Foreign Keys and Query Structure

CMPSC 305 – Database Systems

ALLEGHENY COLLEGE

Simple PRIMARY KEY constraint demo

Spot the integrity constraint's influence

```
DROP TABLE IF EXISTS company;
CREATE TABLE company(
        ID INT PRIMARY KEY NOT NULL,
        NAME TEXT NOT NULL,
        AGE INT NOT NULL ,
        ADDRESS CHAR,
        SALARY REAL DEFAULT 50000.00 );
```

PRIMARY KEY

Simple PRIMARY KEY constraint demo

```
/*Good insert command: complete tuple allowed*/
INSERT INTO COMPANY
            VALUES (221, "Sherlock", 25, "10, Rue du fleur",100000);
```

```
sqlite> select * from company;
221|Sherlock|25|10, Rue du fleur|100000.0
```

Key not unique failure

```
/* Try to reinsert same values again.*/
INSERT INTO COMPANY
            VALUES (221, "Sherlock", 25, "10, Rue du fleur",100000);
```

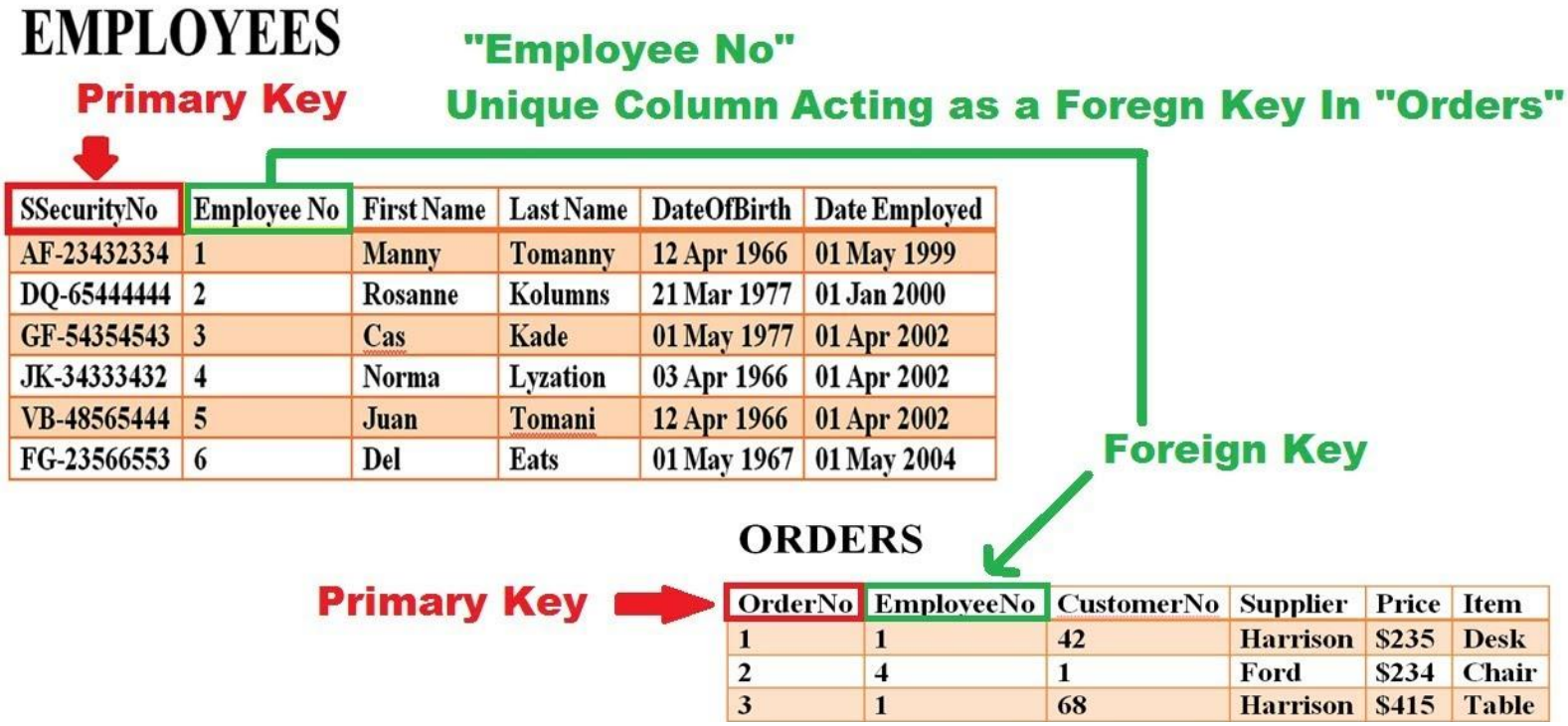
What are the types of keys in databases?

