**1/29/2025**

Self annotation/notes

A screenshot of a computer

Description automatically generated

* It is important to be familiar with the basic terms and concepts of databases in this course.
* A database system is built by using a Database Management System (DBMS).
* One popular DB engine ranking: <http://db-engines.com/en/ranking>.
* We focus on Relational DBMS (RDBMS).
* Examples of Relational DBMS:
  1. Access: most popular 'departmental' DBMS
  2. Oracle: Most popular commercial DBMS
  3. MS SQL server: Likely second most popular commercial DBMS
  4. MySQL: most popular open source DBMS
  5. MariaDB: highly compatible to MySQL (more open source than MySQL)
  6. Postgres: popular open source DBMS known for innovation and functionality.
  7. SQLite: most popular portable DB engine.
* Relational DBMS basically use the relational model (with extensions).
* There are many other models. Examples:
  1. Object-Oriented Database (OODB): e.g., db4o, Gemstone, etc.
  2. Big Data:
     + Document DB: e.g., MongoDB, CouchDB
     + Key-Value DB: e.g., Redis, LevelDB
     + Wide Column DB: optimized over large dataset; store columns together, not rows. E.g. Cassandra and HBase.
     + Graphical DB: e.g., Neo4J

**The Relational Model**

by K. Yue

**1. Introduction**

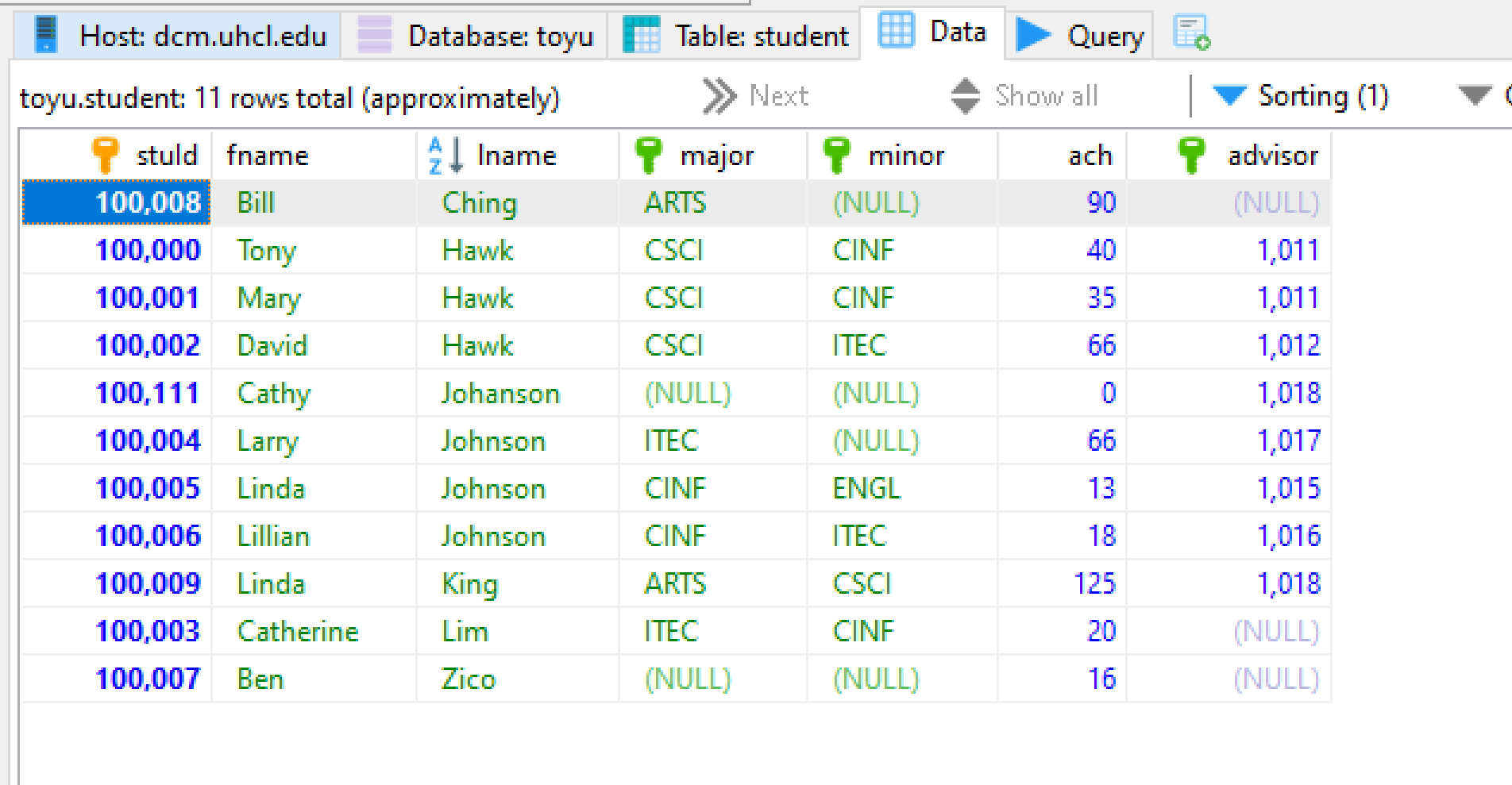
**1.1 Data Model**

* There are many *data models* used by database systems.
* The **data models** of database systems define how data is organized, structured, connected, processed, and queried in the databases.
* It is important to recognize the basic data structures used by these models.
* Examples:
  1. Relational model: set-theoretic relation/table
  2. Excel: table, and to be more exact, a two-dimensional array
  3. Object-oriented model: directed graphs
  4. XML: trees with many different types of nodes, plus sets of attributes.
  5. Cassandra: columnar or wild column model
  6. MongoDB: document model (semi-structured, unstructured)
  7. Neo4J: graph model

