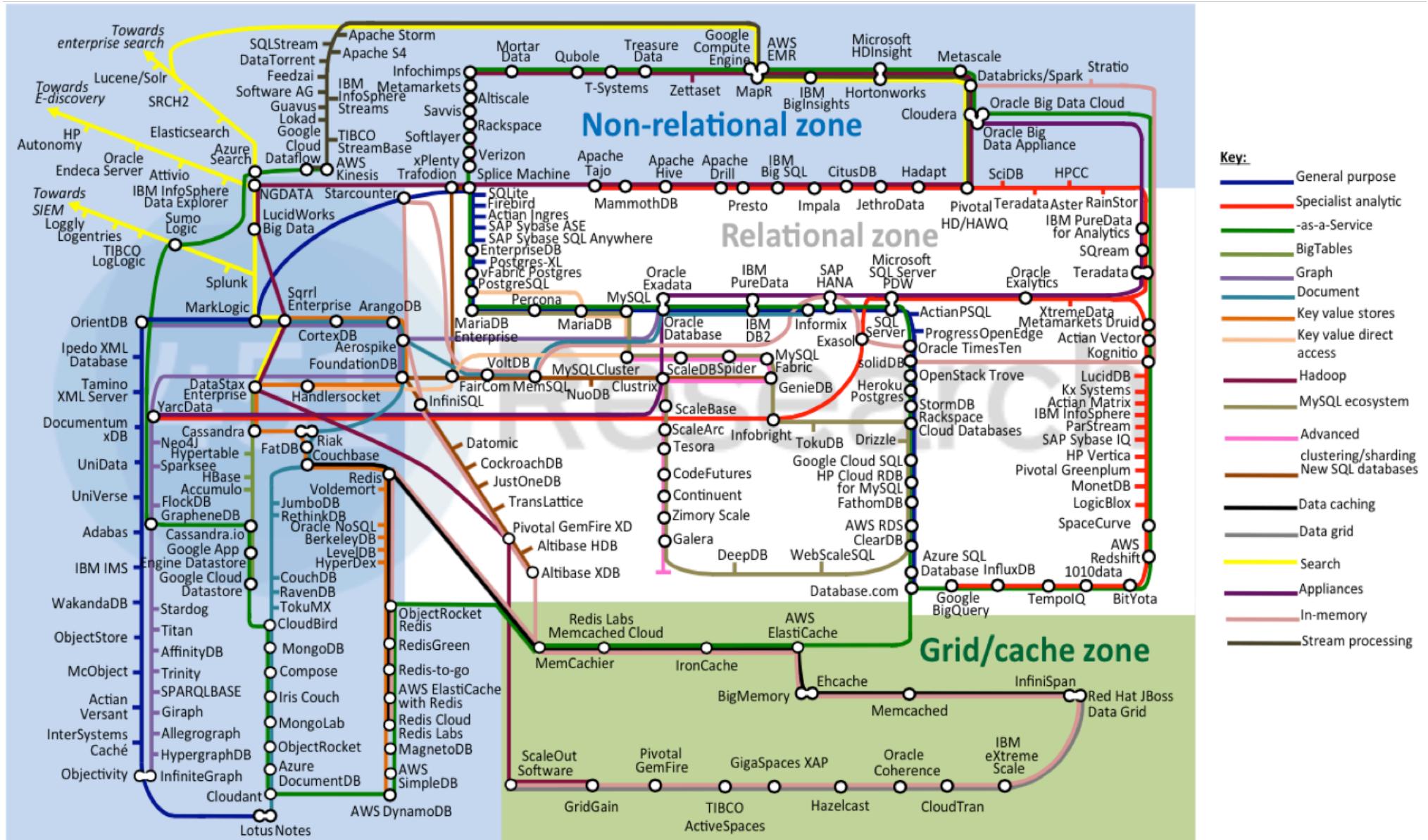
