The Relational Model
(Part I)

Annoucement

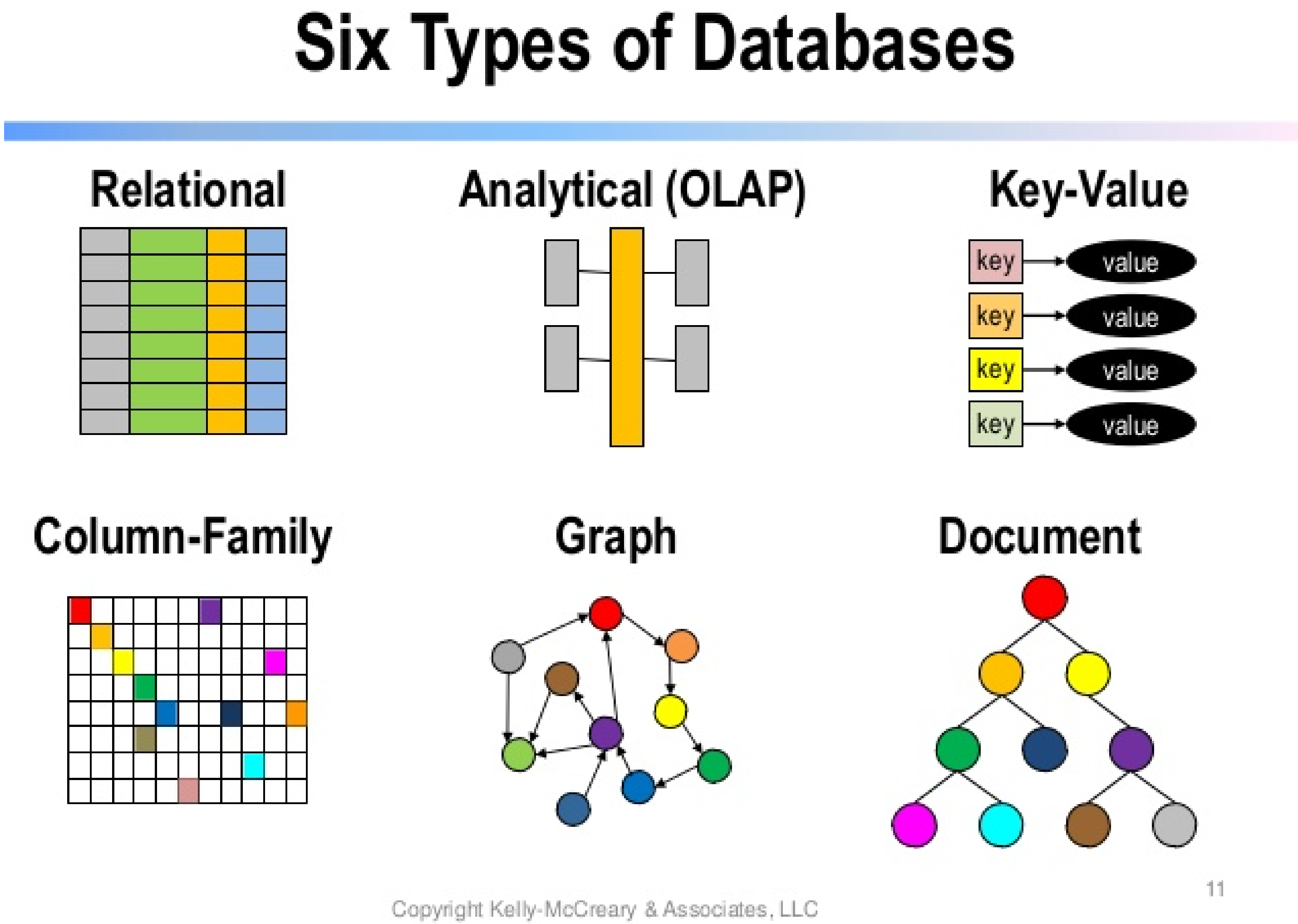
- **Next Tuesday (9/20) class is cancelled**
  - Instructor on conference travel