**1.2 The Relational Model: an introduction**

* The basic relational data model in layman terms:
  + A database is composed of a collection/sets of *tables* (relations).
  + A table contains many *rows* (tuples) and *columns* (attributes/fields).
  + Each row contains many *column values*.
  + Every row of a table has the same set of columns. (Structured Data)
  + A screenshot of a computer

    Description automatically generated



* + Values of the same column have the same data *type*.
  + Keys are sets of columns/attributes.
  + A *candidate key* of a table is a *minimal unique identifier* of a row in the table.

E.g. Student:

CK: (1) StuId

CREATE TABLE IF NOT EXISTS Student (

stuId INT NOT NULL,

fname VARCHAR(30) NOT NULL,

lname VARCHAR(30) NOT NULL,

major CHAR(4) NULL,

minor CHAR(4) NULL,

-- ach: accumulated credit hours, including transferred credits.

ach INTEGER(3) UNSIGNED NULL DEFAULT 0,

advisor INT NULL,

CONSTRAINT Student\_stuId\_pk PRIMARY KEY(stuId),

-- an artificial example of a CHECK constraint.

CONSTRAINT Student\_ach\_cc CHECK ((ach>=0) AND (ach < 250)),

CONSTRAINT Student\_major\_fk FOREIGN KEY (major)

REFERENCES Department(deptCode) ON DELETE CASCADE,

CONSTRAINT Student\_minor\_fk FOREIGN KEY (minor)

REFERENCES Department(deptCode) ON DELETE CASCADE,

CONSTRAINT Student\_advisor\_fk FOREIGN KEY (advisor)

REFERENCES Faculty(facId)

);

E.g. Department:  
  
CK: (1) DeptCode (e.g. CSCI): PK: simple (one attributre in the key)

(2) DeptName: secondary key

CREATE TABLE IF NOT EXISTS Department (

deptCode CHAR(4) NOT NULL,

deptName VARCHAR(30) NOT NULL,

schoolCode CHAR(3) NULL,

numStaff TINYINT NULL,

CONSTRAINT Department\_deptCode\_pk PRIMARY KEY (deptCode),

-- alternate keys: [1] deptName

CONSTRAINT Department\_name\_ck UNIQUE (deptName),

CONSTRAINT Department\_schoolCode\_fk FOREIGN KEY (schoolCode)

REFERENCES School(schoolCode)

);

Student: {stuId, major} an unique identifier? Yes; not minimal; not a CK.

