SQL: Calculations & the Select clause

Database Analysis and Design

The **Select clause** can include

- Column names
- **Expressions or calculations**

Example with a calculation:

```
SELECT    Name,  Test1,   Test2,   Test1+Test2
FROM      StudentResults
```

Name	Test1	Test2	Test1+Test2
Ted	10	5	15
Sue	5	6	11
Emma	9	9	18

```
SELECT      Name,  Test1,   Test2,   Test1+Test2
FROM        StudentResults
ORDER BY    4 DESC;
```

Sort the result set by the 4th column

Name	Test1	Test2	Test1+Test2
Emma	9	9	18
Ted	10	5	15
Sue	5	6	11

KNOW ING

Column Headings & the Select Clause

The **Select clause** can include

- **Alternative columns headings** (only used in the result set)
  - Use the **AS** keyword
    - **AS** keyword is **optional**!
  - New heading is specified in **DOUBLE QUOTES**
    - Double quotes are optional if single word is used

Example with a new column headings:

```
SELECT      Name,  Test1,   Test2,
            Test1+Test2  As "Total  Score"
FROM        StudentResults
```

Name	Test1	Test2	Total Score
Ted	10	5	15
Sue	5	6	11
Emma	9	9	18

```
SELECT      Name "Student   Name",  Test1,   Test2,
            Test1+Test2  "Total  Score"
FROM        StudentResults
ORDER BY  Name;
```

Student Name	Test1	Test2	Total Score
Emma	9	9	18
Sue	5	6	11
Ted	10	5	15

```
SELECT      Name Studentname, Test1+Test2  Total,  …
```

SQL:Table name qualification

Last week, we wrote a simple SQL query using the Select statement

```
SELECT LecId, LecName, Age FROM  lecturer
```

Whenever we use a column name in a Query, we can qualify (prefix)
with the appropriate table name

```
SELECT lecturer.LecId, lecturer.LecName, lecturer.Age
FROM  lecturer
```

The usefulness of this feature becomes apparent when
SQL statements refer to columns from two or more tables

KNOW ING

SQL:Table name aliases

Tablenames in SQL statements may be assigned a temporary alias

- The alias only applies to the current SQL statement (not remembered)
- Once an alias is used within an SQL statement, you can cannot refer to the original tablename within that SQL statement

1
```
SELECT lecturer.LecId,  lecturer.LecName,  lecturer.Age,
FROM lecturer
WHERE lecturer.age => 50;
```
no alias used

2
```
SELECT L.LecId,  L.LecName,  L.Age
FROM lecturer L
WHERE L.age => 50;
```
L is the alias for the lecturer table

3
```
SELECT budget.deptno, budget.monthly_amt
FROM    wr207_V_X5 budget
WHERE budget.year = 2011;
```
budget is the alias for the table named wr207_V_X5

SQL: Query involving two tables

So far our SQL queries have retrieved data from a **single** table.

- Often a query needs to retrieve data from **two** tables.
- Usually the tables are **related** via Foreign Key / Primary Key relationships
  - E.g. List all subject codes and their matching convenor names

SubjectCode	LecName
INF11002	John Smith
INF11007	Carol Kent
INF35700	John Smith
INF35606	Jane Pitt

SUBJECT

SubjectCode	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

LECTURER

LecID	LecName	Age
207	John Smith	37
119	Jane Pitt	26
345	Carol Kent	34

SQL: Query involving two tables

Suppose we write this:

```
SELECT SubjectCode, LecName FROM Subject, Lecturer
```

- The result set seems wrong. **Every row** in the **first table** has been **matched** with **every row** in the **second table**
- This is called a **Cartesian Product**
- A **Cartesian Product** is usually a **mistake**

LECTURER

LecID	LecName	Age
207	John Smith	37
119	Jane Pitt	26
345	Carol Kent	34

SUBJECT

SubjectCode	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

SubjectCode	LecName
INF11002	John Smith
INF11002	Jane Pitt
INF11002	Carol Kent
INF11007	John Smith
INF11007	Jane Pitt
INF11007	Carol Kent
INF35700	John Smith
INF35700	Jane Pitt
INF35700	Carol Kent
INF35606	John Smith
INF35606	Jane Pitt
INF35606	Carol Kent

SQL: Cartesian Product problem

- A Cartesian Product occurs because
  - **Select** statements **ignore** all **referential integrity** constraints that may have already been implemented
  - So an **SQL Select** statement does **not** 'know' how two tables are **related** (it doesn't consider any existing FK – PK constraints)
    - Why? SQL has to allow for queries based on non-related tables (perhaps even originating in different databases).
- **Solution**
  - You must specify how two tables are **related** in **every** Select statement that you write
  - This is known as a **Join**
  - There are a **few** ways that tables can be **joined**
    - We will concentrate on **Inner Joins**