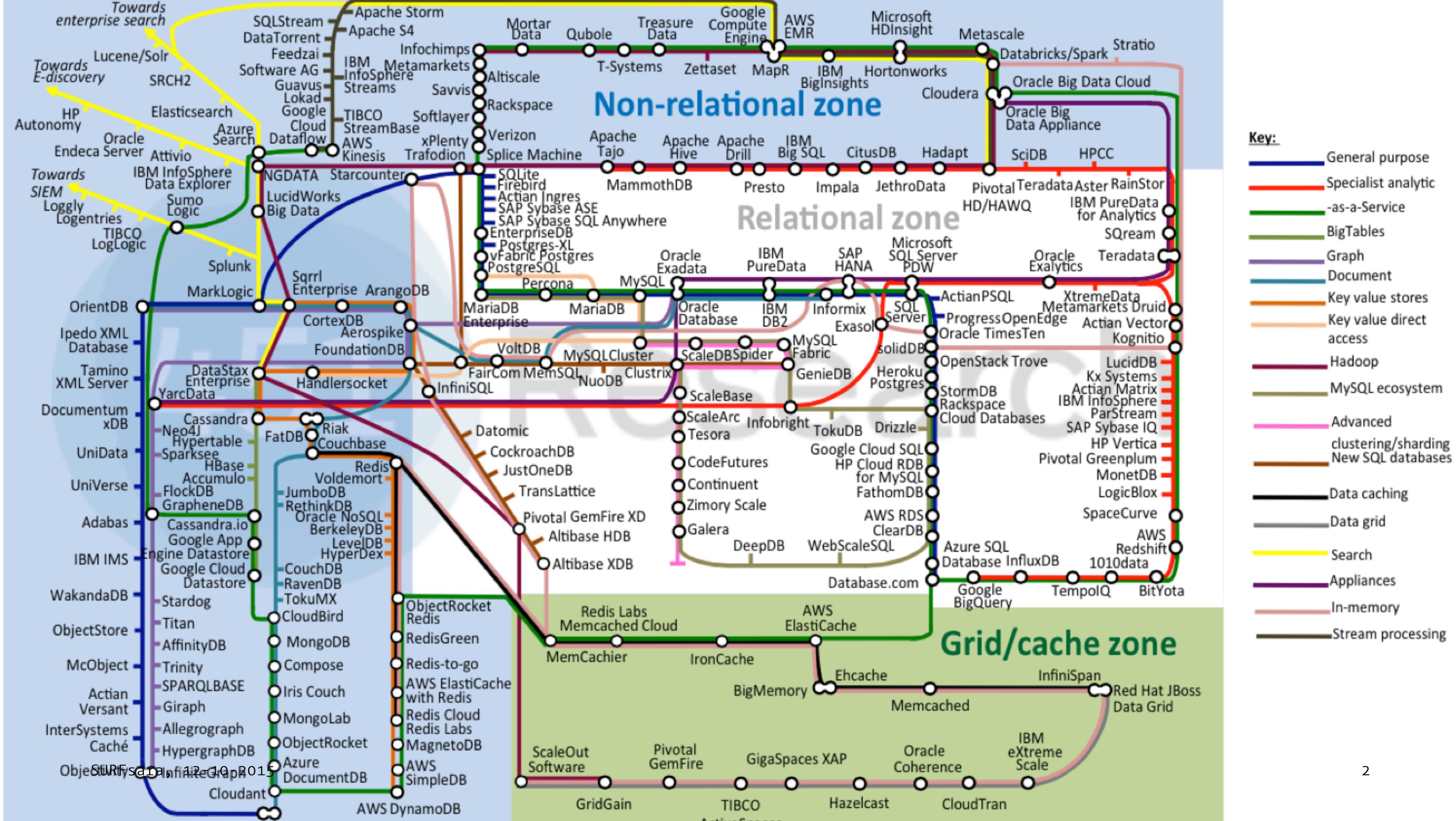