Steps in Database Design

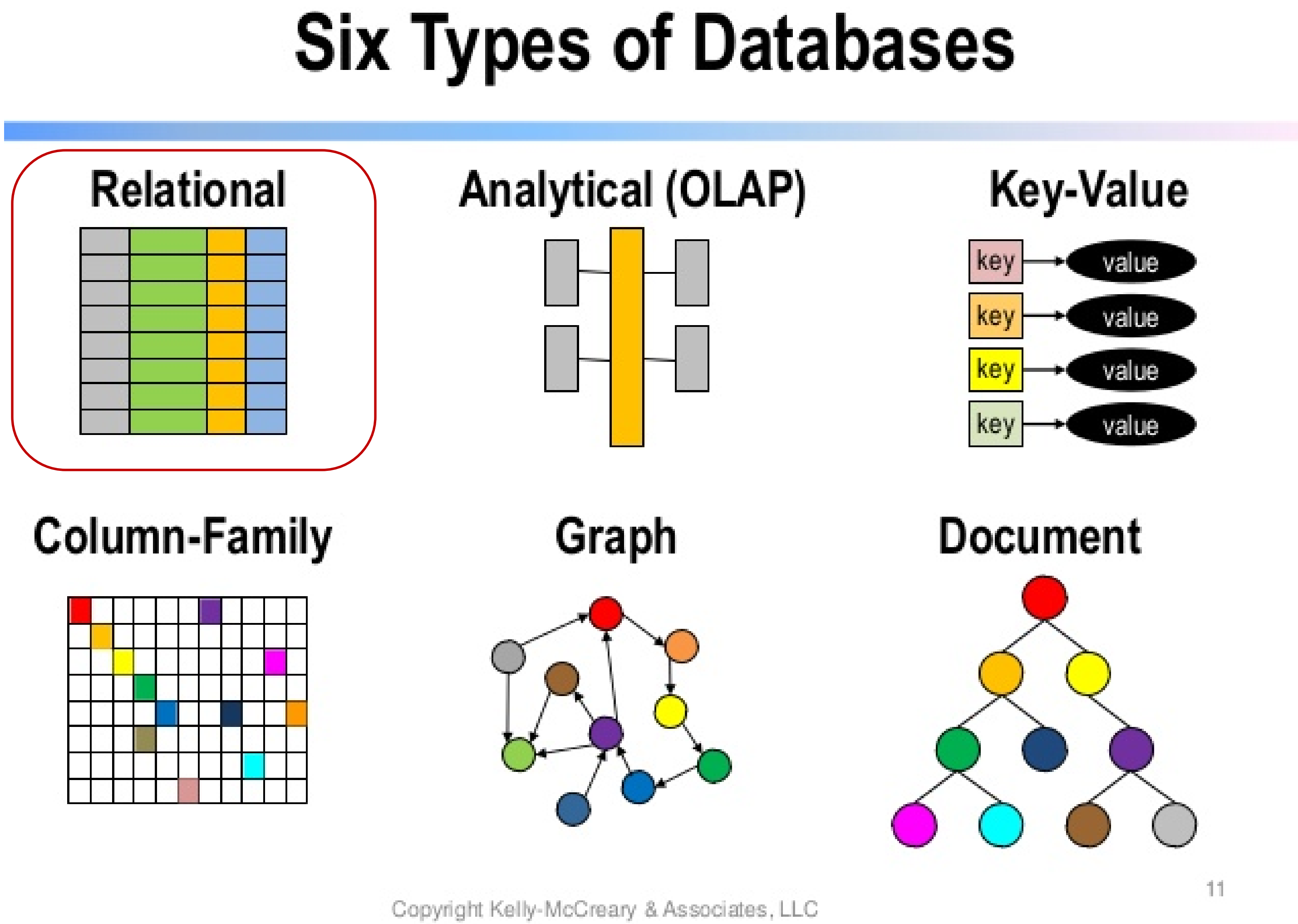
- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - high level description: ER
- Logical Design
  - translate ER into DBMS data model
- Schema Refinement
  - consistency, normalization

