COMP421 Crib Sheet Francis Piché

Transactions

-A sequence of reads **r(x)** and writes **w(x)**

-**Atomic** (all or nothing)

 -Keep *backup* of state before transaction

 -Restore to this point in case of failure

-**Consistency** (preserve consistency)

-**Isolation** must have serial equivalent

-**Durability** must be permanent/fault tolerant

Transactions can be **aborted**

 -*Global recovery:*

  -Transactions committed before crash are in effect.

  -Transactions aborted before crash are reversed

  -Transactions active at time of crash are reversed

  -Assume disk doesn't crash

Logs: are kept because holding back writes is insufficient. Limited number of buffer frames means transactions cannot all be atomic.

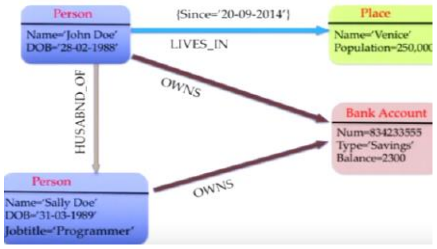
Graph Databases: (FLEXIBLE)

-Each vertex has own properties

-Properties are K-V pair

-Can easily be extended. No pre-planning required

-Edges can have properties too (are directional)



Cypher:

TRAVERSALS:

(e)-[*]->(n)	// All the way (outgoing edges)
(e)-[*..5]->(n)	// Up to a depth of 5 edges (outgoing)
(e)-[*3..]->(n)	// 3 or more edges (outgoing)
(e)-[*3..5]->(n)	// 3 to 5 edges (outgoing)
(e)<-[*3..5]-(n)	// 3 to 5 edges (incoming)
(e)-[*3..5]-(n)	// 3 to 5 edges (incoming or outgoing)

General	`DISTINCT`
Math	`+, -, *, /, %, ^`
Comparison	`=, <>, <, >, <=, >=, IS NULL, IS NOT NULL`
String comparison	`STARTS WITH, ENDS WITH, CONTAINS`
Boolean	`AND, OR, XOR, NOT`
String operators	`+ (Concatenation), =~ (regex matching)`

SELECT * FROM Employees	MATCH(e:Employee) RETURN e;
SELECT email FROM Employees	MATCH(e:Employee) RETURN e.email;
…. ORDER BY email	… RETURN e ORDER BY e.email;
…WHERE name = 'Janet'	MATCH(e:Empl {ename: 'Janet'} RETURN e;
… WHERE deptid IS NULL	… WHERE NOT (e)-[:WORKS_IN]-() …
	… WHERE e.job IS NULL … (treat non-exist property as NULL)
INSERT INTO …	CREATE (e:Empl {name: 'Jane'}-[:WORKS_IN]->(d:Depart {dname:'PR'});
New edge b/w existing nodes:	MATCH (n1: Empl {eid: 101}), (n2: …) CREATE (n1)-[:MANAGES]->(n2);
DELETE FROM … (Must delete relationships)	MATCH(e: …)-[r:WORKS_IN]->(d:Dep..) DELETE e, r, d;
Delete all edges connected to this node	DETACH DELETE e;

Can combine conditions by comma separating:

How to find a list of people who manages someone who mentors more than one employee ?

```
MATCH (b:Employee)-[:MANAGES]->(m:Employee)
          ,(m)-[:MENTORS]->(e1:Employee)
          , (m)-[:MENTORS]->(e2:Employee)
WHERE e1 <> e2
RETURN DISTINCT b
```

EACH EDGE IS TRAVERSED ONLY ONCE TO AVOID CYCLES