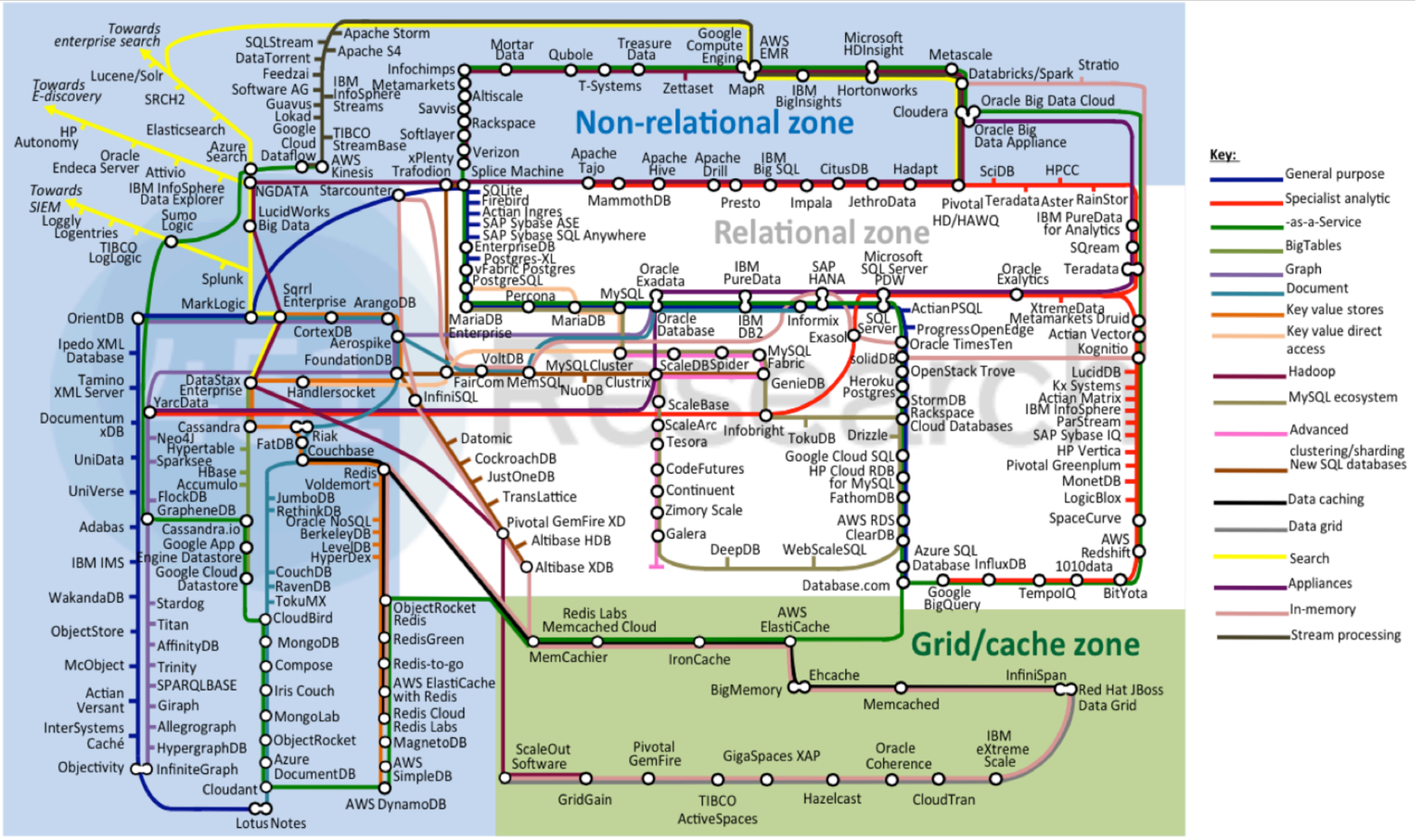
