The University of Texas at Austin
McCombs School of Business

MIS 381 INTRO. TO DATABASE MANAGEMENT

Oracle SQL Developer

Data Definition Language

Tayfun Keskin
Visiting Clinical Professor, The University of Texas at Austin, McCombs School of Business
Associate Teaching Professor, University of Washington Seattle, Foster School of Business

QUESTIONS

Any questions

before we begin …

AGENDA

Lecture	Oracle SQL Developer Data Def. Language
Hands-On	Exercises
Looking Forward	Exam 1 Homework 2

REVIEW QUESTION

What is SQL?

WHAT IS SQL?

Structured Query Language: Programming language used to access data stored in your Oracle Database

WHAT IS SQL?

- SQL is an industry standard
- All RDBMS (not only Oracle) implements SQL as the mechanism (language / syntax) to control data
- Think of SQL as the application programming interface (API) – the primary language – to interact with your Oracle Database

WHAT CAN YOU DO WITH SQL?

- **Run queries:** choose what data you want to see
- **Modify data:** insert new data, update or delete existing data
- **Modify objects:** create new tables. Modify the structure of existing tables, such ass adding or deleting columns

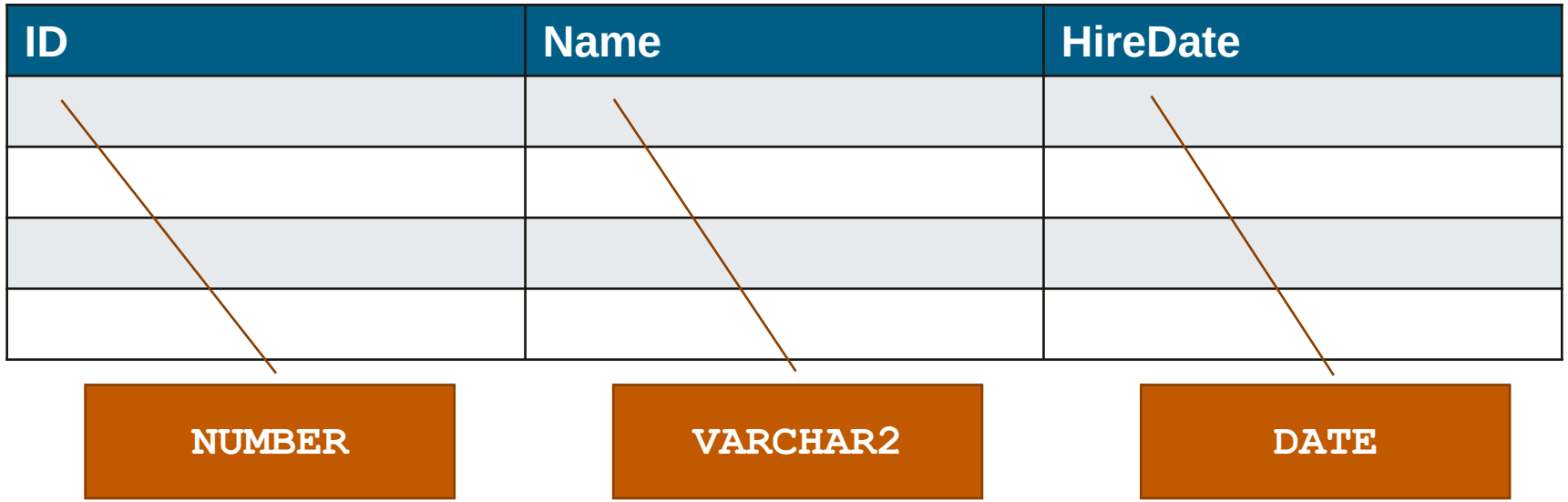
QUESTION

What is the most fundamental logical unit of storage in relational databases?

Where to we store data in relational databases?

DATABASE TABLES

- The most fundamental logical unit of storage in databases
- We must assign a datatype for each column



QUESTION

Why is it important (mandatory) to specify a Datatype in Oracle Database?