SQL: Inner Joins

- An **Inner Join** returns a result set that contains only data that satisfies a **Foreign Key – Primary Key condition**

- Syntax:

```
SELECT      <column-names>
FROM        <table-name1>
INNER JOIN  <table-name2>
ON          <join-condition>
```

- The **<join-condition>** is normally in the format

  <foreign-key column-name> = <primary-key column name>

```
SELECT      S.SubjectCode, L.LecName
FROM        Subject S
INNER JOIN  Lecturer L
ON          S.LecId = L.LecId
```

SQL: Inner Joins

Each row in Subject will only be joined with a row in Lecturer where the **Foreign Key – Primary Key** condition is **satisfied**

```
SELECT     S.SubjectCode,   L.LecName
FROM            Subject S
INNER JOIN      Lecturer L
ON         S.LecId = L.LecId
```

RESULT SET

SubjectCode	**LecName**
INF11002	John Smith
INF11007	Carol Kent
INF35700	John Smith
INF35606	Jane Pitt

SUBJECT

SubjectCode	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

LECTURER

LecID	LecName	Age
207	John Smith	37
119	Jane Pitt	26
345	Carol Kent	34

KNOW
ING

Column Prefixes

SQL interpreters are smart.

If every column name used in a query then alias prefixes are not required*

```
SELECT   SubjectCode, Title, CreditPoints, LecName
FROM          Subject S
INNER JOIN    Lecturer L
ON       S.LecId = L.LecId
```

No alias used

Alias prefix <u>must</u> be used

SUBJECT

<u>SubjectCode</u>	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

LECTURER

<u>LecID</u>	LecName	Age
207	John Smith	37
119	Jane Pitt	26
345	Carol Kent	34

Column Prefixes

Generally, database professionals **prefix all** column names with **aliases** in their SQL

- even when the code only uses 1 table
- even if all columns names are unique
- Future-proofs queries. A query may be modified to include new tables / columns.

```
SELECT     S.SubjectCode, S.Title, S.CreditPoints, L.LecName
FROM              Subject S
INNER JOIN    Lecturer L
ON             S.LecId = L.LecId
```

* Always use Alias prefixes

SUBJECT

SubjectCode	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

LECTURER

LecID	LecName	Age
207	John Smith	37
119	Jane Pitt	26
345	Carol Kent	34

SQL: Avoid old style Joins

Back in the **dark ages** (pre 2000), some DBMS products **did NOT use/have** the **Inner Join** keyword.

- Instead the Join was expressed as part of the **where clause** within the Select statement

Many **old-time** developers, **old-time** web sites, **old-time** books & authors still use this old-style in their code and in their examples

```
SELECT    S.SubjectCode,  L.LecName
FROM      Subject S,  Lecturer L
WHERE     S.LecId = L.LecId
```

SQL: Avoid old style Joins

- This **old style** join gets **confusing** when you add other restriction criteria to your **where clause.** **Avoid old style join.**
  - The **Where** clause
    - becomes **cluttered and complex:**
    - may require use of **additional brackets**
    - Becomes source of **bugs**
      - these two examples will produce different results

```
SELECT   S.SubjectCode,   L.LecName
FROM     Subject S,  Lecturer L
WHERE    S.LecId = L.LecId
     AND L.Age > 50
      OR L.Title = 'Professor'
```

```
SELECT   S.SubjectCode,   L.LecName
FROM     Subject S,  Lecturer L
WHERE    S.LecId = L.LecId
     AND ( L.Age > 50
      OR   L.Title = 'Professor' )
```

When you think you have mastered SQL (say in 4 weeks time) then try to uses this old-style technique.
You will see it used in industry by dinosaur SQL developers (anyone over 42 years of age)