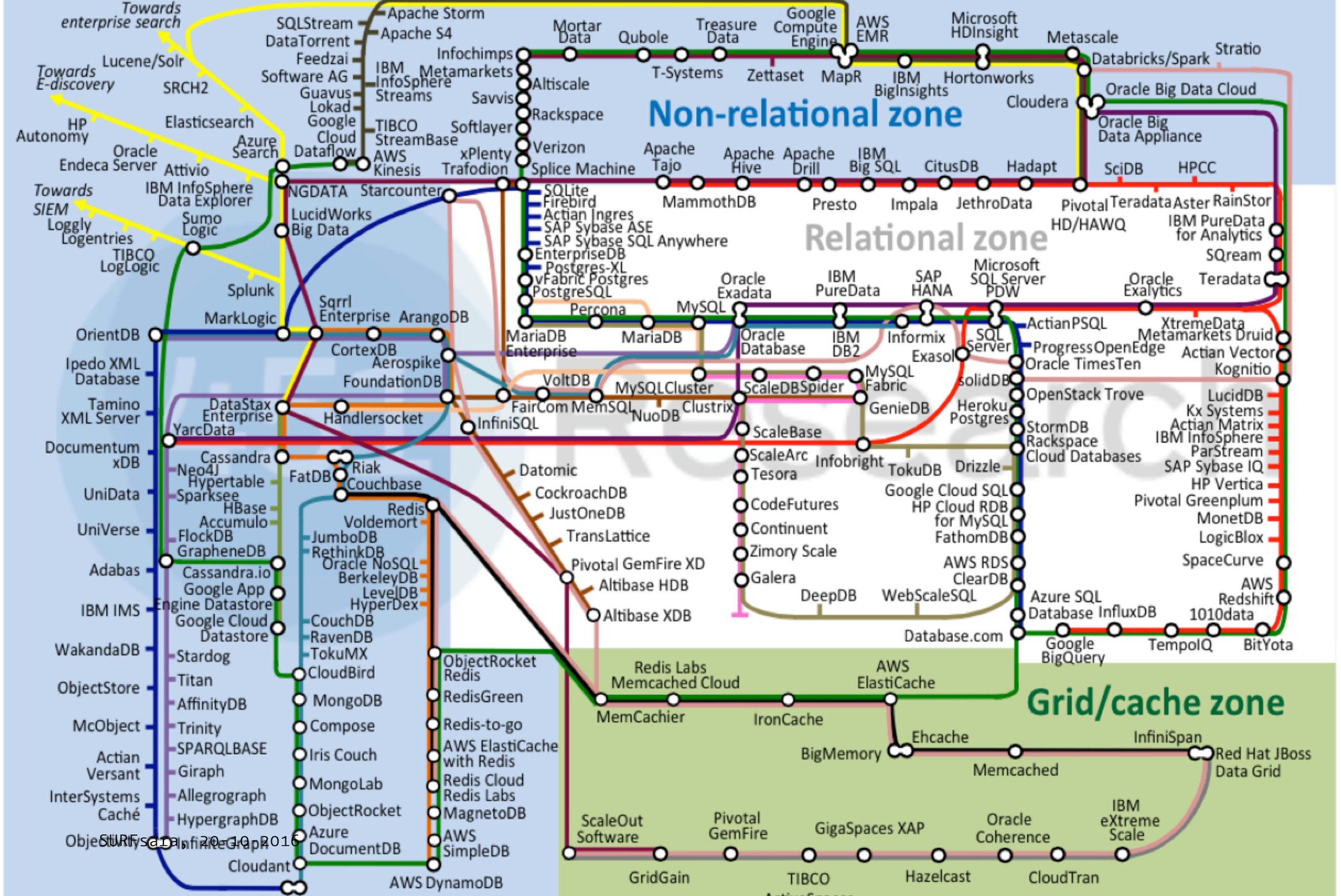
