# MASUD RAHMAN

(a.k.a., Mohammad Masudur Rahman)

**PhD** Student
**Software Research Lab**
Department of Computer Science
University of Saskatchewan, Canada

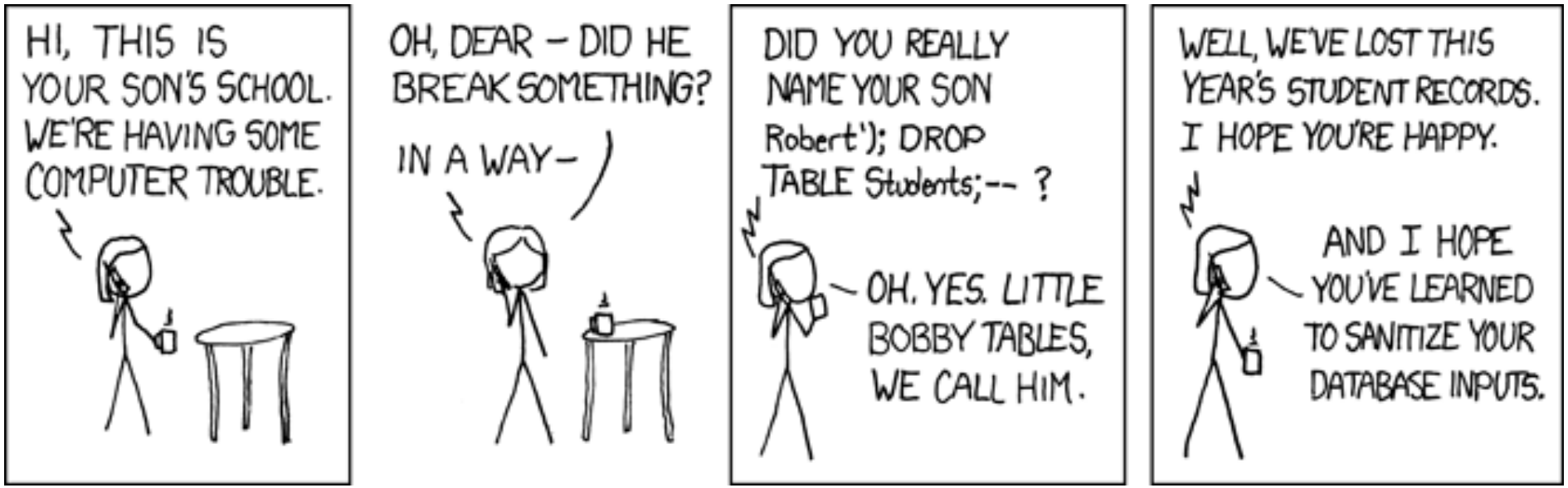
Email: **masud.rahman@usask.ca**

# Tutorials (Intermediate Software Engineering (CMPT 370))

- **SDLC** Phases (6)
- *Requirement Gathering & Analysis*
  - **UML**
- *System Design*
  - **Database**
- *Implementation & Coding*
  - **Source code documentation**
  - **Version control systems**
- *Software Testing*
  - **Unit & Integration testing**
- *System Deployment*
  - **Build tools**
- *Software Maintenance & Evolution*
  - **Clone detection, Concept Location**

# Tutorial #2

## Databases (Crash Course)



## Overview

- Relational Databases Overview
- Basic SQL
- Accessing Your Database
- Where to find more information

# Relational Databases

# Formal Definitions

- Database
  - A shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization.

- Relational Database
  - A collection of normalized relations (tables) with distinct relation names.
  - Based on the relational Model

# Formal Definitions

- Database Management System (DBMS)
  - A software system that enables users to define, create, maintain and control access to a database.
  - Implements a standard, but also may include vendor-specific features.
  - Examples:
    - Postgresql, MySql, Oracle, etc

# Informal Definition

Relational Database

- A bunch of data tables with logical relationships.
- Tables designed to store the information needs of an organization/program(s).
- Bundled with a language for defining the tables and querying information from it (SQL).