E.g. Enroll:

CS/PK: (1) {stuId, classId}: composite key (more than one attribute in the key)

* + A *primary key* is a selected candidate key, CK (for storing the table).
  + *Alternative/secondary keys* are candidate keys not selected as the primary key.
  + A *foreign key* of a relation references a primary key of another relation (known as the parent or referenced table).

E.g. join student s with department d:

s.major = d.deptCode

**Foreign keys:**  
  
1. Student(advisor) references Faculty(facId)  
2. Student(major) references Department(deptCode)  
3. Student(minor) references Department(deptCode)  
4. Faculty(deptCode) references Department(deptCode)  
5. Department(schoolCode) references School(schoolCode)  
6. Enroll(stuId) references Student(stuId)  
7. Enroll(classId) references Class(classId)  
8. Enroll(grade) references Grade(grade)  
9. Class(courseId) references Course(courseId)  
10. Class(facId) references Faculty(facId)  
11. Course(Rubric) references Department(deptCode)

More theoretically:

* The (theoretical) relational model is based on the concept of a relation.
* It is a *set-theoretic* model: the definitions are based on mathematical sets.
* If you are not familar with set theory, read about it. This is a basic, short, good, and good-enough introduction: <https://www.ucl.ac.uk/~ucahmto/0005_2021/Ch2.S1.html> (note that in the set builder form, the author used ":" to represent "such that". We usually use "|" instead.)
* Note that practical DBMS do not implement the pure relational model.
* In the theoretical relational model:
  1. An *attribute* (*column/field*) is a name.
  2. A *domain* is a *set* of values an attribute can take.
     1. It is the set of values of the*data type* of the attribute.
     2. The value of an attribute should be *atomic* (cannot be divided into smaller components with individual meanings):
        1. If all attributes of a relation are atomic, the relation is said to be in *First Normal Form*.
     3. *Null* may or may not be an acceptable value for an attribute. It depends on problem requirements.
  3. A *relation schema*, R, is a *set* of attributes A1, A2,…,An with their domains D1, D2,…, Dn.
* Schema: structure of a table.
* R: Student  
  { A1, A2,…,An }: {stuId, fname, lname, ... }
* { domains D1, D2,…, Dn}: {INT, VARCHAR(30),…
* CREATE TABLE IF NOT EXISTS Student (
* stuId INT NOT NULL,
* fname VARCHAR(30) NOT NULL,
* lname VARCHAR(30) NOT NULL,
* major CHAR(4) NULL,
* minor CHAR(4) NULL,
* -- ach: accumulated credit hours, including transferred credits.
* ach INTEGER(3) UNSIGNED NULL DEFAULT 0,
* advisor INT NULL,
* CONSTRAINT Student\_stuId\_pk PRIMARY KEY(stuId),
* -- an artificial example of a CHECK constraint.
* CONSTRAINT Student\_ach\_cc CHECK ((ach>=0) AND (ach < 250)),
* CONSTRAINT Student\_major\_fk FOREIGN KEY (major)
* REFERENCES Department(deptCode) ON DELETE CASCADE,
* CONSTRAINT Student\_minor\_fk FOREIGN KEY (minor)
* REFERENCES Department(deptCode) ON DELETE CASCADE,
* CONSTRAINT Student\_advisor\_fk FOREIGN KEY (advisor)
* REFERENCES Faculty(facId)
* );

A screenshot of a computer

Description automatically generated

* 1. A *tuple* (*row*) is a *set* of *mapping* of a *set* of attributes to a *set* of values: Ai -> di where di ∈ Di, for i = 1 to n (∈: belongs to)
  2. A *relation* (*instance*) is a set of tuples.
  3. The *degree* (or *arity*) of a relation is the number of attributes in its schema.
* Some advantages of the *relational model* and relational DBMS (as compared to other databases):
  1. Strong mathematical foundation
  2. Simple
  3. Strong design theory
  4. Strong support of data integrity and consistency
  5. Strong support of transactions
  6. Strong industrial support and community
  7. High popularity
* Some disadvantages of the relational model and relational DBMS:
  1. The data model may not match the problem requirements well.
  2. Impedance mismatch with object-oriented models.
  3. Do not scale well.
  4. Structured data may be too restrictive for specific problems.

