IS475/675 Agenda for 03/12/2025

Discuss

- SQL programming language.
- Database constraints.

Review

- CREATE TABLE statements.

Do

- SQL Lab #1 or if you have finished, then do HW#5

What is SQL?

- Structured Query Language (SQL) is a non-procedural language designed to process and manage data stored in a relational database.
- Data processing operations through SQL include:
  - Creating data objects and enforcing constraints
  - Adding, changing and deleting data from tables
  - Accessing data from tables
- SQL is **not** designed to create programs that interact directly with users.
- SQL interfaces are also available for databases that are based on other models (NO-SQL databases – means “not only SQL).

SQL is standardized across vendors

- SQL is an ANSI-standard language (American National Standards Institute) which means that it has consistent syntax across different database platforms.
- Changes to the language are done through a committee.
- Standardization allows people to use the same database language with different DBMS's.
- The database management system we are using is sold from Microsoft and is called SQL Server.
- Each vendor provides extensions/enhancements to the SQL language. Microsoft's is called Transact-SQL or T-SQL.

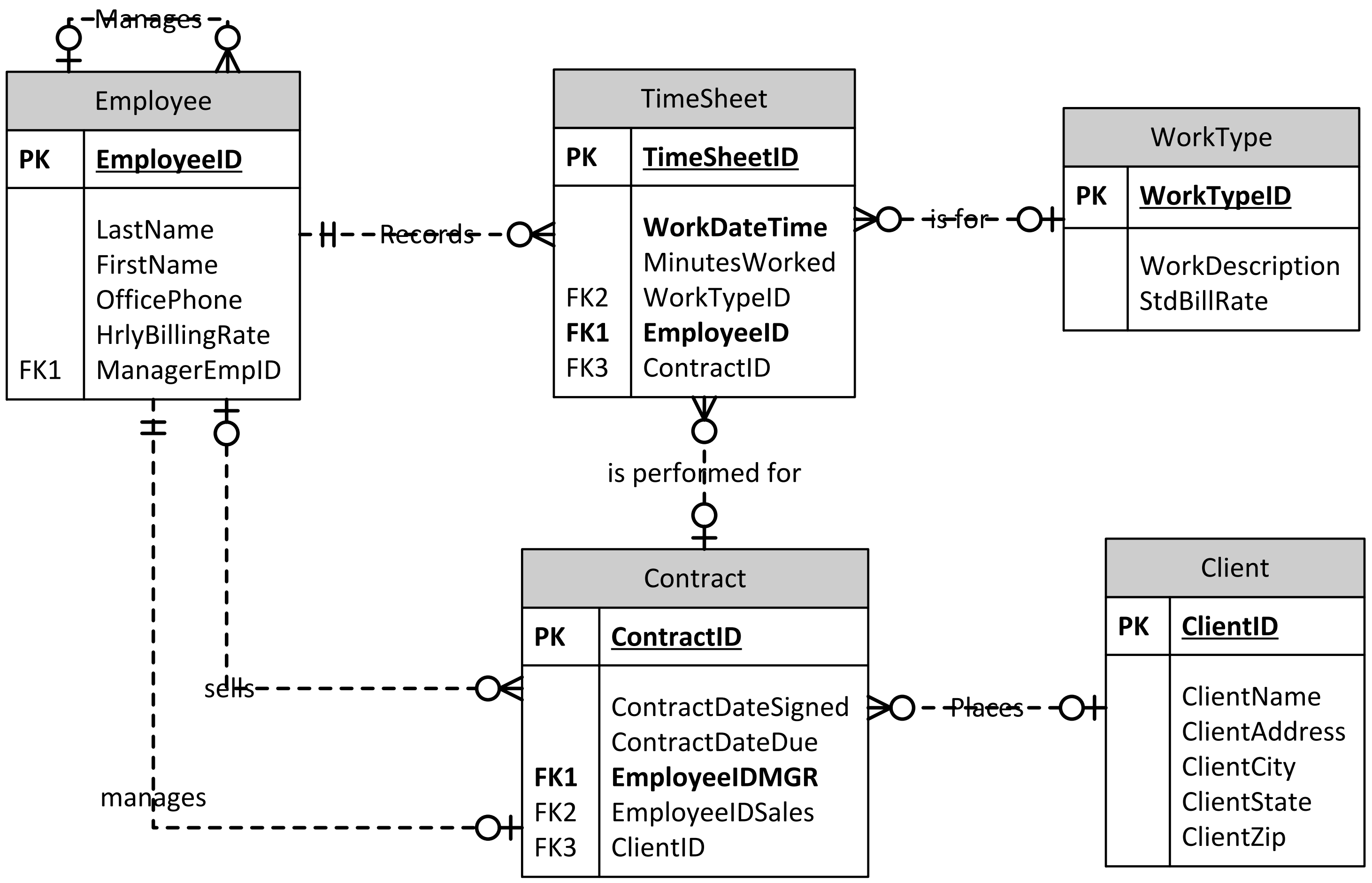
SQL Keywords/Commands

Data Definition Language Commands (DDL)	Data Manipulation Language Commands (DML)	Data Control Language Commands (DCL)
CREATE	INSERT	GRANT
ALTER	UPDATE	REVOKE
DROP	DELETE	COMMIT
	TRUNCATE	ROLLBACK
	SELECT	SET TRANSACTION

Guidelines for writing SQL

- SQL statements start with a command, and then include few or many modifiers/extensions for the command.
- SQL statements are not case sensitive.
- The data stored in a database IS case sensitive.
- SQL statements can span more than one physical line; it is a free form language.
- SQL keywords cannot be abbreviated or split across lines.

Database for SQL Lab Exercise 1



SQL code to create a table