SQL QUERY STRUCTURE

- **`SELECT ID, Name`**
  - This clause specifies the column names we want to return (* for all)
- **`FROM Employees`**
  - This clause specifies the table (or tables) we want to retrieve data
- **`WHERE Name = 'John' or Name = 'Jane'`**
  - Optional, filter clause

DDL REFERENCE GUIDE - MURACH

		Components of a table				
		Table	Column	Data Type	Column Constraint	Table Constraint
Actions you can take	Create	You can create a table using the CREATE TABLE command. (pg.315 - Examples 1,2,3)	You create columns as part of the table creation process when using the CREATE TABLE command. (pg.315 - Examples 1, 2, 3). You can also “ADD” a column to an existing table as part of the ALTER TABLE command. (pg.323 – Example 1)	N/A - Technically you don't "create" a data type. Rather you are required to "define" it for each column when you create your table. (pg.315 - Examples 1,2,3)	You can create constraints on a columns when using CREATE TABLE command. Common types are NOT NULL, Unique, Default, Primary Key, and Check. While not always required, you can give a constraint a unique name which makes it easier to edit it later and in error handling (pg.317 - Example 1&2, pg.321 – Example 1) You can also add constraints to an existing table using ALTER TABLE command (pg.325 – Example 7, 8). Note: Oracle requires you ADD unique constraints but to add not null constraint you use MODIFY column command. Constraints are enabled by default when they are created but you can create constraints in disabled state too in case you’re not ready for them to be enabled (pg.325 – Example 3)	You can create/name a table constraint when you use CREATE TABLE. If a constraint involves two or more columns, you must define at table level. Common table constraints that you can create are: Composite PK (pg.317 Example 4), Foreign Key (pg.319 Example 2), and Check (pg.321 Example 2) You can also ADD constraints to an existing table using ALTER TABLE command (pg.325 – Example 1,3,6)
	Edit	You use the ALTER TABLE command to make edits to a table’s columns, column data types, or constraints. (pg.323 – All examples) You can also RENAME a table (pg.327)	You can MODIFY column’s max length or default value with ALTER TABLE (pg.323 – Example 3). You can change a column’s name when you ALTER a table. (Example).	You can MODIFY a column’s data type when you ALTER a table. (pg.323 – Example 4).	There technically isn’t a “modify constraint” command with the exception of using MODIFY to change a column’s default value (pg.323 – Example 5). If you want to edit a constraint, it’s easier to DROP it and then recreate it with ADD command. You can ENABLE or DISABLE a constraint when you ALTER a TABLE. (pg.325 Example 4,5)	There technically isn’t a “modify constraint” command. If you want to edit a constraint, it’s easier to DROP it and then recreate it with ADD command. You can ENABLE or DISABLE a constraint when you ALTER a TABLE. (pg.325 Example 4,5)
	Delete	You can DROP a table which is the same as deleting it. You can also TRUNCATE a table which purges it of data but doesn’t delete it. (pg.327)	You can DROP a column when using the ALTER TABLE command. (pg.323 – Example 2)	N/A - Since every column must have a data type defined, there would never be a case where you’d want to “delete” a data type. You can modify a data type.	You can DROP a constraint from column when you ALTER a table. To do so, reference the name of the constraint. (pg.325 - Example 2)	You can DROP a constraint from column when you ALTER a table. To do so, reference the name of the constraint. (pg.325 - Example 2)

HANDS ON PRACTICE: OPEN ORACLE SQL DEVELOPER

IN-CLASS EXERCISE FILES

- ICE 1 – Basic SQL (using Oracle SQL Developer worksheet)
- ICE 2 – Drop all tables file
- ICE 3 – DDL Practice
- ICE 4 – Members and Committees

LOOKING FORWARD

Read chapters 2 and 10

Exam 1

Homework 2

THANK YOU

BACKUP SLIDES

PART 1

ICE 3 – Chapter 10

The syntax of the CREATE TABLE statement

```
CREATE TABLE [schema_name.]table_name
(
  column_name_1 data_type [column_contsraint],
  column_name_2 data_type [column_constraint] [...],
  table_level_constraints
)
```

Common column constraints

- NOT NULL
- UNIQUE
- DEFAULT

Common data types

- NUMBER
- VARCHAR
- DATE

[] = OPTIONAL

The syntax of the CREATE TABLE statement

```
CREATE TABLE [schema_name.]table_name
(
      column_name_1       data_type    [column_constraints]
    ,[column_name_2       data_type    [column_constraints]]...
    ,[table_level_constraints]
)


CREATE TABLE table_name
(
      column_name_1       data_type     column_constraints
)
```