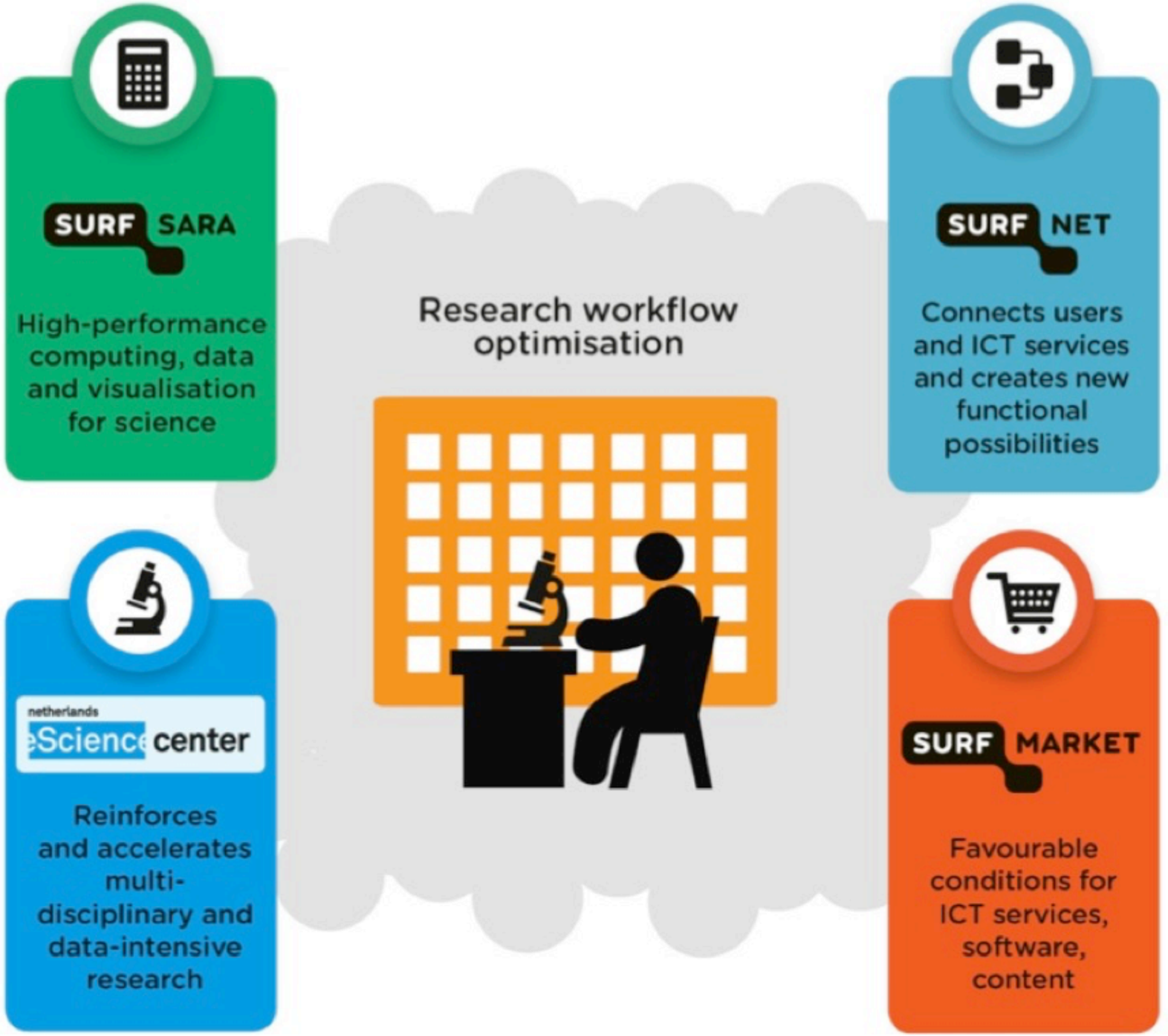
Databases - Part I