Today's lecture

- **Introduction to DBMS data model**
  - Concepts (table, schema, row, column, ...)
  - Using SQL to create table, add and delete tuples
  - Integrity constraints


Six Types of Databases
Relational
Analytical (OLAP)
Key-Value
key
value
key
value
key
value
key
value
Column-Family
Graph
Document
11
Copyright Kelly-McCreary & Associates, LLC

CS442
focus


Six Types of Databases
Relational
Analytical (OLAP)
Key-Value
key
value
key
value
key
value
key
value
Column-Family
Graph
Document
11
Copyright Kelly-McCreary & Associates, LLC

Relational Database: Definition

- ***Relational database:*** **a set of *relations*.**
- ***Relation:*** **made up of 2 parts:**
  - *Schema* : specifies the name and *attributes* of relation
  - *Instance*
    - A *table* with rows and columns.
    - Consistent with schema

Relational Schema

- A *Schema* for a relation is represented in the format:

  relation_name (attr1:type, … attrn:type)

  – Example:

  Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *GPA*: real)

- Attributes are referenced by name, not column locations
- Attribute names must be unique

Exercise

Student	Course	Course
Anna	DB Mgt.	Operating System
Bob	DB Mgt.	Web Programming
Cathy	Operating System	Artificial Intelligence

- **Is this table a valid relational table?**

Exercise (Cont.)

Student	Course1	Course 2
Anna	DB Mgt.	Operating System
Bob	DB Mgt.	Web Programming
Cathy	Operating System	Artificial Intelligence

- **Is this table a valid relational table?**

Relational Instances

- *Instance:* a *table*, with rows and columns
- *Attributes* (or fields) are stored in columns.
- *Tuples* (or records) are stored in rows.
- Attributes have a domain – an atomic type.
- The *cardinality* of the relation *R* = the number of rows in R (excludes the first row!)
- The *degree/arity* of the relation *R* = the number of columns in R

Notes of Relational Model

- No duplicate tuples in a relation
  - Question: what if we want to insert duplicate tuples? What can we do?
- Ordering
  - No ordering of tuples in a relation
- The value of each attribute is either drawn from its domain or the special value *NULL*
- Attribute's values are *atomic*.

Example Instance of Students Relation

Schema

Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *GPA*: real)

Instance

SID	Name	Login	Age	GPA
53666	Jones	Jones@cs	18	3.4
53668	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree/arity = 5.

Questions:

(1) Do the values in each column have to be distinct?
(2) Does each tuple (i.e., each row) have to be distinct?

Today's lecture

- **Introduction to DBMS data model**
  - Concepts (table, schema, row, column, ...)
  - Using SQL to create table, add and delete tuples
  - Integrity constraints

Defining a Relation Schema in SQL

- **SQL (pronounced SEQUEL): standard language to describe and manipulate relational database**
  - Data Definition Language (DDL)
    - Create, modify, delete relations
    - Specify constraints
    - Administer users, security, etc.
  - Data Manipulation Language (DML)
    - Specify *queries* to find tuples that satisfy criteria
    - Add, modify, remove tuples

Creating Schema in SQL

- SQL syntax

```
CREATE TABLE table_name (
            field1 datatype,
            field2 datatype,
            …
);
```

Data Types in CREATE TABLE Statement

All attributes must have a data type

1) **Character**: strings of fixed or varying length

CHAR(n): a fixed-length string of *n* character

VARCHAR(n): a set of characters of up to *n* characters.

2) **BOOLEAN**: an attribute whose value is logical

- The possible values are TRUE, FALSE, and UNKNOWN

3) **DATE & TIME**: represent dates and times

Data Types (Cont.)

