# CS2015 Data Management

## The Relational Model – Lecture 2

* Concept of model
* Definitions and terminology regarding relational model
* Why relational model?

**Machine**

* Physical storage of data
* Includes low level details about data that are not important for user

**User**

* Logical view of data (tables)

User’s data model is what we focus on.

The relational data model sits **in between** the machine and the user.

We want to model the data with an abstract way so we can manipulate it etc. easily.

* A model represents something
* E.g. architects can model a bridge before it is built.
* Models can be blueprints used for testing and learning

The theory of DBs involves many models.

* Relational model, ER model, Object model, Network model, Document model

**The relational model** is the user’s data model; we learn it first.

Apply our learning on a few real **RDBMS** (Relational Database Management System), MS Access & MySQL

Relational models are simple, we can use mathematical approaches with them. The model describes the pure concepts behind real database systems.

The **Relational Model** is a collection of connected tables (relations).

**Named tables (relations) have**:

* Named columns (attributes)
* Unnamed rows (tuples)

**Domain**: a set of allowable values

**Relationships**: Links or connections

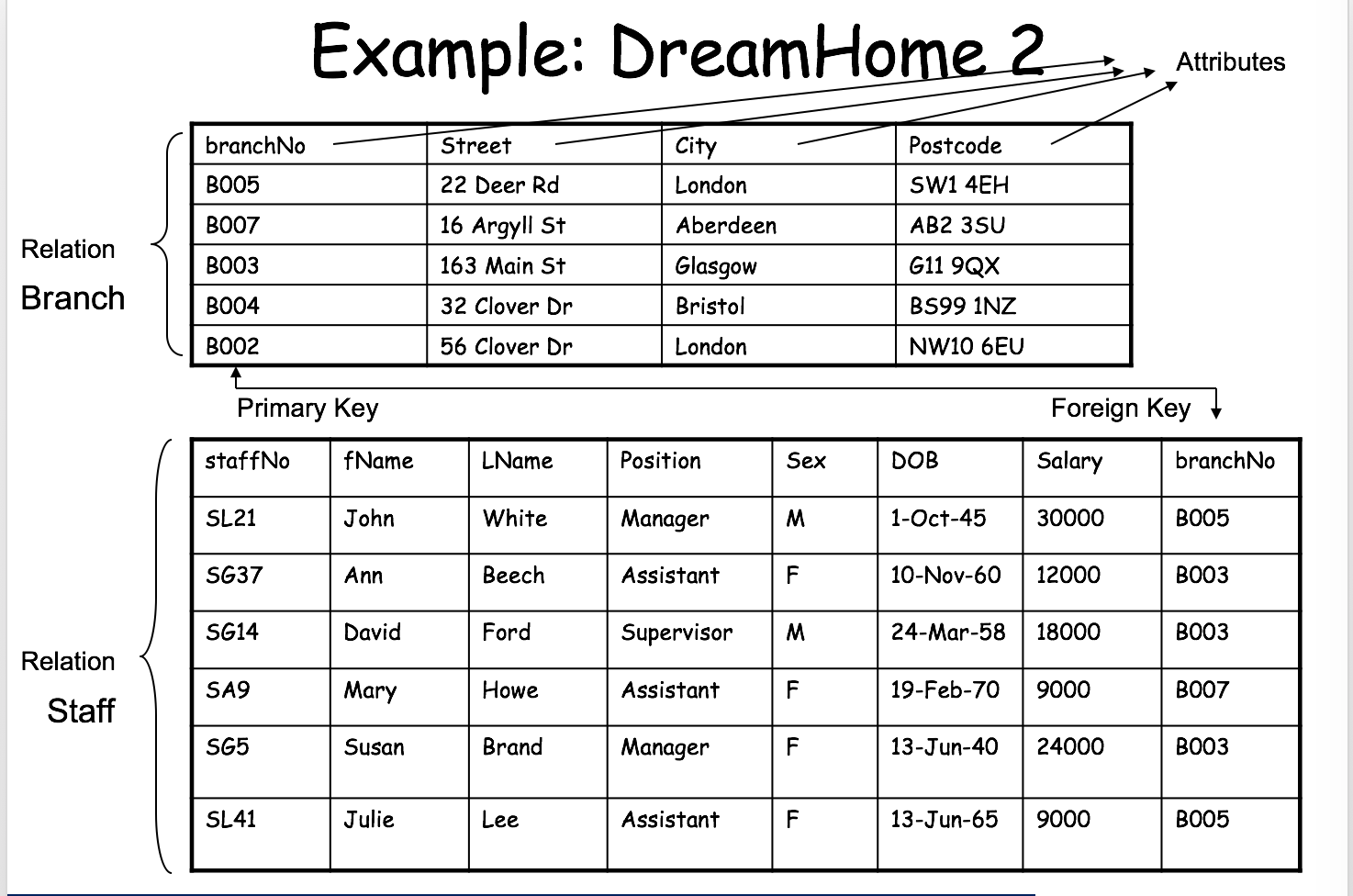
**Keys:** Primary, foreign

**Integrity Constraints**

Foreign and primary keys can link tables together. E.g. A table with branch locations with unique BranchNo, and this can link to the staff table who work at different BranchNo’s.

**Primary key** must be unique, since it is the branch identifier. Cannot be null.

**Foreign key** can be duplicated, since many people can work for the same branch. Refers to another table.



# Relation (table)

**Schema -**  The name of the relation, set of attributes & their respective domains.

* E.g. Branch (BranchNo: domain1, street: domain2, city: domain3, domain etc.)

**Instance –** Set off attributes and their values in a tuple.

* E.g. (Branch: 001, Street: Mary lane, City: Aberdeen etc.)

Keys is an attribute or set of attributes that uniquely identify a row.

**Properties of relations**

Rows & Columns can be in any order.