**1.3 Toyu Example**

[Toyu](https://dcm.uhcl.edu/yue/courses/joinDB/Spring2025/notes/toyu/toyu.html): A drastically simplified university  
  
SQL statement: SELECT \* FROM class;

Select every column from the table class.

MariaDB [toyu]> SELECT \* FROM class;  
+---------+----------+----------+------+-------+------+  
| classId | courseId | semester | year | facId | room |  
+---------+----------+----------+------+-------+------+  
|   10000 |     2000 | Fall     | 2019 |  1011 | D241 |  
|   10001 |     2001 | Fall     | 2019 |  1011 | D242 |  
|   10002 |     2002 | Fall     | 2019 |  1012 | D136 |  
|   10003 |     2020 | Fall     | 2019 |  1014 | D241 |  
|   10004 |     2021 | Fall     | 2019 |  1014 | D241 |  
|   10005 |     2040 | Fall     | 2019 |  1015 | D237 |  
|   10006 |     2041 | Fall     | 2019 |  1019 | D217 |  
|   10007 |     2060 | Fall     | 2019 |  1020 | B101 |  
|   10008 |     2080 | Fall     | 2019 |  1018 | D241 |  
|   11000 |     2000 | Spring   | 2020 |  1011 | D241 |  
|   11001 |     2001 | Spring   | 2020 |  1012 | D242 |  
|   11002 |     2002 | Spring   | 2020 |  1013 | D136 |  
|   11003 |     2020 | Spring   | 2020 |  1016 | D217 |  
|   11004 |     2061 | Spring   | 2020 |  1018 | B101 |  
+---------+----------+----------+------+-------+------+  
14 rows in set (0.005 sec)

A tuple/row: {classId: 10004, courseId: 2021, semester: 'Fall', year: 2019, facId: 1014, room: 'D241'}. Note that it is a set of mappings from attribute names to attribute values.

It can also be represented as:

{classId: 10004, facId: 1014, room: 'D241', semester: 'Fall', year: 2019, courseId: 2021}

or

{10004, 2021, 'Fall', 2019, 1014, 'D241'} if the attribute names are assumed in be in the right order.

or using a more computer science-style notation.

(10004, 2021, 'Fall', 2019, 1014, 'D241')

Identify some examples of the database terms in the class table above as much as possible.

* Some important properties of a relation:
  + There is no duplicate tuple.
    1. Because a relation is a set.
    2. Consequence: the relational model does not support 'object identity' directly.
  + The relational model is known to be '*value-oriented*':
    1. A row is a set of attribute values.
    2. Two rows with the same values in all attributes are the same row.
    3. Cannot store two duplicate rows in a table.
  + The terms tables and relations are not exactly the same. "Table" is a more generic term.
* Tuples within a relation are unordered.
  + Changing the order of displaying the tuples does not change (the meaning of) the relation.
* Attributes within a relation schema are unordered.
  + Changing the order of the attributes within a relation schema does not change the information stored in the relation.

*Example:*

+---------+----------+----------+------+-------+------+  
| classId | courseId | semester | year | facId | room |  
+---------+----------+----------+------+-------+------+  
|   10000 |     2000 | Fall     | 2019 |  1011 | D241 |  
|   10001 |     2001 | Fall     | 2019 |  1011 | D242 |  
|   10002 |     2002 | Fall     | 2019 |  1012 | D136 |  
|   10003 |     2020 | Fall     | 2019 |  1014 | D241 |  
|   10004 |     2021 | Fall     | 2019 |  1014 | D241 |  
|   10005 |     2040 | Fall     | 2019 |  1015 | D237 |  
|   10006 |     2041 | Fall     | 2019 |  1019 | D217 |  
|   10007 |     2060 | Fall     | 2019 |  1020 | B101 |  
|   10008 |     2080 | Fall     | 2019 |  1018 | D241 |  
|   11000 |     2000 | Spring   | 2020 |  1011 | D241 |  
|   11001 |     2001 | Spring   | 2020 |  1013 | D242 |  
|   11002 |     2002 | Spring   | 2020 |  1013 | D136 |  
|   11003 |     2020 | Spring   | 2020 |  1016 | D217 |  
|   11004 |     2061 | Spring   | 2020 |  1018 | B101 |  
+---------+----------+----------+------+-------+------+

