CS 61: Database Systems

Advanced data modeling

Adapted from Silberschatz, Korth, and Sundarshan unless otherwise noted

Agenda

1. Choosing Primary Keys
    - Desirable properties
    - When to use composite keys
    - When to use surrogate keys

2. Time-variant data

3. Inheritance

Primary Keys uniquely identify rows; sometimes there are “natural keys”

Primary keys

Primary keys:

- Single attribute or a combination of attributes (called a composite primary key)
- Uniquely identifies each or row in relation
- Function is to guarantee entity integrity, not to “describe” entity (if describing entity, use a non-key attribute)
- Works with foreign keys to implement relationships between entities

Natural key:

- Real-world identifier than can uniquely identify real-world objects
- Sometimes, but not always present (e.g., CS61 natural key for this class)
- Familiar to end users and forms part of their day-to-day business vocabulary
- Can sometimes be used as the primary key of the entity being modeled

Surrogate key:

- System generated key
- Often generated with auto_increment

Primary Keys should have several desirable qualities

Primary keys

PK Characteristic	Notes
Unique values	PK must uniquely identify each tuple. Must be able to guarantee unique values, nulls not allowed

Based on Coronel and Morris

Primary Keys should have several desirable qualities

Primary keys

PK Characteristic	Notes
Unique values	PK must uniquely identify each tuple. Must be able to guarantee unique values, nulls not allowed
Nonintelligent	PK should not have embedded semantic meaning (if has semantic meaning, use as attribute, not key!). Example: StudentID f12345a better than Smith, Mary B.

Based on Coronel and Morris

Primary Keys should have several desirable qualities

Primary keys

PK Characteristic	Notes
Unique values	PK must uniquely identify each tuple. Must be able to guarantee unique values, nulls not allowed
Nonintelligent	PK should not have embedded semantic meaning (if has semantic meaning, use as attribute, not key!). Example: StudentID f12345a better than Smith, Mary B.
Does not change over time	Attributes with semantic meaning sometimes change over time (names are not good PKs)

Based on Coronel and Morris

Primary Keys should have several desirable qualities

Primary keys

PK Characteristic	Notes
Unique values	PK must uniquely identify each tuple. Must be able to guarantee unique values, nulls not allowed
Nonintelligent	PK should not have embedded semantic meaning (if has semantic meaning, use as attribute, not key!). Example: StudentID f12345a better than Smith, Mary B.
Does not change over time	Attributes with semantic meaning sometimes change over time (names are not good PKs)
Preferably single attribute	PK should minimum number of attributes possible (irreducible). Single attribute desirable, but not required. Single attribute PKs simplify FKs

Based on Coronel and Morris

Primary Keys should have several desirable qualities

Primary keys

PK Characteristic	Notes
Unique values	PK must uniquely identify each tuple. Must be able to guarantee unique values, nulls not allowed
Nonintelligent	PK should not have embedded semantic meaning (if has semantic meaning, use as attribute, not key!). Example: StudentID f12345a better than Smith, Mary B.
Does not change over time	Attributes with semantic meaning sometimes change over time (names are not good PKs)
Preferably single attribute	PK should minimum number of attributes possible (irreducible). Single attribute desirable, but not required. Single attribute PKs simplify FKs
Preferably numeric	Database can assign new tuples a unique value simply by incrementing the last value such as auto_increment (Mongo does this differently)

Based on Coronel and Morris

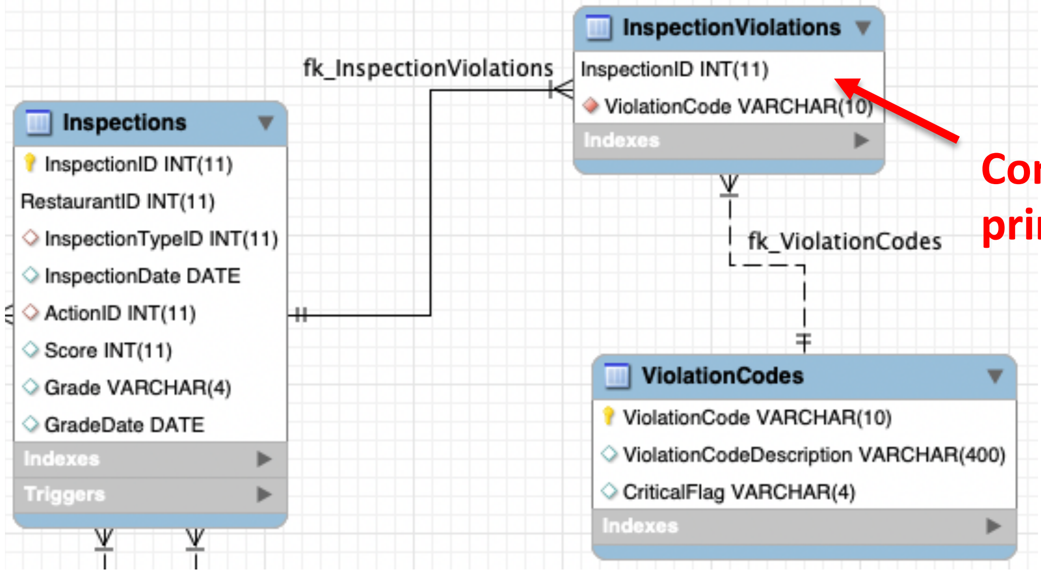
Primary Keys should have several desirable qualities