SNE Master: Essential Skills




Towards enterprise search
Towards E-discovery
Towards SIEM
Non-relational zone
Relational zone
Grid/cache zone
Apache Storm
SQLStream
Apache S4
DataTorrent
Feedzai
Software AG
Guavus
Lokad
Google Cloud Dataflow
IBM InfoSphere Streams
TIBCO StreamBase
AWS Kinesis
Infochimps
IBM Metamarkets
Altiscale
Savvis
Rackspace
Softlayer
Verizon
xPlenty
Trafodion
Splice Machine
Mortar Data
Qubole
Treasure Data
Google Compute Engine
AWS EMR
Microsoft HDInsight
Metascale
Databricks/Spark
Stratio
T-Systems
Zettaset
MapR
IBM BigInsights
Hortonworks
Cloudera
Oracle Big Data Cloud
Oracle Big Data Appliance
Lucene/Solr
SRCH2
Elasticsearch
Azure Search
HP Autonomy
Oracle Endeca Server
Attivio
IBM InfoSphere Data Explorer
Sumo Logic
Loggly
Logentries
TIBCO LogLogic
Splunk
NGDATA
Starcounter
LucidWorks Big Data
Sqrrl Enterprise
MarkLogic
ArangoDB
Apache Tajo
Apache Hive
Apache Drill
IBM Big SQL
CitusDB
Hadapt
SciDB
HPCC
MammothDB
Presto
Impala
JethroData
Pivotal HD/HAWQ
Teradata Aster
RainStor
IBM PureData for Analytics
SQream
Teradata
SQLite
Firebird
Actian Ingres
SAP Sybase ASE
SAP Sybase SQL Anywhere
EnterpriseDB
Postgres-XL
vFabric Postgres
PostgreSQL
Percona
MariaDB
MariaDB Enterprise
MySQL
Oracle Exadata
IBM PureData
SAP HANA
Microsoft SQL Server PDW
Oracle Exalytics
Oracle Database
IBM DB2
Informix
Exasol
SQL Server
ActianPSQL
ProgressOpenEdge
Oracle TimesTen
solidDB
XtremeData
Metamarkets Druid
Actian Vector
Kognitio
OrientDB
Ipedo XML Database
Tamino XML Server
Documentum xDB
UniData
UniVerse
Adabas
IBM IMS
WakandaDB
ObjectStore
McObject
Actian Versant
InterSystems Caché
ObjectivityInfiniteGraph
CortexDB
Aerospike
FoundationDB
DataStax Enterprise
YarcData
Handlersocket
InfiniSQL
VoltDB
FairCom MemSQL
NuoDB
MySQLCluster
Clustrix
ScaleDB
Spider
MySQL Fabric
GenieDB
ScaleBase
ScaleArc
Tesora
CodeFutures
Continuent
Zimory Scale
Galera
Infobright
TokuDB
Drizzle
Heroku Postgres
StormDB
Rackspace Cloud Databases
OpenStack Trove
LucidDB
Kx Systems
Actian Matrix
IBM InfoSphere
ParStream
SAP Sybase IQ
HP Vertica
Pivotal Greenplum
MonetDB
LogicBlox
SpaceCurve
AWS Redshift
1010data
Cassandra
Neo4J
Hypertable
Sparksee
HBase Accumulo
FlockDB
GrapheneDB
Cassandra.io
Google App Engine Datastore
Google Cloud Datastore
Stardog
Titan
AffinityDB
Trinity
SPARQLBASE
Giraph
Allegrograph
HypergraphDB
InfiniteGraph
FatDB
Riak
Couchbase
Redis
Voldemort
JumboDB
RethinkDB
Oracle NoSQL
BerkeleyDB
LevelDB
HyperDex
CouchDB
RavenDB
TokuMX
CloudBird
MongoDB
Compose
Iris Couch
MongoLab
ObjectRocket
Azure DocumentDB
Cloudant
AWS DynamoDB
ObjectRocket Redis
RedisGreen
Redis-to-go
AWS ElastiCache with Redis
Redis Cloud
Redis Labs
MagnetoDB
AWS SimpleDB
Datomic
CockroachDB
JustOneDB
TransLattice
Pivotal GemFire XD
Altibase HDB
Altibase XDB
DeepDB
WebScaleSQL
AWS RDS
ClearDB
FathomDB
Google Cloud SQL
HP Cloud RDB for MySQL
Azure SQL Database
InfluxDB
Database.com
Google BigQuery
TempoIQ
BitYota
Redis Labs Memcached Cloud
AWS ElastiCache
MemCachier
IronCache
Ehcache
BigMemory
Memcached
InfiniSpan
Red Hat JBoss Data Grid
ScaleOut Software
Pivotal GemFire
GigaSpaces XAP
Oracle Coherence
IBM eXtreme Scale
GridGain
TIBCO
Hazelcast
CloudTran
Key:
General purpose
Specialist analytic
-as-a-Service
BigTables
Graph
Document
Key value stores
Key value direct access
Hadoop
MySQL ecosystem
Advanced clustering/sharding
New SQL databases
Data caching
Data grid
Search
Appliances
In-memory
Stream processing


SURF SARA
High-performance computing, data and visualisation for science
Research workflow optimisation
SURF NET
Connects users and ICT services and creates new functional possibilities
netherlands eScience center
Reinforces and accelerates multi-disciplinary and data-intensive research
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: ER modelling
- Database design: Normalization
- Concepts: Indexes, Transactions, ACID