4) **INT or INTEGER**: denotes typical integer values

5) **FLOAT**: denotes floating-point numbers
   - Real numbers with a fixed decimal point
   - **DECIMAL (n,d)** allows values that consists of n decimal digits, with the decimal point assumed to be *d* positions from the right

Example: Creating Schema in SQL

- **SQL syntax**

CREATE TABLE <name> (<field> <domain>, …)

- **Example: creates the Students relation.**

Schema

Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *GPA*: real)

```
CREATE TABLE Students
      (sid CHAR(20),
       name CHAR(20),
       login CHAR(10),
       age INTEGER,
       GPA FLOAT);
```

Note: the type (domain) of each field is specified, and enforced by the DBMS

Creating Schema in SQL - Exercise

Write the SQL statement that creates the *Enrolled* table.

Schema

Enrolled (*sid*: string, *cid*: string, *grade:real*)

Case-sensitivity of Table/Column Names

- **Is the table/column name in SQL statement case sensitive?**
  - By default,
    - Case-sensitive on Linux
    - Case-insensitive on Windows
  - MySQL has a configuration option to enable/disable it.

Creating Instances in SQL

SQL syntax

```
INSERT INTO table_name (field1, field1, …)
VALUES (value1, value2, …);
```

Note: the order of columns and their values must match

- **Insert a single tuple**

```
INSERT INTO  Students (SID, Name, Login, Age, GPA)
VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2);
```

- **Insert multiple tuples**

```
INSERT INTO  Students (sid, name, login, age, GPA)
     VALUES ('53666', 'Jones', 'jones@cs', 18, 3.4),
            ('53690', 'Smith', 'smith@eecs', 18, 3.2),
            ('53650', 'Smith', 'smith@math', 19, 3.8);
```

Creating Instances in SQL (Cont.)

- Insert data only in specified columns

```
INSERT INTO table_name (field1, field1, …)
 VALUES (value1, value2, …)
```

Example:
```
INSERT INTO Students (SID, Name)
       VALUES('53688', 'Smith’);
```

Note: key attributes must have inserted values.

- Insert data in all columns
  - Do not need to specify the column names in the statement.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Example:
```
INSERT INTO Students
       VALUES('53688', 'Smith', 'smith@ee’, 18, 3.5);
```

  - Make sure the order of the values is in the same order as the columns in the table.

Today's lecture

- **Introduction to DBMS data model**
  - Concepts (table, schema, row, column, ...)
  - Using SQL to create schema and table
  - Integrity constraints

Integrity Constraints

- **Integrity constraints (ICs): conditions specified on a database schema**

- **Types of ICs**
  - Keys
  - Foreign keys
  - Domain constraints: (e.g., the age of the driver license holders must be at least 16)

Superkey, Key, Primary Key, Candidate Key

- Similar to superkey, key, and primary key in ER diagram
- **Superkey**: A set of fields is a *superkey* if no two distinct tuples have same values in these key fields
  - *ALL* attributes of a relation together form a super key for the relation
- **Key**: Minimal superkey (no subset of the key is a superkey)
- **Primary key** and **candidate keys**:
  - >1 key for a relation?
    - One of the keys is chosen (by DBA) to be the *primary key*.
    - The other keys are called as *candidate keys*

Non-NULL Key Values in Relational Models

- All the keys (primary and candidate keys cannot contain a NULL value on any of their attributes
  - E.g., if the key is defined as a composite key (A, B), all records must have values on both A and B attributes.
    - Nulls value on either attribute A or B is not allowed.

Multiple Keys

- An instance can have multiple keys.
  - For example, the *student* table can have 3 keys: (SID), (Name, BirthDate), and (Name, DormRoomNumber)

- Composite keys can overlap
  - For example, (Name, BirthDate) and (Name, DormRoomNumber)

- Question: can the composite keys and singleton keys overlap?

Exercise

SID	Name	Login	Age	GPA
53666	Jones	Jones@cs	18	3.4
53668	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- **Question 1**: for the relation above, can the attribute *Name* be used as a key? What about *Login*? What about *(Name, Age)*?
- **Question 2**: for the relations below, are the following keys correct?