Primary keys

PK Characteristic	Notes
Unique values	PK must uniquely identify each tuple. Must be able to guarantee unique values, nulls not allowed
Nonintelligent	PK should not have embedded semantic meaning (if has semantic meaning, use as attribute, not key!). Example: StudentID f12345a better than Smith, Mary B.
Does not change over time	Attributes with semantic meaning sometimes change over time (names are not good PKs)
Preferably single attribute	PK should minimum number of attributes possible (irreducible). Single attribute desirable, but not required. Single attribute PKs simplify FKs
Preferably numeric	Database can assign new tuples a unique value simply by incrementing the last value such as auto_increment (Mongo does this differently)
Security compliant	Do not use attributes that have security risks such as social security numbers!

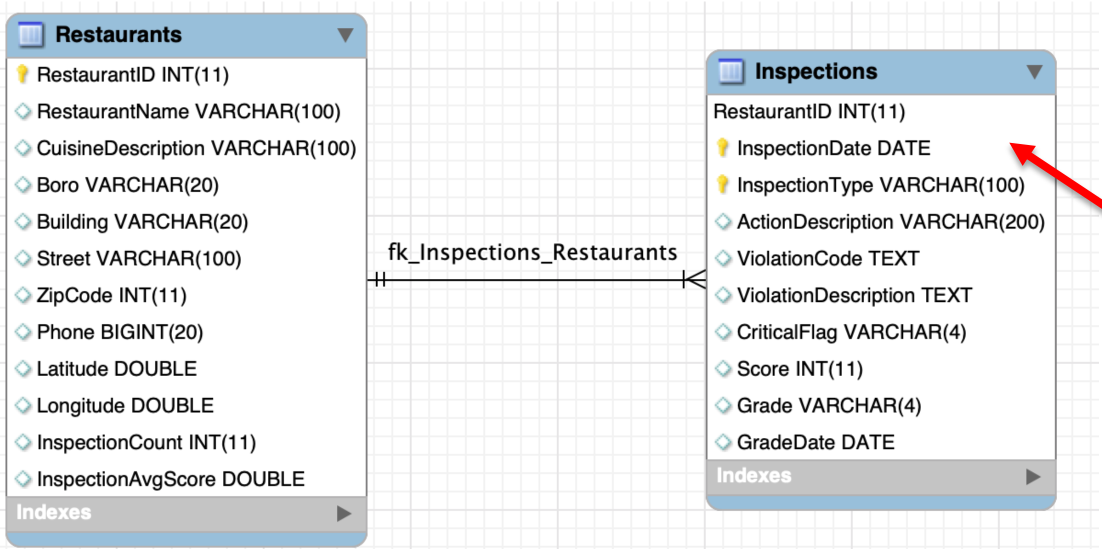
Based on Coronel and Morris

There are two common reasons to use a composite primary key vs. a single attribute