Today

Part II: NoSQL and NewSQL

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

Today

Part III: Hands-on PostgreSQL and MongoDB

Find the exercises here:
https://github.com/sara-nl/sne-es-db/tree/master/hands-on

- Create an ER 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[1]

[1] A Relational Model of Data for Large Shared Databanks. Communications of the ACM, 13, Issue 6, June 1970

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 statements:

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 value1
AND column_name2 operator value2;
```

Or:

```
SELECT column_name1, column_name2
FROM table_name
WHERE column_name1 operator value1
OR column_name2 operator value2;
```

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

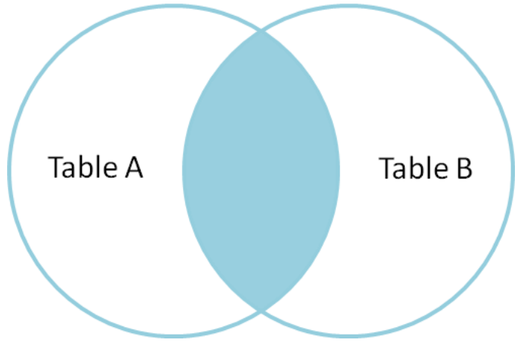
```
SELECT t1.column_name1, t2.column_name3
FROM table_name1 t1, table_name2 t2
WHERE t1.column_name1 = t2.column_name1
```

Or:

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

Note: JOIN is the same as

INNER JOIN



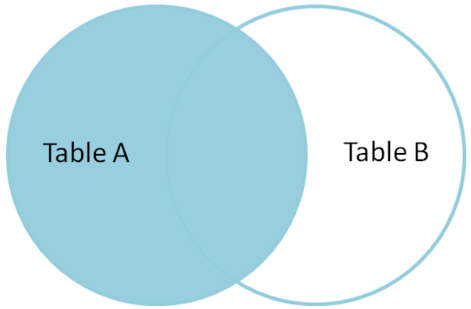
Data manipulation: More joins

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

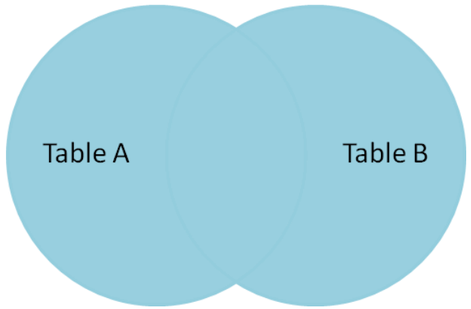
```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

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

Data manipulation: More joins



LEFT JOIN



FULL OUTER JOIN

See: Coding Horror: A visual explanation of SQL joins

Data definition

SQL has several data definition statements:

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

- Declarative 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: ER modelling

Entity/relationship modelling: an approach to semantic modelling originally defined by P. Chen[2]

Three fundamental components for an ER 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

[2] The Entity-Relationship Model: Toward a unified view of data. ACM Transactions on Database Systems, 1, Issue 1, March 1976

Database design: ER 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: ER 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

