULL using "IS NULL" rather than "= NULL"
- Nulls cause problems because they introduce three-value logic where Boolean is two-valued

The Need for Null Values



DDL

Data Definition Language

DDL Statements

- CREATE – creates new objects (tables, indexes, views, etc.)
- ALTER – changes the definition of existing objects
- DROP – destroys objects
- TRUNCATE – efficiently empties a table (without the need for row-by-row deletion of the data)
  - Not described in the text because TRUNCATE was added to the standard after the text was written

Creating Tables

```
CREATE TABLE table_name
 (column_name data_type [(length)] [NOT NULL], …)
 IN tablespace_name
 PRIMARY KEY [key_name] (column list)
 FOREIGN KEY [constraint_name] (column list)
    REFERENCES table_name (column list)
   ON DELETE {RESTRICT | CASCADE | SET NULL}
```


CREATE TABLE Example

```
DROP TABLE INVOICE;
CREATE TABLE INVOICE
   ( ID           CHAR(5),
     ORDER_DATE   DATE,
     AMOUNT       DECIMAL(8,2),
     CUSTOMER_ID  CHAR(5),
     EMPLOYEE_ID  CHAR(11)
   );
```

Primary Key Constraint

- A Primary Key constraint forces uniqueness on a column or set of columns. The DBMS creates a Unique Index to enforce the constraint.
- Each table may have only one primary key (but that key may be one or several columns)
- Alternatively, you can create a Unique Index to do the same thing, but it won't have a constraint name.

Primary Key Example

```
ALTER TABLE INVOICE
 ADD CONSTRAINT  PK_INVOICE PRIMARY KEY (ID);

--VS.--

CREATE UNIQUE INDEX PK_INVOICE
 ON INVOICE (ID);
```


Altering Table Definitions

```
ALTER TABLE table_name
  ADD column_name…
  RENAME column_name TO new_name
  MODIFY column_name …
  DROP PRIMARY KEY
  DROP FOREIGN KEY constraint_name
  CHECK (check constraint condition)
```

Alter Table

```
ALTER TABLE CUSTOMER
  ADD CONSTRAINT PK_CUSTOMER PRIMARY KEY (ID);

ALTER TABLE INVOICE
  ADD CONSTRAINT FK_CUST_ID FOREIGN KEY (CUSTOMER_ID)
  REFERENCES CUSTOMER(ID);
```


Create Index

- Indexes can make queries run more efficiently but they are never necessary to achieve correct results
- Indexes are used to enforce uniqueness of single columns or sets of columns

- Example:

```
CREATE INDEX invoice_customer_ix
    ON invoice (customer_id);
```


DROP Statement

- DROP statement destroys an object and its contents.
  - If you merely want to remove the data from a table, use TRUNCATE or DELETE instead.

```
DROP INDEX RACE_RESULT_HORSE;

DROP TABLE RACE_RESULT
CASCADE CONSTRAINTS;
```

DQL

Data Query Language (The SELECT Statement)

Relational Operations

- ***Selection***: Defines a relation that contains only those tuples (rows) that satisfy the specified condition (***predicate***)
- ***Projection***: Defines a relation that extracts the values of specified attributes (and eliminates duplicates)
- ***Cartesian Product***: Defines a relation that is the concatenation of every tuple in one relation with every tuple in another

Relational Operations

- ***Union***: The concatenation of two relations (duplicates eliminated); relations must be union-compatible
  - ***Union-compatible***: relations with the same number of attributes and matching domains
- ***Set Difference***: Defines a relation that contains all tuples from one relation that are not contained in another; must be union-compatible

Relational Operations

- ***Join***: a combination of two relations to form a new relation; a derivative of a Cartesian product that uses a ***join predicate*** to specify how tuples are to be joined
  - There are multiple types of joins, but in a language like SQL, they are all written using the same syntax

Relational Joins

- ***Equi-join***: A join where the join predicate contains only the equality operator (=)
  - ***The most commonly used form***
- ***Theta Join***: A join where the join predicate contains any comparison operator: <, <=, >, >=, != (<>), BETWEEN

Relational Joins

- ***Outer Join***: A join that includes a row from one of the relations even when there is no matching row in the other; missing attributes appear as null
  - ***Left Outer Join***: Outer join that returns every row for the left-hand relation
  - ***Right Outer Join***: Outer join that returns every row from the right-hand relation

SQL Demo

(see separate presentation)

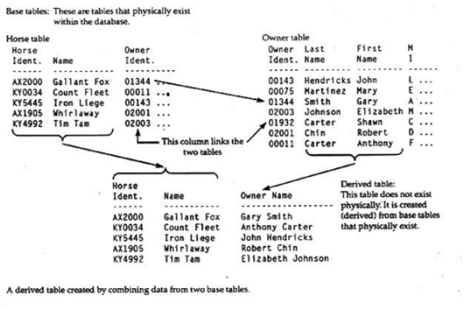
Views

Views

- ***View***: A dynamic result of one or more relational operations on the base relations to produce another relation; a ***virtual relation*** that does not exist in the database, but is produced upon request
- ***Base Relation***: A named relation corresponding to an entity in the conceptual schema whose tuples are physically stored in the database

Views

- Views can be constructed “on the fly” by performing various relational operations
- The definition of a view can be permanently stored in the catalog and used in relational operations in the same way as tables; stored views never contain data
- Changes made to the data in the base relations are automatically reflected in the view.