CID	SID	Grade
CS442	27001	A
CS510	27001	B
CS443	21097	A
CS510	24331	B

Key1: (CID)
Key2: (SID)
Key3: (CID, Grade)
Key4: (CID, SID, Grade)

Name	Grade
Alice.S	C
Alan.B	B
Adam.H	A
Alan.B	B

Key1: (Name)
Key2: (Name, Grade)

Nice Properties of Keys

Given a set of attributes A (can have 1 or multiple attributes):

- If A is a key, then any superset A′ of A ($A \subseteq A'$) CANNOT be a key (*because key must be minimal*)
  - E.g., if (SSN) is a key, then any superset of SSN is not a key

- If A is not a key, then any subset A′ of A ($A' \subseteq A$) CANNOT be a key.
  - E.g., if (Name, BirthDate) is not a key, then neither (Name) nor (BirthDate) is a key

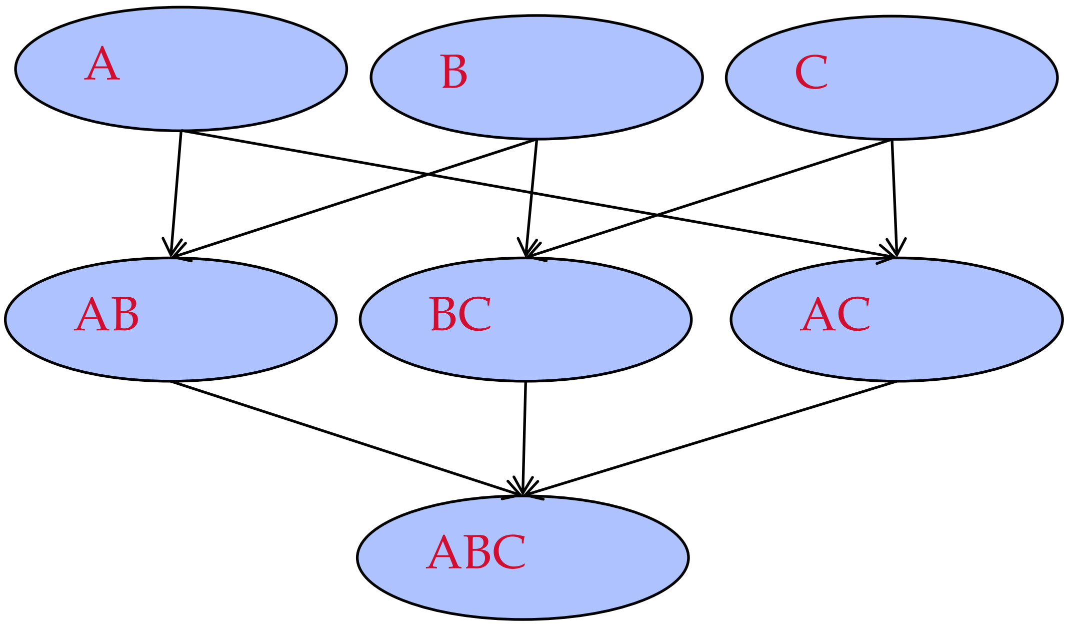
How to Find *All* Keys Efficiently?

- Naïve way: enumerate all possible attribute
  - E.g., A table with 3 attributes (A, B, C)
  - All possible attribute sets (7 in total): (A), (B), (C), (A, B), (A, C), (B, C), (A, B, C)
  - There are $2^k - 1$ such sets in total for k attributes

- An efficient way
  - Construct an *attribute lattice* (explained in the next slide) to enumerate all possible combinations of attributes
  - Use the two properties of keys (previous slide) when traversing the attribute lattice to find keys.