No two rows are identical.

All values belonging to an attribute are from the same domain.

Attributes can have a null value.

The main strength of the relational model is that it supports powerful data retrieval (queries).

Output of queries is displayed in a relation(table).

# Sep 20th SQL

Lecture

* Basic concepts of SQL

SQL is based on structured English. It is a simple declarative language.

Developed by IBM in 1975.

Java, ruby etc. is a procedural language. It goes through the code top to bottom, line by line. Tells computer what to do.

SQL is declarative, you tell the computer what you want and the computer does the ‘work’.

ANSI SQL = American National Standards Institute.

There are different dialects of SQL.

SQL syntax

CREATE TABLE Staff (StaffNo INTEGER, Salary FLOAT, Lname VARCHAR(20) );

INSERT INTO Staff VALUES (32,25000.0,’Smith’);

* Keywords are upper case
* Text data is enclosed in single quotes (‘’)
* Round brackets (‘(‘) are used to group related items
* Commas separate items in a list
* Statements are terminated with a semicolon;

SELECT statement retrieves & formats data

SELECT \* FROM Staff

* \* is a wild card- all columns

SELECT DISTINCT target\_list FROM STAFF ->> Avoids duplicates

SELECT target\_list FROM relation-list WHERE qualification

* Qualification – comparisons and attributes
* SELECT Lname, Position, Salary FROM Staff WHERE Salary > 20000;

Building more complex queries

SELECT \* FROM Staff WHERE (Position = ‘Manager’) OR (Position = ‘Assistant AND Salary > 10000);

Other predicates: BETWEEN, IN, LIKE

SELECT \* FROM Staff WHERE (Salary BETWEEN 10000 AND 20000) AND (Position IN(‘Manager’,’Assistant’)) AND (Lname LIKe ‘S%’ OR Lname LIKE ‘W\_\_\_\_’);

Describing SQL Syntax using BNF notation

# SQL Components DDL, DCL & DML

**DDL:** Data Definition Language for creating a DB, Create, Drop, Alter

**DCL:** Data Control Language for administering a DB e.g. GRANT, DENY, USE

**DML:** Data Manipulation Language to access a DB e.g. SELECT, INSERT, UPDATE, DELETE

CB use a special BNF (Backus-Normal form= is a meta language

**Meta language** a language that describes language.

SELECT [ DISTINCT | ALL ]

{ \* | [ Colexpr [ AS Newcol ] ] [, ...] }

FROM TableName [ Alias ] [, ...]