# Relational Model

Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

# Relational Model

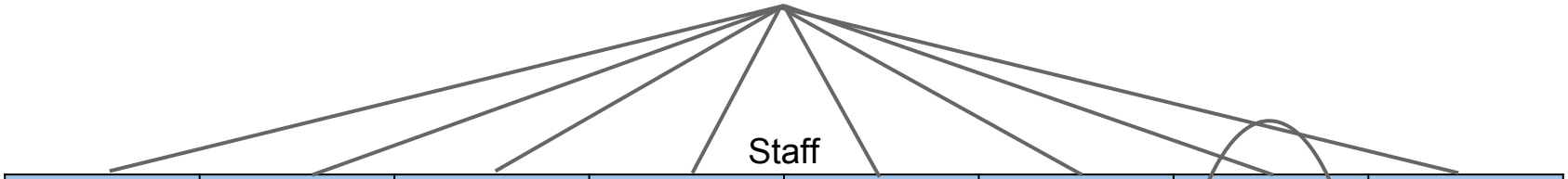
Relation - A table with columns and rows. (Table, File)

Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

# Relational Model

Attribute - A named column of a relation. (Column, Field)



Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

Attributes are assigned a domain of allowed values.

# Relational Model

Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

tuple - A row of a relation (a record).

# Properties of Relations

- Each relation has a unique name.
- Each cell contain a single value.
- Attributes have distinct names (table scope).
- Values of an attribute are from the same domain.
- Each tuple is distinct, no duplicates.
- The order of attributes holds no meaning.
- The order of tuples holds no meaning.

# Relational Keys

- Groups of one or more attributes that identify a single unique tuple within a relation.

# Key Types

Superkey

- An attribute, or set of attributes, that uniquely identifies a tuple within a relation.

Candidate Key

- A superkey such that no proper subset is a superkey within the relation.

# Keys

Candidate Key

Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

# Relational Model

A superkey (in fact any attribute set containing staffNo is a superkey for Staff)

Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

# Key Types

Primary Key

- The candidate key that is selected to identify tuples uniquely in a relation.

Foreign Key

- An attribute, or set of attributes, within one relation that matches the candidate key of some (possibly the same) relation.
- **Used to define logical relationships between tables.**

Primary Key

Foreign Key

Staff

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | Jane | Smith | Manager | F | 1980-01-01 | 150000 | 1 |
| 2 | Joe | Campbell | Employee | M | 1983-05-03 | 70000 | 1 |
| 3 | Carol | Martin | Employee | F | 1975-04-02 | 70000 | 1 |
| 4 | John | Smith | Manager | M | 1981-01-02 | 150000 | 2 |
| 5 | Diane | Williams | Employee | F | 1977-05-12 | 70000 | 2 |
| 6 | Daniel | Lee | Employee | M | 1985-11-06 | 70000 | 2 |

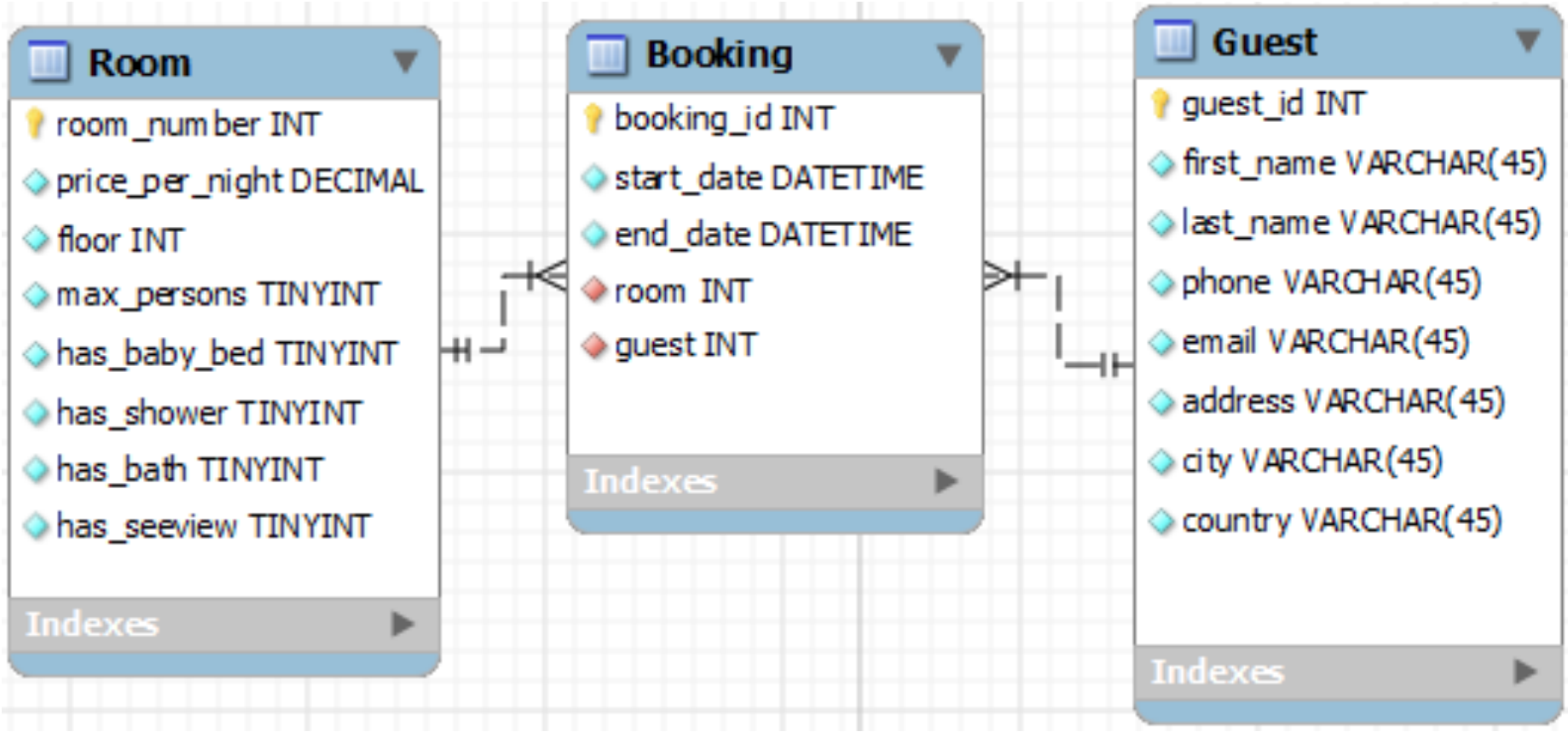
Defines a many to one relationship.