Composite primary key

1. **In a joining table for an M:N relationship**



2. **In an identifying relationship**

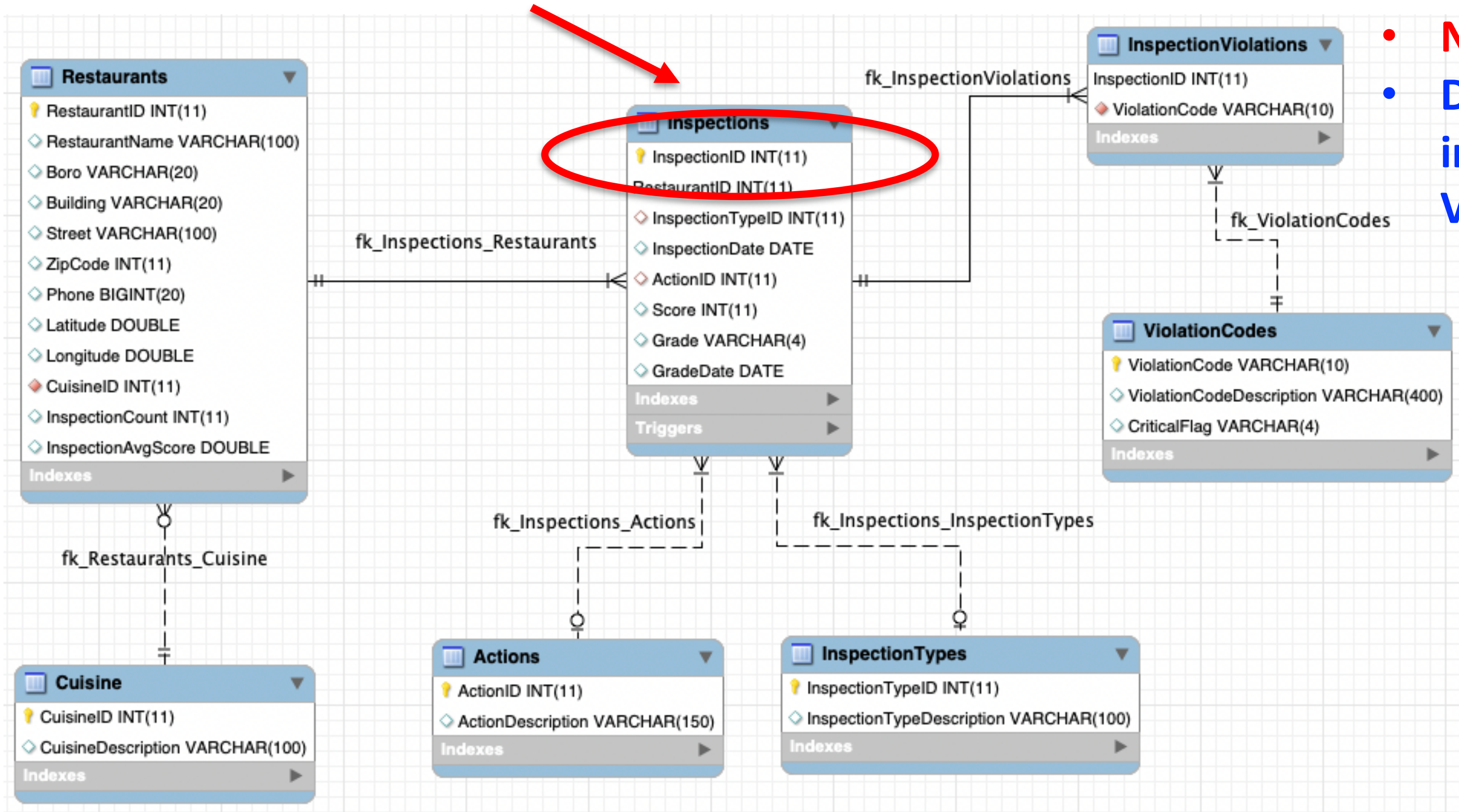
PK: RestaurantID, InspectionDate, InspectionType

Strong relationship has part of parent's PK in its PK

Use surrogate key when there is no natural key, or the natural key is unsuitable

Primary keys

Can uniquely identify Inspections on RestaurantID, InspectionDate, and InspectionType



Could use these attributes together as a composite PK, but this key would:

- **Have semantic meaning**
- **Not numeric**
- **Difficult to use as FK in Inspection Violations table**

Instead can use a numeric *surrogate key* (PK created by database to uniquely identify tuples) when key too long or multiple data types

Problem using auto_increment to generate surrogate keys; consider UUID

Problem with using auto_increment for primary key

- Could be easy to guess
- Consider API route: http://<your domain>/api/employees/5
- Might guess there are IDs 4 and 6 (and beyond)
- Adversary could try plugging in random values to see what they can find