[ WHERE Predicate ] [ GROUP BY Columnlist ]

[ HAVING Predicate ] [ ORDER BY Columnlist ] [;]

Query by example (QBE) – MS Access makes simple form-based methods of specifying queries, which essentially show SQL queries in a simple table format.

# SQL II

## Use of Distinct

* List the property numbers of properties that have been viewed to reduce duplication.
* Add distinct to the SQL command to get rid of duplicates.
* E.g. SELECT DISTINCT propertyNo FROM viewing;

## Ordering the query

* SELECT staffNo, fName, lName, salary FROM Staff ORDER BY Salary DESC;
* Or
* SELECT staffNo, fName, lName, salary FROM Staff ORDER BY 4 ASC;

## Calculated fields

* Produce a list of monthly salaries, not yearly salaries
* SELECT StaffNo, fName, lName, salary/12 FROM Staff;

## Renaming Columns

* We can rename columns if we e.g. use calculations with them
* SELECT StaffNo, fName, lName, Salary/12 AS monthlySalary FROM Staff;

## SQL Aggregate Functions

* To summarize data
* Aggregate functions: SUM, AVG, MIN, MAX, COUNT
* SELECT MIN(Salary) AS myMin, MAX(salary) AS mymax FROM Staff;

## Count (\*)

* Counts the number of rows in a table
* SELECT COUNT(\*) as WomenStaff FROM Staff WHERE Sex=’F’;

## Group By

* Summarise whole columns of data into one row.
* E.g. Find the number of staff working in each branch and the sum of their salaries
* SELECT BranchNo, COUNT(staffNo) AS myCount, SUM(Salary) AS mySum FROM Staff GROUP BY BranchNo;

## Having

* SELECT BranchNo, COUNT(staffNo) AS myCount, SUM(salary) AS mySum FROM Staff GROUP BY BranchNo HAVING COUNT(staffNo)>1;
* You need to use HAVING instead of WHERE when you use aggregate functions.

## Adding data using INSERT

* INSERT INTO Staff (StaffNo, Salary, Position, Lname) VALUES (322,15000,’Assistant’,’Smith’);
* If you specify the insert, you can put them in any order otherwise they must be in the order of the DB

UPDATE

DELETE FROM

DROP TABLE Staff;

You can join query results together with UNION or JOIN

## Set Operations

* Null can do a lot of shit
* SELECT \* FROM Staff WHERE Fname IS NULL;
* SELECT .. UNION/INTERSECT/EXCEPT SELECT …

(SELECT City FROM Branch) INTERSECT (SELECT City FROM PropertyForRent);

You can nest select statements. E.g. find all members of staff who earn more than Smith

SELECT \* FROM Staff WHERE Salary > (SELECT Salary FROM Staff WHERE Lname ='Smith');

# SQL III

## Querying multiple tables

* To do this, you need to join the tables

In our example we have tables for Property for rent, client and viewing.

We want to list all the properties that a given client has viewed.

First we can:

* List the property numbers viewed by client number CR56
* *SELECT PropertyNo FROM viewing WHERE ClientNo = ‘CR56’;*

*But we want to see client name & property details*

## Apply heading 2: CMD+Opt+2

**2nd try:**

* SELECT Viewing, PropertyNo, Street, City, ViewDate
* FROM Viewing, PropertyForRent
* WHERE ClientNo = ‘CR56’
* AND Viewing.PropertyNo = PropertyForRent.PropertyNo;

*However,* this does not show the Name of the customer, just the ClientNo

**3rd try:**

* SELECT Fname, Lname, Street, City, ViewDate
* FROM Viewing, PropertyForRent, Client
* WHERE Viewing.ClientNo = ‘CR56’
* AND Viewing.PropertyNo = PropertyForRent.PropertyNo;
* AND Viewing.ClientNo = Client.ClientNo;