Primary Key

Branch

| branchNo | street | city | postalcode |
|---|---|---|---|
| 1 | 123 1st Street | Saskatoon | A1A 1A1 |
| 2 | 456 5th Street | London | B2B 2B2 |

# Small Database Example (Hotel)




Many to one relationships between Room and Booking and Guest and Booking.

Many to Many relationship between Guest and Room (requires a "junction" table).

# Basic SQL Tutorial

# Structured Query Language (SQL)

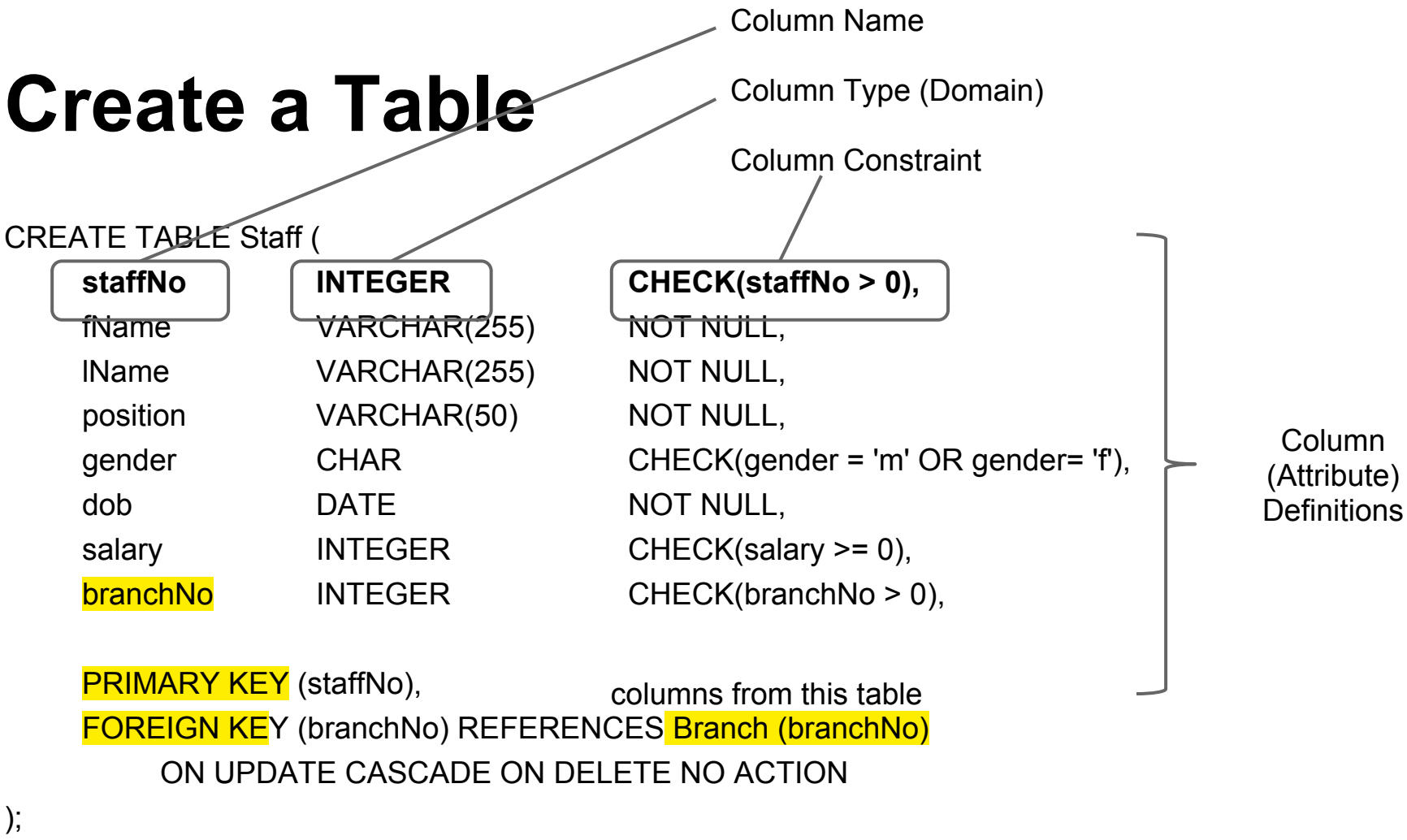
- Standard database definition and manipulation language.
  - Implemented by most DBMS.
- Used to define data (tables, domains) and query the database for the information it contains.
- A scripting language:
  - Command1;
  - Command2;
  - ....
- Ran in an interactive console, or sent from a program or programming API.