- Primary Keys:
  - Ensures uniqueness in a table.
  - All entries in an attribute-primary never repeat
  - Is a unique identifier (i.e., social security number, telephone number, etc)

What are the types of keys in databases?

- Foreign Keys:
  - A constraint to enforce the relationships between tables.
  - Create a reference to specific information from another table.
  - Foreign key constraints allow checking the referential integrity between tables.
  - Only values that are supposed to appear in a particular table are permitted

Primary and Foreign Keys in Two Tables

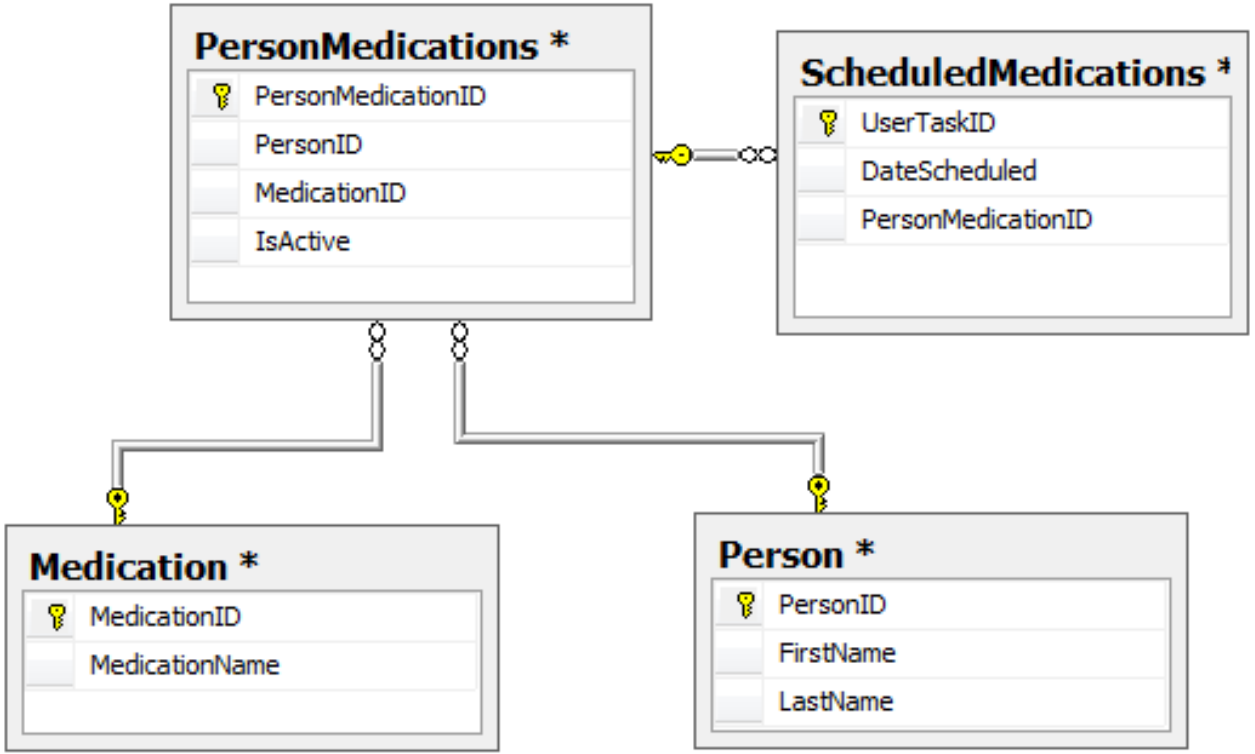


SSecurityNo	Employee No	First Name	Last Name	DateOfBirth	Date Employed
AF-23432334	1	Manny	Tomanny	12 Apr 1966	01 May 1999
DQ-65444444	2	Rosanne	Kolumns	21 Mar 1977	01 Jan 2000
GF-54354543	3	Cas	Kade	01 May 1977	01 Apr 2002
JK-34333432	4	Norma	Lyzation	03 Apr 1966	01 Apr 2002
VB-48565444	5	Juan	Tomani	12 Apr 1966	01 Apr 2002
FG-23566553	6	Del	Eats	01 May 1967	01 May 2004

OrderNo	EmployeeNo	CustomerNo	Supplier	Price	Item
1	1	42	Harrison	$235	Desk
2	4	1	Ford	$234	Chair
3	1	68	Harrison	$415	Table

Another Example of the Keys