A statement that creates a table without column attributes

```
CREATE TABLE vendors
(
  vendor_id      NUMBER,
  vendor_name    VARCHAR2(50)
)
```

A statement that creates a table with column attributes

```
CREATE TABLE vendors
(
  vendor_id      NUMBER           NOT NULL     UNIQUE,
  vendor_name    VARCHAR2(50)     NOT NULL     UNIQUE
)
```

Another statement that creates a table with column attributes

```
CREATE TABLE invoices
(
  invoice_id      NUMBER         NOT NULL  UNIQUE,
  vendor_id       NUMBER         NOT NULL,
  invoice_number  VARCHAR2(50)   NOT NULL,
  invoice_date    DATE                     DEFAULT SYSDATE,
  invoice_total   NUMBER(9,2)    NOT NULL,
  payment_total   NUMBER(9,2)             DEFAULT 0
)
```

The syntax of a column-level primary key constraint

```
[CONSTRAINT constraint_name] PRIMARY KEY
```

The syntax of a table-level primary key constraint

```
[CONSTRAINT constraint_name]
PRIMARY KEY (column_name_1 [, column_name_2]...)
```

A table with column-level constraints

```
CREATE TABLE vendors
(
  vendor_id      NUMBER         PRIMARY KEY,
  vendor_name    VARCHAR2(50)  NOT NULL       UNIQUE
)
```

A table with named column-level constraints

```
CREATE TABLE vendors
(
  vendor_id      NUMBER
                 CONSTRAINT vendors_pk PRIMARY KEY,
  vendor_name    VARCHAR2(50)
                 CONSTRAINT vendor_name_nn NOT NULL
                 CONSTRAINT vendor_name_un UNIQUE
)
```

A table with table-level constraints

```
CREATE TABLE vendors
(
  vendor_id      NUMBER,
  vendor_name    VARCHAR2(50)     NOT NULL,
  CONSTRAINT vendors_pk PRIMARY KEY (vendor_id),
  CONSTRAINT vendor_name_uq UNIQUE (vendor_name)
)
```

A table with a two-column primary key constraint

```
CREATE TABLE invoice_line_items
(
  invoice_id               NUMBER         NOT NULL,
  invoice_sequence         NUMBER         NOT NULL,
  line_item_description    VARCHAR2(100)  NOT NULL,
  CONSTRAINT line_items_pk
      PRIMARY KEY (invoice_id, invoice_sequence)
)
```

A table with a column-level foreign key constraint

```
CREATE TABLE invoices
(
  invoice_id      NUMBER    PRIMARY KEY,
  vendor_id       NUMBER    REFERENCES vendors (vendor_id),
  invoice_number VARCHAR2(50) NOT NULL     UNIQUE
)
```

A table with a table-level foreign key constraint

```
CREATE TABLE invoices
(
  invoice_id        NUMBER           NOT NULL,
  vendor_id         NUMBER           NOT NULL,
  invoice_number    VARCHAR2(50)     NOT NULL    UNIQUE,
  CONSTRAINT invoices_pk
    PRIMARY KEY (invoice_id),
  CONSTRAINT invoices_fk_vendors
    FOREIGN KEY (vendor_id)
        REFERENCES vendors (vendor_id)
)
```

The syntax of a check constraint

```
[CONSTRAINT constraint_name] CHECK (condition)
```

A statement with check constraints

```
CREATE TABLE invoices
(
  invoice_id        NUMBER           PRIMARY KEY,
  invoice_total     NUMBER(9,2)      NOT NULL
                    CHECK (invoice_total >= 0),
  payment_total     NUMBER(9,2)      DEFAULT 0
                    CHECK (payment_total >= 0)
)
```

A statement with table-level check constraints

```
CREATE TABLE invoices
(
  invoice_id        NUMBER          PRIMARY KEY,
  invoice_total     NUMBER(9,2)     NOT NULL,
  payment_total     NUMBER(9,2)     DEFAULT 0,
  CONSTRAINT invoices_ck CHECK (invoice_total >= 0
                                AND payment_total >= 0)
)
```

An INSERT statement that fails due to a check constraint

```
INSERT INTO invoices
VALUES (1, 99.99, -10)
```