and

+---------+------+----------+----------+------+-------+  
| classId | year | semester | courseId | room | facId |  
+---------+------+----------+----------+------+-------+  
|   11004 | 2020 | Spring   |     2061 | B101 |  1018 |  
|   10007 | 2019 | Fall     |     2060 | B101 |  1020 |  
|   10002 | 2019 | Fall     |     2002 | D136 |  1012 |  
|   11002 | 2020 | Spring   |     2002 | D136 |  1013 |  
|   11003 | 2020 | Spring   |     2020 | D217 |  1016 |  
|   10006 | 2019 | Fall     |     2041 | D217 |  1019 |  
|   10005 | 2019 | Fall     |     2040 | D237 |  1015 |  
|   10000 | 2019 | Fall     |     2000 | D241 |  1011 |  
|   11000 | 2020 | Spring   |     2000 | D241 |  1011 |  
|   10003 | 2019 | Fall     |     2020 | D241 |  1014 |  
|   10004 | 2019 | Fall     |     2021 | D241 |  1014 |  
|   10008 | 2019 | Fall     |     2080 | D241 |  1018 |  
|   10001 | 2019 | Fall     |     2001 | D242 |  1011 |  
|   11001 | 2020 | Spring   |     2001 | D242 |  1013 |  
+---------+------+----------+----------+------+-------+  
  
  
store the same information.

The second table can be obtained in SQL by:

SELECT classId, year, semester, courseId, room, facId  
FROM class  
ORDER BY room, facId;

**2. Keys**

* A*set* of attributes K is a *candidate key* (CK) of *a relation R* if it *minimally* identifies a tuple *at any time*:
  1. *Uniqueness*: No two tuples of R have the same value of K.
  2. *Minimality*: No *proper* subset of K has the uniqueness property.
* A candidate key is a property of the semantic (meaning) of a relation.
* In other words, being a candidate key is the result of the requirements of an application.
* A relation always has at least one candidate keys. Why?
  1. Because a relation instance r is a set of rows, no two rows will have the exact same values.
  2. Thus, the relation schema R by itself satisfies the uniqueness property.
  3. R or one of its proper subsets will satisfy the minimal property as extraneous attributes are removed.
* A set of attributes may be a candidate key of a relation R but not a candidate key of another relation S.
* For *any* relation instance, the candidate key of a tuple must have an unique value.
* A *primary* key is a selected candidate key for a relation in the DBMS. It is used for practical purpose (of how the relation is stored physically) and does not have a special meaning in the theory of the relational model.
* Some criteria for selecting a primary key:
  1. Stable and relative immutable.
  2. Simple: contains one attribute
  3. Concise: short storage space and faster comparison.
* An*alternate/secondary* key is a candidate key that is not the primary key.
* Questions people may ask:
  1. What is the key of the database toyu? Wrong question.
  2. What is the primary key of the database toyu? more precise but still wrong question.
  3. What is the key of the school table of the toyu database? good question but a bit ambiguous. Answer: schoolCode.
  4. What is the primary key of the school table of the toyu database? schoolCode.
  5. What are the candidate keys of the school table of the toyu database? [1] schoolCode, [2] schoolName.
  6. What are the secondary/alternative keys of the school table of the toyu database? schoolName.

