1)

Primary Key: A restrição PRIMARY KEY identifica exclusivamente cada registro em uma tabela.

As chaves primárias devem conter valores UNIQUE e não podem conter valores NULL.

Uma tabela pode ter apenas UMA chave primária; e na tabela, essa chave primária pode consistir em uma ou várias colunas (campos).

Unique: A restrição UNIQUE garante que todos os valores em uma coluna sejam diferentes.

As restrições UNIQUE e PRIMARY KEY fornecem uma garantia de exclusividade para uma coluna ou conjunto de colunas.

Uma restrição PRIMARY KEY possui automaticamente uma restrição UNIQUE.

No entanto, você pode ter muitas restrições UNIQUE por tabela, mas apenas uma restrição PRIMARY KEY por tabela.

Auto incremento: O incremento automático permite que um número exclusivo seja gerado automaticamente quando um novo registro é inserido em uma tabela.

Geralmente, esse é o campo da chave primária que gostaríamos de ser criado automaticamente sempre que um novo registro fosse inserido.

2) Resumo INNODB

Todas as tabelas INNODB são representadas em disco por um arquivo de extensão *.frm* no diretório da banco de dados e armazena dados e índices em *tablespaces*. Tais *tablespaces*são padronizadas e são localizadas no diretório *data*ou *datadir*. Também por padrão, todas as tabelas INNODB de um servidor compartilham um mesmo *tablespace*. O INNODB ainda nos permite configurar que cada tabela tenha sua própria *tablespace*, evitando problemas com arquivos grandes;

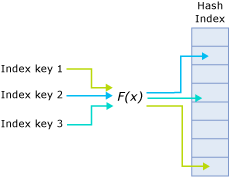
* INNODB dá suporte à transações e é totalmente adequada às propriedades ACID. O multi-versionamento é utilizado para garantir o isolamento de uma transação em relação à outra;
* INNODB provê *auto\_recovery* após um *crash* ou queda no servidor onde o MySQL roda;
* INNODB tem multi-versionamento e bloqueio em nível de linha, ou seja, caso uma transação A esteja atualizando uma determinada linha de uma tabela, esta será bloqueada até que haja um COMMIT ou ROLLBACK. O multi-versionamento provê à transação uma visão própria do estado do banco de dados, combinado com o bloqueio em nível de linha, provê o mínimo possível de contenções;
* Prevalece a *“good query concurrency”* ou boa concorrência entre consultas. Isso é estabelecido através do multi-versionamento, que diminui o tempo de contenção, mas, sendo possível o acontecimento de *Deadlocks*, que são detectados pelo SGBD e a menos transação envolvida é finalizada;
* INNODB tem suporte a integridade referencial, ou seja, tem suporte a criação de FOREIGN KEY, incluindo cascateamentos em DELETE e UPDATE. As propriedades de integridade referencial suportados pelo INNODB são: { RESTRICT | CASCADE | SET NULL | NO ACTION }
* O INNODB trabalha basicamente utilizando dois recursos em disco, os tablespaces que armazenam dados e índices e um conjunto de arquivos de log que registram todas as transações que manipulam dados em tabelas INNODB no servidor MySQL;
* O tablespace do INNODB é portável e pode ser copiado diretamente para outro servidor, juntamente com os arquivos *.frm*. Alguns pontos devem ser analisados com cuidado para que não haja problemas com o backup físico do banco de dados. Considere fazer um backup lógico com o mysqldump a copiar arquivos;

Controle de transações com COMMIT, ROLLBACK, ROLLBACK TO SAVEPOINT e SAVEPOINT;

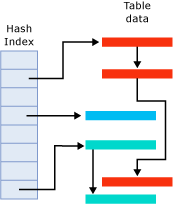
3) Índice de Hash

Os índices são usados como pontos de entrada para tabelas com otimização de memória. A leitura das linhas de uma tabela requer um índice para localizar os dados na memória.

Um índice de hash consiste em uma coleção de buckets organizados em uma matriz. Uma função de hash mapeia chaves de índice para buckets correspondentes no índice de hash. A figura a seguir mostra três chaves de índice mapeadas para três buckets diferentes no índice de hash. Para fins ilustrativos, o nome da função de hash é f(x).



A estrutura de índice de hash na memória consiste em uma matriz de ponteiros de memória. Cada bucket é mapeado para um deslocamento nesta matriz. Cada bucket na matriz aponta para a primeira linha desse bucket de hash. Cada linha no bucket aponta para a próxima linha, resultando em uma cadeia de linhas para cada bucket de hash, conforme ilustrado na figura a seguir.



A figura tem três buckets com linhas. O segundo bucket na parte superior contém as três linhas vermelhas. O quarto bucket contém uma única linha azul. O bucket inferior contém as duas linhas verdes. Essas versões podem ser diferentes e estarem na mesma linha.

4)

Um índice é uma estrutura em disco associada a uma tabela ou view, que agiliza a recuperação das linhas. Um índice contém chaves criadas de uma ou mais colunas e essas chaves são armazenadas em uma estrutura (árvore B) que habilita o SQL Server a localizar a linha ou as linhas associadas aos valores de chave de forma rápida e eficaz.

Índices desnecessários podem ser um problam?

Basicamente é porque cada novo índice exige mais tempo para atualizar o banco em qualquer alteração que afete estes índices. Já vi gente sugerir fazer índices para tudo. Com apenas 6 ou 7 campos, para todas as combinações são necessários dezenas para centenas de índices e todos precisariam ser atualizados mesmo que altere apenas um campo. O tempo gasto pode afetar a escalabilidade do banco.

Em alguns casos a consulta pode ser prejudicada também. O espaço ocupado prejudicará o cache dos dados/índices que realmente são importantes. Além disto acessar primeiro o índice para depois acessar o dado tem um custo e ele pode ser maior que acessar os dados diretamente em certos padrões. E nem sempre o otimizador do sistema detecta corretamente se ele deve fazer o acesso direto.

Da mesma forma as recomendações automáticas dos sistemas SGDBs para criar índices nem sempre são corretas.

Além disto cada novo índice é um recurso a mais para dar manutenção. Viola o YAGNI, Inclusive dificulta *upgrades* no modelo de uma base de dados no sistema em produção.