The name of the table is zEmployee.

```
CREATE TABLE    zEmployee
(EmployeeID     INT  PRIMARY KEY,
 LastName       VARCHAR(30),
 FirstName      VARCHAR(25),
 OfficePhone    CHAR(10),
 HrlyBillRate   MONEY,
 ManagerEmpID   INT);
```

If the primary key is only one field (not concatenated) then can say "primary key" along with the field

Each field has one and only one data type

Field names (attributes) are traditionally declared one per line

SQL Lab Exercise 1 - continued

- We are starting from page 15, Task 9 – Create Tables with new constraints.

What is a database constraint?

- A rule/restriction assigned to a database object and stored in the data dictionary.
- Database constraints are enforced by the database management system.
- Database constraints cannot be circumvented by application programmer or by database users.
- We use database constraints to protect the integrity of the data.
- Constraints are used to centralize data integrity restrictions at the database level, rather than relying on individual programmers to program all data integrity validation.

Database constraints for HW#5

These are the constraints you need to know to be able to create the tables for HW#5

Primary key: Must be a unique data value, must not be null.

Data type: Numeric vs. text vs. date.

CHECK: Can check for any value or range.

NOT NULL: Must not be null.

REFERENCES: Enforces referential integrity.

Constraints can be declared on an individual field, or at the table-level.

Sample SQL Code to add constraints

```
CREATE TABLE                    zWorkType
(WorkTypeID                     INT  PRIMARY KEY,
 WorkDescription                VARCHAR(60) NOT NULL,
 StdBillRate                    MONEY  CHECK (StdBillRate > 10))
```

Creates a primary key, which forces a unique value for the WorkTypeID. Also ensures that the WorkTypeID is never NULL

Makes sure that a person does not enter alphabetic characters for the standard billing rate.