***Example:***

Consider:  
MariaDB [toyu]> SELECT \* FROM faculty;  
+-------+----------+----------+----------+---------------------+  
| facId | fname    | lname    | deptCode | rank                |  
+-------+----------+----------+----------+---------------------+  
|  1011 | Paul     | Smith    | CSCI     | Professor           |  
|  1012 | Mary     | Tran     | CSCI     | Associate Professor |  
|  1013 | David    | Love     | CSCI     | NULL                |  
|  1014 | Sharon   | Mannes   | CSCI     | Assistant Professor |  
|  1015 | Daniel   | Kim      | CINF     | Professor           |  
|  1016 | Andrew   | Byre     | CINF     | Associate Professor |  
|  1017 | Deborah  | Gump     | ITEC     | Professor           |  
|  1018 | Art      | Allister | ARTS     | Assistant Professor |  
|  1019 | Benjamin | Yu       | ITEC     | Lecturer            |  
|  1020 | Katrina  | Bajaj    | ENGL     | Lecturer            |  
|  1021 | Jorginlo | Neymar   | ACCT     | Assistant Professor |  
+-------+----------+----------+----------+---------------------+  
11 rows in set (0.001 sec)  
  
MariaDB [toyu]> SELECT \* FROM department;  
+----------+------------------------------+------------+----------+  
| deptCode | deptName                     | schoolCode | numStaff |  
+----------+------------------------------+------------+----------+  
| ACCT     | Accounting                   | BUS        |       10 |  
| ARTS     | Arts                         | HSH        |        5 |  
| CINF     | Computer Information Systems | CSE        |        5 |  
| CSCI     | Computer Science             | CSE        |       12 |  
| ENGL     | English                      | HSH        |       12 |  
| ITEC     | Information Technology       | CSE        |        4 |  
| MATH     | Mathematics                  | CSE        |        7 |  
+----------+------------------------------+------------+----------+  
7 rows in set (0.001 sec)

* deptCode is a candidate key of the relation department. Assumptions made:
  1. Each tuple in the department relation represents an unique department.
  2. Each department has an unique code. No two departments can have the same code.
* deptName is also a candidate key of the relation department. Assumptions made:
  1. Each tuple in the department relation represents an unique department.
  2. Each department has an unique name. No two departments can have the same name.
* deptCode is selected as the primary key of the relation department since it is concise and more stable.
* deptCode is not a candidate key of the relation faculty. Assumptions made:
  1. Each tuple in the faculty relation represents a faculty member.
  2. Each faculty serves only one department.
  3. A department may have many faculty members.

Consider the tables student and enroll:

* stuid is a candidate key of the relation student. Assumptions made:
  1. Each tuple in the student relation represents an unique student.
  2. Each student has an unique student id, or no two students can have the same stuid.
* stuid is not a candidate key of the relation enroll. Assumptions made:
  1. Each tuple in the Enroll relation represents the enrollment of a student in a class, ending up with a grade.
  2. A student can take many classes.
  3. A class can have many students.
* Note again that a candidate key is a property of a relation.

**2.1 More about keys:**

* A key is *simple* if it has only one attribute.
* A key is a *composite* key if it has more than one attributes.
* A key is a *compound* key if it is a composite and each attribute in the key is a foreign key.
* Every relation has at least one candidate key.
* A *foreign key* of a relation is a set of attributes that is a candidate key in another relation. The other relation is sometimes called the parent (or referenced) table of the foreign key.
* A foreign key may or may not have null value. It depends on the problem requirements.
* For uses in normalization theory:
  + An attribute that appears in one or more candidate keys is a *prime attribute* (or key attribute).
  + An attribute that does not appears in any candidate key is a *non-prime* (non-key) attribute.
  + A *superkey* of a relation is a set of attributes that uniquely identify a row (uniqueness). It may not be minimal. (A Superkey may have extraneous attributes not needed for unique identification.)

Consider the example above. The attribute deptCode is a foreign key in the relation faculty, referencing department(deptCode)

* A deptCode in the department relation must be referring to a department in the database, identified by deptCode in the relation department (referential integrity).
* In relational DB, rows (data) from different tables are linked together using *foreign keys*.
* A foreign key of a child table links to a primary key of the parent table.

Likewise, stuid is a foreign key in the relation Enroll: stuid references student(stuid)

Note that the student table has three foreign keys.