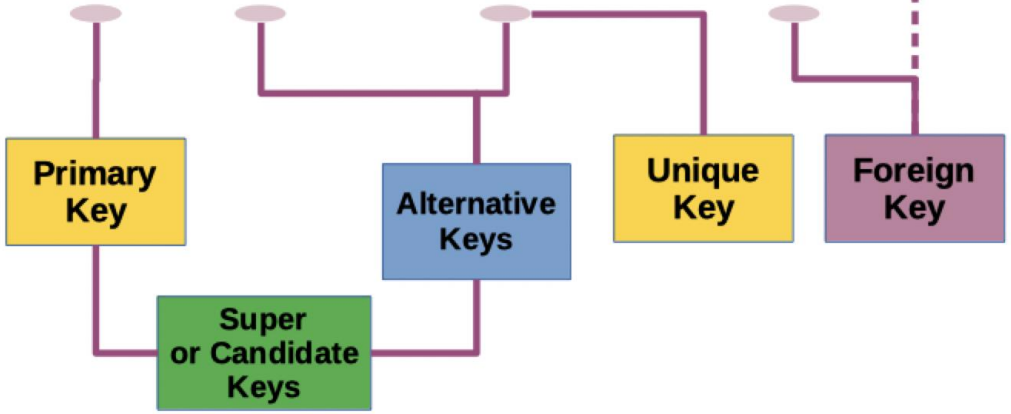
Primary keys indicated by a key icon



Yet Another Example of the Keys!

RowID	OrderNumber	BuyerName	BuyerID	BuyerAddress	DeptID
1	8000	Holmes	101	221b Baker Street	A
2	9192	Watson	102	221b Baker Street	A
3	1000	Hudson	103	221b Baker Street	A
4	4005	Moriarty	104	London	B
5	5003	Lestrade	105	Scotland yards	A
6	2303	Adler	106	London	B
7	5191	Milverton	107	London	B

DeptID	Type	Address
A	Detective	London
B	Criminal	London



The Theory of Foreign Keys

- Unless you have already made a reservation in restaurant, you cannot book a table
- If you have not already booked a hotel room in advance, you cannot get a room

Foreign Keys

Code in sandbox/foreignKeyDemo.txt

Foreign Keys

- A foreign key is a way to enforce referential integrity within your SQLite database. A foreign key means that values in one table must also appear in another table. The referenced table is called the parent table while the table with the foreign key is called the child table
- An enforced relationship between two tables.
- Information cannot be added unless it behaves according to the established relationship between two or more tables.

