Database Systems 2

Lecture 13

Physical Design 1

Overview of the Methodology

Conceptual Database Design

1 Build local conceptual data model for each user view.

Logical Database Design

2 Build and validate local logical data model for each user view

3 Build and validate global logical data model

Physical Database Design

4 Translate global logical data model for target DBMS

5 Design physical representation

6 Design security mechanisms

7 Monitor and tune operational system

Physical Database Design

4 Translate global logical data model for target DBMS

4.1 Design base relations for target DBMS

4.2 Design enterprise constraints for target DBMS

4.1 Design base relations

Write the scripts to create tables and constraints, and insert data:

```
CREATE TABLE Property_for_Rent
(
Property_No         VarChar(5),
Street              VarChar(15),
Area                VarChar(15),
City                VarChar(15),
Postcode            VarChar(8),
Type                Char(1),
Rooms               Integer(2),
Rent                Decimal(8,2),
FK_Owner_No         VarChar(5),
FK_Branch_No        VarChar(5),

CONSTRAINT PropPK PRIMARY KEY (Property_No),
CONSTRAINT PropFKOwner FOREIGN KEY (FK_Owner_No) REFERENCES Owner(Owner_No),
CONSTRAINT PropFKBranch FOREIGN KEY (FK_Branch_No) REFERENCES Branch(Branch_No)
)
```

Helpful hint: Make all primary and foreign key fields a standard size - say VARCHAR(5).

4.2 Design Enterprise Constraints

Some Enterprise contraints can be defined on tables (but, in Oracle, not on Views).

```
ALTER TABLE person
  ADD CONSTRAINT check_person_type
  CHECK (
              (
                UPPER(person_type) = 'EMPLOYEE'
                    AND
                employment_date > '15-May-09'
                    AND
                manager_id IN ('M01','M02','M03')
              )
           OR
              (
                UPPER(person_type) = 'CLIENT'
                  AND
                employment_date IS NULL
                  AND
                manager_id IS NULL
              )
         )
```

4.2 Design Enterprise Constraints

For more complex constraints, Oracle allows you to create triggers:

```
CREATE TRIGGER staff_not_handling_too_much
BEFORE
       INSERT OR UPDATE ON property_for_rent
AS
   IF ( (SELECT COUNT(*)
          FROM propertyForRent p
          WHERE p.staffNo = INSERTED.staffNo) >= 10
        )
BEGIN
  PRINT "Staff member already managing 10 properties"
  ROLLBACK TRANSACTION
END
```

Physical Database Design

5 Design physical representation

- 5.1 Analyse transactions
- 5.2 Choose file organisations
- 5.3 Choose secondary indexes
- 5.4 Consider the introduction of controlled redundancy
- 5.5 Estimate disk space requirements

5.1 Analyse Transactions

Objective

To understand the functionality of the transactions that will run on the database and to analyze the important transactions.

Attempt to identify performance criteria, such as:

- **transactions that run frequently and will have a significant impact on performance;**
- **transactions that are critical to the business;**
- **times during the day/week when there will be a high demand made on the database (called the *peak load*).**