# Databases - Part I

## SNE Master: Essential Skills







### SURF SARA

High-performance computing, data and visualisation for science



### netherlands eScience center

Reinforces and accelerates multi-disciplinary and data-intensive research

## Research workflow optimisation



### SURF NET

Connects users and ICT services and creates new functional possibilities



### SURF MARKET

Favourable conditions for ICT services, software, content

# Today

## Part I: Relational DB's

- What is a database
- Relational concepts
- SQL Basics
- Database design: E/R modelling
- Database design: Normalization
- Concepts: Indexes, Transactions, ACID

# Today

## Part II: NoSQL and NewSQL

- Scaling out and distributed databases
- CAP Theorem
- Data models
- NewSQL

The logo consists of the words "Not Only SQL" in a bold, sans-serif font. The word "Not" is in red, "Only" is in white, and "SQL" is in black. The background is a light pink gradient.

# Today

## Part III: Hands-on PostgreSQL and MongoDB

- Create an E/R model
- Implement this in PostgreSQL
- Denormalize and implement in MongoDB



# What is a DBMS?

Consider a manufacturing firm wishing to computerize stock control:

1. *Production*: {StockNo, Description, Level, Re\_Order\_Level, Unit\_Cost}
2. *Sales*: {Customer\_Name, Address, Invoice\_No, ItemNo, Description, Amount, Item\_Cost, Order\_Cost, Credit\_Limit}
3. *Finance*: {CustomerName, Invoice\_No, Order\_Cost, Payment\_Received, Credit\_Limit}

Straightforward to implement as applications with dedicated files;  
but..

# What is a DBMS?

DBMS: Reduce redundancy and entropy:

1. Ambiguity: ItemNo == StockNO?
2. Inconsistency:
  - update propagation: updates necessary on all three applications.
  - value/type inconsistency

# **What is a DBMS?**

1. Share and integrate data between different applications
2. Support multiple views of the same data
3. Controlled concurrent access to data
4. Ensure security and integrity

# Relational databases

**Two dimensional tables of rows and columns:**

*Tables*: relations

*Rows*: tuples

*Columns*: attributes

*Key attributes*: primary, candidate and foreign

Proposed by Dr E.F. Codd in the 1970s<sup>1</sup>

<sup>1</sup> A Relational Model of Data for Large Shared Databanks. Communications of the ACM, 13, June.

# Relational databases

## Relational algebra:

*RESTRICT*: return tuples based on conditions

*PROJECT*: return tuples with only a set of attributes

*TIMES*: return Cartesian product of two relations

*UNION*: union of two sets of tuples

*MINUS*: subtract two sets of tuples

Derived: JOIN, INTERSECT, DIVIDE

# Structured Query Language: SQL

- SQL is a **declarative language** translated to relational algebra operations
- SQL has been the **de facto industry standard** for relational systems
- SQL is a complete database language for:
  1. **data definition**
  2. **data manipulation**
  3. **data control** (access control to views and tables)

# Data manipulation

**SQL has four data manipulation commands:**

*SELECT*: for retrieving data

*INSERT*: for inserting data

*UPDATE*: for altering data

*DELETE*: for removing data

# Data manipulation: SELECT

```
SELECT column_name, column_name  
FROM table_name;
```

Or:

```
SELECT *  
FROM table_name;
```

Or:

```
SELECT DISTINCT column_name  
FROM table_name
```

# Data manipulation: SELECT ... WHERE

```
SELECT column_name, column_name  
FROM table_name  
WHERE column_name operator value;
```

Operators:

= equals

<> or != not equal

> and < greater than and lesser than

>= and <= greater than or equal and lesser than or equal

BETWEEN between an inclusive range

LIKE search for a pattern

IN to specify multiple values for a column

# Data manipulation: SELECT ... WHERE ... AND ... OR

And:

```
SELECT column_name1,column_name2  
FROM table_name  
WHERE column_name1 operator value  
AND column_name2 operator value;
```

Or:

```
SELECT column_name1,column_name2  
FROM table_name  
WHERE column_name1 operator value  
OR column_name2 operator value;
```

# Data manipulation: Subqueries

```
SELECT column_name1, column_name2  
FROM table_name  
WHERE column_name1 IN  
( SELECT column_name3  
    FROM other_table_name  
   WHERE column_name3 operator value );
```

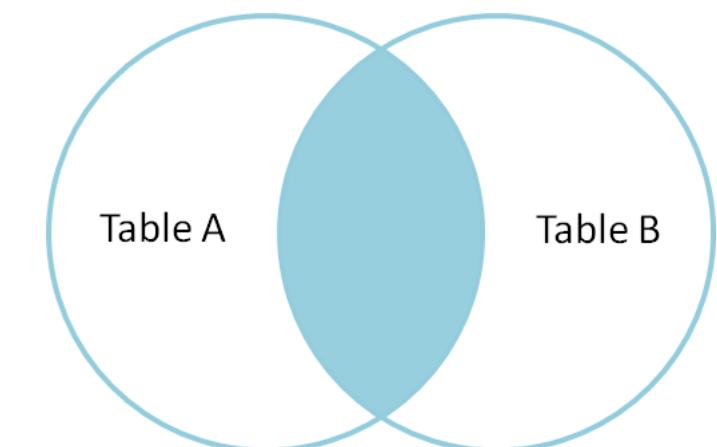
Note: the scope of the subquery

# Data manipulation: Joins

```
SELECT t1.column_name1,t1.column_name3  
FROM table_name1 t1, table_name2 t2  
WHERE t1.column_name1 == t2.column_name1
```

Or:

```
SELECT column_name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column_name=table2.column_name;
```