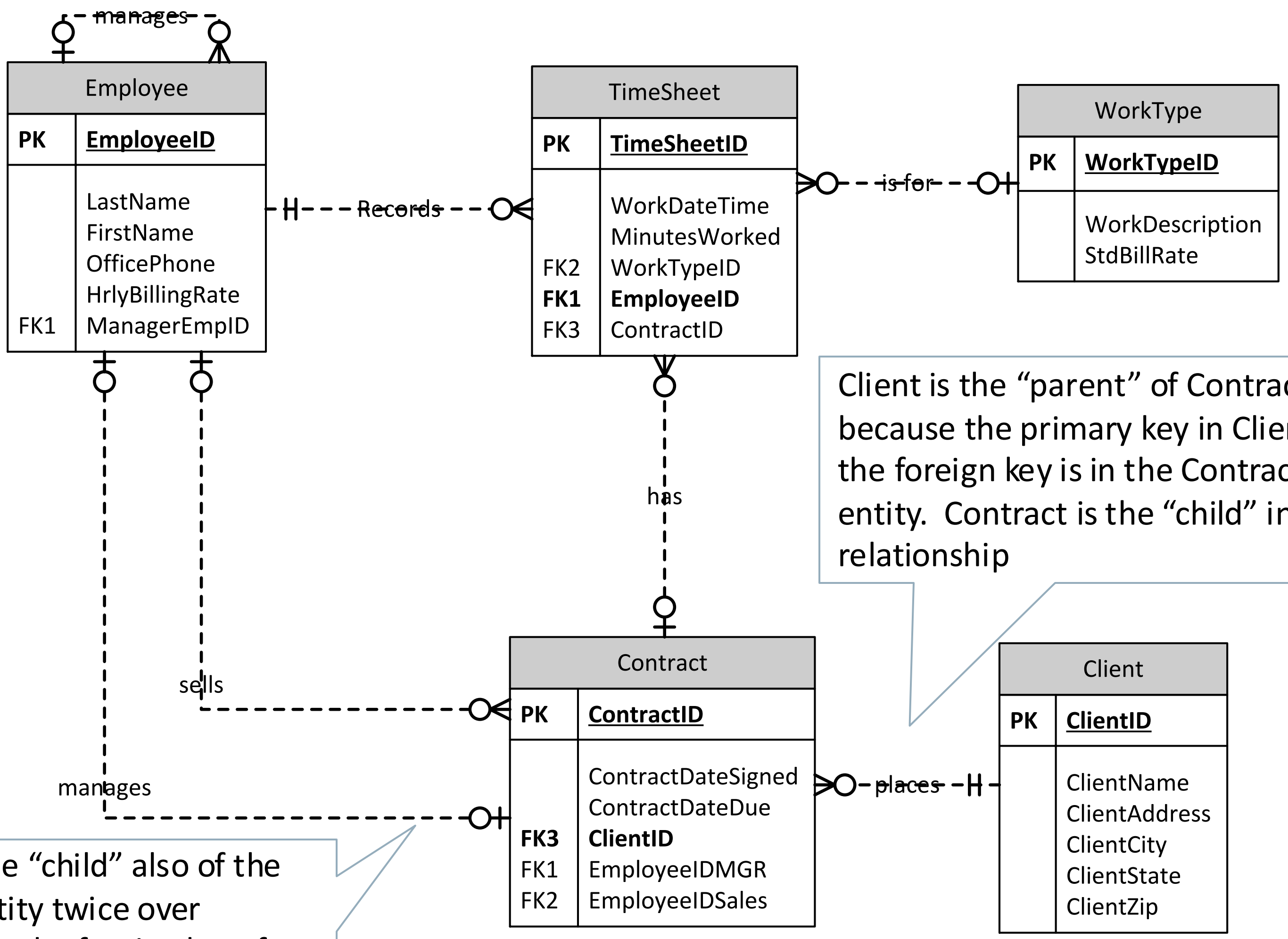
Ensures that the standard billing rate for a work type is always greater than 10

Forces data entry of a value for the WorkDescription – it can't be a NULL value

Referential integrity constraint

- A rule/restriction that states that either each foreign key value must match a primary key value in another entity, or the foreign key must be a null value.
- Another way to look at it – a row must exist in a "parent" entity before it can be placed in a "child" entity."