Step 5.1 Analyze Transactions

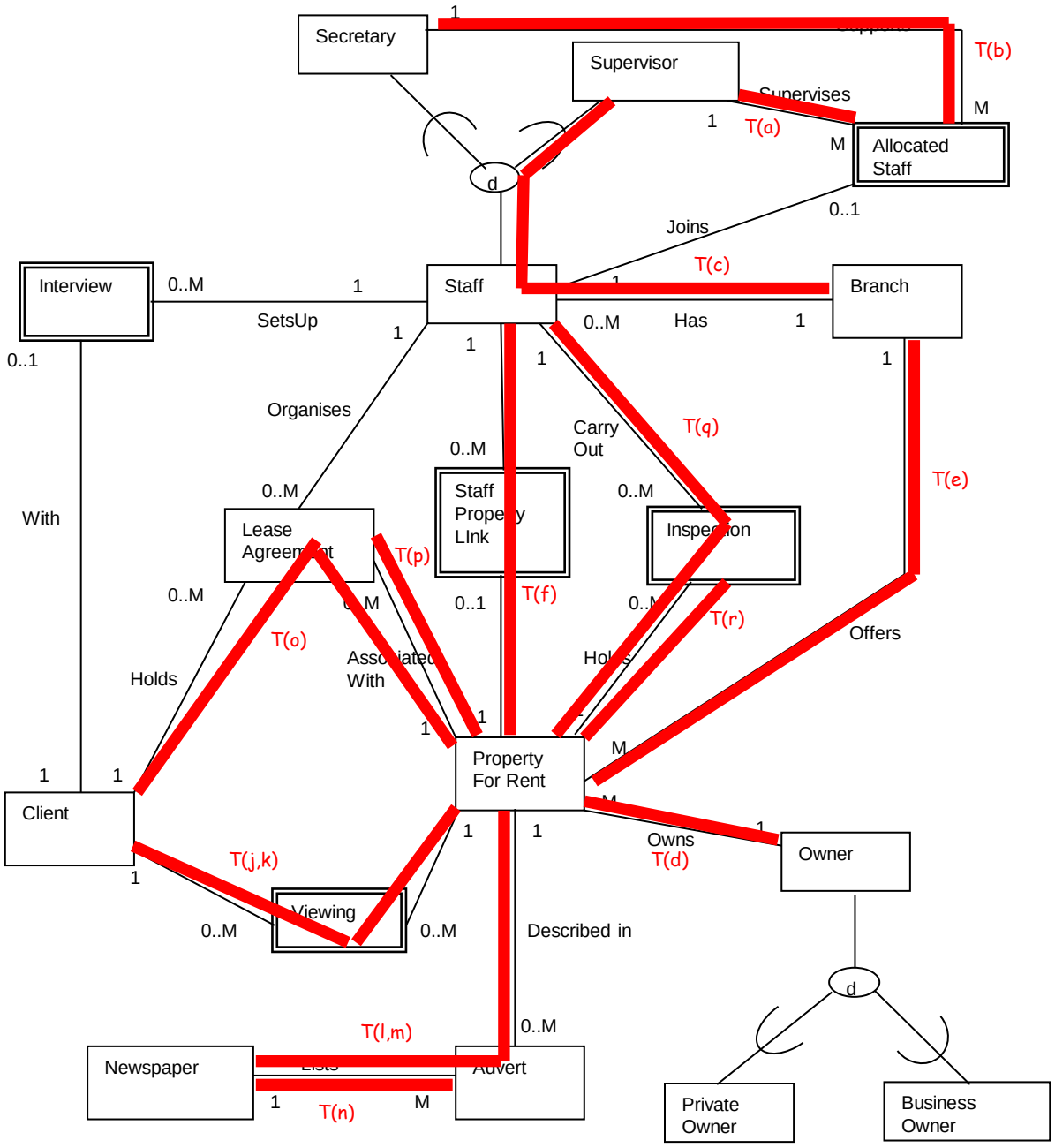
Often not possible to analyze all expected transactions, so investigate most 'important' ones. To help identify which transactions to investigate, can use one or more of the following methods:

- ***Transaction map,***
  showing which tables are most 'important'

- ***Transaction / Relation cross-reference matrix*,**
  showing the tables that each transaction accesses

Having identified the important transactions, they should be analyzed more fully:

- ***Transaction Usage Map***
- ***Transaction Analysis Form***



Transaction Map

Which tables are of most interest?

What isn't shown here?

Cross-reference Matrix

List the transactions:

(A) Enter the details for a new property and the owner

(B) Update/Delete the details of a property.

(C) Identify the total number of staff in each position at branches in Glasgow

(D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent

(E) List the details of properties for rent managed by a named member of staff

(F) Identify the total number of properties assigned to each member of staff at a given branch

Cross-reference Matrix

Table 16.1 Cross-referencing transactions and relations.

Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X					X	X	X						X				X				X		
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