*However,* this requires the ClientNo which users shouldn’t have to know about

**4th try:**

* SELECT Fname, Lname, Street, City, ViewDate
* FROM Viewing, PropertyForRent, Client
* WHERE Fname = ‘Aline’ AND Lname = ‘Stewart’
* AND Viewing.PropertyNo = PropertyForRent.PropertyNo;
* AND Viewing.ClientNo = Client.ClientNo;

*This shows a table with the first and last names properly!*

## Table Aliases

In the FROM part, you can specify the table with an alias ‘x’ to reduce typing.

* … FROM Viewing V, PropertyForRent P, Client C …

## Natural Joins

Use Natural Join.

The DBMS knows which columns are key columns.

The following is exactly the same as the 4th try earlier:

SELECT C.Fname, C.Lname, P.Street, P.City, V.ViewDate

FROM Client C NATURAL JOIN

(Viewing V NATURAL JOIN PropertyForRent P)

WHERE C.Fname = 'Aline' AND C.Lname = 'Stewart';

*This is not required stuff*

## ANSI SQL SYNTAX

SQL Inner, Left, Right and full join.

## Cross join (Carthesian Product)

Multiplies tables with each other, creates a massive table.

## Theta Joins

Use a WHERE predicate, or symbols like < > => <=

= equal join

E.g.