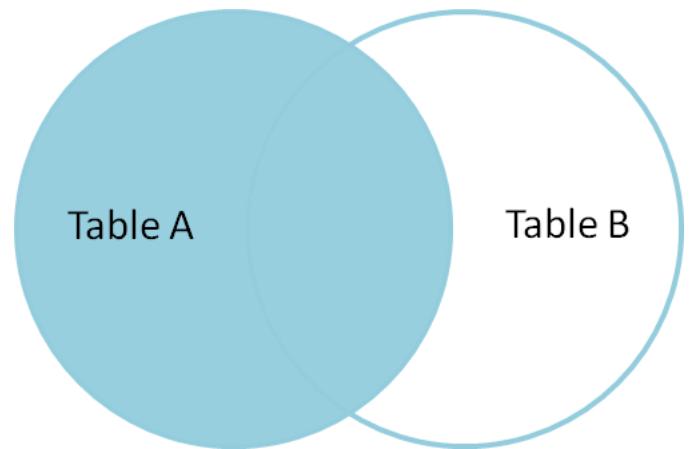
# Data manipulation: More joins

```
SELECT column_name(s)  
FROM table1  
LEFT JOIN table2  
ON table1.column_name=table2.column_name;
```

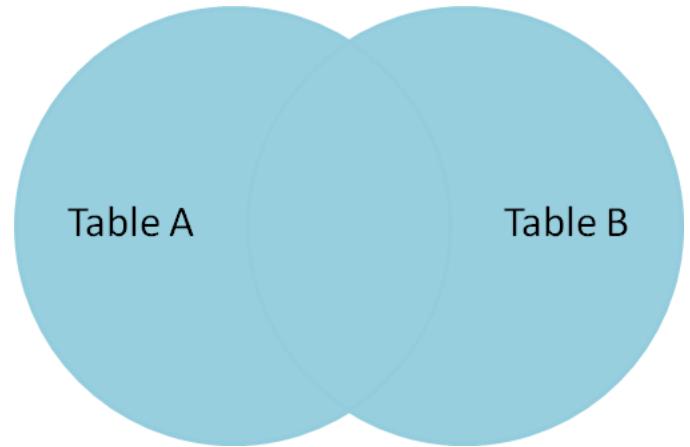
```
SELECT column_name(s)  
FROM table1  
FULL OUTER JOIN table2  
ON table1.column_name=table2.column_name;
```

Note: 'RIGHT/LEFT JOIN' is the same as RIGHT/LEFT OUTER JOIN

# Data manipulation: More joins



LEFT JOIN



FULL OUTER JOIN

# Data definition

**SQL has several data definition commands:**

*CREATE TABLE*: for creating and specifying a table.

*ALTER TABLE*: for altering a table

*DROP TABLE*: for dropping a table

# Data definition CREATE TABLE

```
CREATE TABLE table_name
(
    column_name1 data_type(size) constraint_name,
    column_name2 data_type(size) constraint_name,
    column_name3 data_type(size) constraint_name,
    ...
);
```

## Constraints:

NOT NULL to indicate that a column cannot store NULL values

UNIQUE to indicate that each row should have a unique value

PRIMARY KEY combination of NOT NULL and UNIQUE

FOREIGN KEY ensure referential integrity

CHECK ensures that the value meets a specific condition

DEFAULT specifies a default for a column

Foreign key constraints: (RESTRICT, SET NULL, CASCADE) on UPDATE, DELETE

# Data manipulation: INSERT, UPDATE, DELETE

```
INSERT INTO table_name  
VALUES (value1,value2,value3,...);
```

```
INSERT INTO table_name (column1,column2,column3,...)  
VALUES (value1,value2,value3,...),(valueN,valueNN,valueNNN,...);
```

```
UPDATE table_name  
SET column1=value1,column2=value2,...  
WHERE some_column=some_value;
```

```
DELETE FROM table_name  
WHERE some_column=some_value;
```

# **SQL summary**

- Declaritive language for:
  - data definition
  - data manipulation
  - data control
- ANSI SQL Standard, but dialects per database implementation
- Typically implemented as relational algebra on relational model

# Database design: E/R modelling

Entity/relationship modelling: an approach to semantic modelling originally defined by P. Chen<sup>2</sup>.

Three fundamental components for an E/R model of a database:

1. **Entities**: items in the real world capable of unique existence.
2. **Attributes**: things describing an entity. Includes a key attribute: that part of an entity which gives it a unique identity
3. **Relationships**: a relationship represents the interaction between entities.

Each relationship has a cardinality indicating the number of entities.