Add a primary key

```
/* Enable foreign keys */
/* Turn off maintainance of foreign */
/* key constraints to allow table alterations. */

PRAGMA foreign_keys = OFF;

DROP TABLE IF EXISTS Cars;
CREATE TABLE Cars (
carMake VARCHAR PRIMARY KEY,
registration VARCHAR,
capacity INT,
topSpeed INT );
```

The attribute carMake ensures uniqueness for a forced relationship from Agents.vehicleMake

Now, add a foreign key

```
DROP TABLE IF EXISTS Agents;

CREATE TABLE Agents (
            id INT PRIMARY KEY,
            lastName VARCHAR,
            vehicleMake VARCHAR,
            worksFor VARCHAR,
            FOREIGN KEY(vehicleMake) REFERENCES Cars(carMake) );

/* Turn on maintenance of foreign key constraints */
PRAGMA foreign_keys = ON;
```

The attribute vehicleMake associates this table to Cars.carMake

Populate Cars.carMake → AstonMartin

First, handle the primary key of the Cars table

We add the vehicle brand AstonMartin to Cars.carMake

```
INSERT INTO Cars values ('AstonMartin', 'MI6', 2, 130);
```

Now, populate the Agents table

Since the carMake attribute is “registered” we can add associated data

```
INSERT INTO Agents values (1007, 'Bond', 'AstonMartin', 'MI6');
INSERT INTO Agents values (1008, 'Wayne', 'AstonMartin', 'MI6');
INSERT INTO Agents values (1009, 'Smith', 'AstonMartin', 'MI6');
INSERT INTO Agents values (1010, 'Jones', 'AstonMartin', 'MI6');
INSERT INTO Agents values (1011, 'Nicholson', 'AstonMartin', 'MI6');
INSERT INTO Agents values (1012, 'Luxon', 'AstonMartin', 'MI6');
INSERT INTO Agents values (1013, 'Churchill', 'AstonMartin', 'MI6');
```

```
SELECT * FROM Cars;
SELECT * FROM Agents;
```

Populate Cars.carMake → Buick

First, handle the primary key of the Cars table

We add the vehicle brand Buick to Cars.carMake

...

Now, populate the Agents table again

If we add a carMake attribute is not “registered” then we get an error!

```
/* Error! Oh no! */
INSERT INTO Agents values(2008, 'Billy', 'Buick', 'MI6');
/* Error: need to first add "Buick" to the Cars table! */
```

Only AstonMartin drivers here ...

```
SELECT * FROM Cars;
SELECT * FROM Agents;
```

Populate Cars.carMake → Buick

First, handle the primary key of the Cars table

We add the vehicle brand Buick to Cars.carMake

```
INSERT INTO Cars values ('Buick', 'MI6', 5, 60);
```

Now, populate the Agents table

Since the carMake attribute is “registered” we can add associated data

```
INSERT INTO Agents values (2008, 'Billy', 'Buick', 'MI6');
INSERT INTO Agents values (2011, 'E-jay', 'Buick', 'MI6');
INSERT INTO Agents values (2012, 'Brick', 'Buick', 'MI6');
INSERT INTO Agents values (2013, 'Wedge', 'Buick', 'MI6');
INSERT INTO Agents values (2014, 'Orville', 'Buick', 'MI6');
INSERT INTO Agents values (2015, 'Lester', 'Buick', 'MI6');
INSERT INTO Agents values (2016, 'Wilbur', 'Buick', 'MI6');
INSERT INTO Agents values (2017, 'Rufus', 'Buick', 'MI6');
```

```
SELECT * FROM Cars;
SELECT * FROM Agents;
```

Populate Cars.carMake → Buick

First, handle the primary key of the Cars table

We write a query where we link Cars.carMake to Agents.vehicleMake

```
SELECT
        Agents.id, Agents.lastname, Cars.carMake, Agents.vehicleMake
FROM
        Cars, Agents
WHERE
        Cars.carMake == Agents.vehicleMake;
```

```
1007|Bond|AstonMartin|AstonMartin
1008|Wayne|AstonMartin|AstonMartin
1009|Smith|AstonMartin|AstonMartin
1010|Jones|AstonMartin|AstonMartin
1011|Luxon|AstonMartin|AstonMartin
2008|Billy|Buick|Buick
2011|E-jay|Buick|Buick
2012|Brick|Buick|Buick
2013|Wedge|Buick|Buick
2014|Orville|Buick|Buick
2015|Lester|Buick|Buick
2016|Wilbur|Buick|Buick
2017|Rufus|Buick|Buick
```

Consider this

- Can you create a similar base where a foreign key governs the data of another table?
- Can you write a query to show how the foreign key works?