# Create a Table

```
CREATE TABLE Staff (
        staffNo         INTEGER         CHECK(staffNo > 0),
        fName           VARCHAR(255)    NOT NULL,
        lName           VARCHAR(255)    NOT NULL,
        position        VARCHAR(50)     NOT NULL,
        gender          CHAR            CHECK(gender = 'm' OR gender= 'f'),
        dob             DATE            NOT NULL,
        salary          INTEGER         CHECK(salary >= 0),
        branchNo        INTEGER         CHECK(branchNo > 0),

        PRIMARY KEY (staffNo),
        FOREIGN KEY (branchNo) REFERENCES Branch (branchNo)
                ON UPDATE CASCADE ON DELETE NO ACTION
);
```

# Create a Table



# Standard Data Types

| | |
|---|---|
| boolean: | BOOLEAN |
| character: | CHAR, VARCHAR(X) |
| numeric: | NUMERIC, DECIMAL, INTEGER, SMALLINT |
| floating: | FLOAT, REAL, DOUBLE PRECISION |
| datetime: | DATE, TIME, TIMESTAMP |
| interval: | INTERVAL |
| large objects: | CHARACTER LARGE OBJECT, BINARY LARGE OBJECT |

See vendor specific documentation for enhanced list of data types

# Column Constraints

NOT NULL

- Column must contain a value.

DEFAULT defaultValue

- Column's default value if none specified.

CHECK (condition)

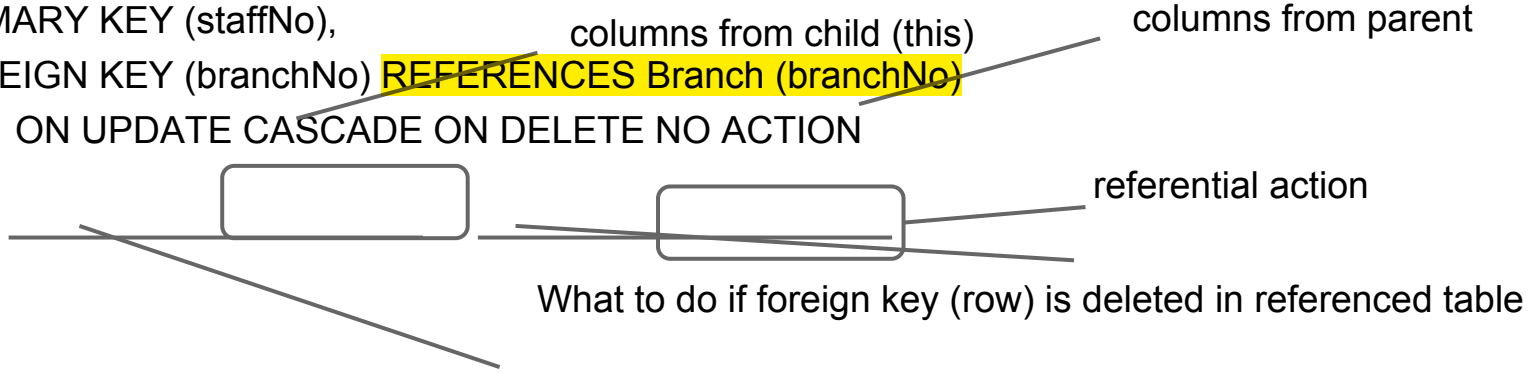
- Boolean condition referencing column.
  - AND, OR, NOT, =, <, >, <=, >=, != (or <>)