<sup>2</sup> The Entity-Relationship Model: Toward a unified view of data. ACM Transactions on Database Systems, 1, March.

# Database design: E/R modelling - Entities

We can identify **Entity types**:

**Strong entity**: Capable of 'independent' existence

**Weak entity**: Can only exist in terms of the relationship in which it participates

We can use **subtyping**.

# Database design: E/R modelling - Attributes

## Types of attributes:

- Simple: atomic values
- Composite: multiples of simple attributes
- Derived: attributes derived from other attributes
- Single-valued attributes: contain a single value per entity
- Multi-valued attributes: may contain multiple values per entity

## Entity sets and keys:

- Super key: a set of attributes that identify an entity in an entity set
- Candidate key: a minimal super key
- Primary key: a candidate key chosen to uniquely identify an entity

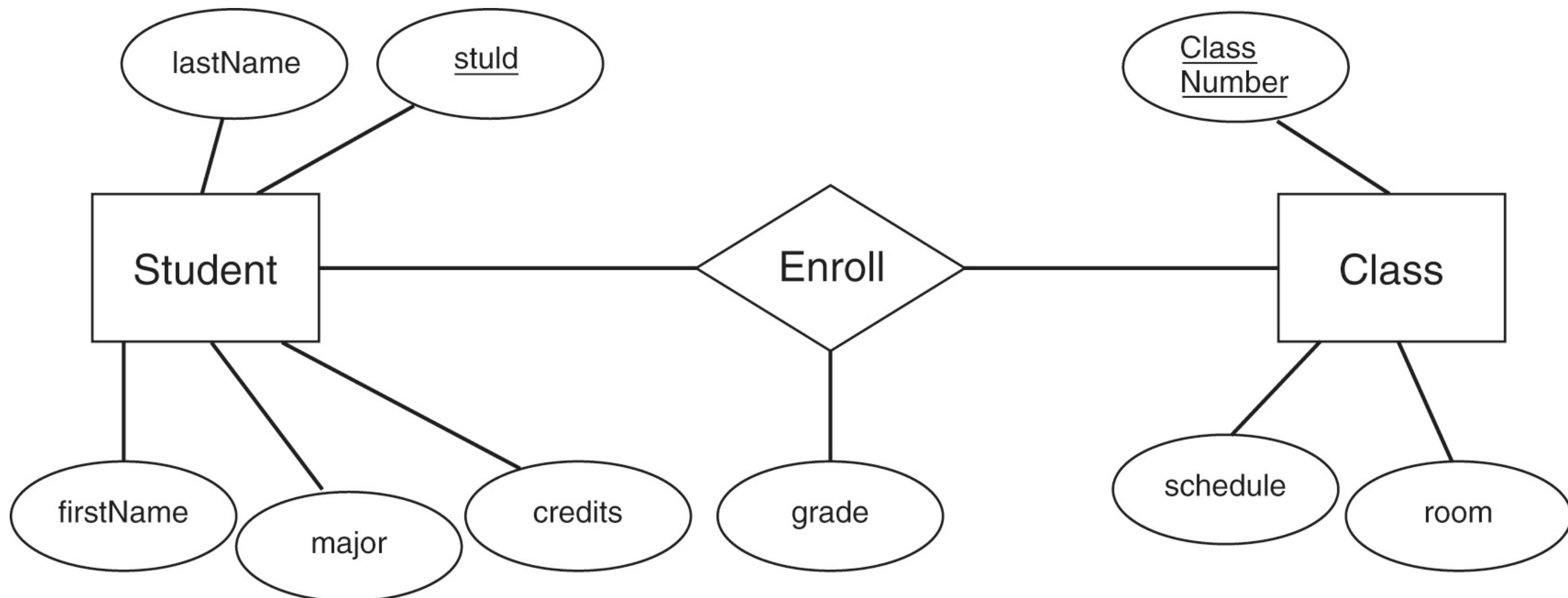
# Database design: E/R modelling - Relationships

Like entities relationships can have **descriptive attributes**

**Cardinality** defines the number of participating entities:

- One-to-one
- One-to-many
- Many-to-one
- Many-to-many

# A simple E/R diagram



# From E/R model to database: 8 steps

1. For each strong entity create a table with columns for each simple attribute. The key attribute becomes the primary key.
2. For each weak entity create a table with columns for each simple attribute and include columns for the primary keys of those entities on which the entity depends (foreign keys).
3. When two entities are in a one-to-many relationship, the entity with the many cardinality must have a foreign key representing this relationship.
4. When two entities are in a one-to-one relationship, a foreign key must be included in one of the two.
5. When two entities are in a many-to-many relationship a table must be created consisting of foreign keys for the two entities.
6. When an entity has a multi-valued attribute, create a table with a column as foreign key to the entity and a column for the multi-valued attribute
7. When more than two entities participate in a relationship then a table must be created consisting of foreign keys to those entities.
8. When a subtyping is defined by attributes create separate tables for each subtype consisting of those attributes which are peculiar to the subtype