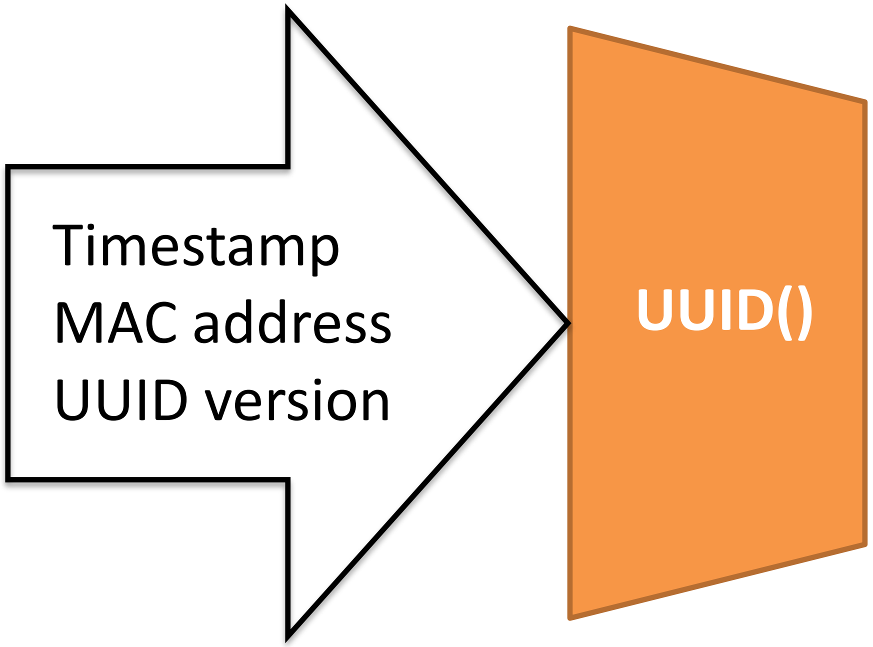
Universally Unique Identifier UUID() function will generate a 128-bit value unique across tables, databases, and servers

- UUID values do not expose the information about your data so they are safer to use in a URL
- Allow you to merge rows from different databases or distribute databases across servers
- Can be generated offline
- Can update parent and subtype in one transaction

Source: https://mysqlserverteam.com/mysql-8-0-uuid-support/

UUIDs are guaranteed to be unique, even if generated on different servers

Universally unique identifier (UUID)



12345678-1234-5678-1234-567812345678

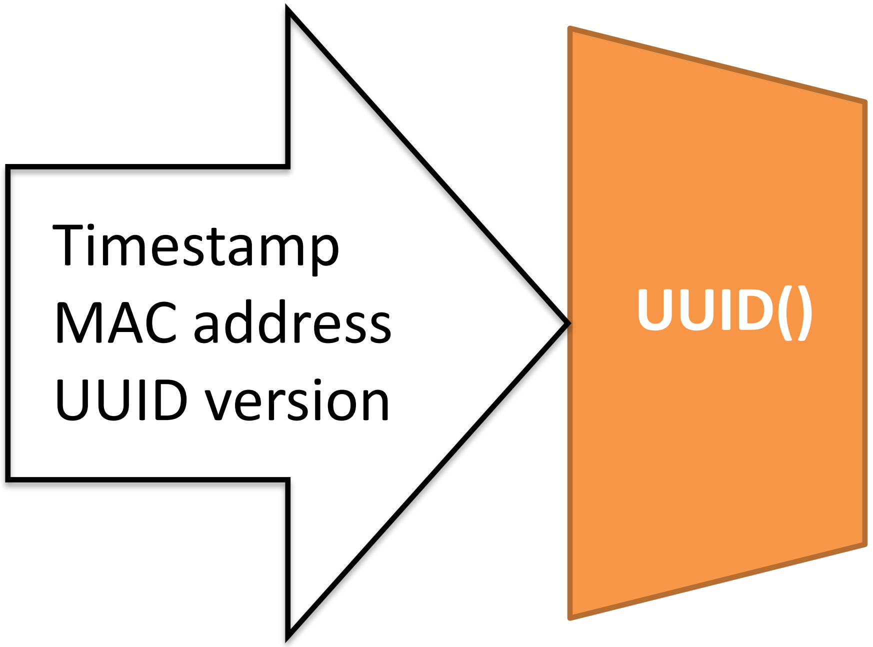
Output is 32-digit string of numbers plus four dashes, 36 characters total
Can use this as a primary key

Time will never be the same again
No two computers will have the same MAC address
Therefore no UUIDs will be the same

Source: https://mysqlserverteam.com/mysql-8-0-uuid-support/

UUIDs have their downsides too: they are big and unordered!