# Create a Table

Referential Integrity: If a foreign key contains a value (not null), that value must refer to an existing, valid row in the parent table.

```
CREATE TABLE Staff (
        staffNo         INTEGER             CHECK(staffNo > 0),
        fName           VARCHAR(255)  NOT NULL,
        lName           VARCHAR(255)  NOT NULL,
        position        VARCHAR(50)         NOT NULL,
        gender          CHAR                CHECK(gender = 'm' OR gender= 'f'),
        dob             DATE                NOT NULL,
        salary          INTEGER             CHECK(salary >= 0),
        branchNo        INTEGER             CHECK(branchNo > 0),

        PRIMARY KEY (staffNo),
        FOREIGN KEY (branchNo) REFERENCES Branch (branchNo)
                ON UPDATE CASCADE ON DELETE NO ACTION
);
```



# Create a Table (General Syntax)

```
CREATE TABLE TableName (
        {(columnName dataType [NOT NULL][UNIQUE][DEFAULT defOption]
                [CHECK (columnCondition)] [, ...]}
        [PRIMARY KEY (listOfColumns),]
        {[UNIQUE (listOfColumns] [, ...]}
        {[FOREIGN KEY (listOfForeignKeyColumns)
                REFERENCES ReferencedTableName [(listOfCandidateKeyColumns)]
                        [MATCH {PARTIAL | FULL}]
                        [ON UPDATE referentialAction]
                        [ON DELETE referentialAction]] [, ...]}
        {[CHECK (searchCondition)] [, ...]}
);
```

[, ...] == may repeat

# Deleting a Table

## General Syntax:

DROP TABLE TableName [RESTRICT | CASCADE]

## Example:

DROP TABLE Staff;

Restrict: Reject DROP operation if any objects exist which depend on this table.

Cascade: Cascade DROP operation to objects which depend on this table.

Objects: Such as foreign keys, views, etc.

# Insert a Row (Record/Tuple/...)

## General Syntax:

If not specified it assumes all columns are represented and in the order of their definition. Best to list them to be safe!

```
INSERT INTO TableName [(columnList)]
VALUES (dataValueList);
```

## Example:

```
INSERT INTO Staff (staffNo, fName, lName, position, gender, dob, sallary,
    branchNo)
VALUES (1, 'Jane', 'Smith', 'Manager', '1980-01-01', 'F', 150000, 1);
```

** Note, this insert will only succeed if the a row does not already exist with staffNo = 1 (primary key!) and if the Branches table contains a row with branchNo = 1 (foreign key!).

# Update a Row

## General Syntax:

```
UPDATE TalbeName
SET columnName1 = value [, columnName2 = value, ...]
[WHERE searchCondition];
```

## Example:

```
UPDATE Staff
SET salary = salary * 1.05
WHERE position = 'manager';
```

# Delete Row(s)

## General Syntax:

```
DELETE FROM TableName
[WHERE searchCondition];
```

## Example:

```
DELETE FROM Staff        -- fire employ with primary key staffNo=1
WHERE staffNo = 1;

DELETE FROM Staff;       -- fire all staff :(

DELETE FROM Staff        -- fire overpaid managers
WHERE salary > 300000 AND position = 'manager';
```

# Select Row(s)

## General Syntax:

```
SELECT [DISTINCT | ALL] {* | [columnExpression [AS newName]] [,...]}
FROM    TableName [alias] [, ...]
[WHERE condition]
[GROUP BY columnList] [HAVING condition]
[ORDER BY columnList [ASC | DSC]];
```

Select statements return data in table format.

The select statement syntax is complex and flexible, there is too much to cover in this short tutorial.

