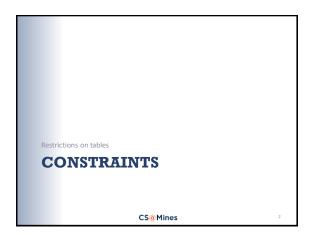
### CSCI 403 Database Management

8

Constraints, keys, indexes

CS@Mines



### **Types of Constraints**

- Implicit (model-based)
- Explicit (schema-based)
- Application-based

CS@Mines

### **Implicit Constraints**

Implied by model of world represented by DB

- E.g., a column named "instructor" should contain person names
- Not (usually) enforced

CS@Mines

### **Application-Based Constraints**

- Not defined in DB
- Enforced by applications using DB
- AKA "business rules"
- Often too complex to reasonably implement on database side
- Example:

"no employee can have a hire date that is not on the first of some month"

CS@Mines

### **Explicit Constraints**

- Domain constraints
  - Values in a column must match the type domain
- Primary key constraints (next section)
- Foreign key constraints (next section)

CS@Mines

### **KEYS**CS@Mines

### Keys (Theory)

Keys have a very specific definition in relational database theory.

We'll revisit this topic when we get to the theory portion of the class.

For now, we'll discuss the practical applications in an SQL database.

CS@Mines

### **Primary Key**

- Each table can have zero or one primary key
- Primary key applies to a subset of the columns
  - i.e., one or more identified columns form the key
  - Can be all columns!
- Does three things:
  - Constrains data in rows to be unique for the combination of columns in the primary key (example next page)
  - Constraints data in primary key columns to be not null
  - Creates an *index* on the combination of columns

CS@Mines

### Primary Key Example

Consider a person table:

ssn	first	last	birthdate	
123-45-6789	Wood	Carpenter	4/17/1975	
111-22-3333	Opal	Miner	12/1/1982	
454-45-4545	Jane	Doe	6/2/1995	

SSN would make a good primary key. What about (first, last)?

Or (first, last, birthdate)?

CS@Mines

### **Uniqueness Constraint**

If we make (first, last) a primary key: cannot have more than one Jane Doe.

SSN is probably okay, unless someone has a fake identity...

If SSN is primary key, we *cannot* insert a duplicate SSN into the table!

CS@Mines

### SQL Example

```
CREATE TABLE person (
    ssn text PRIMARY KEY,
    first text,
    last text,
    birthdate date);

INSERT INTO person VALUES ('123-45-6789',
    'Wood', 'Carpenter', '4/17/1975'); OK

INSERT INTO person VALUES ('123-45-6789',
    'Evil', 'Impostor', '6/22/1963'); ERROR!
```

# Indexes Short explanation: Indexes speed searching by indexed columns E.g., SELECT \* FROM person WHERE ssn = '123-45-6789'; Speeds up most if all indexed columns are searched Can get some speed up by searching first listed column, first two, etc. Longer explanation: Without index, must do linear search − look at every row With index, search through an optimized data structure called a Btree − look at only a fraction of rows We'll study Btrees later in the semester For the curious: Like a binary search tree but with >2 children per node Performance is log₀(n), where b is typically large (e.g. b=100)

```
Creating Primary Keys

CREATE TABLE person (
    ssn text PRIMARY KEY,
    first text,
    last text,
    birthdate date);

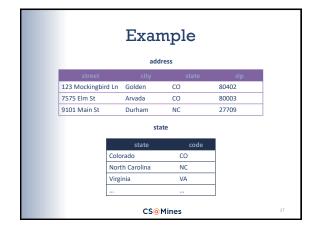
Vd person

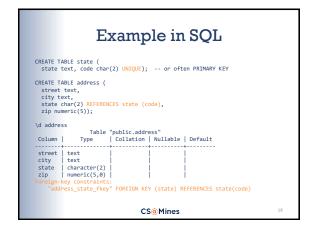
    Table "public.person"
Column | Type | Collation | Nullable | Default

    ssn | text | | not null |
    first | text | | last | text |
    last | text | | last | last | text |
    last | text | | last | last | text |
    lindexes:
    "person_pkey" PRIMARY KEY, btree (ssn)
```

### Multiple Column Primary Key CREATE TABLE person2 ( first text, last text, birthdate date. PRIMARY KEY (first, last)); \d person2 Table "public.person2" Column | Type | Collation | Nullable | Default first | text | | not null last text birthdate | date | Indexes: 'person2\_pkey" PRIMARY KEY, btree (first, last) CS@Mines







## FK Constraint Example INSERT INTO address VALUES ('129 West 81st Street', 'New York', 'NY', 10024) OK INSERT INTO address VALUES ('221B Baker St.', London, NULL, NULL) OK INSERT INTO address VALUES ('4222 Clinton Way', 'Los Angeles', 'XX', 90204) ERROR CS@Mines

```
Another FK Example

INSERT INTO address
VALUES ('129 West 81st Street', 'New York', 'NY', 10024) OK

DELETE FROM state WHERE code = 'NY'; ERROR
(If we had no addresses in NY, would be fine.)
```

```
Multiple Column Foreign Key

CREATE TABLE foo (
   x integer,
   y date,
   PRIMARY KEY (x, y)); -- or UNIQUE

CREATE TABLE bar (
   zz text,
   xx integer,
   yy date,
   FOREIGN KEY (xx, yy) REFERENCES foo(x, y));

CS@Mines
```

```
MISCELLANEOUS
CONSTRAINTS
CS@Mines
```

```
Multiple Column Uniqueness

CREATE TABLE bar (
    a integer,
    b text,
    UNIQUE (b, a));

d bar

Table "public.bar"

Column | Type | Collation | Nullable | Default

a | integer | | | |
    b | text | | |

Indexes:

"bar_b_a_key" UNIQUE CONSTRAINT, btree (b, a)
```

### Multiple Column Uniqueness

```
CREATE TABLE bar (
a integer,
b text,
UNIQUE (b, a));

INSERT INTO bar VALUES (1, 'apple'); OK
INSERT INTO bar VALUES (2, 'pear'); OK
INSERT INTO bar VALUES (1, 'banana'); OK
INSERT INTO bar VALUES (3, 'pear'); OK
INSERT INTO bar VALUES (1, 'apple'); ERROR

CS@Mines
```

### **Notes**

- Constraints can be combined, e.g.
   CREATE TABLE foo (x integer UNIQUE NOT NULL ...);
- Foreign key constraints also known as referential integrity constraints
- Foreign key relationships often mirror a common/likely choice of join

CS@Mines

### **Up Next**

Next lecture:

Miscellaneous DDL: default column values, sequences, ALTER TABLE, views, indexes, DROP

CS@Mines