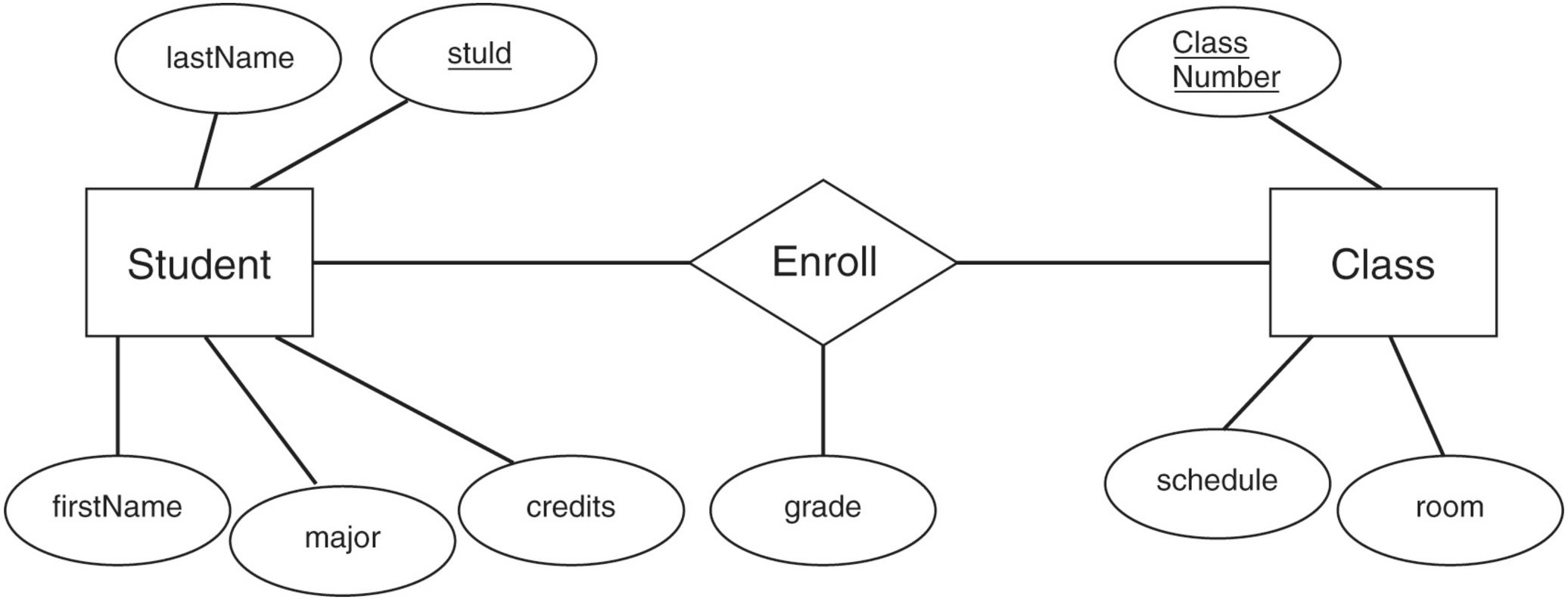
Database design: ER modelling - Relationships

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 ER diagram



From ER model to database: 8 steps (1/2)

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

From ER model to database: 8 steps (2/2)

1. When two entities are in a many-to-many relationship a table must be created consisting of foreign keys for the two entities
2. 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
3. When more than two entities participate in a relationship then a table must be created consisting of foreign keys to those entities
4. 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:

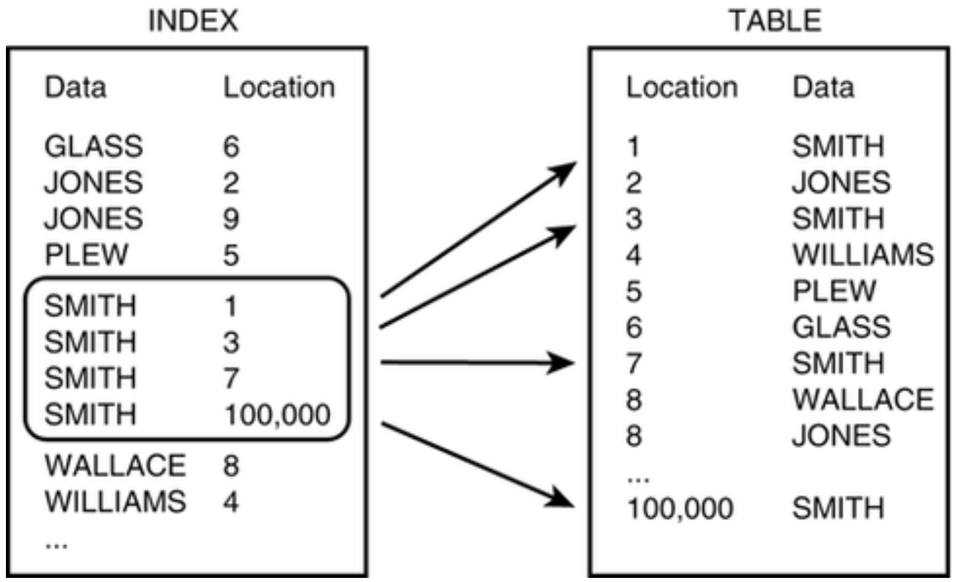
— eliminate duplicate columns from the same table

— create separate tables from data which applies to multiple rows

— remove columns that are not dependent on the primary key (e.g. derived attributes)