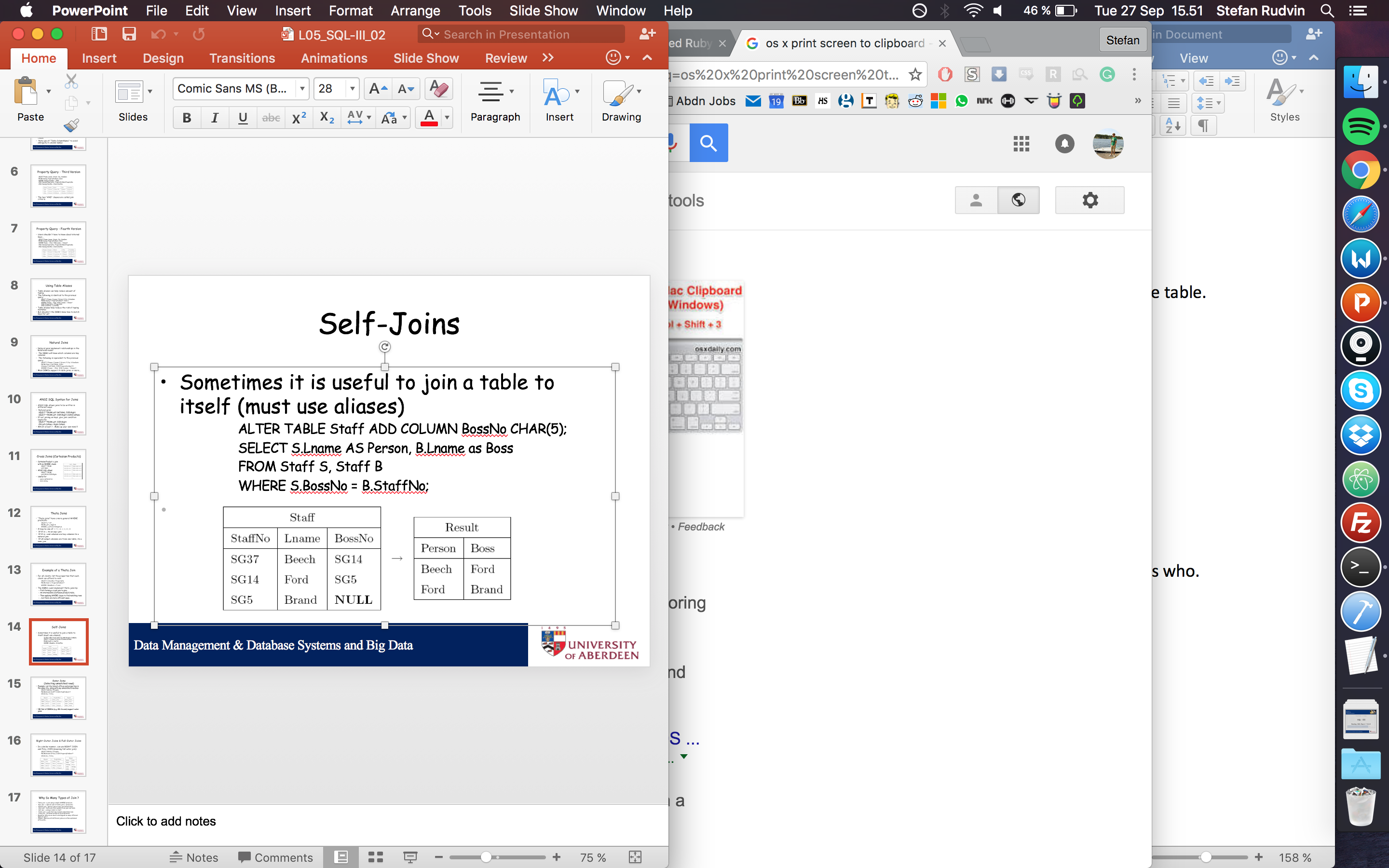
SELECT C.ClientNo, P.PropertyNo

FROM Client C, PropertyForRent P

WHERE C.MaxRent >= P.rent;

## Self-Join

Join a table with itself to e.g. find out who manages who.



## Outer Join

Select unmatched rows.

## Right Outer & Full Outer Join

List everything not matched etc.

## Queries with different joins are often optimised differently

# Database System Lifecycle – Lecture

## Database Design

* DBMS selection
* App design
* Prototyping (optional)
* Implementation
* Data conversion and loading
* Testing
* Operational Inheritance

## Phases of DB Design

Database Design Achieved in three phases:

* Conceptual – model data independent of all physical considerations
* Logical – refine and map conceptual model onto relational model
* Physical – map logical model onto a specific DBMS e.g. MySQL

A good design facilitates efficient storage and retrieval of data

USE UML Notation for conceptual and Logical DB design!

7 Oct Data Modelling

# Logical Database Design

Derive a logical model from the info represented in the ER model.

Validate the logical model to check if it fulfils clients:

* Data and
* Transaction requirements

We focus on one type of logical model which is the relational model

Our method of designing relational (logical) model uses information form the ER model.

ER model:

* Partitions information in a domain into Entities and attributes
* Links entities up into a network to reflect the relationships from the real world domain.

Relational model:

* Partitions info into tables(relations)
* Links tables up into a network to reflect relationships existing among data.

ER and relational models have similarities.

We exploit these similarities to carry out relational database design.

## Step procedure for logical DB design

* Derive relations for logical data model
* Validate relations using normalization
* Validate relations against user transactions
* Check integrity constraints
* Review logical data model with user
* Check for future growth.

We focus on the first two steps.

Conceptual -> Logical -> Physical Modelling

Logical = relational model in our case.

An **entity relationship model**, also called an **entity-relationship** (**ER**) **diagram**, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

DB Design – Normalization

Read C & B book on DB design

Relations derived from the ER model may be ‘faulty’

* May cause data redundancy and insert/delete

## Mathematical notions

Mathematically represent collections of objects in the form of sets.

e.g. of fName, lName, make a dict set of all this data.

Cartesian product – cross product of sets.

Denote relations for related objects.

Relation – defined between two sets and is a subset of cross product between those two sets.

## Function

A special kind of relation

Some queries give us answers that are derived from a function.

Normalization – reduce amount of duplication within the database.

Functional dependency – is a property of the meaning of the attributes in a relation

Given X you can derive Y: Y is dependent on X

In databases we work with relations in general form (A, B, C, D... Z)

## Data Redundancy

Major aim of relational DB design is:

* Group attributes into relations to minimize data redundancy
* Reduce file storage space required by base relations

It is undesirable because there are INSERT, DELETE, UPDATE anomalies.