Universally unique identifier (UUID)



12345678-1234-5678-1234-567812345678

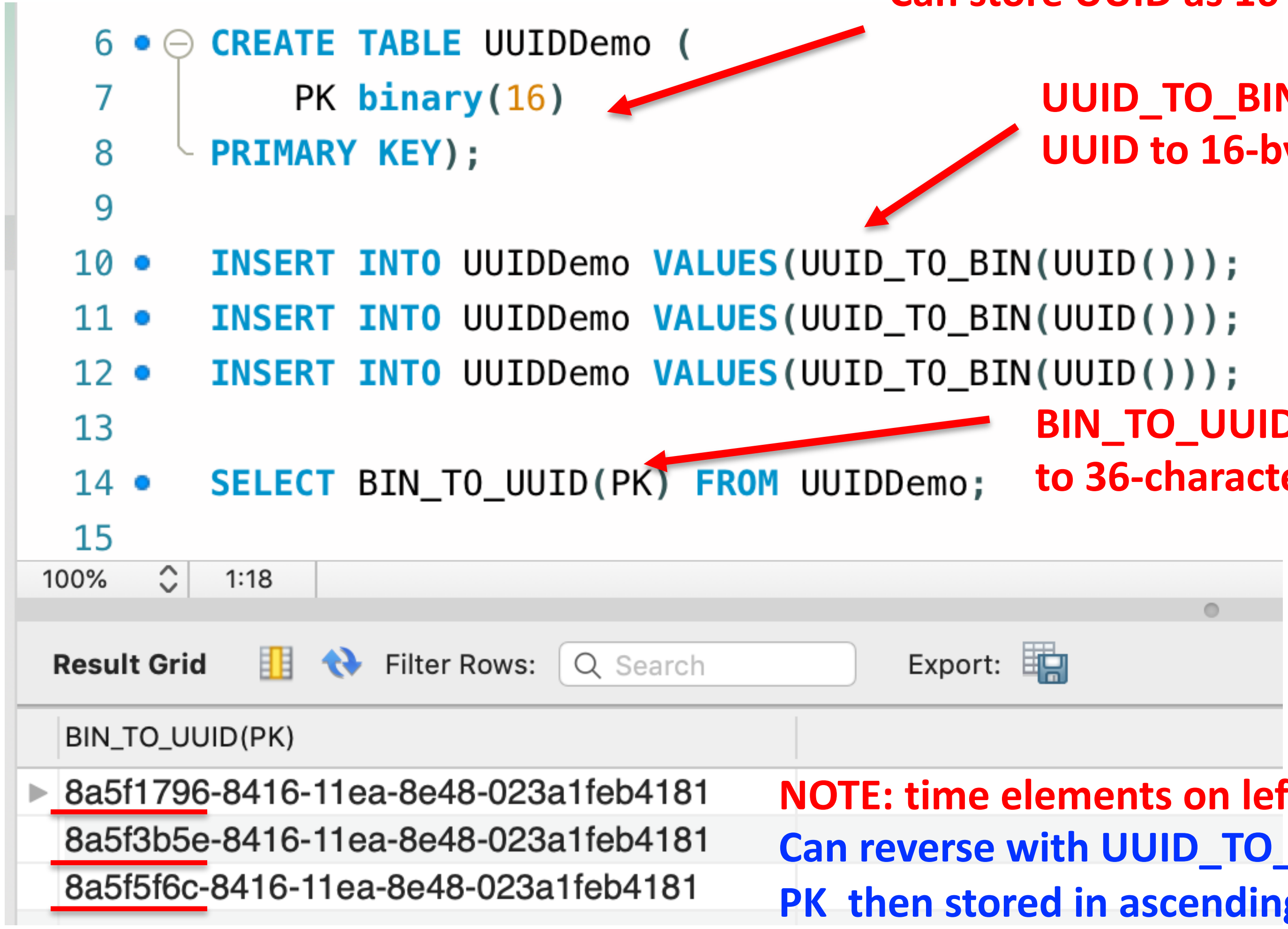
Downsides:

- **Increased storage – 32 characters (plus 4 dashes) vs. Integer at 4 bytes**
- **Harder to debug:**
  SELECT * FROM Table
  WHERE ID = '12345678-1234-5678-1234-567812345678'
- **Performance issues: large key size and not ordered**