# Database design: Normalization

Normalization: ensure efficient data structures

Efficient data structures:

- Are not redundant in order to defeat inconsistency and ambiguity
  - look for update anomalies
  - look for insert anomalies
- Have a minimal use of NULL values
- Prevent loss of information
  - look for deletion anomalies

Common problem: represent entities as (collection) of attributes

# Database design: Normalization

S_id	S_Name	S_Address	Subject_opted
401	Adam	Noida	Bio
402	Alex	Panipat	Maths
403	Stuart	Jammu	Maths
404	Adam	Noida	Physics

# Database design: Normal forms

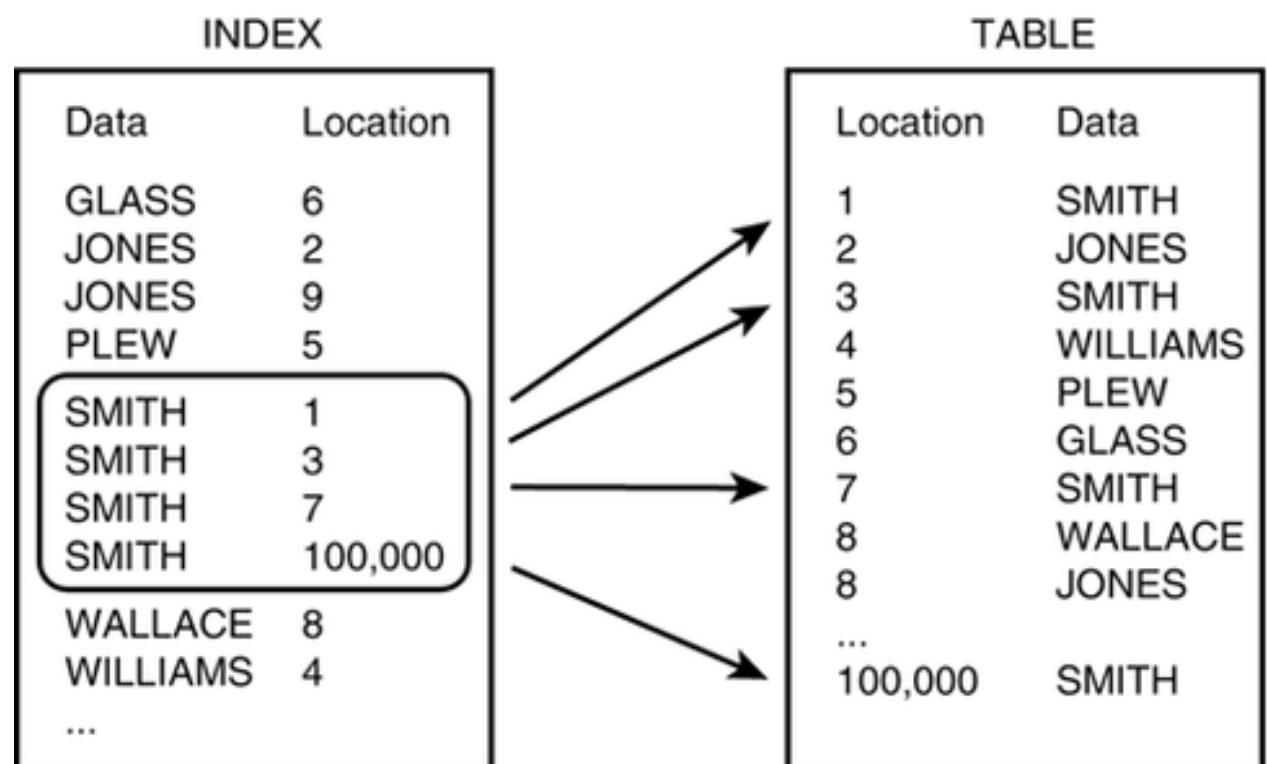
A good database schema is in BCNF.