The response from the system

```
SQL Error: ORA-02290: check constraint (EX.INVOICES_CK)
violated 02290. 00000 -  "check constraint (%s.%s)
violated"
*Cause:    The values being inserted do not satisfy the
           named check
*Action:   do not insert values that violate the
           constraint.
```

The syntax for modifying the columns of a table

```
ALTER TABLE [schema_name.]table_name
{
ADD         column_name data_type [column_attributes] |
DROP COLUMN column_name |
MODIFY      column_name data_type [column_attributes]
}
```

A statement that adds a new column

```
ALTER TABLE vendors
ADD last_transaction_date DATE;
```

A statement that drops a column

```
ALTER TABLE vendors
DROP COLUMN last_transaction_date;
```

A statement that changes the length of a column

```
ALTER TABLE vendors
MODIFY vendor_name VARCHAR2(100);
```

A statement that changes the type of a column

```
ALTER TABLE vendors
MODIFY vendor_name CHAR(100);
```

A statement that changes a default value

```
ALTER TABLE vendors
MODIFY vendor_name DEFAULT 'New Vendor';
```

A statement that fails because it would lose data

```
ALTER TABLE vendors
MODIFY vendor_name VARCHAR2(10);
```

The response from the system

```
SQL Error: ORA-01441: cannot decrease column
length because some value is too big
```

The syntax for modifying the constraints of a table

```
ALTER TABLE table_name
{
ADD        CONSTRAINT   constraint_name
                        constraint_definition [DISABLE] |
DROP       CONSTRAINT   constraint_name |
ENABLE   [NOVALIDATE]   constraint_name |
DISABLE                 constraint_name
}
```

A statement that adds a new check constraint

```
ALTER TABLE invoices
ADD CONSTRAINT invoice_total_ck
        CHECK (invoice_total >= 0);
```

A statement that drops a check constraint

```
ALTER TABLE invoices
DROP CONSTRAINT invoice_total_ck;
```

A statement that adds a disabled constraint

```
ALTER TABLE invoices
ADD CONSTRAINT invoice_total_ck
        CHECK (invoice_total >= 1) DISABLE;
```

A statement that enables a constraint for new values only

```
ALTER TABLE invoices
ENABLE NOVALIDATE CONSTRAINT invoice_total_ck;
```

A statement that disables a constraint

```
ALTER TABLE invoices
DISABLE CONSTRAINT invoice_total_ck;
```

A statement that adds a foreign key constraint

```
ALTER TABLE invoices
ADD CONSTRAINT invoices_fk_vendors
FOREIGN KEY (vendor_id) REFERENCES vendors (vendor_id);
```

A statement that adds a unique constraint

```
ALTER TABLE vendors
ADD CONSTRAINT vendors_vendor_name_uq
UNIQUE (vendor_name);
```

A statement that adds a not null constraint

```
ALTER TABLE vendors
MODIFY vendor_name
CONSTRAINT vendors_vendor_name_nn NOT NULL;
```

How Oracle handles new constraints

- By default, Oracle verifies that existing data satisfies a new constraint.
- If that's not what you want, you can add a disabled constraint.

A statement that renames a table

```
RENAME vendors TO vendor
```

A statement that deletes all data from a table

```
TRUNCATE TABLE vendor
```

A statement that deletes a table from the current schema

```
DROP TABLE vendor
```

