Cal Poly **CPE/CSC 365:** Introduction to Database Systems Alexander Dekhtyar

Relational Database Model

Relational Model

- One single data modeling "tool": relation, or a 2D table;
- A relational database is a collection of relations;
- High degree of data independence
- Association between information elements (constraints)

More Formally

Relation: a two-dimensional table of columns and rows.

Attribute, Field: name of a *column* in the relation.

- take values from predefined *domains*

Record, tuple: a single row in the relation: a collection of *attribute values.*

Schema: the name of a *relation* plus the set of *attributes* of the relation (and their domains).

- E.g. Book(ISBN string, Title string, Author string, year integer).

Relation instance: a set of *tuples* for a given *relation.*

- changes with time (as stuff gets added, deleted, modified)
- schema usually does not change (althought it might in some cases)

Cardinality: number of tuples in a relation

Degree: number of attributes in a relation

Constraints

Superkey a collection of attributes in a relation that *uniquely identifies each tuple in it.*

Candidate key a *superkey* that has no *superkey subsets*

Primary key one *candidate key* per relation, designated to be the main way of maintaining tuple uniqueness.

Key constraint : each relation must have a primary key.

Foreign key a *primary key* of one relation, included in the attributes of another relation (usually for the purpose of linking two components of the database together).

Referential integrity constraint each collection of values of a *foreign key* in a relation must appear as a *primary key* in the referenced relation.

Null value : a "no value" value for a relational attribute. Lack of value, or value not yet available.

not null constraint : a statement that a specific attribute is not allowed to have null values. (e.g., primary key attributes).

Examples

Example 1. A university registrar wants to organize a list of courses offered at a university. The information available about each course is its prefix (e.g., 'CSC'), its number (e.g., 365), its title (e.g., `'Introduction to Databases'`), its catalog description, and the department responsible for scheduling the course. The registrar creates the following relational table to represent the courses:

```
Courses(Prefix String,  CourseNumber Integer,  Title String,
          Catalog  String, Department String)
```

A portion of this table would look as follows:

Prefix	CourseNumber	Title	Catalog	Department
CSC	365	Introduction to Databases	Basic principles of database management...	CSSE
CSC	357	Systems Programming	C programming language from a system...	CSSE
CHEM	212	Introduction to Organic Chemistry	Structure, isomerism, nomenclature, fundamental...	Chemistry
STAT	331	Statistical Computing with R	Importing, managing, and cleaning data from ...	Statistics
STAT	365	Statistical Communication	Written communication of statistical ideas...	Statistics

Determining the keys:

- The pair of attributes `(Prefix, CourseNumber)` *together* is a candidate key for the `Courses` table. The requirement that the `Prefix-CourseNumber` pair is unique for each course comes from *outside* - this is university catalog rule established before the `Courses` table is introduced.
- The `Prefix` attribute alone is not enough to uniquely identify every row in the table (e.g., there are two that have the `CSC` prefix).

- The `CourseNumber` attribute alone is also not enough: there are two rows describing different courses whose number is `365`.
- The second candidate key in the table could be `Title`. This means that the Registrar's office requires all courses to have a unique title.
- Between the `Title` and the `(Prefix, CourseNumber)`, the registrar selects `(Prefix, CourseNumber)` as the **primary key**.

Example 2. After examining the `Courses` table, the registrar arrives to the conclusion that they want another attribute in the table that merges (concatenates) the course prefix and the course number[1] As a result, the new `Courses` table looks as follows:

```
Courses(CourseId String,  Prefix String,  CourseNo Integer,
        Title String,     Catalog  String, Department String)
```

A portion of this table looks as follows:

CourseId	Prefix	CourseNo	Title	Catalog	Department
CSC 365	CSC	365	Introduction to Databases	Basic principles of database management. . .	CSSE
CSC 357	CSC	357	Systems Programming	C programming language from a system. . .	CSSE
CHEM 212	CHEM	212	Introduction to Organic Chemistry	Structure, isomerism, nomenclature. . .	Chemistry
STAT 331	STAT	331	Statistical Computing with R	Importing, managing, and cleaning data . . .	Statistics
STAT 365	STAT	365	Statistical Communication	Written communication of statistical ideas. . .	Statistics

We now have the following candidate keys in this table:

- `(Prefix, CourseNo)` - our old primary key
- `Title` - per our assumption that the Registrar's office will not let two courses with exactly the same name to co-exist.
- `CourseId` - as this is literally concatenation of the two values that forms a candidate key.

Given these three candidate keys, the registrar now selects `CourseId` as the **primary key** for this table.

Example 3. In addition to the courses, the registrar wants to organize a list of the departments. The registrar needs the following information about each department: its full name, the college it is in, it's location on campus (building number and room number). The first version of the `Departments` table looks as follows:

```
Departments(Name String, College String, Building Integer, Room String)
```

(note: rooms may have values like `'245a'` so, they are strings.)

A portion of the table looks as follows:

Name	College	Building	Room
Computer Science and Software Engineering	CENG	14	245
Statistics	COSAM	25	107D
Chemistry and Biochemistry	COSAM	180	206
Physics	COSAM	180	207

Keys:

[1] This is because there is a lot of code being written that accesses the table and has to concatenate these two attributes every single time.

- The `Name` attribute is a candidate key (no two departments can have the same name).

- Assuming no two departments share the same office - a reasonable assumption at Cal Poly, the pair `(Building, Room)` is also a candidate key.

- These are the only candidate keys in the table (`College` by itself is NOT a candidate key, and any other combination of attributes either contains a key, or is not unique - e.g., `(College, Building)` is not a candidate key, because multiple departments can have offices in the same building, and `(College, Room)` is not unique because multiple departments can have offices with the same room number but in different buildings).

- Between the two candidate keys, the registrar selects `Name` as the **primary key**.