Parent/Child terminology



Client is the "parent" of Contract because the primary key in Client is the foreign key is in the Contract entity. Contract is the "child" in this relationship

Contract is the "child" also of the Employee entity twice over because it has the foreign keys for the two relationships with Employee.

```
CREATE TABLE zContract
(ContractID                    INT   PRIMARY KEY,
 ContractDateSigned            DATE,
 ContractDateDue               DATE,
 ClientID                      INT,
 EmployeeIDMGR                 INT,
 EmployeeIDSales               INT)
```

Foreign keys, but no referential integrity. This is an acceptable way of creating foreign keys that have **no constraints**.

```
CREATE TABLE zContract
(ContractID                    INT   PRIMARY KEY,
 ContractDateSigned            DATE,
 ContractDateDue               DATE,
 ClientID                      INT   FOREIGN KEY REFERENCES zClient (ClientID),
 EmployeeIDMGR                 INT   FOREIGN KEY REFERENCES zEmployee (EmployeeID),
 EmployeeIDSales               INT   FOREIGN KEY REFERENCES zEmployee (EmployeeID));
```

Create Foreign keys and enforce referential integrity by creating referential integrity constraints.

Alternate formats for declaration of primary key and referential integrity constraints

```
CREATE TABLE zContract
(ContractID                 INT  PRIMARY KEY,
 ContractDateSigned         DATE,
 ContractDateDue            DATE,
 ClientID                   INT      REFERENCES zClient (ClientID),
 EmployeeIDMGR              INT      REFERENCES zEmployee (EmployeeID),
 EmployeeIDSales            INT      REFERENCES zEmployee (EmployeeID));
```

```
CREATE TABLE zContract
(ContractID                     INT,
 ContractDateSigned   DATE,
 ContractDateDue                DATE,
 ClientID                       INT,
 EmployeeIDMGR                  INT,
 EmployeeIDSales                INT,
 CONSTRAINT ContractPK PRIMARY KEY (contractID),
 CONSTRAINT ContractFK1 FOREIGN KEY ClientID          REFERENCES zClient(ClientID),
 CONSTRAINT ContractFK2 FOREIGN KEY (EmployeeIDmgr)   REFERENCES zEmployee(EmployeeID),
 CONSTRAINT ContractFK3 FOREIGN KEY (EmployeeIDSales) REFERENCES zEmployee(EmployeeID));
```

How to code a CHECK with a list of values

```
CREATE TABLE          zWorkType
(WorkTypeID           INT PRIMARY KEY,
 WorkDescription      VARCHAR(60),
 WorkCode             CHAR(2)
      CHECK (WorkCode IN (‘01’, ‘A6’, ‘B4’, ‘78’)));
```