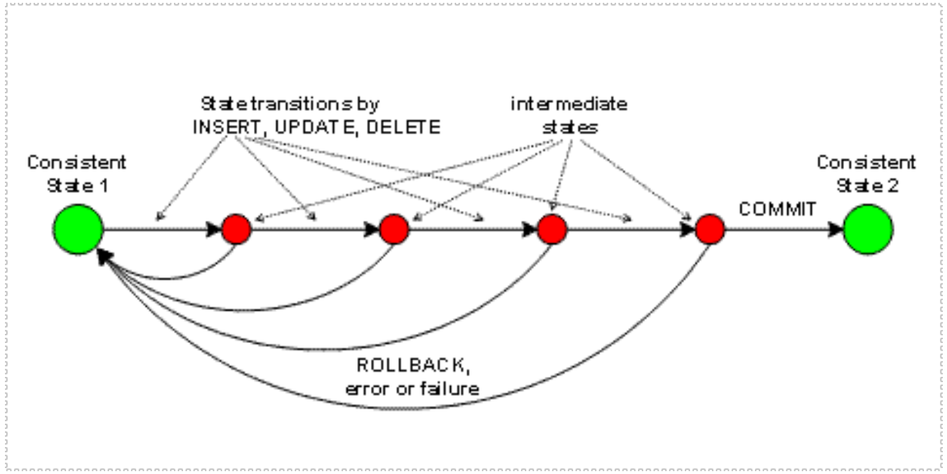
Concepts: Indexes

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: Concurrency and Locking

Shared (read) and exclusive (update) locks:

- Shared locks may be placed on shared locks
- An exclusive lock may not be placed on shared locks
- When an exclusive lock is placed no other lock may be placed

Concepts: Two-phase locking

Protocol: do not place lock when a lock has been released earlier (i.e. do not release locks before all locks have been placed)

1. TA places S lock on O1
2. TB places S lock on O2
3. TA places S lock on O1
4. TB request X lock on O1 -> denied; TB waits
5. TA places X lock on O1
6. TA releases all locks
7. TB can now place X lock on O1
8. TB places S lock on O2
9. TB releases all locks