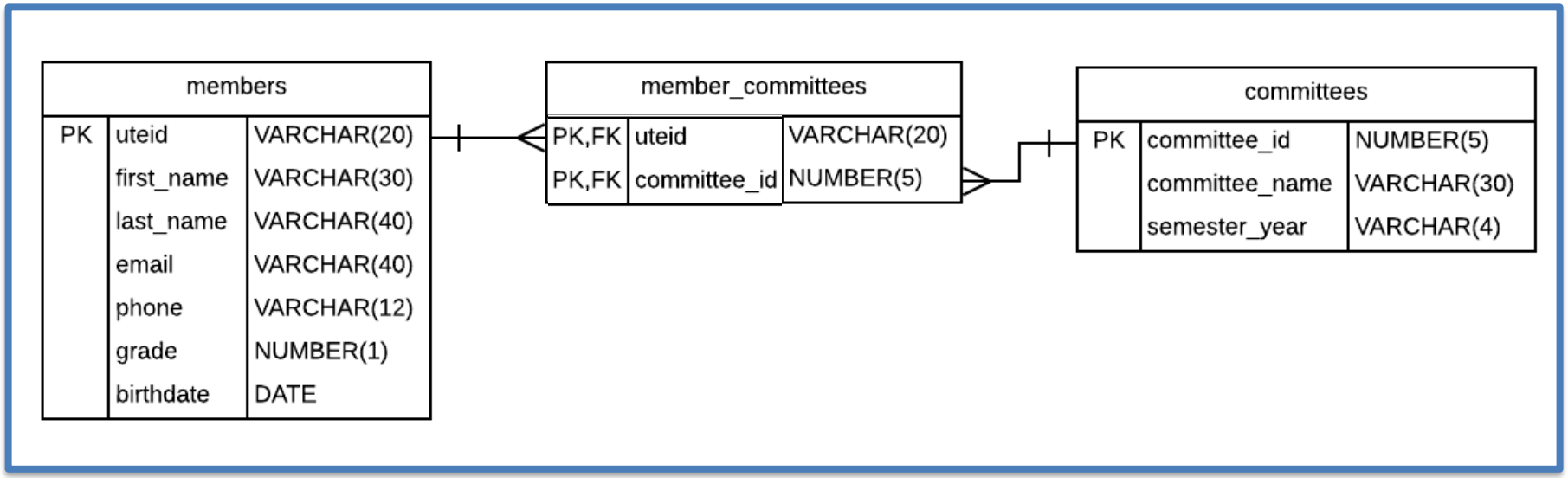
A statement that qualifies the table to be deleted

```
DROP TABLE ex.vendor
```

PART 2

ICE 4 - UBC Committee Track

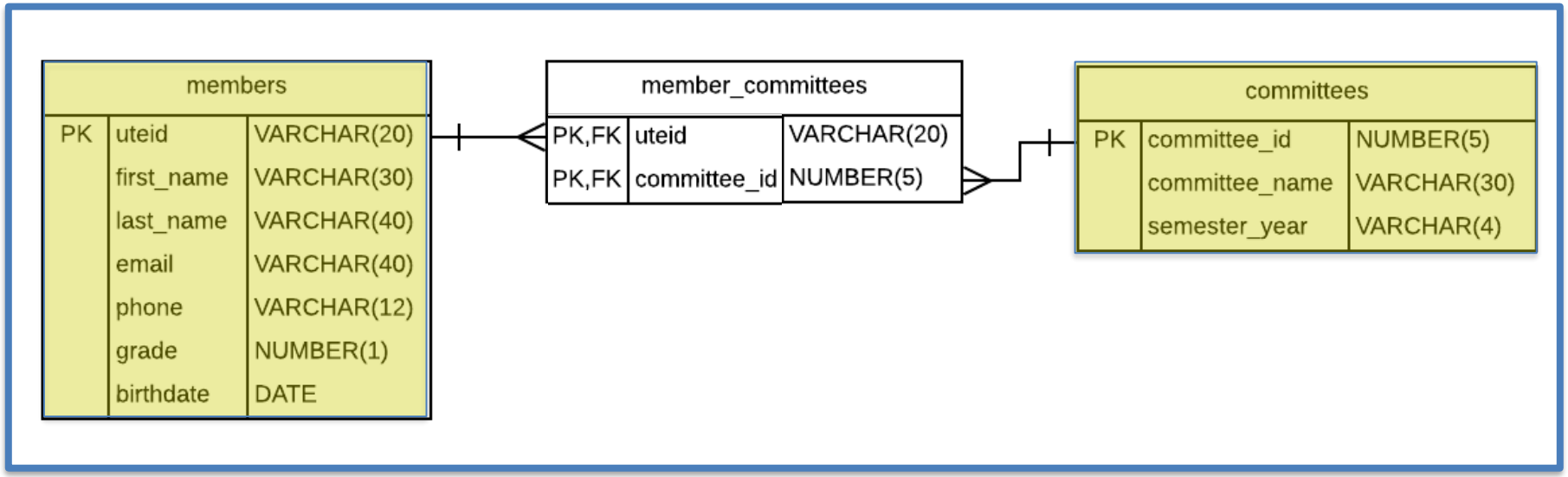
Let's Practice: UBC Committee Track (w/o Roles)



Practice: Create table with column-level constraints

Task:

1. Create the ***members*** table with the key and data types noted in ERD and test with seeding data
2. Create the ***committees*** table with the key and data types noted in ERD and test



NOTE:

- Always make sure your code is **clean/readable**

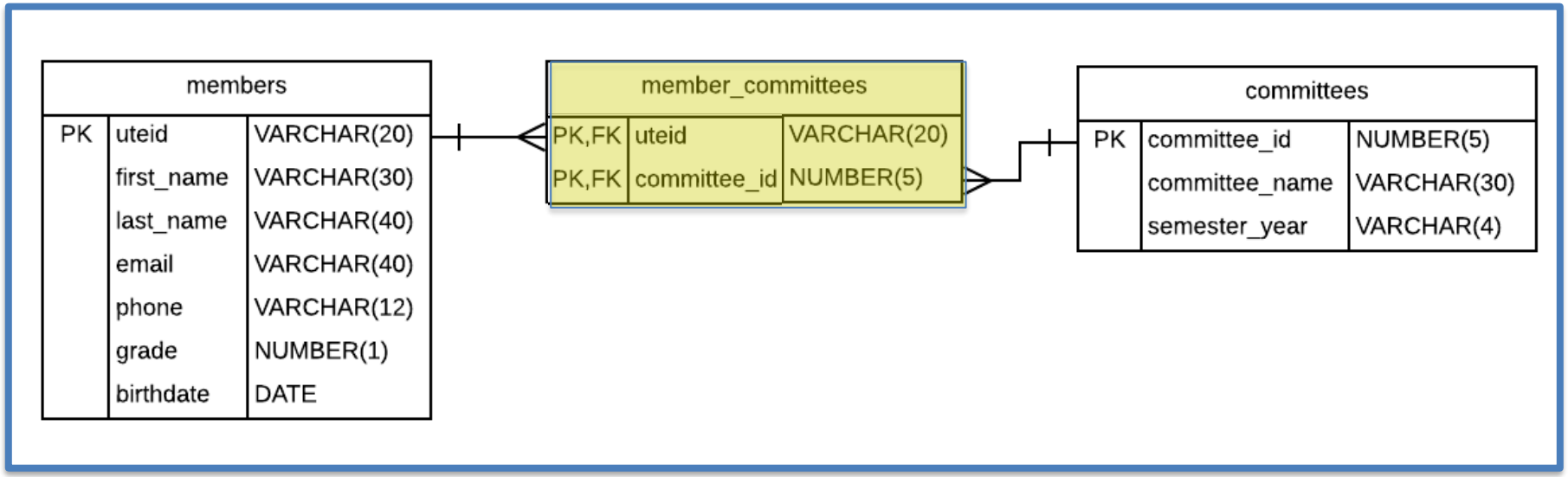
Syntax Hint:

```
CREATE TABLE table_name
(
    column_name_1    data_type    [column_constraints],
    column_name_2    data_type    [column_constraints]
)
```

Practice: Create tables with table-level constraints

Task:

3. Create the ***members_committees*** table with composite PKs and two FKs. This requires table-level constraints.
4. Test constraints work