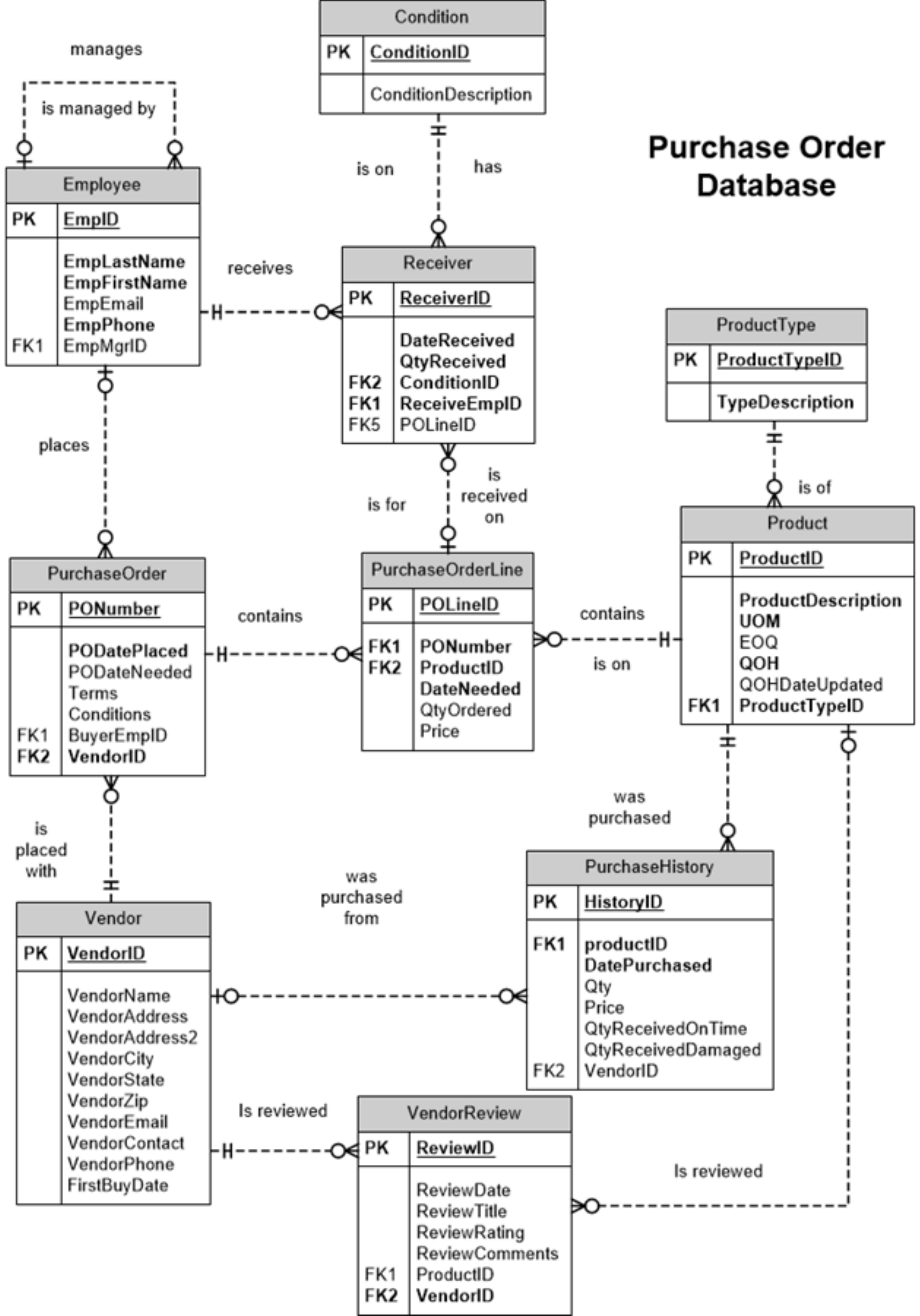
SQL DDL and DML Commands

- SQL Lab 1 teaches:
  - How to create and drop tables.
  - How to create tables with constraints.
  - How to populate tables using INSERT, SQL Server Designer utilities, and the SQL Server Import/Export Wizard.
- What doesn’t it teach?
  - How to change (ALTER) the structure of a table.
  - How to access data from a table beyond a simple SELECT all the data in a table.