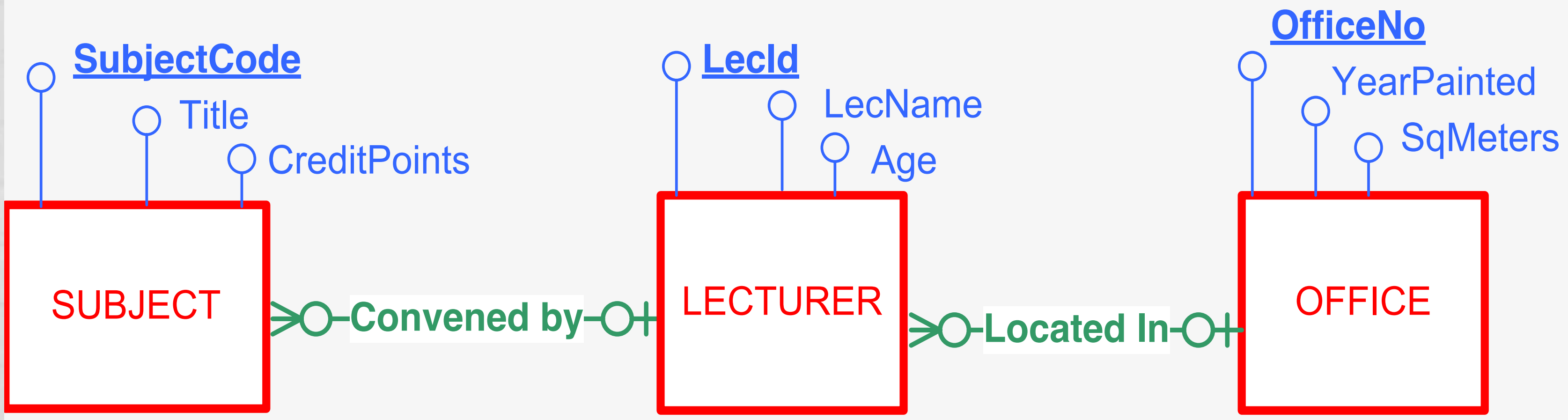
ERD with multiple relationships

The ERD below has been modified and has an **additional** relationship

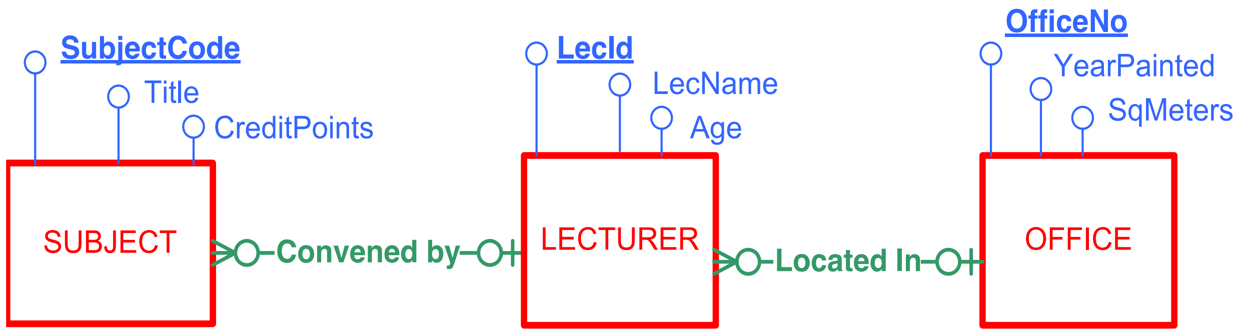
- One Lecturer May be Located in **ONE** office
- One Office May have **MANY** Lecturers

Remember:

- Each M:1 relationship will **always** generate a **FK** in the relational model
- A FK is **always** located at the **MANY** end of a M:1 relationship



Convert ERD to Relational Model



The ERD has two M:1 relationships

The Relational Schema will have **two FKs**

Each M:1 relationship generates a FK in the relational model

OFFICE (<u>OfficeNo</u>, YearPainted, SqMeters)

LECTURER (<u>LecId</u>, LecName, Age, OfficeNo)
Foreign Key OfficeNo References OFFICE

SUBJECT(<u>SubjectCode</u>, Title, CreditPoints, LecId)
Foreign Key LecId References LECTURER

SQL: Query involving two tables

Q. How large is the office of the convenor of INF11002?

Q. When was the office of the convenor of Media Thesis A last painted?

LECTURER

LecID	LecName	Age	OfficeNo
207	John Smith	37	EN710
119	Jane Pitt	26	EN505
345	Carol Kent	34	EN710

SUBJECT

SubjectCode	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

OFFICE

OfficeNo	YearPainted	SqMeters
EN710	2004	12
EN505	2011	8.2
BA915	2007	10.5

SQL: Query involving 3 tables / 2 inner joins

This single SQL statement uses **two Inner Joins** to retrieve data from the
three related tables

```
SELECT       S.SubjectCode, L.LecName, O.SqMeters
FROM         Subject  S

INNER JOIN   Lecturer  L
ON            S.LecId = L.LecId

INNER JOIN   Office  O
ON            L.OfficeNo = O.OfficeNo

ORDER BY     L.LecName, S.SubjectCode
```

SUBJECTCODE	LECNAME	SQMETERS
INF11007	Carol Kent	12
INF35606	Jane Pitt	8.2
INF35700	John Smith	12
INF11002	John Smith	12

SUBJECT

SubjectCode	Title	CreditPoints	LecId
INF11002	Intro to Web	12.5	207
INF11007	EBIS	12.5	345
INF35700	Honours Project	50	207
INF35606	Media Thesis A	25	119

LECTURER

LecID	LecName	Age	OfficeNo
207	John Smith	37	EN710
119	Jane Pitt	26	EN505
345	Carol Kent	34	EN710

OFFICE

OfficeNo	YearPainted	SqMeters
EN710	2004	12
EN505	2011	8.2
BA915	2007	10.5