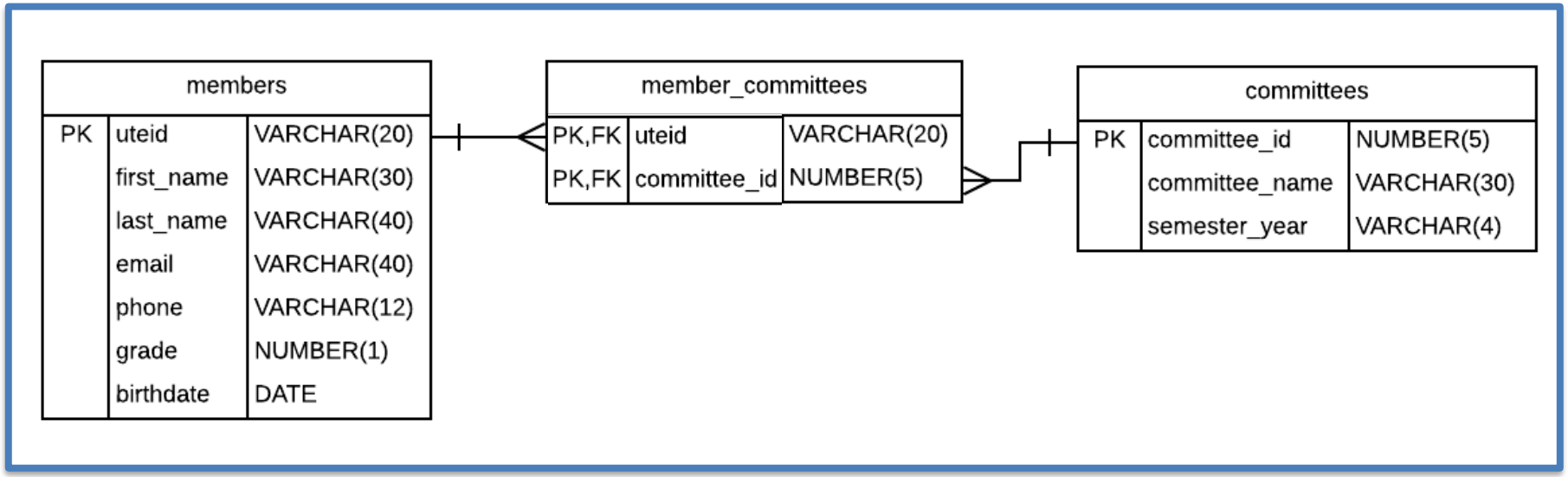
Syntax Hint:

```
CREATE TABLE table_name
(
      column_name_1      data_type     [column_constraints],
      column_name_2      data_type     [column_constraints],
      CONSTRAINT [constraint_name] PRIMARY KEY (column_name_1, column_name_2),
      CONSTRAINT [constraint_name] FOREIGN KEY (fk_column) REFERENCES table (pk_column)
)
```

Practice: Add a table-level *check* constraint

Task:

1. Add Check constraint to **members**. ***grade*** should be greater than 0.
2. Add a Check constraint ***grade*** to only allow grade column to contain a value of 1,2,3, or 4.



Syntax Hint:

```
ALTER TABLE table_name
      ADD CONSTRAINT constraint_name CHECK (column_name condition) [DISABLE];
```

Practice: Editing table components

- ❑ Change a column name. Note: this isn't in book so you'll need to google it.
- ❑ Change the length of a column. e.g. UTEID should be a max length of 10.
- ❑ Change a column's data type. e.g. Change phone from VARCHAR to CHAR
- ❑ Add a new column with a default constraint. e.g. Add a "status" column to committees with a default value of "active". Test constraint.
- ❑ Edit the "status" column by modifying the data max length to be 1 and the default constraint to be "A". Also add a check to allow a value of "A" or "I" only for active or inactive.
- ❑ Change the name of a table. Refresh left panel in SQL Developer to see change took affect

Syntax for modifying the table columns

```
ALTER TABLE table_name
ADD         column_name   data_type  [column_constraint]
DROP COLUMN column_name
MODIFY      column_name   data_type  [column_constraint]
}
```

Syntax for modifying the table constraints

```
ALTER TABLE table_name
ADD       CONSTRAINT  constrnt_name  definition [DISABLE]
DROP      CONSTRAINT  constrnt_name
ENABLE  [NOVALIDATE]  constrnt_name
DISABLE               constrnt_name
```

Practice: Deleting table components

- ❑ Drop a constraint
- ❑ Drop a column
- ❑ Truncate a table
- ❑ Drop a table

Syntax for modifying the table columns

```
ALTER TABLE table_name
ADD         column_name   data_type  [column_constraint]
DROP COLUMN column_name
MODIFY      column_name   data_type  [column_constraint]
}
```

Syntax for modifying the table constraints

```
ALTER TABLE table_name
ADD       CONSTRAINT  constrnt_name  definition [DISABLE]
DROP      CONSTRAINT  constrnt_name
ENABLE  [NOVALIDATE]  constrnt_name
DISABLE               constrnt_name
```

Syntax to rename a table

```
RENAME table_orig_name TO table_new_name
```

Syntax to purge all data in a table

```
TRUNCATE TABLE vendor
```

Syntax to delete a table

```
DROP TABLE vendor
```

Practice: Try using GUI to manipulate your database

- ❑ Drop a constraint
- ❑ Drop a column
- ❑ Truncate a table
- ❑ Drop a table