Functional dependency can reduce the amount of data we store, when many tables have the same rows with the same data.

# First Normal Form (1NF)

Reorganising table & data into a table format that doesn’t require more attributes.

# Second Normal Form (2NF)

Based on concept of full functional dependency:

* A and B are attributes of a relation R
* B is fully dependent on A if by is functionally dependent on A but not on any proper subset of A

Identify primary key

Identify functional dependencies

If partial dependencies exist on the primary key, remove them by placing them in a new relation along with copy of their determinant.

# Third Normal Form (3NF)

Based on transitive dependencies

A relation that is in 1NF and 2NF in which no non-primary-key attribute is transitively dependent on the primary key.

You need to know up to third normal form, probably on exam

SQL’s Integrity Enhancement Features (IEF)

Database integrity is important because real databases are used and change all the time.

IEF’s allow the DB designer to specify & enforce:

* Domain constraints
* Required Data
* Entity integrity
* Referential integrity
* Enterprise Constraints

## Creating tables – Data definition

* CREATE TABLE is used to define relational tables
* Defines data type for each column
* Defines rules for how data can be inserted and deleted

CREATE TABLE Staff (StaffNo VARCHAR(5),

Lname VARCHAR(20),

Salary FLOAT,

HireData DATE);

These are very important. You can also use “Blob” that can store any file type.

## ANSI Standard Terms in SQL

* CHARACTER (CHAR), VARCHAR
* Etc.

You can also define your own data type(domain) to use in your DB.

CREATE DOMAIN SexType AS CHAR(1)

DEFAULT ‘M’

CHECK(VALUE IN(‘M’, ‘F’));

NOT NULL and CHECK give domain constraints, and are crucial.

E.g. Salary FLOAT CHECK(Salary BETWEEN 50 and 20000);

## Dynamic domain constraints

* Determined based upon data inserted into DB

## Scalar functions

Used to convert/manipulate data values

e.g. SELECT SUBSTRING(Lname FROM 1 TO 3),

CONVERT(INTEGER Salary),

EXTRACT(YEAR FROM HireDate)

FROM Staff

## Entity Integrity – Primary Keys

Foreign keys: Links a child table to a parent table.

SQL has keywords to identify the foreign keys.

Referential Integrity and referential actions

Foreign keys can reference a table on DELETE/UPDATE etc.

Enterprise constraints

Constraint data types by number or another value.

## Triggers

Triggers contain **procedural code** that enter an action. E.g. send an email if a data value becomes too large.

# Schemas

Schemas are a data model, constraint the whole database. These evolve over time.

Access Control & Views

## Reading D & B Chapter 7

All information is up to date with the University MySQL version. (not current)

In large organisation, DBMSs are used by a range of staff.

Consequently, access to data in different tables may need to be controlled

* E.g. Access control to salary information in a company
* Restrict users based on tables, operations performed on tables etc.

## SQL’s Access Control Model

* Access control is similar to multi-user operating systems, Unix, Windows etc.
* User supplies an Authorisation Id and password
* DBMS opens a session for the user
* DBMS runs SQL statements on *behalf* of the user
* User becomes the *owner* of any objects he creates
* By default, only the *owner* may access his objects
* Owner can *grant* or *revoke* **access privileges** to other users.

## Granting Privileges

* Might not work in UNI system
* Controversial, secretive stuff (?)

GRANT {PrivilegeList | ALL PRIVILEGES }

ON ObjectName

TO {AuthIdList | PUBLIC } [WITH GRANT OPTION]

## Views

* A **view** is a virtual table, constructed from base tables
* Views are manipulated by the schema of the privileges granted to the user
* Can grant and revoke privileges on a view

## Creating Views

* Views are created with a command and an SQL statement
* Vertical & Horizontal views: restrict columns that may be seen e.g. Branch manager doesn’t need to see all employees in the country.
* User admin owns all tables, can grant and revoke privileges to parts of a DB

Database Security – Guest talk

Data- Information is valuable

Thieves can sell or use information.