I = Insert; R = Read; U = Update; D = Delete

Determine Frequency Information

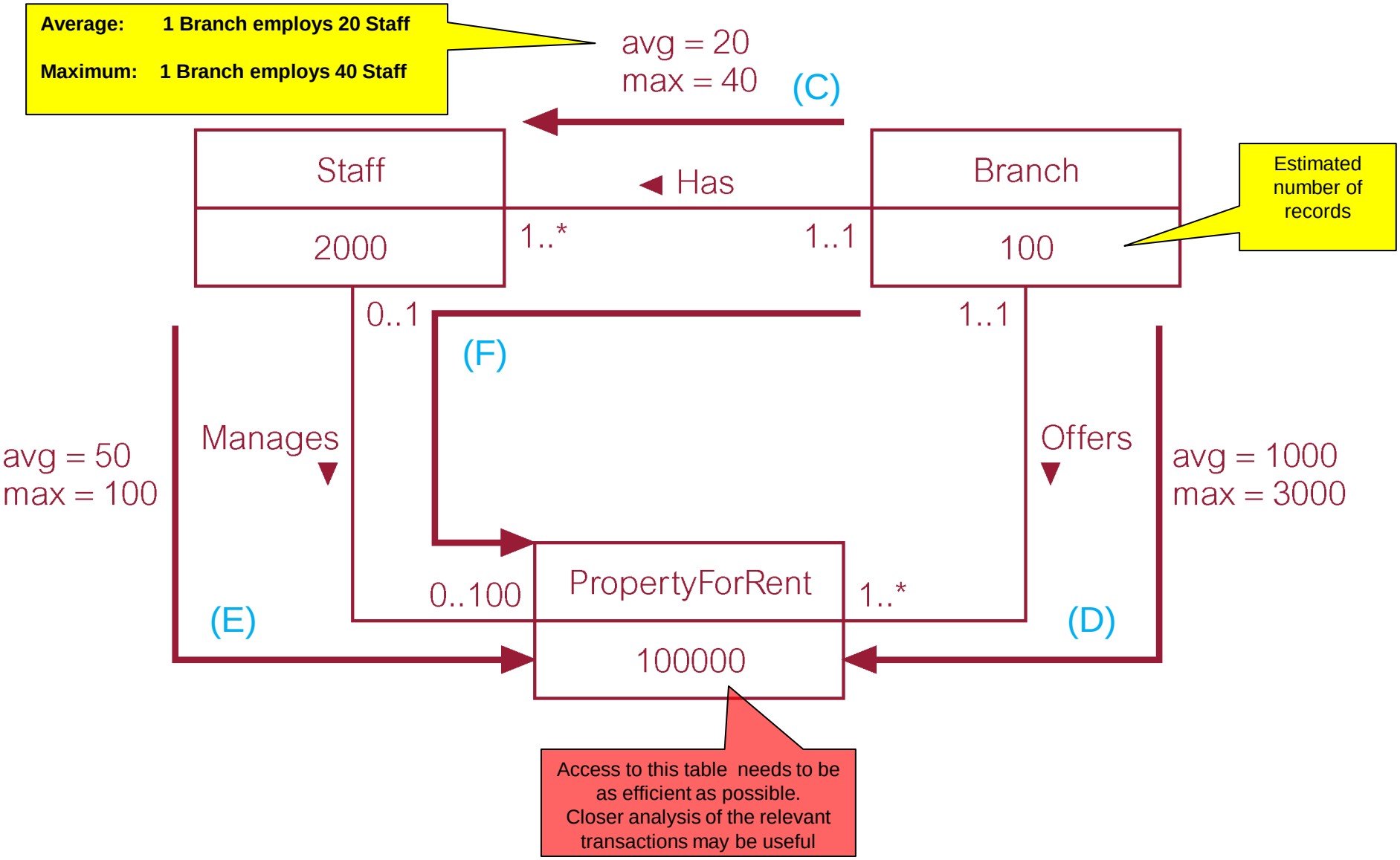
When interviewing users, you need to determine the following for each transaction:

- The average number of times/hour that it will run
- The maximum number of times/hour that it will run
- The day and time it will run (peak load)

Estimates are better than nothing.

This information can be summarised in a transaction usage map and transaction analysis form