Source: https://mysqlserverteam.com/mysql-8-0-uuid-support/

MySQL has commands that solve these problems

Universally unique identifier (UUID)



```
CREATE TABLE UUIDDemo (
      PK binary(16)
PRIMARY KEY);

INSERT INTO UUIDDemo VALUES(UUID_TO_BIN(UUID()));
INSERT INTO UUIDDemo VALUES(UUID_TO_BIN(UUID()));
INSERT INTO UUIDDemo VALUES(UUID_TO_BIN(UUID()));

SELECT BIN_TO_UUID(PK) FROM UUIDDemo;
```

```
BIN_TO_UUID(PK)
8a5f1796-8416-11ea-8e48-023a1feb4181
8a5f3b5e-8416-11ea-8e48-023a1feb4181
8a5f5f6c-8416-11ea-8e48-023a1feb4181
```

Can store UUID as 16 bytes

UUID_TO_BIN converts 36-character UUID to 16-byte binary

BIN_TO_UUID converts binary back to 36-character string

NOTE: time elements on left, change most rapidly

Can reverse with UUID_TO_BIN(UUID(), true)

PK then stored in ascending order

Source: https://mysqlserverteam.com/mysql-8-0-uuid-support/

Agenda

1. Choosing Primary Keys
   - Desirable properties
   - When to use composite keys
   - When to use surrogate keys

2. Time-variant data

3. Inheritance

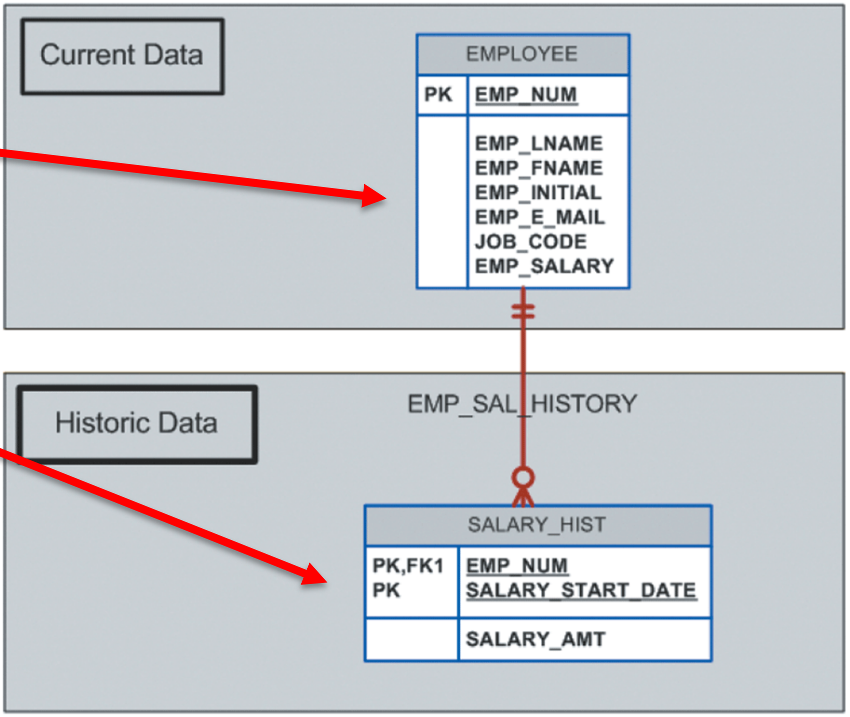
Time variant data's values change over time and requires a 1:M relationship

Time-variant data: data whose values change over time and for which a history of the data changes must be retained

- Requires creating a new entity in a 1:M relationship with the original entity
- New entity contains the new value, date of the change, and any other pertinent attribute

Keep current values in main table

Create second table with effective date as part of the PK



Based on Coronel and Morris

Agenda

1. Choosing Primary Keys
   - Desirable properties
   - When to use composite keys
   - When to use surrogate keys

2. Time-variant data

3. Inheritance

Practice

DartAir airline has employees who are either:

Pilots
- License type
- Rating type
- Medical type

Mechanics
- Type
- Certification

Accountants
- Title
- CPA date

Other

All employees have common attributes:
- First name, last name, middle initial, date of hire

Each type of employee (other than 'other') have additional job-related attributes as shown above

Use MySQL Workbench to create a specialization hierarchy model for the airline
- If an employee is deleted, make sure subtype entries are deleted also!
- Forward engineer your design
- Insert a pilot, mechanic, and 'other' into your database