Types of Constraints
- Implicit (model-based)
- Explicit (schema-based)
- Application-based

Implicit Constraints
Implied by model of world represented by DB
- E.g., a column named “instructor” should contain person names
- Not (usually) enforced

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”

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

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)
  - Constrains data in primary key columns to be not null
  - Creates an index on the combination of columns

Primary Key Example

Consider a *person* table:

<table>
<thead>
<tr>
<th>ssn</th>
<th>first</th>
<th>last</th>
<th>birthdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>Wood</td>
<td>Carpenter</td>
<td>4/17/1975</td>
</tr>
<tr>
<td>111-22-3333</td>
<td>Opal</td>
<td>Miner</td>
<td>12/1/1982</td>
</tr>
<tr>
<td>454-45-4545</td>
<td>Jane</td>
<td>Doe</td>
<td>6/2/1995</td>
</tr>
</tbody>
</table>

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

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!

SQL Example

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

```sql
```

```sql
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 \( \log_b(n) \), 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);

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Collation</th>
<th>Nullable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>text</td>
<td></td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td>text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>last</td>
<td>text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>birthdate</td>
<td>date</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexes:
- "person_pkey" PRIMARY KEY, btree (ssn)

Multiple Column Primary Key

CREATE TABLE person2 (first text, last text, birthdate date, PRIMARY KEY (first, last));

FOREIGN KEY constraints:
- "person2_pkey" PRIMARY KEY, btree (first, last)

Foreign Key

A foreign key defines a constraint between two tables:
- An FK applies to a subset of columns of one table
- The FK references a subset of columns of second table
- FK constraint:
  - Values in FK column(s) must either:
    - Exist in columns in referenced table OR
    - Be NULL

Example

<table>
<thead>
<tr>
<th>street</th>
<th>city</th>
<th>state</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 Mockingbird Ln</td>
<td>Golden</td>
<td>CO</td>
<td>80402</td>
</tr>
<tr>
<td>7575 Elm St</td>
<td>Arvada</td>
<td>CO</td>
<td>80003</td>
</tr>
<tr>
<td>9101 Main St</td>
<td>Durham</td>
<td>NC</td>
<td>27709</td>
</tr>
</tbody>
</table>

Example in SQL

CREATE TABLE state (state text, code char(2) UNIQUE; -- or often PRIMARY KEY
CREATE TABLE address (street text, city text, state char(2) REFERENCES state (code), zip numeric(5));

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Collation</th>
<th>Nullable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>street</td>
<td>text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>city</td>
<td>text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>character(2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zip</td>
<td>numeric(5)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FOREIGN KEY constraints:
- "address_state_fkey" FOREIGN KEY (state) REFERENCES state(code)
**FK Constraint Example**

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

```sql
INSERT INTO address
VALUES ('221B Baker St.', London, NULL, NULL) OK
```

```sql
INSERT INTO address
VALUES ('4222 Clinton Way', 'Los Angeles', 'XX', 90204) ERROR
```

**Another FK Example**

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

```sql
DELETE FROM state WHERE code = 'NY'; ERROR
```

(If we had no addresses in NY, would be fine.)

**Multiple Column Foreign Key**

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

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

**MISCELLANEOUS CONSTRAINTS**

- NOT NULL – disallows NULL values in column
- UNIQUE – imposes uniqueness on a column or set of columns (and creates index)
- CHECK constraints – requires meeting a Boolean condition
- Examples:
  ```sql
  CREATE TABLE foo (    
  id integer CHECK (id > 0),  
  name text NOT NULL,  
  x integer UNIQUE );
  ```

```sql
Table "public.foo"  
Column | Type   | Collation | Nullable | Default  
--------+--------+-----------+----------+----------
    id  | integer |           | not null |          
     x  | integer |           |          |          
 Islands  | "iso_key" UNIQUE CONSTRAINT, btree (x)  
Check constraints:  
"foo_id_check" CHECK (id > 0)
```

**Multiple Column Uniqueness**

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

```sql
Table "public.bar"  
Column | Type   | Collation | Nullable | Default  
--------+--------+-----------+----------+----------
    a   | integer |           |          |          
    b   | text   |           |          |          
Indexes:  
"bar_b_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

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

Up Next

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