Transaction usage map for Transactions C, D, E, F



Example transaction analysis form

Transaction Analysis Form **1-Sept-2001**

Transaction (D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent

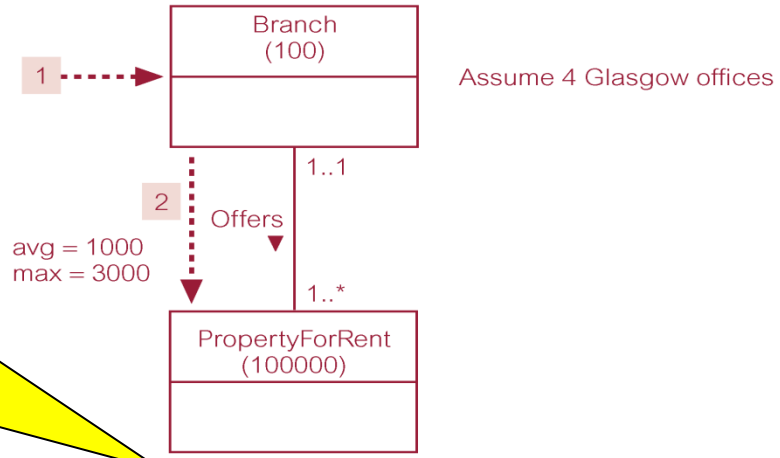
Transaction volume

Average: 50 per hour
Peak: 100 per hour (between 17.00 and 19.00 Monday–Saturday)

```
SELECT propertyNo, p.street, p.postcode, type, rent
FROM Branch b INNER JOIN PropertyForRent p ON
     b.branchNo = p.branchNo
WHERE p.city = 'Glasgow'
ORDER BY rent;
```

Predicate:	**p.city** = 'Glasgow'
Join attributes:	**b.branchNo** = **p.branchNo**
Ordering attribute:	**rent**
Grouping attribute:	none
Built-in functions:	none
Attributes updated:	none

Transaction usage map



Must read each Branch record to determine if they are a Glasgow branch or not.

On average each Branch will match 1000 properties, but could be up to 3000 properties.

So 4 Glasgow branches will match 4000 properties, but could be up to 120000 properties.

Multiply by 50 to get Average per hour

Multiply by 100 to get Peak per hour

Access	Entity	Type of Access	No. of References		
			Per Transaction	Avg Per Hour	Peak Per Hour
1	**Branch** (entry)	R	100	5000	10000
2	**PropertyForRent**	R	4000–12000	200000–600000	400000–1200000
Total References			**4100–12100**	**205000–605000**	**410000–1210000**

So, we may need to make changes to improve performance for certain tables or transactions.

One way is to choose suitable:

- File Organisations
- Secondary Indexes

5.2 Choose File Organisations

Serial (or unordered, or heap)

- Records are written to secondary storage in the order in which they are created.

Hash

- A 'hash' function is applied to each record key, which returns a number used to indicate the position of the record in the file. The hash function must be used for both reading and writing.