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

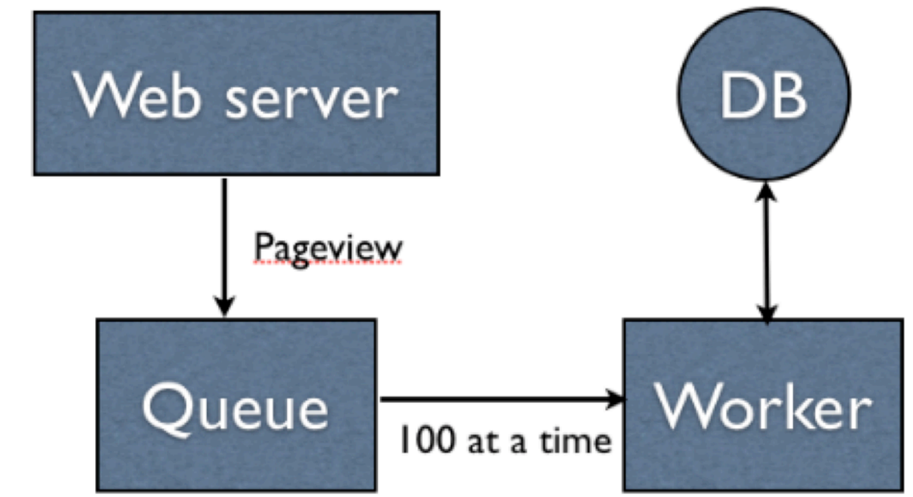
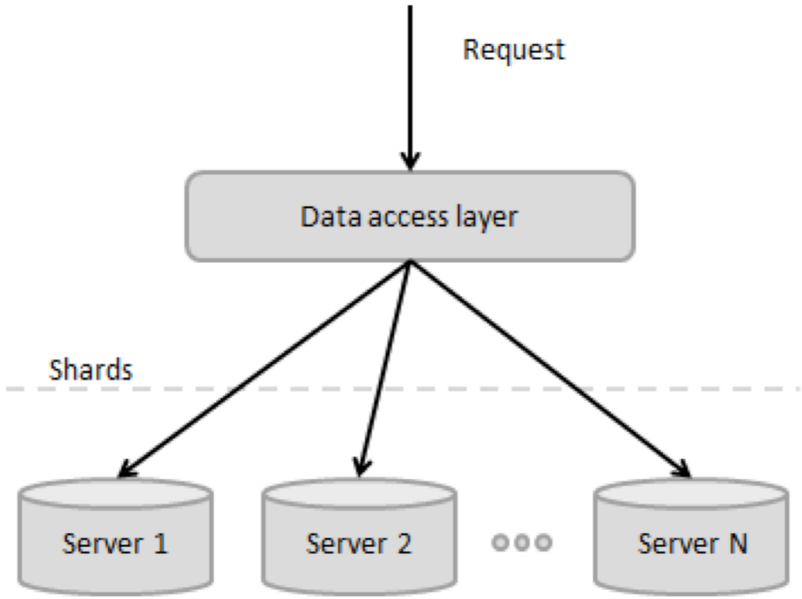
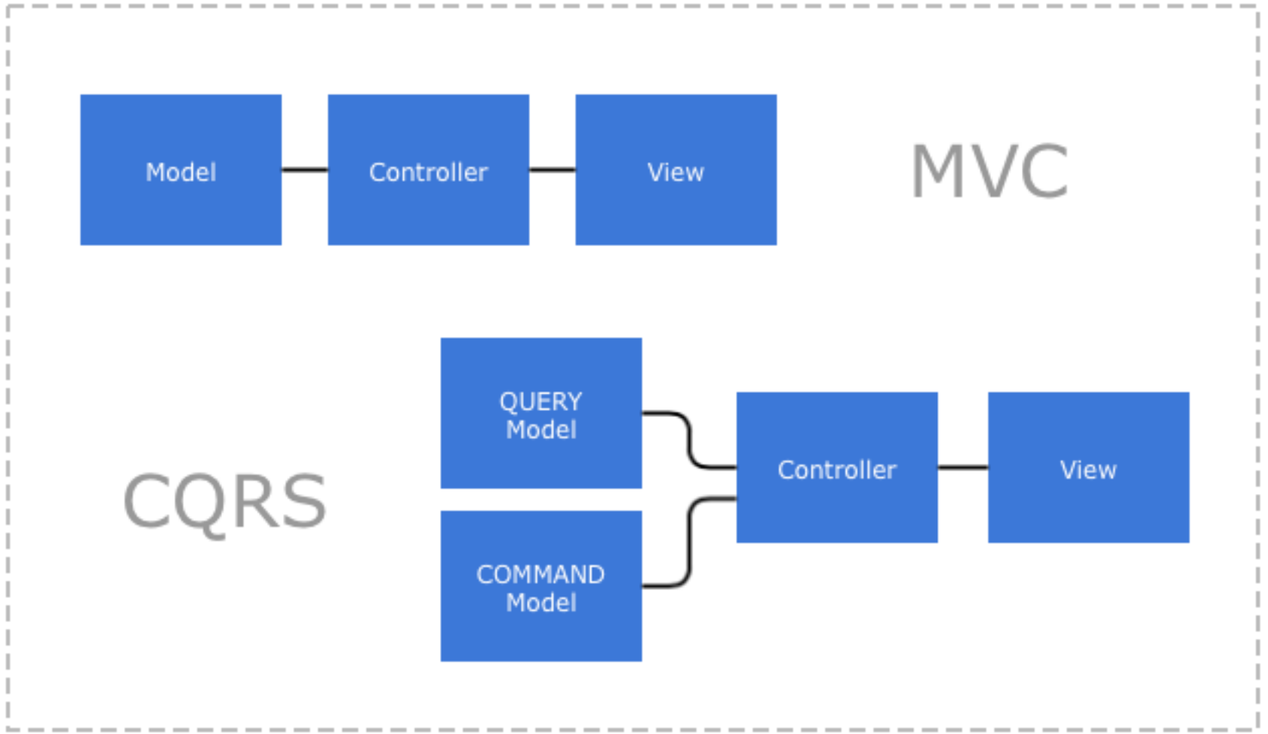


Figure 1.2 Batching updates with queue and worker

Scaling relational databases



Scaling relational databases



Questions?