# Simple Select Examples

-- Grab a single record from a table (general)
SELECT * FROM TableName
WHERE pKeyColumn1 = value1 AND pKeyColumn2 = value2 AND ...;

-- Real Example
SELECT * FROM Staff WHERE staffNo = 1;

-- Result:

| staffNo | fName | lName | position | gender | dob | salary | branchNo |
|---|---|---|---|---|---|---|---|
| 1 | FirstName | LastName | Manager | F | 1980-01-01 | 150000 | 1 |

# Simple Multitable Example

```
-- example
SELECT b.branchNo, b.city, s.staffNo, s.position, s.firstName, s.lastName
FROM Branch b, Staff s
WHERE b.branchNo = s.branchNo
ORDER BY city ASC;
```

| branchNo | city | staffNo | position | firstName | lastName |
|---|---|---|---|---|---|
| 2 | London | 4 | Manager | John | Smith |
| 2 | London | 5 | Employee | Diane | Williams |
| 2 | London | 6 | Employee | Daniel | Lee |
| 1 | Saskatoon | 1 | Manager | Jane | Smith |
| 1 | Saskatoon | 2 | Employee | Joe | Campbell |
| 1 | Saskatoon | 3 | Employee | Carol | Martin |

# Transactions

- A logical unit of work consisting of one or more SQL statements that is guaranteed to be atomic.

- Effect of the transaction appears to occur atomically to concurrent users.

- Can be configured to give various levels of concurrency control (see documentation on SET TRANSACTION for your DBMS).

# Transactions

## Syntax:

```
BEGIN;          -- starts a transaction**

COMMIT;         -- ends a transaction normally, statemetns are atomically
                   applied

ROLLBACK; -- aborts transaction, all statements are aborted
```

** BEGIN is not part of SQL spec. SQL spec states that COMMIT begins a new transaction (as does logging in). Systems which use "auto commit" by default encapsulate every SQL statement in a transaction. In these systems BEGIN; is required to start a multi-statement transaction.

# Transactions

## Example (auto-commit system):

```
BEGIN;
INSERT INTO ....
UPDATE ...
DROP TABLE importantTable; --oops!!
ROLLBACK;
BEGIN;
INSERT INTO ...
UPDATE ...
COMMIT;
```

# SQL Resources

Database Systems: A Practical Approach to Design, Implementation and Management (Thomas Connolly, Carolyn Begg)

http://www.w3schools.com/sql/ (a bit limited)

http://www.postgresql.org/docs/9.2/static/sql.html (specific to postgresql)

http://dev.mysql.com/doc/refman/5.5/en/sql-syntax.html (specific to mysql)

# Connecting to Your Database

Postgresql:

```
psql -h edjo.usask.ca -d dname -U uname
```

MySQL:

```
mysql --host=edjo.usask.ca --database=dname --user=uname -p
```

* This will get you into an interactive console for your database.
* These commands both work on tuxworld
* Easiest way to access from home is to ssh into tuxworld.
  (Use terminal in OSX/Linux, or Putty with windows)
* Or you can install postgresq/mysql at home and use the console/gui tools.

# Connecting to your Database (GUI)

MySQL:

http://www.mysql.com/downloads/workbench

- GUI application for viewing/managing database.

Postgresql:

http://www.dbvis.com/

# Next Tutorial: JDBC

Connecting to and operating your database from java.

# Transactions

## Example (standard SQL):

INSERT INTO ....

UPDATE ...

DROP TABLE importantTable; --oops!!

ROLLBACK;

INSERT INTO ...

UPDATE ...

COMMIT;

# Referential Actions

| Referential Action | ON UPDATE | ON DELETE |
|---|---|---|
| CASCADE | Update is cascaded to the rows referencing the updated one. | Cascades the delete operation to rows referencing the deleted one. |
| SET NULL (required columns allowed null) | Set effected column to null, if allowed. | Set effected column to null, if allowed. |
| SET DEFAULT (requires columns have default value) | Set effected column to default, if one is specified, else reject. | Set effected column to default, if one is specified, else reject. |
| NO ACTION (default) | Reject original operation. | Reject original operation. |

# Domains (Custom "data types")

## General Syntax:

```
CREATE DOMAIN Name [AS] dataType
[DEFAULT defaultOption]
[CHECK (searchCondition)];
```

## Example:

```
CREATE DOMAIN gender AS CHAR
DEFAULT 'F'
CHECK (VALUE IN ('M', 'F'));
```