A derived table created by combining data from two base tables.

Purpose/Benefits of Views

- Provides a powerful and flexible security mechanism by hiding parts of the database from certain users
- Permits users to access data that is customized to their needs
- Can simplify complex operations on base relations (e.g. views can handle joins)

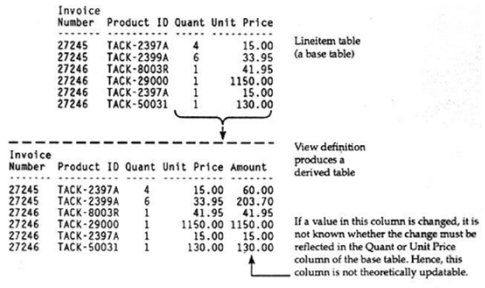
Updating Views

- The ability to apply updates to views varies widely from one DBMS to another
- Most DBMSs recognize views as “updateable” and “not updateable”; a few recognize “partially updateable” views
- Generally, views formed form a single relation that contain either a primary or candidate key can be updated

Updating Views

- Updates are generally not allowed through views involving multiple base relations
- Updates are generally not allowed through views involving aggregation or grouping operations
- Updates are generally not allowed on derived/calculated attributes in views

Updating Views



DML

Data Manipulation Language

INSERT Statement

- Adds new rows to a table
- Column list is optional, but highly recommended for flexibility and data independence
- Two forms:
  - INSERT with VALUES:
    - VALUES clause provides values for a row of data
    - Some implementation support multiple VALUES clauses, each of which causes one row to be added
  - INSERT with Subquery
    - SELECT statement provides the rows and column values to be inserted.

INSERT Statement Examples

```
INSERT INTO RACE_RESULT
(TRACK_ID, RACE_NUMBER, RACE_DATE, HORSE_ID, PLACE
 PURSE, LENGTHS_BEHIND)
 VALUES ('Winona','1','2015-07-04','KY0993',4,0,10);

INSERT INTO RACE_RESULT
(TRACK_ID, RACE_NUMBER, RACE_DATE, HORSE_ID, PLACE
 PURSE, LENGTHS_BEHIND)
 VALUES ('Winona','1','2015-07-04','QH0334',1,2500,0),
        ('Winona','1','2015-07-04','QH3993',5,0,11),
        ('Winona','1','2015-07-04','QH0443',3,500,1.5);
```

INSERT Statement Examples

```
INSERT INTO RACE_RESULT
(TRACK_ID, RACE_NUMBER, RACE_DATE, HORSE_ID, PLACE
 PURSE, LENGTHS_BEHIND)
   SELECT TRACK_ID, RACE_NUMBER, RACE_DATE,
          HORSE_ID, PLACE, PURSE, LENGTHS_BEHIND
     FROM RACE_RESULT
    WHERE RACE_DATE BETWEEN '2015-01-01'
                        AND '2015-12-31';
```


Update Statement

- Modifies data values within a table
- SET clause specifies the list of columns to be updated, along with an expression that provides a new value for each column
- If WHERE clause is omitted, will attempt to update very row in the table

```
UPDATE OWNER
   SET OWNER_LAST_NAME = 'McMillan‘,
       STREET_ADDRESS = '3205 Equine Ct., Apt. 6'
  WHERE OWNER_ID = '00011';
```


DELETE Statement

- Removes rows from a table
- If WHERE clause is omitted, attempts to delete all the rows in the table

```
DELETE FROM RACE_RESULT
 WHERE RACE_DATE < '2012-01-01';
```

Data Control Language

DCL

Privileges

- DCL statements manage privileges for database accounts.
- Privileges can be:
  - System privileges, which permit the grantee to perform a general DBMS function, such as creating views or connecting to the database.
  - Object privileges, which permit the grantee to perform specific actions on specific objects, such as selecting from the OWNER table or inserting into the RACE_RESULT table

GRANT Statement

- Gives one or more privileges to grantee(s).
- Examples:

```
GRANT CREATE  VIEW TO HR;

GRANT SELECT, INSERT, UPDATE
    ON RACE_RESULT TO Mgr123;
```

REVOKE Statement

- Takes privileges away from grantees
- Examples:

```
REVOKE CREATE VIEW FROM HR;

REVOKE SELECT, INSERT, UPDATE
    ON RACE_RESULT FROM Mgr123;
```