Normal forms:

- eliminate duplicate columns from the same table
- create separate tables from data which applies to multiple rows
- remove columns that are not dependent on the primary key (e.g. derived attributes)

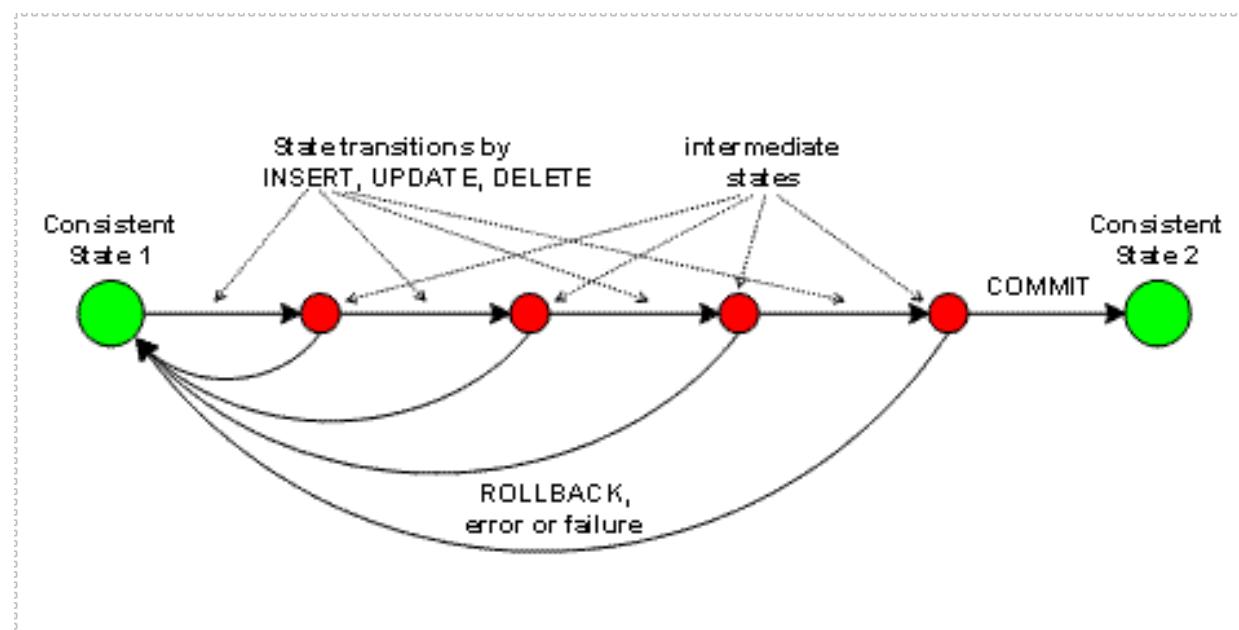
# Concepts: Indexes

Extra data structure to **speed up** access to a set of records.



# Concepts: Transactions

A transaction is a **logical unit of work**: Group commands into a single transaction with rollback and commit options.