***Example***: In the department table in toyu:

Department(deptCode, deptName, schoolCode, numStaff)

Candidate keys:

1. deptCode (i.e. {deptCode})
2. deptName

Primary key: deptCode

Alternate key:

1. deptName

Prime attributes:

1. deptCode
2. deptName

Non-prime attributes:

1. schoolCode
2. numStaff

Superkeys:

1. deptCode
2. deptName
3. deptCode, deptName (i.e. {deptCode, deptName})
4. deptCode, schoolCode
5. deptCode, numStaff
6. deptCode, deptName, SchoolCode
7. deptCode, deptName, numStaff
8. deptCode, SchoolCode, numStaff
9. deptCode, deptName, SchoolCode, numStaff
10. deptName, SchoolCode
11. deptName, numStaff
12. deptName, SchoolCode, numStaff

***Example:***

Study the list of all foreign keys in [toyu](https://dcm.uhcl.edu/yue/courses/joinDB/Spring2025/notes/toyu/toyu.html):

* The INNER JOIN operation can be used to join tables through foreign keys.

***Example:***

-- INNER JOIN  
SELECT \* FROM faculty;  
SELECT \* FROM department;  
SELECT s.fname, s.lname, s.advisor, f.`rank`  
FROM student AS s INNER JOIN faculty AS f  
   ON (s.advisor = f.facId);  
    
Result:

MariaDB [toyu]> SELECT \* FROM faculty;  
+-------+----------+----------+----------+---------------------+  
| facId | fname    | lname    | deptCode | rank                |  
+-------+----------+----------+----------+---------------------+  
|  1011 | Paul     | Smith    | CSCI     | Professor           |  
|  1012 | Mary     | Tran     | CSCI     | Associate Professor |  
|  1013 | David    | Love     | CSCI     | NULL                |  
|  1014 | Sharon   | Mannes   | CSCI     | Assistant Professor |  
|  1015 | Daniel   | Kim      | CINF     | Professor           |  
|  1016 | Andrew   | Byre     | CINF     | Associate Professor |  
|  1017 | Deborah  | Gump     | ITEC     | Professor           |  
|  1018 | Art      | Allister | ARTS     | Assistant Professor |  
|  1019 | Benjamin | Yu       | ITEC     | Lecturer            |  
|  1020 | Katrina  | Bajaj    | ENGL     | Lecturer            |  
|  1021 | Jorginlo | Neymar   | ACCT     | Assistant Professor |  
+-------+----------+----------+----------+---------------------+  
11 rows in set (0.000 sec)  
  
MariaDB [toyu]> SELECT \* FROM department;  
+----------+------------------------------+------------+----------+  
| deptCode | deptName                     | schoolCode | numStaff |  
+----------+------------------------------+------------+----------+  
| ACCT     | Accounting                   | BUS        |       10 |  
| ARTS     | Arts                         | HSH        |        5 |  
| CINF     | Computer Information Systems | CSE        |        5 |  
| CSCI     | Computer Science             | CSE        |       12 |  
| ENGL     | English                      | HSH        |       12 |  
| ITEC     | Information Technology       | CSE        |        4 |  
| MATH     | Mathematics                  | CSE        |        7 |  
+----------+------------------------------+------------+----------+  
7 rows in set (0.000 sec)  
  
MariaDB [toyu]> SELECT s.fname, s.lname, s.advisor, f.`rank`  
    -> FROM student AS s INNER JOIN faculty AS f  
    -> ON (s.advisor = f.facId);  
+---------+----------+---------+---------------------+  
| fname   | lname    | advisor | rank                |  
+---------+----------+---------+---------------------+  
| Tony    | Hawk     |    1011 | Professor           |  
| Mary    | Hawk     |    1011 | Professor           |  
| David   | Hawk     |    1012 | Associate Professor |  
| Larry   | Johnson  |    1017 | Professor           |  
| Linda   | Johnson  |    1015 | Professor           |  
| Lillian | Johnson  |    1016 | Associate Professor |  
| Linda   | King     |    1018 | Assistant Professor |  
| Cathy   | Johanson |    1018 | Assistant Professor |  
+---------+----------+---------+---------------------+  
8 rows in set (0.000 sec)

***Classroom Exercise:***

(1) Give a realistic example of a relation with two candidate keys. State the assumptions you have made.

(2) A relation R has an *arity* of 4. What are the possible minimum and maximum number of superkeys of R? What are the possible minimum and maximum number of candidate keys of R?

What about the general case?