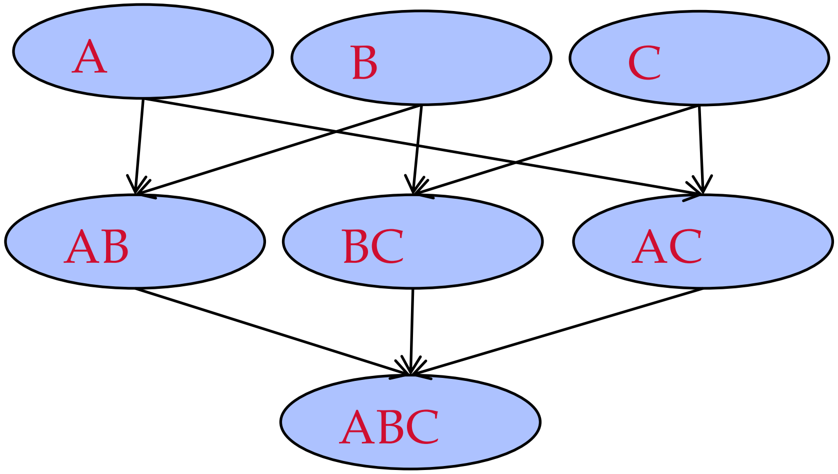
Attribute Lattice

- Top of lattice: each node contains a single attribute
- Bottom of lattice: one node that contains all attributes
- The *ith* level of lattice: each node contains *i* attributes
- Edge N1->N2: N1 is a strict subset of N2



An example of attribute lattice of 3 attributes {A, B, C}

Finding *ALL* Keys from Attribute Lattice



- Starting from the top of lattice, for each node N
  - Check if N can be a key.
  - If N is a key, remove all descendants of N from lattice (because they cannot be keys as they are not minimal).
- Continue until all nodes in the lattice are traversed.

Exercise

- Consider the following facts, list ALL keys.
  - Each student has a student ID (SID), his/her name, and age;
  - Each student has a unique SID;
  - Some students can have the same name;
  - Some students can have the same age;
  - No two students have the same name and the same age.

Define Primary Keys by SQL

SQL Syntax for Primary Key definition:

```
CREATE TABLE table_name (
            field1 datatype,
            field2 datatype,
             …
            PRIMARY KEY (att1, att2…)
 );
```

Notes:

- The key can contain either 1 attribute or multiple attributes
- Always put the key attributes in parentheses, even when there is only one attribute.

Example: Define Keys

- **Example: creates the Students relation.**

Schema

Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *GPA*: real)

```
CREATE TABLE Students
        (sid CHAR(20),
         name CHAR(20),
         login CHAR(10),
         age INTEGER,
         GPA FLOAT,
          PRIMARY KEY (sid));
```

Define Candidate Keys in SQL

```
CREATE TABLE table_name (
                field1 datatype,
                field2 datatype,
                 …
                PRIMARY KEY (k1_att1, k2_att2…),
                UNIQUE (ck1_att1, ck1_att2…),
                UNIQUE (ck2_att1, ck2_att2…)
                …
   );
```

Notes:

- K candidate keys should have k UNIQUE statements, one statement per candidate key.
- Always put the key attributes in parentheses, even when there is only one attribute.

A Simpler Way to Define Singleton Keys

Singleton keys (either primary or candidate keys)

- Place PRIMARY KEY or UNIQUE right after the type in the declaration of the key attribute.
- This is only allowed for singleton keys.
  - Composite keys only can be defined by using PRIMARY KEY and UNIQUE statements.
- Example:

```
CREATE TABLE Students (
                       SID CHAR(20) PRIMARY KEY,
                       SSN CHAR(9) UNIQUE,
                       age CHAR(20)
 );
```

Exercise

sid	cid	grade
10001	CS442	A
10002	CS442	B
10001	CS510	A

- Given schema: *Enrolled (sid: string, cid: string, grade: real*)
- Facts:
  - Each student has a unique sid.
  - Each course has a unique cid.
  - Each student can enroll in multiple courses. But each student can take the same course only once.
  - No two students in the same course receive the same grade.
- Questions:
  - What are the keys? (tips: there are 2 keys)
  - Write the SQL statement to define the *Enrolled* table with the key constraints.