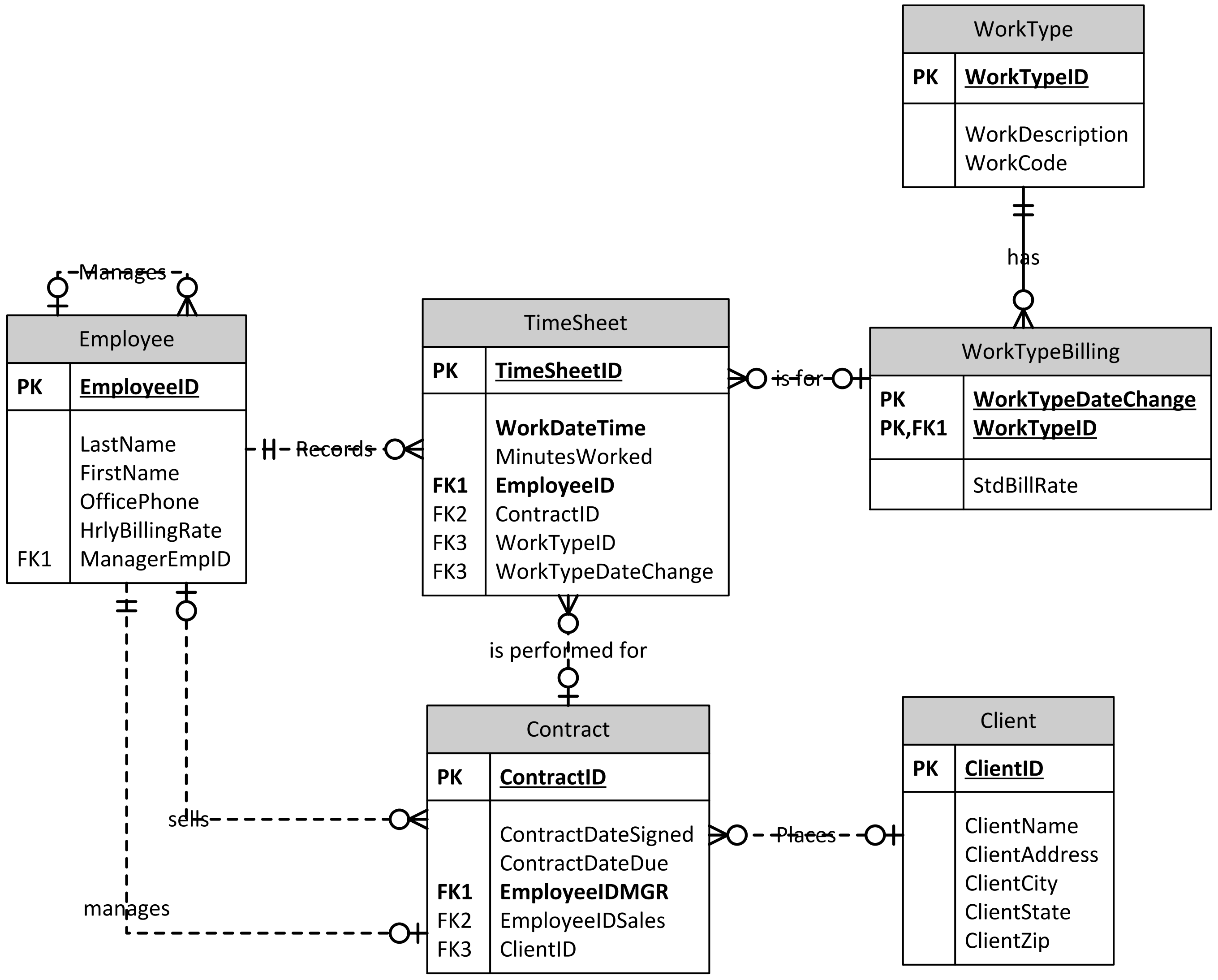
What is required for HW#5

- Create 10 interrelated tables for a database to support a purchase order system.
- Populate 10 interrelated tables using a method of your choice.
- The structure of the tables and the data for the tables is provided for you.
- The data is clean – you do not have to clean/transform it to load it into your tables.

Purchase Order Database



Concatenated primary and foreign keys – not required for HW#5



How to code a concatenated primary key

```
CREATE TABLE zWorkTypeBilling
(WorkTypeDateChange          DATETIME,
 WorkTypeID                  INT CHECK (StdBillRate > 10),
 StdBillRate                 MONEY,
 PRIMARY KEY (WorkTypeDateChange, WorkTypeID),
 FOREIGN KEY WorkTypeID references WorkType(WorkTypeID)
```

A concatenated primary key CANNOT have the declaration of "Primary Key" directly on the primary key attributes. There can be only one primary key declaration, so if the key is concatenated, it must have a primary key declaration as a table-level constraint.

How to code a concatenated foreign key

```
CREATE TABLE zTimeSheet
(TimeSheetID     INT  IDENTITY (1,1) PRIMARY KEY,
 WorkDateTime    DATETIME,
 MinutesWorked   DECIMAL(5,1),
 EmployeeID      INT NOT NULL FOREIGN KEY REFERENCES zEmployee (EmployeeID),
 ContractID      INT FOREIGN KEY REFERENCES zContract (ContractID),
 WorkTypeID      INT,
 WorkTypeDateChange  DATETIME,
 FOREIGN KEY (WorkTypeDateChange, WorkTypeID) REFERENCES zWorkTypeBilling
                (WorkTypeDateChange, WorkTypeID));
```

A concatenated foreign key CANNOT have the declaration of "FOREIGN KEY" directly on the foreign key attributes. IF the key is concatenated, it must have a foreign key declaration as a table-level constraint.