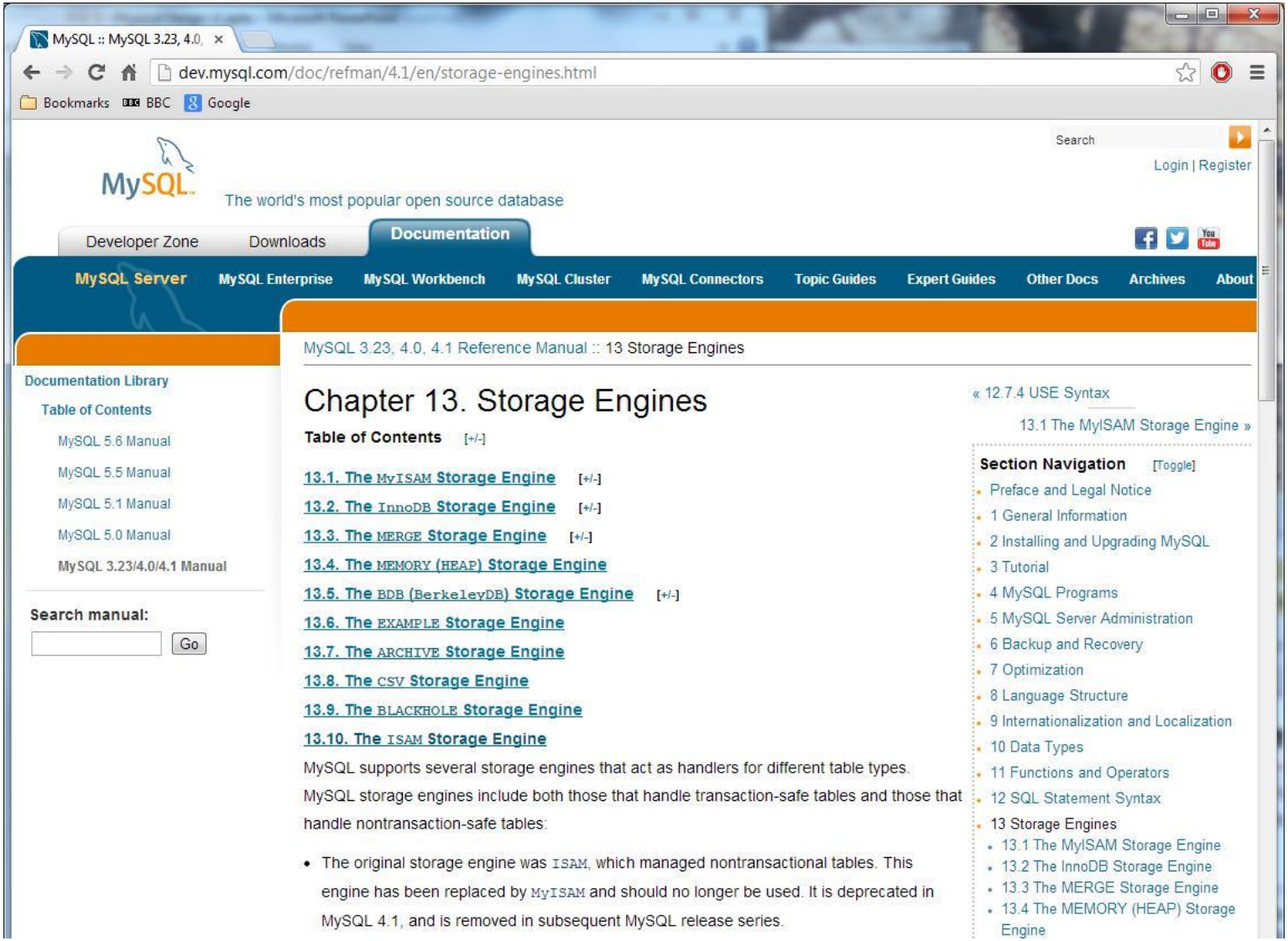
Indexed Sequential Access Method

- The location in secondary storage of some (partial index) or all (full index) records is noted in an index.

B+ Tree

- A self-balancing index which reorganises itself as new items are added.

If you are using MySQL



Oracle has it's own terminology

Oracle provides a number of other structures that can be used make the database more efficient and improve performance.

- Partitioned Tables
- Bitmap Indexes
- Bitmap Join Indexes
- Clusters
- Index Organized Tables
- Externally Organized Tables
- Global Temporary Tables
- Materialized Views
- Partition Views
- Pipeline / Table Functions
- Parameterised Views

See:
http://www.orafaq.com/tuningguide/advanced%20objects.html
(Very Advanced - Not examinable)

5.3 Choose Secondary Indexes

- A DBMS may use different file organisations for its own purposes.
- A DBMS user is generally given little choice of file type.
- A B+ Tree is likely to be used wherever an index is needed.
- Indexes are generated:
  - By default (usually) for fields specified with 'PRIMARY KEY' or 'UNIQUE' constraints in a CREATE TABLE statement.
  - For fields specified in SQL statements such as:

    CREATE INDEX indexname
    ON tablename (colname);

Guidelines for Indexing

Index the primary keys

Index the foreign keys

Index attributes that restrict queries

- (WHERE age > 18)

Index attributes that are sorted

- (ORDER BY salary)

Do not index attributes with few values

- (lots of NULLs)

Do not index every attribute

- (Why?)

Avoid indexing small relations, frequently updated columns, or those with long strings.

In SQL

To create an index

```
CREATE INDEX index_name
ON table_name(field_name );
```

To see all indexes created

```
SELECT INDEX_NAME, INDEX_TYPE , TABLE_OWNER, TABLE_NAME,
   TABLE_TYPE, UNIQUENESS
FROM IND;
```

To delete an index

```
DROP INDEX index_name;
```

5.4 Controlled Redundancy

Also called denormalisation.

Deliberate introducing redundant copies of some data items to speed up query execution time.

Tempting to do this, but remember that it:

- Makes implementation more complex.
- Sacrifices flexibility
- May speed up retrievals, but it slows down updates.

5.5 Estimate Disk Space Requirements

In general, the estimate is (for each table) the size of a row
x the number of rows predicted for the table.

```
CREATE TABLE testtable
(
Id_field        VARCHAR(5),
Name_field      VARCHAR(20),
Age_field       INTEGER(3)
)
```

5 bytes
20 bytes
4 bytes
========
29 bytes x 10,000 rows

= 29,000 bytes

Have you planned for expansion?

Be aware that indexes do have a storage overhead.

Physical Database Design

6 Design security mechanisms

6.1 Design user views

6.2 Design access rules

7 Monitor and tune operational system