## **Concepts: ACID**

**Atomicity:** each transaction is all or nothing.

**Consistency:** each transaction will bring the database from one valid state to the next.

**Isolation:** the concurrent (parallel) execution of transactions has the same result as executing them serially.

**Durability:** when a transaction has been completed, it will remain so

# Scaling relational databases

Column name	Type
id	integer
user_id	integer
url	varchar(255)
pageviews	bigint

Figure 1.1 Relational schema for simple analytics application

# Scaling relational databases

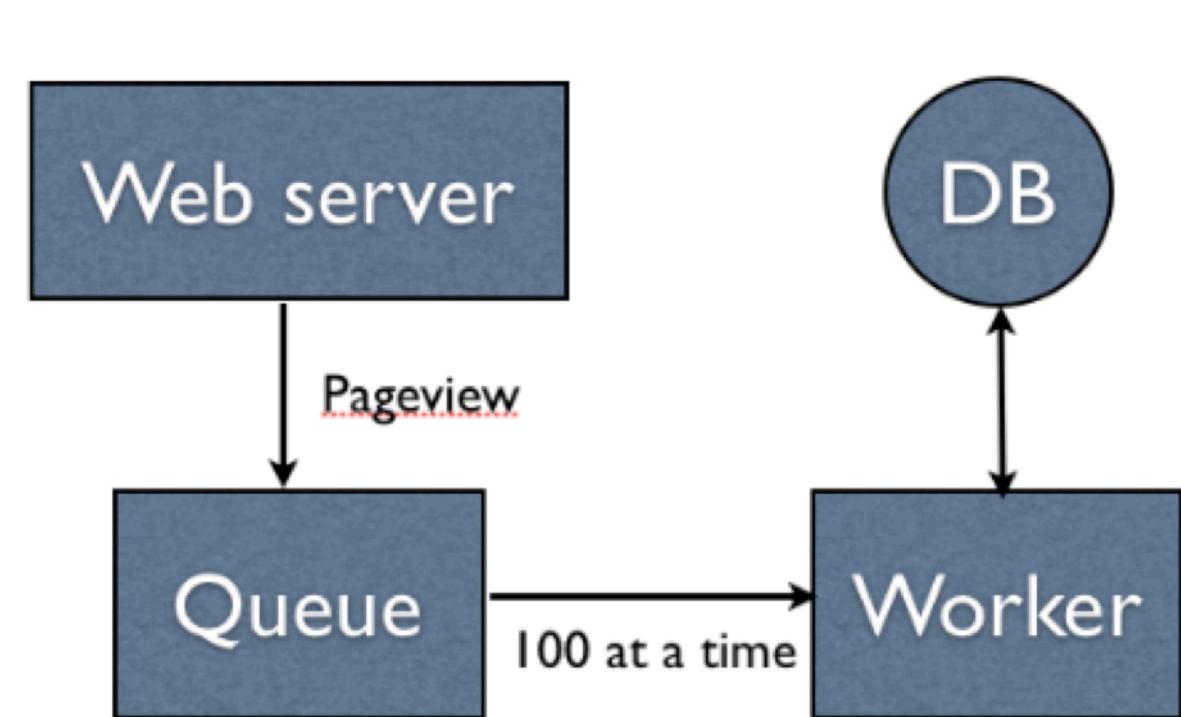
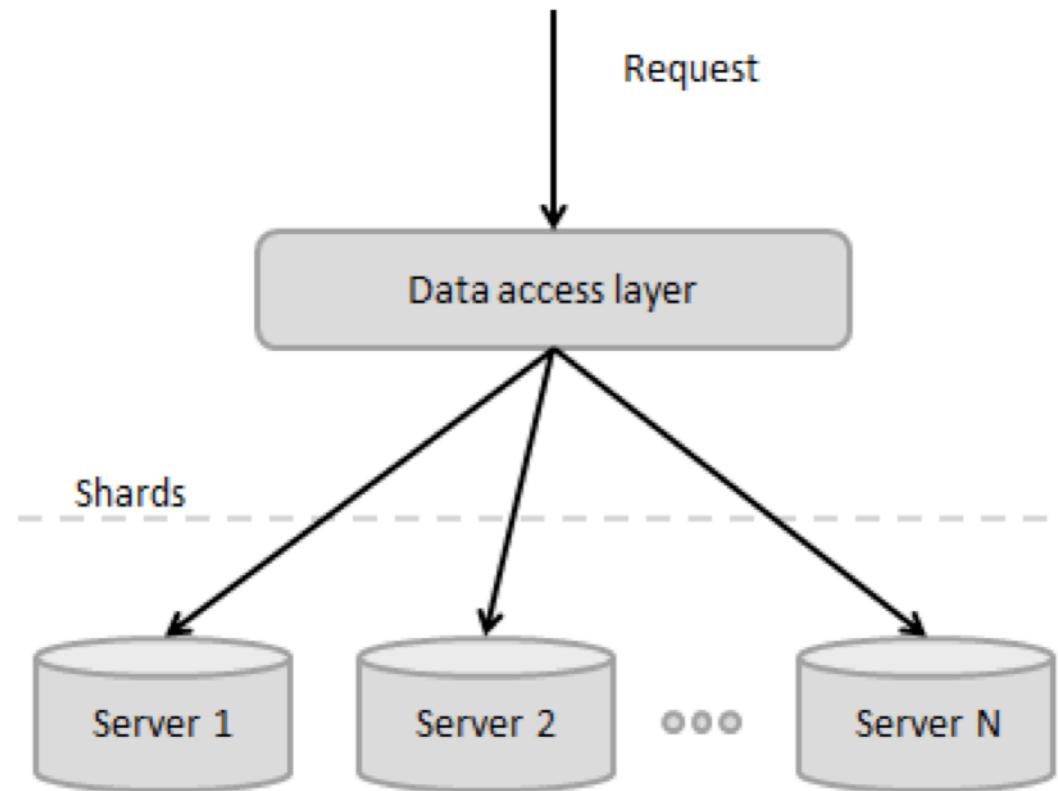


Figure 1.2 Batching updates with queue and worker

# Scaling relational databases



# Questions?

